SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks ; **00**:1–14
DOI: 10.1002/sec

RESEARCH ARTICLE

Modeling and Verification of Trust and Reputation Systems

Alessandro Aldini

Department of Base Science and Fundamentals, University of Urbino, Italy
Email: alessandro.aldini@uniurb.it

ABSTRACT

Trust is a basic soft-security condition influencing interactive and cooperative behaviors in online communities. Several systems and models have been proposed to enforce and investigate the role of trust in the process of favoring successful cooperations while minimizing selfishness and failure. However, the analysis of their effectiveness and efficiency is a challenging issue. This paper provides a formal approach to the design and verification of trust infrastructures used in the setting of software architectures and computer networks supporting online communities. The proposed framework encompasses a process calculus of concurrent systems, a temporal logic for trust, and model checking techniques. Both functional and quantitative aspects can be modeled and analyzed, while several types of trust models can be integrated.

KEYWORDS

trust models; cooperative networks; process algebra; temporal logics; model checking

Received . . .

1. INTRODUCTION

Cooperation is a key factor behind the success and the growth of service- and user-centric online communities sharing and exchanging remotely user-generated contents, services, and resources, possibly among unknown participants without the assistance of (trusted) third parties [1]. Trust and reputation systems provide explicitly means to favor cooperation in spite of typical obstacles, like, e.g., selfishness, malicious behaviors, and mistrust towards unknown users. Whenever these systems are based on computational notions of trust, the metrics provided help to estimate the subjective reliance on the ability, integrity, honesty and disposition of each user, possibly making such an estimation available to the community with the aim of making explicit a collective notion of reputation [2]. Even more important, trust and reputation are defined not only to give a representation of the public trustworthiness of users, but also to provide enabling and advantageous conditions for participating actively in the community.

The design and verification of communicating systems and of trust systems is not an easy task as it depends on several, orthogonal aspects. On one hand, architectures of communicating elements are concerned with several issues, ranging from the communication protocols to the dynamic composition of mobile components [3, 4, 5]. On the other hand, analogous difficulties are concerned with the definition of trust systems, which can be centralized or distributed, could rely on the presence of a trusted third party, may use first-hand or second-hand reputation systems employing (non-)linear adjustment mechanisms, can involve explicit (based, e.g., on voting) or implicit evaluation means, and so on. The variety of obstacles to cooperation, like selfishness, lack of motivation, and free-riding, and of attacks, like slandering, self-promoting, and sybil, make the analysis of the effectiveness and efficiency of such systems a challenging issue [6].

In this paper, we propose a formal framework encompassing three fundamental capabilities. First, it supports the functional modeling of the behavior of cooperative, concurrent, and distributed systems by means of a process algebraic architectural description language. Second, it includes a mathematical paradigm for the specification of trust and reputation infrastructures governing the interactions in these systems. On one hand, behavioral modeling and trust modeling are defined separately at the syntactic level, thus facilitating all the design issues. On the other hand, these two different aspects are joined automatically at the semantic level. In particular, the computational notions of trust deriving from the trust model are used either as side conditions enabling specific functional behaviors, or as metrics governing prioritized and/or probabilistic behaviors based on trust. Third, the proposed framework enables the formal verification of the effectiveness and efficiency of the policies based on the trust infrastructures and used

Prepared using secauth.cls [Version: 2010/06/28 v2.00]

to stimulate cooperation and to contrast attacks. The separation of concerns surveyed above favors the execution of sensitivity analysis aimed at evaluating the properties of the system architecture and of the trust model. The formal specification of trust-based properties relies on a temporal logic for trust that extends classical state-based and action-based logics, while the verification of such properties is guaranteed by tool-supported model checking techniques.

The paper is organized as follows. In the rest of this section, we introduce a real-world case study, which accompanies the presentation of the formal framework as a running example, through which we show how to apply our approach in practice. In Section 2, we present the syntax for a calculus of concurrent processes and the syntax for a specification language of trust systems. Then, we define a unifying formal semantics, which subsumes the definition of specific labeled state-transition systems. Quantitative extensions of the semantics are also considered for dealing with trust-based prioritized and probabilistic behaviors. Moreover, to emphasize the flexibility of the proposed approach, we show how to integrate in our framework a well-known formal model of trust, that is Subjective Logic [7]. In Section 3, we define a temporal logic for specifying trust properties that can be model checked through standard techniques, while in Section 4 conclusions about related work and future directions terminate the paper. A preliminary version of this work has been presented at the IFIP Trust Management conference [8].

1.1. Running Example

As a real-world example, we consider an incentive-based cooperation model for wireless and mobile user-centric environments that has been proposed recently [9]. Basically, in such a model we have users providing services, called *requestees*, and recipients of such services, called *requesters*. For the sake of simplicity, in the following we assume that each user behaves either as requester or as requestee. The cooperation model is based on the integration of two mechanisms, trust management and virtual currency, which are used in a process entailing four phases:

1. the requester looks for a service provided by the other members of the community and then sends a request to the chosen requestee;
2. the two parties negotiate parameters and cost of the transaction;
3. if an agreement is reached, then the requestee provides the negotiated service while the requester pays for it;
4. both parties evaluate the quality of experience and provide feedback.

In each phase, trust is used to govern choices and to provide incentives for both parties, e.g., by making offered quality of service and related cost directly dependent on trust. The objective consists of inducing a prosocial attitude to collaboration while isolating selfish and cheating behaviors.

In the following, we abstract away from the details of the specific incentive strategies. Rather, we concentrate on showing how to employ our approach in order to model a scenario like the one surveyed above, specify the underlying trust and reputation models, and perform model checking based sensitivity analysis aiming at demonstrating the influence of each policy and configuration parameter chosen by the involved parties.

2. MODELING TRUST SYSTEMS

In this section, we show how to define separately functional behavior of the system and trust infrastructure. In both cases, we present formal syntax, semantics, and examples related to our running case study. Then, we show how to extend the semantics in a quantitative setting in which trust is used as a metric governing prioritized and probabilistic behaviors. Finally, as an example, we show how to integrate the well-known trust model based on Subjective Logic [7] in such a framework.

From the modeling standpoint, by following principles inspired by architectural description languages [5], we distinguish between *process behaviors*, which describe behavioral patterns, and *process instances*, which represent specific entities exhibiting a certain behavioral pattern. Analogously, we separate the definition of individual entities from the specification of their parallel composition and communication interface. This separation of concerns, which is intended to facilitate compositional reasoning, is applied also to distinguish the description of a system of interacting entities from the specification of the infrastructure governing any interaction based on trust. The objective is a general improvement of usability concerning the modeling issues of the different aspects that come into play in the specification of trust-based distributed systems.

2.1. Modeling Individual Processes

We start the presentation of our trust calculus (TC, for short) by introducing the grammar for the operators used to specify individual process terms, which represent process behaviors modeling behavioral patterns. We denote with $Name$ the set of visible action names, ranged over by $a, b, \ldots$, and we use the fresh name τ to represent invisible, internal actions.

The set of process terms of TC is generated through the following syntax:

$$P ::= \underline{0} \mid a \,.\, P \mid \tau \,.\, P \mid P + Q \mid a.P \mp b.Q \mid B$$

where:

- $\underline{0}$ represents the inactive, terminated process term.
- $a \,.\, P$ (resp., $\tau \,.\, P$) denotes the process term that executes a (resp., τ) followed by the behavior of P.

Security Comm. Networks ; **00**:1–14 © John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

- $P + Q$ represents a nondeterministic choice between process terms P and Q.
- $a.P \mp b.Q$, which is called *trusted choice* operator, denotes an external, guarded choice based on trust.
- B represents a process constant equipped with a defining equation of the form $B \stackrel{\text{def}}{=} P$, which establishes that process constant B behaves as process term P, thus enabling the possibility of defining recursive behaviors.

In the following, we restrict ourselves to consider guarded process terms, i.e., all of the (finite) occurrences of process constants are immediately preceded by the action prefix operator. Before detailing the interpretation of these operators, we introduce the underlying semantic model, which is based on classical labeled transition systems.

Definition 1
A labeled transition system (LTS) is a tuple (S, s_0, L, R) where S is a finite set of states, of which s_0 represents the initial one, L is a finite set of labels, and $R \subseteq S \times L \times S$ is a finitely-branching transition relation, such that $(s, l, s') \in R$ is denoted by $s \stackrel{l}{\longrightarrow} s'$. □

Let $Act = \{\tau\} \cup Name \cup \{\underline{a} \mid a \in Name\}$ be the set of actions. Then, the behavior of process term P is defined by the smallest LTS (S, s_0, L, R) such that S represents the set of process terms of TC (with P being the initial state s_0), $L = Act$, and the transitions in R are obtained through the application of the operational semantics rules of Table I. In the following, we denote with $[\![P]\!]$ the semantics of process term P and we assume Act ranged over by $\alpha, \ldots$.

The semantic rules for prefix, nondeterministic choice, and recursion are standard, while the trusted choice operator establishes that $a.P \mp b.Q$ executes either a followed by P or (a decorated version of) b followed by Q. The intuition is that this operator is used to communicate one of two possible actions to another process and that the choice will be guided by the trust towards such a process: if trust is beyond a certain threshold, then the offered action is a, otherwise it is b. The isolated semantics of this operator offers both actions, as the identity of the interacting process is still unknown, but it uses a decoration to distinguish which action is to be considered in the absence of sufficient trust.

Example 1
With respect to our running example, let us model the behavior of a generic (potentially dishonest) requester, possibly interacting with n requestees, and the behavior of a generic requestee, possibly interacting with m requesters. The process terms describing the requester and requestee behavioral patterns are reported in Table II. □

A process instance, called entity, is an element exhibiting the behavior associated to a process term. The kernel $[\![I]\!]$ of the semantics of an entity I belonging to the behavioral pattern defined by process term P is given by $[\![P]\!]$ (i.e., the semantics of P), in which every action α is renamed to $I.\alpha$ [5]. With abuse of terminology, we say that I is of type P, and we write $I.B$ to specify that the behavior of I in the current state is given by the process term associated to B.

Example 2
In our running example, we consider a system with a single requester, modeled by the entity Req_A of type $Requester$, and three requestees, which are represented by the entities Req_1, Req_2, and Req_3, each one of type $Requestee$. □

2.2. Modeling Trust and Reputation

The execution of the interactions in which every entity in a system is involved depends strictly on the trust infrastructure describing the trust relationships within the community. Hence, before introducing the semantics for interaction, we first define formally such an infrastructure with respect to a set $\mathcal{S}$ of individual entities, by assuming that each entity name is unique to avoid ambiguity.

Let $IName = \{I.a \mid I \in \mathcal{S} \wedge a \in Name\}$ be the set of interacting action names, denoting the actions through which the entities communicate via synchronization, and $\mathbb{T}$ be the domain of trust values. Even if in principle we may adopt any trust domain by adequately defining the semantics of the structures manipulating trust values, for the sake of presentation in the following we assume $\mathbb{T}$ to be a totally ordered set, the maximum (resp., minimum) value of which is denoted by $\top$ (resp., $\bot$).

A trust system is a tuple consisting of a set $\mathcal{S}$ of interacting processes and of the following structures:

- *Trust table* $tt : \mathcal{S} \times \mathcal{S} \to \mathbb{T}$, such that $tt[I; J]$ denotes the direct trust of entity I towards entity J as a result of previous interactions between them. Each row $tt[I; \_]$ is initialized with the dispositional trust of I, which is the initial willingness of I to trust unknown users.
- *Recommendation table* $rt : \mathcal{S} \times \mathcal{S} \times \mathcal{S} \to \mathbb{T}$, such that $rt[I; J; K]$ contains either the trust value recommended by I about J to K, or the special symbol δ to specify that I does not provide recommendations about J to K.
- *Trust threshold function* $tth : \mathcal{S} \to \mathbb{T}$, such that $tth(I)$ represents the minimum amount of trust (towards other entities) required by I to execute a trusted interaction.
- *Trust variation function* $tv : IName \to \mathbb{T}$, such that $tv(I.a)$ is the trust feedback that I associates to the execution of an interaction via action a.
- *Trust function* $tf : \mathcal{S} \times \mathcal{S} \to \mathbb{T}$, such that $tf(I, J)$ computes the trust of I towards J according to a trust formula taking into account direct trust (deriving from the trust table) and reputation (deriving from the recommendation table).

We implicitly assume that the trust system is parameterized with respect to a given type of service, and that several, mutual independent structures are needed when modeling a system that includes different types of

Table I. Semantics rules.

$$\begin{array}{rll} \textit{prefix} & a\,.\,P \xrightarrow{a} P & \tau\,.\,P \xrightarrow{\tau} P \\ \textit{choice} & \dfrac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'} & \dfrac{Q \xrightarrow{\alpha} Q'}{P+Q \xrightarrow{\alpha} Q'} \\ \textit{trusted choice} & a.P \mp b.Q \xrightarrow{a} P & a.P \mp b.Q \xrightarrow{b} Q \\ \textit{recursion} & B \stackrel{\text{def}}{=} P & \dfrac{P \xrightarrow{\alpha} P'}{B \xrightarrow{\alpha} P'} \end{array}$$

Table II. Specification of the requester and requestee behavioral patterns

$$\begin{array}{rcll} \textit{Requester} & \stackrel{\text{def}}{=} & \textit{send\_req\_1}.\textit{Wait}_1 + \cdots + \textit{send\_req\_n}.\textit{Wait}_n & \\ \textit{Wait}_i & \stackrel{\text{def}}{=} & \textit{rec\_accept\_i}.\textit{Service}_i + \textit{rec\_refuse\_i}.\textit{Requester} & 1 \leq i \leq n \\ \textit{Service}_i & \stackrel{\text{def}}{=} & \textit{pay\_i}.\textit{Requester} + \textit{not\_pay\_i}.\textit{Requester} & 1 \leq i \leq n \\ \textit{Requestee} & \stackrel{\text{def}}{=} & \textit{rec\_req\_1}.\textit{Decision}_1 + \cdots + \textit{rec\_req\_m}.\textit{Decision}_m & \\ \textit{Decision}_i & \stackrel{\text{def}}{=} & \textit{send\_accept\_i}.\tau.\,\textit{Payment}_i \mp \textit{send\_refuse\_i}.\textit{Requestee} & 1 \leq i \leq m \\ \textit{Payment}_i & \stackrel{\text{def}}{=} & \textit{rec\_pay\_i}.\textit{Requestee} + \textit{not\_rec\_pay\_i}.\textit{Requestee} & 1 \leq i \leq m \end{array}$$

services, each one requiring separate trust information. In this case, every action must be parameterized as well with respect to the service type, in order to guide each interaction among entities according to the related trust information. We now explain in detail every structure of the trust system.

As far as the trust function tf is concerned, its specification strictly depends on the chosen trust model and, as we will see, it does not affect the definition of the semantics for interacting processes. Function tf may be based on several different methods [10, 11, 12], an example of which will be given with respect to our case study. Similarly, in the following we will also show how to map the Subjective Logic model [7] in our trust system.

Function tth is used to apply a trust-based enabling condition for the execution of any interaction. In fact, every kind of interaction requires a minimum level of trust towards the other party in order to be executed. As we will see, it plays a fundamental role in conjunction with the trust-based choice operator. After the execution of any interaction, function tv is used to express the feedback needed to adjust the trust towards the entity with which the interaction has been completed.

A specific relation exists between the values of the trust table and the values of the recommendation table. Typically, such a relation affects the way in which an entity provides feedback to other entities on the basis of personal experience, and may change drastically depending whether the reputation system is centralized or distributed.

In a centralized scenario, we can envision a trusted third party collecting trust information from all the entities and contributing to construct the reputation of each entity as perceived by the community. As a consequence, every entity requiring a recommendation has access to such a central repository in the same way and obtains the same feedback.

For instance, in a very simple centralized scenario, the recommendation provided by I about J, which is collected centrally and made available to any other entity K, is exactly the trust of I towards J, under the assumption that I had some direct experience with J (otherwise the suggested value would be simply the dispositional trust of I). Formally, let $ct : \mathcal{S} \times \mathcal{S} \to \{0, 1\}$ be the *contact table*, such that $ct[I; J] = 1$ if and only if entities I and J interacted with each other (initially, $ct[I; J] = 0$ for each pair of entities in the set $\mathcal{S}$). Then, the relation between trust table and recommendation table is described by the following equation:

$$rt[I; J; K] = \begin{cases} tt[I; J] & \textit{if } ct[I; J] = 1 \\ \delta & \textit{otherwise} \end{cases} \quad (1)$$

thus assuming that all the entities recommend exactly the trust values resulting from their own experience, if any. Notice that $rt[I; J; K] = rt[I; J; K']$ for every pair of entities K, K', as the recommendation is managed through a central authority sharing such an information with any entity in the same way. It is worth observing that it is possible to model the behavior of an entity I providing inaccurate feedback or cheating deliberately by changing Eq. 1 for specific values of J.

In a distributed scenario, the absence of a centralized trusted third party has two important effects. Firstly, different entities may have access to different information if they are in contact with different neighbors. Secondly, an entity may provide, for the same recommendation,

Security Comm. Networks ; **00**:1–14 © John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

different values to different entities. As a consequence, we may have $rt[I;J;K] \neq rt[I;J;K']$ for $K \neq K'$.

Example 3
In the running example, let trust be a discrete metric such that $\mathbb{T} = [0..10]$. Initially, the trust table is as follows:

	Req_A	Req_1	Req_2	Req_3
Req_A		8	8	8
Req_1	2		2	2
Req_2	3	3		3
Req_3	5	5	5	

The recommendation table is calculated by means of Eq. 1. We do not consider self-promoting behaviors, which, however, could be easily modeled. Moreover, even if we assume a distributed scenario, requester and requestees are connected without any restriction and can communicate with each other. The trust threshold function establishes that the requester issues requests without any reputation constraint, $tth(Req_A) = 0$, and that for each requestee the service trust threshold is equal to requestee's dispositional trust: $tth(Req_1) = 2$, $tth(Req_2) = 3$, and $tth(Req_3) = 5$.

The trust variation function establishes that the requester increases (resp., decreases) by one unit the trust towards any requestee accepting (resp., refusing) a request, namely $tv(Req_A.rec\_accept\_i) = 1$ and $tv(Req_A.rec\_refuse\_i) = -1$ for $1 \leq i \leq 3$. Each requestee increases trust towards the requester in case of paid service, $tv(Req_i.rec\_pay\_1) = 1$ for $1 \leq i \leq 3$. The first two requestees decrease trust by the same amount in case of unpaid service, $tv(Req_i.not\_rec\_pay\_1) = -1$ for $1 \leq i \leq 2$, while the third one is more cautious and applies the maximum penalty, $tv(Req_3.not\_rec\_pay\_1) = -10$. All the other actions do not imply any trust variation.

Finally, the trust formula is abstracted as follows. Let:

$$Rec_{I,J} = \mathcal{S} \backslash \{\{I, J\} \cup \{K \mid rt[K;J;I] = \delta\}\}$$

be the set of entities from which I receives recommendations about J. Then, the trust function is as defined in Table III, where ρ_I represents the risk factor for I, i.e., how much of its trust towards other entities depends on previous direct experience. The term that is multiplied by $1 - \rho_I$ represents the average trust towards J resulting from recommendations provided by third entities. For the three requestees, in the following we assume that the risk factor is equal to 0.5, 0.8, and 0.8, respectively.

In general, notice that the most risky profile is adopted by the first requestee, while the third requestee is characterized by the most cautious behavior [13]. □

2.3. Modeling Interacting Processes

The semantics of interacting entities arises from the parallel composition of a set $\mathcal{S}$ of individual entities following the communication rules established by a synchronization set SS, which is a set of names of the form $I.a\_to\_J.b$. In particular, action $I.a\_to\_J.b$ denotes a synchronization between entities I and J in which I offers action a and J responds with action b. In this view, $I.a$ is the output part of the communication, $J.b$ represents the input counterpart, and $I.a\_to\_J.b$ is the name of the synchronized action.

Example 4
In the running example, the synchronization set for the group of entities $\{Req_A, Req_1, Req_2, Req_3\}$ contains the actions:

$$Req_A.send\_req\_i\_to\_Req_i.rec\_req\_1$$
$$Req_i.send\_accept\_1\_to\_Req_A.rec\_accept\_i$$
$$Req_i.send\_refuse\_1\_to\_Req_A.rec\_refuse\_i$$
$$Req_A.pay\_i\_to\_Req_i.rec\_pay\_1$$
$$Req_A.not\_pay\_i\_to\_Req_i.not\_rec\_pay\_1$$

where $1 \leq i \leq 3$.

The system topology resulting from such a synchronization set reveals that the requester may interact with every requestee, while communications among requestees do not occur, except for the potential exchange of recommendations. Notice that such an exchange is modeled implicitly through the definition of the recommendation policy. According to the trust infrastructure described in the previous section, the system topology has the following effect on the calculation of reputation. Each requestee receives recommendations from any other requestee who has previously interacted with the requester, while the requester does not receive recommendations, meaning that $tf(Req_A, Req_i) = tt[Req_A; Req_i]$ for $1 \leq i \leq 3$ independently of the risk factor chosen by Req_A. □

The interacting semantics of $\mathcal{S}$ is given by the parallel composition of the semantics $[\![I]\!]$ of all the entities $I \in \mathcal{S}$. To facilitate modeling, we represent the composed system simply by the pool $\mathcal{S}$ of interacting entities, without introducing syntactically a specific operator for parallel composition. In the semantics rules, presented in Table IV, $P, P', Q, Q', \ldots$ are process terms representing the local behavior $[\![I]\!]$ of any entity $I \in \mathcal{S}$. A vector of local behaviors – containing as many elements as the number of entities in $\mathcal{S}$, each one expressing the current local behavior of the related entity – represents the global state, denoted by $\mathcal{P}, \mathcal{P}'$, of the composed system. Moreover, $\mathcal{P}[P'/P]$ represents the substitution of P with P' in $\mathcal{P}$. This notation is not ambiguous as every entity name in $\mathcal{S}$ is unique and P, P' express the semantic model of an entity I whose transitions are of the form $I.\alpha$.

The first semantic rule of Table IV establishes that every entity executes its internal actions independently from each other. On the other hand, based on the trust information, interactions among entities occur (or do not occur) and their execution provides feedback, see the second and third rules of Table IV. In order to emphasize the separation of concerns between trust modeling and behavior modeling, the rule premises concerned with the trust structures are specified syntactically as external side

Table III. Trust function for the running example

$$tf(I,J) = \begin{cases} tt[I;J] & \text{if } Rec_{I,J} = \emptyset \\ \rho_I \cdot tt[I;J] \ + \ (1-\rho_I) \cdot \frac{\sum_{K \in Rec_{I,J}} rt[K;J;I]}{|Rec_{I,J}|} & \text{otherwise} \end{cases}$$

conditions. Function $update : \mathbb{T} \times \mathbb{T} \to \mathbb{T}$ formalizes the effect of the interaction upon the trust between the involved parties. To this aim, we assume $\mathbb{T}$ to be a numeric-valued domain and $tv(I.a)$ to represent the variation denoting the trust gain/loss.

The second semantic rule requires entities I and J to enable, respectively, the interacting actions $I.a$ and $J.b$. In such a case, if the communication is allowed, i.e., $I.a\_to\_J.b \in SS$, and if J is trusted enough by I, i.e., $tf(I,J) \geq tth(I)$, then the interaction is executed and, as a post-condition, both I and J update their mutual trust accordingly. We point out that the trust-based premise checks the trust of I towards J (and not vice versa) because the interaction is guided by I. Hence, synchronization is asymmetric and is governed by the entity offering the output counterpart of the communication.

The third semantic rule behaves essentially the same, except that it models the case in which the communication from I to J occurs if I does not trust J enough, see action $I.\underline{a}$ and the premise $tf(I,J) < tth(I)$, in compliance with the semantics of the trusted choice operator. Notice that, in order to consider the case in which the contact table is necessary for the trust calculation, the update $ct[I;J] = 1$ must be added to the premises to keep track of the interaction.

The separation of concerns – between functional behavior modeling and trust representation – is realized at the syntax level and favors independent reasoning and control. All the information and policies concerning trust are not involved syntactically in the specification of the process terms modeling the functional behavior of systems. Instead, they are described in a separate infrastructure, thus facilitating modeling and then sensitivity analysis. Functional behavior and trust management are combined at the semantics level in a fully automatic way governed by the operational semantic rules for parallel composition.

As far as the resulting semantic model is concerned, if trust has a finite value domain, then a concrete treatment of semantics is applied, meaning that the actual instantiations of the trust parameters become part of the formal semantics by contributing to label the states of the labeled transition system expressing the system behavior. Such a condition is achieved whenever trust is a finite, discrete metric, as usual in several trust-based systems [2]. The definition of the formal semantics of a system of interacting entities is based on the extension of LTS taking into account in each state the trust information affecting the application of the semantic rules. In particular, it is worth noticing that the variables of the trust infrastructure needed to determine the enabled transitions are represented by the entries of the trust and recommendation tables. For the sake of presentation, we limit ourselves to consider the case of the trust table, as the extension including both tables is straightforward.

Definition 2
Given a domain V of trust variables and a domain $\mathbb{T}$ of trust values, a trust labeled transition system (*t*LTS) is a tuple (S, s_0, L, R, T, P) where:

- (S, s_0, L, R) is a LTS.
- T is a finite set of trust predicates of the form $v = k$, where $v \in V$ and $k \in \mathbb{T}$.
- $P : S \to 2^T$ is a labeling function that associates a subset of T to each $s \in S$. □

The semantics of a trust system made out of a set $\mathcal{S} = \{I_1, \ldots, I_n\}$ of entities obeying the synchronization set SS and the trust table tt (such that V is the set of its entries) is the smallest *t*LTS satisfying the following conditions:

- each global state $s \in S$ is a n-length vector of process terms modeling the local behavior of the entities I_j, $1 \leq j \leq n$, such that the initial global state s_0 is associated to the vector modeling the initial local state of each entity;
- each trust predicate in T denotes an entry in the trust table tt;
- the labeling function P associates a configuration of the trust table to each state, by assuming that the initial state of the *t*LTS is labeled by the initialization of tt according to the given trust infrastructure;
- the transitions in R are obtained through the application of the operational semantics rules of Table IV;
- $L = IAct$, ranged over by $i, \ldots$, contains internal actions of the form $I_j.\tau$ and interactions in SS.

Therefore, a transition $(s, i, s') \in R$ determines, depending on the current global state s – which encodes both the local state of each entity in the pool and the current trust information – and the action i, the next global state s'.

Example 5
The initial state of the *t*LTS related to our running example is associated to the vector of process terms $[Req_A.Requester,\ Req_1.Requestee,\ Req_2.Requestee,\ Req_3.Requestee]$ and is labeled by the trust predicates defined in the trust table of Example 3. The transitions departing from this state are three, which are labeled with $Req_A.send\_req\_i\_to\_Req_i.rec\_req\_1$, $1 \leq i \leq 3$, respectively. □

Security Comm. Networks ; **00**:1–14 © John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

Table IV. Semantics rules for parallel composition

$$\frac{P \in \mathcal{P} \quad P \xrightarrow{I.\tau} P'}{\mathcal{P} \xrightarrow{I.\tau} \mathcal{P}[P'/P]}$$

$$\frac{P,Q \in \mathcal{P} \quad I.a\_to\_J.b \in SS \quad P \xrightarrow{I.a} P' \quad Q \xrightarrow{J.b} Q'}{\mathcal{P} \xrightarrow{I.a\_to\_J.b} \mathcal{P}[P'/P,Q'/Q]} \quad \begin{array}{l} tf(I,J) \geq tth(I) \\ tt[I;J] = update(tt[I;J], tv(I.a)) \\ tt[J;I] = update(tt[J;I], tv(J.b)) \end{array}$$

$$\frac{P,Q \in \mathcal{P} \quad I.a\_to\_J.b \in SS \quad P \xrightarrow{I.\underline{a}} P' \quad Q \xrightarrow{J.b} Q'}{\mathcal{P} \xrightarrow{I.a\_to\_J.b} \mathcal{P}[P'/P,Q'/Q]} \quad \begin{array}{l} tf(I,J) < tth(I) \\ tt[I;J] = update(tt[I;J], tv(I.a)) \\ tt[J;I] = update(tt[J;I], tv(J.b)) \end{array}$$

$$update(v,k) = \begin{cases} max(\bot, v+k) & if\ k < 0 \\ min(\top, v+k) & if\ k > 0 \end{cases}$$

2.4. Prioritized Choice based on Trust

Trust metrics provide not only means to take binary decisions concerned with the potential execution of an interaction. They can also be used to govern the choice among several alternative interactions. In this section, we add the possibility of defining actions subject to prioritized choice based on trust, by assuming that the trust domain is totally ordered. Intuitively, whenever in the same global state several such interactions are enabled and governed by a given entity I, i.e., they represent trusted communications whose output counterpart is offered by I, then only the trusted interactions with the most trusted entity – among the entities available to interact with I – can be executed, while all the others are pre-empted. In other words, the choice of the entity with which I interacts is prioritized based on entity's reputation.

Inspired by [14], we use the syntactic notation $\bar{a}$ to express an action a subject to prioritized choice. Hence, the syntax of TC is enriched with the production $\bar{a}.P$, whose semantics rule is $\bar{a}.P \xrightarrow{\bar{a}} P$. From now on, we assume that the set Act of action names includes also the set $\{\bar{a} \mid a \in Name\}$.

Basically, priorities come into play in the semantics for interacting processes, where we have to express formally the pre-emption rule, as formalized in the semantics rule defined in Table V. In particular, the rule states that a prioritized interaction governed by entity I towards entity J is enabled only if J is the most trusted entity among those with which I can communicate through a prioritized action. The trust level associated with the most trusted entity available to interact with prioritized actions offered by I in the current global state $\mathcal{P}$ is calculated by function $pri(\mathcal{P}, I)$. If several prioritized interactions are enabled, or interactions that are not subject to the priority mechanism are enabled too, then the choice is nondeterministic. Similarly, the choice among interactions governed by different entities is nondeterministic too.

Example 6
In our running example, let us replace every action $Req_A.send\_req\_i$, where $1 \leq i \leq 3$, with the prioritized counterpart, $Req_A.\underline{send\_req\_i}$.

According to the trust table illustrated in Example 3, in the initial global state, defined in Example 5, the three requestees are chosen nondeterministically as they have the same reputation. Assume, e.g., to follow the path in which Req_1 is chosen, whose first action is $Req_A.send\_req\_1\_to\_Req_1.rec\_req\_1$, which is followed by $Req_1.send\_accept\_1\_to\_Req_A.rec\_accept\_1$, by virtue of which we derive the trust update $tt[Req_A; Req_1] = 9$, see the trust rules illustrated in Example 3. Then, as a consequence of the prioritized choice based on reputation, the next request by Req_A is issued deterministically to Req_1 again. □

2.5. Probabilistic Choice based on Trust

In this section, assuming to deal with a totally-ordered numeric-valued trust domain, we model choices solved probabilistically, where trust towards entities is interpreted as a probabilistic weight used to guide the choice among trusted interactions. To this aim, it is not necessary to enrich the syntax of TC with probabilistic information, which instead is extracted from the trust infrastructure. On the other hand, the semantics for interacting processes, expressed by *t*LTS, must be extended to represent probabilistic moves. The resulting semantic model combines probabilistic and nondeterministic choices. In particular, the choice among actions governed by a given entity I, i.e., its internal actions and the interactions of the form $I.a\_to\_J.b$, is probabilistic, while the choice among actions governed by different entities is left nondeterministic.

Definition 3
A probabilistic LTS (*p*LTS) is a tuple (S, s_0, L, R) where S is a finite set of states, of which s_0 represents the initial one, L is a finite set of labels, and $R \subseteq S \times L \times (0,1] \times S$ is a finitely-branching transition relation, for which there exists a partition L/B induced by an equivalence relation

Table V. Semantics for trust-based prioritized communication

$$\frac{P,Q \in \mathcal{P} \quad I.a\_to\_J.b \in SS \quad P \xrightarrow{I.\bar{a}} P' \quad Q \xrightarrow{J.b} Q'}{\mathcal{P} \xrightarrow{I.a\_to\_J.b} \mathcal{P}[P'/P, Q'/Q]} \quad \begin{array}{l} tf(I,J) = pri(\mathcal{P},I) \geq tth(I) \\ tt[I;J] = update(tt[I;J], tv(I.a)) \\ tt[J;I] = update(tt[J;I], tv(J.b)) \end{array}$$

$$pri(\mathcal{P},I) = \max\{tf(I,J) \mid \exists I.a\_to\_J.b \in SS \wedge \exists P,Q \in \mathcal{P} \wedge \exists P',Q'.\, P \xrightarrow{I.\bar{a}} P' \wedge Q \xrightarrow{J.b} Q'\}$$

B such that for all $s \in S$ and $C \in L/B$:

$$\sum\{\!|\, p \mid \exists l \in C, s' \in S.\, (s,l,p,s') \in R \,|\!\} \in \{0,1\}$$

where the summation returns 0 whenever the multiset is empty. Transition $(s,l,p,s') \in R$ is denoted by $s \xrightarrow{l,p} s'$. □

This definition extends Def. 1 by adding a probability value to the label of each transition. In particular, the outgoing transitions of any given state s are grouped into bundles defined by partition L/B, while the probabilities associated to the transitions departing from s within any bundle C define a probability distribution.

This model has strong similarities with Markov Decision Processes (MDPs), in which the transition relation is $R : S \to 2^{L \times Dist(S)}$, where $Dist(S)$ denotes the set of probability distributions over S. In fact, the two models coincide if we restrict MDPs to assume $R : (S \times L) \to Dist(S)$ - i.e., only one possible distribution is associated to an action name for a given state - and *p*LTS to assume that each $C \in L/B$ is a singleton, thus denoting that a probability distribution is associated to the possible derivatives of the transitions labeled with the same action name.

A trust probabilistic labeled transition system (*tp*LTS) is a *t*LTS whose underlying LTS is replaced by a *p*LTS. The semantics of a trust system made out of a pool $\mathcal{S}$ of entities obeying the synchronization set SS and the trust table tt is defined as in the case of *t*LTS with the following additional condition. For each $I \in \mathcal{S}$, we denote with $IAct_I$ the subset of $IAct$ including only action $I.\tau$ and actions of the form $I.a\_to\_J.b$, for some $a,b \in Name$ and J in the pool. Then, relation B is such that $(i,j) \in B$ if and only if $\exists I \in \mathcal{S}.\, i,j \in IAct_I$. Intuitively, for each global state, all the outgoing transitions representing actions governed by an entity I are grouped into a bundle, while every bundle – one for each entity governing at least an action enabled in the global state – is associated to a probability distribution. In the following, we show how to calculate such a distribution. To this aim, we recall that the trust towards J is used as a weight to compute the probability of any interaction with J. However, trust is not a probability value, thus a normalization must be taken into account. First, we define a function $trust : IAct \to \mathbb{T}$ that returns the trust towards the entity with which a given interaction is completed. By default, we assume that internal actions are associated with the maximum trust level, as we may interpret them as self-interactions. Hence, we have:

$$trust(i) = \begin{cases} \top & \text{if } i = I.\tau \\ tf(I,J) & \text{if } i = I.a\_to\_J.b \end{cases}$$

Then, the overall trust associated to the bundle of actions governed by I in a given global state $\mathcal{P}$ is calculated as follows:

$$\mathcal{T}(\mathcal{P},I) = \sum\{\!|\, trust(i) \mid i \in IAct_I \wedge \exists \mathcal{P}'.\, \mathcal{P} \xrightarrow{i} \mathcal{P}' \,|\!\}$$

The execution probability of an action $i \in IAct_I$ in a global state $\mathcal{P}$ is calculated as the ratio between its associated trust level $trust(i)$ and the overall trust $\mathcal{T}(\mathcal{P},I)$. Formally, the *tp*LTS modeling the interacting semantics of a pool $\mathcal{S}$ is obtained by replacing transition $\mathcal{P} \xrightarrow{i} \mathcal{P}'$ in the conclusion of every semantics rule of Table IV with $\mathcal{P} \xrightarrow{i,trust(i)/\mathcal{T}(\mathcal{P},I)} \mathcal{P}'$, where $i \in IAct_I$.

Example 7
Let us apply the new semantics to our running example. For every global state including the local state $Req_A.Requester$, the bundle of interactions governed by entity Req_A, which contains actions $Req_A.send\_req\_i$, with $1 \leq i \leq 3$, is associated to a probability distribution expressing that a requestee is chosen probabilistically on the basis of requestee's reputation. For instance, according to the trust table illustrated in Example 3, in the initial global state, defined in Example 5, the three requestees are chosen with probability $\frac{8}{24}$ each. □

Finally, we point out that the probabilistic model of choice can coexist with the prioritized model of choice. Basically, if both are applied, for each entity I the related bundle of actions enabled in a certain global state is determined by following the priority based mechanism, and only later the associated probability distribution is calculated.

2.6. Integration with Subjective Logic

Even if in the previous sections we have dealt with numeric-valued trust opinions, several different trust models can be embedded in the proposed framework, depending on the way in which the structures of the trust system of Section 2.2 are defined. To put in evidence such a level of flexibility, here we show how to integrate (a fragment of) Subjective Logic [7] in our framework.

In Subjective Logic, trust is expressed through belief triples of the form (b,d,u), where the three components

Security Comm. Networks ; **00**:1–14 © John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

sum up to 1 and represent belief, disbelief, and uncertainty, respectively. Several operators are defined to manage trust opinions, among which we consider *consensus* (denoted by $\oplus$), which formalizes the aggregation of opinions, and *discounting* (denoted by $\otimes$), which formalizes trust chaining, which is useful to combine trust towards recommenders with trust reported by their recommendations.

The embedding requires the following modifications in the syntax and semantics of the trust system. First, as an immediate consequence of the representation of trust opinions, the trust values in the domain $\mathbb{T}$ become belief triples. The most interesting effect of this variation is on the trust variation function tv, which associates each interaction to the related trust opinion. For instance, in a very simple scenario in which an interaction denotes either success, modeled through the triple $(\frac{1}{2}, 0, \frac{1}{2})$, or failure, modeled through the triple $(0, \frac{1}{2}, \frac{1}{2})$, then these two triples represent the actual codomain for function tv. In general, more specific triples can be related to interactions denoting intermediate levels of satisfaction between these two limiting results. Then, the trust function tf, which merges personal experience and recommendations, is defined by taking the consensus of the user own trust and every discounted recommended trust opinion:

$$tf(I,J) = tt[I;J] \oplus \sum_{K \in Rec_{I,J}} tt[I;K] \otimes rt[K;J;I]$$

where summation is over the operator $\oplus$ and no risk factor is used.

Finally, we need to update accordingly the semantics of the operations performed in the side conditions of the semantics rules of Table IV. In particular, the comparison operators $\geq$ and $\leq$ are extended to deal with belief triples. The simplest interpretation of this extension consists of extracting the belief components of the triples, which are then compared through the classical comparison operators. Similarly, the definition of function $update$, which formalizes the adjustment of trust by virtue of a new opinion, is given in terms of the consensus operator:

$$update(t_1, t_2) = t_1 \oplus t_2$$

By virtue of these assumptions, the obtained result is an integrated model in which functional behavior of the system and trust are defined, respectively, in terms of a process algebraic specification language and Subjective Logic. As we will see in the next section, an integrated model such as this can be analyzed dynamically through model checking techniques.

3. MODEL CHECKING TRUST

In this section, we show how to perform model checking based verification whenever the formal semantics of a trust system of interacting processes is based on *t*LTS. First, we notice that the *t*LTS model represents an instance of doubly labeled transition systems [15], and of Kripke transition systems [16]. Hence, it is possible to employ temporal logics for such systems in order to define a trust logic for specifying both conditions based on the actions labeling the transitions and requirements based on the trust information labeling the states. We call such a language trust temporal logic (TTL). In particular, TTL embodies features of the classical branching-time state-based Computation Tree Logic [17] and of its action-based variant ACTL [18].

TTL includes the definition of state formulas, which are applied to states of a *t*LTS, and path formulas, which are applied to sequences of transitions of a *t*LTS. The syntax of TTL is defined as follows:

$$\begin{array}{rcl} \Phi & ::= & true \mid i \mid v \geq k \mid \Phi \wedge \Phi \mid \neg\Phi \mid A\pi \mid E\pi \\ \pi & ::= & \Phi \,{}_{\mathcal{A}_1}U\, \Phi \mid \Phi \,{}_{\mathcal{A}_1}U_{\mathcal{A}_2}\, \Phi \end{array}$$

where $v = tt[I;J]$, with I and J entity names, $k \in \mathbb{T}$, $i \in IAct$, and $\mathcal{A}_1, \mathcal{A}_2 \subseteq IAct$. Inspired by other logics merging action/state-based predicates [19], atomic propositions are either actions or trust predicates of the form $v \geq k$, where variable v denotes any entry of the trust table and k belongs to the trust domain. State formulas are ranged over by Φ. Intuitively, a state satisfies the atomic proposition i if it enables a transition labeled with i, while it satisfies the atomic proposition $v \geq k$ if it is labeled with a trust predicate that assigns to v a value greater than (or equal to) k. Composite state formulas are obtained through the classical connectives. The operators A and E denote the universal and existential path quantifiers. A state satisfies $A\pi$ (resp., $E\pi$) if *every* path (resp., *at least one* path) departing from such a state satisfies the path formula π. Path formulas are ranged over by π, while U is the indexed until operator. Intuitively, a path satisfies the until formula $\Phi \,{}_{\mathcal{A}_1}U\, \Phi'$ if the path visits a state satisfying Φ', and visits states satisfying Φ while performing only actions in $\mathcal{A}_1$ until that point. Similarly, the until formula $\Phi \,{}_{\mathcal{A}_1}U_{\mathcal{A}_2}\, \Phi'$ is satisfied by a path if the path visits a state satisfying Φ' after performing an action in $\mathcal{A}_2$, and visits states satisfying Φ while performing only actions in $\mathcal{A}_1$ until that point. We observe that a path satisfying $\Phi \,{}_{\mathcal{A}_1}U_{\mathcal{A}_2}\, \Phi'$ must include a transition to a state satisfying Φ', while this is not required for $\Phi \,{}_{\mathcal{A}_1}U\, \Phi'$ if the initial state of the path satisfies Φ'.

Similarly as argued in the previous section, if the states of the *t*LTS include reputation-based information deriving from the recommendation table, we can enrich TTL with reputation-based state predicates.

Now, let us define formally some notion about paths with respect to a *t*LTS (S, s_0, L, R, T, P). A path σ is a (possibly infinite) sequence of transitions of the form: $s_0 \xrightarrow{i_0} s_1 \ldots s_{j-1} \xrightarrow{i_{j-1}} s_j \ldots$ where $s_{j-1} \xrightarrow{i_{j-1}} s_j \in R$ for each $j > 0$. Every s_j in the path is denoted by $\sigma(j)$. Moreover, let $s_j \xrightarrow{\mathcal{A}} s_{j+1}$ if and only if $i_j \in \mathcal{A} \subseteq L$. We denote with $Path(s)$ the set of paths starting in state

Table VI. Semantics of TTL

$s \models true$	*holds always*
$s \models v \geq k$	*iff* $(v = k') \in P(s) \wedge k' \geq k$
$s \models i$	*iff* $\exists s' : s \xrightarrow{i} s' \in R$
$s \models \Phi \wedge \Phi'$	*iff* $s \models \Phi$ *and* $s \models \Phi'$
$s \models \neg\Phi$	*iff* $s \not\models \Phi$
$s \models A\pi$	*iff* $\forall \sigma \in Path(s) : \sigma \models \pi$
$s \models E\pi$	*iff* $\exists \sigma \in Path(s) : \sigma \models \pi$
$\sigma \models \Phi\ {}_{\mathcal{A}_1}U\ \Phi'$	*iff* $\exists k \geq 0 :$ $\sigma(k) \models \Phi' \wedge (for\ all\ 0 \leq i < k : \sigma(i) \models \Phi \wedge \sigma(i) \xrightarrow{\mathcal{A}_1} \sigma(i+1))$
$\sigma \models \Phi\ {}_{\mathcal{A}_1}U_{\mathcal{A}_2}\ \Phi'$	*iff* $\exists k > 0 :$ $\sigma(k) \models \Phi' \wedge (for\ all\ 0 \leq i < k-1 : \sigma(i) \models \Phi \wedge$ $\sigma(i) \xrightarrow{\mathcal{A}_1} \sigma(i+1)) \wedge \sigma(k-1) \models \Phi \wedge \sigma(k-1) \xrightarrow{\mathcal{A}_2} \sigma(k)$

$s \in S$. Then, the formal semantics of TTL is as shown in Table VI.

TTL can be mapped to the logic UCTL [15], for which an efficient on-the-fly model checking algorithm is implemented. The unique non-trivial difference between the two logics is that TTL allows for action-based atomic propositions, while UCTL does not. The atomic proposition i of TTL can be represented through the UCTL until operator as follows. Denoted with *false* the formula $\neg true$, then i is expressed by the formula $E(false\ {}_{\emptyset}U_{\{i\}}\ true)$, which establishes that from the current state a transition labeled with i is enabled that leads to a state satisfying the atomic formula $true$, i.e., given s the current state, it holds that $\exists s' : s \xrightarrow{i} s' \in R$.

Finally, we provide two flavors of classical operators like *next* (X), *eventually* (F), and *always* (G), depending on the kind of until operator used. To this end, we introduce the following notations:

$$\begin{aligned} X\Phi &= false\ {}_{\emptyset}U_{IAct}\ \Phi \\ X_{\mathcal{A}_1}\Phi &= false\ {}_{\emptyset}U_{\mathcal{A}_1}\ \Phi \end{aligned}$$

$$\begin{aligned} EF\Phi &= E(true\ {}_{IAct}U\ \Phi) \\ EF_{\mathcal{A}_1}\Phi &= E(true\ {}_{IAct}U_{\mathcal{A}_1}\ \Phi) \end{aligned}$$

$$\begin{aligned} AF\Phi &= A(true\ {}_{IAct}U\ \Phi) \\ AF_{\mathcal{A}_1}\Phi &= A(true\ {}_{IAct}U_{\mathcal{A}_1}\ \Phi) \end{aligned}$$

$$\begin{aligned} EG\Phi &= \neg AF\neg\Phi \\ EG_{\mathcal{A}_1} &= \neg AF_{IAct-\mathcal{A}_1}\ true \end{aligned}$$

$$\begin{aligned} AG\Phi &= \neg EF\neg\Phi \\ AG_{\mathcal{A}_1} &= \neg EF_{IAct-\mathcal{A}_1}\ true \end{aligned}$$

For instance, $EG\Phi$ holds in s if there exists a path in $Path(s)$ every state of which (including s) satisfies Φ, while $EG_{\mathcal{A}_1}$ holds in s if there exists a path in $Path(s)$ every transition of which is labeled with an action in $\mathcal{A}_1$.

Example 8
With respect to our running example, we focus on the comparison between the two limiting profiles, i.e., risky and cautious, which characterize the behavior of the requestees. After adequate translation of the model, the following properties have been recast and checked both in PRISM [20] and in NuSMV [21].

The first parameter under analysis is the risk factor and the related impact upon the capability of being influenced by recommendations. To this aim, we formulate the following condition to check. Can the risky requestee accept a request without sufficient direct trust towards the requester? The related property is stated formally as follows:

$$EF(tt[Req_1; Req_A] < 2 \wedge Req_1.send\_accept\_1\_to\_Req_A.rec\_accept\_1).$$

The formula schema EF expresses the eventuality of reaching a state satisfying the following expression, i.e., the conjunction of two predicates: the state predicate $tt[Req_1; Req_A] < 2$ describes the trust condition, while the action predicate $Req_1.send\_accept\_1\_to\_Req_A.rec\_accept\_1$ formalizes the behavior to observe in the state. The property is satisfied, because by virtue of the assumption $\rho_{Req_1} = 0.5$, positive recommendations provided to the risky requestee can balance (and overcome the effect of) negative direct experiences. The same property can be recast in the case of the cautious requestee:

$$EF(tt[Req_3; Req_A] < 5 \wedge Req_3.send\_accept\_1\_to\_Req_A.rec\_accept\_3)$$

which is not satisfied, thus confirming the prudent behavior of this requestee.

An interesting analysis concerns the consequences of a malicious behavior of the requester. The following property:

$$AG(Req_A.not\_pay\_3\_to\_Req_3.not\_rec\_pay\_1 \rightarrow AG(\neg Req_3.send\_accept\_1\_to\_Req_A.rec\_accept\_3))$$

is satisfied, thus establishing that after experiencing a cheating behavior of the requester (see action

Security Comm. Networks ; **00**:1–14 © John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

$Req_A.not\_pay\_3\_to\_Req_3.not\_rec\_pay\_1$) then the cautious requestee does not trust the requester anymore (in every future state it holds that the action $Req_3.send\_accept\_1\_to\_Req_A.rec\_accept\_3$ cannot be enabled). By replacing the cautious requestee with the risky requestee, the corresponding property is violated. Actually, not very surprisingly, even the following property is satisfied:

$$EF(EG_{\mathcal{A}_1})$$

where $\mathcal{A}_1$ is the set of actions:

$$\{Req_A.send\_req\_1\_to\_Req_1.rec\_req\_1,\\ Req_1.send\_accept\_1\_to\_Req_A.rec\_accept\_1,\\ Req_1.\tau,\\ Req_A.not\_pay\_1\_to\_Req_1.not\_rec\_pay\_1\}.$$

This means that a certain point can be reached starting from which the requester can obtain services from the risky requestee infinitely often without paying for any of them. This situation is an immediate consequence of the first property, which demonstrates that the direct mistrust of the risky requestee towards the requester is not sufficient to exclude the cheating behavior.

On the other hand, let us now consider a completely honest requester. This variant can be obtained either by eliminating from requester's process terms any action not_pay_i or, even better, by removing the related actions from the synchronization set SS. In this scenario, we verify whether eventually a point is reached starting from which every issued request is accepted:

$$EF(AG_{IAct-\mathcal{A}_1})$$

where $\mathcal{A}_1$ is the set of actions:

$$\{Req_1.send\_refuse\_1\_to\_Req_A.rec\_refuse\_1,\\ Req_2.send\_refuse\_1\_to\_Req_A.rec\_refuse\_2,\\ Req_3.send\_refuse\_1\_to\_Req_A.rec\_refuse\_3\}.$$

Such a property holds as expected.

Separating functional behavior modeling and trust management specification allows for a clear verification of the impact of trust policies upon specific properties by simply adjusting the trust parameters of certain entities. For instance, let us replace the cautious requestee with a paranoid requestee characterized by strict trust requirements, and then let us consider the capability of such an entity of accepting services. To this aim, we adjust the trust infrastructure only, by tuning ρ, tth, and dispositional trust for entity Req_3. As an example, with $\rho_{Req_3} = 0.8$, $tth(Req_3) = 5$ (as for the cautious requestee), and dispositional trust less than 4, we obtain that the following property is not satisfied:

$$EF(Req_3.send\_accept\_1\_to\_Req_A.rec\_accept\_3)$$

meaning that the paranoid requestee does not serve any request. The property turns out to hold if the dispositional trust is set to 4, in which case we also observe that, given $\mathcal{A}_1 = \{Req_3.send\_accept\_1\_to\_Req_A.rec\_accept\_3\}$, then the property:

$$E((tt[Req_1; Req_A] < 10 \wedge\\ tt[Req_2; Req_A] < 10)\ {}_{IAct}U_{\mathcal{A}_1}\ true)$$

does not hold. More precisely, at least one of the other two requestees must recommend top trust towards the requester in order to allow the paranoid requestee to accept a request.

Finally, let us consider a coalition attack by two requestees against the third one. The condition of interest is formulated as follows. Can malicious requestees provide false feedback to the risky requestee thus avoiding her/him from accepting any request? To this aim, it is sufficient to extend the recommendation table by setting $rt[Req_2; Req_A; Req_1] = rt[Req_3; Req_A; Req_1] = 0$ (while all the other entries are as usual), and then check the TTL formula:

$$\neg EF(Req_1.send\_accept\_1\_to\_Req_A.rec\_accept\_1).$$

This property is satisfied, thus revealing the effectiveness of the attack. By tuning the dispositional trust of the risky requestee, we observe that the attack can be avoided if and only if such a parameter is set to at least 4. On the other hand, if the false feedback is provided by Req_3 only, Req_1 can accept requests (even without altering her/his dispositional trust), but only after a successful interaction between Req_2 and Req_A. In this case, we have also verified that extremely positive recommendations by Req_2 ($rt[Req_2; Req_A; Req_1] = 10$) protect Req_1 from coalition attacks of (up to) 4 malicious requestees. □

3.1. Probabilistic Model Checking

Extending model checking to the analysis of probabilistic systems, ranging from discrete- or continuous-time Markov chains to Markov decision processes, received a lot of attention in the literature, and a number of probabilistic model checking techniques as well as probabilistic temporal logics have been proposed to cover such an extension. In this section, we briefly discuss how to extend TTL to deal with the analysis of *pt*LTS, by following the approach implemented in the PRISM software tool [22] to the analysis of MDPs.

While in purely probabilistic systems, like DTMCs, verifying quantitative properties reduces to construct and analyze the probability space over the set of (infinite) paths, in the setting of MDPs the same approach is possible only once any form of nondeterminism has been solved. Every possible resolution of nondeterminism defines an *adversary*, so that verifying quantitative properties of MDPs equates to evaluate the behavior obtained under every adversary, possibly concentrating on determining the best- or worst-case behavior. Based on the observations of Section 2.5, while in MDPs the adversary is responsible for choosing an action name in each state, in the setting of *pt*LTS the adversary is responsible for choosing an entity in each state. In fact, in the semantics of TC extended

with trust-based probabilistic choice, the unique source of nondeterminism derives from the concurrent execution of several entities.

Formally, a path σ through a *pt*LTS is a (possibly infinite) sequence of the form:

$$s_0 \stackrel{i_0,p_0}{\longrightarrow} s_1 \ldots s_{j-1} \stackrel{i_{j-1},p_{j-1}}{\longrightarrow} s_j \ldots$$

where $s_{j-1} \stackrel{i_{j-1},p_{j-1}}{\longrightarrow} s_j \in R$ for each $j > 0$. Notice that a path solves both the nondeterministic choice among the entities enabled to perform a move at each step and the probabilistic choice among the actions enabled by the chosen entity. In particular, the nondeterministic choice is governed by the adversary α, which is a function mapping every finite path $\sigma \in Path(s)$, whose last state is denoted by $last(\sigma)$, to the bundle of transitions associated to the chosen entity I, namely:

$$\{last(\sigma) \stackrel{i,p}{\longrightarrow} s' \in R \mid i \in IAct_I\}$$

Then, it is possible to define the probability measure $Prob_s^\alpha$ over the set of paths departing from state s and guided by adversary α. We skip the details of such a theory, which, e.g., is presented in [22], and we concentrate on the probabilistic extension of TTL, which is inspired by Probabilistic CTL. Syntactically, the TTL path quantifiers $A\pi$ and $E\pi$, which guard any path formula, are replaced by the probabilistic path operator:

$$P_{\bowtie p}\pi$$

where $\bowtie \in \{<, \leq, =, \geq, >\}$ and $p \in [0, 1]$. Informally, a state satisfies $P_{\bowtie p}\pi$ if, under each adversary, the probability of taking any path from s satisfying the path formula π respects the condition $\bowtie p$. Formally:

$$s \models P_{\bowtie p}\pi \text{ } \textit{iff} \text{ } Prob_s^\alpha(\{\sigma \in IPath(s) \mid \sigma \models \pi\}) \bowtie p$$

for all adversaries α, assumed that $IPath(s)$ is the set of infinite paths starting from s and $Prob_s^\alpha(\{\sigma \in IPath(s) \mid \sigma \models \pi\})$ is the unique measure denoting the probability associated to the paths in $IPath(s)$ that satisfy π under the adversary α. With this in view, the same theory known for reasoning about MDPs [22] can be applied in our context.

Example 9
Let us apply the probabilistic interpretation of semantics in the setting of the running example. By employing the functionalities of the model checker PRISM, we can estimate the maximum/minimum probability of satisfying the properties considered in Example 8, based on the best/worst strategy followed by the adversary. For instance, the maximum probability of satisfying the first reachability property discussed in Example 8 is 1. Then, at the end of Example 8, we have analyzed the behavior of a paranoid requestee replacing the third, cautious requestee, by revealing the requirements needed to satisfy the following property:

$$EF(Req_3.send\_accept\_1\_to\_Req_A.rec\_accept\_3).$$

In the probabilistic setting, we can now observe more specifically that the maximum probability of satisfying the path condition:

$$F(Req_3.send\_refuse\_1\_to\_Req_A.rec\_refuse\_3)$$

(i.e., reaching a state in which the paranoid requestee does not accept the request) is $p = 83.7\%$, thus confirming quantitatively the non-cooperative character of such a profile. □

4. CONCLUSION AND RELATED WORK

This paper describes a process algebraic framework in which trust modeling and system specification are combined and model checking techniques are applied to verify the effects of trust opinions and related parameters upon cooperation in concurrent and distributed systems.

In the literature, formal methods have been used successfully to model and analyze trust and trust relationships [23, 24, 25, 26, 27, 28, 29, 30]. However, usually these works do not integrate trust modeling with concurrent/distributed systems modeling in a unifying formal framework. Theoretical analysis of cooperation strategies is proposed by employing formal approaches like, e.g., game theory [31], and the theory of semirings [32]. The analysis of trust chains is investigated also in a process algebraic setting, either with a specific focus on access control policies [33], or by employing equivalence checking based analysis [34].

A natural extension of our work is concerned with the use of reward structures expressing metrics that can be related to trust. This is the case, e.g., of the service cost, as well as any other parameter related to the quality of experience that may be influenced by (or may affect) trust. Then, similarly as done in the setting of quantitative model checking [35, 19, 36, 22], we can employ the probabilistic version of TTL extended with rewards to estimate the tradeoff existing between trust and other metrics, which is necessary to evaluate mixed cooperation incentive strategies [13].

Acknowledgment: This work has been partially supported by the MIUR project CINA (Compositionality, Interaction, Negotiation, Autonomicity for the future ICT society).

REFERENCES

1. Aldini A, Bogliolo A ((eds.)). *User-Centric Networking – Future Perspectives*. Lecture Notes in Social Networks, Springer, 2014.

2. Jøsang A. Trust and reputation systems. *Foundations of Security Analysis and Design IV (FOSAD'07)*, *LNCS*, vol. 4677, Aldini A, Gorrieri R (eds.). Springer, 2007; 209–245.
3. Bernardo M, Inverardi P ((eds.)). *Formal Methods for Software Architectures*, *LNCS*, vol. 2804. Springer, 2003.
4. Shaw M, Clements P. The golden age of software architecture. *IEEE Software* 2006; **23**(2):31–39.
5. Aldini A, Bernardo B, Corradini F. *A Process Algebraic Approach to Software Architecture Design*. Springer, 2010.
6. Marmol FG, Perez GM. Security threats scenarios in trust and reputation models for distributed systems. *Computers and Security* 2009; **28**(7):545–556.
7. Jøsang A. A logic for uncertain probabilities. *Int. Journal of Uncertainty, Fuzziness, and Knowledge-Based Systems* 2001; **9**(3):279–311.
8. Aldini A. A calculus for trust and reputation systems. *8th IFIP WG 11.11 Int. Conf. on Trust Management*, *IFIP AICT*, vol. 430, Zhou J, *et al.* (eds.), Springer, 2014; 173–188.
9. Bogliolo A, *et al.*. Virtual currency and reputation-based cooperation incentives in user-centric networks. *8th Int. Wireless Communications and Mobile Computing Conf. (IWCMC'12)*, IEEE, 2012; 895–900.
10. Kamvar SD, Schlosser MT, Garcia-Molina H. The eigentrust algorithm for reputation management in p2p networks. *12th Conf. on World Wide Web (WWW'03)*, ACM, 2003; 640–651.
11. Zhou R, Hwang K. Powertrust: a robust and scalable reputation system for trusted peer-to-peer computing. *Transactions on Parallel and Distributed Systems* 2007; **18**(4):460–473.
12. Zhang Y, Lin L, Huai J. Balancing trust and incentive in peer-to-peer collaborative system. *Journal of Network Security* 2007; **5**:73–81.
13. Aldini A, Bogliolo A. Modeling and verification of cooperation incentive mechanisms in user-centric wireless communications. *Security, Privacy, Trust, and Resource Management in Mobile and Wireless Communications*, Rawat D, Bista B, Yan G (eds.). IGI Global, 2014; 432–461.
14. Cleaveland R, Lüttgen G, Natarajan V. Priority and abstraction in process algebra. *Information and Computation* 2007; **205**(9):1426–1458.
15. ter Beek M, Fantechi A, Gnesi S, Mazzanti F. An action/state-based model-checking approach for the analysis of communication protocols for service-oriented applications. *12th Workshop on Formal Methods for Industrial Critical Systems (FMICS'07)*, *LNCS*, vol. 4916, Springer, 2008; 133–148.
16. Chaki S, Clarke E, Ouaknine J, Sharygina N, Sinha N. State/event-based software model checking. *Conf. on Integrated Formal Methods (IFM'04)*, *LNCS*, vol. 2999, Springer, 2004; 128–147.
17. Clarke E, Emerson E, Sistla A. Automatic verification of finite state concurrent systems using temporal logic specifications. *Transactions on Programming Languages and Systems* 1986; **8**(2):244–263.
18. De Nicola R, Vaandrager F. Actions versus state based logics for transition systems. *Semantics of Systems of Concurrent Processes - LITP Spring School on Theoretical Computer Science*, *LNCS*, vol. 469, Guessarian I, *et al.* (eds.). Springer, 1990; 407–419.
19. Aldini A, Bernardo B, Sproston J. Performability measure specification: Combining csrl and msl. *16th Workshop on Formal Methods for Industrial Critical Systems (FMICS'11)*, *LNCS*, vol. 6959, Springer, 2011; 165–179.
20. Kwiatkowska M, Norman G, Parker D. Prism 4.0: verification of probabilistic real-time systems. *23rd Int. Conf. on Computer Aided Verification (CAV'11)*, *LNCS*, vol. 6806, Springer, 2011; 585–591.
21. Cimatti A, *et al.*. Nusmv 2: An opensource tool for symbolic model checking. *14th Conf. on Computer Aided Verification (CAV'02)*, *LNCS*, vol. 2404, Springer, 2002; 359–364.
22. Forejt V, Kwiatkowska M, Norman G, Parker D. Automated verification techniques for probabilistic systems. *Formal Methods for Eternal Networked Software Systems*, *LNCS*, vol. 6659, Bernardo M, Issarny V (eds.). Springer, 2011; 53–113.
23. Jøsang A. Subjective logic book 2013. URL http://folk.uio.no/josang/papers/subjective_logic.pdf.
24. He F, Zhang H, Wang H, Xu M, Yan F. Chain of trust testing based on model checking. *2nd Int. Conf. on Networks Security Wireless Communications and Trusted Computing (NSWCTC'10)*, IEEE, 2010; 273–276.
25. Huang J. A formal-semantics-based calculus of trust. *Internet Computing* 2010; **14**(5):38–46.
26. Muller T. Semantics of trust. *7th Int. Workshop on Formal Aspects in Security and Trust (FAST'10)*, *LNCS*, vol. 6561, Springer, 2010; 141–156.
27. Trcek D. A formal apparatus for modeling trust in computing environments. *Mathematical and Computer Modelling* 2009; **49**(1–2):226–233.
28. Huang J, Nicol D. A calculus of trust and its application to pki and identity management. *8th Symposium on Identity and Trust on the Internet (IDtrust'09)*, ACM, 2009; 23–37.
29. Giorgini P, Massacci F, Mylopoulos J, Zannone N. Requirements engineering meets trust management - model, methodology, and reasoning. *iTrust04*, *LNCS*, vol. 2995, Springer, 2004; 176–190.
30. Nielsen M, Krukow K. Towards a formal notion of trust. *5th ACM SIGPLAN Conf. on Principles and Practice of Declaritive Programming (PPDP'03)*, ACM, 2003; 4–7.

31. Li Z, Shen H. Game-theoretic analysis of cooperation incentives strategies in mobile ad hoc networks. *Transactions on Mobile Computing* 2012; **11**(8):1287–1303.
32. Theodorakopoulos G, Baras JS. On trust models and trust evaluation metrics for ad hoc networks. *Journal on Selected Areas in Communications* 2006; **24**:318–328.
33. Martinelli F. Towards an integrated formal analysis for security and trust. *7th Conf. on Formal Methods for Open Object-Based Distributed Systems (FMOODS'05)*, *LNCS*, vol. 3535, Springer, 2005; 115–130.
34. Carbone M, Nielsen M, Sassone V. A calculus of trust management. *Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'04)*, *LNCS*, vol. 3328, Springer, 2004; 161–173.
35. Baier C, Cloth L, Haverkort B, Hermanns H, Katoen JP. Performability assessment by model checking of markov reward models. *Formal Methods in System Design* 2010; **36**:1–36.
36. Chen T, Forejt V, Kwiatkowska M, Parker D, Simaitis A. Automatic verification of competitive stochastic systems. *Formal Methods in System Design* 2013; **43**(1):61–92.

DOI: 10.1002/sec
Prepared using secauth.cls