A Language for Trust Modelling

Tim Muller Jie Zhang Yang Liu

Nanyang Technological University, Singapore

Abstract

The computational trust paradigm supposes that it is possible to quantify trust relations that occur within some software systems. The paradigm covers a variety of trust systems, such as trust management systems, reputation systems and trust-based security systems. Different trust systems have different assumptions, and various trust models have been developed on top of these assumptions Typically, trust models are incomparable, or even mutually unintelligible; as a result their evaluation may be circular or biased. We propose a unified language to express the trust models and trust systems. Within the language, all trust models are comparable, and the problem of circularity or bias is mitigated. Moreover, given a complete set of assumptions in the language, a unique trust model is defined.

1 Introduction

People interact over the internet. Opportunities may arise for people to betray others. Hence, people need to trust over the internet. Computational trust is a paradigm that deals with quantifying trust, mitigating risks and selecting trustworthy agents [9].

Within the computational trust paradigm, there are different *trust systems*. A trust system, here, refers to online systems involving trust values. A typical centralised trust system, for example, collects ratings, aggregates these into a single score, and distributes these scores. The effectiveness of a trust system is not straightforward to ascertain. What is, e.g., the correct way to aggregate the ratings into a single score, and what does it mean? In order to interpret the trust values and determine their validity, a trust model is required.

Within the computational trust paradigm, there are also different *trust models*. A trust model dictates the meaning of trust values, and what appropriate trust values are. A trust model can be used to evaluate trust systems, for example for simulations to measure the effectiveness of a trust system. Different trust systems can be compared using a sufficiently general trust model. However, there is no fully general trust model that everyone agrees on.

In fact, there probably cannot such a fully general trust model, since different trust systems require different assumptions. For example, some trust models assume trust is transitive (A trusts B and B trusts C implies A trusts C) to some extent, and others do not [1]. The variety of assumptions that underlie different trust models (which underlie different trust systems) leads to confusion: Implicit assumptions may cause misinterpretation. Overly strong assumptions may yield meaningless results. Assumptions may be shared between the system and its (experimental) analysis, making its analysis pointless. Two systems may have similar assumptions, which unexpectedly lead to fundamentally different conclusions.

We propose a shared language for these assumptions. The computational trust paradigm is captured in three core *principles* (see Section 3). We assert that every trust model should adhere to these principles. Moreover,

Copyright © by the paper's authors. Copying permitted only for private and academic purposes.

In: J. Zhang, R. Cohen, and M. Sensoy (eds.): Proceedings of the 18th International Workshop on Trust in Agent Societies, Singapore, 09-MAY-2016, published at http://ceur-ws.org

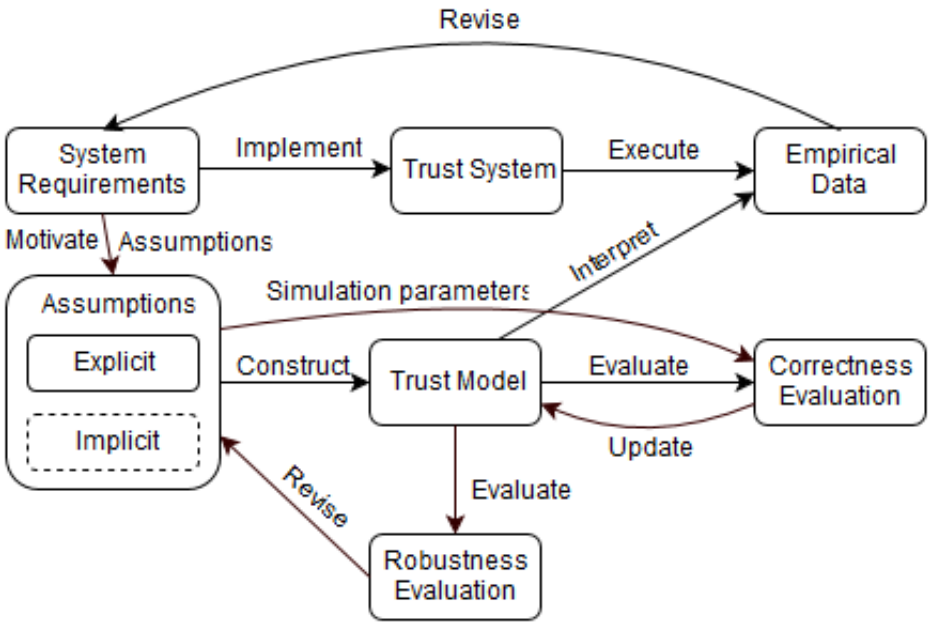

Figure 1: The typical life cycle of trust models.

if indeed a trust models adheres to the principles, then it can be described in our language. In this paper, we demonstrate the generality and validity of the principles within the computational trust paradigm. Moreover, we reformulate existing trust models into the universal language, both to show feasibility and to exemplify the approach.

The language to express the assumptions is a distribution over strategies for each class of users. An assumption about honest users must define exactly what the behaviour of an honest user can be, and the prior probability that a user is honest. (The different strategies need not be finite, or even countable.) The major benefit of the proposed format for assumptions, is that if the assumptions are sufficiently strong, they define a trust model. We refer to the process of obtaining a trust model by merely formulating the assumptions as *trust model synthesis*. There are yet many hurdles to take before trust model synthesis leads to automated trust modelling in practice. We demonstrate, in this paper, both the potential of trust model synthesis (see Section 2) and the feasibility of trust model synthesis in practice (see Section 5).

The document is organised as follows: First we look at the concrete consequences of our proposal in Section 2. There, we also address the shortcomings of the traditional approaches, and motivate our alternative. Then we formally introduce the principles that the framework is built on, in Section 3. Finally, we discuss the feasibility of automated trust modelling in Section 5, and look ahead for possible future challenges and improvements in Section 6.

2 Modelling Trust

Existing trust models and trust systems are being improved by ongoing research and by superior implementations. We refer to the general notion of continuous improvement as the life cycle. The skeleton of the life cycle, is that first a problem or shortcoming is identified, then a solution or idea is proposed, implemented, verified, and possibly accepted. There are some problems with the life cycle, that we address in this section. Throughout this section, we suppose that our three core principles (discussed in Section 3) are sufficient to perform trust model synthesis (discussed in Section 5).

Figure 1 depicts the typical life cycle of trust models and trust systems. The two life cycles are tightly coupled.

The trust system life cycle starts with a set of requirements on a system. The requirements are implemented into a trust system. The implementation of the trust system asserts a certain trust model. Then, the trust system is executed. (Partial) runs of the system are analysed using a trust model (same or other). The empirical analysis may lead to updating the requirements (e.g. if a new attack is found) or to updating the trust model (e.g. if the trust model incorrectly models real users).

The trust model life cycle starts with a set of explicit assumptions, partially based on the requirements of the

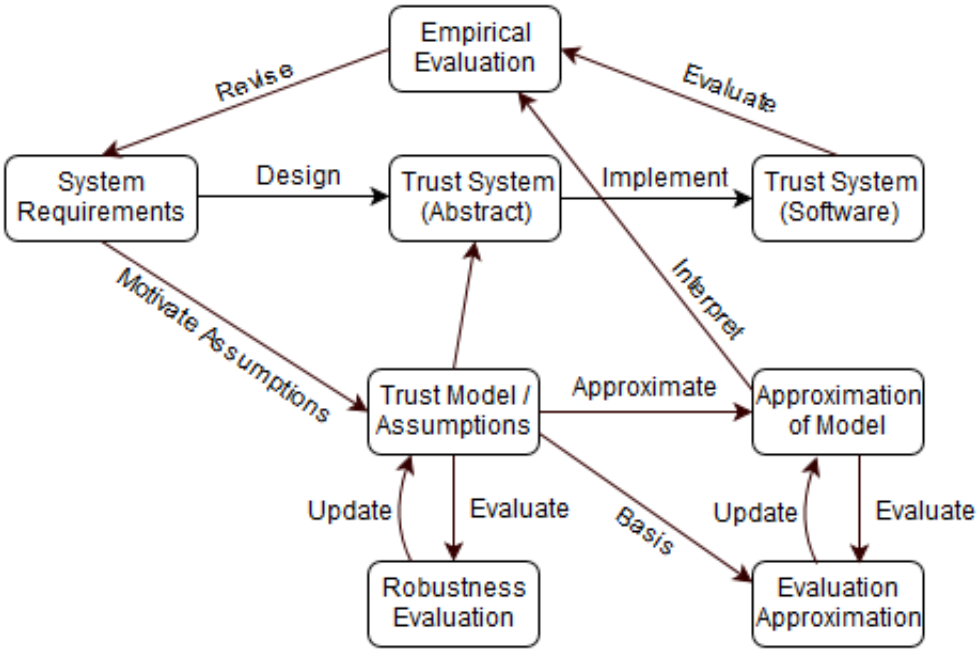

Figure 2: The proposed life cycle of trust models.

system. Based on the assumptions, a trust system can be formulated. Typically, the trust system introduces a set of implicit assumptions. The trust model can be theoretically analysed using simulation or verification. Its results may lead to identifying a correctness problem. Typically, correctness problems are addressed by updating the trust model, not the assumptions. Occasionally, the correctness problems leads to the identification of an implicit assumption. Another theoretical analysis is robustness evaluation, where at least one user may violate any assumptions made about him. Its results may lead to identifying a robustness problem. A robustness problem typically induces updating the assumptions.

Not all modellers follow the life cycle to the letter, but it is a reasonable description of how different factors influence or determine others. Some research focusses only on particular phases. Ideally, their solutions can be reused across different settings. Unfortunately, the classic life cycle hinders this to some extent. For example, it may be difficult to publish a paper that merely identifies some problem, as the audience may expect a solution. Similarly, a solution may require an implementation, and an implementation may require an empirical evaluation, etc.

The life cycle of the trust system remain largely unchanged, except that it now includes an abstraction of the trust system. Note that the abstraction could be made after or before implementation. The trust model life cycle lacks implicit assumptions. The explicit assumptions and the trust model are one and the same. Unfortunately, there is no guarantee that the trust model is computationally feasible. We may be satisfied with trust values that are approximations of the true values predicted by the model. However, these approximations would need additional analysis. Correctness evaluation is no longer a necessity (unless we need approximations, in which case their analysis suffices), and robustness evaluation is streamlined.

Empirical, correctness and robustness evaluation. When it comes to theoretical evaluation, in the classic life cycle of trust systems, correctness evaluation of trust models is emphasised heavily as a motivation to use the trust model. Correctness evaluation allows one to ascertain that the model satisfies the assumptions. It is performed under the assumptions made in the trust model, and the assumptions themselves are not scrutinised. This problem is known, and efforts to mitigate this problem are not novel. The ART testbed [2], for example, is a well-known example of a testbed designed to validate different trust models with a unified procedure.

However, fully general correctness evaluation methods cannot exist [8]. Even the ART testbed can only evaluate those trust models that relate to e-commerce, and then over a limited amount of aspects. More importantly, the ART testbed can only evaluate complete trust systems with trust models, that cover all aspects of e-commerce. It is not a tool that can validate partial models intended to solve specific problems.

An alternative evaluation is empirical evaluation. Empirical evaluation suffers less from circularity issues. However, empirical data still requires interpretation, and is not immune to biased assumptions. Any healthy

life cycle of trust systems must incorporate empirical data at some point. However, as our life cycle does not suffer from the issue of circularity of correctness evaluation, empirical data is not necessary to show internal consistency. As a result, the empirical evaluation is much more loosely coupled to the theoretical evaluation. This allows researchers to specialise on specific subproblems, rather than to force all research to directly translate to a complete trust system.

In our proposed life cycle, it is the assumptions themselves that are evaluated. Since, with trust model synthesis, the model is merely the assumptions, evaluating the trust model equates to evaluating the model assumptions. Note that with the classical approach, there may be implicit or ambiguous assumptions, meaning that evaluating the stated assumptions is insufficient. In our approach, the model assumptions must be assumptions about the behaviour of the agents – concrete and explicit. Our argument is that by forcing the model assumptions to be concrete and explicit, evaluation and comparison of the trust models is more transparent.

3 The Principles of Computational Trust

The three principles that we introduce to capture the paradigm of computational trust are: 1) a trust system is a (timed) process with partially observable states, 2) users' behaviour is dictated by a (probabilistic) strategy and 3) trust values reflect the user's possible behaviour based on evidence. Multi-agent systems, including trust systems, typically satisfy (1) and (2). Furthermore, principle (3) has also been asserted in the computational trust paradigm [10]. Principles (1), (2) and (3) are not novel. The notion that, together, the three principles are sufficiently strong to define a trust model, is novel.

A variation of each of the principles is present in many trust systems. Trust models typically treat trust systems as a process with some properties – specifically what actions are possible at which time. When reasoning about a trust model, one must reason about what certain past actions of an agent say about future actions, and to do this, one must categorise users. The last principle is typically seen as a requirement, e.g. a "good" trust model provides trust values that reflect the user's behaviour. We are going to sever the ties with existing methods, and rigourously define the three principles, even if that excludes some existing models.

Principle 1: Trust System. Principle one is based on processes that can be expressed as deterministic labelled transition systems:

Definition 1. *A* deterministic labelled transition system *is a tuple* (S, A, s_0, t)*, where* S *is a set of states,* A *is a set of actions,* $s_0 \in S$ *is the initial state and* $t : S \times A \to S$ *is the transition function.*

A trace τ is a list of actions $a_0, \ldots, a_n$, and T the set of all traces.

Users $u \in U$ are agents that use the system. Users may fully, partially, or not observe particular actions.

Definition 2. *A* blinding *is a partial function* $\delta : A \times U \nrightarrow A$*.*

When u cannot observe a, then $\delta(a, u)$ is undefined. When u can only partially observe a, then $\delta(a, u) = a'$, where a' is the partial observation. We also allow blinding of traces, denoted with Δ. In $\Delta(\tau, u)$, the elements a in τ are replaced by $\delta(a, u)$, if defined, and omitted otherwise. Thus, $\Delta(\tau, u)$ provides the perspective of agent u, when the system trace is τ.

Based on the notion of deterministic labelled transition systems, blinding and users, we can formally define trust systems:

Definition 3. *A* trust system *is a tuple* $(S, U, A, s_0, t, \delta)$*, where* S *is a set of states,* U *is a set of users,* A *is a set of actions,* s_0 *is the initial state,* $t : S \times U \times A \to S$ *is the transition function and* δ *a blinding.*

Principle 1 supposes that a real trust system can be represented as our abstract notion of trust system.

Principle 2: Strategies. The trust system is simply an automata with branching. We need to grant the users agency, which we provide in the form of a strategy. Strategy may refer to a rational strategy, as often assumed in game theory [6]. But a strategy may also refer to, e.g., a taste profile – what are the odds that a user enjoys something.

In most trust systems, several agents may be allowed to perform an action at a given time. Quicker agents may have an advantage, so timing must play a role in the agents' strategies. We suppose that the time before an action happens is exponentially distributed – for its convenient properties. The exponential distribution has one parameter, which is the expected time until the action, called the *rate* – not to be confused with a (trust) rating.

Traces, users and actions are as defined in the trust system. A (rated) *move* is an assignment of rates to actions, denoted $A \to \mathbb{R}^{\geqslant 0}$. Every user u has a strategy, which dictates the moves of the user, given the circumstances. We use the notion of (blinded) traces to model the circumstances.

Definition 4. *A* strategy *of a user is a function* $f : T \times A \to \mathbb{R}^{\geqslant 0}$. *A* behaviour *of a user is a distribution over strategies,* $b : (T \times A \to \mathbb{R}^{\geqslant 0}) \to [0, 1]$. *W.l.o.g.* $f \in F = \{f | b(f) > 0\}$.

The strategy of a user is in the extensive form. That definition has been chosen for maximal generality. In practice, models of users are often far simpler.

Remark 1. *Definition 4 asserts that all users operate independently. To model Sybil attacks (or forms of collusion), the designer needs to allow a single user to operate multiple accounts. Any cooperative strategy of a set of users that all own a single account can be mimicked by a strategy of single user creating/owning multiple accounts.*

Principle 2 supposes that every user initially has a behaviour.

Principle 3: Trust Values. The trust values should reflect the probabilities of the possible actions that a user may perform. In a simple trust model, for example, actions may be classified as "good" or "bad", and a trust value denotes the probability of "good". In more sophisticated models, more actions are available, and cannot generally be classified as just good or bad. We want our trust values to reflect the likelihood of all possibilities.

We propose to use behaviour (i.e. a distribution over strategies) as a trust value. Suppose that the user is aware of the unblinded trace τ. Assuming discrete probability distributions, users can compute the rate of an action a by u, as $\sum_{f \in F} b(f) \cdot f(\tau, a)$, and the probability as $\sum_{f \in F} b(f) \cdot \frac{f(\tau,a)}{\sum_{a' \in A} f(\tau,a')}$. (1) The generalisation to blinded traces is not much more complicated, and presented in Section 5.

The trust value can typically not be displayed as a single value. In special cases, a compact representation exists (e.g. in Subjective Logic [3], with three values). Usually, however, there is no human-friendly representation.

4 Assumptions

The format of assumptions that exist to support trust models is currently heterogeneous. There have been statistical assumptions, axiomatic assumptions, logical assumptions and informal assumptions. The assumptions are made about trust itself, the trust system, honesty and malice, and about behaviour. The principles cover assumptions about the trust system (1), and about trust itself (3). We further argue that (2) implies that it suffices to formulate the remaining assumptions about behaviour.

In [11], the authors propose a way of dividing model assumptions into two groups. They introduce *fundamental assumptions* and *simplifying assumptions*. Fundamental assumptions are assumptions intended to reflect the nature of the object of study (in [11], "trustee and truster agents are self-interested" is an example). Simplifying assumptions are assumptions necessitated by practical limitations of the model (in [11], "the majority of third-parties testimonies are reliable" is an example). Trust models cannot be formulated without a good deal of fundamental and simplifying assumptions on top of the three principles and a trust system.

Both fundamental assumptions and simplifying assumptions can be encoded into a selection of behaviours. The example simplifying assumption can be encoded by letting the behaviour of those agents that sometimes provide testimonies assign a probability of over 0.5, to those strategies that are reliable[1]. The example fundamental assumption – that users are self-interested – can be encoded by assigning no probability to dominated strategies. User will not have strategies where users can unilaterally increase their own profit.

Without loss of generality, let $C = \{c_0, c_1, \dots\}$ be a partition over U (thus every u_i occurs in exactly one c_j). We call c_i a class of users. For example, we may have a class of buyers and a class of sellers. For each class c, we must assume:

- A set F_c of strategies that users in class c may have.
- A distribution b_c over these strategies.

[1]By making encoding assumption into behaviour, we realise that we have an implicit assumption about what it means to be reliable. Forcing such implicit assumptions to be made explicit is an important benefit of our proposed approach. Here, an educated guess would be that reliable recommenders always provide truthful testimonies about objective events.

Our *language* is a partition of users into classes, with a prior behaviour for each class of users. The language covers all the assumptions that a trust model needs to make (at least in combination with the three principles), and it fulfills the role of a trust model.

The important question is whether it is always possible to translate the assumptions from arbitrary format, to our language. Note that in the classic life cycle (Figure 1), a correctness evaluation is performed, typically using a simulation. All users in that model are simulated using agents with a defined behaviour – the simulation code defines the behaviour of the agents. That code must define, in all possible traces, what the behaviour of the agent is. Therefore, there exists a behaviour for a user, such that it acts exactly like the simulation. Thus, if there exists a trust model with a positive correctness evaluation, then there exists a set of assumptions in our language for the same trust model.

5 Trust Model Synthesis

In order to do trust model synthesis, the modeller must supply a trust system and behaviour, according to principles 1 and 2. Typically, the trust system is a given. The synthesised trust model can provide trust values, according to principle 3. To illustrate the approach with an example:

Example 1. *Take a simple system called MARKET(4,3) with seven users, dividable into two classes: four buyers b_1, b_2, b_3, b_4 and three sellers s_1, s_2, s_3. Buyers b may initiate(b, s) a purchase with any seller s, whenever they do not have an outstanding purchase, after a purchase, they can score it score(b, s, r) where $r \in \{1, 2, 3, 4, 5\}$. The seller can either deliver(s, b) or betray(s, b), which finalises b's purchase. Only the b and s can see initiate(b, s), deliver(s, b) or betray(s, b), meaning that $\delta($initiate$(b, s), u) =$ initiate(b, s), if $u = b$ or $u = s$, and undefined otherwise; and similarly for deliver and betray.*

There is only one buyer strategy. The rate for iniate(b, s) is 1*, if the seller(s) has the highest probability of deliver(s, b) according to the buyer's trust value, and* 0 *otherwise. Letting the unit of time, e.g., be a week, then the buyer buys from a maximally reliable seller on average once per week. There are two seller strategies,* honest *and* cheater*, where, after initiate(s, b), the honest seller performs deliver(s, b) with rate* 0.9 *and betray(s, b) with rate* 0.1*, and the cheater vice versa. Both honest sellers and cheaters take, on average, a week to act, but the honest seller delivers with high probability, whereas the cheater betrays with high probability. After receiving deliver(s, b), b performs score(b, s, r) with rate r. and after betray(s, b), b performs score(b, s, r) with rate $6 - r$.*

The two buyers are users of the trust system. The trust system facilitates interactions between buyers and sellers, by allowing buyers to initiate interactions and sellers to finalise them. Furthermore, the system allows buyers to send ratings, which the other buyers can use. The question is, what will happen in the system? What should a buyer do when given a set of ratings? What will a buyer do? The (synthesised) trust model can answer these questions.

Given the three principles, we can make exact predictions. Given a blinded trace τ, let $\nabla(\tau, u)$ be the set of traces τ', such that $\Delta(\tau', u) = \tau$. Assuming the set of actions A is finite (countable), $\nabla(\tau, u)$ is finite (countable), and can be computed in at most finitely[2] (countably) many steps. Using equation (1), we can compute the probability of performing action a_{i+1}, given $a_0, \ldots, a_i$, and the behaviours of the agents. Since the trust system is deterministic, that implies that for each $\tau' = a_0, \ldots, a_n \in \nabla(\tau, u)$, we can compute the probability of τ' in n steps. Given a distribution over traces, a user can perform a Bayesian update of the behaviours, using equation (1), when he observes an action a. The complexity of the Bayesian update is linear in the number of traces and the number of agents (and constant in the length of the traces). The approach is highly similar to POMDPs, except for the existence of invisible actions and the lack of reward.

Remark 2. *So far, we have not discussed the notions of rewards or goals. The reason is that goals are orthogonal to our approach. A modeller is simply asked how he expects the users to behave, and to write this down mathematically, and he can synthesise a trust model. However, in reality users do have goals, and their goals are relevant to other aspects than the synthesis. First, the modeller may expect the behaviour because of the goals. In Example 1, the fact that buyers select the most trustworthy seller reflects their goal of not being betrayed. The split between the two classes of sellers as honest and cheater also reflects that sellers may have two goals (to make money honestly, or to cheat). Second, the goals pop up in robustness analysis. If a strategy is found that achieves the goals far better than other strategy and/or harms other users in achieving their goals, then it may be prudent to add that strategy into the behaviour of users of that class. (Example 1 is not robust against a reputation lag*

[2] As long as the probability of extremely large invisible subtraces is negligible.

attack. A seller becomes the most trustworthy one, lets others buy from him, but wait with betraying until all four have made a purchase. Such a strategy can then be added to the behaviour of sellers.)

Although the problem is theoretically computable, the approach is nevertheless intractable in full generality. Observe the following challenges: behaviours with incomputable probability density, strategies encoding NP-hard problems, and statespace explosion (number of traces). These practical issues are, of course, to be expected. However, practical models typically have simple probabilities, strategies are not based on hard problems, and agents do not exploit the entire statespace.

To illustrate the practical computability, consider Example 1: In the system MARKET(4,3), for all τ, $\nabla(\tau, b_1)$ is infinite. However, all $\tau' \in \nabla(\tau, b_1)$ are *probabilisticly bisimilar* [5], when we restrict to blinding all actions with b_1. When two states are probabilisticly bisimilar, it means that the two states are completely identical, and we can collapse the two states.

6 Research Problems

Implementation. The first step towards real automated trust modelling, is a prototype tool. The simplest approach to such a tool is to transform the assumptions into a probabilistic automaton, and use probabilistic verification tools, such as PRISM [4], to automate the synthesis. Likely, general purpose tools are not sufficiently powerful, and specialised tools need to be developed for simple, realistic models (e.g. Beta-type models [3]).

The problems that the special purpose tool would have to overcome, are similar to those of probabilistic verification tools. We cannot yet envision the precise challenges, but statespace reduction will be a necessity. We saw that for MARKET, probabilistic bisimulation [5] reduces the statespace from countably infinite to 1. More importantly, possible forms of statespace reduction exist for our purpose, such as: Letting the system signal partial information about an action (e.g. an e-market place could signal that a transaction occurred, even if it is unaware of the outcome), and generating equivalence classes over blinded traces.

The user of the synthesised trust model may not be interested in the exact probability values. If the user allows an absolute error ϵ, and there is an upper bound m to the rate of the outgoing actions, then the statespace can be trivially bounded to a finite size. Assuming all states have an outgoing rate of m, the probability that the length of the trace exceeds n, at time x, is exponentially distributed as $e^{-\frac{m}{n} \cdot x}$. Given m, x, ϵ, it is always possible to pick n, such that $e^{-\frac{m}{n} \cdot x} < \epsilon$. Thus, by introducing a small error ϵ, we can restrict the traces to the traces of length at most n.

The authors are currently working on a tool that can do robustness verification for generic trust models. Robustness verification is an excellent way to find new possible attacks, which, in turn, can help us construct behaviours that take into account future attacks and responses to defences.

Application. The theoretical concepts of trust model synthesis are surprisingly powerful. In order to judge the practical power of trust model synthesis, a real model should be encoded, synthesised and compared with the original. The second core principle forces the modeller to be explicit and concrete with model assumptions. This means that, e.g., "trust is transitive" is not a valid assumption, and should be replaced by assumptions about the behaviour of users. Finding a general, but concrete, translation of that assumption is an interesting challenge. A multitude of similar assumptions exist, which pose equally interesting challenges for the modeller.

Evalutation and Analysis. We have shortly addressed the notions of evaluation and analysis. Our approach to validation is complementary to the orthodox approach to validation (e.g. ART testbed [2]). Due to the concrete nature of the assumptions, they can directly be contrasted with reality. There is, however, always a degree of interpretation. How to minimise the effect of interpretation is currently an open question.

The approach opens new doors for robustness analysis. In security analysis, it is common to reason about users that violate the assumptions of a security protocol, and to automatically verify the security of the protocol. The question is to what extend these methods can apply to our domain. Recent research indicates that such methods are feasible for the domain [7]. Attempting to automatically verify the impact of breaking the assumptions is a difficult challenge.

7 Conclusion

We have presented a novel approach to constructing trust models. The main advantage is the lack of hidden assumptions that may introduce unseen problems. The key contribution is a generic language to formulate assumptions about trust models. The language consists of describing the behaviour of (classes of) users. We have formulated 3 major principles that we argue apply to all trust systems. The validity of the language hinges

on these principles. We have formulated how the design and construction of trust systems and models can be streamlined by our proposal. Finally, parts of the tasks of the trust system can be generated automatically, using trust model synthesis.

There are several ways in which the language can help in future research: One way is by providing a link towards automation helps researchers tackle problems that are more suitable to be address by computers. Furthermore, two mutually intelligible trust models can now be provided a common foundation for comparison. Finally, we hope that vague or hidden assumptions are eventually considered unacceptable, and our language is one of several approaches to bring rigour.

References

[1] Rino Falcone and Cristiano Castelfranchi. Transitivity in trust: a discussed property. In *Proceedings of the 10th Workshop on Objects and Agents (WOA)*, 2010.

[2] Karen K Fullam, Tomas B Klos, Guillaume Muller, Jordi Sabater, Andreas Schlosser, Zvi Topol, K Suzanne Barber, Jeffrey S Rosenschein, Laurent Vercouter, and Marco Voss. A specification of the agent reputation and trust (art) testbed: experimentation and competition for trust in agent societies. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 512–518. ACM, 2005.

[3] Audun Jøsang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(03):279–311, 2001.

[4] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In P. Kemper, editor, *Proc. Tools Session of Aachen 2001 International Multiconference on Measurement, Modelling and Evaluation of Computer-Communication Systems*, pages 7–12, September 2001.

[5] Kim G Larsen and Arne Skou. Bisimulation through probabilistic testing (preliminary report). In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 344–352. ACM, 1989.

[6] Kevin Leyton-Brown and Yoav Shoham. Essentials of game theory: A concise multidisciplinary introduction. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2(1):1–88, 2008.

[7] Tim Muller, Yang Liu, Sjouke Mauw, and Jie Zhang. On robustness of trust systems. In Jianying Zhou, Nurit Gal-Oz, Jie Zhang, and Ehud Gudes, editors, *Trust Management VIII*, volume 430 of *IFIP Advances in Information and Communication Technology*, pages 44–60. Springer Berlin Heidelberg, 2014.

[8] Stewart Robinson. Simulation model verification and validation: Increasing the users' confidence. In *Proceedings of the 29th Conference on Winter Simulation*, WSC '97, pages 53–59, 1997.

[9] Jordi Sabater and Carles Sierra. Review on computational trust and reputation models. *Artificial intelligence review*, 24(1):33–60, 2005.

[10] Yonghong Wang and Munindar P Singh. Formal trust model for multiagent systems. In *IJCAI*, volume 7, pages 1551–1556, 2007.

[11] Han Yu, Zhiqi Shen, CYRIL Leung, Chunyan Miao, and VICTOR R Lesser. A survey of multi-agent trust management systems. *Access, IEEE*, 1:35–50, 2013.