Model Checking Trust-based Multi-Agent Systems

Nagat Drawel

A Thesis

In

The Concordia Institute

for

Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Information Systems Engineering) at

Concordia University

Montréal, Québec, Canada

December, 2019

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Nagat Drawel

Entitled: Model Checking Trust-based Multi-Agent Systems

and submitted in partial fulfillment of the requirements for the degree of

Doctor Of Philosophy (Information Systems Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. S. Samuel Li

_____ External Examiner
Dr. Nadia Tawbi

_____ External to Program
Dr. Emad Shihab

_____ Examiner
Dr. Rachida Dssouli

_____ Examiner
Dr. Roch Glitho

_____ Thesis Supervisor
Dr. Jamal Bentahar

Approved by _____
Dr. Mohammad Mannan, Graduate Program Director

February 10, 2020

_____
Dr. Amir Asif, Dean
Gina Cody School of Engineering & Computer Science

ABSTRACT

Model Checking Trust-based Multi-Agent Systems

Nagat Drawel, Ph.D.

Concordia University, 2019

Trust has been the focus of many research projects, both theoretical and practical, in the recent years, particularly in domains where open multi-agent technologies are applied (e.g., Internet-based markets, Information retrieval, etc.). The importance of trust in such domains arises mainly because it provides a social control that regulates the relationships and interactions among agents. Despite the growing number of various multi-agent applications, they still encounter many challenges in their formal modeling and the verification of agents' behaviors. Many formalisms and approaches that facilitate the specifications of trust in Multi-Agent Systems (MASs) can be found in the literature. However, most of these approaches focus on the cognitive side of trust where the trusting entity is normally capable of exhibiting properties about beliefs, desires, and intentions. Hence, the trust is considered as a belief of an agent (the truster) involving ability and willingness of the trustee to perform some actions for the truster. Nevertheless, in open MASs, entities can join and leave the interactions at any time. This means MASs will actually provide no guarantee about the behavior of their agents, which makes the capability of reasoning about trust and checking the existence of untrusted computations highly desired.

This thesis aims to address the problem of modeling and verifying at design time trust in MASs by (1) considering a cognitive-independent view of trust where trust ingredients are seen from a non-epistemic angle, (2) introducing a logical language named Trust Computation Tree Logic (TCTL), which extends CTL with preconditional, conditional, and

graded trust operators along with a set of reasoning postulates in order to explore its capabilities, (3) proposing a new accessibility relation which is needed to define the semantics of the trust modal operators. This accessibility relation is defined so that it captures the intuition of trust while being easily computable, (4) investigating the most intuitive and efficient algorithm for computing the trust set by developing, implementing, and experimenting different model checking techniques in order to compare between them in terms of memory consumption, efficiency, and scalability with regard to the number of considered agents, (5) evaluating the performance of the model checking techniques by analyzing the time and space complexity.

The approach has been applied to different application domains to evaluate its computational performance and scalability. The obtained results reveal the effectiveness of the proposed approach, making it a promising methodology in practice.

ACKNOWLEDGEMENTS

I would like to express my gratitude to Allah Almighty for granting me the health, ability, and patience to complete this thesis.

Dear professor Jamal Bentaher, I would like to thank you for encouraging me to pursue a Ph.D. and for your trust on me, support, patience, motivation, and for catching many of my flawed ideas with your valuable and insightful discussions that helped me accomplish these five years adventure.

Moreover, I would like to express my deepest appreciation to my Ph.D. committee members Dr. Rachida Dssouli, Dr. Emad Shihab, and Dr. Roch Glitho for giving me the honor by being in my Ph.D. committee and for their time, guidance, valuable comments and suggestions.

My gratitude also goes to all my colleagues in the research laboratory at Concordia University especially Mona Taghavi, Dr. Omar Abdul Wahab, Gaith Rjoub, Ahmad Bataineh, Amine Laarej, Faryed Eltayesh, and Narges Baharloo for unforgettable moments, and for providing me a warm and friendly atmosphere to work in.

This research would not have been possible without the financial support of the Ministry of Higher Education - Libya. This support was very important for me to alleviate the financial burdens and focus on my research duties. Furthermore, I would like to thank Concordia University for all research facilities that have been provided to me to carry out this work.

Finally, I would like to thank my parents, my brothers and sisters for supporting me spiritually throughout writing this thesis and my life in general, and most importantly, I wish to thank my loving and supportive husband, Miftah, and my five wonderful children, Taha, Anas, Sanad, Shahd, and my little girl Ahad, who provide unending inspiration.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ARCTL	Action Restricted Computation Tree Logic
BDD	Binary Decision Diagram
BDI	Beliefs, Desires, and Intentions
CCTL	Counting CTL logic
CTL	Computation Tree Logic
CTLC	Computation Tree Logic of Commitment
CTLK	Computation Tree Logic of Knowledge
ISPL	Interpreted Systems Programming Language
LTL	Linear Temporal Logic
MAS	Multi-Agent System
MCMAS	Model Checker for Multi-Agent Systems
NuSMV	New Symbolic Model Verifier
PCTL	Probabilistic Computation Tree Logic
PRISM	PRobabilistIc Symbolic Model checker
PSPASE	Polynomial Space
SMV	Symbolic Model Verifier
SPIN	Simple Promela INterpreter

Chapter 1

Introduction

In this chapter, we introduce the context of our research and motivations. Then, we identify the methodology, research questions, and contributions of our work. Finally, we conclude this chapter by providing the thesis organization.

1.1 Context of Research

1.1.1 Agents and Multi-Agent Systems

Agents are autonomous entities that have reactive, pro-active, social and rational properties. Reactive property means that an agent is capable of responding to external changes in its environment. Pro-active property refers to its ability to behave with respect to its goals. Social property is an agent's capability to interact and communicate with other agents, and rationality property is an agent's ability to act consistently with its goals [109]. A Multi-Agent System (MAS) composed of multiple agents, which interact in dynamic and uncertain environments in order to achieve their goals [110]. Agents may be heterogeneous, which means that they may have different preferences and behaviours, and they may be independently

developed by different programmers. The ability of agents to communicate and interact with one another is one of their essential properties. Interaction among autonomous and heterogeneous agents is the key aspect for: 1) solving complex problems that an individual agent cannot handle alone, and 2) building effective MASs. These appealing features made MASs successfully adopted in a large number of critical applications such as commercial, industrial, governmental and healthcare systems [12, 111, 107, 58]. Nonetheless, this adoption raises a number of challenges related to their present and future behaviors. The fact that agents are autonomous and have to interact with each other within unreliable environments makes the concept of trust of particular importance for regulating their interactions.

1.1.2 Trust in Multi-Agent Systems

Trust is regarded as being one of the key aspects behind the success and growth of applications based on MASs. Trust has been an essential research topic in several disciplines for many years. Each of these disciplines gives different definitions for trust [63, 65, 46, 105]. For instance, in the field of distributed computing, trust is used mainly to regulate the relationship between service providers and customers [105, 1]. In social science disciplines, trust is seen as a relationship among individuals in social settings [46] (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 C&F) [16], 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. C&F 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 [48]. Indeed, by emphasizing the agent's goal, C&F rely on the internal structures of agents for the fulfillment of their own goals. Since these systems involve autonomous entities that keep their structure private, it is hard to verify if the agents' goals are achieved. To cope with this limitation, we take in this research a new approach towards social perspectives of trust where the trust parties do not have intuition for cognitive goals. 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 numerical approach and the logical approach. The numerical direction treats trust as a function that is calculated based on multiple opinions through feedback, user ratings, or agent monitoring [106, 17, 82, 94]. 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 [77]. Following this work, a number of computational models have been proposed in the MASs literature (see for instance [90]). Nevertheless, in dynamic MASs where agents may join for a short period of time before leaving the interaction, it might be impossible to collect sufficient information to evaluate the trustworthiness of partners. As a consequence, the well established trust relationships is not guaranteed due to misleading trust results. 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 [3, 68], while others extended standard logics of action and belief, or enriched temporal logics with a new modality for trust [27, 48, 98].

1.1.3 Verification of Trust in MASs

Generally, verifying that a system complies with its design requirements is very challenging, especially in multi-agent systems. The existence of many autonomous entities in such systems makes the verification highly difficult due to the increase in their complexity and heterogeneity. The main challenge that faces MASs is how to ensure the reliability of the trust relationships in the presence of misbehaving entities. Such entities not only create an exception for other agents, but also may obstruct their proper work [55]. The fact that such agents are autonomous and have to interact with each other within unreliable environments makes reasoning about trust and checking the existence of untrusted computations highly desired.

Technically, 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 [4, 6, 8]. 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 [9, 71, 108]. 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 [66]. 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 among 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 thesis, 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. Recently, formal evaluation using model checking techniques has been proved to be a highly effective tool [20, 21].

1.2 Motivations

In this research, we are primarily concerned with the issues of reasoning about and verifying trust in the context of MASs using the model checking approach, which has not been deeply investigated yet for trust systems. In the literature, many logical formalism approaches of trust in MASs can be found. However, very few approaches addressed trust from a high level abstraction viewpoint [98]. Modal logic approaches provide powerful mechanisms that can be effectively used for trust reasoning. Such approaches yield a formal semantics to reason about trust properties in various applications such as security protocols, information sources, and e-markets. For instance, in [24, 48, 75], the authors proposed several logical frameworks for the concept of trust. Trust in such logics is mostly expressed as a combination of different modalities based on the logic of action and time [47] and the BDI logic [22]. In [69], a modal logic for reasoning about the interaction between belief, evidence and trust is presented. Other approaches are interested in analyzing trust in information sources [5, 24, 27, 67]. Moreover, some proposals have addressed trust in the context of computer security [45, 75].

Most of these approaches focus on the cognitive side of trust (i.e., trusted agents are capable of exhibiting beliefs, desires, and intentions properties). Hence, the trust is considered as a belief of an agent (the truster) involving ability and willingness of the trustee to perform some actions for the truster. Since these agents are autonomous and heterogeneous, such a mental concept cannot make those agents abide by the language semantics whenever they interact [9]. 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. Moreover, agents are able to show flexible and unpredictable behaviour when they are working together in an unreliable environment. Hence, deciding whether to trust other agents (for instance to perform some actions) or not is a challenging task. For instance, agents may not comply with their obligations (e.g., an agent may not send the payment for goods received). Thus, this raises the need for developing efficient methodologies to handle their present and future behaviors in order to ensure the fulfillment of the system requirements. Currently, the technique of model checking [20, 21] has attracted several contributions with a significant industrial implication. Although these contributions addressed a number of multi-agent aspects such as social commitments [9, 33] and knowledge [73, 104], model checking trust in multi-agent settings has not been sufficiently investigated yet. From this view, we aim in this research to contribute in the modeling and verification of trust systems.

To motivate our study of analyzing the trust in MASs, we use an example in the context of electronic commerce where trust is a highly desired property. Let us consider the buyers-sellers relationships. The buyer requests to purchase one or more items from the seller (i.e., the trust relationship is established between the two parties). Once the former selects an item, and the requested items are paid. The seller confirms the order and starts the delivery process. Finally, the requested items are shipped and the buyer is notified. However, online interactions are characterized by uncertainty and, moreover, the anonymity of the interaction partners. Thus, there is no guarantee that this process will be surely satisfied in concrete applications. Therefore, the need for formally specifying and automatically verifying trust-based interactions among autonomous agents are of great importance.

1.3 Methodology and Research Questions

Our review of the concept of trust in the context of MASs literature has revealed a gap in modeling trust from social perspectives point of view. The existing approaches consider a cognitive concept of trust where trust is defined as an attitude of the truster who believes that the trustee has a given property. Although these approaches are highly appropriate to reason about trust, their verification faces a fundamental limitation due to their reliance on the internal structure of the interacting agents. In fact, the distributed and open nature of MASs makes the capability of handling and verifying the trust interaction issues of such approaches arduous. That is, the challenge of automatically detecting the undesirable behaviors of such agents' and then fixing them according to preset specifications is an important issue that arises in the cognitive semantic approaches. Another issue that has attracted our attention while reviewing the literature is the limitation of the approaches that addressed the model checking problem of trust logics. The current proposals 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. While these approaches have the advantage of detecting and isolating different kinds of attacks, they lack the generality (i.e., the proposed assessment tools can only be applied to a single particular model). Moreover, only known and predefined attacks have been considered in the evaluation process. Besides, theses approaches are not designed to formalize and verify trust for autonomous MASs. 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. 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 research. In order to

do so, different important research questions arise:

Question 1. How can we define a temporal logic that is capable of specifying the trust properties from social perspectives viewpoint?

To address this question, we started by investigating the possibility of using the existing temporal logics such as LTL [91] and CTL [43]. However, we realized that the needed modality for trust cannot be expressed using such logics. In fact, these logics lack the capabilities to model trust interactions and the dynamic behaviors of autonomous agents. Therefore, we propose to extend CTL logic with a trust operator to represent and reason about the properties that an agent requires to be achieved by the trusted agent. The main reasons that encouraged us to extend CTL logic are: (1) CTL logic has grounded semantics, which means it can be associated to computational models, (2) there are several open model checker tools that support this logic, and (3) such logic has a high efficient model checking procedure.

Moreover, we associate with the logic a set of reasoning rules along with their formal proofs to capture the common reasoning patterns that uniformly apply to trust relationships and agents are expected to respect them when they engage in interactions.

From the semantics perspectives, we define a new trust accessibility relation to capture the trust relationship between the interacting agents by extending the original framework of interpreted systems [44]. 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 social concept between interacting agents in MASs. Given that, the direct question is:

Question 2. How can we formally verify the developed temporal logic?.

In order to reduce and eliminate post-development costs and increase confidence on the safety, efficiency and robustness, two verification techniques at design time have been

put forward to verify trust logic: direct and indirect verification techniques. A direct method can be performed by either developing a proper model checker from scratch or by extending existing tools with new algorithms for the needed temporal modalities. In contrast, indirect techniques, also called transformation-based methods, can be performed by applying certain reduction rules in order to transform the problem at hand to an existing model checking problem. In this research, we aim to explore both techniques in order to compare between them in terms of memory consumption, efficiency, and scalability with regard to the number of considered agents. Knowing all these facts, our next research question:

Question 3. How can we evaluate the proposed solution for the model checking problem of the developed temporal logic?

Two evaluation methods have been put forward: Empirical and Theoretical. The first method is evaluated by applying the proposed algorithms in real world case studies and report the experimental results, and the second method is to explore the theoretical analysis by analyzing the time and space complexity. By the end, a key question that need to be asked is how much should we trust?

Question 4. How the degree of trust that an agent has toward anther agent can be computed and verified?

Trust in the existing approaches has often been treated as either true or false, i.e., we either trust the behavior of an agent or not. However, such systems have also quantitative temporal properties (such as degrees of trust), which still need further attention from the logical and model checking perspectives. In fact, in many contexts, it is quite difficult to determine with absolute certainty whether a proposition about the behaviours of potential agents is true or false. For instance, I might trust the agent to a certain degree in relation to given propositions (i.e., I may have only 50% of trust). In our research, we aim to address

this issue by introducing a logical-based framework for quantifying the relationships among the interacting agents.

Figure 1.1 summarizes the research methodology w.r.t the identified research questions and maps them to thesis chapters.

1.4 Contributions

The following contributions are offered by this thesis:

- We introduced the Trust Computation Tree Logic (TCTL) that extended the standard CTL logic with a new temporal modality to represent and reason about preconditional trust, and then defined its formal semantics. We also associated the logic with a set of reasoning rules along with their formal proofs.

- We introduced a new vector-extended version of interpreted systems to capture the trust relationship between the interacting agents.

- We designed a new symbolic model checking algorithm to formally and automatically verify the system under consideration against some desirable properties expressed using the proposed logic. We fully implemented our proposed algorithms by extended the MCMAS model checker for MASs. Hence, a new model checker tool, called MCMAS-T, dedicated to TCTL, along with its new input language VISPL (Vector-extended ISPL) have been introduced.

- We expressed the concept of conditional trust by extending TCTL, the extended logic is called TCTL^C. Then, we developed a new model checking framework for the TCTL logic of preconditional trust that is extended to design a new algorithm to

model check conditional trust TCTL^C. In particular, we introduced transformation-based algorithms and implemented them in a Java toolkit that automatically interacts with the NuSMV model checker of the CTL logic.

- We introduced a logical language called TCTL^G, an extension of TCTL that allows us to formally represent and reason about the quantitative aspect of trust. Moreover, a dedicated symbolic model checking algorithm for TCTL^G implemented on top of MCMAS is presented (MCMAS-G).

- We computed the time and space complexity of the model checking problem of the developed temporal logics. We proved that the time complexity of TCTL, TCTL^C, and TCTL^G model checking algorithms in explicit models is P-complete with regard to the size of the model and length of the formula, and the complexity of the same problems for concurrent programs is PSPACE-complete with respect to the size of the program's components.

- We successfully applied the proposed logic and tools to model check different application domains.

1.5 Thesis Structure

We present in Chapter 2 the background needed to understand the different concepts of our research work. In particular, we give an overview of the CTL logic, summarize the formalism of interpreted systems, and review the model checking techniques. Then, we provide literature reviews on the main approaches on trust modeling. We compare the existing approaches with regard to the proposed criteria and highlight some potential research gaps that are addressed in the thesis.

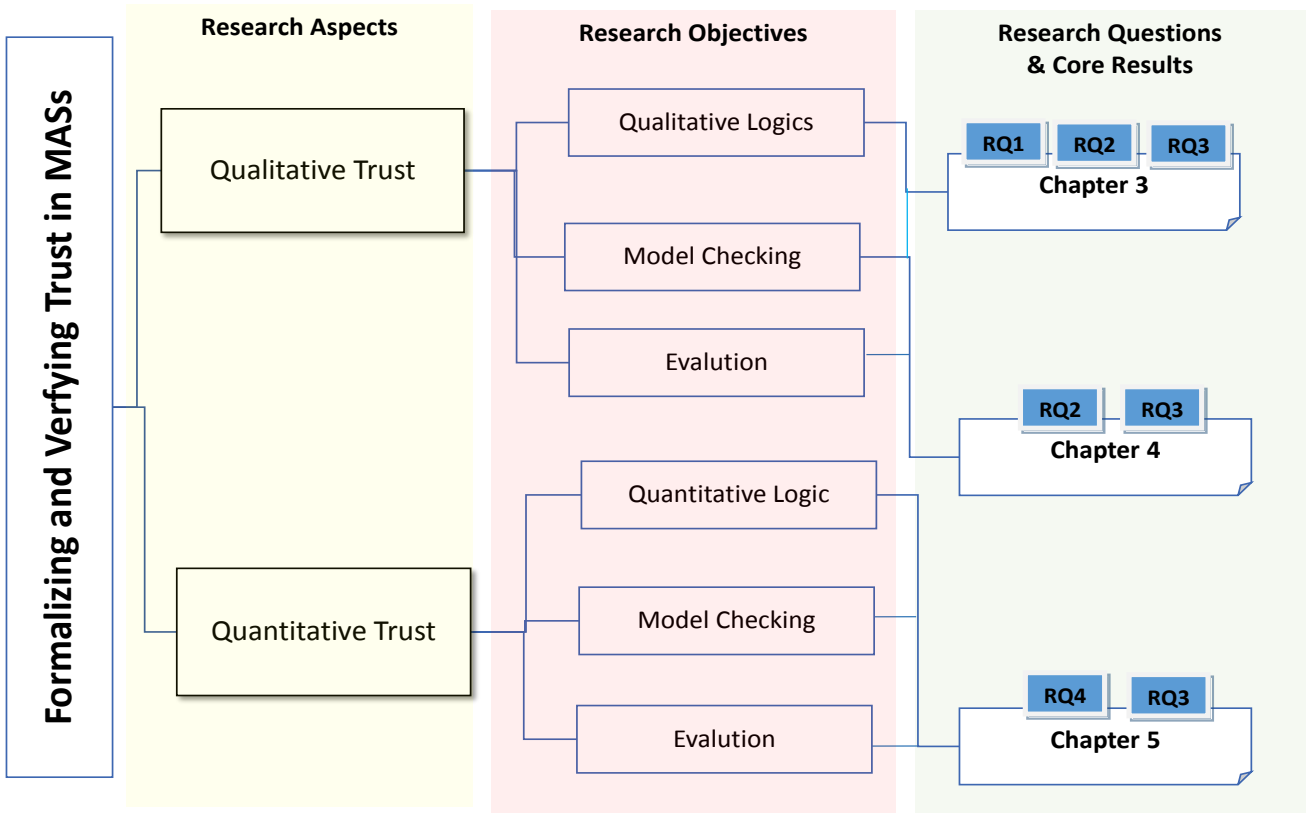


Figure 1.1: Research methodology w.r.t the identified research questions

In Chapter 3, we present the syntax and semantics of the developed TCTL logic. A set of reasoning rules are also presented. To verify the trust interactions, new model checking algorithms dedicated to TCTL logic are introduced. We also present the full implementation of our algorithms on top of the MCMAS symbolic model checker. Our implementation extends the input language of MCMAS in order to parse the TCTL logic syntax and semantics. Thereafter, we evaluate the tool and report experimental results using a real-life scenario in the healthcare platform.

In Chapter 4, we investigate a different model checking technique for TCTL logic. Moreover, we extend TCTL logic by introducing a new modality for conditional trust to produce a new logic called TCTL^C. In this chapter, the problem of model checking TCTL is transformed to the problem of model checking CTL by means of transformation-based algorithms that are extended to design a new algorithm to model check conditional trust TCTL^C. The algorithms are implemented in a Java toolkit that automatically interacts with the NuSMV model checker of the CTL logic. Further, we analyze the time complexity of TCTL model checking in explicit models and its space complexity in concurrent programs. Moreover, we evaluate the effectiveness and efficiency of our approach by performing a set of experiments on a widely-used case study in business domain and compare our results with the results obtained in Chapter 3.

In Chapter 5, we address the quantitative aspect of trust from the modeling and verification perspectives. We start by constructing TCTL^G, a logical language to represent the degrees of trust along with its reasoning postulates. Moreover, a new symbolic model checking algorithm for quantifying the relationships among the interacting agents is presented. The implementation of the model checker MCMAS-G, an extended version of the MCMAS model checker is introduced. Moreover, we investigate the complexity and evaluate our approach using a case study in the health care domain.

We summarize the obtained results and sketch possible extension of this work in Chapter 6

Chapter 2

Background and Literature Review

In this chapter, we briefly present some preliminaries needed for the rest of the thesis. In Section 2.1, we explore Computation Tree Logic (CTL) as the main ingredient that we used to define a formal semantics for trust in our proposed approach. Section 2.2 is devoted to briefly review the formalism of interpreted systems, which provides a very popular framework for modeling and reasoning about MASs. In Section 2.3, we provide the relevant background of symbolic model checking techniques and tools. Thereafter, in Section 2.4, we discuss relevant related work on current logical-based frameworks in trust-based MASs. We explain each proposal and point out if it meets the introduced criteria. Finally, we highlight the main features of our proposed framework and present some potential research gaps.

2.1 Computation Tree Logic - CTL

CTL [43] logic is a branching time logic with modal operators to describe the temporal order of events. The CTL represents a tree-like structure time where each moment in time may split into many possible paths in future.

Definition 2.1. (Syntax of CTL)

The syntax of CTL formulae is defined as follows:

$$\phi ::= \rho \mid \neg\varphi \mid \varphi \vee \varphi \mid EX\varphi \mid EG\varphi \mid E(\varphi U \varphi)$$

where $\rho \in AP$ is an atomic proposition from the set of atomic propositions AP, E is the existential quantifier over paths, the formula $EX\varphi$ stands for "φ holds in the next state in at least one path", $EG\varphi$ stands for "there exists a path in which φ holds globally", and the formula $E(\varphi U \psi)$ holds at the current state if there is some future moment for which ψ holds and φ holds at all moments until that future moment. $EF\varphi$ is the abbreviation of $E(trueU\varphi)$. A, the universal quantifier over paths, can be defined in terms of the above as usual: $AX\varphi = \neg EX\neg\varphi$; $AG\varphi = \neg EF\neg\varphi$; and $A(\varphi U \psi) = \neg(E(\neg\psi U(\neg\varphi \wedge \neg\psi)) \vee EG\neg\psi)$.

The semantics of CTL formulae is given in terms of a transition system $M = (S, R, V, I)$ where S is a nonempty set of states, $R \subseteq S \times S$ is a transition relation, $V : S \to 2^{AP}$ is a valuation function, and $I \subseteq S$ is a set of initial states. The transition relation R models temporal transitions among states: given two states $s, s' \in S$, $(s, s') \in R$ means that s' is an immediate successor of s. The behaviour of the system model M is a set of computation paths. 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 2.2. (Satisfaction)

Given the model M, the satisfaction for a CTL formula φ in a global state s, denoted as $(M, s) \models \varphi$, is recursively defined as follows:

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

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

$-(M, s) \models \varphi_1 \vee \varphi_2$ iff $(M, s) \models \varphi_1$ or $(M, s) \models \varphi_2$;

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

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

– $(M,s) \models E(\varphi_1 U \varphi_2)$ iff there exists a path π starting at s for some $k \geq 0, (M,\pi(k)) \models \varphi_2$ and $\forall\ 0 \leq i < k$, $(M,\pi(i)) \models \varphi_1$;

2.2 Interpreted Systems

An interpreted system [44] 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. The original version of interpreted systems has been extended in various ways. For instance, Bentahar et al. [10] and El-Menshawy et al. [39] extended the formalism of interpreted systems with sets of shared and unshared variables to account for agent communications that occur during the execution of MASs.

Here, we present the standard semantics of interpreted systems as in [44]. Consider a set $Agt = \{1,\cdots,n\}$ of n agents and 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 [44], a global state $s \in S$ represents a snapshot of all agents in the system at a given time. A global state s is a tuple $s = (l_1, \ldots, l_n)$. The set of all global states $S \subseteq L_1 \times \ldots \times L_n$ is a non-empty subset of the Cartesian product of all local states of n agents. The notation $l_i(s)$ is used to represent the local state of agent i in the global state s. $I \subseteq S$ is a set of initial global states for the system. The global evolution function of the system is defined as follows: $\tau : S \times ACT \longrightarrow S$, where $ACT = Act_1 \times \ldots \times Act_n$ and each component $a \in ACT$ is called a joint action, which is a tuple of actions. As in [44], 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.

2.3 Symbolic Model Checking Technique

The technique of model checking [20, 21] is used in both software and hardware industries since the 1980's. The goal of model checking is to verify the system correctness against desired properties at design time.The system is modeled as a finite-state transition system and the properties (the specifications) of the system that need to be verified are formulated in temporal logics, and then it is systematically checked whether the specifications are met for a given system (the model). If they are not met, a counterexample is produced which provides a helpful tool for debugging the system design. Many verification tools are developed for this purpose such as SPIN [50], NuSMV [19], MCMAS [73], and PRISM [62]. SPIN is an automaton-based model checker for Linear Temporal Logic (LTL) focusing on proving the correctness of process interactions. NuSMV is an extended version of Symbolic Model Verifier (SMV) [79] that allows checking finite state systems against specifications in both Computation Tree Logic (CTL) and Linear Temporal Logic (LTL). MCMAS [72] is a model checker for multi-agent systems which can verify a variety of properties specified

by different logics such as CTL, Computation Tree Logic of social Commitments (CTLC), and Computation Tree Logic of Knowledge (CTLK). The dedicated programming language used for describing a MAS in MCMAS is called ISPL (Interpreted Systems Programming Language). The PRISM tool [62] is widely used for checking probabilistic specifications over probabilistic model. The specifications can be expressed either in the Probabilistic Computation Tree Logic (PCTL) or in the Continuous Stochastic Logic (CSL).

Formally, let M be a state-transition graph (the system model), and φ be the property that the model has to satisfy. Then, the model checking technique is used to check whether or not the model M representing the system satisfies the logical formula φ describing a certain property.

Recently, model checking has been extended to MASs. Different approaches have been proposed to extend model checking techniques with the aim of verifying extended temporal logics with agent-related modalities. For instance, verifying epistemic properties expressed using logics of knowledge [70, 71, 89, 59], conditional and unconditional commitments [10, 41, 40, 33] and services composition [7, 34].

The main challenge in the application of model checking is the state space explosion problem. Early implementation of model checking algorithms suffered from this problem, which occurs when the number of components in the system grows up to the degree that makes the number of global states enormous. However, researchers have made considerable progress in dealing with this problem over the last three decades. The introduction of Symbolic Model Checking with Binary Decision Diagrams (BDDs) data structure [14], Bounded Model Checking (BMC), Partial Order Reduction, and Symmetry Reduction techniques succeeded in partially combating this problem. Yet, these methods still suffer from the potential memory explosion problem on modern test cases. In this thesis, we adapt a symbolic model checking (SMC) as our underlying technique due to its effectiveness in

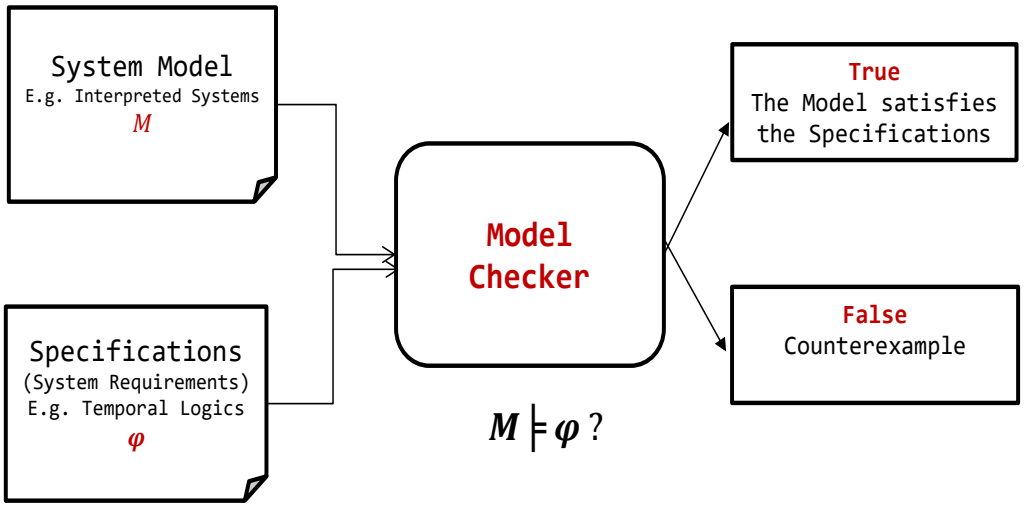


Figure 2.1: A model checker with counter and witness examples

mitigating the state explosion problem. 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)[13] 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 $(M,s) \models \phi \; \forall \; s \in I$.

Figure 2.1 describes a typically model checking process that involves three integrated phases. First, the system design formulated in some precise mathematical language. Then, specifications about the system are expressed as temporal logic formulas. Finally, the verification part where automatically verify whether a given model of a system meets a given specification.

2.4 Literature Review

In this section, we present our methodological review where we classify the current state-of-the-art in the domain of trust in MASs in three parts: qualitative logical-based approaches, quantitative logical-based approaches, and model checking trust-based techniques. In the following, we present a brief overview of each of these constituents and highlight the advantages and limitations of existing approaches and the main features of our proposed framework.

2.4.1 Qualitative Logical-based Approaches

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, [24] has proposed several approaches for trust reasoning. He developed a logic by combining a dynamic logic [47] with the BDI logic [22]. 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 [27, 75]. For instance, in [75], 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 proposed in early work [67] where the BIT, a modal logic that extends the traditional doxastic logic with modalities for representing belief, information acquisition, and trust is presented. In the BIT formalism, the trust operator is interpreted using neighborhood semantics [80]. The logic is provided with a rigorous semantics to precisely characterize 1) the relationships among beliefs and

information acquisition, and 2) how different trust properties are represented by considering various axioms of the logic. Indeed, this well-known, early formal treatment of trust, has been followed by many researchers. For instance, [87] has simplified the semantical treatment of the trust operator given in [67] by retaining only two types of modalities, a belief operator and trust, and they do not make use of the additional information operator. The idea of this work is to concentrate only on the interrelation between trust and belief. Their alternative logic of trust can also explicate a type of trust that is linked to honesty or sincerity. It allows to consider different forms of honesty separately and to incorporate these already into the base logic. Trust in the sincerity is also presented recently in [18]. The authors proposed a modal logic to reason about an agent's trust in the sincerity towards a statement formulated by another agent. In this work, trust is represented as a normal modality that allows to reason about trust when an agent attempt to manipulate other agents. Moreover, the authors exhibited some notable properties such as non-transitivity of trust, and they proved the soundness and completeness of the proposed logic.

Following the path of cognitive trust, [48] proposed a formal logical framework to evaluate agents' behavior in a multi-agent environment. 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. In another work [49], Herzig and his colleagues simplified the previous logic presented in [48] by considering a very simple kind of actions based on the concepts of propositional assignment. That is, the truth values of a propositional variable is assigned to either true or false by the corresponding agents'

actions. The new logic provides a simple framework, and it is expressive enough to account for the cognitive theory of trust. In [69], Liu and Lorini presented a new dynamic logic called DL-BET for reasoning about the interaction between belief, evidence and trust. The authors introduced three modal operators where each of these concepts are respectively represented. In this logic, the trust operator semantics is interpreted using neighborhood semantics [80], which maps each world into a set of subsets of worlds. The authors provided a complete axiomatization for both the static component of the proposed logic and its dynamic extension. Another logic-based proposal for trust has been initiated by [15] 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 formulas in order to manipulate them using first-order logic. Fuchs and colleagues [45] also addressed trust in the context of computer security. Moreover, more recent proposals have made the link between trust and argument-based frameworks [85, 101].

The closest approach to our work is the one presented by Singh in [98] where the social perspective of trust has been put forward. Specifically, the author 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) [91] and modality (C) for commitments [97] and (T) modality for trust. The semantics is interpreted using neighborhood semantics [80], 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 $\mathbb{k}$ is computed, which makes the approach quite difficult to be applied in practice.

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. Moreover, model checking neighborhood semantics-based modal logics is yet to be solved [38, 84].

2.4.2 Quantitative Logical-based Approaches

Indeed, there are relatively small amounts of directly related work. For instance, the work in [26, 5] proposed several approaches that address the graded trust. They developed a logic by combining a dynamic logic [47] with a BDI-like logic [22]. The author in [25] defined a logical framework to represent graded trust in terms of two independent components: graded beliefs and graded regularities. In this work, trust is reduced to graded beliefs, so the graded trust is defined as the strength level of the truster agent belief about the trustee agent sincerity. In another proposal, Demolombe and Lorini [74] have focused on analyzing the trust that can be associated with information sources. The authors have integrated graded beliefs into a logical framework that defines different kinds of trust. In another work [76], Lorini et al. considered the quantitative aspects of trust in a dynamic epistemic logic setting, where the relationship between trust and belief change is presented.

The authors proposed Dynamic Logic of graded Belief and Trust (DL-BT), a modal logic that supports reasoning about agents changing their beliefs based on the degree of trust the receiver agent has in the information source. The proposed logic combines modal operators of knowledge, graded belief and trust with dynamic operators of trust-based belief change. The graded trust operator is interpreted using a neighborhood semantics [80], whose model checking is still an open problem [38]. In this work, two kinds of trust-based belief change policies have been considered: additive and compensatory policies, along with the detailed analysis of their logical properties. Moreover, the authors provided a sound and complete axiomatization. In [81], the authors introduced a logical framework that combines a formal logic based on a logic of belief, a logic of time, and a dynamic logic to cognitively represent the concept of trust (qualitative aspects), and a fuzzy logic to represent the degree (quantitative aspect) of trust. However, the proposed logics mostly focused on agents with mental states where the trusting entity is normally capable of exhibiting beliefs, desires, and intentions, which makes them difficult to be model checked. Moreover, the trust from the quantitative approach is not considered because the grades are embedded in the modal operators. [93] present a logic called Certain Logic to evaluate trust under uncertainty by evaluating a number of Boolean expressions in terms of real values.

The author in [100] proposed a logical language that can be employed to reason about trust, knowledge, and their interaction. The proposed logic is a modal language augmented with a trust operator, which is interpreted using a combination of a neighborhood structure [80] with an added component to assign weights to formulas. Moreover, they provided a mechanism that can produce values to be fed into Subjective Logic's trust manipulation component [53]. The idea of this work is to use the expressive power of the proposed formal language to describe the information possessed by an agent and then transform this

knowledge into an opinion value about a given proposition with all the three major components of subjective logic made explicit. Such components are, respectively, belief, disbelief, and uncertainty. Then, an evidence based logic is built, such that evidences are assigned weights that determine whether an agent trusts a given proposition or not.

Huang et al. [51] considered the setting of stochastic multi-agent systems, where an automated verification framework for quantifying and reasoning about cognitive trust is proposed. The authors focused on a quantitative notion of trust defined as a subjective evaluation in order to capture the social notation of trust. This allows one to quantify the amount of trust as a belief-weighted expectation. In this work, a probabilistic rational temporal logic PRTL*, which extends the logic PCTL* with reasoning about agents' mental states is introduced. However, the model checking of the proposed logic was proven undecidable. Yet, no implementation has been considered and no attempt to evaluate the approach on any case studies. Considering our approach, trust is defined from a high-level abstraction without having to depend on agent's internal mental states, and moreover, quantifies trust by relying only on the accessibility relations. Furthermore, our model checking algorithm is proved space-efficient.

2.4.3 Model Checking-based Approaches

Some relevant approaches with the aim of verifying trust-based models using formal methods have appeared recently. For instance, Aldini in [3] has been introduced a formal framework to evaluate the effectiveness and robustness of trust-based models in order to detect and then isolate different kinds of attacks . Indeed, the author integrates trust modeling with distributed systems. In this work, the system properties are expressed using a trust temporal logic (TTL) which combines CTL [43] and its action-based extension (ACTL) [23]. Moreover, the trust system model is based on an instance of both labeled transition systems

and Kripke structures. The verification of temporal logic properties expressed in TTL has been performed through a mapping to an existing model checking technique. However, the model mapping between the two logics has not been specified and TTL can only specify a single agent model, and it is not adapted to autonomous MASs. In [83], the authors presented a mechanism for specifying and model checking trust specifications against models (such as Deontic models) of multi-agent interactions. The authors in [108] considered the formal verification of auction mechanisms. They implemented simple and intuitive auction models in a BDI-based programming language, to which they applied agent verification techniques. In this work some sophisticated agent aspects such as goals, intentions, beliefs and deliberation within an auction context has been verified. Moreover, they verified some of the scenarios that is involving a dynamic and static notions of trust. The verification carried out using an agent model-checking system and the properties verified are given in a logic of belief, goals and time. However, the approach has been tested on some small multi-agent programs: variations of the contract net protocol and auction systems, and with very fewer agents. Overall, their verification system is relatively slow where the speed and space required were the main problem.

The authors in [11] 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 [42], 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. Further, none of these approaches tend to focus on model checking quantitative trust.

On the other hand, model checking trust can also be achieved by indirect techniques, also called transformation-based methods. The idea is to apply certain reduction rules in order to transform the problem at hand to an existing model checking problem. In fact, transformation has been acknowledged as an alternative mechanism for verifying various MASs aspects. The main advantage of this technique is that it enables the designers of MASs applications to get benefit from powerful and already tested model checkers. This technique has been applied for model checking commitments [36], knowledge [71], and the interaction between knowledge and commitments [2, 99]. To the best of our knowledge,

Table 2.1: Comparison between the list of publications reviewed for this thesis with respect to the proposed criteria.

Approach	C1	C2	C3	C4	C5	C6	C7
[24] [27] [75] [49]	√	–	–	–	–	–	–
[48]	√	–	–	–	–	–	√
[67] [87] [18]	–	√	√	–	–	–	√
[98]	–	√	√	–	–	–	–
[5] [74][25] [81]	√	–	–	–	√	–	–
[3] [42]	–	√	–	–	–	√	–
[76]	√	–	√	–	√	–	–
[26]	√	–	√	–	√	–	√
[69]	√	–		–	–	–	√
[51]	√	–	–	√	√	√	–
Our approach	–	√	√	√	√	√	–

our work is the first attempt that introduces and implements a full transformation technique for verifying trust specifications in MASs.

In summary, we compare only the existing approaches that are mainly related to our research by taking into consideration the following criteria: being cognitive, being social, considering explicit notion of trust, analyzing complexity, dealing with graded modality, supporting model checking, and proving the soundness and completeness. We refer to these criteria as *C1*, *C2*, *C3*, *C4*, *C5*, *C6*, and *C7* respectively. Cognitive and social properties indicate whether the logic is based on BDI logics (i.e., logics that describe the mental attitudes of agents in terms of beliefs, desires and intentions) or other temporal logics. Explicit notion of trust shows if it is possible to express trust by an explicit modality (i.e., logics that rely on a trust modality) or by means of one or more predicates. Moreover, complexity analysis states whether it is considered or not in the proposed approach. Graded modality indicates whether the work addresses the quantitative aspects of trust. Applicability for model checking confirms the presentation of a formal verification technique to verify the proposed approach, or specifying if the approach is applicable for model checking. Finally,

check whether they address the soundness and completeness of the proposed logic. A summary about the comparison between the existing approaches and our approach with respect to the proposed criteria is presented in Table 2.1.

2.5 Summary

In this chapter, we introduced the background and concepts needed for the rest of the thesis. Moreover, we also presented a revision of the most relevant related work. In the next chapter, we propose a new approach for modeling and verifying trust in MASs.

Chapter 3

Logic-based Framework for Specifying and Model Checking Trust in MASs

In this chapter, we present a new logic-based framework for specifying and model checking preconditional trust in MASs. We start by introducing TCTL, a new temporal logic of trust that extends the Computation Tree Logic (CTL) to enable reasoning about trust with preconditions (Section 3.2). A new vector-extended version of interpreted systems is defined to capture the trust relationship between the interacting parties in Section 3.3. In Section 3.4, we introduce a set of reasoning postulates along with formal proofs to support our logic. Moreover, Section 3.5 presents new 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 by extended the MCMAS model checker for MASs. Hence, a new model checker tool called MCMAS-T, dedicated to TCTL, along with its new input language VISPL (Vector-extended ISPL) have been created. We evaluated the tool and reported experimental results using a real-life scenario in the health-care domain[1].

[1]The results of this chapter are collected from our publications in [31, 32]

3.1 An Overview of The Proposed Approach

The study of trust in MASs has been an area of interest for many researchers over the last years. This is due to the fact that trust is the basis for agent communication wherein entities have to operate in a dynamic and uncertain environment. Several approaches have been proposed to define logical semantics for trust in MASs. However, these approaches are limited to reason about trust based on the sole agents' mental states. Therefore, in this research, we consider trust from a high-level abstraction based on the social behaviors of agents. Specifically, we propose a logical framework that allows us to reason about preconditional trust and time. Figure 3.1 illustrates the main parts of the proposed approach. It specifically consists of three different, but fully integrated parts. In the logical language part, we develop a new branching-time trust temporal logic called TCTL. This logic is an extension of the CTL logic [43] with a new operator for trust along with its intuitive semantics to effectively modeling trust interactions as temporal modality. In this part, we also express reasoning rules with proofs that they are supported in the proposed logic. In the algorithmic part, we develop new algorithms to tackle the problem of model checking TCTL by examining both explicit and symbolic state model checking. In the third part, we completely implement our algorithm on top of the model checker MCMAS [73] that results in a new open source tool called MCMAS-T. We also extend the input modeling and encoding language of MCMAS with the vector-based semantics to produce a new one called VISPL.

3.2 Trust Computation Tree Logic TCTL

TCTL is a combination of branching time temporal logic (CTL) [43] with trust modality for reasoning about trust and time.

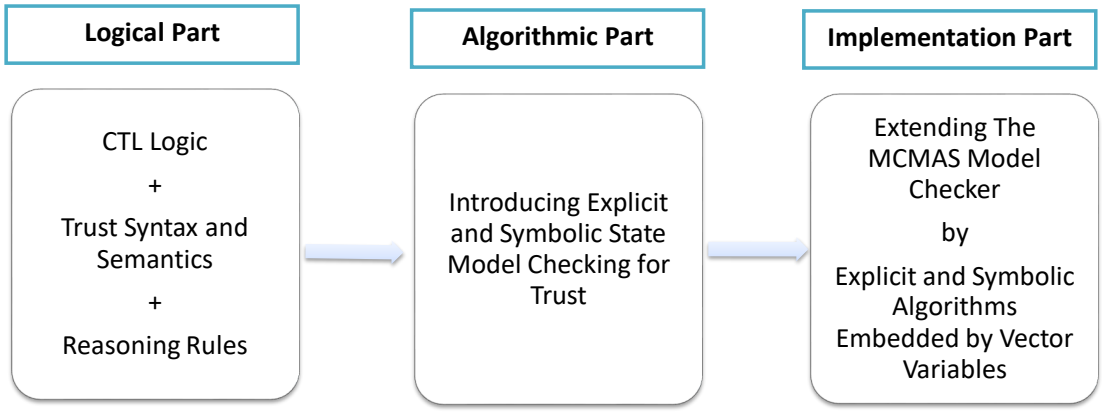


Figure 3.1: The main parts of the proposed approach

Definition 3.1. (Syntax of TCTL)

The syntax of TCTL is defined as follows:

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

where $\rho, E, A, X, G, \vee$, and U are defined in Definition 2.1 (Chapter 2). The modality $T_p(i,j,\psi,\varphi)$ stands for *"Preconditional Trust"* and is read as "the truster i trusts the trustee j to bring about φ given that the precondition ψ holds". That is, we have the trust over the content given that the precondition is satisfied. 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 $i \geq 0$,$(s_i, s_{(i+1)}) \in R$.

Example 3.1. *The following formula represents a simple interaction between a buyer and a seller in an e-commerce setting. It states the buyer trusts the seller will deliver the requested items under the precondition that the latter has received the payment.*

$$T_p(buyer, seller, Items\_Paid, Items\_Delivered)$$

3.3 Vector-based Interpreted Systems

In order to account for the trust relationship between the truster and trustee, we extend the original formalism of interpreted systems [44] introduced in Section 2.2 to explicitly capture the trust relationship that is established between agents engaged in an interaction. We introduce the notion of agents' vector v. That is, for each agent $i \in Agt$, a vector v^i of size $|Agt|$ is associated with each local state $l_i \in L_i$ of this agent. $v^i(i), v^i(j), \ldots, v^i(k)$ are the components of the vector v^i where $(i, j, \ldots, k) \in Agt^{|Agt|}$. This vector will be used to define the trust accessibility relation. Indeed, the set of local vectors v_i represents the vision of agent i with regard to the trust of other agents. This extension allows us to provide an intuitive semantics for direct trust that takes place between interacting parties. The vector-extended formalism is composed of:

- A set $Agt = \{1, \ldots, n\}$ of n agents in which each agent $i \in Agt$ is described by:
  - A non-empty set of local states L_i, which represents the complete information that the agent can access at a particular time;
  - A set of local actions Act_i to model the temporal evolution of the system;
  - A set of local vectors v_i.
  - A local protocol $\rho_i : L_i \rightarrow 2^{Act_i}$ assigning a list of enabled actions that may be performed by agent i in a given local state L_i;
  - A local evolution function τ_i is defined as: $\tau_i = L_i \times Act_i \rightarrow L_i$, which determines the transitions for an individual agent i between local states;
- A set of global states $s \in S$ that represent a snapshot of all agents in the system at a given time. A global state s is a tuple $s = (l_1 \ldots l_n)$. The notation $l_i(s)$ is used to represent the local state of agent i in the global state s. Similarly, the notation $v_i(j)(s)$

is used to represent the jth component of the local vector of agent i in the global state s;

- $I \subseteq S$ is a set of initial global states for the system;
- The global evolution function of the system is defined as follows: $\tau : S \times ACT \longrightarrow S$, where $ACT = Act_1 \times \ldots \times Act_n$ and each component $a \in ACT$ is called a joint action, which is a tuple of actions.

Definition 3.2. (Model of TCTL)

A model of trust generated from the vector-based interpreted systems is a tuple $M = (S, R, I, \{\rightsquigarrow_{i \to j} | (i, j) \in Agt^2\}, V)$, 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;
- $\rightsquigarrow_{i \to j} \subseteq S \times S$ is the direct trust accessibility relation for each truster-trustee pair of agents $(i, j) \in Agt^2$ defined by $\rightsquigarrow_{i \to j}$ iff:
  - $l_i(s)(v^i(j)) = l_i(s')(v^i(j))$;
  - s' is reachable from s using transitions from the transition relation R;
- $V : S \to 2^{AP}$ is a labeling function, where AP is a set of atomic propositions.

The intuition of the relation $\rightsquigarrow_{i \to j}$ is, for agent i to gain trust in agent j, the former identifies the states that are compatible with their trust vision with regard to the latter, i.e., where agent i is expecting that agent j is trustful. Specifically, for two global states $s, s' \in S$, $s \rightsquigarrow_{i \to j} s'$ obtained by comparing the element $v^i(j)$ in the local state l_i at the global state

s (denoted by $l_i(s)(v^i(j))$) with $v^i(j)$ in the local state l_i at the global state s' (denoted by $l_i(s')(v^i(j))$). Thus, the trust accessibility of agent i towards agent j (i.e., $\rightsquigarrow_{i\to j}$) does exist only if the element value that we have for agent j in the vector of the local states of agent i for both global states is the same, i.e., $l_i(s)(v^i(j)) = l_i(s')(v^i(j))$. Finally, infinite sequences of states linked by transitions define paths. If π is a path, then $\pi(i)$ is the $(i+1)$th state in π. This idea is illustrated in Figure 3.2. In the figure, the solid line represents the transition relation from R, and the dashed line represents the direct trust accessibility relation $\rightsquigarrow_{i\to j}$. In this example, s' is compatible with s with regard to the trust of agent i towards the agent j. We assign a vector to each agent's local states. v_3^i 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 s and s'. The particular value of the $v^i(j)$ of agent i is the same in both states.

Definition 3.3. (Semantics of TCTL)

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:

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

The formal semantics of the CTL formulae, a temporal fragment of TCTL, is introduced in Definition 2.2. For the state formula $T_p(i,j,\psi,\phi)$, it is satisfied in the model M at s iff (1) there exists a state s' such that $s' \neq s$ and $s \rightsquigarrow_{i\to j} s'$, 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.

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

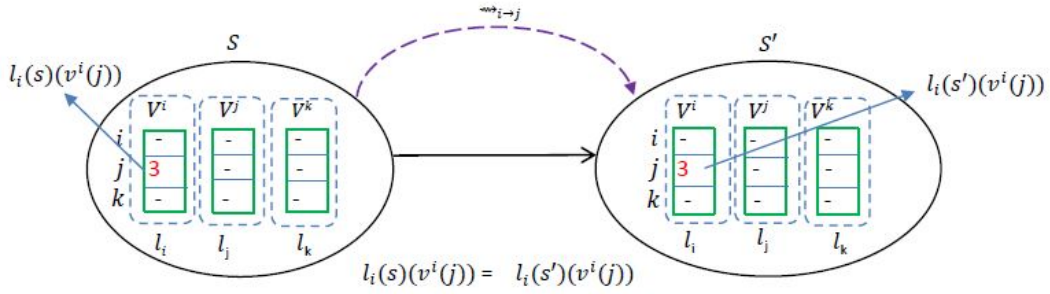


Figure 3.2: An example of accessibility relation $\rightsquigarrow_{i\to j}$

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

3.4 Reasoning Postulates and The Case Study

In this section, we consider the Breast Cancer Diagnosis and Treatment (BCDT) [2] 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.

3.4.1 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*). In [102, 33], the authors

[2]Available at: `http://aspe.hhs.gov/sp/reports/2010/PathRad/index.html`

formalized this scenario in terms of commitments, identifying the contractual business relationships among the parties involved. Indeed, such relationships can be founded as a basic of defining trust specifications as requirements for engineering contracts among parties.

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.

3.4.2 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 [98].

P1: Fulfillment. $\phi \rightarrow \neg T_p(i, j, \psi, \phi)$

- **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 $T_p(i, j, \psi, \phi)$ 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: $Mammo\_Req \rightarrow \neg T_p(p, ph, Mass\_Not, Mammo\_Req)$.

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

- **Meaning:** The trust to bring about a conjunction is the trust for each part separately unless it does already hold.

- **Proof:** Assume that $(M, s) \models T_p(i, j, \psi, \phi_1 \wedge \phi_2) \wedge \neg\phi_1$. 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 $s \rightsquigarrow_{i \rightarrow j} s'$, and all the trust accessible states s' such that $s \neq s'$ and $s \rightsquigarrow_{i \rightarrow j} s'$ satisfy $\phi_1 \wedge \phi_2$. Thus, these states also satisfy ϕ_1 . Consequently, $(M, s) \models T_p(\psi, \phi_1)$.

- **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 a mammography is requested, then the physician trusts the radiologist for the collection of the diagnostic tissue, unless this tissue has been already obtained. Formally, $T_p(ph, rg, Mammo\_Req, Tiss\_Coll \wedge Tiss\_Send) \wedge \neg Tiss\_Coll \rightarrow T_p(i, j, ph, rg, Mammo\_Req, Tiss\_Coll)$.

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

- **Meaning:** If the trust to bring about a conjunction holds, then at least the trust to bring about one part holds.

- **Proof:** Assume that $(M,s) \models T_p(i,j,\psi,\phi_1 \wedge \phi_2)$. 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 $s \rightsquigarrow_{i\to j} s'$, and all the trust accessible states s' such that $s \neq s'$ and $s \rightsquigarrow_{i\to j} s'$ satisfy $\phi_1 \wedge \phi_2$. Thus, for all those states s', $(M,s') \models \phi_1$ or $(M,s') \models \phi_2$. Consequently, $T_p(i,j,\psi,\phi_1) \vee T_p(i,j,\psi,\phi_2)$ 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, $T_p(ph,rg,Mammo\_Req,Tiss\_Coll \wedge Tiss\_Send) \rightarrow T_p(ph,rg,Mammo\_Req,Tiss\_Coll) \vee T_p(ph,rg,Mammo\_Req,Tiss\_Send)$.

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

- **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 $s \rightsquigarrow_{i\to j} s'$, 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: $T_p(p,ph,Mass\_Not,Mammo\_Req) \rightarrow \neg T_p(p,ph,Mass\_Not,\neg Mammo\_Req) \wedge \neg T_p(p,ph,\neg Mass\_Not,Mammo\_Req)$.

P5: Non-Vacuity. From $\psi \vdash \phi$ infer $\neg T_p(i,j,\psi,\phi)$[3]

- **Meaning:** Trust must be for something tangible.
- **Proof:** Assume that $T_p(i,j,\psi,\phi)$. 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.
- **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 $\neg T_p(ph, rg, Tiss\_Send, Tiss\_Coll)$.

P6: Precondition Slackening: From $T_p(i,j,\psi_1,\phi), \psi_1 \vdash \psi_2$ infer $T_p(i,j,\psi_2,\phi)$

- **Meaning:** If trust holds for a stronger precondition, then it holds for a weaker one.
- **Proof:** Assume that $(M,s) \models T_p(i,j,\psi_1,\phi)$. 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 $s \rightsquigarrow_{i \to j} s'$, 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:

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

1- $T_p(i,j,\psi,\phi) \rightarrow T_p(i,j,\top,\phi)$. 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- $T_p(i,j,\psi_1 \wedge \psi_2,\phi) \rightarrow T_p(i,j,\psi_1,\phi)$. This means when the trust holds for a conjunctive precondition, then it comes into effect for each part of this conjunction.

3- $T_p(i,j,\psi_1,\phi) \rightarrow T_p(i,j,\psi_1 \vee \psi_2,\phi)$. 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. $T_p(i,j,\psi_1,\phi) \wedge \psi_2 \rightarrow T_p(i,j,\psi_1 \wedge \psi_2,\phi)$

- **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 $T_p(i,j,\psi_1,\phi)$, 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. $T_p(i,j,\psi_1,\phi) \wedge \psi_2 \rightarrow T_p(i,j,\psi_2,\phi)$

- **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 P6.

- **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. $T_p(i,j,\psi_1,\phi_1) \wedge T_p(i,j,\psi_2,\phi_2) \rightarrow T_p(i,j,\psi_1,\phi_2)$

- **Meaning:** Once a trust holds, its precondition can be exchanged with the precondition of another holding trust.

- **Proof:** From $T_p(i,j,\psi_1,\phi_1)$, we have $(M,s) \models \psi_1$, and from $T_p(i,j,\psi_2,\phi_2)$, $(M,s) \models \neg\phi_2$ and there exists a state $s' \in S$ such that $s \neq s'$ and $s \rightsquigarrow_{i \rightarrow j} s'$, 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. $T_p(i,j,\psi_1,\phi_1) \wedge T_p(i,j,\psi_2,\phi_2) \rightarrow T_p(i,j,\psi_1 \wedge \psi_2, \phi_1 \wedge \phi_2)$

- **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 $s \rightsquigarrow_{i \rightarrow j} s'$, 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 health-care 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- $T_p(i,j,\psi_1,\phi) \wedge T_p(i,j,\psi_2,\phi) \rightarrow T_p(i,j,\psi_1 \wedge \psi_2,\phi)$. 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- $T_p(i,j,\psi,\phi_1) \wedge T_p(i,j,\psi,\phi_2) \rightarrow T_p(i,j,\psi,\phi_1 \wedge \phi_2)$. 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 $T_p(i,j,\psi,\phi_1), \phi_1 \vdash \phi_2, \neg\phi_2$ **infer** $T_p(i,j,\psi,\phi_2)$

- **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 $T_p(i,j,\psi,\phi_1)$, $(M,s) \models \psi$, there exists a state $s' \in S$ such that $s \neq s'$ and $s \rightsquigarrow_{i \rightarrow j} s'$, 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. $T_p(i,j,\psi,\phi) \rightarrow EF(\phi \wedge \neg T_p(i,j,\top,\neg\phi))$

- **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 $(M,s) \models T_p(i,j,\psi,\phi)$, 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. $s'_{\rightsquigarrow_{i\to j}} s''$. In this case $s' \models \neg T_p(i,j,\top,\neg\phi)$
  - Case 2: $\exists s'' \neq s'$ s.t. $s'_{\rightsquigarrow_{i\to j}} s''$. 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 $s' \models \neg T_p(i,j,\top,\neg\phi)$.

  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. $AG\neg\phi \rightarrow \neg T_p(i,j,\psi,\phi)$

- **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 $s \rightsquigarrow_{i\to j} s'$ are reachable.

- **Instance:** The following rule is a consequence of the content nonexistence postulate:

1- $\neg T_p(i, j, \psi, \bot)$. This represents the content consistency and means trust to false cannot hold.

P14: Precondition Nonexistence. $AG\neg\psi \rightarrow \neg T_p(i, j, \psi, \phi)$

- **Meaning:** There is no trust if the precondition never holds.
- **Proof:** The proof is straightforward since the first condition for $T_p(i, j, \psi, \phi)$ to hold is the satisfaction of ψ in the current state.
- **Instance:** The following rule is a consequence of the precondition nonexistence postulate:

  1- $\neg T_p(i, j, \bot, \phi)$. This represents the precondition consistency and means trust with a false precondition cannot come to effect.

3.5 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 explore two model checking paradigms: explicit state model checking and symbolic algorithm. We start by introducing explicit state model checking algorithm for TCTL. Then, we extend the standard CTL symbolic algorithm introduced in [21] by adding the procedure that deals with the trust modality in our logic.

3.5.1 Explicit Algorithm of Trust

Here, we introduce 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 $T_p(i, j, \psi, \phi)$ 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 $\rightsquigarrow_{i\rightarrow j}$. 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 1** describes the approach where the procedure $MC_T(i, j, \psi, \phi, M)$ 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 non-trivial 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 $\rightsquigarrow_{i\rightarrow j}$ (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 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 $T_p(i, j, \psi, \phi)$. 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 commitment formula $T_p(i, j, \psi, \phi)$, 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 $\neg T_p(i,j,\psi,\phi)$ 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 1 $MC_T(i,j,\psi,\phi,M)$: the set [[$T_p(i,j,\psi,\phi)$]]

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: $\quad \mathbf{V} \leftarrow \{s \in S \mid s \text{ is reachable from } x\} \setminus \{x\}$;
5: $\quad \mathbf{V}' \leftarrow \{s \in \mathbf{V} \mid l_i(s)(v^i(j)) = l_i(x)(v^i(j))\}$;
6: $\quad$ **if** $\mathbf{V}' \neq \emptyset$ **and** $\mathbf{V}' \cap \mathbf{Y} = \emptyset$ **then**
7: $\quad\quad \mathbf{Z} \leftarrow \mathbf{Z} \cup \{x\}$;
8: **end for**;
9: **return** Z;

Figure 3.3 depicts an example illustrating the algorithm. The example shows a particular case where the state s_4 that satisfies the trust formula $T_p(i,j,\psi,\phi)$ is reachable through a non-self-loop: $s_4, s_5, s_0, s_1, s_2, s_4$. Although the state s_4 has an accessible and reachable state that satisfies $\neg\phi$, which is the sate 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 1 regarding the trust formula $T_p(i,j,\psi,\phi)$ 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 $\{s_4\}$.

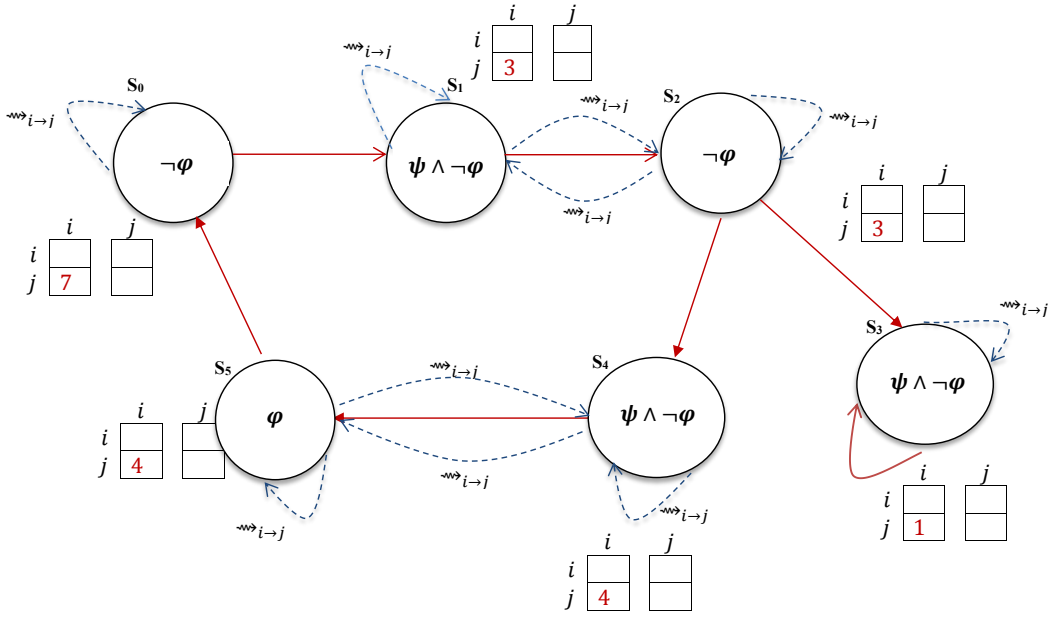


Figure 3.3: Example to illustrate Algorithm 1

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 introduce symbolic algorithm 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). We refer to explicit algorithm (**Algorithm 1**) as direct approach and symbolic algorithm (**Algorithm 3**) as revisited approach.

3.5.2 BDD-based Algorithm of Trust

We start by presenting the main algorithm (**Algorithm 2**) 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 2**), the lines 1 to 6 invoke the standard algorithms used in CTL to compute the set of states that satisfy regular CTL formulas. Line 7 calls our procedure which computes the set of states that satisfy the trust formula.

Algorithm 2 $SMC(\phi, M)$: the set $[[\phi]]$ of states satisfying the TCTL formula ϕ

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

Algorithm 3 reports the symbolic approach. In this algorithm, the computation of the set of states that satisfy the trust formula $T_p(i, j, \psi, \phi)$ is performed as follows. First, the algorithm considers the transition relation without self-loop denoted as $\mathscr{T}^F$ 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 $\rightsquigarrow_{i\rightarrow j}$ (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 $T_p(i,j,\psi,\phi)$ 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 $T_p(i,j,\psi,\phi)$ 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 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

Algorithm 3 $SMC_T(i, j, \psi, \phi, M)$: the set [[$T_p(i, j, \psi, \phi)$]]

1: $\mathscr{T}^F$ 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**

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 $\mathbf{W}'$.

Algorithm 4 $TrustRelation(\mathbf{W}, \mathbf{X}, \overline{\mathbf{X}})$

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: $\mathbf{V}' \leftarrow Preimage(\mathbf{W}', \mathscr{T}^F) \cap \overline{\mathbf{X}}$;
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

Figure 3.4 illustrates an example of a flat model. The computation of **Algorithm 3** regarding the trust formula $T_p(i, j, \psi, \phi)$ 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.

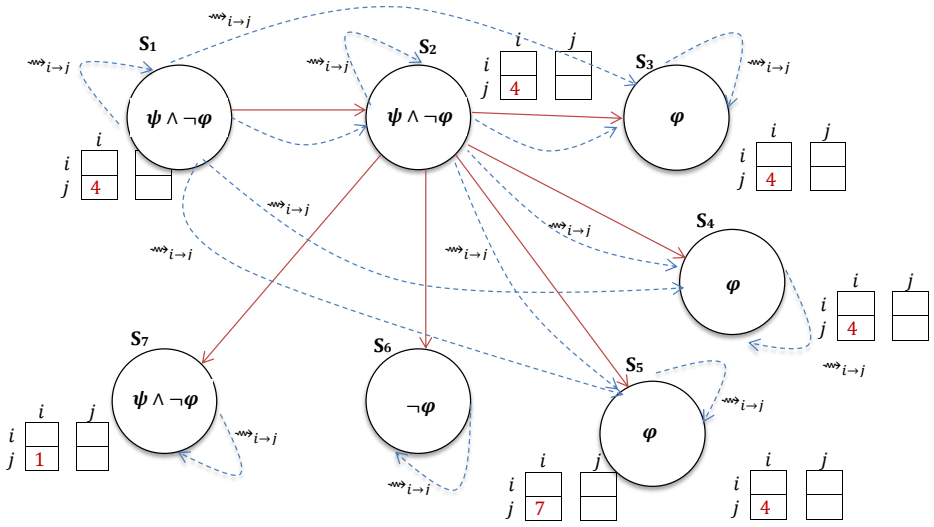


Figure 3.4: Illustrative example of model without-loop (flat)

To show that **Algorithm 3** works only for flat models which do not include non-self-loops, let us consider the case depicted in Figure 3.4. 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.

3.6 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 3.5 into the symbolic model checker MCMAS [73]. 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 [7]. Moreover, the tool has been extended to MCMAS+ to deal with social commitments [33] and has been used as the core for SMC4AC, a model checker recently launched for intelligent agent communication [35]. 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 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) respectively [4].

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

3.6.1 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 3.4. [33] 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 $M = (S, R, I, \rightsquigarrow_{i \to j}, V)$ 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.

1. $T1 = T_p(p, ph, Mass\_Not, Mammo\_Req)$; which means the patient trusts the physician to request a mammography under the precondition that a suspicious mass is noticed.

2. $T2 = T_p(ph, rg, Cal\_Det, Bio\_Rec)$; which means that the physician trusts that the radiologist will recommend a biopsy knowing that the latter has noticed a calcification.

3. $T3 = T_p(p, ph, Rep\_Rec, Treat\_Plan\_Agr)$; 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 VISPL 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.

3.6.2 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:

$\phi = AG(Mass\_Not \rightarrow EF\ T_p(p, ph, Mass\_Not, AF\ Mammo\_Req))$.

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:

$\phi = AG\neg(Mass\_Not \wedge \neg T_p(p, ph, Mass\_Det, AF\ Mammo\_Req))$.

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:

$\phi = AG(Clac\_Det \rightarrow AF\ T_p(ph, rg, \top, Biop\_Rec))$.

3.6.3 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 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 32 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 1)**, which we call hereafter the *direct technique* with the *revisited* one **(Algorithm 3)**.

Table 3.1: Verification results of the BCDT protocol using the direct algorithm

Agents	States	Time (sec)	Mem.(MB)
6	17	0.111	9
12	289	18.881	30
18	4913	201.313	46
24	83521	6883.12	48

(a) Model with-loop (non-flat)

Agents	States	Time (sec)	Mem.(MB)
6	17	0.093	9
12	289	0.953	15
18	4913	43.191	54
24	83521	1806.45	48

(b) Model without-loop (flat)

First, we start by presenting the experimental results obtained from the direct approach (**Algorithm 1**) 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 3.1 - (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

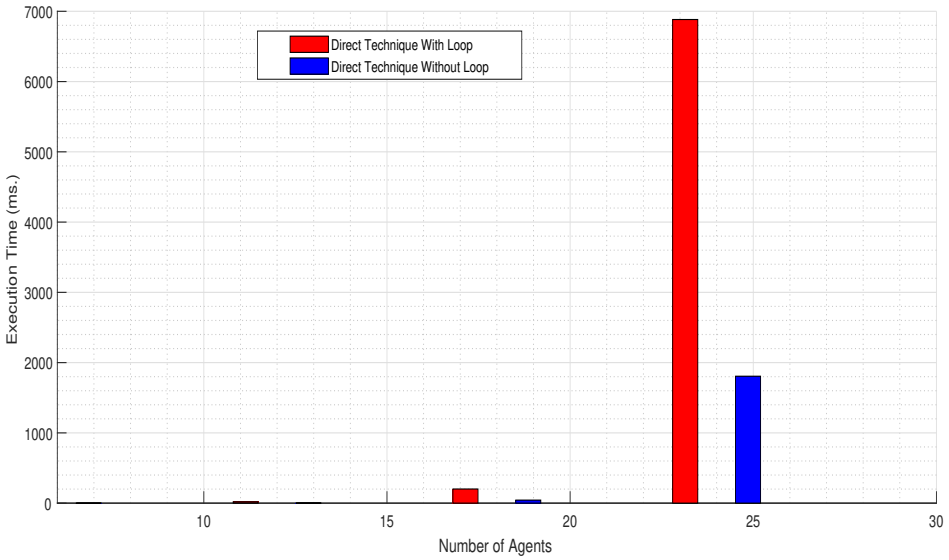


Figure 3.5: Comparison results between models with and without-loops using the direct algorithm

3.2 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 seconds even for 36 agents. Moreover, the memory usage is very comparable with the results in Table 3.1. 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

Table 3.2: Verification results of the BCDT protocol using the revisited algorithm

Agents	States	Time (sec)	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

numerical results in the form of graphs as shown in Figures 3.5 and 3.6. Figure 3.5 shows the execution time as function of the number of agents for the direct approach in models with and without-loops. Figure 3.6 compares the direct and revisited approaches using the same metric (execution time as function of the number of agents) for the models without-loops.

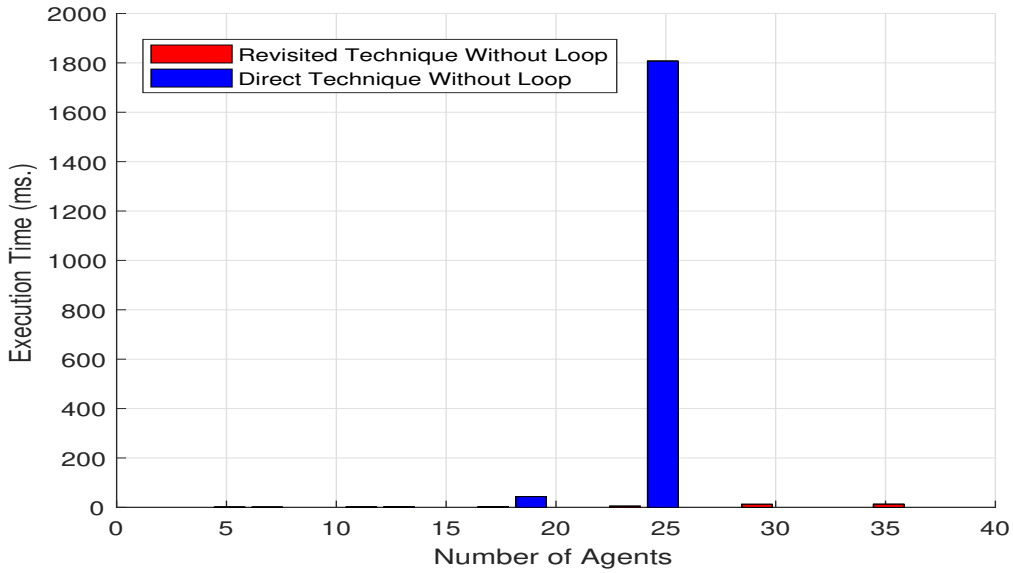


Figure 3.6: Comparison results between the direct and revisited algorithms using a flat model

3.7 Summary

In this chapter, 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 chapter 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 second contribution is the 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 health-care 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.

In the next chapter, we will introduce the notion of conditional trust by extending the logic of trust TCTL, and we will also examine a different and new model checking technique for TCTL logic.

Chapter 4

Transformation-based Model Checking Temporal Trust

This chapter starts by extending TCTL logic with a new modality for conditional trust to produce a new logic called TCTL^C (Section 4.2). Then, a new model checking framework for the TCTL logic of preconditional trust that is extended to design a new algorithm to model check conditional trust TCTL^C in MASs is presented. In particular, we introduce transformation-based algorithms and implemented them in a Java toolkit that automatically interacts with the NuSMV model checker of the CTL logic in Section 4.3. Our verification approach automatically transforms the problem of model checking TCTL into the problem of model checking CTL. Further, we prove that although TCTL and TCTL^C extend CTL, their model checking algorithms still have the same time complexity for explicit models and the same space complexity for concurrent programs (Section 4.4). In Section 4.5, we evaluate the effectiveness and efficiency of our approach by performing a set of experiments on a widely-used case study in business domain and compare our results with the results that have been obtained in chapter 3 [1].

[1]The results of this chapter are collected from our publication in [29]

4.1 An Overview of The Proposed Approach

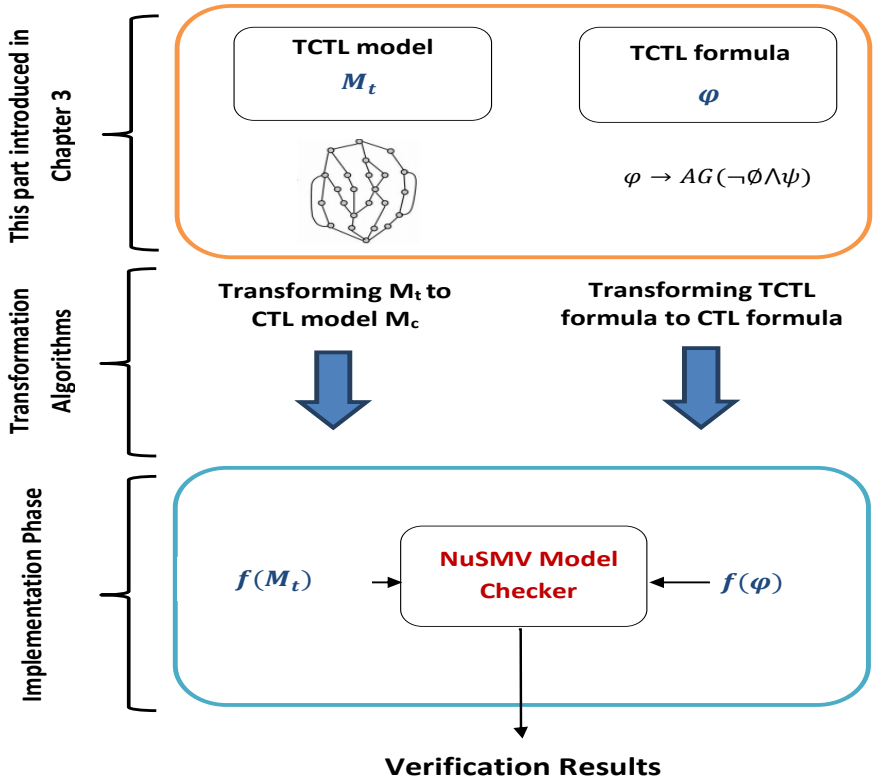


Figure 4.1: A schematic view of our TCTL model checking approach

Figure 4.1 illustrates the overall approach of model checking TCTL, which consists of three integrated phases. In the first phase, we recall the logic TCTL and its formal model defined using our formalism of vector-extended interpreted systems introduced in Chapter 3. In the second phase, we introduce our formal verification technique based on transforming the problem of model checking TCTL into the problem of model checking CTL along with the complexity analysis of the proposed technique. In the third phase, we implement our transformation technique in a Java toolkit that automatically interacts with the NuSMV model checker and report the verification results using a case study.

4.2 Conditional Trust TCTL^C

In [98], Singh propounds that trust must be conditional, meaning that trust should be expressed using antecedents and consequents. For example, a customer may trust a merchant as follows: "if I pay, then I trust the merchant will deliver the goods" [98]. Such a statement expresses the customer's expectation and the effect of this expectation on their future plans. Our preconditional trust modality that assumes the prior satisfaction of the precondition is different from conditional trust. Expressing conditional trust requires an extension of TCTL, and to distinguish the two languages, the extended one is called TCTL^C. However, there is a logical relationship between preconditional and conditional trust. In fact, as our main objective in this chapter is the verification of temporal trust, we will show how this logical relationship will be exploited to inaugurate a model checking procedure for conditional trust (see Section 4.3.3). The idea we aim to convey is that it is possible to decide if a given state, and thus a given model, satisfies a conditional trust formula by calling the model checking of TCTL. To show this, let us first introduce the syntax and semantics of conditional trust. From the syntax perspective, $T_c(i, j, \psi, \varphi)$ is read as "agent i trusts agent j about the consequent φ when the antecedent ψ holds". It is worth noticing that in the case of precondition trust, for the trust to take place between the interacting agents i and j, the condition $\psi \wedge \neg\varphi$ must be satisfied in the current state s_t to ensure that the precondition ψ holds before the trust content φ is brought about, while conditional trust requires the existence of at least one accessible state satisfying the antecedent ψ. This condition captures the intuition that the satisfaction of the antecedent is possible in some future. The semantics of $T_c(i, j, \psi, \varphi)$ is as follows:

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

The non satisfaction of the consequent φ complies with the first postulate in [98] stating that when the consequent holds, the trust in this consequent is "completed and is, therefore, no longer active". The following proposition shows the logical link between conditional and preconditional trust:

Proposition 4.1 (**Conditional and Preconditional Trust**). $T_c(i,j,\psi,\varphi) \wedge \psi \equiv T_p(i,j,\top,\psi \rightarrow \varphi) \wedge \neg T_p(i,j,\top,\neg\psi)$.

The proof of this proposition is direct from the semantics.

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

Proposition 4.2 (**Complexity of the Accessibility Relation**). *The accessibility relations of the model M_t can be computed in space $O(\log^2|Agt| + log^2|M_t|)$*

Proof. Computing the accessibility relation $s_t \rightsquigarrow_{i \rightarrow j} s'_t$ given two agents i and j requires computing the reachability from the state s_t. The reachability from s_t is a graph accessibility problem, and it is known by Jones [52] that the problem is in NLOGSPACE, so it can be done nondeterministically in space $O(log|M_t|)$, or, by Savitch's theorem [95], deterministically in space $O(log^2|M_t|)$. Since computing the reachability is independent from the agents, computing the accessibility for all the agents will require computing the reachability once, and compare the values of the local vectors for each pair of agents. The problem of comparing the values of all the pairs is in NLOGSPACE and can be solved nondeterministicaly in $O(\log|Agt|)$ by guessing a pair each time until all the pairs are covered

by transitivity. Thus, the problem can be solved deterministically in $O(\log^2 |Agt|)$, so the proposition. □

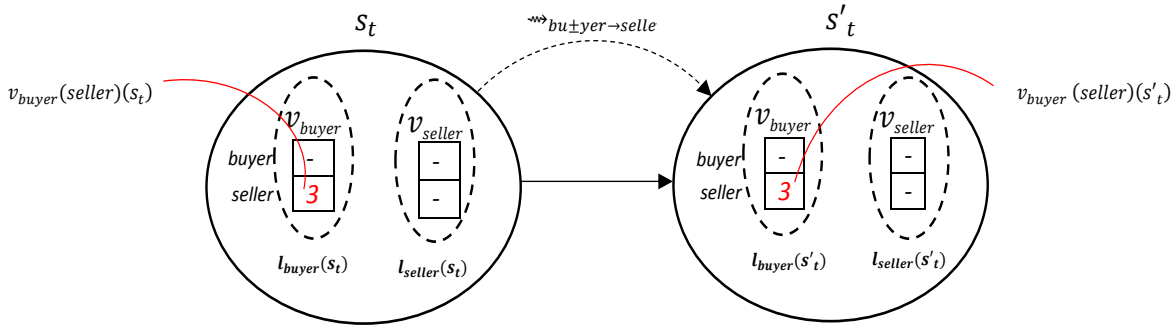


Figure 4.2: An example of trust accessibility relation $\rightsquigarrow_{i\rightarrow j}$

Figure 4.2 depicts an example of a trust accessibility relation between two states (s_t and s'_t). In this example, the solid line represents the transition relation from R_t, and the dashed line represents the direct trust accessibility relation $\rightsquigarrow_{i\rightarrow j}$ between such states. The state s'_t is compatible with s_t with regard to the trust the buyer agent has towards the seller agent. In the figure, we assign a vector to each agent's local states as follows: v_{buyer} and v_{seller} are the vectors of buyer and seller agents respectively. The buyer agent compares the element of her vector with regard to the seller agent at global states s_t and s'_t. The particular element value of the buyer agent is the same in both global states (i.e., $v_{buyer}(seller)(s_t) = v_{buyer}(seller)(s'_t) = 3$).

4.3 Formal Transformation to Model Check TCTL and Conditional Trust

In this section, we first introduce a transformation-based approach to address the problem of model checking TCTL. In a nutshell, given a model M_t representing a trust based MAS and a TCTL formula φ that describes the property that the model M_t has to satisfy, the problem of model checking TCTL can be defined as verifying whether or not φ holds in M_t, which is formally denoted by $M_t \models \varphi$. In particular, we apply specific reduction rules to formally transform the problem of model checking TCTL into the problem of model checking CTL [43]. This provides a way to perform our implementation on NuSMV. Technically, our transformation method encompasses two stages. First, we apply a set of formal rules to transform vector-extended transition systems into Kripke structures. Then, we transform TCTL formulae to CTL ones based on certain rules developed specifically for this purpose. Such a transformation is performed by developing two formal methods that provide accurate alignments between source and target models, and at the same time preserve TCTL semantics without losing the validity of the original model properties. This transformation is then extended to check conditional trust.

4.3.1 Transformation of TCTL Model

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

Definition 4.1. Model of CTL

A CTL formula is interpreted over a Kripke Structure $M_c = (S_c, R_c, I_c, V_c)$, where:

- S_c is a non-empty set of states for the system;

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

Having presented the CTL model, the next step is to establish our transformation technique. Given a TCTL model $M_t = (S_t, R_t, I_t, \{\rightsquigarrow_{i \rightarrow j} | (i,j) \in Agt^2\}, V_t)$, Algorithm 5 shows how this model is transformed into a CTL model $M_c = (S_c, R_c, I_c, V_c)$. The algorithm takes as input a model M_t (line 1) and outputs the transformed model M_c (line 2). First, the corresponding model M_c has the same set of system states and initial states (i.e., $S_c = S_t$; $I_c = I_t$). Thereafter, the algorithm initializes the set R_c, and then the set $V_c(s)$ to be equal to the set $V_t(s)$ (i.e., at the beginning, states are labeled with the same atomic propositions). We define a new set of atomic propositions needed to represent the trust accessibility relation to capture the semantics of trust as follows $X = \{\alpha^{ij} | (i,j) \in Agt^2\}$. Moreover, we define a new fresh atomic proposition χ that will be used to preserve the actual temporal transition relation. Thus, the set AP_c is as follows: $AP_c = X \cup AP_t \cup \{\chi\}$. The algorithm proceeds to transform transition and trust accessibility relations to constitute the transition relations in R_c based on two conditions. The first condition checks if the states s_t and s'_t have a transition relation in R_t, then this relation becomes a transition relation in R_c (lines 8 & 9). For the second condition, it checks if the current state s_t has an accessible state s'_t using the accessibility relation $\rightsquigarrow_{i \rightarrow j}$ for any truster-trustee pair of agents and this state is different from the state itself, moreover, if the two states are not in R_c, then a new state s''_t is added to the set of system states S_c along with a new transition from the corresponding s_t to the corresponding s''_t and from s''_t to s'_t in R_c. Further, the new state s''_t is labeled with the atomic propositions α^{ij} and χ in order to distinguish the states that are accessible from any other next state that satisfies the trust formulae without having accessibility to the

current state (line 14 & 15). However, if s_t'' is already added for some other accessibility relations, we only add the atomic proposition α^{ij} to mark the accessible state for any other interacting agents (lines 11 & 12). Finally, the algorithm returns the transformed model M_c after iterating over all the transitions.

Algorithm 5 Transform $M_t = (S_t, R_t, I_t, \{\rightsquigarrow_{i\to j} | (i,j) \in Agt^2\}, V_t)$ into $M_c = (S_c, I_c, R_c, V_c)$

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

Proposition 4.3 (Boundedness of Model Transformation). *$|M_c| \leq 3|M_t|^2$ where $|M_t|$ (resp. $|M_c|$) is the size of the input model M_t (resp. the output model M_c).*

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

4.3.2 Transformation of TCTL Formulae

This section presents our method to formally transform any TCTL formula φ to a CTL formula $f(\varphi)$ using a recursive transformation function f. The details of this method are illustrated in Algorithm 6. The transformation of the CTL fragment of TCTL is straightforward (lines 1-3). Yet, for the temporal operators (lines 4-6), we need to make sure that the transformation does not affect the CTL semantics. That is, since a new state and new transitions are added to the corresponding model M_c, we have to make sure that the path through which a formula is satisfied in the original model M_t is still satisfied in the corresponding path of the translated model M_c. Indeed, this is the main reason behind the conjunction of $\neg\chi$ for the temporal operators. This allows us to exclude the additional state and transitions when we consider the satisfaction of the formulae. For instance, the formula $EX\varphi$ is transformed into a CTL formula stating that there exist a path in the next state where the transformation of φ and the negation of the atomic proposition χ (added to represent the temporal transition) is true in this state. For the trust modality (line 7), the trust formula is transformed inductively into CTL according to the defined semantics as follows: the transformation of the formula $\psi \wedge \neg\varphi$ should hold in the current state, there exists a path where next state satisfies the added atomic proposition α^{ij}, which captures the existence of an accessible state, and along each path, if the next state on that path satisfies the atomic proposition α^{ij}, then the next state of the added state also satisfies the transformation of the trust content φ.

Figure 4.3 depicts an example illustrating the transformation of a TCTL model and some formulae. On the left side of the figure (part a), the model M_t consists of four global states s_0, s_1, s_2, and s_3. The states s_2 and s_3 are accessible from s_0. Furthermore, the trust formula $(T_p(i,j,\psi,\varphi))$ holds in s_0 (i.e., $(M_t,s_0) \models T_p(i,j,\psi,\varphi)$). According to the semantics, we obtain $(M_t,s_0) \models \psi \wedge \neg\varphi$, and there exists a state s' such that $s' \neq s$ and $s \rightsquigarrow_{i \to j} s'$

Algorithm 6 Transform TCTL formula φ into CTL formula $f(\varphi)$

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

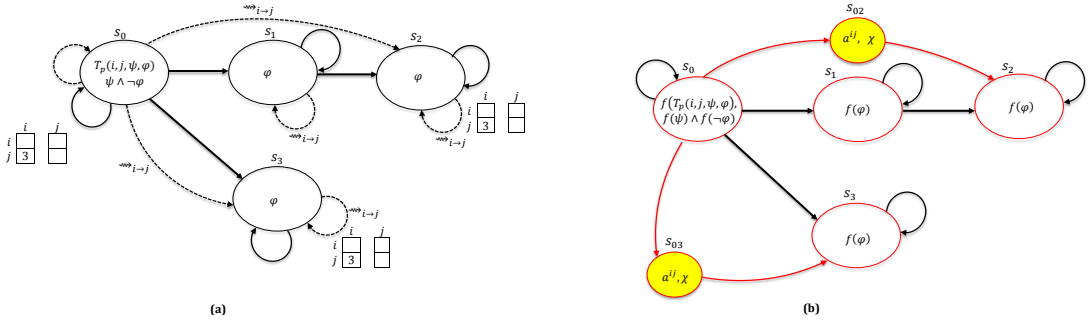


Figure 4.3: Example of the transformation methods

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

trust content φ hold as well for all next states.

Proposition 4.4 (**Boundedness of Formula Transformation**). *Let φ be a TCTL formula and f the transformation function defined in Algorithm 6. There exists a constant k such that $|f(\varphi)| < k|\varphi|$.*

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

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

□

Theorem 4.1 (**Soundness and Completeness of the Transformation**). *Let M_t and φ be respectively a TCTL model and formula and let M_C and $f(\varphi)$ be the corresponding model*

and formula in CTL. We have $(M_t, s_t) \models \varphi$ *iff* $(M_c, s_c) \models f(\varphi)$*, where* s_c *is the corresponding state of* s_t *in* M_c*.*

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

- For the formula $\phi = EX\varphi$, we have $(M_t, s_t) \models EX\varphi$ iff there exists an immediate successor to a state where φ holds. Consequently, from the definition of $f(M_t)$ and $f(\varphi)$, we obtain $(M_t, s_t) \models EX\varphi$ iff $(M_c, s_c) \models EX(f(\varphi) \wedge \neg\chi)$. That is, we are excluding the new added path as this path will never be considered because next state (the added state) has χ and we are forcing $\neg\chi$.

- For the formula $\phi = E(\varphi \cup \psi)$, and from the definition of f, $(f(\varphi) \wedge \neg\chi) \cup (f(\psi) \wedge \neg\chi)$ captures the semantics of Until in CTL which states the existence of a path starting in the current state that satisfies $(\varphi \wedge \neg\chi)$ until reaching a state in which $(\psi \wedge \neg\chi)$ holds.

- The trust formula: $T_p(i, j, \psi, \varphi)$. The first and second parts: $(f(\psi) \wedge f(\neg\varphi))$ capture the first condition of the semantics where the current state should satisfy $\psi \wedge \neg\varphi$. The third part $(EX(\alpha^{ij}))$ captures the second condition, i.e., the existence of an accessible state different from the current state since α^{ij} holds only in such accessible states. Finally, the fourth part $(AX(\alpha^{ij} \rightarrow AXf(\varphi)))$ captures the last condition in the semantics of the trust formula where all accessible states (those satisfying α^{ij} in M_c) should satisfy φ.

□

4.3.3 Model Checking Conditional Trust

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

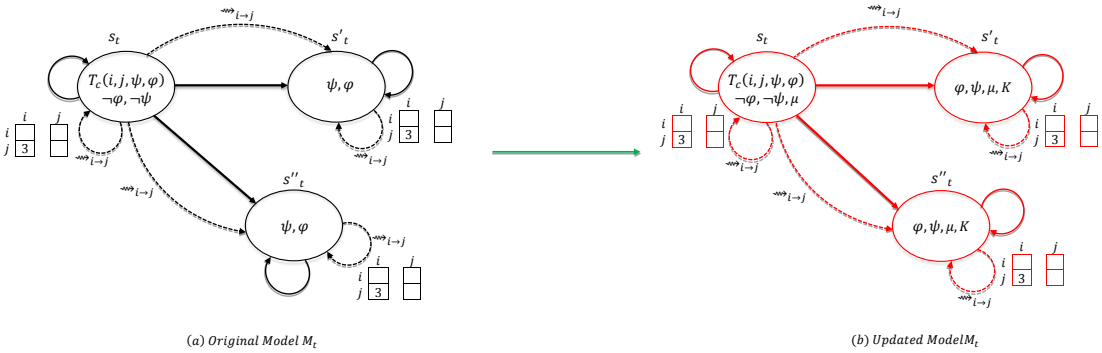
$$f(T_c(i,j,\psi,\varphi)) = f(\neg\varphi) \wedge EX(\alpha^{ij}) \wedge AX(\alpha^{ij} \rightarrow AX(f(\psi) \rightarrow f(\varphi)));$$

In this section, we introduce an alternative solution that uses the developed model checking algorithm of TCTL. The algorithm of model checking conditional trust (Algorithm 7) capitalizes on the equivalence shown in Proposition 4.1 (line 3). If $T_p(i,j,\top,\psi \rightarrow \varphi) \wedge \neg T_p(i,j,\top,\neg\psi)$ does not hold, then updating the evaluation function will be needed. Such an update works in all cases, but to be more efficient, the algorithm uses it only if the direct condition (line 3) fails. The update, which also exploits Proposition 4.1, introduces two fresh atomic propositions: μ and κ. μ holds in the current state (line 4) and in every state where ψ holds (line 7). Thus, if $T_p(i,j,\top,\neg\mu)$ does not hold, which means $\neg T_p(i,j,\top,\neg\mu)$ holds (condition 1), then the only reason is because there is an accessible state different from the current state where ψ holds. This is because the first condition in the semantics of $T_p(i,j,\top,\neg\mu)$ already holds since μ holds in s_t (line 4). κ does not hold in the current state, but holds in every state where $\psi \rightarrow \varphi$ holds. Consequently, if $T_p(i,j,\neg\varphi,\kappa)$ holds (condition 2), then $\neg\varphi$ holds and all accessible states different from the current state satisfy $\psi \rightarrow \varphi$. The algorithm returns true if the two conditions 1 and 2, which correspond to the semantics of $T_c(i,j,\psi,\varphi)$, hold (line 9), false otherwise (line 10). Figure 4.4 depicts an example illustrating Algorithm 7. In this example, the condition of line 3 does not hold since ψ does not hold in s_t. Thus, the model update is required as illustrated in the part (b). The condition of line 9 holds in the updated model, making the conditional trust formula $T_c(i,j,\psi,\varphi)$ true. The following theorem is direct from the semantics of conditional trust and Proposition 4.1.

Theorem 4.2 (**Soundness and Completeness of Algorithm 7**). *Algorithm 7 returns true iff* $(M_t,s_t) \models T_c(i,j,\psi,\varphi)$.

Algorithm 7 Model Check $T_c(i, j, \psi, \varphi)$

Input: M_t, s_t, i, j, ψ, φ
Output: **true** if $(M_t, s_t) \models T_c(i, j, \psi, \varphi)$; **false** otherwise
if $(M_t, s_t) \models T_p(i, j, \top, \psi \rightarrow \varphi) \wedge \neg T_p(i, j, \top, \neg\psi)$ **then return true**;
$V_t(s_t) := V_t(s_t) \cup \{\mu\}$;
for all $s'_t \neq s_t$
    if $(M_t, s'_t) \models \psi \rightarrow \varphi$ **then** $V_t(s'_t) := V_t(s'_t) \cup \{\kappa\}$;
    if $(M_t, s'_t) \models \psi$ **then** $V_t(s'_t) := V_t(s'_t) \cup \{\mu\}$;
end for
if $(M_t, s_t) \models T_p(i, j, \neg\varphi, \kappa) \wedge \neg T_p(i, j, \top, \neg\mu)$ **then return true**;
return false;



(a) Original Model M_t (b) Updated Model M_t

Figure 4.4: Illustrative example of Algorithm 7

4.4 Complexity Analysis

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

As our approach is transformation-based, we start by analyzing the time complexity of transforming the TCTL model and formula with respect to explicit models, where all

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

Theorem 4.3 (**Explicit Model Checking TCTL: Upper Bound**). *The TCTL model checking problem can be solved in time $O(|M_t|^2 \times |\varphi|)$ where $|M_t|$ and $|\varphi|$ are the size of the vector-extended model and length of the TCTL formula, respectively.*

Proof. TCTL extends CTL, and it is known from [21] that CTL model checking can be done in a linear time with respect to the size of the CTL model and formula, i.e., $O(|f(M_t)| \times |f(\varphi)|)$. From Proposition 4.3, $|f(M_t)| \leq 3|M_t|^2$, i.e., the size of $f(M_t)$ is polynomial with the size of M_t. Moreover, from Proposition 4.4, the length of $f(\varphi)$ is linear with the length of φ. Indeed, Algorithm 6 takes the TCTL formula φ as input and writes in a recursion manner the corresponding CTL formula according to the structure of φ. The time complexity of transforming the TCTL formula is linear with respect to the length of the input formula φ. This follows from the fact that (1) the length of the recursion is bounded

by the size of the input formula φ, and (2) the size of $f(\varphi)$ is bounded by the size of φ, so the theorem. □

Corollary 4.1 (**Explicit Model Checking Conditional Trust: Upper Bound**). *Model checking conditional trust can be solved in time* $O(|M_t|^2 \times |\varphi|)$. *where* $|M_t|$ *and* $|\varphi|$ *are the size of the vector-extended model and length of the conditional trust formula, respectively.*

Proof. The result is straightforward for the first method (transformation method) as it is similar to the model checking of TCTL. For the second solution, Algorithm 7 is simply calling the model checking of TCTL with an additional update of the model. The result follows from the fact that the update does not affect the asymptotic size of the model or the formula, but only adds atomic propositions to some states. □

Theorem 4.4 (**Explicit Model Checking TCTL: Completeness**). *The problem of TCTL model checking is P-complete.*

Proof. Membership (i.e., upper bound) in P follows from Theorem 4.3. Hardness (i.e., lower bound) in P is a result of the polynomial reduction from model checking CTL proved to be P-complete in [96]. □

The following corollary is direct from Theorem 4.4 and Corollary 4.1:

Corollary 4.2 (**Explicit Model Checking Conditional Trust: Completeness**). *The problem of model checking conditional trust is P-complete.*

4.4.1 Space Complexity

In this subsection, we will prove that the complexity of TCTL model checking for concurrent programs is PSPACE-complete and so is TCTL^C model checking. This result means

that there is an algorithm solving the problem in polynomial space in the size of the components constituting concurrent programs and the length of the formula being model checked.

Concurrent Programs

A concurrent program P as introduced in [61], is composed of n concurrent processes P_i (modules, protocols, or agents). Each process is described by a transition system D_i defined as follows: $D_i = (AP_i, AC_i, S_i, \Delta_i, s_i^0, L_i)$, where AP_i is a set of local atomic propositions, AC_i is a local action alphabet, S_i is a finite set of local states, $\Delta_i \subseteq S_i X AC_i X S_i$ is a local transition relation, $s_i^0 \in S_i$ is an initial state, and $L_i : S_i \rightarrow 2^{AP_i}$ is a local state labeling function.

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

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

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

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

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

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

2. for all $1 \leq i \leq n$ such that $a \notin AC_i$ we have $s[i] = s'[i]$

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

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

Theorem 4.5 (**Polynomial Reduction of Model Checking TCTL: Upper Bound**). *Let* $\sqsubseteq_{psr}$*denote the polynomial-space reduction. The problem of model checking TCTL can be reduced into the problem of model checking CTL in a polynomial space, i.e., MC(TCTL)* $\sqsubseteq_{psr}$ *MC(CTL).*

Proof. The transformation of the TCTL model and TCTL formula into the corresponding CTL model and formula could be computed by a deterministic Turing Machine (TM) in space $O(\log n)$ where n is the size of the input TCTL model, and polynomial space w.r.t. the size of the TCTL formula. For the model, TM reads in the input tape a model of TCTL and produces in the output tape, one-by-one, the same states including the initial ones and the same valuations. Then, for the transitions (s_t, s_t') in the input model, it writes one-by-one, the transitions in the set R_c. Moreover, it reads the accessibility relations $\rightsquigarrow_{i \to j}$ between two given states in the input model one-by-one and for each one, it adds an intermediate state to the set S_c labeled with two fresh atomic propositions: 1) α^{ij} that depends on the accessibility relation, and 2) χ, along with two transitions if such a state does not already exist; otherwise, only the atomic proposition α^{ij} is added. All these writing operations are clearly logarithmic in space because this transformation is done on-the-fly, step-by-step. Moreover, we showed in Proposition 4.4 that any TCTL formula is transformable into a CTL formula whose size is linearly bounded by the size of the input formula. All these recursive transformations are clearly polynomial space in the length of the input formula, so the theorem. □

Theorem 4.6 (**Model Checking TCTL for Concurrent Programs: Completeness**). *The space complexity of the TCTL model checking for concurrent programs is PSPACE-complete with respect to the size of the components of these programs and the length of the formula.*

Proof. Since model checking CTL is PSPACE-complete for concurrent programs [96], the lower bound of model checking TCTL is PSAPCE as well. In fact, TCTL subsumes CTL

as it integrates the CTL modalities and the trust modality. The upper bound in PSPACE follows from Theorem 4.5, so the result. □

Corollary 4.3 (**Model Checking Conditional Trust for Concurrent Programs: Completeness**). *The space complexity of model checking conditional trust for concurrent programs is PSPACE-complete with respect to the size of the components of these programs and the length of the formula.*

Proof. The corollary is direct from Algorithm 7 and Theorem 4.6 since adding atomic propositions to particular states and calling TCTL model checking with formulae having linear size with the size of the input formula are not source of additional space complexity. □

4.5 Implementation and Experimental Results

4.5.1 Insurance Claim Processing: A Case Study

To illustrate and implement our approach, we use a standard industrial case study [103]. The case study outlines the process by which auto insurance claims are handled by an insurance company, AGFIL. There are multiple parties involved in the AGFIL cooperation process: *AGFIL*, *Policyholder*, *Europ Assist*, *Lee Consulting Services*, *Garage*, and *Assessor*. The participating parties work together to provide a trusted service which facilitates efficient claim settlement. The process starts when the policyholder phones the call center Europ Assist to notify a new claim. Thereafter, Europ Assist registers the information and assigns an appropriate garage to provide the repair service to the policyholder. It then notifies the insurance company AGFIL which checks whether the policy is valid or not, and it confirms the claim coverage. AGFIL then sends the claim details to Lee Consulting Services (Lee CS) which is responsible for managing the operation of this service. Lee CS normally

appoints an assessor to conduct a physical inspection of damaged vehicle and checks vehicle repair estimates with the garage. When repairs are completed, the garage will issue an invoice to Lee CS which will check the invoice against the original estimate. Lee CS sends all invoices to AGFIL, which in turn finalizes the payment processes.

This scenario has been formalized, modeled, and verified in terms of commitments for instance by [28, 9, 56] where the contractual business relationships among the interacting parties are clearly identified, and also in terms of trust in [98]. Our modeling of trust is based on the assumption that the trust relationship among the parties involved influences their decisions without any contractual relationships (commitments) among them.

4.5.2 Implementation

To carry out the experimental process as efficiently as possible, we have developed an open source java toolkit [2] with a high-degree of automation that interacts automatically with the NuSMV model checker. Our toolkit essentially consists of two main modules implementing the proposed transformation algorithms (Algorithms 5,6, and 7). The main feature of our tool is its ability to automatically transform TCTL models and formulae into the corresponding CTL models and formulae. Technically, the NuSMV model checker is used as a core component by the toolkit engine in order to perform the verification aspects. Our tool takes as input the encoding of the MAS model, analyzes the code with respect to a set of formulae, and generates the required SMV modules according to the number of interacting agents. Besides, we implemented lexer and parser that check the syntax of both TCTL and TCTL^C logics.

In our encoding, we formalized the above MAS business scenario using our trust model M_t associated with the formalism of vector-extended interpreted systems introduced

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

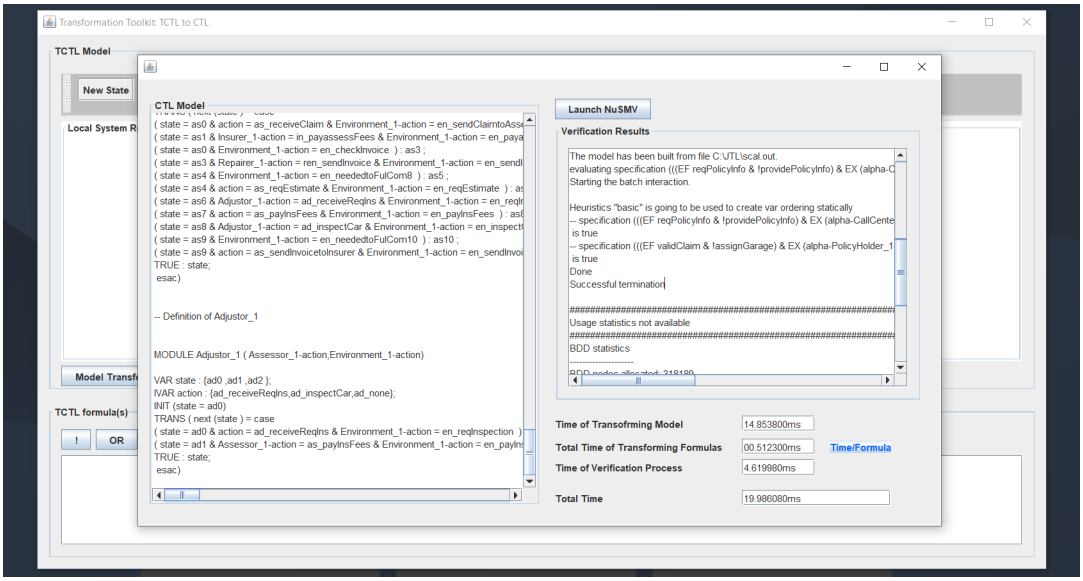


Figure 4.5: Screenshot of the generated NuSMV modules and the verification results

earlier in Section 4.3 to formally model the protocol. Thus, we have six interacting agents: *Ins* playing the role of AGFIL, PolicyHolder, CallCenter, *Repairer* playing the role of Garage, *Assessor* playing the role of Lee CS, and Adjustor plus the environment agent that facilitates the communication among these agents. More precisely, we encoded each agent in our VISPL input language of the MCMAS-T model checker introduced in [32] as a set of local states and actions, the local protocol, the local evolution function, and the initial states for each agent. We also considered the accessibility relations between agents by encoding the vector variables, which give a particular agent the possibility to establish the trust towards other agents. We validate our modeling using the capability in the MCMAS-T called `Explicit interactive mode` which is running the system interactively to check that it functions as intended. Thereafter, we used our transformation tool to transform the VISPL encoding model and trust formulae into the SMV model and CTL formulae in order to be able to start the verification process using the NuSMV model checker. Figure 4.5 shows the transformation process of the model and formula using our toolkit. Our toolkit has the capability to scale MASs (`scalability` button) with respect to a certain

modeling interleaved technique. In this technique, each agent is paired with another agent and all the resulting pairs move in a parallel way. Notice from the figure that the left panel displays the generated NuSMV modules of the MAS business scenario, which indeed is a CTL model. The `Launch NuSMV` button runs the NuSMV tool in order to display the verification results in the right panel of the figure. Furthermore, `Time/Formula` button pops up an information dialog box to show the transformation time of each formula (Figure 4.6).

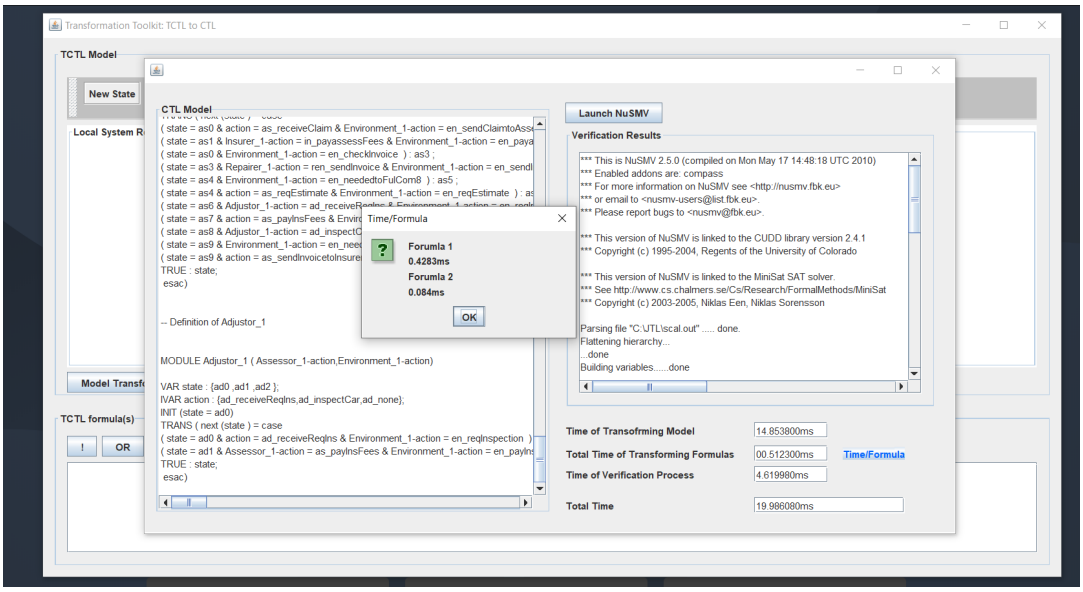


Figure 4.6: Screenshot of the information dialog box that shows the transformation time of each formula

4.5.3 Properties

In the above scenario, the participating parties have to ensure that the trust relationships are correctly established on one another to perform their tasks accordingly. To verify the correctness of the AGFIL scenario at design time, we have to express a set of properties. We used the safety (something bad will never happen) and liveness (something good will eventually occur) properties expressed using our logic. Such important properties have been

widely investigated in different contexts, see for instance [32, 37, 54]. Formally, the safety property φ_1 expresses the negation of the bad situation where the insurance company validates the policyholder claim, but the latter never establishes their trust towards the repairer with regard to the vehicle repair.

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

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

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

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

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

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

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

The formula φ_4 expresses the trust relationship between the policy holder and the call-center, so whenever the former reports an accident, the latter will eventually gather information, and the formula φ_5 states that whenever the garage repairs car, then there exist a path in its future the trust towards the insurance will take place with regards to the payment. Indeed, we have verified the above properties in a parametric way in different models having different numbers of agents ranging from 7 to 63. For example, the parametric form of Experiment 9 is generated with the conjunction operator as follows:

$$\varphi_1 = \bigwedge_{i=1}^{n} AG[\neg(validClaim \wedge \neg T_p(policyholder, Garage, validClaim, carRepair))].$$

Where the number of agents is 63 agents. The results will be presented and discussed in the next section.

4.5.4 Experimental Results

The experiments are conducted on AMD FX-8350 - 8 Cores - 4GHZ per core with 32 GB memory. To test the scalability of our algorithms, we report nine experiments in Table 4.1 for both preconditional and conditional algorithms. We developed a code generation script that helps us automatically encode different number of agents. We consider the number of agents (Agents#), the number of reachable states (States#), the transformation times of both models and formulae in milliseconds, and the average total time calculated based on the transformation and verification times. The experiments revealed that all the tested formulae are satisfied in our models. As shown in the table, the number of reachable states reflects the fact that the state space increases exponentially when the number of agent increases. Yet, it is clear that the transformation times of both the models and formulae increase only logarithmically with regard to the number of states. We can also notice that the average total time increases polynomially with respect to the number of states. Finally, it is worth mentioning that the exponential blow-up of the state space with the number of agents is a classic

state explosion problem in MASs and is independent of our model checking algorithms.

Table 4.1: Verification results of the AGFIL protocol using our toolkit

Exp.#	Agents#	States#	Time of model transformation (ms)	Time of formulae transformation (ms)	Total time (ms)
1	7	42	15	0.8	20
2	14	468	17	0.9	119
3	21	5586	22	1	1330
4	28	67236	36	1.1	8049
5	35	809682	38	1.2	45051
6	42	9.74111e+06	42	1.3	210000
7	49	1.29742e+08	48	1.4	390000
8	56	4.49064e+12	53	1.5	540000
9	63	2.52442e+15	57	1.7	792000

To compare our approach with the model checking approach introduced in [32], we run the same experiments of the AGFIL scenario using the MCMAS-T model checker with a different number of agents against the same properties. The experiments revealed that all the tested formulae are satisfied in the models. Table 4.2 reports the comparison of the results using the same machine. We can observe that the number of reachable states increases exponentially with the number of agents as we expected. It is clear that our transformation-based approach provides better results. An important problem encountered when running the models with MCMAS-T was that the experiments go down to almost a halt when we increase the number of agents. That is, while MCMAS-T allowed us to verify models up to only 42 agents, this approach is able to check the same scenario with up to 63 agents. In fact, the performance is higher as we go further and reach more agents. Moreover, the execution time in our approach is also better than the direct approach. We can observe that in our approach, this metric increases only logarithmically with respect to the number of states, and remains below 790 seconds even for 63 agents. As compared with our previous verification technique [32], it is wroth mentioning that the verification results

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

Table 4.2: Comparison of the verification results between Java tool and MCMAS-T

Exp.#	Agents#	States#	Total time (ms)
1	7	42	20
2	14	468	119
3	21	5586	1330
4	28	67236	8049
5	35	809682	45051
6	42	9.74111e+06	210000
7	49	1.29742e+08	390000
8	56	4.49064e+12	540000
9	63	2.52442e+15	792000

(a) Using our transformation toolkit

Exp.#	Agents#	States#	Time (ms)
1	7	42	50
2	14	468	1020
3	21	5586	16340
4	28	67236	99723
5	35	809682	694035
6	42	9.74111e+06	3333680

(b) Using the MCMAS-T model checker

Figure. 4.7 compares the verification results of the tool implementing our transformation-based approach and the MCMAS-T model checker in the form of graph. We use the execution time as function of the number of agents. By observing the figure, we can readily conclude that our developed tool is faster than the tool introduced in Chapter 3.

4.6 Summary

In this chapter, we proposed a new model checking framework for the TCTL logic of preconditional trust that is extended to design a new algorithm to model check conditional trust in MASs. We designed transformation-based algorithms and implemented them in a Java toolkit that automatically interacts with the NuSMV model checker of the CTL logic. Our proposed technique is able to automatically transform the problem of model checking TCTL into the problem of model checking CTL. We also discussed the logical relationship between preconditional and conditional trust, which led to the model checking procedure of

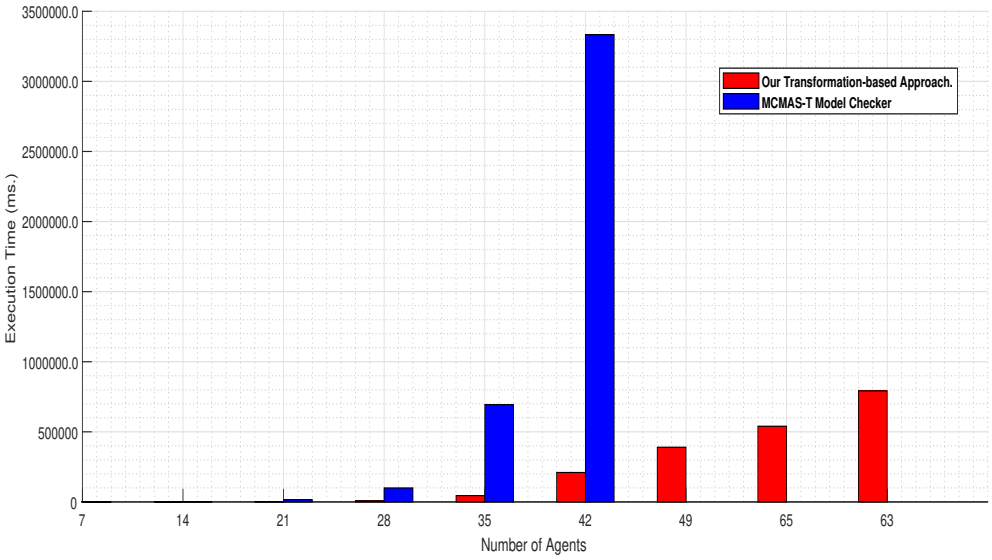


Figure 4.7: Comparison results between our transformation-based tool and MCMAS-T

conditional trust. The proof of the soundness and completeness of our transformation algorithms is provided. Moreover, we proved that (1) the time complexity of TCTL and TCTL^C model checking in explicit models is P-complete with regard to the size of the model and length of the formula; and (2) the complexity of the same problems for concurrent programs is PSPACE-complete with respect to the size of the program's components. Therefore, our model checking algorithms have the same complexity as model checking CTL with regard to both explicit models and concurrent programs. Experiments conducted on a standard industrial case study demonstrated the efficiency and scalability of the technique. When we compared our approach with the one proposed in Chapter 3, we reported that the developed verifier tool was able to verify a variety of formulae correctly and efficiently within a large case study having approximately 2.52442e+15 reachable states. Thanks to the high efficiency of CTL model checking to which the model checking of TCTL and conditional trust is transformed.

In the next chapter, we will address the quantitative aspects of trust from the logical and model checking perspectives.

Chapter 5

Degrees of Trust: Temporal Logic and Model Checking

Although plenty of qualitative logical frameworks have been proposed to evaluate and model trust in multi-agent sittings, these approaches generally ignore reasoning about quantitative aspects such as degrees of trust. In this chapter, we address this limitation from the modelling and verification perspectives. In Section 5.2, we start by constructing the Graded Trust Temporal Logic TCTL^G to reason about the qualitative aspect of trust and present a set of its reasoning postulates. Specifically, we extend TCTL by assigning a weight to the sets of states that satisfy the trust formula. This allows for computing along a run the ratio of states where the formula is satisfied. By doing so, degrees of trust would be obtained from the possible executions of the giving system. Moreover, in Section 5.3, we develop and implement a new symbolic model checking algorithm and open source tool for quantifying the relationships among the interacting agents. Finally, we investigate the complexity and evaluate our approach using a case study in the health care domain in Sections 5.4 and 5.5 [1].

[1]The results of this chapter are collected from our publication in [30]

5.1 An Overview of The Proposed Approach

Qualitative logical frameworks that handle trust in MASs have been widely analyzed in the literature [32, 87, 49, 83, 18]. Trust in these approaches has often been treated as either true or false, i.e., we either trust the behavior of an agent or not. However, such systems have also quantitative temporal properties (such as degrees of trust), which still need further attention from the logical and model checking perspectives. In fact, in many contexts, it is quite difficult to determine with absolute certainty whether a proposition about the behavior of an agent is true or false. For instance, I might trust the agent to a certain degree in relation to such a proposition (i.e., I may have only 50% of trust). That is, although qualitative logical formalisms allow us to reason about various classical properties, their expressiveness is limited in representing some important aspects that deal with the way of capturing our perception of reality [53].

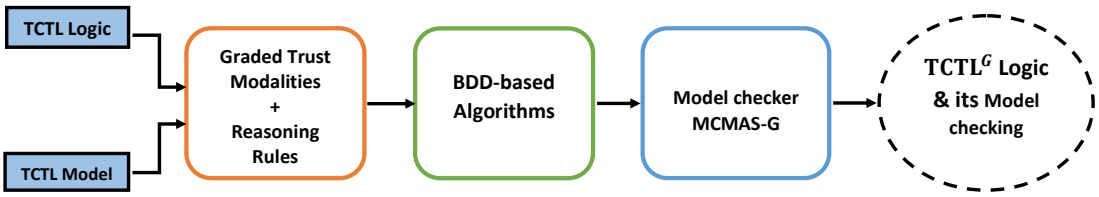


Figure 5.1: The main parts of the proposed approach

In fact, a standard approach of trust quantification involves the use of probability mechanisms accompanied with a representation of agent's beliefs [51, 78, 86]. However, it is worth noting that in this thesis we present a different approach that abstracts from the internal mental states and quantifies trust by relying only on accessibility relations inspired by the work proposed in [92]. However, unlike [92] that focuses on the degrees of beliefs by

extending the temporal-epistemic logic CTLK, our work mainly focuses on modeling and verifying trust by considering a different logic, along with the complexity of its symbolic model checking. Indeed, to the best of our knowledge and from the formal verification point of view, there is no model checking tool for verifying systems against graded trust specifications. Figure 5.1 illustrates our approach.

5.2 Graded Trust Temporal Logic

5.2.1 Syntax and Semantics

In this section, we present the syntax and semantics of TCTL^G.

Definition 5.1 (Syntax of TCTL^G)**.** The syntax of TCTL^G is defined recursively as follows:

$$\varphi ::= \rho \mid \neg\varphi \mid \varphi \vee \varphi \mid EX\varphi \mid E(\varphi\ U\ \varphi) \mid A(\varphi\ U\ \varphi) \mid T$$
$$T ::= T_p^{\Delta k}(i,j,\varphi,\varphi) \mid T_c^{\Delta k}(i,j,\varphi,\varphi)$$

where $\rho, E, A, X, \vee$, and U are defined in Definition 2.1 (Chapter 2). The trust operator T represents the trust relationship between two agents. There are two trust modalities: $T_p^{\Delta k}$ and $T_c^{\Delta k}$, that represent respectively preconditional and conditional graded trust. From the syntax perspective, $T_p^{\Delta k}(i,j,\psi,\varphi)$ expresses that "the truster i trusts the trustee j to bring about φ given that the precondition ψ holds with a degree of trust Δk", where k is a rational number in [0, 1], and Δ is a relation symbol in the set $\{\leq, \geq, <, >, =\}$. While the formula $T_c^{\Delta k}(i,j,\psi,\varphi)$ reads as "agent i trusts agent j about the consequent φ when the antecedent ψ holds with a degree of trust Δk". It is worth pointing that the advantage of representing the trustworthiness of an agent by a single real number format is that it is obvious for an agent to estimate her degrees of trust and to distinguish between certain agents in order to

choose the one satisfying their personal expectations. In fact, we can say that when $k = 0$, it means the trust has not been achieved, however, when $k = 1$, the trust has been perfectly fulfilled. Moreover, when the degree of trust $k = 1$, the standard trust operators $T_p(i, j, \psi, \varphi)$ and $T_c(i, j, \psi, \varphi)$ can be obtained as abbreviations:
$T_p(i, j, \psi, \varphi) \triangleq T_p^{\geq 1}(i, j, \psi, \varphi)$ and $T_c(i, j, \psi, \varphi) \triangleq T_c^{\geq 1}(i, j, \psi, \varphi)$.

Definition 5.2 (Semantics of TCTL^G)**.** The semantics of TCTL^G formulae is interpreted using a model (M_G) generated from Vector-based interpreted systems introduced in Section 3.3 above. Given the model M_G, the satisfaction of a TCTL^G formula φ in a global state s, denoted as $(M_G, s) \models \varphi$, is recursively defined as follows:

- $(M_G, s) \models \rho$ iff $\rho \in V_G(s)$;
- $(M_G, s) \models \neg\varphi$ iff $s \not\models \varphi$;
- $(M_G, s) \models \varphi_1 \vee \varphi_2$ iff $s \models \varphi_1$ or $s \models \varphi_2$;
- $(M_G, s) \models EX\varphi$ iff there exists a path π starting at s such that $\pi(1)) \models \varphi$;
- $(M_G, s) \models E(\varphi_1\ U\ \varphi_2)$ iff there exists a path π starting at s such that for some $k \geq 0$, $\pi(k)) \models \varphi_2$ and $\forall 0 \leq i < k$, $\pi(i)) \models \varphi_1$;
- $(M_G, s) \models A(\varphi_1\ U\ \varphi_2)$ iff for all paths π starting at s, there exists some $k \geq 0$ such that $\pi(k)) \models \varphi_2$ and $\forall 0 \leq i < k$, $\pi(i)) \models \varphi_1$;
- $(M_G, s) \models T_p^{\Delta k}(i, j, \psi, \varphi)$ iff $s \models \psi \wedge \neg\varphi$ and $\exists s' \neq s$ such that $s \rightsquigarrow_{i \to j} s'$, and $\frac{|s \rightsquigarrow_{i \to j} s' :\ s' \neq s\ \ \&\ \ s' \models \varphi|}{|s \rightsquigarrow_{i \to j} s' :\ s' \neq s|}\ \Delta k$;
- $(M_G, s) \models T_c^{\Delta k}(i, j, \psi, \varphi)$ iff $s \models \neg\varphi$ and $\exists s' \neq s$ such that $s \rightsquigarrow_{i \to j} s'$ and $s' \models \psi$, and $\frac{|s \rightsquigarrow_{i \to j} s' :\ s' \neq s\ \ \&\ \ s' \models \psi \Rightarrow \varphi|}{|s \rightsquigarrow_{i \to j} s' :\ s' \neq s|}\ \Delta k$.

For atomic propositions, Boolean connectives, and temporal modalities, the relation $\models$ is defined in the standard manner (see for example [21]). The intuition behind the semantics of $T_p^{\Delta k}(i,j,\psi,\varphi)$ and $T_c^{\Delta k}(i,j,\psi,\varphi)$ is: the degrees of trust that an agent associates to a formula φ in a global state s is the ratio between the number of states s' distinguishable and accessible from s and satisfying φ (i.e., $|s \rightsquigarrow_{i\to j} s' \colon s' \neq s \ \ \& \ \ s' \models \varphi|$), and the total number of distinguishable and accessible states from s (i.e., $|s \rightsquigarrow_{i\to j} s' \colon s' \neq s|$).

Example 5.1. *We now give examples of natural preconditional and conditional quantitative properties that can be expressed with $TCTL^G$. Let us consider a model for On-line Shopping System where atomic propositions include* `Deliver` *and* `Pay`*. Formula (5.1) specifies that it is not possible, with degree at least* 0.95*, for the buyer to trust the seller to deliver the requested items if the payment has not been made.*

$$\neg \ EF \quad T_c^{\geq 0.95}(buyer, seller, \neg payment, deliver) \tag{5.1}$$

Formula (5.2) states that the buyer trusts that the seller will deliver the requested items in 75% *of the cases under the condition that the latter has already received the payment.*

$$EF \quad T_p^{\geq 0.75}(buyer, seller, payment, deliver) \tag{5.2}$$

Figure 5.2 illustrates the model of Formula (5.2)*.*

5.2.2 Reasoning Postulates

We consider in this section several postulates that reflect common reasoning patterns that are valid in all $TCTL^G$ models. These postulates hold for both preconditional and conditional trust. Thus, we will use $T^{\Delta k}$ as a common operator instead of $T_p^{\Delta k}$ and $T_p^{\Delta k}$. Moreover,

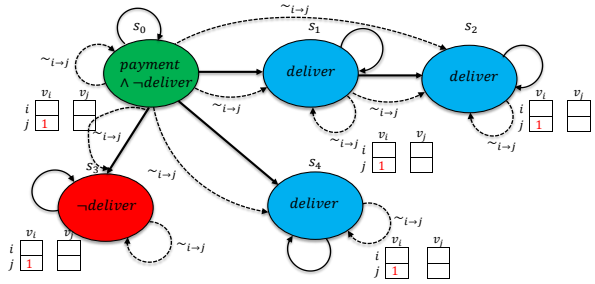


Figure 5.2: A model that satisfies the formula (5.2)

we omit i and j in the postulates as far as the truster and trustee are understood, so we simply write $T^{\Delta k}(\psi, \varphi)$.

1. $T^{\Delta k_1}(\psi, \varphi) \Rightarrow \nexists k_2 : T^{\Delta k_2}(\psi, \neg\varphi)$

2. $T^{\geq 1}(\psi_1, \varphi_1) \wedge T^{\Delta k}(\psi_2, \varphi_2) \Rightarrow T^{\Delta k}(\psi_1 \wedge \psi_2, \varphi_1 \wedge \varphi_2)$

   The following rules are instances of this postulate:

   - $T^{\geq 1}(\psi_1, \varphi_1) \wedge T^{\leq 0}(\psi_2, \varphi_2) \Rightarrow T^{\leq 0}(\psi_1 \wedge \psi_2, \varphi_1 \wedge \varphi_2)$
   - $T^{\geq 1}(\psi_1, \varphi_1) \wedge T^{\geq 1}(\psi_2, \varphi_2) \Rightarrow T^{\geq 1}(\psi_1 \wedge \psi_2, \varphi_1 \wedge \varphi_2)$

3. $T^{\leq k}(\psi, \varphi_1) \Rightarrow T^{\leq k}(\psi, \varphi_1 \wedge \varphi_2)$

   The following postulate (4) derives from postulate 3:

4. $T^{\leq k_1}(\psi_1, \varphi_1) \wedge T^{\leq k_2}(\psi_2, \varphi_2) \Rightarrow T^{\leq min(k_1,k_2)}(\psi_1 \wedge \psi_2, \varphi_1 \wedge \varphi_2)$

5. $T^{\leq k_1}(\psi_1, \varphi_1) \wedge T^{\Delta k_2}(\psi_2, \varphi_2) \Rightarrow T^{\leq max(k_1,k_2)}(\psi_1 \wedge \psi_2, \varphi_1 \wedge \varphi_2)$

6. $T^{\leq k_1}(\psi, \varphi_1 \wedge \varphi_2) \wedge \neg\varphi_1 \Rightarrow \exists k_2 \geq k_1 \quad s.t. \quad T^{\leq k_2}(\psi, \varphi_1)$

7. $T^{\geq k}(\psi, \varphi_1 \wedge \varphi_2) \wedge \neg\varphi_1 \Rightarrow T^{\geq k}(\psi, \varphi_1)$

8. $T^{\geq k_1}(\psi_1, \varphi_1) \wedge T^{\geq k_2}(\psi_2, \varphi_2) \Rightarrow T^{\geq max(k_1+k_2-1,0)}(\psi_1 \wedge \psi_2, \varphi_1 \wedge \varphi_2)$

9. $T^{\leq k}(\psi, \varphi_1 \vee \varphi_2) \Rightarrow T^{\leq k}(\psi, \varphi_1)$

10. $T^{\geq k}(\psi, \varphi_1) \wedge \neg\varphi_2 \Rightarrow T^{\geq k}(\psi, \varphi_1 \vee \varphi_2)$

11. $T^{\geq k_1}(\psi, \varphi_1) \vee T^{\geq k_2}(\psi, \varphi_2) \wedge \neg\ (\varphi_1 \vee \varphi_2) \Rightarrow T^{\geq min(k_1,k_2)}(\psi, \varphi_1 \vee \varphi_2)$

12. $T^{\geq k}(\psi, \varphi_1 \vee \varphi_2) \Rightarrow \exists k_1, k_2 \quad s.t. \quad T^{\geq k_1}(\psi, \varphi_1) \wedge T^{\geq k_2}(\psi, \varphi_2) \wedge k_1 + k_2 \geq k$

13. From $T^{\geq k}(\psi, \varphi_1)$ and $\varphi_1 \vdash \varphi_2$ and $\neg\varphi_2$ infer $T^{\geq k}(\psi, \varphi_2)$

14. From $T^{\leq k_1}(\psi, \varphi_1)$ and $\varphi_1 \vdash \varphi_2$ and $\neg\varphi_2$
    infer $\exists k_2 \geq k_1 \quad s.t. \quad T^{\leq k_2}(\psi, \varphi_2)$

5.3 Model Checking TCTL^G

Model checking is the problem of automatically establishing whether or not a formula is satisfied on a given model. In this section, we present an efficient algorithm for the TCTL^G model-checking problem. We start by presenting the main algorithm (**Algorithm** 8) that extends the standard symbolic model checking algorithm for CTL [21]. In this algorithm, we simply call the known procedures for the CTL modalities (i.e., SMC_{EX}, SMC_{EU}, and SMC_{AU}) for computing the set of states satisfying the corresponding modalities.

Algorithm 8 works as follows. First, it takes as input the model M_G and the TCTL^G formula Φ and returns the set $[[\Phi]]$ of states that satisfy Φ in M_G. By giving the model M_G, the algorithm recursively goes through the structure of Φ and constructs the set $[[\Phi]]$ with respect to a set of Boolean operations applied to sets. The lines 1 to 6 invoke the standard procedures used in CTL to compute the set of states that satisfy regular CTL formulae. The lines 7 and 8 call our procedures which compute the set of states that satisfy the graded trust formulae.

Algorithm 8 $SMC(\Phi, M_G)$: the set $[[\Phi]]$ of states satisfying the TCTL^G formula Φ

1: Φ is ρ: **return** $\{s \in S_G \mid \rho \in V_G(s)\}$;
2: Φ is $\neg\varphi$: **return** $S - SMC(\varphi, M_G)$;
3: Φ is $\varphi_1 \vee \varphi_2$: **return** $SMC(\varphi_1, M_G) \cup SMC(\varphi_2, M_G)$;
4: Φ is $EX\varphi$: **return** $SMC_{EX}(\varphi, M_G)$;
5: Φ is $E(\varphi_1\ U\ \varphi_2)$: **return** $SMC_{EU}(\varphi_1, \varphi_2, M_G)$;
6: Φ is $A(\varphi_1\ U\ \varphi_2)$: **return** $SMC_{AU}(\varphi_1, \varphi_2, M_G)$;
7: Φ is $T_p^{\Delta k}(i, j, \psi, \varphi)$: **return** $SMC_{T_p}(i, j, \psi, \varphi, \Delta k, M_G)$;
8: Φ is $T_c^{\Delta k}(i, j, \psi, \varphi)$: **return** $SMC_{T_c}(i, j, \psi, \varphi, \Delta k, M_G)$;

Algorithm 9 $SMC_{T_p}(i, j, \psi, \varphi, \Delta k, M_G)$

1: $\mathbf{Y} \leftarrow SMC(\neg\varphi, M_G)$;
2: $\mathbf{X1} \leftarrow SMC(\psi, M_G) \cap \mathbf{Y}$;
3: $\mathbf{X2} \leftarrow SMC(\varphi, M_G)$;
4: $\mathbf{Z} \leftarrow$ Compute the set of states in **X1** s.t. their number of accessible states that are in **X2** over the total number of their accessible states - 1 is Δk
5: **return** Z

5.3.1 BDD-based Algorithm of Graded Trust

This section introduces the model checking algorithms for both the $T_p^{\Delta k}$ and $T_c^{\Delta k}$ operators. Given a TCTL^G formula Φ and a TCTL^G model M_G over the vector-based interpreted system, the two algorithms compute the set of states of M_G in which Φ holds. **Algorithm** 9 describes the procedure $SMC_{T_p}(i, j, \psi, \phi, \Delta k, M_G)$. This procedure returns the set of states in which the preconditional graded 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 **X1** (the set of states satisfying $\psi \wedge \neg\varphi$). Thereafter, it assigns to the set **X2** the set of states where the formula φ holds. Thereafter, the algorithm proceeds to build the set **Z** by computing the set of states in **X1** such that their number of accessible states that are in **X2** over the total number of their accessible states mines 1 satisfy the appropriate relation Δk. Finally the procedure returns the set **Z** of the states that satisfy the formula $T_p^{\Delta k}(i, j, \psi, \varphi)$.

To compute the formula $T_c^{\Delta k}(i, j, \psi, \varphi)$, we follow the same steps in **Algorithm** 9,

except lines 2 and 3 which assign to the set **X1** the set of states satisfying $\neg\varphi$, and to the set **X2** the set of states satisfying $\psi \Rightarrow \varphi$. Indeed, this is based on our proposed semantics of conditional graded trust where the set of global states satisfying the formula $T_c^{\Delta k}(i, j, \psi, \varphi)$ in a given model M_G is computed by calculating and checking if the ratio between the number of states satisfying $\psi \Rightarrow \varphi$ over the total number of all states that can reach and see such states through the accessibility relation $\rightsquigarrow_{i \to j}$ satisfies the appropriate relation Δk.

5.4 Complexity Analysis

In this section, we will show that the complexity of model checking TCTL^G is PSPACE-complete for concurrent programs. Concurrent programs are composed of n concurrent agents, where each agent is described by a transition system. In these structures, states and transitions are not listed explicitly, but having instead compact representations that still correspond to the actual system. In fact, in symbolic model checking, "the Kripke structures to which model checking is applied are often obtained by constructing the reachability graph of concurrent programs" [60]. It is worth mentioning that the complexity analysis only considers the case of flat trust formulae where the content could be any formula but not a trust one.

To prove the PSPACE-completeness result, we introduce a 2-stage transformation procedure. In the first stage, we transform the problem of model checking TCTL^G into the problem of model checking ARCCTL$_{\div 1}$, a new logic that we define in this research. In the second stage, the problem of model checking this new logic is transformed into the one of model checking Action-Restricted CTL (ARCTL) proposed in [88]. We will introduce two transformation functions: f_1 for the first stage and f_2 for the second stage. Both functions include rules for transforming the model and formulae from a source language to a target one. Although a direct transformation from model checking TCTL^G into model checking

ARCTL is theoretically possible in one stage, the 2-stage procedure that uses an intermediate language, namely ARCCTL$_{\div 1}$, makes the transformation more natural and easy to follow from the methodological perspective.

ARCTL is an extension of CTL with action formulae. We use these actions to capture the accessibility relations in the original TCTL^G model. ARCCTL$_{\div 1}$ merges a fragment of the Counting CTL logic (CCTL) [64] with ARCTL. The reason of using ARCCTL$_{\div 1}$ as intermediate language for our transformation procedure is that CCTL counts the number of states satisfying certain sub-formulae along paths and uses this number as a constraint of the until temporal operator. This allows us to capture the number of accessible states used to define the semantics of TCTL^G. Finally, by merging a fragment of CCTL with ARCTL, we capture the accessibility relations in the first stage through action-labeled transitions and use these transitions in the second stage toward the target language ARCTL.

Before introducing ARCCTL$_{\div 1}$, we briefly review ARCTL [88].

Definition 5.3 (Syntax of ARCTL)**.**

$$\varphi ::= \rho \mid \neg\varphi \mid \varphi \vee \varphi \mid E_\alpha X\varphi \mid E_\alpha(\varphi\ U\ \varphi) \mid A_\alpha(\varphi\ U\ \varphi)$$

φ is a state formula and α is an atomic action formula ($\alpha \in AC_A$ the set of atomic actions). Instead of considering composed action formulae as in the original ARCTL logic, we only consider here atomic actions, which are enough to capture the labeled transitions. ARCTL restricts path quantifiers with an action formulae that must be satisfied along the path (i.e., labeling each transition of the path) in order to determine the precise paths over which path formulae are evaluated.

Definition 5.4 (Model of ARCTL)**.** The model of ARCTL is a tuple $M_A = (S_A, AC_A, I_A, R_A, V_A)$ where S_A is a nonempty set of states; AC_A is a set of actions; $I_A \subseteq S_A$ is a set of initial states;

$R_A \subseteq S_A \times AC_A \times S_A$ is a labeled transition relation; $V_A : S_A \to 2^{AP}$ is a function labeling states with subsets of atomic propositions AP.

A path of M_A is an infinite sequence of states and actions. $\Pi^{\alpha}(s)$ is the set of paths (called α-paths) starting at s and where all transitions are labeled with the atomic action α.

The satisfaction relation $(M_A, s) \models \varphi$ is given as follows (we omit the semantics of Boolean connectives and propositional atoms):

- $(M_A, s) \models E_{\alpha} X \varphi$ iff there exists a path $\pi \in \Pi^{\alpha}(s)$ and $(M_A, \pi(1)) \models \varphi$;
- $(M_A, s) \models E_{\alpha}(\varphi_1 \ U \ \varphi_2)$ iff there exists a path $\pi \in \Pi^{\alpha}(s)$ such that for some $k \geq 0, (M_A, \pi(k)) \models \varphi_2$ and $(M_A, \pi(j)) \models \varphi_1$ for all $0 \leq j < k-1$;
- $(M_A, s) \models A_{\alpha}(\varphi_1 \ U \ \varphi_2)$ iff for all paths $\pi \in \Pi^{\alpha}(s)$ there exists some $k \geq 0$, such that $(M_A, \pi(k)) \models \varphi_2$ and $(M_A, \pi(j)) \models \varphi_1$ for all $0 \leq j < k-1$.

Definition 5.5 (Syntax of ARCCTL$_{\div 1}$)**.**

$$\varphi ::= \rho \mid \neg\varphi \mid \varphi \vee \varphi \mid E_{\alpha} X \varphi \mid E_{\alpha}(\varphi \ U_{[c]} \ \varphi) \mid A_{\alpha}(\varphi \ U_{[c]} \ \varphi)$$
$$c ::= \frac{\#\varphi}{\tau} \ \Delta k;$$

α is an atomic action formula as in Definition 5.3, ΔK is as in Definition 5.1, and c is a constraint based on counting the number of states satisfying φ ($\#\varphi$) and τ is a strictly positive natural number.

We only use a fragment of CCTL where only one formula (φ) is counted in the constraint through the division operator instead of counting different formulae (φ_i) and going through the sum of their corresponding states as in the full CCTL logic. $\#\varphi$ captures the number of states satisfying φ along a given prefix and τ represents the total number of states of that prefix (the formal definition will follow).

ARCCTL$_{\div 1}$ uses the standard abbreviations. For instance: $E_\alpha F_{[c]}\varphi = E_\alpha(\top\ U_{[c]}\ \varphi)$, $A_\alpha F_{[c]}\varphi = A_\alpha(\top\ U_{[c]}\ \varphi)$, $A_\alpha G_{[c]}\varphi = \neg E_\alpha F_{[c]}\neg\varphi$. The size of a formula takes into account the size of the constraint formula. For instance $|E_\alpha(\varphi_1\ U_{[\frac{\#\varphi}{\tau}]}\ \varphi_2)| = |E_\alpha(\varphi_1\ U\ \varphi_2)| + |\varphi|$.

Definition 5.6 (Model of ARCCTL$_{\div 1}$)**.** ARCCTL$_{\div 1}$ is interpreted over a labeled Kripke strcture $M_\div = (S_\div, AC_\div, I_\div, R_\div, V_\div)$ where $S_\div$ is a nonempty set of states; $AC_\div$ is a set of actions; $I_\div$ is a set of initial states; $R_\div \subseteq S_\div \times AC_\div \times S_\div$ is a labeled transition relation; $V_\div : S_\div \to 2^{AP}$ is a valuation function.

The satisfaction relation $(M_\div, s) \models \varphi$ is given as follows (we omit the semantics of propositional atoms, Boolean connectives and the next operator):

- $(M_\div, s) \models E_\alpha(\varphi_1\ U_{[c]}\ \varphi_2)$ iff there exists a path $\pi \in \Pi^\alpha(s)$ such that for some $i \geq 0, (M_\div, \pi(i)) \models \varphi_2$ and $(M_\div, \pi_{(i-1)}) \models c$ and for all $0 \leq j < i$, $(M_\div, \pi(j)) \models \varphi_1$;
- $(M_\div, s) \models A_\alpha(\varphi_1\ U_{[c]}\ \varphi_2)$ iff for all paths $\pi \in \Pi^\alpha(s)$, there is some $i \geq 0, (M_\div, \pi(i)) \models \varphi_2$ and $(M_\div, \pi_{(i-1)}) \models c$ and for all $0 \leq j < i$, $(M_\div, \pi(j)) \models \varphi_1$.

where $\Pi^\alpha(s)$ is the set of α paths starting at s, $\pi(i)$ is the state s_i of the path π and $\pi_{(i)}$ is the prefix $s_0 \ldots s_i$ of π. Notice that $\pi_{(-1)} = \varepsilon$ is the empty prefix.

For every finite run prefix $\pi_{(i)} = s_0 \ldots s_i$, the meaning of $(M_\div, \pi_{(i)}) \models c$ is based on the interpretation of $\frac{\#\varphi}{\tau}$ over $\pi_{(i)}$, which is the number of states among $s_0 \ldots s_i$ satisfying φ over the size τ of the prefix $(\tau = i+1)$, or formally, $\frac{|\{j | 0 \leq j \leq i \wedge \pi(j) \models \varphi\}|}{i+1}$. Notice that $\tau = 1$ when the prefix is empty, which is also the case when the prefix has only one element. Under this semantics, $E_\alpha(\varphi_1\ U\ \varphi_2)$ is equivalent to $E_\alpha(\top\ U_{[\frac{\#\neg\varphi_1}{\tau}=0]}\ \varphi_2)$ and $A_\alpha(\varphi_1\ U\ \varphi_2)$ is equivalent to $A_\alpha(\top\ U_{[\frac{\#\neg\varphi_1}{\tau}=0]}\ \varphi_2)$.

We proceed now with the first transformation function f_1 (i.e., from TCTL^G to ARCCTL$_{\div 1}$). We assume that the states of the input TCTL^G model are ordered. We can use any arbitrary

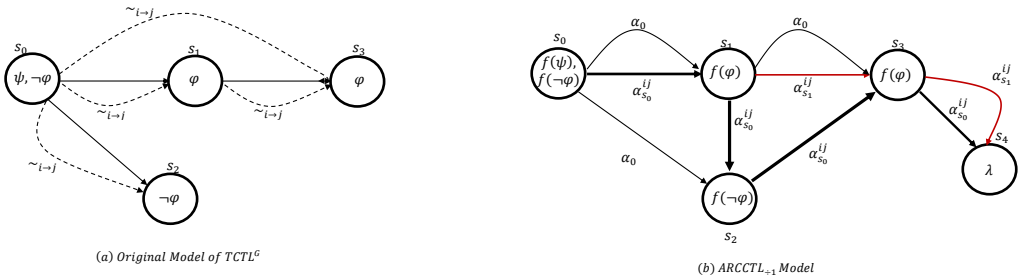


Figure 5.3: f_1 model transformation from TCTL^G to $\text{ARCCTL}_{\div 1}$

order, so we simply need to assume a particular one, for instance: $s_0, s_1, \ldots, s_m$. Assuming this order, we use the notation: $s \rightsquigarrow_{i\to j} (s_1, s_2 \ldots, s_m)$ to denote $s \rightsquigarrow_{i\to j} s_1, s \rightsquigarrow_{i\to j} s_2, \ldots, s \rightsquigarrow_{i\to j} s_m$.

The model transformation f_1 is as follows: $S_\div = S_G \cup \{s_{new}\}$; $I_\div = I_G$; Initialize $AC_\div$ with $\{\alpha_0\}$ and $\forall s$ s.t. $\exists s'$ where $s \rightsquigarrow_{i\to j} s'$ add α_s^{ij} to $AC_\div$; $\forall s \in S_G, V_\div(s) = V_G(s)$; $V_\div(s_{new}) = \{\lambda\}$; $\forall (s,s') \in R_G$ add (s, α_0, s') to $R_\div$; and if $s \rightsquigarrow_{i\to j} (s_1, s_2, \ldots, s_m)$ then add $\{(s, \alpha_s^{ij}, s_1), (s_1, \alpha_s^{ij}, s_2), \ldots, (s_{m-1}, \alpha_s^{ij}, s_m), (s_m, \alpha_s^{ij}, s_{new})\}$ to $R_\div$. Figure 5.3 illustrates a transformation example of this stage.

The formula transformation f_1 is defined recursively as follows:

- $f_1(M_G, s) \models f_1(\rho)$ **iff** $(M_\div, s) \models \rho$;
- $f_1(M_G, s) \models f_1(\neg\varphi)$ **iff** $(M_\div, s) \models \neg f_1(\varphi)$;
- $f_1(M_G, s) \models f_1(\varphi_1 \vee \varphi_2)$ **iff** $(M_\div, s) \models f_1(\varphi_1) \vee f_1(\varphi_2)$;
- $f_1(M_G, s) \models f_1(EX\varphi)$ **iff** $(M_\div, s) \models E_{\alpha_0} X f_1(\varphi)$;
- $f_1(M_G, s) \models f_1(E(\varphi_1\ U\ \varphi_2))$ **iff** $(M_\div, s) \models E_{\alpha_0}(f_1(\varphi_1)\ U\ f_1(\varphi_2))$;
- $f_1(M_G, s) \models f_1(A(\varphi_1\ U\ \varphi_2))$ **iff** $(M_\div, s) \models A_{\alpha_0}(f_1(\varphi_1)\ U\ f_1(\varphi_2))$;
- $f_1(M_G, s) \models f_1(T_p{}^{\Delta k}(i, j, \psi, \varphi))$ **iff** $(M_\div, s) \models f_1(\psi) \wedge f_1(\neg\varphi) \wedge E_{\alpha_s^{ij}} X E_{\alpha_s^{ij}} F_{[\frac{\#f_1(\varphi)}{\tau}\Delta k]}\ \lambda$;

- $f_1(M_G, s) \models f_1({T_c}^{\Delta k}(i, j, \psi, \varphi))$ **iff** $(M_{\div}, s) \models f_1(\neg\varphi) \wedge E_{\alpha_s^{ij}} F(f_1(\psi))$
  $\wedge \quad E_{\alpha_s^{ij}} X E_{\alpha_s^{ij}} F_{[\frac{\#f_1(\psi \Rightarrow \varphi)}{\tau}\Delta k]} \quad \lambda.$

All the cases are straightforward, except the trust operators ${T_p}^{\Delta k}$ and ${T_c}^{\Delta k}$ that make use of the semantics and exploit the constrained until operator of ARCCTL$_{\div 1}$ from the next state. This operator counts the number of states satisfying the content $f_1(\varphi)$ or the condition $f_1(\psi \Rightarrow \varphi)$ over the total number of states in the prefix starting from the next state to the last state preceding the state satisfying λ, which means the newly added state. This prefix corresponds to the accessible states from s, which are captured through the transitions labeled by α_s^{ij}. For the ${T_c}^{\Delta k}$ operator, ψ should be satisfied in one of the accessible states, which means in one of the future states through an α_s^{ij} path, i.e., $E_{\alpha_s^{ij}} F(f_1(\psi))$.

For the second stage transformation f_2 (i.e., from ARCCTL$_{\div 1}$ to ARCTL), the model transformation is straightforward; it is a simple mapping of each element of $M_{\div}$ to the corresponding element of M_A. The formula transformation is given in **Function** $f_2(\Phi$: ARCCTL$_{\div 1})$. To transform the constrained until operator over a certain path (**Function** Trans$_{EU}$), the function computes $\prod_{E(\varphi_1 U \varphi_2)}$, the set of prefixes of paths satisfying the formula $E(f_2(\varphi_1)\ U\ f_2(\varphi_2))$. If one of these prefixes satisfies the constraint in terms of the number of states $|S_{\varphi}^{U}|$ over the size of the prefix, the unconstrained until formula is returned. Otherwise, the formula cannot be satisfied, so false is returned instead. The case of the universal constrained until is similar (**Function** Trans$_{AU}$). The following theorem holds. It can be proved by induction over the structure of the formula φ.

Theorem 5.1 (**Soundness and Completeness of the Transformation**).

1. $(M_G, s) \models \varphi$ ***iff*** $(M_{\div}, s) \models f_1(\varphi)$;
2. $(M_{\div}, s) \models \varphi$ ***iff*** $(M_A, s) \models f_2(\varphi)$;

Function $f_2(\Phi$: ARCCTL$_{\div 1})$: ARCTL formula

1: **switch** (Φ):
2: **case** ρ : **return** ρ;
3: **case** $\neg\varphi$: **return** $\neg f_2(\varphi)$;
4: **case** $\varphi_1 \vee \varphi_2$: **return** $f_2(\varphi_1) \vee f_2(\varphi_2)$;
5: **case** $E_\alpha X \varphi$: **return** $E_\alpha X f_2(\varphi)$;
6: **case** $E_\alpha(\varphi_1 U_{[\frac{\#\varphi}{\tau}\Delta k]} \varphi_2)$: **return** Trans$_{EU}$ $(\varphi_1, \varphi_2, \varphi, \Delta k)$;
7: **case** $A_\alpha(\varphi_1 U_{[\frac{\#\varphi}{\tau}\Delta k]} \varphi_2)$: **return** Trans$_{AU}$ $(\varphi_1, \varphi_2, \varphi, \Delta k)$;
8: **end switch**

Function Trans$_{EU}$ $(\varphi_1, \varphi_2, \varphi, \Delta k)$: ARCLT formula

1: **if** $E_\alpha(f_2(\varphi_1)\ U\ f_2(\varphi_2)) = \bot$ **then return false**
2: **else**
3: Compute $\prod_{E(\varphi_1 U \varphi_2)}$
4: $S_\varphi := \{\varphi \in S_\div | \varphi \models f_2(\varphi)\}$
5: **for each** $P_{E(\varphi_1 U \varphi_2)} \in \prod_{E(\varphi_1 U \varphi_2)}$
6: $S_\varphi^U = S_\varphi \cap P_{E(\varphi_1 \vee \varphi_2)}$
7: **if** $\frac{|S_\varphi^U|}{|P_{E(\varphi_1\ U\ \varphi_2)}|}\Delta k$ **then return** $E_\alpha(f_2(\varphi_1)\ U\ f_2(\varphi_2))$
8: **end for**
9: **return false**

Proposition 5.1. *Let $MC(\mathscr{L})$ be the problem of model checking the language $\mathscr{L}$ for concurrent programs and $\preceq_p$ be the polynomial-space reduction. We have MC(ARCCTL$_{\div 1}$) $\preceq_p$ MC(ARCTL).*

Proof. Regarding the model, it is easy to see that any model of ARCCTL$_{\div 1}$ is also a model of ARCTL. So, we can easily imagine a deterministic Turing machine TM that can compute the model reduction in space $O(\log n)$ where n is the size of the input ARCCTL$_{\div 1}$ model. In fact, TM simply looks at the input and writes in its output tape, one by one, the states (including the initial ones), the actions, labeling functions, and transitions. With regard to the formula, for the 4 first cases of the function f_2, it is easy to see that the transformation is polynomial in the size of the input formula. For the 5^{th} and 6^{th} cases, we need to store

Function Trans$_{AU}$ $(\varphi_1, \varphi_2, \varphi, \Delta k)$: ARCLT formula

1: **if** $A_\alpha(f_2(\phi_1)\ U\ f_2(\phi_2)) = \bot$ **then return false**
2: **else**
3: Compute $\prod_{A(\varphi_1 U \varphi_2)}$
4: $S_\varphi := \{\varphi \in S_\div | \varphi \models f_2(\varphi)\}$
5: **for all** $P_{A(\varphi_1 U \varphi_2)} \in \prod_{A(\varphi_1 U \varphi_2)}$
6: $S_\varphi^U = S_\varphi \cap P_{A(\varphi_1 U \varphi_2)}$
7: **if** $\dfrac{|S_\varphi^U|}{|P_{A(\varphi_1\ U\ \varphi_2)}|} \not\Delta\ k$ **then return false**
8: **end for**
9: **return** $A_\alpha(f_2(\varphi_1)\ U\ f_2(\varphi_2))$

the sets $\prod_{E(\varphi_1 U \varphi_2)}$, $\prod_{A(\varphi_1 U \varphi_2)}$ and S_φ, which are all logarithmic in the size of the input model and formula. In fact, the transformation function f_2 is recursive and the depth of the recursion is bounded by the length of the input formula. □

Theorem 5.2. *$MC(\text{ARCCTL}_{\div 1})$ is PSPACE-complete.*

Proof. The lower bound (i.e., PSPACE hardness) follows from the fact that $MC(\text{CTL}) \preceq_p MC(\text{ARCCTL}_{\div 1})$ and $MC(\text{CTL})$ is PSPACE-complete [60]. Since $MC(\text{ARCTL})$ is also PSPACE-complete [57], the upper bound follows from Proposition 5.1. □

Proposition 5.2. *Let $Mod(\mathscr{L})$ be the model of the language $\mathscr{L}$ and $\preceq_{\log}$ denote the log-space reduction. We have $Mod(\text{TCTL}^G) \preceq_{\log} Mod(\text{ARCCTL}_{\div 1})$.*

Proof. We show that the model reduction from TCTL^G to ARCCTL$_{\div 1}$ presented above can be computed by a deterministic Turing machine TM in space $O(\log(|Mod(\text{TCTL}^G)|))$. TM reads in the input tape a model of TCTL^G and generates in the output tape, one by one, the same states with the same state ordering, the same state labeling function, the same transitions as the input after labeling them with α_0 and writing α_0 in the set of atomic actions AC_A, and an additional state s_{new} to which it associates the last ordering rank with a unique label λ. For each state s and each pair of agents (i, j), TM reads the accessibility

relations $\rightsquigarrow_{i\to j}$ from this state one by one in a sequential way in the same order of the accessible states, adds α_s^{ij} to the set AC_A and adds transitions labeled by α_s^{ij}, first from s to the first accessible state, then between each two adjacent accessible states according to their order, and finally from the last accessible state to the newly added state s_{new}. Thus, to transform the accessibility relations, we only need to record 3 states at each moment of time: the original state s, the current accessible state, and the last accessible state to which a transition labeled by α_s^{ij} has been added. Since all these operations can be be done in a logarithmic space in the size of the input model, we are done. □

Proposition 5.3. $MC(\text{TCTL}^G) \preceq_p MC(\text{ARCCTL}_{\div 1})$

Proof. The model part is proven in Proposition 5.2. For the formula, we need to show that $|f_1(\Phi)|$ is polynomial in the length of the $\text{ARCCTL}_{\div 1}$ formula Φ. We prove this by induction over the structure of Φ. The proposition holds for the atomic case, and we have: $|f_1(\neg\varphi)| = 1+|f_1(\varphi)|$; $|f_1(\varphi_1 \vee \varphi_2)| = 1+|f_1(\varphi_1)|+|f_1(\varphi_2)|$; $|f_1(EX\varphi)| = 2+|f_1(\varphi)|$ (note that $|\alpha_0| = 1$); $|f_1(E(\varphi_1 U \varphi_2))| = 2+|f_1(\varphi_1)|+|f_1(\varphi_2)|$; $|f_1(A(\varphi_1 U \varphi_2))| = 2+|f_1(\varphi_1)|+|f_1(\varphi_2)|$; $|f_1(T_p^{\Delta k}(i,j,\varphi_1,\varphi_2))| = 7+|f_1(\varphi_1)|+2|f_1(\varphi_2)|$; $|f_1(T_c^{\Delta k}(i,j,\varphi_1,\varphi_2))| = 10+2|f_1(\varphi_1)|+2|f_1(\varphi_2)|$. Thus, if the proposition holds for φ, φ_1, and φ_2, then it holds for $f_1(\neg\varphi)$, $f_1(\varphi_1 \vee \varphi_2)$, $f_1(EX\varphi)$, $f_1(E(\varphi_1 U \varphi_2))$, $f_1(A(\varphi_1 U \varphi_2))$, $f_1(T_p^{\Delta k}(i,j,\varphi_1,\varphi_2))$ and $f_1(T_c^{\Delta k}(i,j,\varphi_1,\varphi_2))$. □

Theorem 5.3. *$MC(\text{TCTL}^G)$ is PSPACE-complete.*

Proof. The lower bound follows from the polynomial-space reduction from $MC(\text{CTL})$ proven to be complete for PSPACE [60]. The PSPACE upper bound follows from Proposition 5.3 and Theorem 5.2. □

5.5 Implementation and Experiments

We implemented our algorithms on top of MCMAS [73]. One of the appealing features of MCMAS-G, our new open source model checker, is the ability to provide the validation by means of a graphical user interface that is based on Eclipse to generate counterexamples and witnesses for several classes of TCTL^G formulae. The toolkit modeling language is VISPL [32], which is a simple description of the states and transitions in a model embedded with the vector-based semantics. The source and executable files of the resulting toolkit are available online at

In the following section, we consider the Breast Cancer Diagnosis and Treatment (BCDT) protocol as an illustrative application example introduced in Section 3.4.1 to show how our model checking technique can efficiently be applied on a medical health care platform to check the trust transactions against some quantified temporal trust conditions.

5.5.1 Performance Evaluation

We use our formal model M_G associated to the vector-based interpreted systems introduced earlier in Section 3.3 to formally model the BCDT protocol. According to this protocol, five parties are involved in the cancer diagnosis process, which are: *patient* denoted by *Pa*, *physician* (*Ph*), *pathologist* (*Pt*), *radiologist* (*Rd*), and *registrar* (*Rg*). Moreover, an environment agent *e* is added to model the BCDT process. In this scenario, the graded trust relationships between the participating parties express the system requirements that regulate the interacting agents. Such requirements are specified using our graded logic of trust TCTL^G. Indeed, trust relationships among parties evolve with interactions, and the following atomic propositions represent potential states in the evolution of these relationships: `MassDetected` for mass noticed, `MammoRefered` for mammography referral, `Cal_Det` for calcification detected, `Biop_Rec` for biopsy recommended. Moreover, the

atomic propositions, `TreatmentPlanAgreed`, `ReportReceived`, `TissueReceived`, `TissueAnalyzed`, and `ResultsAccommodated`. The involved parties must have the possibility of reaching states in which some of these propositions hold. Thus, the trust relationships are instantiated and help prospective agents decide how much should they trust other agents.

The following protocol properties are expressed in the TCTL^G logic to check the correctness of the process model.

$$\phi_1 = EF \;\; Tp^{\geq 0.75}(Pa, Ph, MassDetected, MammoRefered)$$

$$\phi_2 = EF \; T_c^{\geq 0.95}(Pa, Ph, MassDetected, AF \; MammoRefered)$$

$$\phi_3 = EF \;\; \neg T_c^{\leq 0.5}(Rd, Pt, TissueReceived, TissueAnalyzed)$$

$$\phi_4 = AG \;\; (ResultsAccommodated \Rightarrow$$

$$EF \; T_c^{\geq 0.75}(Pa, Ph, ReportReceived, AF \; TreatmentPlanAgreed))$$

$$\phi_5 = AG(Clac\_Det \Rightarrow AF \; T_p^{\geq 0.75}(Ph, Rd, \top, Biop\_Rec))$$

These formulae express reachability and liveness properties for both preconditional and conditional trust. For example, the formula ϕ_1 encodes the fact that there exists a state reachable from the initial state, such that the *Patient* trusts the *Physician* to refer her to a radiologist for a mammography upon he detects a suspicious mass with a degree ≥ 0.75. The formula ϕ_2 states that whenever the *physician* detects a suspicious mass in the patient's breast, then in all future computations, the latter trusts that *physician* to eventually refer her to a radiologist for a mammography with a degree at least 0.95. Moreover, in terms of liveness property, ϕ_5 states that 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

with a trust degree more than 0.75.

5.5.2 Experimental Results

In order to assess the scalability of our technique and implementation, we measured the model checking processing time to construct the model and the BDD memory usage to successfully perform the verification task when running on a machine Intel(R) Core(TM) i7-6700 CPU - 3.40GHZ with 16 GB memory. We run our experiments with a number of agents ranging from 6 to 30. Our motivation of considering different number of agents is to achieve different levels of scalability that makes the problem complex enough to observe significant results. The experiments revealed that all the tested formulae are satisfied. Table 5.1 recorded the verification results along with the number of agents and the reachable states in the model constructed. We can observe that the number of reachable states reflects the fact that the state space increases exponentially with the number of agents. It is also worth noticing that the memory consumption increases polynomially, which confirms the PSPACE-completeness result. However, the program timed out when the number of agents exceeds 30. Yet, it is still acceptable for detecting design errors in scalable models. In fact, we are unable to provide a full comparison of these results to other implementations as, to the best of our knowledge, there is no model checker tool that can be used to verify properties of quantified trust as we do in this work.

Table 5.1: Verification results of the BCDT protocol against TCTL^G formulae

Exp#	Agents#	States#	Time (sec)	Mem.(MB)
1	6	19	0.098	10
2	12	361	1.114	16
3	18	6859	26.13	45
4	24	117325	1614.5	48
5	30	2.00752e+06	57646	65

5.6 Summary

In this chapter, we introduced TCTL^G, a logical language that extends the Trust Computation Tree Logic TCTL to formally represent and reason about the quantitative aspect of trust in MASs. We assigned a weight to the sets of states that satisfy the trust formula. Thus, the degrees of trust can be obtained from the possible executions of the giving system. Moreover, we presented a model checking algorithm for TCTL^G that extends the CTL symbolic algorithm and its implementation that results in a new open source tool called MCMAS-G. Moreover, we proved through a 2-stage transformation procedure that the complexity of TCTL^G symbolic model checking for concurrent programs is PSPACE-complete with respect to the size of the program's components. 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. The experimental results confirmed the theoretical space complexity.

Chapter 6

Conclusions and Future Directions

This chapter gives a summary of the main contributions of the thesis. First, it presents the answer to our research questions, the concepts that we introduced, and the results that we obtained. Then, a sketch of possible extension of this work and the open questions remaining are stated .

6.1 Summary

Agents in an open environment are interacting with each other for different reasons in order to meet their goals. Model checking trust in MASs at design stage, to have confidence that the system will work as desired, is a challenging issue. While the number of proposals on trust modeling is significant, they differ, however, in the topics they addressed and the systems they implemented. In this thesis, we were primarily concerned with the issues of reasoning about and verifying trust in the context of MASs using the model checking approach, which has not been deeply investigated yet for trust systems. Differently from existing definitions in the literature, in this thesis, we considered a cognitive-independent view of trust where trust ingredients are seen from a non-epistemic angle. Trust in our work

is defined from a high-level abstraction perspective without having to depend on individual agents' internal states. We presented trust as a direct relation from one agent, the truster, toward another agent, the trustee, where such a relation presupposes specific conditions with respect to a particular content. Indeed, we have put forward a new logical framework for trust-based MASs to enable trusted agent interactions in open, heterogeneous and autonomous systems. In particular, we addressed the semantics, model checking, and complexity challenges. We conducted an in-depth literature review to guarantee the originality of our work and its effectiveness in filling the state-of-the-art research gaps. The framework consists of different components that are mainly introduced in three chapters. Chapters 3 and 4 revolved around our first three research questions and chapter 5 answered our last research question. In chapter 3 and from the modeling and specification perspectives, we achieved our goal of presenting the Trust Computation Tree Logic (TCTL), an extension of the CTL logic [43] to formally represent and reason about trust in a system of agents. Equipped with the presented reasoning postulates, TCTL does not only provide a formal basis for reasoning about trust states with preconditions, but it can also be seen as a formal modeling of the social trust interactions among agents. A major contribution of our approach is the new provided semantics of trust based on a new trust accessibility relation, which has not been addressed in the existing approaches. In particular, we provided an intuitive and grounded computational semantics based on a new vector-based definition of the formalism of interpreted systems. We were able to formally specify and automatically verify trust-based interactions among autonomous agents. A further novelty is the development of new model checking algorithms dedicated to the proposed logic (TCTL) and their implementations that result in a new tool called MCMAS-T along with its vector-based input language VISPL. We investigated the most intuitive and efficient algorithm for computing the trust set by introducing and comparing two different model checking algorithms

and analyzing two types of models (models with and without-loops).

In Chapter 4, we first extended the TCTL language with a modality to describe conditional trust. The specificity of this modality is its compatibility with the literature where trust is dealt with as conditional, meaning that trust should be expressed using antecedents and consequents. By doing so, a new language called TCTL^C is presented. Then, we investigated a different model checking technique to verify preconditional and conditional trust. In particular, we developed a new model checking framework for the TCTL logic of preconditional trust that is extended to design a new algorithm to model check conditional trust in MASs. We applied a set of formal rules to transform vector-extended transition systems into Kripke structures. Then, we transformed TCTL formulae to CTL ones based on certain rules developed specifically for this purpose. Such a transformation is performed by developing two formal methods that provide accurate alignments between source and target models, and at the same time preserve TCTL semantics without losing the validity of the original model properties. Our main challenge here was to make sure that the path through which a formula is satisfied in the original model TCTL is still satisfied in the corresponding path of the translated CTL model. Moreover, to perform this transformation, we developed a Java toolkit that automatically interacts with the NuSMV model checker of the CTL logic. The proof of the soundness and completeness of our transformation algorithms is provided. Furthermore, we proved that (1) the time complexity of TCTL and TCTL^C model checking in explicit models is P-complete with regard to the size of the model and length of the formula; and (2) the complexity of the same problems for concurrent programs is PSPACE-complete with respect to the size of the program's components. Therefore, our model checking algorithms have the same complexity as model checking CTL with regard to both explicit models and concurrent programs. Experiments conducted on a standard industrial case study demonstrated the efficiency and scalability of the technique. When we

compared this approach with the results proposed in Chapter 3, we reported that the developed verifier tool was able to verify a variety of formulae correctly and efficiently within a large case study having approximately 2.52442e+15 reachable states. Thanks to the high efficiency of CTL model checking to which the model checking of TCTL and conditional trust is transformed.

In Chapter 5, we introduced TCTL^G, a logical language that extends the Trust Computation Tree Logic TCTL to formally represent and reason about the quantitative aspect of trust in MASs. One of our main challenges here was the weight that we assigned to the sets of states that satisfy the trust formula. Thus, the degrees of trust can be obtained from the possible executions of the giving system. Moreover, we presented a model checking algorithm for TCTL^G that extends the CTL symbolic algorithm and its implementation that results in a new open source tool called MCMAS-G. We proved through a 2-stage transformation procedure that the complexity of TCTL^G symbolic model checking for concurrent programs is PSPACE-complete with respect to the size of the program's components. 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. The experimental results confirmed the theoretical space complexity.

6.2 Future Directions

Despite the fact that we successfully applied our specification languages and their model checker tools to different application domains, we believe, based on our literature reviews, that the following few points are worth investigating in the future:

- There is a need to tackle the runtime verification problem to investigate the dynamic changes of agents' behavior and their impact on trust. Perhaps this is an issue that we

see emerging in our approach. We believe that model checking and runtime verification can be integrated in a unified framework to verify trust-based MASs.

- Trust and commitments are two independent concepts. However, in concrete applications such as business interactions, there exist situations where performing trust and commitment scenarios contribute towards improving the efficiency of MASs. Indeed, trust provides a complementary aspect to commitments (i.e., trust and commitment are correlated to each other). In this thesis, we made a first step toward modeling the trust from a high level abstraction perspective. Further efforts are needed to induce commitments that would support trust by exploring the interaction between trust and social commitments from the specification and model checking standpoints.

- We noticed a strong connection between trust and reputation in social settings. Consequently, we would like to combine our formalism by defining a suitable semantics for the notion of reputation along with the associated model checking algorithm. This is extremely important to show how reputation can be exploited by an agent to build its trust degree in other agents.

- Considering the degree of confidence that an agent has in its trust value and incorporating this degree in our logic are interesting issues for future investigation.

Bibliography

[1] Sibel Adalı. *Modeling trust context in networks*. Springer, 2013.

[2] Faisal Al-Saqqar, Jamal Bentahar, Khalid Sultan, Wei Wan, and Ehsan Khosrowshahi Asl. Model checking temporal knowledge and commitments in multi-agent systems using reduction. *Simulation Modelling Practice and Theory*, 51:45–68, 2015.

[3] Alessandro 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*, pages 19–30, 2016.

[4] Hind Alotaibi and Hussein Zedan. Runtime verification of safety properties in multi-agents systems. In *10th International Conference on Intelligent Systems Design and Applications (ISDA)*, pages 356–362, 2010.

[5] Leila Amgoud and Robert Demolombe. An argumentation-based approach for reasoning about trust in information sources. *Argument & Computation*, 5(2-3):191–215, 2014.

[6] Najwa Abu Bakar and Ali Selamat. Runtime verification of multi-agent systems interaction quality. In *Asian Conference on Intelligent Information and Database Systems*, pages 435–444. Springer, 2013.

[7] Ahmed Saleh Bataineh, Jamal Bentahar, Mohamed El Menshawy, and Rachida Dssouli. Specifying and verifying contract-driven service compositions using commitments and model checking. *Expert Systems with Applications*, 74:151–184, 2017.

[8] Andreas Bauer, Martin Leucker, and Christian Schallhart. Runtime verification for ltl and tltl. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 20(4):14, 2011.

[9] Jamal Bentahar, Mohamed El-Menshawy, Hongyang Qu, and Rachida Dssouli. Communicative commitments: Model checking and complexity analysis. *Knowledge-Based Systems*, 35:21–34, 2012.

[10] Jamal Bentahar, Mohamed El-Menshawy, Hongyang Qu, and Rachida Dssouli. Communicative commitments: Model checking and complexity analysis. *Knowledge-Based Systems*, 35:21–34, 2012.

[11] Amir Jalaly Bidgoly and Behrouz Tork Ladani. Trust modeling and verification using colored petri nets. In *Information Security and Cryptology (ISCISC), 2011 8th International ISC Conference on*, pages 1–8. IEEE, 2011.

[12] Pratik K Biswas. Towards an agent-oriented approach to conceptualization. *Applied Soft Computing*, 8(1):127–139, 2008.

[13] Randal E Bryant. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on*, 100(8):677–691, 1986.

[14] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 1020 states and beyond. *Information and Computation*, 98(2):142 – 170, 1992.

[15] Michael Burrows, Martin Abadi, and Roger M Needham. A logic of authentication. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 426, pages 233–271. The Royal Society, 1989.

[16] Cristiano Castelfranchi and Rino Falcone. Principles of trust for MAS: cognitive anatomy, social importance, and quantification. In *Proceedings of the Third International Conference on Multiagent Systems, ICMAS*, pages 72–79, 1998.

[17] Zhimin Chen, Yi Jiang, Yao Zhao, et al. A collaborative filtering recommendation algorithm based on user interest change and trust evaluation. *JDCTA*, 4(9):106–113, 2010.

[18] Leturc Christopher and Grégory Bonnet. A normal modal logic for trust in the sincerity. In *17th International Conference on Autonomous Agents and Multiagent Systems*, Stockholm, Sweden, 2018.

[19] Alessandro Cimatti, Edmund Clarke, Fausto Giunchiglia, and Marco Roveri. Nusmv: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, 2000.

[20] Edmund M Clarke, E Allen Emerson, and Joseph Sifakis. Model checking: algorithmic verification and debugging. *Communications of the ACM*, 52(11):74–84, 2009.

[21] Edmund M Clarke, Orna Grumberg, and Doron Peled. *Model checking*. MIT press, 1999.

[22] Philip R Cohen and Hector J Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.

[23] Rocco De Nicola and Frits Vaandrager. Action versus state based logics for transition systems. In *Semantics of Systems of Concurrent Processes*, pages 407–419. Springer, 1990.

[24] Robert Demolombe. To trust information sources: A proposal for a modal logical framework. In *Trust and deception in virtual societies*, pages 111–124. Springer, 2001.

[25] Robert Demolombe. Graded trust. *AAMAS Trust*, pages 1–12, 2009.

[26] Robert Demolombe and Churn-Jung Liau. A logic of graded trust and belief fusion. In *Proc. of the 4th Workshop on Deception, Fraud and Trust in Agent Societies*, pages 13–25, 2001.

[27] Robert Demolombe and Emiliano Lorini. A logical account of trust in information sources. In *Proceedings of the 11th International Workshop on Trust in Agent Societies*, 2008.

[28] Nirmit Desai, Amit K Chopra, and Munindar P Singh. Amoeba: A methodology for modeling and evolving cross-organizational business processes. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 19(2):6, 2009.

[29] Nagat Drawel, Jamal Bentahar, Mohamed El-Menshawy, and Amine Laarej. Verifying temporal trust logic using ctl model checking. In *TRUST@ AAMAS*, pages 62–74, 2018.

[30] Nagat Drawel, Jamal Bentahar, and Hongyang Qu. Degrees of trust: Temporal logic and model checking. In *TRUST@ AAMAS*, pages 62–74, 2019.

[31] Nagat Drawel, Jamal Bentahar, and Elhadi Shakshuki. Reasoning about trust and time in a system of agents. *Procedia Computer Science*, 109:632–639, 2017.

[32] Nagat Drawel, Hongyang Qu, Jamal Bentahar, and Elhadi Shakshuki. Specification and automatic verification of trust-based multi-agent systems. *Future Generation Computer Systems*, 2018.

[33] Warda El Kholy, Jamal Bentahar, Mohamed El-Menshawy, Hongyang Qu, and Rachida Dssouli. Conditional commitments: Reasoning and model checking. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 24(2):Article 9, 2014.

[34] Warda EL Kholy, Jamal Bentahar, Mohamed El-Menshawy, Hongyang Qu, and Rachida Dssouli. Modeling and verifying choreographed multi-agent-based web service compositions regulated by commitment protocols. *Expert Systems with Applications*, 41(16):7478 – 7494, 2014.

[35] Warda El Kholy, Jamal Bentahar, Mohamed El Menshawy, Hongyang Qu, and Rachida Dssouli. Smc4ac: A new symbolic model checker for intelligent agent communication. *Fundamenta Informaticae*, 152(3):223–271, 2017.

[36] Mohamed El-Menshawy, Jamal Bentahar, and Rachida Dssouli. Symbolic model checking commitment protocols using reduction. In *International Workshop on Declarative Agent Languages and Technologies*, pages 185–203. Springer, 2010.

[37] Mohamed El Menshawy, Jamal Bentahar, Warda El Kholy, and Amine Laarej. Model checking real-time conditional commitment logic using transformation. *Journal of Systems and Software*, 2018.

[38] Mohamed El-Menshawy, Jamal Bentahar, Warda El Kholy, Pınar Yolum, and Rachida Dssouli. Computational logics and verification techniques of multi-agent commitments: survey. *The Knowledge Engineering Review*, 30(5):564–606, 2015.

[39] Mohamed El-Menshawy, Jamal Bentahar, Warda El Kholy, and Rachida Dssouli. Reducing model checking commitments for agent communication to model checking ARCTL and GCTL*. *Autonomous Agents and Multi-Agent Systems*, 27(3):375–418, 2013.

[40] Mohamed El-Menshawy, Jamal Bentahar, Hongyang Qu, and Rachida Dssouli. On the verification of social commitments and time. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 483–490, 2011.

[41] Mohamed El-Menshawy, Wei Wan, Jamal Bentahar, and Rachida Dssouli. Symbolic model checking for agent interactions. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1555–1556, 2010.

[42] Jumana El-Qurna, Hamdi Yahyaoui, and Mohamed Almulla. A new framework for the verification of service trust behaviors. *Knowledge-Based Systems*, 2017.

[43] E. Allen Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics*, pages 995–1072. MIT Press, 1990.

[44] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[45] Andreas Fuchs, Sigrid Gürgens, and Carsten Rudolph. A formal notion of trust–enabling reasoning about security properties. In *IFIP International Conference on Trust Management*, pages 200–215. Springer, 2010.

[46] Diego Gambetta et al. Can we trust trust. *Trust: Making and breaking cooperative relations*, 13:213–237, 2000.

[47] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT press, 2000.

[48] Andreas Herzig, Emiliano Lorini, Jomi F Hübner, and Laurent Vercouter. A logic of trust and reputation. *Logic Journal of IGPL*, 18(1):214–244, 2010.

[49] Andreas Herzig, Emiliano Lorini, and Frédéric Moisan. A simple logic of trust based on propositional assignments. In Fabio Paglieri, Luca Tummolini, and Rino Falcone, editors, *The Goals of Cognition. Essays in Honour of Cristiano Castelfranchi*, Tributes, pages 407–419. College Publications, 2012.

[50] Gerard J. Holzmann. The model checker spin. *IEEE Transactions on software engineering*, 23(5):279–295, 1997.

[51] Xiaowei Huang and Marta Zofia Kwiatkowska. Reasoning about cognitive trust in stochastic multiagent systems. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[52] Neil D. Jones. Space-bounded reducibility among combinatorial problems. *Computer and System Sciences*, 11(1):68–85, 1975.

[53] Audun Jøsang. *Subjective logic*. Springer, 2016.

[54] Özgür Kafalı, Nirav Ajmeri, and Munindar P Singh. Kont: Computing tradeoffs in normative multiagent systems. In *Proceedings of the 31st Conference on Artificial Intelligence (AAAI)*, pages 3006–3012, 2017.

[55] Özgür Kafalı and Pınar Yolum. Detecting exceptions in commitment protocols: Discovering hidden states. In *International Workshop on Languages, Methodologies and Development Tools for Multi-Agent Systems*, volume 6039 of *LNCS*, pages 112–127, 2009.

[56] Anup K Kalia and Munindar P Singh. Muon: designing multiagent communication protocols from interaction scenarios. *Autonomous Agents and Multi-Agent Systems*, 29(4):621–657, 2015.

[57] Warda El Kholy, Jamal Bentahar, Mohamed El-Menshawy, Hongyang Qu, and Rachida Dssouli. SMC4AC: A new symbolic model checker for intelligent agent communication. *Fundam. Inform.*, 152(3):223–271, 2017.

[58] Haeng-Kon Kim. Convergence agent model for developing u-healthcare systems. *Future Generation Computer Systems*, 35:39–48, 2014.

[59] Panagiotis Kouvaros, Alessio Lomuscio, Edoardo Pirovano, and Hashan Punchihewa. Formal verification of open multi-agent systems. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 179–187. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[60] O. Kupferman, M. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.

[61] Orna Kupferman, Moshe Y Vardi, and Pierre Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM (JACM)*, 47(2):312–360, 2000.

[62] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM: Probabilistic symbolic model checker. In *Computer Performance Evaluation / TOOLS*, pages 200–204, 2002.

[63] Bernd Lahno. Olli lagerspetz: Trust. the tacit demand. *Ethical Theory and Moral Practice*, 2(4):433–435, 1999.

[64] François Laroussinie, Antoine Meyer, and Eudes Petonnet. Counting ctl. In *International Conference on Foundations of Software Science and Computational Structures*, pages 206–220. Springer, 2010.

[65] John D Lee and Katrina A See. Trust in automation: Designing for appropriate reliance. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 46(1):50–80, 2004.

[66] Martin Leucker and Christian Schallhart. A brief account of runtime verification. *The Journal of Logic and Algebraic Programming*, 78(5):293–303, 2009.

[67] Churn-Jung Liau. Belief, information acquisition, and trust in multi-agent systems–a modal logic formulation. *Artif. Intell.*, 149(1):31–60, 2003.

[68] Chuchang Liu, Maris A Ozols, and Mehmet 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*, pages 142–156. Springer, 2004.

[69] Fenrong Liu and Emiliano Lorini. Reasoning about belief, evidence and trust in a multi-agent setting. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 71–89. Springer, 2017.

[70] Alessio Lomuscio and Jakub Michaliszyn. Model checking multi-agent systems against epistemic hs specifications with regular expressions. In *Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 298–307. AAAI Press, 2016.

[71] Alessio Lomuscio, Charles Pecheur, and Franco Raimondi. Automatic verification of knowledge and time with NuSMV. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1384–1389, 2007.

[72] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. Mcmas: A model checker for the verification of multi-agent systems. In *Proceedings of the 21st International Conference on Computer Aided Verification*, CAV '09, pages 682–688, Berlin, Heidelberg, 2009. Springer-Verlag.

[73] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. MCMAS: an open-source model checker for the verification of multi-agent systems. *STTT*, 19(1):9–30, 2017.

[74] Emiliano Lorini and Robert Demolombe. From binary trust to graded trust in information sources: a logical perspective. In *International Workshop on Trust in Agent Societies*, pages 205–225. Springer, 2008.

[75] Emiliano Lorini and Robert Demolombe. Trust and norms in the context of computer security: A logical formalization. In *International Conference on Deontic Logic in Computer Science*, pages 50–64. Springer, 2008.

[76] Emiliano Lorini, Guifei Jiang, and Laurent Perrussel. Trust-based belief change. In *European Conference on Artificial Intelligence-ECAI 2014*, pages pp–549, 2014.

[77] Stephen Paul Marsh. Formalising trust as a computational concept. 1994.

[78] Karsten Martiny and Ralf Moeller. Pdt logic: a probabilistic doxastic temporal logic for reasoning about beliefs in multi-agent systems. *Journal of Artificial Intelligence Research*, 57:39–112, 2016.

[79] Kenneth McMillan. *Symbolic Model Checking: An Approach to the State Explosion Problem*. PhD thesis, 1992. Carnegie Mellon University.

[80] Richard Montague. Universal grammar. *Theoria*, 36(3):373–398, 1970.

[81] Manh Hung Nguyen. Combination of formal logic and hedge algebra to estimate the degree of trust. *Journal of Computer Science and Cybernetics*, 31(3):203, 2015.

[82] Eugénio Oliveira, Henrique Lopes Cardoso, Maria Joana Urbano, and Ana Paula Rocha. Normative monitoring of agents to build trust in an environment for b2b. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 172–181. Springer, 2014.

[83] Nardine Osman and David Robertson. Dynamic verification of trust in distributed open systems. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 1440–1445, 2007.

[84] Eric Pacuit. *Neighborhood Semantics for Modal Logic*. Springer, 2017.

[85] Simon Parsons, Katie Atkinson, Zimi Li, Peter McBurney, Elizabeth Sklar, Munindar Singh, Karen Haigh, Karl Levitt, and Jeff Rowe. Argument schemes for reasoning about trust. *Argument & Computation*, 5(2-3):160–190, 2014.

[86] Simon Parsons, Yuqing Tang, Elizabeth Sklar, Peter McBurney, and Kai Cai. Argumentation-based reasoning in agents with varying degrees of trust. 2011.

[87] David Pearce and Levan Uridia. Trust, belief and honesty. In *GCAI 2015. Global Conference on Artificial Intelligence*, pages 215–228. EasyChair, 2015.

[88] Charles Pecheur and Franco Raimondi. Symbolic model checking of logics with actions. In *International Workshop on Model Checking and Artificial Intelligence*, pages 113–128. Springer, 2006.

[89] Wojciech Penczek and Alessio Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *Fundamenta Informaticae*, 55(2):167–185, 2003.

[90] Isaac Pinyol and Jordi Sabater-Mir. Computational trust and reputation models for open multi-agent systems: a review. *Artificial Intelligence Review*, 40(1):1–25, 2013.

[91] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57, 1977.

[92] Giuseppe Primiero, Franco Raimondi, and Neha Rungta. Model checking degrees of belief in a system of agents. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 133–140. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[93] Sebastian Ries, Sheikh Mahbub Habib, Max Mühlhäuser, and Vijay Varadharajan. Certainlogic: A logic for modeling trust and uncertainty. In *International Conference on Trust and Trustworthy Computing*, pages 254–261. Springer, 2011.

[94] Noel Sardana, Robin Cohen, Jie Zhang, and Shuo Chen. A bayesian multiagent trust model for social networks. *IEEE Transactions on Computational Social Systems*, 5(4):995–1008, 2018.

[95] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Computer and System Sciences*, 4(2):177–192, 1970.

[96] Philippe Schnoebelen. The complexity of temporal logic model checking. In Philippe Balbiani, Nobu-Yuki Suzuki, Frank Wolter, and Michael Zakharyaschev, editors, *Advances in Modal Logic 4, papers from the fourth conference on "Advances in Modal logic," held in Toulouse, France, 30 September - 2 October 2002*, pages 393–436. King's College Publications, 2002.

[97] Munindar P. Singh. Semantical considerations on dialectical and practical commitments. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI, Chicago, Illinois, USA*, pages 176–181, 2008.

[98] Munindar P Singh. Trust as dependence: a logical approach. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, pages 863–870, 2011.

[99] Khalid Sultan, Jamal Bentahar, Wei Wan, and Faisal Al-Saqqar. Modeling and verifying probabilistic multi-agent systems using knowledge and social commitments. *Expert Systems with Applications*, 41(14):6291–6304, 2014.

[100] Mirko Tagliaferri and Alessandro Aldini. A trust logic for pre-trust computations. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 2006–2012. IEEE, 2018.

[101] Yuqing Tang, Kai Cai, Peter McBurney, Elizabeth Sklar, and Simon Parsons. Using argumentation to reason about trust and belief. *Journal of Logic and Computation*, 22(5):979–1018, 2012.

[102] Pankaj R Telang, Anup K Kalia, and Munindar P Singh. Modeling healthcare processes using commitments: An empirical evaluation. *PloS one*, 10(11):e0141202, 2015.

[103] Pankaj R Telang and Munindar P Singh. Enhancing tropos with commitments. In *Conceptual Modeling: Foundations and Applications*, pages 417–435. Springer, 2009.

[104] Ron van der Meyden and Kaile Su. Symbolic model checking the knowledge of the dining cryptographers. In *17th IEEE Computer Security Foundations Workshop*, pages 280–291, 2004.

[105] R Vijayan and N Jeyanthi. A survey of trust management in mobile ad hoc networks. *International Journal of Applied Engineering Research*, 11(4):2833–2838, 2016.

[106] Omar Abdul Wahab, Jamal Bentahar, Hadi Otrok, and Azzam Mourad. Towards trustworthy multi-cloud services communities: A trust-based hedonic coalitional game. *IEEE Transactions on Services Computing*, In press, 2016.

[107] Fenghui Wang, Ming Yang, and Ruqing Yang. Simulation of multi-agent based cybernetic transportation system. *Simulation Modelling Practice and Theory*, 16(10):1606–1614, 2008.

[108] Matt Webster, Louise Dennis, and Michael Fisher. Model-checking auctions, coalitions and trust. Technical report, University of Liverpoo, 2009.

[109] Michael Wooldridge. *Introduction to multiagent systems*. Wiley, 2002.

[110] Michael Wooldridge and Nicholas R Jennings. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(02):115–152, 1995.

[111] Michael J Wooldridge. *Agent technology: foundations, applications, and markets*. Springer Science & Business Media, 1998.