Formalizing Group and Propagated Trust in Multi-Agent Systems

Nagat Drawel , **Jamal Bentahar** , **Amine Laarej** and **Gaith Rjoub**
Concordia University, Montreal, Canada
n_drawe@encs.concordia.ca, bentahar@ciise.concordia.ca, laarej.amine@gmail.com,
g_rjoub@encs.concordia.ca

Abstract

We present a formal framework that allows individual and group of agents to reason about their trust toward other agents. In particular, we propose a branching time temporal logic BT which includes operators that express the concepts of everyone trust, distributed trust and propagated trust. We analyze the satisfiability and model checking problems of this logic using a reduction technique.

1 Introduction

Trust is a crucial basis for the development of effective Multi-Agent System (MAS) applications. It has been extensively addressed in many research contexts (e.g., peer-to-peer networks and grid computing). These researches are mainly placed under two major streams: trust computational models and logical trust formalization. The former measures the value of trust to compute the strength level in which an agent trusts other agents in order to establish future interactions [Wahab *et al.*, 2018; Sardana *et al.*, 2018; Nayak *et al.*, 2019]. In this stream, trust was first formalized as a measurable concept in [Marsh, 1994]. Following this work, a number of trust models have been put forward and several proposals investigated trust propagation [Jamali and Ester, 2010; Chakraborty and Karform, 2012]. On the other hand, the latter deals with how one agent in the system can trust that another agent behaves and will perform an action in a certain way [Singh, 2011; Primiero and Raimondi, 2014; Primiero, 2016; Liu and Lorini, 2017; Drawel *et al.*, 2020].

Trust is a complex concept that is hard to be precisely defined. Different meanings of trust have been given in various domains. In the context of MASs, [Castelfranchi and Falcone, 1998] defined trust as mental attitudes of the truster who believes that the trustee is capable to act and achieve a given goal. From logical point of view, modal logics have been used to reason about cognitive aspects of trust by many researchers. For instance, in [Herzig *et al.*, 2012], the authors proposed a logical framework for the concept of trust where trust is basically expressed as a combination of different modalities based on the logic of action and time [Harel *et al.*, 2000] and the BDI logic [Cohen and Levesque, 1990]. In [Liu and Lorini, 2017], the authors presented a new dynamic logic called DL-BET for reasoning about the interaction between belief, evidence and trust. Huang et al. [Huang *et al.*, 2019] considered the setting of stochastic multi-agent systems, where an automated verification framework for quantifying and reasoning about cognitive trust is proposed. Moreover, some proposals have abstracted from the cognitive stance and presented trust as a direct modality. In [Drawel *et al.*, 2020], a new branching temporal logic of preconditional trust, which extends the Computation Tree Logic (CTL) is introduced along with its model checking technique. [Singh, 2011] provided a formal semantics for trust with various logical postulates used to reason about trust from an architectural perspective. However, most of these approaches focus solely on individual trust that defines trust as a relationship that only involves two agents and is not propagated to other agents.

In this paper, we are interested in trust that goes beyond individuals where one individual agent trusts another agent, toward a group trust where a group of agents trusts a particular agent. We aim to capture the concepts of everyone trust and distributed trust toward a particular agent. Everyone trust is when all the members of the group agree on trusting another agent. Distributed trust is when the trust is distributed among the members of the group. We are also interested in trust that can propagate through the MAS from one agent to another. We are considering these two concepts from the logical perspective, in particular formalization, model checking and satisfiability problems.

A branching temporal logic extending CTL has been introduced to reason about trust and time in MASs [Drawel *et al.*, 2017; Drawel *et al.*, 2020]. This logic, called TCTL provides an interesting framework to reason about individual trust. However, we will show in this paper that this logic cannot be extended to accommodate group and propagated trust because of its limited expressive power. The logic fails to appropriately represent nested trust formulae. We will present a branching time temporal logic named BT that refines TCTL while being expressive enough to go beyond individual trust and include operators that express the concepts of everyone trust, distributed trust and propagated trust. Moreover, we will analyze the model checking and satisfiability problems of this logic using a sound reduction technique.

The paper is organized as follows. We first present the syntax and semantics of TCTL and discuss its limitation in Section 2. In Section 3, we introduce the Branching Trust Logic

(BT). The model checking and satisfiability problems of BT are addressed using a transformation procedure in Section 4. The complexity of these problems is analyzed in Section 5. In Section 6, we present the experimental results. We end by concluding the paper in Section 7.

2 Trust Computational Temporal Logic

2.1 Preliminaries

The semantics of TCTL formulae is interpreted using a model generated from the extended Interpreted System formalism introduced in [Drawel *et al.*, 2017]. This formalism includes the notion of agents' vector ν to account for the interaction that occurs during the execution of MAS. That is, for all states $s, s' \in S$ and $i, j \in Agt$, a vector of size n is associated with each local state $l_i \in L_i$ of the n agents. The vector ν is used to define the trust accessibility relation $\sim_{i,j}$. The idea is, the relation $\sim_{i,j}$ relates the states that are considered to be trustful from the vision of agent i with regard to agent j. Specifically, this is obtained by comparing the element $\nu^i(j)$ in the local state l_i at the global state s (denoted by $l_i(s)(\nu^i(j))$) with $\nu^i(j)$ in the local state l_i at the global state s' (denoted by $l_i(s')(\nu^i(j))$). Thus, the trust accessibility of agent i toward agent j does exist between two global states 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.

Definition 1 (Model of TCTL). *A model of trust is a tuple* $M_t = (S, I, R, \{\sim_{i,j} |(i,j) \in Agt^2\}, V)$
*where: S is a non-empty set of reachable global states for the system; $I \subseteq S$ is a set of initial global states for the system; $R \subseteq S \times S$ is the transition relation; $\sim_{i,j} \subseteq S \times S$ is the direct trust accessibility relation for each truster-trustee pair of agents $(i,j) \in Agt^2$ defined by $\sim_{i,j}$ iff: **(1)** $l_i(s)(\nu^i(j)) = l_i(s')(\nu^i(j))$ and **(2)** 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 propositional variables.*

Definition 2 (Syntax of TCTL). *The syntax of TCTL is defined recursively as follows:*

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

The CTL fragment formulae are defined as usual (see [Emerson, 1990]). The formula $T(i,j,\varphi)$ called *trust* formula is read as "agent i trusts agent j to bring about φ".

Definition 3 (Semantics of TCTL). *Given the model M_t, the satisfaction of a TCTL formula φ in a global state s, denoted as $(M_t, s) \models \varphi$, is recursively defined. The semantics of the CTL fragment is as usual [Emerson, 1990]. The semantics of the operator T is as follows:*

$(M_t, s) \models T(i,j,\varphi)$ iff $s \models \neg\varphi$ and $\forall s' \neq s$ such that $s \sim_{i,j} s'$ we have $(M_t, s') \models \varphi$.

The state formula $T(i,j,\varphi)$ is satisfied in the model M_t at s iff φ does not hold in s and all the trust accessible states s' that are different from the current state satisfy the content φ.

2.2 Discussion

Although TCTL has been presented with interesting reasoning postulates [Drawel *et al.*, 2017; Drawel *et al.*, 2020], a deep analysis of this logic reveals a major paradox resulted from the underlying assumption that complies with the first postulate in [Singh, 2011] stating that "when the content holds, the trust in this content is completed and is, therefore, no longer active". Indeed, enforcing the condition $\neg\varphi$ to be satisfied in the current state s for the trust to take place yields the following paradoxical postulate:

$$T(i,j,\varphi) \Rightarrow T(i,j,\neg T(i,j,\varphi))$$

which means if i trusts j about φ, then i trusts this agent that the first trust does not hold. The proof is straightforward since all accessible states satisfy $\neg T(i,j,\varphi)$ because of the satisfaction of φ and the current state satisfies $T(i,j,\varphi)$.

Therefore, this logic is not suitable to reason about trust properties that need nested trust formulae to hold. We present then a refined logic that does not have this postulate, so it can express nested formulae needed for propagated trust, everyone trust and distributed trust. In this logic, tautologies are trusted, but this is less important in practical scenarios.

3 Branching Trust Logic (BT)

Definition 4 (Syntax of BT). *Let $G \subseteq Agt$ be a group of agents and ρ a propositional variable from the set AP_b. The syntax of BT is defined recursively as follows:*

$$\varphi ::= \rho \mid \neg\varphi \mid \varphi \vee \varphi \mid EX\varphi \mid EG\varphi \mid E(\varphi U \varphi) \mid T$$
$$T ::= T(i,j,\varphi) \mid E_T(G,j,\varphi) \mid D_T(G,j,\varphi) \mid P_T(i,j,\varphi)$$

The formula T represents trust and three notions of group and propagated trust: E_T, D_T, and P_T. The formula E_T refers to "Everyone trusts" and means that everyone in the group G trusts agent j to bring about φ. Technically, "Everyone trusts" can be seen as the conjunction of the individual trust of each agent in the group. Moreover, D_T denotes "Distributed trust". That is, a group has its trust distributed among its member agents. Propagated trust P_T indicates that a new trust relationship can be derived from preexisting agent's trust. BT formulae are evaluated over an extended Interpreted System $M_b = (S_b, I_b, R_b, \{\sim_{i,j} |(i,j) \in Agt^2\}, V_b)$, which is the same as M_t.

The semantics of group and propagated trust is defined using new accessibility relations derived from the trust accessibility relation as follows.

Definition 5 (Group/Propagation Accessibility Relations). *The group and propagation accessibility relations are:*

- $\sim^E_{G,j} = \bigcup_{i \in G} \sim_{i,j}$, *i.e., the union of the trust accessibility relations between every agent of the group G and the agent j.*
- $\sim^D_{G,j} = \bigcap_{i \in G} \sim_{i,j}$, *i.e., is the intersection of the trust accessibility relations between every agent in the group G and the agent j.*
- $s \sim^P_{i,j} s'$ *iff there are agents* $i_1, \ldots, i_{n-1}$ *and states* $s_1, \ldots, s_{n-1}$ *s.t.* $s \sim_{i,i_1} s_1, s_1 \sim_{i_1,i_2} s_2, \ldots, s_{n-1} \sim_{i_{n-1},j} s'$.

Definition 6 (Satisfaction). *Given the model M_b, the semantics of trust and group trust operators is recursively defined as follows:*

$(M_b, s) \models T(i, j, \varphi)$ *iff* $\forall s' \neq s$ *such that* $s \sim_{i,j} s'$, *we have* $(M_b, s') \models \varphi$.

$(M_b, s) \models E_T(G, j, \varphi)$ *iff* $\forall s' \neq s$ *such that* $s \sim^{E}_{G,j} s'$, *we have* $(M_b, s') \models \varphi$.

$(M_b, s) \models D_T(G, j, \varphi)$ *iff* $\forall s' \neq s$ *such that* $s \sim^{D}_{G,j} s'$, *we have* $(M_b, s') \models \varphi$.

$(M_b, s) \models P_T(i, j, \varphi)$ *iff* $\forall s' \neq s$ *such that* $s \sim^{P}_{i,j} s'$, *we have* $(M_b, s') \models \varphi$.

The following reasoning postulates hold in BT:

1. $T(i, j_1, T(j_1, j_2, \varphi)) \Rightarrow P_T(i, j_2, \varphi)$
2. $P_T(i, j_1, T(j_1, j_2, \varphi)) \Rightarrow P_T(i, j_2, \varphi)$
3. $T(i, j_1, P_T(j_1, j_2, \varphi)) \Rightarrow P_T(i, j_2, \varphi)$
4. $P_T(i, j_1, P_T(j_1, j_2, \varphi)) \Rightarrow P_T(i, j_2, \varphi)$
5. $T(i_1, j, \varphi) \wedge T(i_2, j, \varphi \Rightarrow \psi) \Rightarrow D_T(\{i_1, i_2\}, j, \psi)$
6. $\bigvee\limits_{i \in G} T(i, j, \varphi) \Rightarrow D_T(G, j, \varphi)$
7. $E_T(G, j, \varphi) \Leftrightarrow \bigwedge\limits_{i \in G} T(i, j, \varphi)$

The first four postulates capture the trust propagation among agents in one step (1st postulate) or many steps. The 5th and 6th postulates capture the properties of the distributed trust among the group members. The 7th postulate shows the everyone trust property.

4 Transformation Procedure

To address the model checking and satisfiability problems of BT, we introduce a transformation procedure [El-Menshawy *et al.*, 2010] that allows us to leverage the model checking and satisfiability procedures of CTL [Emerson, 1990; Emerson and Halpern, 1985]. A transformation from TCTL model checking to CTL model checking has been attempted in [Drawel *et al.*, 2018]. However, this approach does not capture the alignments between source and target models, which results in unsound transformations. Technically, the transformation algorithms for both the model and formulae overlook some critical cases. More precisely, the technique provided for the model consists of adding transitions to represent the accessibility relations, and so, it does not distinguish between original transitions in the original TCTL model and the transitions added in the CTL model. It turns out that some formulae with temporal operators EX and U become true in the transformed CTL model while they are false in the original TCTL model. Our procedure avoids these problems by capturing accessibility relations through distinguishable states and transitions using atomic propositions added in the transformed CTL formulae.

4.1 From BT Model to CTL Model

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

Definition 7 (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; and* $V_c : S_c \rightarrow 2^{AP_c}$ *is a labeling function that maps each state to the set of CTL propositional variables* AP_c *holding in it.*

Having presented the CTL model, the next step is to establish our transformation procedure. Given a BT model $M_b = (S_b, R_b, I_b, \{\sim_{i,j} |(i, j) \in Agt^2\}, V_b)$, Algorithm 1 shows how this model, taken as input, is transformed into a CTL model $M_c = (S_c, R_c, I_c, V_c)$ as output. Initially, the output model M_c has the same set of system states, initial states, transitions and valuation function as M_b (i.e., $S_c = S_b$; $I_c = I_b$; $R_c = R_b$; and $V_c = V_b$). Thus, at the beginning, the states of M_c are labeled with the same propositional variables as the states in M_b. We define a new set of fresh propositional variables $Prop$ for the CTL logic needed to represent the trust accessibility relations to capture the semantics of trust operators. Moreover, we define a new atomic proposition χ for CTL that will be used to preserve the actual temporal transition relation by distinguishing the added states and transitions from the original ones in M_b. The set AP_c is then defined as the set AP_b augmented with χ and the propositional variables $\alpha^{ij}, \alpha^{Ej}, \alpha^{Dj}$, and α^{Pj} for the individual, group, and propagated trust accessibility relations for all the agents i, j and groups G. The algorithm proceeds to transform the trust accessibility relations for all the agents. First, it checks if there is an accessibility relation between each pair of distinct states s and s'. Based on the type of all the possible accessibility relations, it assigns a specific propositional variable to the set $Prop$. A new state s'' is added to the set of system states S_c along with new transitions from s to s'' and from s'' to s' in R_c. Further, the new state s'' is labeled with χ and the atomic propositions in the set $Prop$ 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. However, to avoid adding many states for different accessibility relations between the same two states, the algorithm checks if s'' is already added for another accessibility relation. If s'' already exists, the algorithm will only add the atomic propositions $Prop$ to mark the accessible state for any other interacting agents. Finally, the algorithm returns the transformed model M_c after iterating over all the transitions.

Proposition 1 (Boundedness of Model Transformation). *Let* M_c *be the model obtained from* M_b *using Algorithm 1 and* $|M_c|$ *and* $|M_b|$ *be the size of* M_c *and* M_b *respectively.* $|M_c| < 3|M_b|$

Proof. Let $|A_b|$ be the number of accessibility relations in M_b, i.e., $|A_b| = |\{\sim_{i,j} |(i, j) \in Agt^2\}|$. We have: $|M_c| = |S_c| + |R_c|$ and $|M_b| = |S_b| + |R_b| + |A_b|$. In the worst case, each pair of distinct states $(s, s') \in R_b$ is connected by exactly one accessibility relation. Since each accessibility relation is translated into one state and two transitions in M_c, we obtain: $|M_c| = |S_b| + |R_b| + 3|A_b|$. In the general case, more than one accessible state might exist between each pair of states. Since the other accessibility relations benefit from

Algorithm 1 Transform $M_b = (S_b, R_b, I_b, \sim_{i,j} \ |(i,j) \in Agt^2\}, V_b)$ into $M_c = (S_c, I_c, R_c, V_c)$

$S_c := S_b$; $I_c := I_b$; $R_c := R_b$; $V_c := V_b$;
for each $(s, s') \in S_b^2$ s.t. $s' \neq s$ **and all** i, j, G **do**
  $Prop := \emptyset$;
  if $s \sim_{i,j} s'$ **then** $Prop := Prop \cup \{\alpha^{ij}\}$;
  if $s \sim_{G,j}^{E} s'$ **then** $Prop := Prop \cup \{\alpha^{Ej}\}$;
  if $s \sim_{G,j}^{D} s'$ **then** $Prop := Prop \cup \{\alpha^{Dj}\}$;
  if $s \sim_{i,j}^{P} s'$ **then** $Prop := Prop \cup \{\alpha^{Pj}\}$;
 if $\exists s''$ s.t. $(s, s''), (s'', s') \in R_c$ and $\chi \in V_c(s'')$ **then**
  $V_c(s'') := V_c(s'') \cup Prop$;
 else
  $S_c := S_c \cup \{s''\}$; $R_c := R_c \cup \{(s, s''), (s'', s')\}$;
  $V_c(s'') := \{\chi\} \cup Prop$;
 end if
end for
return M_c;

the already added states and transitions, we obtain: $|M_c| \leq |S_b| + |R_b| + 3|A_b|$. The result then follows. □

4.2 From BT Formulae to CTL Formulae

Algorithm 2 illustrates the formulae transformation function defined over the structure of the original BT formula. The function is recursive with the propositional variables of AP_b being the base case ($AP_b \subset AP_c$). For the temporal operators, 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_b 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, group trust, and propagated trust modalities, each formula is transformed inductively into CTL according to the defined semantics as follows: along each path, if the next state on that path satisfies the corresponding atomic proposition (from the set $Prop$), then the next state of the added state also satisfies the transformation of the trust content φ. The state that satisfies the fresh propositional variable is the added state to capture the corresponding accessibility, which explains the double use of AX.

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

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

- The result holds for the base case (atomic propositions).

Algorithm 2 Transform BT formula φ into CTL formula $f(\varphi)$

$f(\rho) = \rho$ if $\rho \in AP_b$;
$f(\neg\varphi) = \neg f(\varphi)$;
$f(\varphi \vee \psi) = f(\varphi) \vee f(\psi)$;
$f(EX\varphi) = EX(f(\varphi) \wedge \neg\chi)$;
$f(E(\varphi \ U \ \psi)) = E((f(\varphi) \wedge \neg\chi) \ U \ (f(\psi) \wedge \neg\chi))$;
$f(EG\varphi) = EG(f(\varphi) \wedge \neg\chi)$;
$f(T(i,j,\varphi)) = AX(\alpha^{ij} \rightarrow AXf(\varphi))$;
$fE_T(G,j,\varphi) = AX(\alpha^{Ej} \rightarrow AXf(\varphi))$;
$fD_T(G,j,\varphi) = AX(\alpha^{Dj} \rightarrow AXf(\varphi))$;
$fP_T(i,j,\varphi) = AX(\alpha^{Pj} \rightarrow AXf(\varphi))$;

- For the formula $\phi = \neg\varphi$, we have $|f(\phi)| = |f(\varphi)| + 1$. Therefore, by assumption that the proposition holds for the formula φ, $\exists k_1$ such that $|f(\phi)| < k_1|\varphi| + 1$. Since $|\varphi| < |\phi|$, and $|\phi| > 1$, we get $|f(\phi)| < (k_1 + 1)|\phi|$, so the proposition.
- 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 proof for $EG\varphi$ is similar.
- For the formula $\phi = E(\varphi \ U \ \psi)$, we have $|f(\phi)| = |f(\varphi)| + |f(\psi)| + 7$. Thus. by assumption that the proposition holds for the formulae φ and ψ, $\exists k_1, k_2$ such that $|f(\phi)| < k_1|\varphi| + k_2|\psi| + 7$. Because $|\varphi| < |\phi|$, $|\psi| < |\phi|$, and $|\phi| > 1$, we obtain $|f(\phi)| < (k_1 + k_2 + 7)|\phi|$. The proof for $\varphi \vee \psi$ is similar ($k = k_1 + k_2 + 1$).
- For the formula $\phi = T(i,j,\varphi)$, we have $f(T(i,j,\varphi)) = AX(\alpha^{ij} \rightarrow AXf(\varphi))$. Thus, $|f(\phi)| = |f(\varphi)| + 4$, and 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 proof is similar for the group and propagated trust formulae E_T, D_T, and P_T.

□

Model Checking

Given a BT model M_b representing a MAS and a BT formula φ describing the property that the model M_b has to satisfy, the problem of model checking BT is the problem of verifying whether or not $M_b \models \varphi$. The introduced transformation procedure provides a solution to this problem, by simply calling the model checking procedures of CTL as stated by the following theorem.

Theorem 1 (BT Model Checking). *Let M_b and φ be respectively a BT model and formula and let $f(M_b)$ and $f(\varphi)$ be the CTL model and formula obtained via Algorithms 1 and 2. We have $(M_b, s) \models \varphi$ iff $(f(M_b), s) \models f(\varphi)$.*

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

- For the propositional variables from the set AP_b, the result is straightforward (notice that $AP_b \subset AP_c$). The result is also direct for the negation and disjunction cases.

- For the formula $\phi = EX\varphi$, we have $(M_b, s) \models EX\varphi$ iff there exists an immediate successor to s where φ holds. Consequently, from the definition of $f(M_b)$ and $f(\varphi)$, we obtain $(M_b, s) \models EX\varphi$ iff $(M_c, s) \models EX(f(\varphi) \wedge \neg\chi)$. 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, where only original temporal transitions are considered.
- The trust formula: $T(i, j, \varphi)$. $AX(\alpha^{ij} \rightarrow AXf(\varphi))$ captures the semantics of the trust formula where all accessible states (those satisfying α^{ij} in M_c) should satisfy φ. The proof is similar for the group and propagated trust formulae.

□

Satisfiability

Given a BT formula φ, the satisfiability of BT is the problem of deciding if there exists a model M_b such that $M_b \models \varphi$. As for model checking problem, the transformation procedure provides also a solution for the BT satisfiability problem.

Theorem 2 (BT Satisfiability). *Let φ be a BT formula and $f(\varphi)$ the CTL formula obtained via Algorithm 2. We have φ is satisfiable iff $f(\varphi)$ is satisfiable.*

Proof. The right implication is direct from Theorem 1 because if there is a model M_b satisfying φ, then $f(M_b)$ that satisfies $f(\varphi)$ does exist. For the left implication, we can prove by induction on the structure of the formula that if there is a model M_c that satisfies $f(\varphi)$, we can always find a model M'_c that satisfies the same formula where states satisfying fresh propositional variables from $AP_c \setminus AP_b$ do not satisfy any other non-fresh propositional variable from AP_b. From M'_c we can construct a model M_b s.t. $f(M_b) = M'_c$ □

5 Complexity Analysis

In this section, we will first analyze the time complexity of model checking BT with regard to the size of the explicit model M_b and length of the formula to be checked. Since we are using symbolic model checking of CTL, we will also analyze the space complexity of model checking BT for concurrent programs [Kupferman *et al.*, 2000] with respect to the size of the components of these programs and length of the formula. The complexity of the BT satisfiability problem will end this section.

As our approach is transformation-based, we start by analyzing the time complexity of transforming the BT model and formula with respect to explicit models, where all states and transitions are enumerated. Specifically, we prove that these two transformations are linear with respect to both the input BT model and the formula. The linear complexity of these two transformations entails the P-completeness of the BT model checking problem in explicit models. Given that, we proceed to analyze the space complexity of the BT model checking problem and prove its 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 procedure for concurrent programs that provide compact representations of the systems to be checked is that in practice, existing model checking tools (e.g., MCMAS and NuSMV) do not support explicit representations where states and transitions are listed explicitly (as Kripke-like structures). In fact, only local states and transitions of each component are represented. Therefore, the actual system can still be represented by combining local states and transitions to build reachable states.

5.1 Model Checking Time Complexity

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

Theorem 3 (Explicit BT Model Checking: Upper Bound). *The BT model checking problem can be solved in time $O(|M_b| \times |\varphi|)$.*

Proof. BT can be reduced to CTL, and it is known from [Clarke *et al.*, 1999] that CTL model checking can be done in a linear time with respect to the size of the CTL model and formula, i.e., $O(|f(M_b)| \times |f(\varphi)|)$. From Proposition 1, $|f(M_b)| < 3|M_b|$, i.e., the size of $f(M_b)$ is linear with the size of M_b. Moreover, from Proposition 2, the length of $f(\varphi)$ is linear with the length of φ. Indeed, Algorithm 2 takes the BT 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 BT 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. □

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

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

5.2 Model Checking Space Complexity

In this subsection, we will prove that the complexity of BT model checking for concurrent programs is PSPACE-complete. 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.

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

Proof. The transformation of the BT model and BT 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 BT model, and polynomial space w.r.t. the length of the BT formula. For the model, TM reads in the input tape a model of BT 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, s') in the input model, it writes one-by-one, the transitions in the set R_c. It also reads the accessibility relations $\sim_{i,j}$ 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 propositional variables: 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 propositional variable α^{ij} is added. The group and propagated accessibility relations are done in the same way. 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 2 that any BT formula is transformable into a CTL formula whose length is linearly bounded by the length of the input formula. All these recursive transformations are clearly polynomial space in the length of the input formula, so the theorem. □

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

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

5.3 Satisfiability Complexity

Theorem 7 (BT Satisfiability: Completeness). *The BT satisfiability problem is EXPTIME-complete.*

Proof. Membership. Using the result of Theorem 2, we can imagine an EXPTIME algorithm that solves the BT satisfiability problem as follows: 1) Transform the input BT formula φ to the CTL formula $f(\varphi)$ using Algorithm 2. As mentioned in the proof of Theorem 3, this can be done in a linear time; 2) Call the algorithm to solve the satisfiability of $f(\varphi)$ which can be done in EXPTIME [Emerson and Halpern, 1985]. Hardness. The hardness follows from the fact that BT subsumes CTL proven to be EXPTIME-complete [Emerson and Halpern, 1985]. □

6 Implementation and Experimental Results

To support the modeling and verification of BT logic for MASs, we implemented the transformation procedure and developed a tool that automatically: (i) transforms a given BT model and formulae to a valid CTL model and formulae; (ii) interacts with the NuSMV model checker in order to perform the verification process. For testing, we used the well-known ordering protocol that regulates the interaction between seller and buyer agents introduced in [Desai *et al.*, 2005]. Our motivation is to formalize the protocol requirements using BT model M_b and by expressing a set of properties in BT logic in order to assess the scalability of our technique. We expressed a set of group and propagated trust properties. For example, the formula $EF\ E_T\ (Group, Seller, EF\ DeliverGoods)$, checks whether or not there exists a possibility that every member in the group trusts the seller for delivering the requested goods. We measured the transformation times of the models and formulae along with the verification time in seconds and memory usage in megabyte 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 3 to 66. We considered different numbers of agents to achieve different levels of scalability that makes the problem complex enough to observe significant results. The verification results in Table1 reveal that the number of reachable states reflecting the size of the model increases exponentially with the number of agents, while the space increase is only polynomial. This confirms the PSPACE complexity result. Moreover, the execution time (i.e., the transformation time of both the models and formulae, and the time of the verification process) shows a clear polynomial increase up to 33 agents. After 33, the increase rate is much higher but still polynomial with the number of states, which is inline with the fact that PSPACE=APTIME.

Agents#	States#	Time (sec.)	Memory (MB)
3	5	0.0105	0.22
6	25	0.0112	1.50
9	110	0.0163	3.73
12	550	0.0165	5.02
15	2750	0.0638	16.80
18	13750	0.0864	38.90
.	.	.	.
.	.	.	.
30	7.21875e+006	0.2615	640.44
33	2.44141e+008	0.4076	848.05
.	.	.	.
.	.	.	.
66	2.38419e+015	1841.0002	2,377.50

Table 1: Verification results up to 66 agents

7 Conclusion

The main contributions of the paper are: (1) a new logical language that allows us to express individual, group and propagated trust; (2) a sound transformation procedure that provides a solution of the model checking and satisfiability problems of the logic, along with their complexity analysis. The procedure has been fully implemented and the experiments conducted on a large case study reaching 2.38419e15 states confirmed the theoretical results. For future work, we plan to tackle the run-time verification to investigate the dynamic changes of agents' behavior and their impact on group trust.

References

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

[Chakraborty and Karform, 2012] Partha Sarathi Chakraborty and Sunil Karform. Designing trust propagation algorithms based on simple multiplicative strategy for social networks. *Procedia Technology*, 6:534–539, 2012.

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

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

[Desai *et al.*, 2005] Nirmit Desai, Ashok U. Mallya, Amit K. Chopra, and Munindar P. Singh. Interaction protocols as design abstractions for business processes. *IEEE Trans. Software Eng.*, 31(12):1015–1027, 2005.

[Drawel *et al.*, 2017] Nagat Drawel, Jamal Bentahar, and Elhadi Shakshuki. Reasoning about trust and time in a system of agents. In *the 8th International Conference on Ambient Systems, Networks and Technologies (ANT)*, volume 109 of *Procedia Computer Science*, pages 632–639, 2017.

[Drawel *et al.*, 2018] Nagat Drawel, Jamal Bentahar, Mohamed El-Menshawy, and Amine Laarej. Verifying temporal trust logic using CTL model checking. In *the 20th International Trust Workshop co-located with AAMAS/IJCAI/ECAI/ICML*, pages 62–74, 2018.

[Drawel *et al.*, 2020] Nagat Drawel, Hongyang Qu, Jamal Bentahar, and Elhadi Shakshuki. Specification and automatic verification of trust-based multi-agent systems. *Future Generation Comp. Syst.*, 107:1047–1060, 2020.

[El-Menshawy *et al.*, 2010] Mohamed El-Menshawy, Jamal Bentahar, and Rachida Dssouli. Symbolic model checking commitment protocols using reduction. In *the 8th International Workshop on Declarative Agent Languages and Technologies VIII, DALT*, volume 6619 of *Lecture Notes in Computer Science*, pages 185–203, 2010.

[Emerson and Halpern, 1985] Allen Emerson and Joseph Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *J. Comput. Syst. Sci.*, 30(1):1–24, 1985.

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

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

[Herzig *et al.*, 2012] 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.

[Huang *et al.*, 2019] Xiaowei Huang, Marta Kwiatkowska, and Maciej Olejnik. Reasoning about cognitive trust in stochastic multiagent systems. *ACM Trans. Comput. Log.*, 20(4):21:1–21:64, 2019.

[Jamali and Ester, 2010] Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *the ACM Conference on Recommender Systems, RecSys*, pages 135–142. ACM, 2010.

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

[Liu and Lorini, 2017] 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, 2017.

[Marsh, 1994] Stephen Marsh. *Formalising trust as a computational concept*. PhD thesis, University of Stirling, 1994.

[Nayak *et al.*, 2019] Abhaya Nayak, Kinzang Chhogyal, Aditya Ghose, and Dam Hoa. A value based trust assessment model for multi-agent systems. In *28th International Joint Conference on Artificial Intelligence, IJCAI*, 2019.

[Primiero and Raimondi, 2014] Giuseppe Primiero and Franco Raimondi. A typed natural deduction calculus to reason about secure trust. In *Twelfth Annual International Conference on Privacy, Security and Trust*, pages 379–382. IEEE Computer Society, 2014.

[Primiero, 2016] Giuseppe Primiero. A calculus for distrust and mistrust. In *Trust Management X - 10th IFIP WG 11.11 International Conference, IFIPTM*, volume 473 of *IFIP Advances in Information and Communication Technology*, pages 183–190, 2016.

[Sardana *et al.*, 2018] 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.

[Schnoebelen, 2002] Philippe Schnoebelen. The complexity of temporal logic model checking. In *the 4th conference on Advances in Modal Logic*, pages 393–436, 2002.

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

[Wahab *et al.*, 2018] Omar Abdel Wahab, Jamal Bentahar, Hadi Otrok, and Azzam Mourad. Towards trustworthy multi-cloud services communities: A trust-based hedonic coalitional game. *IEEE Trans. Serv. Comput.*, 11(1):184–201, 2018.