ascens

www.ascens-ist.eu

ASCENS

Autonomic Service-Component Ensembles

D5.3: Third Report on WP5

Verification Techniques and Security Issues

Grant agreement number: **257414**
Funding Scheme: **FET Proactive**
Project Type: **Integrated Project**
Latest version of Annex I: **Version 2.2 (30.7.2011)**

Lead contractor for deliverable: **UJF-VERIMAG**
Author(s): **Saddek Bensalem (UJF-Verimag), Michele Boreale (UDF), Jacques Combaz (UJF-Verimag), Rocco De Nicola (IMT), Jan Kofroň (CUNI), Gianluca Mezzetti (UNIPI), Pavel Parízek (CUNI), Francesco Tiezzi (IMT)**

Reporting Period: **3**
Period covered: **October 1, 2012 to September 30, 2013**
Submission date: **November 8, 2013**
Revision: **Final**
Classification: **PU**

Project coordinator: **Martin Wirsing (LMU)**
Tel: **+49 89 2180 9154**
Fax: **+49 89 2180 9175**
E-mail: **wirsing@lmu.de**

Partners: **LMU, UNIPI, UDF, Fraunhofer, UJF-Verimag, UNIMORE, ULB, EPFL, VW, Zimory, UL, IMT, Mobsya, CUNI**

SEVENTH FRAMEWORK PROGRAMME

Executive Summary

This document summarizes the work performed in Year 3 targeted to develop new techniques and theories to support the design and the implementation of correct and reliable service components (SC) and service component ensembles (SCE). We have worked in several different directions in order to get closer to our goal. Our contributions can be grouped into the following main directions: (1) We have presented a recent work that aim at providing models and tools for the quantitative assessment of the correct functioning of distributed systems. (2) We proposed two approaches for network-aware evaluation of reputations systems, at the design stage and at implementation/deployment stage. (3) We have proposed a model-driven security approach for SCE. The developed framework guarantees automated verification and implementation of secure information flow systems with respect to specific definition of non-interference. (4) We have presented a new method for the verification of timed systems. It consists in adapting the compositional verification approach, developed in D-Finder, to timed system. The main challenge was to be able to capture the relations between the local timing of the components induced by their interactions. (5) We introduced a framework for verification of dynamic systems based on nominal automata, for which we investigated several fundamental theoretical questions. (6) Finally, we designed and implemented support for automated verification of jDEECo applications with the Java Pathfinder model checker.

Contents

1 Introduction

Our goal in WP5 is to develop new techniques and the underlying theories to support the design and the implementation of correct and reliable service components (SCs) and service component ensembles (SCEs). We are working in several directions. The first one deals with correctness on the level of a service component, considering in particular non-functional properties like resource-awareness. The second deals with correctness of ensembles of service components mainly focusing on constructive techniques. Given the particular importance of security in ensembles, the third will address the problem of building secure ensembles. Finally, we work on techniques to check if a SC implementation complies with a high-level specification. In year 3, we have worked on the three tasks:

1. In Task T5.2 *Verification of Service Component Ensembles*, we worked on:

   - *Compositional Invariant Generation for Timed Systems*. In this task we address the state-explosion problem inherent to model-checking timed systems with a great number of components. The main challenge is to obtain pertinent global timing constraints from the timings in the components alone. To this end, we make use of auxiliary clocks to automatically generate new invariants, which capture the constraints induced by the synchronizations between components. The method has been implemented as an extension of the D-Finder tool [BBSN08] and successfully experimented on several case studies.
   - *Towards Nominal Automata Model Checking*. Standard automata-based techniques in software verification do not scale up to manage *Ensembles* with massive numbers of components, operating in open and non-deterministic environments, and dynamically adapting to new requirements and environmental conditions. To deal with the that, we exploit *nominal techniques* to abstract the unbounded set of entities that may occur during a computation. We developed a foundational model for context-free nominal languages allowing to *dispose* and to *reuse* resources, and proved fundamental results about expressiveness, closure under union, concatenation and intersection, and, finally, decidability of the emptiness problem.

2. In Task T5.3 *Security Policies and Access Control*, we worked on:

   - *Quantitative Distributed Models of Confidentiality and Trust*. Trust and reputation systems are decision support tools used to drive parties' interactions on the basis of parties' reputation. In such systems, parties rate with each other after each interaction. Reputation scores for each ratee are computed via reputation functions on the basis of collected ratings. In this work, we propose a general framework based on Bayesian decision theory for the assessment of such systems, with respect to the number of available ratings. Given a reputation function g and n independent ratings, one is interested in the value of the loss a user may incur by relying on the ratee's reputation as computed by the system. To this purpose, we study the behavior of both Bayes and frequentest risk of reputation functions with respect to the number of available observations. We provide results that characterize the asymptotic behavior of these two risks, describing their limits values and the exact exponential rate of convergence. One result of this analysis is that decision functions based on Maximum-Likelihood are asymptotically optimal. We also illustrate these results through a set of numerical simulations.
   - *Network-Aware Analysis of Reputation Systems.* We addressed the design and the implementation of reputation systems using KLAIM, a network-aware coordination language equipped with a formal semantics. The proposed approach can be conveniently extended

to SCEL as KLAIM mechanisms are at the basis of SCEL. To check the system at design time, we enrich KLAIM specifications with stochastic aspects using STOKLAIM, and verify them against desired properties expressed by the stochastic logic MOSL using the tool SAM. Once a design is verified, it can be implemented using the proposed framework NEVER (Network-aware EValuation Environment for Reputation systems), which allows to describe, to implement, and to evaluate a reputation system while taking into account real-world implementation details and the network environment where they have to be deployed. Our approach distinguishes itself amongst existing ones as it evaluates the system through experiments on real networks rather than simulations based on models that abstract from many details. In this way, given a specific network environment, we can study the system behavior to find the configuration that better meets the system requirements by tuning its parameters. Moreover, since we consider reputation systems at implementation level, the analyzed systems could be then directly used in the corresponding end-user applications.

- *Model-driven Information Flow Security for SCE*. We propose a framework for information flow security in component-based systems, which follows the model-driven security approach. The security policy is defined and verified from the early steps of the system design. In this work, two kinds of non-interference properties are formally introduced and for both of them, sufficient conditions that ensure and simplify the automated verification are proposed. The verification is compositional: it checks independently the behavior of every atomic component and their communications and coordinations. The benefit of the approach is illustrated through an application to secure heterogeneous distributed systems.

3. For Task T5.4 *Verification of SCs implementation compliance with high-level specification*, we worked on the design and the implementation of a support for automated verification of jDEECo applications with the Java Pathfinder (JPF) model checker. Our work in this direction consists of two steps: (1) making the jDEECo runtime framework amenable to practical verification with JPF and (2) implementing support for checking specific properties relevant to jDEECo applications.

Relation to Other Work Packages

This year, we mainly strengthened collaborations with WP7 by applying our verification and validation techniques to the three case studies of the project (see Deliverable JD3.1). WP5 has also the following relations to other work packages of the project as explained as follows.

- We used the SCEL language developed in WP1 for specifying network-aware reputation systems (see Section 3).
- Work Packages 2 and 5 have BIP as a common language for expression of SCEs and their analysis. However, even if BIP has been extended to reconfigurable architectures, verification and validation tools for BIP models are currently restricted to static architectures.
- We currently see no interaction with WP3.
- We developed a simulation framework for adaptation patterns proposed in WP4, which has been applied to e-Mobility case study (see Deliverable JD2.2).
- We have started or completed the integration of several verification and validation tools such as BIP, jSAM, MESSI, FACPL, or Gimple.

- As mentioned above, we had interactions with WP7 when applying the techniques proposed in WP5 to case studies (see Deliverable JD3.1).

- We contributed to the Ensemble Development Life Cycle (EDLC) of ASCENS proposed in WP8, by the integration of techniques proposed in WP5 (see D8.3). The ASCENS EDLC also clarifies which techniques can be used at design time and which ones at runtime.

2 Quantitative Distributed Models of Confidentiality and Trust

Information-hiding systems, understood in a broad sense as means to protect sensitive information (users' personal data and identities, keys, passwords,...), and reputation systems, employed as decision support tools in a variety of contexts (e-commerce, social networks), are at the core of today's distributed computing. As a matter of fact, these systems are most often designed and deployed without any formal guarantee of correctness. In this section, we review some recent work that aims at providing models and tools for the *quantitative* assessment of the correct functioning of these systems. A unifying trait of our treatment is an extensive use of Bayesian reasoning to formalize the flows of probabilistic information involved in both Trust and Confidentiality evaluation. This section is based on the papers [BB12] and [BC13]. A more detailed discussion follows.

2.1 Worst- and average-case privacy breach in randomization mechanisms

In a variety of contexts, randomization is regarded as an effective means to conceal sensitive information. For example, anonymity protocols like Crowds [RR] or the Dining Cryptographers [DC] rely on randomization to "confound" the adversary as to the true actions undertaken by each participant. In the fields of Data Mining and Privacy, techniques have been proposed by which datasets containing personal information that are released for business or research purposes are perturbed with noise, so as to prevent an adversary from re-identifying individuals or learning sensitive information about them – see e.g. Dwork's Differential Privacy [DP] and references therein. In both secret-key and public-key cryptography, noise can be inserted into the plaintext so as to make the resulting ciphertext hard to distinguish from purely random data.

In the last few years, interest in the theoretical principles underlying randomization-based information protection has been steadily growing. Two major areas have by now clearly emerged: *Quantitative Information Flow* (QIF) [BCP09, KS10, BPP11b, BPP11a, CPP08a, CPP08b, Smith] and *Differential Privacy* (DP) [DP, DMNS, McS, Talwar, Fried, SKG, Ghosh, KAS, Kifer]. As discussed in [BK11], QIF is mainly concerned with quantifying the degree of protection offered against an adversary trying to guess the whole secret; DP is rather concerned with protection of individual bits of the secret, possibly in the presence of background information, like knowledge of the remaining bits. The areas of QIF and DP have grown separately for some time: only recently researchers have begun investigating the relations between these two notions [AAC$^+$a, AAC$^+$b, AAC$^+$c, BK11].

Our work is an attempt at distilling and systematizing the notions of security breach underlying QIF and DP. We view a randomization mechanism as an information-theoretic channel with inputs in $\mathcal{X}$ and outputs in $\mathcal{Y}$. The starting point of our treatment is a semantical notion of breach. Assume $\mathcal{X}$ is a finite set of items containing the secret information X, about which the adversary has some background knowledge or belief, modeled as a prior probability distribution $p(x)$. Consider a predicate $Q \subseteq \mathcal{X}$ – in a dataset about individuals, one may think of Q as gender, or membership in a given ethnical group etc. The mere fact that X is in Q or not, if ascertained, may convey sensitive information about X. Henceforth, any observation $y \in \mathcal{Y}$ that causes a significant change in the adversary's posterior belief about $X \in Q$ must be regarded as dangerous. In probabilistic terms, Q is a *breach* if, for some prior

probability on $\mathcal{X}$, the posterior probability of Q after interaction with the randomization mechanism exhibits a significant change, compared to its prior probability. We decree a randomization mechanism as secure at level ϵ, if it exhibits *no breach* of level $> \epsilon$, independently of the prior distribution on the set of secret data $\mathcal{X}$. The smaller ϵ, the more secure the mechanism. This simple idea, or variations thereof, has been proposed elsewhere in the Data Mining literature – see e.g. [EGS]. Here, we are chiefly interested in analyzing this notion of breach according to the following dimensions.

1. Worst- vs. average-case security. In the worst-case approach, one is interested in bounding the level of any breach, independently of how likely the breach is. In the average-case, one takes into account the probability of the observations leading to the breach.

2. Single vs. repeated, independent executions of the mechanism.

3. Expected utility of the mechanism and its asymptotic behavior. Utility can be formally defined as the average loss incurred by a user when mistaking the noisy answer reported by the mechanism with the "true" answer. This of course depends on both the number of observations and on a user-defined loss function – typically some distance on the space of possible answers.

To offer some motivations for the above list, we observe that worst-case is the type of breach considered in DP, while average-case is the type considered in QIF. In the worst-case scenario, another issue we consider is resistance to background information. In the case of DP, this is often stated in the following terms [DP]: *Regardless of external knowledge, an adversary with access to the sanitized database draws the same conclusions whether or not my data is included.*We investigate how this kind of resistance relates to the notion of privacy breach we consider, which also intends to offer protection against arbitrary background knowledge.

Concerning the second point, a scenario of repeated observations seems to arise quite naturally in many applications. For example, in a sensor networks scenario, an attacker can easily gather multiple observations related to an individualThe same is true in a scenario of anonymity networks comprising corrupted nodes. For another example, in a de-anonymization scenario similar to [NS], [BPP11a] shows that gathering information about a target individual can be modeled as collecting multiple observations from a certain randomization mechanism. Again, an online, randomized data-releasing mechanism might offer users the possibility of asking the same query a number of times, thus potentially allowing an adversary to remove enough noise to learn valuable information about the secret. This is an instance of the *composition* attacks well known in the context of DP, where they are thwarted by allotting each user or group of users a *privacy budget* that limits the overall number of queries to the mechanism.In general, one would like to assess the security of a mechanism in these situations. In particular, one would like to determine exactly *how fast* the level of any potential breach grows, as the number n of independent observations grows.

The third point, concerning utility, has been the subject of intensive investigation lately – see the related work paragraph. Here, we are interested in studying the growth of expected utility in the model of Ghosh et al. [Ghosh] as the number of independent observations grows, and to understand how this is related to security. In summary, the main results we obtain are the following.

1. In the scenario of a single observation, both in the average and in the worst case, we characterize the security level (absence of breach above a certain threshold) of the randomization mechanism in a simple way that only depends on certain row-distance measures of the underlying matrix.

2. We prove that our notion of worst-case security is stronger than DP. However, we show the two notions coincide when one confines to background information that factorizes as the product of independent measures over all individuals. This, we think, sheds further light on resistance of DP against background knowledge.

3. In the scenario of repeated, independent observations, we determine the exact asymptotic growth rate of the (in)security level, both in the worst and in the average case.

4. In the scenario of repeated, independent observations, we determine the exact asymptotic growth rate of any reasonable expected utility.

A problem left open by our study is the exact relationship between our average-case security notion and the maximum leakage considered in QIF – see e.g. [KS10]. We would also like to apply and possibly extend the results of the present paper to the setting of de-anonymization attacks on dataset containing micro-data. [NS] has shown that the effectiveness of these attacks depends on certain features of sparsity and similarity of the dataset, which roughly quantify how difficult it is to find two rows of the dataset that are similar. The problem can be formalized in terms of randomization mechanisms with repeated observations – see [BPP11a] for some preliminary results on this aspect. Then the row-distance measures considered in our work appear to be strongly related to the notion of similarity, and might play a crucial role in the formulation of a robust definition of dataset security.

The definitions of semantic security investigated involve a quantification over all possible priors. In certain contexts, however, it may be sensible to put constraints on the form of the prior: one case we have considered is independent participation of individuals to a database. Another possibility is to assume lower bounds on the entropy of the prior. Indeed, certain proofs obtained here - and elsewhere in the literature - exploit in a crucial way the existence of low entropy priors, where most of the probability mass is concentrated on very few elements. In several application scenarios (e.g. secret-key cryptography), such priors are unrealistic and could be ruled out. This would be in the line of work in entropic security, see e.g. [DS05].

2.2 Asymptotic risk analysis for trust and reputation systems

Trust and reputation systems are used as decision support tools for different applications in several contexts. Probably the best known applications are related to e-commerce: well-known examples in this context are the auction site eBay and the online shop Amazon, [JIB]. Trust management systems are used in many other contexts and applications, where huge amount of data related to reputations of peers are usually available, such as ad-hoc networks [NCL], P2P networks [XL, WV] and sensor networks [BXE07].

The idea at the base of trust and reputation systems is to let users of the system, the raters, rate the provided services, or rates, after each interaction. Then other users or the parties themselves may use aggregate ratings to compute reputation scores for a given party: such values are used to drive parties' interactions. This approach to trust managing is referred to as computational trust. Whereas traditional credential-based approaches [EFLR, NT] rely on access control policies and/or use of certificates for evaluating parties' trustworthiness, in computational trust parties' trustworthiness is evaluated by on the basis of the parties' past behavior.

In our work, we focus on *probabilistic* trust [JI, Desp, MMH], which represents a specific approach to computational trust. The basic postulate of probabilistic trust is that the behavior of each party can be modeled as a probability distribution, belonging to a given family, over a certain set of interaction outcomes – success/failure being the simplest case. Once this postulate is accepted, the task of computing reputation scores boils down to inferring the true distribution's parameters for a given party. Information about party's past behavior is used for parameters inference: rating values are treated as statistical data.

The most prominent aspect of a reputation system is how decisions are computed on the basis of the observed data. We use the term *decision* in a broad sense here, meaning e.g. whether or not a given system should be deemed trusted, or decide what is the most likely outcome of the next

interaction. We will refer to this element of the system as the *decision function* or *strategy*. One example of decision function is the Maximum Likelihood (ML) rule from Statistics, which, given a set of specified possible behaviors, picks up the one whose induced distribution on the observations is most similar to the observed, empirical one. The Bayesian Maximum A Posteriori (MAP) rule is similar, but a prior probability on the set of possible behaviors is also taken into account.

The potential usefulness and applicability of probabilistic trust is by now demonstrated by a variety of tools that have been experimentally tested in several contexts. One important example is the work on the TRAVOS system [TPJL]. On the contrary, there are very few analytical results on the behavior of such systems – with the notable exception of the work by Sassone and collaborators [SNK] (see [BC13] for a discussion of the relationship between our work and theirs). Examples of questions that could be addressed analytically are: How do we quantify the confidence in the decisions calculated by the system? And how is this confidence related to such parameters as decision strategy and number of available ratings? Is there an optimal strategy that maximizes confidence as more and more information becomes available?

In our work, we address the above questions, and propose a framework to analyze probabilistic trust systems based on Bayesian decision theory [Rob, Berg, LHG]. A prominent aspect of this approach is the use of probability distributions to model prior *beliefs* on the set of possible parties' behaviors. However, we also consider confidence measures that dispense with such prior beliefs. We study the behavior of trust and reputation systems relying on the concept of *loss function*: this defines the loss incurred by a user when the decision computed by the mechanism is not the correct (or "true") one. We quantify confidence in the system in terms of risk quantities based on expected (a.k.a. Bayes) and worst-case loss. We study the behavior of these quantities with respect to the available information, that is the number of available rating values. Our results allow to characterize the asymptotic behavior of probabilistic trust systems. In particular, we show how to determine the limit value of both Bayes and worst risks, and the exact exponential *rate* of convergence, in the case of independent and identically distributed (i.i.d) observations. Indeed, given a decision framework, it is important to establish not only the limit of the Bayes and worst-case risk functions, denoted respectively as r^n and w^n, as the number n of available ratings grows; but also how fast this limit is approached. The concept of rate is important for two reasons. Firstly, it is desirable to distinguish between reputation functions with different rates, as a reputation function with a high rate may require considerably less observations in order to achieve an improvement of the risks value, compared to a reputation function with a low rate. Secondly, knowledge of the rate will allow us to obtain quick and accurate estimations of the risk functions r^n and w^n depending on n. The main results we obtain can be described as follows.

1. The best achievable rate of convergence to the minimum values of any decision function, for both Bayes and worst risks, is bounded above by R, where $R \triangleq \min_{\theta \neq \theta'} C(p_\theta; p_{\theta'})$ is the least Chernoff Information between any pair of distinct distributions p_θ and $p_{\theta'}$ in the given family $\mathcal{F} = \{p(\cdot|\theta) | \theta \in \Theta\}$. More formally, assume $\lim r^n$ exists. Then

   - $\lim r^n \geq r^*$
   - if $\lim r^n = r^*$ then $\mathrm{rate}(r^n) \leq R$ if defined.

   Similarly for the worst risks w^n and w^*.

2. Both MAP and ML are asymptotically optimal decision functions. Such functions achieve minimum loss value and maximum rate of convergence, that means that, in these cases, $\lim_n r^n = r^*$ and moreover $\mathrm{rate}(r^n) = R$. Similarly for w^n.

In essence, the above results can be summarized by saying that, under an optimal decision function, r^n behaves as $\approx r^* + 2^{-nR}$, and w^n as $\approx w^* + 2^{-nR}$. In our discussion, we also describe the form

of the (optimal) prediction function for the next observation o, given past observation o^n: as expected, given o^n, one has to first identify the underlying distribution p_θ and then report as an answer the observation o that maximizes $p_\theta(o)$.

We also discuss the probability that the loss deviates from its minimal value above a prescribed threshold ϵ. These results apply to the case of ML or MAP decision functions, and do not depend on the set of parameters Θ, but only on the number n of observations and the threshold value fixed for the loss. Fix any $\theta \in \Theta$ and assume O^n is a n-sequence of random variables i.i.d. given θ, that is $O_i \sim p(\cdot|\theta)$. Let $\epsilon, \gamma > 0$. Then:

- considering the KL-loss function $L(\theta; \theta') \triangleq D(p(\cdot|\theta)||p(\cdot|\theta'))$[1]

$$\Pr(L(\theta; g(O^n)) > \epsilon) \leq (n+1)^{|O|} 2^{-n\epsilon}$$

- considering the norm-1 loss function $L(\theta; \theta') \triangleq ||p(\cdot|\theta) - p(\cdot|\theta')||_1$

$$\Pr(L(\theta; g(O^n)) > \gamma) \leq (n+1)^{|O|} 2^{\frac{\gamma^2}{2 \ln 2}}$$

We complement these theoretical results with set of numerical simulations.

Future research directions include the extension of our framework to different data models, with rating values released in different ways. In [BC13], we briefly discuss a possible refinement of the observation and rating mechanism that takes into account possible raters' mis-behavior. We would like to take into account the case in which each rater is characterized by a (unobservable, possibly malicious) bias, which can lead him to under- or over-evaluate its interactions with the rates. In order to do that, we have to introduce a refined data generation model, where the probability of observing a given rating value does not depend solely on the behavior of the rate, but also on the unobservable bias. We argue that a data-model with hidden variables is well-suited to model this kind of scenario; this naturally prompts the use of Expectation-Maximization (EM) algorithm [Barb] to practically perform parameter estimation in this context. The EM algorithm can be used for efficiently implementing the decision functions we have considered. Another issue is how to evaluate the fitness of the model to the data actually available; and, in general, how to assess the trade-off between tractability and accuracy of the model.

3 Network-Aware Analysis of Reputation Systems

In ensemble-like systems, as those considered in the ASCENS project, parties are likely to be disconnected from their preferred security infrastructures and/or may have no trusted information about their partners. Therefore, they have to rely on a specific apparatus to build up relationships of trust among each other. For this purpose, we have seen in different areas of ICT an increasing use of *trust and reputation systems*, from e-commerce to different forms of open computer networking (such as P2P, ad-hoc, or sensor networks).

In order to establish such trust relationships, parties in a reputation system rate each other (e.g., after completing an interaction) and use aggregated ratings about a given party to derive a *reputation score*, i.e. a collective measure of trustworthiness based on the ratings from the members of a community. Such value is used when deciding whether to interact with specific partners. Thus, by relying on reputation systems, parties can adapt their behavior to the environment in which they are operating.

Parties of a reputation system can interact and exchange ratings by relying on a *networked trust infrastructure*, graphically depicted in Figure 1. The *rating server* collects the ratings from system's

[1] Recall that $D(p(\cdot|\theta)||p(\cdot|\theta'))$ is defined as $\sum_o p(o|\theta) \log(\frac{p(o|\theta)}{p(o|\theta')})$.

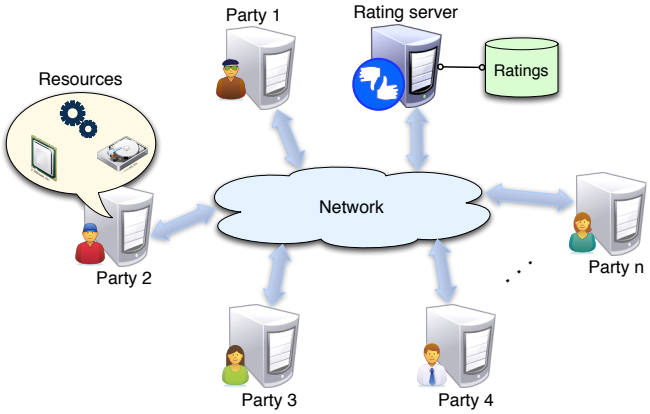

Figure 1: Infrastructure of a reputation system

parties and makes them publicly available. A *party* can be a client, a provider, or both, and may require/offer different kinds of resources (CPU, disk space, files, services, etc.). Whenever a party needs a specific resource, it queries the rating server to determine the reputation of all those parties that can offer it. Then, it selects the provider with the highest reputation score and, after the interaction, rates the provider according to the quality of the provided resource. Thus, the use of reputation systems should disallow parties to interact with providers that do not behave as expected.

Different kinds of reputation system, which mainly differ by the model used to aggregate ratings when computing reputation scores, can be used on top of this general infrastructure. Indeed, due to the widespread use of reputation systems, several models have been proposed. Thus, once a reputation system has to be deployed in a network environment, the following issues have to be considered:

- the fitting of the model to the environment;
- the correspondence between parties behavior and their reputation;
- the impact of the initial reputation scores;
- how the reputation system should be configured in order to meet the desired behavior.

We address these issues, at two different stages of the development process, by proposing two approaches for network-aware evaluation of reputation systems:

1. at *design* stage: a verification methodology for reputation systems based on stochastic analysis;
2. at *implementation/deployment* stage: a software framework supporting rapid prototyping of reputation systems and analysis of their executions on given network infrastructures.

Notably, the two approaches are deemed to be network-aware because they take into account details of the network environment where the systems have to be deployed, rather than evaluating them in isolation. These analysis approaches differ from the theoretical one described in Section 2.2 not only for the network-awareness aspect, but also for the use of specific software tools to support the evaluation.

Both approaches relies on KLAIM [DFP98], a network aware coordination language equipped with a formal semantics. The coordination mechanism of KLAIM is based on multiple tuple-spaces, which enable to effectively model and program distributed systems operating in open and non-deterministic environments. In particular, with respect to other coordination languages, KLAIM has the additional advantages of being equipped with a formal semantics, of being supported by a runtime environment and a number of tools for performing systems analysis and, most importantly, of offering the possibility of modelling explicit localities and thus supporting network-aware programming of distributed

applications. It is also worth noticing that KLAIM mechanisms are at the basis of the SCEL language [DFLP11, DLPT13], thus the proposed approaches can be conveniently extended to SCEL.

Another common point between the two proposals is the use of a probabilistic approach for computing reputation scores. The basic postulate of probabilistic trust [Gam88] is that party's behavior can be modeled as a probability distribution over a given set of interaction outcomes.

3.1 A stochastic verification methodology

Our methodology [CDT13c] consists of the following three steps:

1. We model the considered reputation system with KLAIM. Specifically, we provide a 'schema' specification that is parametric w.r.t. the reputation model used to determine the parties' reputation (and w.r.t. other parameters of the system). The specification of the given system is obtained by appropriately instantiating the parameters of the generic specification.

2. We enrich the specification with stochastic aspects, using STOKLAIM [DKL$^+$07], the stochastic extension of KLAIM, and formally express the desired properties using the stochastic logic MOSL [DKL$^+$07].

3. We check the properties of interest against the STOKLAIM specification by means of the analysis tool SAM [Lor10].

To demonstrate feasibility and effectiveness of our proposal, we have specified and analyzed two models of reputation systems, namely the Beta model [JI02] and a model based on maximum likelihood estimation (ML model) [DA04].

The results of some simulation runs of the STOKLAIM specifications, performed by using SAM, are reported in Figure 2 (we refer the interested reader to [Cel13] for an account of the stochastic specifications and for other analysis results). The chart presents the trend of the reputation of a given party in the system and the error made by the reputation model in the computation of reputation scores. In the chart, an horizontal line denotes the *true* party's behavior. The x-axis reports the numbers of rating values used to compute reputation scores. The y-axis reports both the reputation scores and the error made. As measure of the error we used the Norm-1 distance between probability distributions. Our choice of the error measure follows by the study presented in [BC13]. From simulation data shown in Figure 2, we observe that in general the ML model performs better than the Beta model if we consider the computed reputation score. Indeed the reputation score computed by the ML model is always closer to the true party's behavior than the reputation score computed by the Beta model. Such evaluation does not give us a complete information about models performances. Indeed looking at the error introduced by the models, we observe that the ML model does not perform always better than the Beta model. In particular, with a low number of ratings, the ML model tends to calculate reputation values that significantly underestimate or overestimate the party's behavior (this is reflected by an high error). Instead, the reputation trend results from the average of values obtained from many simulation runs and, hence, hides such error information.

We have also analyzed some properties of these reputation systems by formalizing them as MOSL formulae that can be checked over the STOKLAIM specification by means of the SAM tool. As an example, we have considered the property "*the reputation of s_{party_i} converges to its actual behavior within time t*" that is expressed in MOSL by the following formula

$$\mathsf{tt}\ {}_{\top}\mathcal{U}^{\leq t}(\ \langle \text{“}reputationConverged\text{”}, s_{party\_i}\rangle @ s_{rating}\ \rightarrow\ \mathsf{tt}\ )$$

This is an *until* formula[2] of the form $\mathsf{tt}\ {}_{\top}\mathcal{U}^{\leq t}\Phi$, which is satisfied by all the runs that reach within t time units a state satisfying Φ. In this case, formula Φ relies on the *consumption* operator $\langle T\rangle @ s \rightarrow$

[2]Notably, tt and $\top$ are the state formula and the action formula always satisfied, respectively.

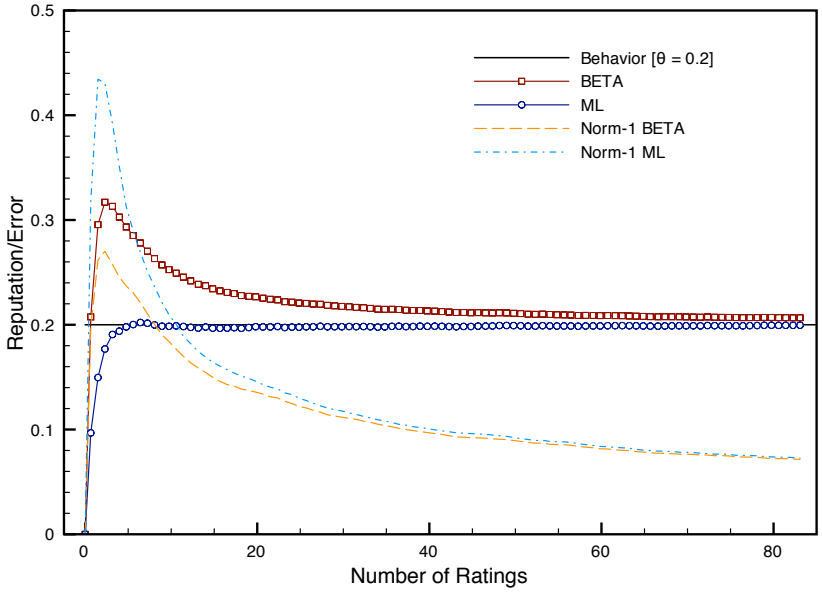

Figure 2: Reputation and error trends for a party with behavior $\theta = 0.2$ and no initial ratings assigned

"the reputation of s_{party_i} converges to its actual behavior within time t"		
Initials Ratings	Beta Model	ML Model
0	0.696688528852	0.651790070646
2	0.404763836437	0.692622253392
4	0.200094638272	0.402222414274

Table 1: Satisfaction probability for party's behavior $\theta = 0.9$ and time $t = 50$

Φ', which is satisfied whenever a tuple matching template T is located at s and the remaining part of the system satisfies Φ' (in this case Φ' simply is tt). Hence, the formula is satisfied if a tuple $\langle \text{“}reputationConverged\text{”}, s_{party\_i} \rangle$ is stored in the node s_{rating}, which happens when the rating server computes a provider's reputation score that is equal to the provider's behavior up to a given error δ. The model checking analysis has been then performed by estimating the total probability of the set of runs satisfying such formula, the maximal time t has been set to 50 seconds and the error δ to 0.05. In Table 1 some analysis results are reported. We consider a party with behavior $\theta = 0.9$ and the presence of 0, 2 or 4 initial ratings that fix parties' initial reputation to 0.5. We observe that in presence of no ratings value at the start of the system, the Beta model achieves better results. Indeed, the satisfaction probability of the formula for the Beta is higher than that for the ML model. Instead, with 2 or 4 initial ratings ML model performs better than Beta model.

3.2 A network-aware evaluation environment

We have developed a software tool, called NEVER (Network-aware EValuation Environment for Reputation systems) [CDT13a], for describing, implementing, evaluating reputation systems while taking into account real-word implementation details of such systems and of the network environment where they have to be deployed.

More specifically, on the one hand, we provide a framework for rapidly developing Java-based implementations of reputation system models and for easily configuring different networked execution environments on top of which the systems will run. On the other hand, we have developed a tool that automatically performs experiments on the reputation system implementations according to

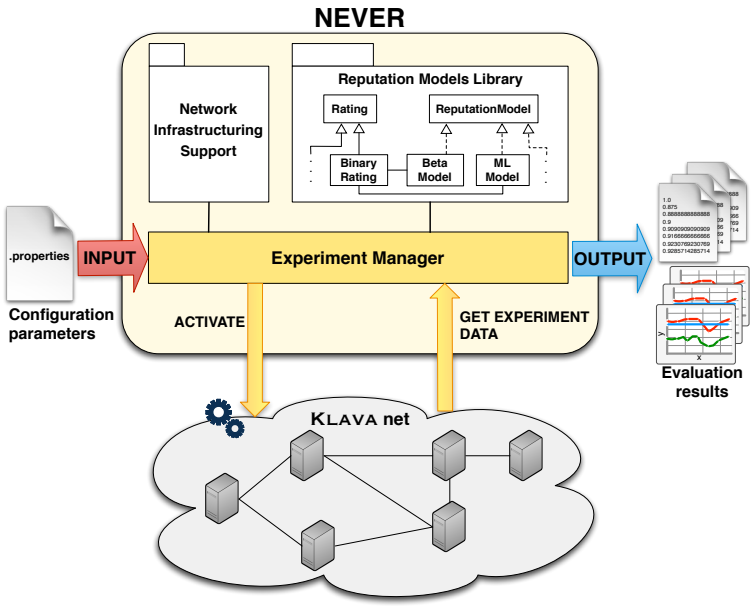

Figure 3: NEVER architecture and workflow

user-specified parameters; this enables the study of their behavior while executing on given network infrastructures. The main novelty of this approach, with respect to other proposals in the literature with a similar aim, is that it allows the evaluation of implemented reputation systems through experiments on real networks, rather than performing simulation of models of reputation systems that abstract from many details. In this way, given a specific network environment, we can study the system behavior to find the configuration that better meets the system requirements by tuning its parameters (reputation model, response timeouts, resource quality evaluation, ratings aging, etc.). Moreover, since we consider reputation systems at implementation level, the analyzed systems could be then directly used in the corresponding end-user applications.

The architecture and the workflow of NEVER is graphically depicted in Figure 3. The NEVER tool consists of three main components: (1) the experiment manager, (2) the network infrastructuring support, and (3) the reputation models library.

The *experiment manager* is the component playing the main role, because it is in charge of managing the execution of each experiment. An experiment consists of a user-specified number of runs, each run performed with the same configuration. The number of runs and their duration, together with other experiments characteristics, are defined by users through configuration parameters.

The *network infrastructuring support* provides the libraries (i.e., classes and interfaces) required to create and set up the network implementing the general infrastructure graphically depicted in Figure 1. In particular, this component of the tool relies on KLAVA [BDP02], a Java library that provides the run-time support for KLAIM actions within Java code. In fact, since the network infrastructure is designed using KLAIM, the use of KLAVA permits a straightforward implementation of such network. Each element of the infrastructure is a node hosted by a (possibly remote and/or virtual) machine. The NEVER tool takes as input the addresses of the hosting machines and automatically activates the nodes forming the wanted network infrastructure.

The *reputation models library* acts as a framework allowing the user to define the trust and reputation models under evaluation. The library is a Java package containing a number of abstract classes and interfaces necessary to implement the models. In this way, the NEVER tool is customizable and extendible by the user. Specifically, a reputation model is defined by a class implementing the `ReputationModel` interface and, possibly, a class extending the abstract class `Rating`. The for-

mer class defines how reputation scores are computed, which rating values are used by the system and how parties in the system evaluate interactions. The latter class defines the kind of rating values and how to manage them. Thus, the addition of new reputation models to NEVER can be achieved by implementing `ReputationModel` and, if necessary, by extending `Rating`.

At the end of the required experiment runs, data are analyzed and provided as output, both in form of textual files and charts. In [CDT13b], we show how NEVER works by means of experimental data obtained through the evaluation of some of the implemented models. As an example, Figure 4 reports a chart produced as output by NEVER showing the reputation trends of four parties with respect to the number of available ratings for each of them.

Notably, given a limited number of physical machines at our disposal, we have performed our experiments by using virtual machines running on the IMT installation of the Zimory Enterprise Cloud [Gmb13].

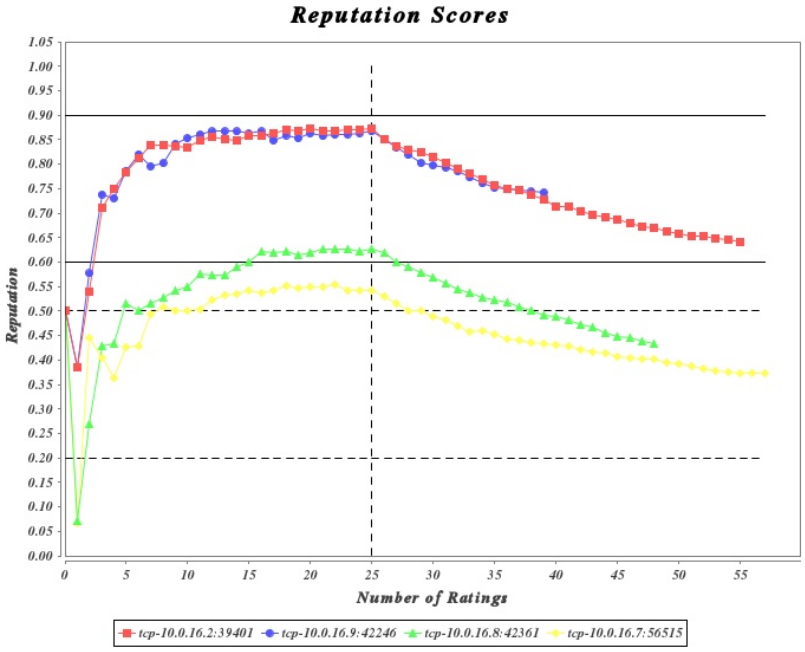

Figure 4: Reputation trend of four parties

4 Model-driven Information Flow Security for Component-Based Systems

Systems and software conceived nowadays knows a continuous increase. Information protection and secure information flow between these systems is paramount and represent a great design challenge. Model driven security (MDS) [BDL06] is an innovative approach that tend to solve system-level security issues by providing an advanced modeling process representing security requirements at a high level of abstraction. Indeed, MDS guarantees separation of concerns between functional and security requirements, from early phases of the system development till final implementation.

Information flow security can be ensured using various mechanisms. Amongst the first approaches considered, ones find access control mechanisms [SSM98, Kuh98], that allow protecting data confidentiality by limiting access to data to be read or modified only by authorized users. Unfortunately, these mechanisms have been proven incomplete and limited since only by preventing the direct access to data, indirect (implicit) information flows are still possible given rise to the so called covert

channels [SQSL05]. As an alternative, non-interference has been studied as a global property to characterize and to develop techniques ensuring information flow security. Initially defined by Goguen and Meseguer [JAJ82], non-interference ensures that the system's secret information does not affect its public behavior.

In this work, we adapt the Model driven security (MDS) [BDL06]approach to develop a component-based framework, named *secBIP*, that guarantees automated verification and implementation of secure information flow systems with respect to specific definition of non-interference. In general, component-based frameworks allow the construction of complex systems by composition of atomic components with communication and coordination operators. That is, systems are obtained from unitary atomic components that can be independently deployed and composed with other components. Component-based frameworks are usually well adopted for managing key issues for functional design including heterogeneity of components, distribution aspects, performance issues, etc. Nonetheless, the use of component-based frameworks is also beneficial for establishing information flow security. Particularly, the explicit system architecture allows tracking easily intra and inter-components information flow.

The *secBIP* framework is built as an extension of the *BIP* [BBB$^+$11] framework encompassing information flow security. *secBIP* allows to create systems that are secure by construction if certain local conditions hold for composed components. The *secBIP* extension includes specific annotations for classification of both data and events. Thanks to the explicit use of composition operators in *BIP*, the information flow is easily tracked within models and security requirements can be established in a compositional manner, first locally, by checking the behavior of atomic components and then globally, by checking the communication and coordination inter-components.

Information flow security has been traditionally studied separately for language-based models [SS01, SV98] (see also the survey [SM03]) and trace-based models [McC88, McL94, ZL97, Man00]. While the former mostly focus on verification of data-flow security properties in programming languages, the latter is treating security in event-based systems. In *secBIP*, we achieve a useful combination between both aspects, data-flow and event-flow security, in a single semantics model. We introduce and distinguish two types of non-interference, respectively *event non-interference* and *data non-interference*. For events, non-interference states that the observation of public events should not allow to deduce any information about the occurrence of secret events. For data, it states that there is no leakage of secret data into public ones.

The rest of our contribution is structured as follows. Subsection 4.1 formally introduces the security extension and we provide the two associated definitions of non-interference, respectively for data flows and event flows. Next, in subsection 4.2 we formally establish non-interference based on unwinding relations and we provide sufficient conditions that facilitate its automatic verification. In subsection 4.3, we provide a use-case as illustrative example. Subsection 4.4 concludes and presents some lines for future work.

4.1 Information Flow Security

We explore information flow policies [DD77, BLP76, JAJ82] with focus on the non-interference property. In order to track information we adopt the classification technique and we define a classification policy where we annotate the information by assigning security levels to different parts of *secBIP* model (data variables, ports and interactions). The policy describes how information can flow from one classification with respect to the other.

As an example, we can classify public information as a Low (L) security level and secret (confidential) information as High (H) security level. Intuitively High security level is more restrictive than Low security level and we denote it by $L \subseteq H$. In general, security levels are elements of a security

domain, defined as follows:

Definition 1 (security domain) *A security domain is a lattice of the form $\langle S, \subseteq, \cup, \cap \rangle$ where:*

- *S is a finite set of security levels.*
- *$\subseteq$ is a partial order "can flow to" on S that indicates that information can flow from one security level to an equal or a more restrictive one.*
- *$\cup$ is a "join" operator for any two levels in S and that represents the upper bound of them.*
- *$\cap$ is a "meet" operator for any two levels in S and that represents the lower bound of them.*

As an example we consider the set $S = \{L, M_1, M_2, H\}$ of security levels that are governed by the "can flow to" relation $L \subseteq M_1$, $L \subseteq M_2$, $M_1 \subseteq H$ and $M_2 \subseteq H$. M_1 and M_2 are incomparable and we note $M_1 \nsubseteq M_2$ and $M_1 \not\supseteq M_2$. This security domain is graphically illustrated in figure 5.

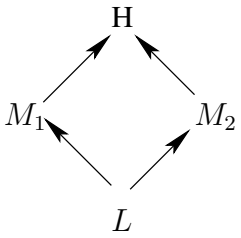

Figure 5: Security domain

We briefly recall the definition of BIP models. For a detailed description the reader can refer to [BBB+11]. We assume that the system is represented by a set of atomic components B_i, $1 \leq i \leq n$, interacting through multiparty interactions (i.e. synchronizations between two or more components). Each component B_i defines an interface consisting of communication ports associated with variables, and its behavior is given by an automaton whose transitions τ are labelled by ports, are guarded by boolean expressions g_τ on variables, and updates values of variables according to functions f_τ. An interaction a between components B_i, $i \in I \subseteq \{1, \ldots, n\}$, is defined as a subset of ports such that it has at most one port of each components, i.e. $a = \{p_i\}_{i \in I}$. Moreover, an interaction a can be guarded by a boolean condition G_a on the variables associated to its ports, and may assigns new values to these variables according to a function F_a. Given a set of interactions γ, we denote by $C = \gamma(B_1, \ldots B_n)$ the composition of $B_1, \ldots, B_n$ with respect to interactions γ. In the composite component C, transitions of atomic components B_i are executed synchronously according to interactions γ, that is, if for an interaction $a = \{p_i\}_{i \in I}$ of γ the guard G_a evaluates to true and all components B_i enable transitions τ_i labelled by p_i, then a can execute if C which corresponds to the synchronous execution of all transitions τ_i after execution of transfer function F_a. A detailed formalization of the model provided in [BBB+11]. In the following, we write X (resp. P) for the set of all variables (resp. ports) defined in all atomic components $(B_i)_{i=1,n}$ of C. We also write Q_C (resp. Q_C^0) for the set of states of C (resp. initial states of C). A security assignment for C with respect to a security domain $\langle S, \subseteq, \cup, \cap \rangle$ assigns security levels to variables, ports and interactions in a consistent way. It is defined as follows.

Definition 2 (security assignment) *A security assignment for component C is a mapping $\sigma : X \cup P \cup \gamma \rightarrow S$ that associates security levels to variables, ports and interactions such that, moreover, the security levels of ports matches the security levels of interactions, that is, for all $a \in \gamma$ and for all $p \in a$ it holds $\sigma(p) = \sigma(a)$.*

In atomic components, the security levels considered for ports and variables allow to track intra-component information flows and control the intermediate computation steps. Moreover, inter-components communication, that is, interactions with data exchange, are tracked by the security levels assigned to interactions.

Let σ be a security assignment for C.

For a security level $s \in S$, we define $\gamma\downarrow_s^\sigma$ the restriction of γ to interactions with security level at most s that is formally, $\gamma\downarrow_s^\sigma = \{a \in \gamma \mid \sigma(a) \subseteq s\}$.

For a security level $s \in S$, we define $w|_s^\sigma$ the projection of a trace $w \in \gamma^*$ to interactions with security level lower or equal to s. Formally, the projection is recursively defined on traces as $\epsilon|_s^\sigma = \epsilon$, $(aw)|_s^\sigma = a(w|_s^\sigma)$ if $\sigma(a) \subseteq s$ and $(aw)|_s^\sigma = w|_s^\sigma$ if $\sigma(a) \not\subseteq s$. The projection operator $|_s^\sigma$ is naturally lifted to sets of traces W by taking $W|_s^\sigma = \{w|_s^\sigma \mid w \in W\}$.

For a security level $s \in S$, we define the equivalence $\approx_s^\sigma$ on states of C. Two states q_1, q_2 are equivalent, denoted by $q_1 \approx_s^\sigma q_2$ iff (1) they coincide on variables having security levels at most s and (2) they coincide on control locations having outgoing transitions labeled with ports with security level at most s.

We are now ready to define the two notions of non-interference.

Definition 3 (event non-interference) *The security assignment σ ensures event non-interference of $\gamma(B_1, \ldots, B_n)$ at security level s iff,*

$$\forall q_0 \in Q_C^0 : \textsc{traces}(\gamma(B_1, \ldots, B_n), q_0)|_s^\sigma = \textsc{traces}((\gamma\downarrow_s^\sigma)(B_1, \ldots, B_n), q_0)$$

Event non-interference ensures isolation/security at interaction level. The definition excludes the possibility to gain any relevant information about the occurrences of interactions (events) with strictly greater (or incomparable) levels than s, from the exclusive observation of occurrences of interactions with levels lower or equal to s. That is, an external observer is not able to distinguish between the case where such higher interactions are not observable on execution traces and the case these interactions have been actually statically removed from the composition. This definition is very close to Rushby's [Rus92] definition for transitive non-interference. But, let us remark that event non-interference is not concerned about the protection of data.

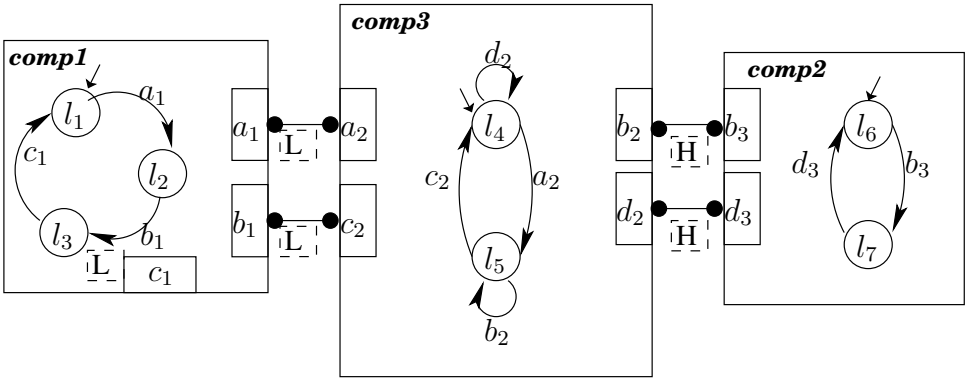

Figure 6: Example for event non-interference.

Example 1 *Figure 6 presents a simple illustrative example for event non-interference. The model consists of three atomic components $comp_{i,i=1,2,3}$. Different security levels have been assigned to ports and interactions: $comp_1$ is a low security component, $comp_2$ is a high security component, and*

comp$_3$ is mixed security component. The security levels are represented by dashed squares related to interactions, internal ports and variables. As a convention, we apply high (H) level for secret data and interactions and low(L) level for public ones. The set of traces is represented by the automaton in figure 7 (a). The set of projected execution traces at security level L is represented by the automaton depicted in figure 7 (b). This set is equal to the set of traces obtained by restricted composition, that is, using interaction with security level at most L and depicted in figure 7 (c). Therefore, this example satisfies the event non-interference condition at level L.

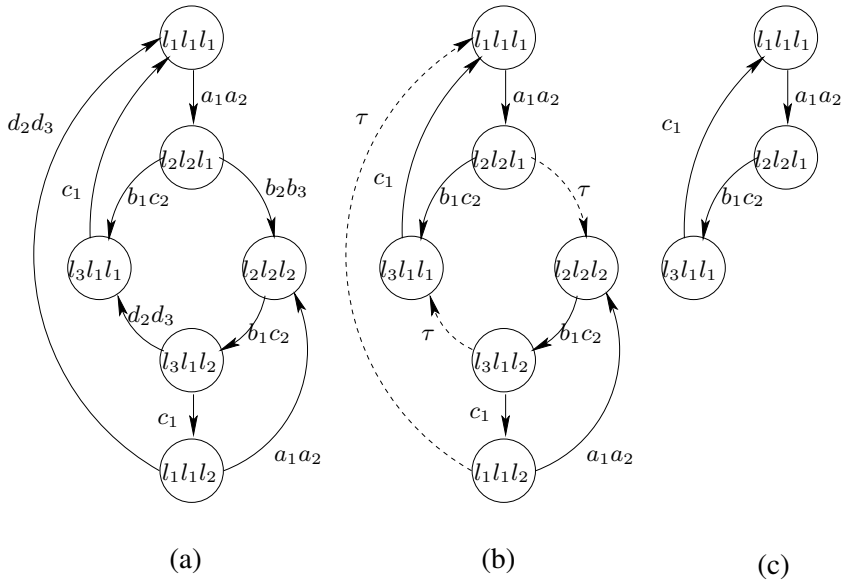

Figure 7: Sets of traces represented as automata.

Definition 4 (data non-interference) *The security assignment σ ensures data non-interference of $C = \gamma(B_1, \ldots, B_n)$ at security level s iff,*

$$\begin{aligned}
&\forall q_1, q_2 \in Q_C^0 : \; q_1 \approx_s^\sigma q_2 \Rightarrow \\
&\quad \forall w_1 \in \text{TRACES}(C, q_1), w_2 \in \text{TRACES}(C, q_2) : \; w_1|_s^\sigma = w_2|_s^\sigma \Rightarrow \\
&\quad\quad \forall q_1', q_2' \in Q_C : \; q_1 \xrightarrow[C]{w_1} q_1' \wedge q_2 \xrightarrow[C]{w_2} q_2' \Rightarrow q_1' \approx_s^\sigma q_2'
\end{aligned}$$

Data non-interference provides isolation/security at data level. The definition ensures that, all states reached from initially indistinguishable states at security level s, by execution of arbitrary but identical traces whenever projected at level s, are also indistinguishable at level s. That means that observation of all variables and interactions with level s or lower excludes any gain of relevant information about variables at higher (or incomparable) level than s. Compared to event non-interference, data non-interference is a stronger property that considers the system's global states (local states and valuation of variables) and focus on their equivalence along identical execution traces (at some security level).

Example 2 *Figure 8 presents an extension with data variables of the previous example from figure 6. We consider the following two traces $w_1 = \langle a_1a_2, b_2b_3,\ c_2b_1, d_2d_3, c_1, a_2a_1\rangle$ and $w_2 =$*

$\langle a_1a_2, b_2b_3, c_2b_1, c_1, a_2a_1\rangle$ *that start from the initial state* $((l_1, u = 0, v = 0), (l_4, x = 0, y = 0), (l_6, z = 0, w = 0))$. *Although the projected traces at level L are equal, that is,* $w_1|^{\sigma}_{L} = w_2|^{\sigma}_{L} = \langle a_1a_2, c_2b_1, c_1, a_1a_2\rangle$, *the reached states by* w_1 *and* w_2 *are different, respectively* $((l_2, u = 4, v = 2), (l_5, x = 3, y = 2), (l_6, z = 1, w = 1))$ *and* $((l_2, u = 4, v = 2), (l_5, x = 2, y = 2), (l_7, z = 1, w = 0))$ *and moreover non-equivalent at low level L. Hence, this example is not data non-interferent at level L.*

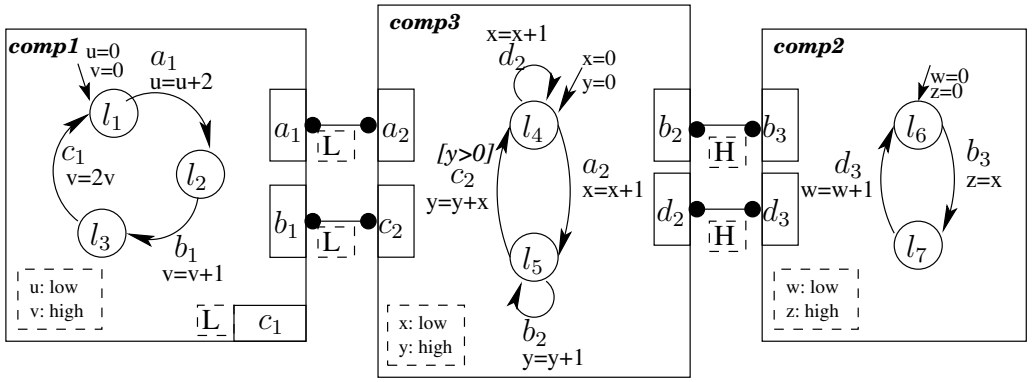

Figure 8: Example for data non-interference.

Definition 5 (secure component) *A security assignment* σ *is secure for a component* $\gamma(B_1, \ldots, B_n)$ *iff it ensures both event and data non-interference, at all security levels* $s \in S$.

4.2 Verification

The verification technique of non-interference proposed for *secBIP* models is using the so-called unwinding conditions. This technique was first introduced by Goguen and Meseguer for the verification of transitive non-interference for deterministic systems [JAJ82]. The unwinding approach reduces the verification of information flow security to the existence of certain unwinding relation. This relation is usually an equivalence relation that respects some additional properties on atomic execution steps, which are shown sufficient to imply non-interference. In the case of *secBIP*, the additional properties are formulated in terms of individual interactions/events and therefore easier to handle.

Let $C = \gamma(B_1, \ldots, B_n)$ be a composite component and let σ be a security assignment for C.

Definition 6 (unwinding relation) *An equivalence* $\sim_s$ *on states of* C *is called an unwinding relation for* σ *at security level* s *iff the two following conditions hold:*

1. *local consistency*
   $\forall q, q' \in Q_C : \forall a \in \gamma : q \xrightarrow[C]{a} q' \Rightarrow \sigma(a) \subseteq s \vee q \sim_s q'$
2. *output and step consistency*
   $\forall q_1, q_2, q_1' \in Q_C : \forall a \in \gamma :$
   $q_1 \sim_s q_2 \wedge q_1 \xrightarrow[C]{a} q_1' \wedge \sigma(a) \subseteq s \Rightarrow$
   $\exists q_2' \in Q_C : q_2 \xrightarrow[C]{a} q_2' \wedge$
   $\forall q_2' \in Q_C : q_2 \xrightarrow[C]{a} q_2' \Rightarrow q_1' \sim_s q_2'$

The existence of unwinding relations is tightly related to non-interference. The following two theorems formalize this relation for the two types of non-interference defined. Let C be a composite component and σ a security assignment.

Theorem 1 (event non-interference) *If an unwinding relation $\sim_s$ exists for the security assignment σ at security level s, then σ ensures event non-interference of C at level s.*

Theorem 2 (data non-interference) *If the equivalence relation $\approx_s^\sigma$ is also an unwinding relation for the security assignment σ at security level s, then σ ensures data non-interference of C at level s.*

The two theorems above can be used to derive a practical verification method of non-interference using unwinding. We provide hereafter sufficient syntactic conditions ensuring that indeed the unwinding relations $\sim_s$ and $\approx_s$ exist on the system states. These conditions aim to efficiently simplify the verification and reduce it to local constrains check on both transitions (inter-component verification) and interactions (intra-component verification). Especially, they give an easy way to automate the verification.

Definition 7 (security conditions) *Let $C = \gamma(B_1, \ldots, B_n)$ be a composite component and let σ be a security assignment. We say that C satisfies the security conditions for security assignment σ iff:*

(i) the security assignment of ports, in every atomic component B_i is locally consistent, that is:

- *for every pair of causal transitions:*

$$\forall \tau_1, \tau_2 \in T_i : \ \tau_1 = \ell_1 \xrightarrow{p_1} \ell_2, \ \tau_2 = \ell_2 \xrightarrow{p_2} \ell_3 \ \Rightarrow$$
$$\ell_1 \neq \ell_2 \ \Rightarrow \ \sigma(p_1) \subseteq \sigma(p_2)$$

- *for every pair of conflicting transitions:*

$$\forall \tau_1, \tau_2 \in T_i : \ \tau_1 = \ell_1 \xrightarrow{p_1} \ell_2, \ \tau_2 = \ell_1 \xrightarrow{p_2} \ell_3 \ \Rightarrow$$
$$\ell_1 \neq \ell_2 \ \Rightarrow \ \sigma(p_1) \subseteq \sigma(p_2)$$

(ii) all assignments $x := e$ occurring in transitions within atomic components and interactions are sequential consistent, in the classical sense:

$$\forall y \in use(e) : \ \sigma(y) \subseteq \sigma(x),$$

where $use(e)$ denotes the set of variables involved in an expression e

(iii) variables are consistently used and assigned in transitions and interactions, that is,

$$\forall \tau \in \cup_{i=1}^{n} T_i \ \ \forall x, y \in X \ : x \in def(f_\tau), y \in use(g_\tau) \ \Rightarrow$$
$$\sigma(y) \subseteq \sigma(p_\tau) \subseteq \sigma(x)$$
$$\forall a \in \gamma \ \ \forall x, y \in X \ : x \in def(F_a), y \in use(G_a) \ \Rightarrow$$
$$\sigma(y) \subseteq \sigma(a) \subseteq \sigma(x),$$

where $def(F)$ denotes the set of variables modified by a function F.

(iv) all atomic components B_i are port deterministic:

$$\forall \tau_1, \tau_2 \in T_i : \ \tau_1 = \ell_1 \xrightarrow{p} \ell_2, \tau_2 = \ell_1 \xrightarrow{p} \ell_3 \ \Rightarrow$$
$$(g_{\tau_1} \wedge g_{\tau_2}) \textit{ is unsatisfiable}$$

The first family of conditions (i) is similar to Accorsi's conditions [AL12] for excluding causal and conflicting places for Petri net transitions having different security levels. Similar conditions have been considered in [FG01, FGF09] and lead to more specific definitions of non-interferences and bisimulations on annotated Petri nets. The second condition (*ii*) represents the classical condition needed to avoid information leakage in sequential assignments. The third condition (*iii*) tackles covert channels issues. Indeed, (*iii*) enforces the security levels of the data flows which have to be consistent with security levels of the ports or interactions (e.g., no low level data has to be updated on a high level port or interaction). Such that, observations of public data would not reveal any secret information. Finally, conditions (*iv*) enforces deterministic behavior on atomic components.

The relation between the syntactic security conditions and the unwinding relations is precisely captured by the following theorem.

Theorem 3 (unwinding theorem) *Whenever the security conditions hold, the equivalence relation $\approx_s^\sigma$ is an unwinding relation for the security assignment σ, at all security level s.*

The following corollary is the immediate consequence of theorems 1, 2 and 3.

Corollary 1 *Whenever the security conditions hold, the security assignment σ is secure for the component C.*

4.3 Application

We illustrate the *secBIP* framework to handle information flow security issues for a typical example, the web service reservation system introduced in [HV06]. A businessman, living in France, plans to go to Berlin for a private and secret mission. To organize his travel, he uses an intelligent web service who contacts two travel agencies: The first agency, *AgencyA*, arranges flights in Europe and the second agency, *AgencyB*, arranges flights exclusively to Germany. The reservation service obtains in return specific flight information and their corresponding prices and chooses the flight that is more convenient for him.

In this example, there are two types of interference that can occur, (1) data-interference since learning the flight price may reveal the flight destination and (2) event interference, since observing the interaction with *AgencyB* can reveal the destination as well. Thus, to keep the mission private, the flight prices and interactions with *AgencyB* have to be kept confidential.

The modeling of the system using *secBIP* involves two main distinct steps: first, functional requirements modeling reflecting the system behavior, and second, security annotations enforcing the desired security policy. The model of the system has four components denoted: *Travel_A* and *Travel_B* who are instances from the same component and correspond respectively to AgencyA and AgencyB, and components *Reservation* and *Payment*. To avoid figure 9 cluttering, we did not represent the interactions with *Travel_A* component. Search parameters are supplied by a user through the *Reservation* component ports *dests* and *dates* to which we associate respectively variables ($from$, to) and $dates$. Next, through search interaction, *Reservation* component contacts *Travel_B* component to search for available flights and obtains in return a list L of specific flights with their corresponding prices. Thereafter, *Reservation* component selects a ticket t_i from the list L and requests the *Payment* component to perform the payment.

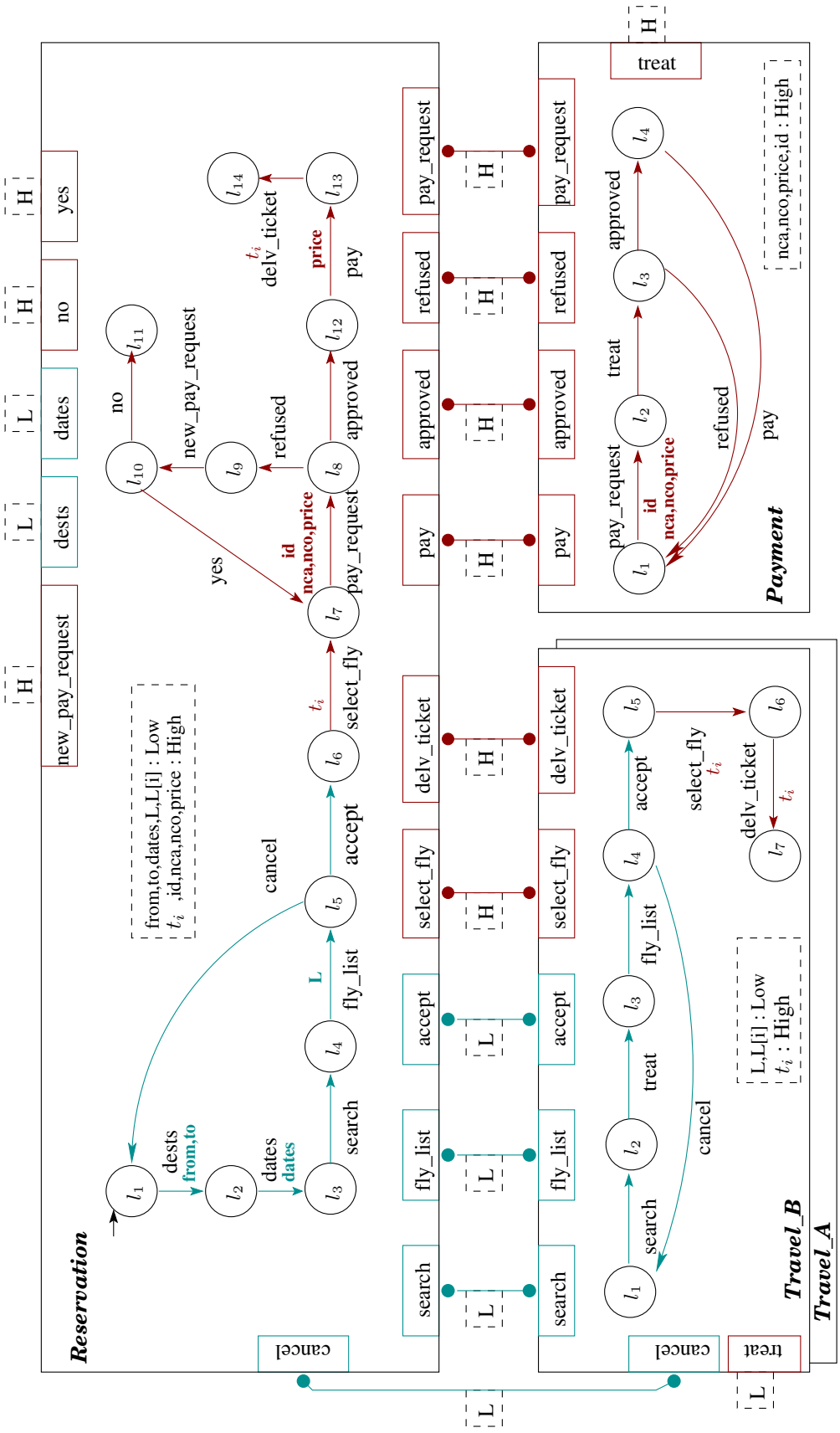

Figure 9: Reservation Web Service composition

All the search parameters $from$, to, $dates$, as well as the flights list L are set to low since users are not identified while sending these queries. Other sensitive data like the selected flight t_i, the price variable p and the payment parameters (identity id, credit card variable cna and code number cno) are set to high. Internal ports *dests* and *dates* as well as *search*, *fly_list*, *accept* interactions are set to low since these interactions (events) do not reveal any information about the client private trip. However, the $select\_fly$ interaction must be set to high since the observation of the selection event from *AgencyB* allow to deduce deduce the client destination. In the case of a selected flight from *AgencyA*, the $select\_fly$ interaction could be set to low since, in this case, the destination could not be deduced just from the event occurrence.

We recall that any system can be proven non-interferent iff it satisfies the syntactic security conditions from definition 7. Indeed, these conditions hold for the system model depicted in Figure 9. In particular, it can be easily checked that all assignments occurring in transitions within atomic component as well as within interactions are sequentially consistent. For example, at the $select\_fly$ interaction we assign a low level security item from the flight list L to a high security level variable ti, formally $t_i = L[i]$. Besides, the security levels assignments to ports exclude inconsistencies due to causal and conflicting transitions, in all atomic components.

4.4 Conclusion

We present a model driven security framework to secure component-based systems. We formally define two types of non-interference, respectively event and data non-interference. We provide a set of sufficient syntactic conditions formulated to simplify non-interference verification. These conditions are extensions of security typed language rules applied to our model. The use of our framework has been demonstrated to secure a web service application.

5 Verification of Timed Systems

We addressed the problem of state-explosion inherent to model-checking of timed systems built from large number of components. Our solution consists in adapting the compositional verification approach of [BBSN08] to timed systems. The main challenge was to be able to capture the relations between the local timing of the components induced by their interactions. Without them the proposed compositional analysis proved to be too weak for verifying even simple systems. The proposed relations take the shape of equalities between the clocks of components used for expressing their timing constraints. We proved the soundness of the proposed approach, and successfully applied it to academic examples and non trivial case studies.

Compositional methods in verification have been developed to cope with state explosion. Generally based on divide-and-conquer principles, compositional methods attempt to break monolithic verification problems into smaller sub-problems by exploiting either the structure of the system or of the property to verify, or both. Compositional reasoning can be used in different flavors, e.g. deductive verification, assume-guarantee reasoning [MC81, Jon83, Pnu84], contract-based verification [Ecdar, LLHSS], compositional generation, etc. Nonetheless, compositional verification has not been very successfully applied to timed systems. The most used tools for the verification of such systems are based on symbolic state space exploration, using efficient data structures and exploration techniques. Few attempts have been made however for exploiting compositionality principles but they remain marginal in the research literature. Nowadays, it is generally admitted that the difficulty for using compositional reasoning is inherently due to the synchronous model of time. Time progress hides continuous synchronization of all the components of the system. Getting rid of such synchronization is necessary for analyzing independently different parts of the system (or of the property) but

also extremely problematic when attempting to re-compose the partial verification results.

We proposed a different approach for exploiting compositionality for analysis of timed systems. We developed a novel compositional method for the generation of invariants for timed systems. In contrast to exact reachability analysis, invariants are symbolic approximations of the set of reachable states of the system. We show that quite precise invariants can be computed compositionally, from the separate analysis of the components in the system and from their composition glue. This method is sound for verification of safety properties, that is, if a given property can be deduced from the invariants computed for the system, then the system is guaranteed to satisfy that property. However, the method is not complete, that is, it may be not able to prove certain properties even if they are satisfied by the system.

Our Compositional Verification Method. The compositional method we propose here is based on the verification rule (VR) from [BBSN08]. Assume that a system consists of n components B_i interacting by means of an interaction set γ, and that the system property of interest is Φ. If components B_i, respectively interactions γ, can be locally characterized by means of invariants $CI(B_i)$, respectively $II(\gamma)$, and if Φ can be proved to be a logical consequence of the conjunction of the local invariants, then Φ is a global invariant. This is what the rule below synthesizes.

$$\frac{\vdash \bigwedge_i CI(B_i) \wedge II(\gamma) \rightarrow \Phi}{\|_\gamma B_i \models \Box\Phi} \text{ (VR)}$$

In the rule (VR), the symbol $\vdash$ is used to underline that the logical implication can be effectively proved (for instance with an SMT solver) and the notation $B \models \Box\Phi$ is to be read as "Φ holds in every reachable state of B".

The key idea behind the compositional generation method is to use additional *history clocks* in order to track the timing of interactions between different components. History clocks allow to decouple the analysis for components and for their composition. On component level, history clocks are used to capture and expose the local timing constraints relevant to their interactions. At composition level, extra constraints on history clocks are enforced due to simultaneity of interactions and to the synchrony of time progress.

Timed Systems. In our framework, the components are timed automata [AD94] and systems are compositions of timed automata with respect to n-ary interactions. Timed automata represent the behavior of components. They have control locations and transitions between these locations. Transitions may have timing constraints, which are defined on clocks. Clocks can be reset and/or tested along with transition execution. Formally, a timed automaton is tuple $(L, l_0, A, T, X, \mathsf{tpc})$ where L is a finite set of control locations, l_0 is an initial control location, A a finite set of actions, X is a finite set of clocks, $T \subseteq L \times (A \times \mathcal{C} \times 2^X) \times L$ is finite set of transitions labeled with actions, guards, and a subset of clocks to be reset, and $\mathsf{tpc} : L \rightarrow \mathcal{C}$ assigns a time progress condition[3] to each location. $\mathcal{C}$ is the set of timing constraints which are predicates on the clocks X defined by the following grammar:

$$C ::= \mathit{true} \mid \mathit{false} \mid x \# ct \mid x - y \# ct \mid C \wedge C$$

with $x, y \in X$, $\# \in \{<, \leq, =, \geq, >\}$ and $ct \in \mathbb{Z}$. Time progress conditions are restricted to conjunctions of constraints as $x \leq ct$. For simplicity, we assume that at each location l the guards of the outgoing transitions imply the time progress condition $\mathsf{tpc}(l)$ of l.

[3]To avoid confusion with invariant properties, we prefer to adopt the terminology of "time progress condition" instead of "location invariants".

A timed automaton is a syntactic structure whose semantics is based on continuous and synchronous time progress. That is, a state is given by a control location paired with real-valued assignments of the clocks. From a given state, a timed automaton can let time progresses when permitted by the time progress condition of the corresponding location, or execute a (discrete) transition if its guard evaluates to true. The effect of time progress of $\delta > 0$ is to increase synchronously all the clocks by the the real value δ. Executions of transitions are instantaneous, that is, they keep values of clocks unchanged except the ones that are reset (i.e. assigned to 0). Because of their continuous semantics, timed automata have in general infinite state spaces. However, they admit finite symbolic representations of their state spaces called zone graph [ACD+92, Alu99, HNSY94, YPD94], in which equivalent assignments of clocks are grouped in a single (symbolic) state call *zone* having the shape of timing constraints defined previously. That is, the reachable states of a timed automata corresponds to a finite number of configurations (l_j, ζ_j), $1 \leq j \leq m$, where for all j, l_j is a control location and ζ_j is a timing constraint.

Examples of timed automata are provided by Figure 10. For instance, components $Worker_i$, $i \in \{1, 2\}$, are implemented by similar timed automata, consisting of two control locations l_1^i and l_2^i and two transitions: a transition from l_1^i to l_2^i labelled by action b_i and having timing constraint $y \geq 8$, and a transition from l_2^i to l_1^i having action d_i and resetting clock y. By convention non displayed guards of transitions and time progress conditions of locations are $true$.

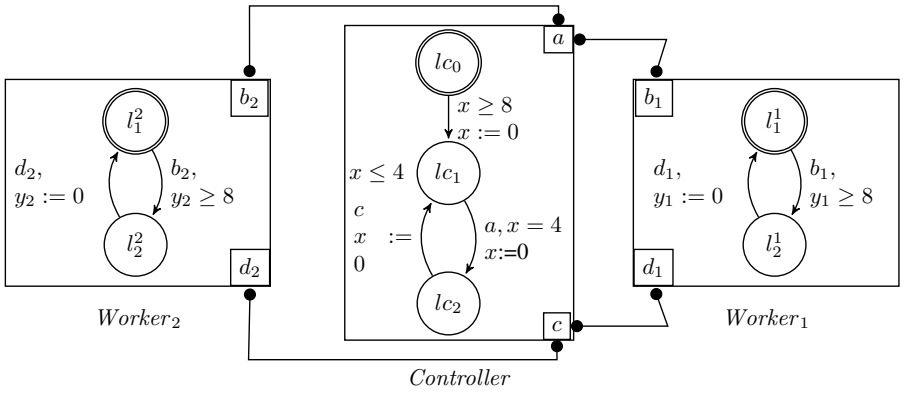

Figure 10: A composition of three timed automata.

In our framework, components interact by means of strong synchronization between their actions. The synchronizations are specified in the so called *interactions* as sets of actions. An interaction can involve at most one action of each component. Given n components (i.e. timed automata) $B_i = (L^i, l_0^i, A^i, T^i, X^i, \mathtt{tpc}^i)$, $1 \leq i \leq n$, and a set of interactions γ, we denote by $\gamma(B_1, \ldots, B_n)$ the composition of components B_i with respect to interactions γ. States of the composition $\gamma(B_1, \ldots, B_n)$ are combinations of the states of the components B_i. In $\gamma(B_1, \ldots, B_n)$, a component B_i can execute an action a_i only as part of an interaction $\alpha \in \gamma$, $a_i \in \alpha$, that is, along with the execution of all the actions participating to α, which corresponds to the usual notion of multi-party interaction. Notice that for a component B_i of a composition $\gamma(B_1, \ldots, B_n)$, the application of interactions γ can only restrict its reachable states. That is, the reachable states of B_i when executed in the composition $\gamma(B_1, \ldots, B_n)$ are included in the reachable states of B_i executed alone (i.e. as a single timed automata). This property is essential for building our compositional verification method, presented below.

Components and Interaction Invariants. To give a logical characterization of a system $S = \gamma(B_1, \ldots, B_n)$ we use invariants. An invariant Φ is a state property which holds in every reachable

state of S, in symbols, $S \models \Box\Phi$.

Component invariants $CI(B_i)$ characterize the reachable states of components B_i when considered alone. Such invariants can easily be computed from the zones of the corresponding timed automata. More precisely, given the reachable (symbolic) states (l_j, ζ_j), $1 \leq j \leq m$, of component B_i, the invariant for B_i is defined by:

$$\bigvee_{1 \leq j \leq m} l_j \wedge \zeta_j,$$

where we abuse of notation and use l_j for the predicate that holds whenever B_i is at location l_j. Notice that zones ζ_j are timing constraints, that is, predicates on clocks. Notice also that invariants $CI(B_i)$ still hold for the composed system $S = \gamma(B_1, \ldots, B_n)$, but are only over approximations of the states reached by each component B_i in S. For example, the component invariants for $Controller$, $Worker_1$ and $Worker_2$ of Figure 10 are as follows:

$$CI(Controller) = (lc_0 \wedge x \geq 0) \vee (lc_1 \wedge x \leq 4) \vee (lc_2 \wedge x \geq 0)$$
$$CI(Worker_i) = (l_1^i \wedge y_i \geq 0) \vee (l_2^i \wedge y_i \geq 8).$$

Interaction invariants $II(\gamma)$ are induced by the synchronizations and have the form of global conditions involving control locations of components. In previous work, we have considered boolean conditions [BBSN08] as well as linear constraints [SBL12] for $II(\gamma)$. For instance, such invariants exclude configurations such that $lc_1 \wedge l_2^i$, that is, they establish $\neg\big(lc_1 \wedge (l_2^1 \vee l_2^1)\big)$. Interaction invariants are not the main purpose of this work, interested readers should refer to [BBSN08] and [SBL12] for detailed presentations.

A safety property of interest for example of Figure 10 is absence of deadlocks. A necessary condition for deadlock freedom is that a can synchronize with b_1 or b_2 when the controller is at lc_1 and the workers are at l_1^i, that is, $\Phi = l_{c_1} \wedge l_1^1 \wedge l_1^2 \implies y_1 - x \geq 4 \vee y_2 - x \geq 4$. Even if Φ holds in S, it cannot be proved by applying (VR) using only component invariants $CI(B_i)$ and interaction invariant $II(\gamma)$. A counter example is given by $lc_1 \wedge l_1^1 \wedge l_1^2$ and $x = y_1 = y_2 = 0$, which satisfies the invariant $CI(Controller) \wedge CI(Worker_1) \wedge CI(Worker_2) \wedge II(\gamma)$[4] but violate property Φ, that is, $CI(Controller) \wedge CI(Worker_1) \wedge CI(Worker_2) \wedge II(\gamma) \not\Longrightarrow \Phi$. One problem is that the proposed invariants cannot relate values of clocks of different components according to their synchronizations (e.g. synchronous reset of clocks).

Adding History Clocks. To strengthen computed invariants, we proposed to equip each component B_i (and later, interactions) with *history clocks*: one clock h_{a_i} per action of a_i of B_i. A history clock h_{a_i} is reset on all transitions executing a_i. Notice that since there is no timing constraint involving history clocks, the behavior of the components remain unchanged after the addition of the history clocks, which shown in [LA]. They are only introduced for establishing properties. Each time an interaction $\alpha \in \gamma$ is executed, all the history clocks corresponding to the actions participating in α are reset synchronously, and then become identical at the next state (until another interaction is executed). Moreover, history clocks of actions of the last executed interaction α are necessarily lower than the ones of actions not participating in α, since they are the last being reset. This is captured by the following invariant:

$$\mathcal{E}(\gamma) = \bigvee_{\alpha \in \gamma} \Big(\big(\bigwedge_{\substack{a_i, a_j \in \alpha \\ a_k \notin \alpha}} h_{a_i} = h_{a_j} \leq h_{a_k} \big) \wedge \mathcal{E}(\gamma \ominus \alpha) \Big),$$

[4] Notice that interaction invariants cannot exclude $lc_1 \wedge l_1^1 \wedge l_1^2$ since it is a reachable configuration.

where $\gamma \ominus \alpha = \{\beta \setminus \alpha \mid \beta \in \gamma \wedge \beta \not\subseteq \alpha\}$. It can be shown that $\mathcal{E}(\gamma)$ is an invariant of the system [LA]. For example of Figure 10, invariant $\mathcal{E}(\gamma)$ is given by:

$$\mathcal{E}(\gamma) \quad = \quad (h_a = h_{b_1} \leq h_{b_2} \vee h_a = h_{b_2} \leq h_{b_1}) \wedge (h_c = h_{d_1} \leq h_{d_2} \vee h_c = h_{d_2} \leq h_{d_1}).$$

Component invariants for example of Figure 10 including the history clocks are as follows:

$$\begin{aligned} CI(Controller^h) &= lc_0 \vee (lc_1 \wedge x \leq 4 \wedge (h_a = h_c \geq 8 + x \ \vee x = h_c \leq h_a)) \vee \\ &\qquad (lc_2 \wedge x = h_a \wedge (h_c \geq h_a + 12 \vee h_c = h_a + 4)) \\ CI(Worker_i^h) &= (y = h_{d_i} \wedge l_1^i \wedge h_{d_i} \leq h_{b_i}) \vee (l_2^i \wedge h_{d_i} \geq 8 + h_{b_i}). \end{aligned}$$

Such invariants proved to be sufficient for stating deadlock-freedom for a similar example involving only one worker, but are too weak for establishing deadlock-freedom with two workers. When interactions are conflicting on shared action a_i, the proposed invariants for history clock h_{a_i} always consider that any of these interactions can execute. For instance, in example of Figure 10 our invariants cannot capture the fact that if action a of $Controller$ synchronizes with b_1 of $Worker_1$, then the following execution of action c of $Controller$ can only synchronize with d_1 of $Worker_1$ (it cannot synchronize with d_2 of $Worker_2$).

Handling Conflicting Interactions. We developed a general way for computing stronger invariants relating execution of the interactions. The principle is to add again history clocks h_α for each the interaction α of γ, and to reset h_α each time α is executed by the means of an additional component and adequate synchronizations. A full description of this approach can be found in [LA]. For an action a_i of component B_i, we define the separation constraint $\mathcal{S}(\gamma, a_i)$ as:

$$\mathcal{S}(\gamma, a_i) = \bigwedge_{\substack{\alpha,\beta \in \gamma \mid a_i \in \alpha,\beta \\ \alpha \neq \beta}} \mid h_\alpha - h_\beta \mid \geq \delta_{a_i},$$

where δ_{a_i} is a lower bound of the time elapsed between two consecutive executions of a_i in B_i, which can be statically computed from the timed automata of B_i. It can be shown [LA] that separation constraints $\mathcal{S}(\gamma, a_i)$ are invariants of the system, that is, the following is an invariant of the system:

$$\mathcal{S}(\gamma) = \bigwedge_{1 \leq i \leq n} \bigwedge_{a_i \in A_i} \mathcal{S}(\gamma, a_i).$$

Invariant $\mathcal{E}(\gamma)$ can be rewritten using additional history clocks as follows:

$$\mathcal{E}(\gamma) = \bigwedge_{1 \leq i \leq n} \bigwedge_{a_i \in A_i} h_{a_i} = \min_{\alpha \ni a_i} h_\alpha.$$

This corresponds to the intuition that the history clock of an action a_i equals the history clock of the last executed interaction α involving a_i, which is the one having h_α minimal.

Experimental Results. We have developed a prototype in Scala implementing the approach. It takes as input components B_i, interactions γ, and a global safety property Φ, and checks whether the system satisfy Φ. To this end, it first computes the invariants proposed above, using PPL [PPL]. Then it generates Z3 [Z3W] Python code to check the satisfiability of the following formula:

$$\bigwedge_{1 \leq i \leq n} CI(B_i) \wedge II(\gamma) \wedge \mathcal{E}(\gamma) \wedge \mathcal{S}(\gamma) \wedge \neg\Phi. \tag{1}$$

Notice that when γ has no conflicting interactions we can simply use the initial form for $\mathcal{E}(\gamma)$ and discard $\mathcal{S}(\gamma)$. If (1) is not satisfiable then the system is guaranteed to satisfy Φ (i.e. our approach is sound). Otherwise, Z3 returns an assignment of the variables satisfying (1) and corresponding to a global state of the system that violates property Φ. Since we use over-approximations (i.e. invariants) instead of the exact behavior of the system, this state may be not reachable and Φ may actually hold in the system.

We experimented the approach on several classical examples, namely the *Train-Gate-Controller* (*TGC*), the *Fischer* mutual exclusion protocol, and the *Temperature-Control-System* (*TCS*). We compared our prototype implementation with Uppaal [Upp]. Uppaal is a widely used model-checker for timed systems implementing symbolic reachability of parallel composition of timed automata using zones. We measured execution times for verifying properties of interest for these examples, i.e. mutual exclusion for *TGC* and *Fischer* and deadlock-freedom for *TCS* (see Table 2). Experimental results shown that Uppaal is subject to state-explosion when increasing the number of components, which happened with *TCS* for 16 components or more, and with *Fischer* for 14 components or more. In contrast, our prototype managed to verify *TCS* even for 124 components in less than 20 seconds. We believe that such compositional approach is very interesting for systems composed of large number of identical components (e.g. swarms of robots) since in this case we reuse already computed invariants following incremental approaches of [BGL+11].

Model & Property	Size	Time/Space	
		Our prototype	Uppaal
Train Gate Controller & mutual exclusion	1	0m0.156s/2.6kB+140B	0ms/8 states
	2	0m0.176s/3.2kB+350B	0ms/13 states
	64	0m4.82s/530kB+170kB	0m0.210s/323 states
	124	0m17.718s/700kB+640kB	0m1.52s/623 states
Fischer & mutual exclusion	2	0m0.144/3kB	0m0.008s/14 states
	4	0m0.22s/6.5kB	0m0.012s/156 states
	6	0m0.36s/12.5kB	0m0.03s/1714 states
	14	0m2.840s/112kB	no result in 4 hours
Temperature Controller & absence of deadlock	1	0m0.172s/840B+60B	0m0.01s/4 states
	8	0m0.5s/23kB+2.4kB	11m0.348s/57922 states
	16	0m2.132s/127kB+9kB	no result in 6 hours
	124	0m19.22s/460kB+510kB	no result in 6 hours

Table 2: Experimental results for model-checking tool Uppaal and our prototype tool.

Conclusion. We have presented a compositional verification method for systems subject to timing constraints. It relies on invariants computed separately from system components and their interactions. This method is sound for verification of safety properties, that is, it can be used to prove that the system cannot reach an undesirable configuration. We believe that it is suited to check correctness of

coordinations within distributed systems, usually implemented by communication protocols relying on time. Its applicability in the project has been related in Deliverable JD3.1, in which we considered an ensemble of robots and devices coordinating in real-time to build consistent knowledge and to achieve safe behavior.

6 Towards Nominal Automata Model Checking

The application of automata-based techniques in software verification dates back to Büchi [Büc60] and Elgot [Elg61]. Later, Vardi and Wolper [VW86] emphasised the relevance of automata-based techniques in verifying temporal properties of programs by reducing logical validity to emptiness of Büchi automata: checking validity is reduced to checking the emptiness of the language of a finite automaton. The study of automata models and their classical properties, especially emptiness, is therefore essential in developing the verification machinery for passing from system specification to model-checking.

However, standard automata-based techniques do not scale up to manage *Ensembles*. Ensembles represent the future generation of software-intensive systems, dealing with massive numbers of components, featuring complex interactions among components, with humans and other systems, operating in open, non-deterministic environments, and dynamically adapting to new requirements, technologies and environmental conditions.

The following example shows the way dynamic entities may be plugged into a computation of SCEL, the ASCENS core language developed in WP1. The code snippet below describes the discovery and the invocation of a remote service:

```
fresh(n).
qry(service, aService, ?u)@P.
put(invoke, aService, v, self, n)@u.
get(result, aService, ?x, n)@self.Pc
```

Here, the name n is a unique session token issued from the runtime environment. The service u to be invoked is searched on the knowledge among all the ones satisfying predicate P. Both n and u are fresh names (i.e. that are not currently used) plugged into the computation, taken from set that are very huge (as the set of tokens) or dynamically changing (as the set of the services available at the moment).

We exploit *nominal techniques* [GP02] to deal with this setting, abstracting the unbounded set of entities that may occur during a computation as an infinite alphabets, the symbols of which are indistinguishable.

Consider the following ML-like script, implementing part of an abstract dispatcher of tasks on sockets.

```
let rec exec() =
   if(...)
     let socket = newsocketfromenv();
     send(socket);
     exec();
     release(socket);
   else ...
```

When the function `exec` invokes the `newsocketfromenv` function, the execution environment yields a fresh socket, so to guarantee an exclusive access. Then the action `send` occurs (we omit it below), the function `exec` gets recursively called and eventually the socket is released. An example

of trace generated during a run is
`new(s1)new(s2)new(s3)...release(s3)release(s2)release(s1)`.
Now, forgetting the actions `new`, `release` and only keeping the names of the sockets (taken from an infinite alphabet), we get a word of the form ww^R (w^R means the reverse word of w), where the symbols in w are all different.

The language $\{ww^R\}$ is intuitively context-free and involves an unbound number of resources, that are all different.

Then, to reason about properties of resource usage patterns we need to investigate automata-based models that deal with unbound number of resources and recognise context-free languages. The literature reports on *regular* automata over infinite alphabet and on their recognizers [KF94, BDFZ, Bol11]. Also *context-free* languages over infinite alphabet have been investigated in [CK98, BKL11, Par12, PP04]. However the latter models do not have a fine-grained control of freshness that permits to tell apart an unbound number of resources nor to *reuse* a resource when not used any-more. Hence we propose PSNA, enriched pushdown automata specifically designed to address these issues. Preliminary to that we introduce and study a regular model for nominal languages: Finite State Nominal Automata (FSNA). Our results show that FSNA are not closed under complement and (full) concatenation, but they are closed under union and also under intersection, provided that reuse is forbidden. We compare the expressive power of our approach with respect to related works in the literature: without reuse our class of regular languages includes that of Usage Automata (UA) [BDFZ] and is incomparable with Variable Automata (VFA) [GKS10] and Finite-memory Automata (FMA) [KF94]. When reuse is allowed, instead, VFA and FMA languages are included in our class.

We propose Pushdown Nominal Automata (PSNA) as a novel model for context-free nominal languages, including the one of [CK98] and expressing, e.g. the traces ww^R above. The results show that our model is more expressive of the one in [CK98] because it can deal with traces ww^R where the restriction that the resources of w being different is relaxed at wish, yet keeping freshness. Even without reuse, our model is more powerful than Usages [BDFZ], while, when this constraint is relaxed, it is also more expressive than *quasi context-free languages* (QCFL) [CK98]. We studied formal languages properties, proving that PSNA are only closed under union, a restricted form of concatenation and, more interestingly, when reuse is inhibited, under intersection with FSNA.

Back to the main goal of verification, in particular about proving safety properties, our results show that the model-checking problem for PSNA is decidable since it is decidable their emptiness problem, provided that the reuse is inhibited.

7 jDEECo Verification

jDEECo is a Java-based implementation of the DEECo component model [BGH+12] runtime framework. It allows for convenient management and execution of jDEECo components and ensemble knowledge exchange. DEECo is, in turn, a software-engineering refinement of the SCEL concepts [BGH+12].

The main tasks of the jDEECo runtime framework are providing access to the knowledge repository, storing the knowledge of all the running components, scheduling execution of component processes (either periodically or when a triggering condition is met), and evaluating membership of the running ensembles and, in the positive case, carrying out the associated knowledge exchange (also either periodically or when triggered). In general, the jDEECo runtime framework allows both local and distributed execution; currently, the distribution is achieved on the level of knowledge repository.

We designed and implemented support for automated verification of jDEECo applications with the Java Pathfinder model checker (JPF) [JPFa]. This work consists of two main steps: (1) making the jDEECo runtime framework amenable to practical verification with JPF and (2) implementing support

for checking specific properties relevant to jDEECo applications. First we provide a brief introduction to JPF and then we describe the two main steps.

Java Pathfinder (JPF) is a highly customizable verification framework for Java programs. The core of JPF is a special JVM that supports non-deterministic thread scheduling choices, non-deterministic data choices (for input values), backtracking, and state matching. Using these mechanisms, JPF systematically explores the state space of a given program, in particular all possible thread interleavings, and looks for specific errors such as assertion violations and deadlocks. During the state space traversal, JPF makes thread scheduling non-deterministic choices at (i) bytecode instructions that access global data (e.g., fields of shared heap objects) and (ii) synchronization primitives (locking, calls of the wait() method, etc). JPF has limited support for Java reflection and other library classes that use native methods (e.g., file I/O and networking).

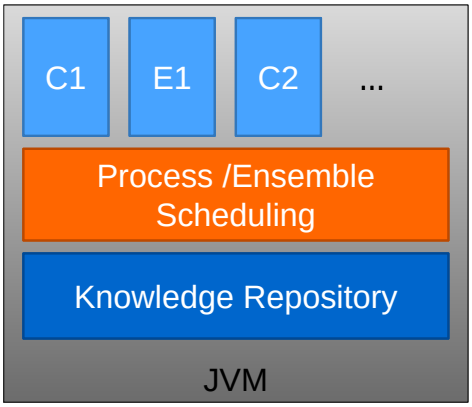

Figure 11: jDEECo: Look inside

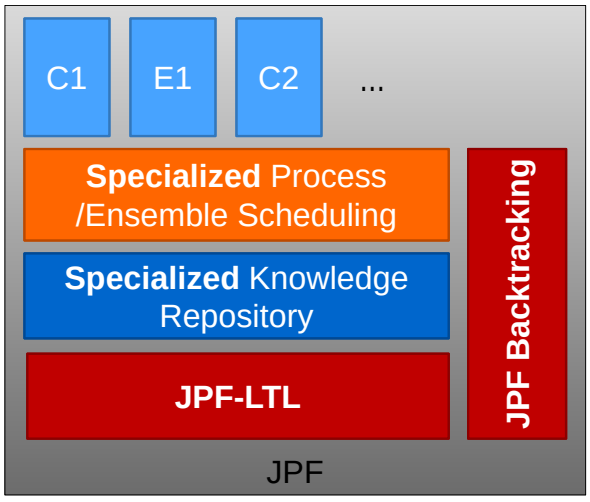

Figure 12: jDEECo + JPF

The main goal in our effort to make the jDEECo framework (Fig. 11) amenable to verification with JPF was to enable systematic traversal of all possible interleavings of sessions executed by component processes, while at the same time mitigating state explosion, i.e. limit the number of unnecessary thread scheduling choices in the state space. We apply JPF to a local version of the jDEECo runtime framework, whose development started in the previous year, in order to precisely verify the concrete

behavior of jDEECo. We do not perform abstraction of any kind. The architecture of the local version being subject to verification is in Fig. 12. Due to the fact that JPF has limited support for native methods (especially reflection), the verification process works as follows. First, all components and ensembles forming the jDEECo application that are subject to verification are serialized into a file. Then, JPF is run on the program consisting of the local version of the jDEECo runtime framework and the given application. Runtime loads all the components and ensembles from the file without the use of reflection, and then starts all the component processes. Note that JPF interprets every action performed by the runtime, starting with the loading of components from the file, up to the finish of the application, and searches for errors. To mitigate state explosion, we configured JPF such that it makes thread scheduling choices at the beginning of each session. This is sufficient in order to let JPF check all interleavings of sessions, because each session makes only a single atomic modification of the knowledge. Components interact only through modifications of the global knowledge. We disabled thread choices at all other places inside the jDEECo framework (e.g., field accesses in classes that implement the knowledge repository). Even though component processes are periodic, JPF does not have to model the periods (real time), because it just has to explore all interleavings of actions that may influence the future behavior of multiple threads. Our solution is to use a thread scheduler that ignores periods and limit the number of iterations of each process by (N*H)/P + 1, where P is the period of a given process, H is the length of one hyperperiod, and N is the number of hyperperiods for which the given application should be tested. The user has to define the number of hyperperiods because this value is typically application-specific. Upon reaching an error state (e.g., an assertion violation), JPF prints the full counterexample, which includes the path leading to the error state and snapshot of the current state. A limitation of this approach is that JPF explores an over-approximation of the set of possible thread interleaving because it does not model periods, and therefore it may report spurious errors that cannot happen in any feasible interleaving of the processes with respect to specific periods.

In the second step of our work, we focused on checking properties of two kinds: temporal behavior of jDEECo applications and data consistency. This includes assertions over values stored in the knowledge repository (e.g., the assertion "$battery.level > 0$" for a component representing a car) and LTL formulas. An example of a LTL formula is "$G(\text{follower\_near\_leader} => F\ \text{follower\_at\_destination})$" for the Convoy demo application, which is a part of the jDEECo distribution. Temporal behavior of the jDEECo applications is interesting because component processes make small steps and it is not clear whether the process will reach the goal state. Atomic propositions in LTL formulas may contain path expressions, arithmetic operators, and logical operators. JPF can search for assertion violations out of the box, so we just had to implement checking of the LTL formulas using JPF. Our solution is based on JPF-LTL [JPFb], which is a third-party extension to JPF that supports checking of LTL formulas. In order to support checking LTL formulas over jDEECo applications, we modified the implementation of JPF-LTL to enable seamless integration into the jDEECo framework and created a new module for jDEECo. The module evaluates atomic propositions based on the content of knowledge repository and stores the current value of each proposition. LTL formulas are checked on-the-fly during the state space traversal by JPF. The process works as follows. When the knowledge repository is updated by some process, our module at the jDEECo side evaluates all atomic propositions and gives the list of satisfied propositions to JPF. At the JPF side, it is then checked whether the LTL formula still holds.

8 Conclusion and Future Work

This year, the most of our efforts were dedicated to security policies and access control. We were interested in privacy of randomization mechanisms, which are used in existing anonymity protocols like Crowds or the Dining Cryptographers to "confound" the adversary as to the true actions undertaken by each participant. We analyzed both worst-case and average-case privacy of such mechanisms. We also

proposed techniques for the design and the implementation of reputation systems, which are involved in a variety of context such as e-commerce, social networking, etc. Most of the time, such systems are deployed without any formal guarantee of correctness. We analyzed how reputation systems behave asymptotically, and how they converge to their asymptotic behavior, using both Bayesian decision theory and stochastic model-checker SAM. For next year we plan to investigate the use of reputation systems to deal with the load balancing issue in an autonomic Cloud environment. Our methodology would help in verifying the effectiveness of different reputation based strategies used to address node selection. A basic problem in such environments is indeed to select the Cloud node capable of successfully executing the incoming application. We also want to extend existing frameworks to different data models taking into account the case in which each rater is characterized by a (unobservable, possibly malicious) bias which can lead him to under- or over-evaluate its interactions with the rates.

Regarding security policies, we proposed a framework for information flow security in component-based systems, in which the security policies are checked early in the design following model-based approaches. We introduced a compositional verification method for non-interference properties. We plan to further extend this work in two directions. First, we are investigating additional security conditions allowing to relax the non-interference property and control where downgrading can occur. Second, we are working towards the implementation of a complete design flow for secure systems based on *secBIP*. As a first step, we shall implement the verification method presented for annotated *secBIP* models. Then, use these models for generation of secure implementations, that is, executable code where the security properties are enforced by construction, at the generation time.

In addition to security and access control, we also completed the verification techniques proposed for the verification of ensembles as explained as follows. We first focused on systems with bounded and static architectures. We extended the verification technique of D-Finder [BBSN08] to encompass timing constraints of components. As shown by the experimental results, and due to its compositional nature, the proposed method scales better to large systems than existing symbolic exploration algorithms such as the ones used in the tool Uppaal [Upp]. One possible extension of this work is to add the support of urgency types [BS00], which we think are more convenient for expressing timing constraints than location invariants. Another extension could be to include invariants on data to have better results for verifying scheduling policies for tasks based systems.

To address dynamic architectures, we developed foundational models using nominal techniques for resource management, and proved several theoretical results. These research are not complete yet, in particular when the disposal and the reuse of resources are allowed. Also, automata are useful when developing verification algorithms, but they cannot be easily used for specifying system, indeed the connection between SCEL and our automata has to be investigated. The effectiveness of our automata has to be validated by showing that the dynamic behavior of the ASCENS case studies can be modelled using our automata.

Finally, we worked towards the verification of service component implementation code by adding a support for jDEECo applications to the Java Pathfinder model checker. Our primary goal for the future is to improve scalability of the verification process. The current approach does not scale well, because each transition in the program state space between two thread choices consists of many bytecode instructions (up to hundreds of thousands), and therefore it takes JPF a very long time to interpret all these instructions (and process such a long transition). A possible solution is to create a simple model (abstraction) of the jDEECo framework that will be analyzed by JPF together with a given application. We will also evaluate our verification approach on larger case studies.

References

[AAC+a] M. S. Alvim, M. E. Andrés, K. Chatzikokolakis, P. Degano, C. Palamidessi. Differential Privacy: on the trade-off between Utility and Information Leakage. In: *FAST* 2011, *LNCS*, 7140,2011.

[AAC+b] M. S. Alvim, M. E. Andrés, K. Chatzikokolakis, C. Palamidessi. Quantitative Information Flow and Applications to Differential Privacy. *FOSAD* VI, *LNCS* 6858: 211-230, 2011.

[AAC+c] M. S. Alvim, M. E. Andrés, K. Chatzikokolakis, C. Palamidessi. On the relation between Differential Privacy and Quantitative Information Flow. *ICALP* (2) 2011: 60-76, 2011.

[ACD+92] Rajeev Alur, Costas Courcoubetis, David L. Dill, Nicolas Halbwachs, and Howard Wong-Toi. An implementation of three algorithms for timing verification based on automata emptiness. In *RTSS*, pages 157–166, 1992.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[AL12] Rafael Accorsi and Andreas Lehmann. Automatic information flow analysis of business process models. In *Proceedings of the 10th international conference on Business Process Management*, BPM'12, pages 172–187. Springer-Verlag, 2012.

[Alu99] Rajeev Alur. Timed automata. In *Proceedings of the 11th International Conference on Computer Aided Verification (CAV)*, LNCS, pages 8–22. Springer, 1999.

[Barb] D. Barber. Bayesian Reasoning and Machine Learning. *Cambridge University Press*, 2012.

[BB12] M. Boreale, M. Paolini. Worst- and average-case privacy breaches in randomization mechanisms. *IFIP TCS 2012*, *LNCS* 7604:72-86, 2012.

[BBB+11] Ananda Basu, Saddek Bensalem, Marius Bozga, Jacques Combaz, Mohamad Jaber, Thanh-Hung Nguyen, and Joseph Sifakis. Rigorous component-based design using the BIP framework. *IEEE Software, Special Edition – Software Components beyond Programming – from Routines to Services*, 28(3):41–48, 2011.

[BBSN08] Saddek Bensalem, Marius Bozga, Joseph Sifakis, and Thanh-Hung Nguyen. Compositional verification for component-based systems and application. In *Proceedings of the 6th International Symposium on Automated Technology for Verification and Analysis*, ATVA '08, pages 64–79, Berlin, Heidelberg, 2008. Springer-Verlag.

[BC13] Michele Boreale and Alessandro Celestini. Asymptotic risk analysis for trust and reputation systems. In *SOFSEM: Theory and Practice of Computer Science*, volume 7741 of *Lecture Notes in Computer Science*, pages 169–181. Springer, 2013.

[BCP09] C. Braun, K. Chatzikokolakis, C. Palamidessi. Quantitative Notions of Leakage for One-try Attacks. *Proc. of MFPS 2009*, *Electr. Notes Theor. Comput. Sci.* 249: 75-91, 2009.

[BDFZ] Massimo Bartoletti, Pierpaolo Degano, Gian Luigi Ferrari, and Roberto Zunino. Model checking usage policies. To appear in *Math. Stuct. Comp. Sci.*, abridged version in TGC 2008, vol 5474 LNCS (2009).

[BDL06] David Basin, Jrgen Doser, and Torsten Lodderstedt. Model driven security: from uml models to access control infrastructures. *ACM Transactions on Software Engineering and Methodology*, 15, 2006.

[BDP02] L. Bettini, R. De Nicola, and R. Pugliese. Klava: a Java Package for Distributed and Mobile Applications. *Software - Practice and Experience*, 32(14):1365–1394, 2002.

[Berg] Berger J. O. Statistical Decison Theory and Bayesian Analysis. *Springer*, 1985.

[BGH+12] Tomas Bures, Ilias Gerostathopoulos, Vojtech Horky, Jaroslav Keznikl, Jan Kofron, Michele Loreti, and Frantisek Plasil. Language Extensions for Implementation-Level Conformance Checking. ASCENS Deliverable D1.5, 2012.

[BGL+11] Saddek Bensalem, Andreas Griesmayer, Axel Legay, Thanh-Hung Nguyen, Joseph Sifakis, and Rongjie Yan. D-finder 2: Towards efficient correctness of incremental design. In Mihaela Gheorghiu Bobaru, Klaus Havelund, Gerard J. Holzmann, and Rajeev Joshi, editors, *NASA Formal Methods*, volume 6617 of *Lecture Notes in Computer Science*, pages 453–458. Springer, 2011.

[BK11] G. Barthe, B. Köpf. Information-theoretic Bounds for Differentially Private Mechanisms. *In 24rd IEEE Computer Security Foundations Symposium, CSF* 2011, 191-204, 2011. *IEEE Computer Society.*

[BKL11] M. Bojańczyk, Bartek Klin, and Slawomir Lasota. Automata theory in nominal sets, 2011. `http://www.mimuw.edu.pl/~sl/PAPERS/lics11full.pdf`.

[BLP76] E. D. Bell and J. L. La Padula. Secure computer system: Unified exposition and multics interpretation, 1976.

[Bol11] Benedikt Bollig. An automaton over data words that captures EMSO logic. In Joost-Pieter Katoen and Barbara König, editors, *CONCUR 2011*, volume 6901 of *LNCS*, pages 171–186. Springer, 2011.

[BS00] Sébastien Bornot and Joseph Sifakis. An algebraic framework for urgency. *Inf. Comput.*, 163(1):172–202, 2000.

[BPP11a] M. Boreale, F. Pampaloni, M. Paolini. Quantitative Information Flow, with a View. *ESORICS 2011*, *LNCS* 6879:588-604, 2011.

[BPP11b] M. Boreale, F. Pampaloni, M. Paolini. Asymptotic information leakage under one-try attacks. *FoSSaCS* 2011, *LNCS* 6604:396-410, 2011. Full version to appear on *MSCS* available at `http://rap.dsi.unifi.it/~boreale/Asympt.pdf`.

[Büc60] J Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960.

[BXE07] A. Boukerch, L. Xu, and K. EL-Khatib. Trust-based security for wireless ad hoc and sensor networks. *Computer Communications*, 30(11-12):2413 - 2427, 2007.

[CDT13a] Alessandro Celestini, Rocco De Nicola, and Francesco Tiezzi. Network-Aware Evaluation Environment for Reputation Systems. In *IFIPTM*, volume 401 of *IFIP Advances in Information and Communication Technology*, pages 231–238. Springer, 2013.

[CDT13b] Alessandro Celestini, Rocco De Nicola, and Francesco Tiezzi. Network-Aware Evaluation Environment for Reputation Systems. Technical Report CSA #5/2013, IMT Institute for Advanced Studies Lucca, 2013. Available at `http://eprints.imtlucca.it/1537/`.

[CDT13c] Alessandro Celestini, Rocco De Nicola, and Francesco Tiezzi. Specifying and analysing reputation systems with a coordination language. In *SAC*, pages 1363–1368. ACM, 2013.

[Cel13] Alessandro Celestini. On the analysis and evaluation of trust and reputation systems, 2013. Phd Thesis, `http://cse.lab.imtlucca.it/rep_sys_eval/thesis.pdf`.

[CK98] Edward Y. C. Cheng and Michael Kaminski. Context-free languages over infinite alphabets. *Acta Inf.*, 35(3):245–267, 1998.

[CPP08a] K. Chatzikokolakis, C. Palamidessi, P. Panangaden. Anonymity protocols as noisy channels. *Information and Computation* 206(2-4): 378-401, 2008.

[CPP08b] K. Chatzikokolakis, C. Palamidessi, P. Panangaden. On the Bayes risk in information-hiding protocols. *Journal of Computer Security* 16(5): 531-571, 2008.

[DA04] Zoran Despotovic and Karl Aberer. A Probabilistic Approach to Predict Peers' Performance in P2P Networks. In *CIA*, volume 3191 of *Lecture Notes in Computer Science*, pages 62–76. Springer, 2004.

[DC] D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology* 1 (1): 65-75, 1988.

[DD77] Dorothy E. Denning and Peter J. Denning. Certification of programs for secure information flow. *Commun. ACM*, pages 504–513, 1977.

[Desp] Z. Despotovic and K. Aberer. A probabilistic approach to predict peers' performance in p2p networks. In 8th International Workshop *CIA* 2004.

[DFLP11] Rocco De Nicola, Gian Luigi Ferrari, Michele Loreti, and Rosario Pugliese. A language-based approach to autonomic computing. In *FMCO*, volume 7542 of *Lecture Notes in Computer Science*, pages 25–48. Springer, 2011.

[DFP98] Rocco De Nicola, Gian Luigi Ferrari, and Rosario Pugliese. KLAIM: A Kernel Language for Agents Interaction and Mobility. *Trans. on Software Engineering*, 24(5):315–330, 1998.

[DKL+07] Rocco De Nicola, Joost-Pieter Katoen, Diego Latella, Michele Loreti, and Mieke Massink. Model checking mobile stochastic logic. *Theor. Comput. Sci.*, 382(1):42–70, 2007.

[DLPT13] R. De Nicola, M. Loreti, R. Pugliese, and F. Tiezzi. SCEL: a language for autonomic computing. Technical Report, January 2013. `http://rap.dsi.unifi.it/scel/pdf/SCEL-TR.pdf`.

[DMNS] C. Dwork, F. McSherry, K. Nissim, A. Smith. Calibrating Noise to Sensitivity in Private Data Analysis. *Proc. of the 3rd IACR Theory of Cryptography Conference*, 2006.

[DP] C. Dwork. Differential Privacy.*ICALP* 2006. *LNCS*, 4052: 1-12, 2006.

[DS05] Y. Dodis and A. Smith. Entropic Security and the encryption of high-entropy messages. *Theory of Cryptography Conference (TCC)*, *LNCS* 3378:556-577, 2005.

[Ecdar] A. David, K. G. Larsen, A. Legay, M. H. Møller, U. Nyman, A. P. Ravn, A. Skou, and A. Wasowski. Compositional verification of real-time systems using Ecdar. *STTT*, 2012.

[EFLR] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. Spki certificate theory. *IETF RFC*, 1999.

[EGS] A. Evfimievski, J. Gehrke, R. Srikant. Limiting Privacy Breaches in Privacy Preserving Data Mining. *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 2003.

[Elg61] Calvin C Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98(1):21–51, 1961.

[FG01] Riccardo Focardi and Roberto Gorrieri. Classification of security properties (part i: Information flow). In *Revised versions of lectures given during the IFIP WG 1.7 International School on Foundations of Security Analysis and Design on Foundations of Security Analysis and Design: Tutorial Lectures*, FOSAD '00, pages 331–396, London, UK, UK, 2001.

[FGF09] Simone Frau, Roberto Gorrieri, and Carlo Ferigato. Formal aspects in security and trust. chapter Petri Net Security Checker: Structural Non-interference at Work, pages 210–225. Berlin, Heidelberg, 2009.

[Fried] A. Friedman, A. Shuster. Data Mining with Differential Privacy. Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD), 2010.

[Gam88] Diego Gambetta. *Trust: Making and Breaking Cooperative Relations*, chapter 13: Can We Trust Trust?, pages 213–237. Basil Blackwell, 1988.

[Ghosh] A. Ghosh, T. Roughgarden, M. Sundararajan. Universally utility-maximizing privacy mechanisms. *In STOC* 2009, 351-360, 2009.

[GKS10] Orna Grumberg, Orna Kupferman, and Sarai Sheinvald. Variable automata over infinite alphabets. In Adrian Horia Dediu, Henning Fernau, and Carlos Martín-Vide, editors, *LATA*, volume 6031 of *LNCS*, pages 561–572. Springer, 2010.

[Gmb13] Zimory GmbH. Zimory Enterprise Cloud, 2013. Web site: `http://www.zimory.de`.

[GP02] M.J. Gabbay and A.M. Pitts. A new approach to abstract syntax with variable binding. *Formal aspects of computing*, 13(3):341–363, 2002.

[Haz05] Philip Hazel. Pcre: Perl compatible regular expressions, 2005. `http://www.pcre.org/pcre.txt`.

[HNSY94] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model checking for real-time systems. *Inf. Comput.*, 111(2):193–244, June 1994.

[HV06] Dieter Hutter and Melanie Volkamer. Information flow control to secure dynamic web service composition. In *In International Conference on Security in Pervasive Computing*, pages 196–210. Springer, LNCS, 2006.

[JAJ82] Goguen Joseph Amadee and Meseguer Jos. Security policy and security models. *In Proceedings of 1982 Symposium on Security and Privecy. IEEE Computer Society Press*, pages 11–20, 1982.

[JI02] Audun Jøsang and Roslan Ismail. The beta reputation system. In *Bled Conference on Electronic Commerce*, 2002.

[JI] A. Jøsang and R. Ismail. The beta reputation system. In Proceedings of *Bled eCommerce Conference*, 2002.

[JIB] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618-644, 2007.

[Jon83] Cliff B. Jones. Specification and design of (parallel) programs. pages 321–332, 1983.

[JPFa] Java PathFinder. `http://babelfish.arc.nasa.gov/trac/jpf/`.

[JPFb] JPF-LTL: An extension to JPF for checking LTL. `https://bitbucket.org/michelelombardi/jpf-ltl`.

[KAS] S. P. Kasiviswanathan, A. Smith . A note on differential privacy: Defining resistance on arbitrary side information. http://arxiv.org/abs/0803.3946, 2008.

[KF94] M. Kaminski and N. Francez. Finite-memory automata. *TCS*, 134(2):329–363, 1994.

[Kifer] D. Kifer, A. Machanavajjhala. No Free Lunch in Data Privacy. *SIGMOD* 2011: 77-88, 2011.

[KS10] B. Köpf, G. Smith. Vulnerability Bounds and Leakage Resilience of Blinded Cryptography under Timing Attacks. *CSF 2010*: 44-56, 2010.

[Kuh98] D. Richard Kuhn. Role based access control on mls systems without kernel changes. In *In Proceedings of the ACM Workshop on Role Based Access Control*, pages 25–32, 1998.

[LA] S. Bensalem M. Bozga J. Combaz L. Astefanoaei, S. Ben Rayana. Compositional invariant generation for timed systems. Technical Report TR-2013-5, Verimag Research Report.

[LHG] S. Lacoste-Julien, F. Husz'ar, and Z. Ghahramani. Approximate inference for the losscalibrated bayesian. In Proceedings of *AISTATS*, 2011.

[LLHSS] S.-W. Lin, Y. Liu, P.-A. Hsiung, J. Sun, and J. S. Dong. Automatic generation of provably correct embedded systems. In *Proceedings of ICFEM*, 2012.

[Lor10] Michele Loreti. SAM: Stochastic Analyser for Mobility, 2010. Available at `http://rap.dsi.unifi.it/SAM/`.

[Man00] Heiko Mantel. Possibilistic definitions of security - an assembly kit. In *Proceedings of the 13th IEEE workshop on Computer Security Foundations*, CSFW '00, pages 185–. IEEE Computer Society, 2000.

[MC81] Jayadev Misra and Kanianthra Mani Chandy. Proofs of networks of processes. page 4:417426, 1981.

[McC88] Daryl McCullough. Noninterference and the composability of security properties. In *Proceedings of the 1988 IEEE conference on Security and privacy*, SP'88, pages 177–186. IEEE Computer Society, 1988.

[McL94] John McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *Proceedings of the 1994 IEEE Symposium on Security and Privacy*, SP '94, pages 79–. IEEE Computer Society, 1994.

[McS] F. McSherry. Privacy Integrated Queries. *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data (SIGMOD)* 2009.

[MMH] L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation. In Proceedings of *HICSS*, 2002., pages 2431-2439, jan. 2002.

[NCL] C. T. Nguyen, O. Camp, and S. Loiseau. A bayesian network based trust model for improving collaboration in mobile ad hoc networks. In *RIVF*, 2007, pages 144 151, 2007.

[NS] A. Narayanan, V. Shmatikov. Robust De-anonymization of Large Sparse Datasets. *Proc. of IEEE Symposium on Security and Privacy*, 2008.

[NT] B.C. Neuman and T. Ts'o. Kerberos: an authentication service for computer networks. Communications Magazine, *IEEE*, 32(9):33-38, sept. 1994.

[Par12] Pawel Parys. Higher-order pushdown systems with data. In Marco Faella and Aniello Murano, editors, *GandALF*, volume 96 of *EPTCS*, pages 210–223, 2012.

[Pnu84] Amir Pnueli. In transition from global to modular temporal reasoning about programs. page 123144, 1984.

[PP04] D. Perrin and J.E. Pin. *Infinite words: automata, semigroups, logic and games*, volume 141 of *Pure and Applied Mathematics*. Elsevier, 2004.

[PPL] PPL. `http://bugseng.com/products/ppl/`.

[Rob] C. P. Robert. The Bayesian Choice. *Springer*, 2001.

[RR] M. K. Reiter, A. D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Trans. Inf. Syst. Secur.* 1(1): 66-92, 1998.

[Rus92] John Rushby. Noninterference, transitivity, and channel-control security policies. Technical report, dec 1992.

[SBL12] Marius Bozga Saddek Bensalem, Benoit Boyer and Axel Legay. Incremental generation of linear invariants for component-based systems. Technical Report TR-2012-15, Verimag Research Report, 2012.

[SKG] S. R. Ganta, S. P. Kasiviswanathan, A. Smith. Composition Attacks and Auxiliary Information in Data Privacy. *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 2008.

[SM03] Andrei Sabelfeld and Andrew C. Myers. Language-based information-flow security. *IEEE Journal on selected areas in communications*, 21(1), 2003.

[Smith] G. Smith. On the Foundations of Quantitative Information Flow. *FoSSaCS 2009*, *LNCS* 5504: 288-302, 2009.

[SNK] V. Sassone, M. Nielsen, and K. Krukow. Towards a formal framework for computational trust. Formal Methods for Components and Objects, *LNCS* 4:175-184, 2007.

[SQSL05] Jianjun Shen, Sihan Qing, Qingni Shen, and Liping Li. Covert channel identification founded on information flow analysis. In *Proceedings of the 2005 international conference on Computational Intelligence and Security - Volume Part II*, CIS'05, pages 381–387. Springer-Verlag, 2005.

[SS01] Andrei Sabelfeld and David Sands. A per model of secure information flow in sequential programs. *Higher Order Symbol. Comput.*, 14(1):59–91, March 2001.

[SSM98] Ravi Sandhu, Ravi S, and Qamar Munawer. How to do discretionary access control using roles, 1998.

[SV98] Geoffrey Smith and Dennis Volpano. Secure information flow in a multi-threaded imperative language. In *Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '98, pages 355–364. ACM, 1998.

[Talwar] F. McSherry, K. Talwar. Mechanism Design via Differential Privacy. *Proceedings Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2007.

[TPJL] W. T. L. Teacy, J. Patel, N. R. Jennings, and M. Luck. Coping with inaccurate reputation sources: Experimental analysis of a probabilistic trust model. In *AAMAS*, 2005.

[Upp] Uppaal. `http://www.uppaal.org/`.

[VW86] Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification. In *LICS*, pages 332–344. IEEE Computer Society, 1986.

[WV] Y. Wang and J. Vassileva. Trust and reputation model in peer-to-peer networks. In *P2P*, 2003, pages 150-157.

[XL] L. Xiong and L. Liu. Peertrust: supporting reputation-based trust for peer-to-peer electronic communities. *IEEE TKDE*, 16(7):843-857, july 2004.

[YPD94] Wang Yi, Paul Pettersson, and Mats Daniels. Automatic verification of real-time communicating systems by constraint-solving. In *FORTE*, pages 243–258, 1994.

[Z3W] Z3. `http://research.microsoft.com/en-us/um/redmond/projects/z3/`.

[ZL97] A. Zakinthinos and E. S. Lee. A general theory of security properties. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, SP '97, pages 94–. IEEE Computer Society, 1997.