www.ascens-ist.eu

ASCENS

Autonomic Service-Component Ensembles

D5.1: First Report on WP5

Verification Techniques for SCs and Correctness Proofs for Negotiate-Commit-Execute Schemes

Grant agreement number: **257414**
Funding Scheme: **FET Proactive**
Project Type: **Integrated Project**
Latest version of Annex I: **7.6.2010**

Lead contractor for deliverable: **UJF-Verimag**
Author(s): **Jacques Combaz, Barbara Jobstmann (UJF-Verimag), Fabio Gadducci (UNIPI), Alberto Lluch Lafuente (IMT), Gianluigi Ferrari (UNIPI), Michele Boreale, Lucia Acciai (UDF), Andrea Vandin (IMT)**

Due date of deliverable: **September 30, 2011**
Actual submission date: **November 15, 2011**
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 our work performed in Year 1 targeted to develop new techniques and theories to support the design and the implementation of correct and reliable service components and service component ensembles. We have worked in several different directions in order to get closer to our goal. They all of have in common that they aim to incorporate quantitative aspects of a SC or SCE (such as probabilistic behaviors, response rates, cost and resource efficiency). Our contributions can be grouped into the following main directions: (1) We have developed a framework and a tool to evaluate and automatically synthesize controllers that behave efficiently with respect to a given cost and reward model. (2) We have developed SMC-BIP (Statistical Model Checker), a stochastic extension of our component-based framework BIP (Behavior-Interaction-Priority) and its toolset. This allows us to analyze component-based systems with stochastic behaviors. (3) We have presented a uniform semantics for name passing calculi, one popular formalisms for the specification of concurrent and distributed systems with a dynamically evolving topology. (4) We have analyzed the theoretical boundaries of behavioral type systems for the pi-calculus, mechanism to enforce desired behavioral properties of SC and SCE. We have also developed an initial translation from SCEL (Service Component Ensemble Language) to BIP. (5) Finally, we have introduced stochastic history expressions, a language to statically analyze security aspects of a system.

Contents

1 Introduction

Our goal *in Workpackage 5* is to develop new techniques and the underlying theories to support the design and the implementation of correct and reliable service components and service component ensembles. 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 an SCs' implementation complies with a high-level specification. We organized the work necessary to achieve our goal in the following four tasks.

5.1 **Verification and Design of Service Components**: This task focuses on algorithmic techniques for verification and design (e.g., MC and synthesis) of SCs with respect to functional and non-functional properties. This includes techniques that extend or improve existing verification and synthesis algorithms, as well as, techniques for verification and synthesis for more complex specifications and properties addressing resource issues or dealing with dynamically changing systems.

5.2 **Verification of Service Component Ensembles**: This task focuses on constructive verification based on compositional reasoning to establish global properties of ensembles from properties of individual service components.

5.3 **Security Policies and Access Control**: This task is dedicated to the study of techniques ensuring security properties of ensembles. We plan to develop a security model – security policies and their enforcement mechanisms – for designing and composing secure ensembles.

5.4 **Verification of SCs' implementation compliance with high-level specification**: This task will develop the code behavior checking techniques which allow checking compliance between implementation of an SC and its high-level specification.

This deliverable summarizes the work performed in Year 1. In Year 1, we have mainly worked on Task 5.1 (Verification and Design of Service Components) and Task 5.3 (Security Policies and Access Control). Work on the two remaining tasks is planed to start in Year 2. In the following we give a detailed description of all four tasks according to the description of work In Section 2, we first summarize our contributions in Year 1 and then give a detailed description of each contribution separately.

Task T5.1: Verification and Design of Service Components.
(Start month: 1, duration 36 months.)

In this task, we study three different generalization of the classical verification setting taking resources and adaptive behaviors into account. In particular, we will model check systems with dynamic allocation of resources, study behavioral types and logics for abstract modeling and verification, and investigate the tools and techniques for Model Checking and Synthesis in a quantitative framework.

Model checking systems with dynamic allocation of resources. The emergence of novel proposals for the use and deployment of connectors, and the "global resources" metaphor interpreting them as suitable service components, require the developments of new foundational methodologies, as well as new verification tools and techniques. Consider for example recent paradigms concerning the specification of distributed systems with changing topologies, often used for security applications, such as nominal calculi. The search for denotational models brought the introduction of new algebraic semantics, e.g. in terms of so-called presheaf models, where the role of names is explicitly taken into account [FMS02]. At the same time, standard finite state techniques proved unsuitable for the handling of name allocation, and fostered new verification tools such as HD-automata [MP05]. The aim of this activity is to build on existing expertise in order to build a formal framework accounting for the specific requirements imposed by service components, in particular their (complex) interaction with the environment, and the possible self-readjustment of their own behavior as a response to such interaction. This will require the development of new foundational models, as well as the enrichment of existing proposals for the verification of systems whose states may have a complex interaction with the environment. The long term goal is to reconcile the denotational and operational views, as done for presheaves and HD-automata [GMM06].

Behavioral types and logics for abstract modeling and verification. In the setting of process calculi, type systems have traditionally been employed to statically enforce safety or liveness communication properties, from simple ones, such as arity mismatch avoidance [Mil91], to more sophisticated ones, such as deadlock freedom [Kob02] and responsiveness [AB08a]. A recent trend is the use of *behavioral* type systems [IK01, CRR02, AB08b], where behavioral abstractions (types) are extracted out of pi-like process specifications and then model-checked against given logical requirements. In order to make this analysis feasible, behavioral types belong to a process calculus (e.g. CCS) more tractable than the original one. The properties of interest are described in a modal logic expressive enough to capture not only behavior-, but also the *spatiality*-related aspects of processes. In this respect, a natural choice is Caires and Cardelli's Spatial logic [CC03]. In a recent work [AB09], decidability of an interesting fragment of this logic has been proved.

We aim at adapting the approach outlined above to the calculi and languages considered in WP1 and the BIP algebra. More specifically, we want to define flexible behavioral type theories that support the assembly of components and ensembles, and can be used as information-gathering tools during the self-modification of SCEs. The basic idea is being able to extract an abstract operational model of the components and of the network accounting for the occurrence of the run-time events of interest (e.g. resource acquisition/consumption, or variation in connectivity or bandwidth, etc.), and then to check their adequacy against given (functional, security, QoS) requirements. Adequacy of ensembles with respect to the given requirements could be assessed by checking separately the abstractions thus obtained, also relying on forms of assume-guarantee reasoning, where the assumptions may be built out of available (typically incomplete) knowledge of the environment. Compared to traditional functional analysis, the new framework calls for an extension of types with quantitative information (probability, cost, . . .) about the events of interest. In order to re-use existing tools, behavioral types must be carefully crafted, as they should basically become the front-end language of an existing model-checker or static analyzer. The new types could then be embedded into a quantitative/probabilistic behavioral representation of the system, aiming at the quantitative evaluation of ensembles, such as the probability that certain (un)desired behavior emerge. The applicability of this framework should not be thought as confined to the design phase of the system. Rather, it should become part of its self-reconfiguration engine, in a Negotiate/Commit/Execute scenario where agents may dynamically and autonomously run verification tools in order to gather (probabilistic) information about the future evolution of the ensemble, given their local knowledge, and use this information in the self-modification process as appropriate.

Model checking and Synthesis in a quantitative Framework. Quantitative constraints such as rewards, timing, or probabilistic specifications have been successfully used to state and analyze non-functional properties such as energy consumption, performance, or reliability (cf. [HKNP, ENLT, Hav98b, HMRT01, BK08b]). Functional properties are typically viewed in a purely qualitative sense.

Desired properties are written in temporal logics and the outcome of verification is a simple Yes or No answer stating that a system satisfies or does not satisfy the desired property. We believe that this black and white view is insufficient both for verification and for synthesis of service components. In particular, for adaptive components a "degree of adaptivity" would be desirable. Therefore, we propose that specifications should have a quantitative aspect.

Our recent research shows that quantitative techniques give new insights into qualitative specifications. In particular, we use probabilities to correct unfeasible specifications, which are specifications that cannot be implemented for instance due to conflicting constraints [CHJ08]. Average-reward properties allow us to express properties like default behavior or preference relations between implementations that all satisfy the functional property [BCHJ09b]. Quantitative constraints also allow us to define a notion of robustness with respect to a functional specification [BGHJ09b]. Intuitively, we classify a system as more robust than another system, if it is able to tolerate more unexpected behaviors of its environment than the other, without violating its specification. Robustness is a key concept that will help dealing with the adaptive nature of SCs.

Apart from robustness, our framework allows the user to state requirements more concisely and more accurately, which is necessary in the presents of adaptive and reconfigurable systems. At the same time, the redefinition of the verification and synthesis problems as quantitative problems yield new theoretical and implementation challenges, which we will address in this task.

Task T5.2: Verification of Service Component Ensembles.
(Start month: 13, duration 36 months.)

In our opinion, any *general* compositional verification theory will be highly intractable and will be of theoretical interest only. We need to study compositionality results for particular classes of properties and/or particular classes of systems as explained below.

Our work on constructive verification led to the development of theory implemented on the D-Finder tool [BBSN08, BBSN09]. D-Finder uses heuristics for proving compositionally global deadlock-freedom of a component-based system, from the deadlock-freedom of its components. The method is compositional and proceeds in two steps.

- First, it checks that individual components are deadlock-free. That is, they may block only at states where they are waiting for synchronization with other components.
- Second, it checks if the components' interaction graph is acyclic. This is a sufficient condition for establishing global deadlock-freedom at low cost. It depends only on the system architecture. Otherwise, D-Finder symbolically computes increasingly strong global invariants of the system, based on results from the first step. Deadlock-freedom is established if there exists some invariant that is satisfied by the system's initial state.

Benchmarks published in [BBSN09] show that such a specialization for deadlock-freedom, combined with compositionality techniques, leads to significantly better performance than is possible with general-purpose monolithic verification tools.

We will investigate compositionality techniques for high-level composition operators and specific classes of properties. We propose to investigate two independent directions:

- One direction is studying techniques for specific classes of properties. For instance, finding compositional verification rules guaranteeing deadlock-freedom or mutual exclusion instead of investigating rules for safety properties in general. Potential deadlocks can be found by analysis of dependencies induced by interactions between components [GS05]. For proving mutual exclusion, a different type of analysis is needed.

- The other direction is studying techniques for particular architectures. Architectures characterize the way interaction among a system's components is organized. For instance, we might profitably study compositional verification rules for ring or star architectures, for real-time systems with preemptable tasks and fixed priorities, for time-triggered architectures, etc. Compositional verification rules should be applied to high-level coordination mechanisms used at the architecture level, without translating them into a low-level automata-based composition.

The results thus obtained should allow us to identify "verifiability" conditions (i.e., conditions under which verification of a particular property and/or class of systems becomes scalable). This is similar to finding conditions for making systems testable, adaptable, etc. In this manner, compositionality rules can be turned into correct-by-construction techniques.

Task T5.3: Security Policies and Access Control.
(Start month: 13, duration 24 months.)

We will investigate secure interaction among service components in an ensemble, considering wrappers that implement Access Control policies, and experiment on using the Negotiate/Commit/Execute paradigm also to deal with Security. Specifically, we will investigate means to: (a) expressing flexible security policies in terms of constraints, building on *C-semirings* and *cc-pi* [BM07]; (b) supporting security decision-making by explicit representation of potential confidentiality leaks and principals' trusting relationships, building on existing information-theoretic [Bor09] and probabilistic models [NKS07]; (c) enforcing secure interaction among service components in an ensemble, considering *wrappers* that implement dynamic access control policies, expressed as polymorphic interface types in context of the MetaKLAIM language [FMP04].

A prominent issue related to security in this setting is expressing flexible and reconfigurable policies that allow components in a *SCE* to self-adapt in reaction to challenges posed by potentially hostile, ever-changing environment and network. *C-semirings* are algebraic structures that allow specifying so-called *soft constraints*, which, upon evaluation, do not just return true or false, but more informative values, of, say, probabilistic or fuzzy nature. C-semirings are a key component of the *cc-pi calculus* [BM07], a model that combines basic features of process calculi and concurrent constraint programming [BMR97]. In cc-pi, constraint resolution between two parties willing to communicate, but possibly posing conflicting requirement on QoS and Security, must happen prior to actual synchronization. In this way, one is able to describe systems not only from the point of view of communication but also from that of negotiation. We plan to enhance the languages developed within work package 1 with constructs inspired by the cc-pi-calculus, thus providing linguistic support to the specification and analysis of dynamical, flexible Access Control policies. Development of reasoning techniques tailored to flexible Access Control based on the resulting model will follow.

Mobility, and more generally change of context, implies that a component might find itself in a hostile environment, or disconnected from its preferred security infrastructure, e.g., its usual certification authorities. Further, the autonomy requirement means that even in this scenario, it must be able to assign privileges to other entities - components or infrastructure elements - based on usually incomplete information about those entities. We plan to support security decision-making by explicit representation of potential confidentiality leaks and principals' trusting relationships. In this respect, we will build on existing information-theoretic of leakage [Bor09] and probabilistic models of computational trust [NKS07].

Using MetaKLAIM, components' interfaces can be modeled via polymorphic types extracted at run-time from code at run-time and used to express trustiness guarantees. The type system for MetaKLAIM permits to ensure security related properties, without resorting to the notion of capability. Types are extracted from code at run-time and used to express trustiness guarantees. Dynamic type checking ensures that the trustiness guarantees of the network applications are maintained whenever trusted components inter-operate with potentially untrusted ones. This is certainly an important aspect of the ensembles of components we want to design. We will however investigate extensions of the types mentioned above, and of the underlying verification mechanism, in order to take into account other security aspects. For example, we will study integrations with the types for controlling access to resources and mobility introduced in KLAIM [DFPV00, GP09].

Finally, we plan to develop a security model – security policies and their enforcement mechanisms – for designing and composing secure ensembles. by extending the approach introduced in [BDFZ08b, BDFZ09b].

Task T5.4: Verification of SCs' implementation compliance with high-level specification.
(Start month: 9, duration 39 months.)

This task will provide techniques for checking that the low-level implementation of an SC is compliant with the high-level formal behavior specification of an SC. In contrast to T5.1 and T5.2, which addresses behavior compliance only for high-level design models, this task will allow to reason about the "business code" of an SC. This makes an important contribution to the SCs development cycle since the business code of an SC is hand-written (i.e. provided by the developer), thus it is prone to be inconsistent with its specification (either when created or when changes occur either in the code or in the high-level specification).

We plan to use the explicit-state model checker GMC (developed by CUNI) separately on individual SCs, thus demonstrating that the implementation of a particular SC fulfills its high-level specification, e.g., in terms of its observable behavior. In this task, we will focus on the business code implementation. Correctness of the communication code is guaranteed by construction, since it is generated directly from the specification.

Further, we will extend the GMC model checker to scale up to large SC's. This will be achieved by adopting techniques from abstraction-based model checking to track some of the SC's data explicitly while using automatically chosen abstract domains for the rest. In addition, we will investigate methods to deduce ownership properties of concurrently shared data structures to further prune the state space to be sought in presence of shared memory concurrency.

2 Discussion on the Work Done

Our work in the Year 1 can be grouped into five main contributions.

1. **Synthesizing Efficient Controllers**
   We present a general framework to evaluate and construct controllers with respect to how efficient they behave in a given probabilistic environment. A controller is a reactive system that regularly observes its environment and then provides control actions to influence the environment in the desired way. Efficiency is defined in terms of a cost model (e.g., energy consumption) and a reward model (e.g., reliability). The cost and the reward model associate to each sequence of actions a number indicating the current costs (and rewards, respectively). The controller aims to find an optimal trade-off between costs and rewards, e.g., to be energy efficient. Given our framework we show how to measure the efficiency of a given controller and how to construct a controller that is optimal, i.e., a controller that minimizes the ratio between the accumulated costs and the accumulated rewards.

2. **Statistical Model-Checking**
   We present SMC-BIP, a stochastic extension of our component-based framework BIP and its toolset. A BIP model consists of a set of components and synchronizations (possibly with priorities) between them. SMC-BIP provides two main features: (1) a stochastic syntax and semantics extension of BIP and (2) a statistical Model Checker (SMC). Our new formalism allows using probabilistic variables assignments to specify stochastic aspects of individual components. The non-determinism resulting from multiply enabled synchronizations is resolving using uniform distribution, therefore SMC-BIP has purely stochastic semantics. Given an SMC-model and a property (either as Bounded Linear Temporal Logic (BLTL) formula or as use-defined trace analyzer) our SMC engine can decide with some confidence whether the system satisfies the given property.

3. **Model Checking Systems with Dynamic Allocation of Resources**
   We analyze name passing calculi, one popular formalisms for the specification of concurrent and distributed systems with a dynamically evolving topology and therefore well-suited for SCE. More precisely, (1) we present a denotational model (based on a presheaf category) for fusion calculi, calculi with complex topologies, where system states are equipped with constraints expressing the identity of some of the allocated names (i.e., resources). (2) We show how to extend these calculi with a hierarchical name structure, which can be used for a white-box approach to network management (i.e., that each system is aware of the topology of the network it is embedded in). (3) In order to tackle the inherent complexity of automated verification in such scenarios, we have developed a state-space reduction technique for rule-based specifications. (4) Finally, we have some preliminary results on extending our reduction technique to exploit name-reuse when reasoning about counterpart models, which are transition systems where states are enriched with information about their internal structure (many-sorted unary algebras) and transitions are labeled with (partial) morphisms, explicitly correlating entities of the source and target states (counterpart relations).

4. **Behavioral Types and Logics for Abstract Modeling and Verification and a translation from SCEL to BIP**
   We studied behavioral types and their properties as a mechanism to enforce desired behavioral properties of SC and SCE. On the theoretical side, we have shown that, under standard complexity-theoretic assumptions, the worst-case complexity of model checking for any arguably interesting fragment of the behavioral type system for the pi-calculus is exponential;

also, we have identified classes of properties for which decidability holds, including safety ones. On the practical side, we have investigated the relationships between a variant of SCEL and BIP and obtained a prototype translation. This translation will make it possible to exploit the BIP framework in WP5, especially to take advantage of the existing analysis tools and instruments. A key point of our translation is modeling the knowledge of each SCEL component (and ensemble) as a BIP component. Any access to Knowledge is then translated into an interaction with this component.

5. **On Quantitative Security Policies**
   We present a way to statically analyze security using a language-based approach. Our main ingredients are: local policies, call-by-contract invocation, type-effect systems, model checking and secure orchestration. Our starting point is the abstraction of system behavior, called history expression, which are processes of a suitable process calculus. Traditionally this approach takes only qualitative aspects of the behavior into account but we believe that quantitative aspect, such as typically rates at which the different activities are performed, are equally important for SC(E).

   In order to handle quantitative aspects with introduced **stochastic history expressions** (HEμ), an extension of history expressions, that associates to each action a rate. We provided a quantitative semantics in terms of continuous-time Markov chains and showed that HEμ is a stochastic extension of Basic Process Algebra with iteration. Our second main contribution is sharpening security policies with quantitative constraints. Roughly, quantitative security policies are safety properties that enforce bounds on the speed at which actions have to be performed. These policies are first class operators inside HEμ, so that security can be taken into account from the very beginning of application development. Given an HEμ expression and a policy we showed how to measure the probability of policy violations using a probabilistic model checker (e.g., PRISM). A high probability does not guarantee conformance with a policy, i.e, there can still be unlikely computations that violate the policy. Therefore, we showed how to use execution monitors that abort such unlikely unsafe computations to enforce a policy during the execution.

A more detailed descriptions of each contribution can be found in subsequent sections.

2.1 Synthesizing Efficient Controllers

Synthesis aims to automatically generate a system from a specification. We focus on synthesizing reactive systems [MP95] from specifications given in temporal logics [Pnu77]. In this setting, specifications are usually given in a qualitative sense, i.e., they classify a system either as good (meaning the system satisfies the specification) or as bad (meaning the system violates the specification). Quantitative specifications assign to each system a value that provides additional information about the system. Traditionally, quantitative techniques are used to analyze properties like response time, throughput, or reliability of a system (cf. [dA97, Hav98a, BK08a, KNP09]).

Recently, quantitative reasoning has also been used to state preference relations between systems satisfying the same qualitative specification [BCHJ09a]. E.g., we can compare systems with respect to robustness, i.e., how reasonable they behave under unexpected behaviors of their environments [BGHJ09a]. A preference relation between systems is particularly useful in synthesis, because it allows the user to guide the synthesizer and ask for "the best" system. In many settings a better system comes with a higher price. E.g., consider an assembly line that can be operated in several modes that indicate the speed of the line, i.e., the number of units produced per step. We would prefer a controller that produces as many units as possible. However, running the line in a faster mode increases the power consumption and the probability to fail, resulting in higher repair costs. We are interested in an "efficient" controller, i.e., a system that minimizes the power and repair costs per produced unit. The efficiency of a system is a natural question to ask; it has also been observed by others, e.g, Yue et al. [YBK10] used simulation to analyze energy-efficiency in MAC Protocol.

We shown how to automatically synthesize a system that has an efficient average-case behavior in a given environment. We defined efficiency as ratio between a given *cost* model and a given *reward* model. To further motivate this choice, consider the following example: assume we want to implement an automatic gear-shifting unit (ACTS) that optimizes its behavior for a given driver profile. The goal of our implementation is to optimize the fuel consumption per kilometer (l/km), a commonly used unit to advertise efficiency. In order to be most efficient, our system has to maximize the speed (given in km/h) while minimizing the fuel consumption (measured in liters per hour, i.e., l/h) for the given driver profile. If we take the ratio between the fuel consumption (the "costs") and the speed (the "reward"), we obtain l/km, the desired measure.

Given an efficiency measure, we ask for a system with an optimal average-case behavior. The average-case behavior with respect to a quantitative specification is the expected value of the specification over all possible behaviors of the systems in a given probabilistic environment [CHJS10]. We describe the probabilistic environment using Markov Decision Processes (MDPs), which is a more general model than the one considered in [CHJS10]. It allows us to describe environments that react to the behavior of the system (like the driver profile).

Formally, given a finite alphabet Σ over a set of events $c, r : \Sigma^* \to \mathbb{N}$ mapping finite sequences of letters to rewards and costs, respectively, the ratio objective $\mathcal{R}_{\frac{c}{r}}$ [BGHJ09a] maps an infinite word $w = w_0 w_1 w_2 \cdots \in \Sigma^\omega$ to the following value:

$$\mathcal{R}_{\frac{c}{r}}(w) = \lim_{n\to\infty} \liminf_{m\to\infty} \frac{\sum_{i=n}^{m} c(w_0 \dots w_i)}{1 + \sum_{i=n}^{m} r(w_0 \dots w_i)} \tag{1}$$

Even though the formula looks rather complicated it is quite intuitive. The two sums are the accumulated rewards of c and r encountered up to position m of the word w. The $+1$ in the denominator avoids division by 0 if the accumulated costs are 0. It has no effect in all other cases. The inner limit ($m \to \infty$) tells us to sum the rewards along the entire infinite word. The outer limit ($n \to \infty$) allows us to ignore a finite prefix of the word. In particular, if the sum in the numerator is finite, then this limit ensures that $\mathcal{R}_{\frac{c}{r}}(w) = 0$. So the value of $\mathcal{R}_{\frac{c}{r}}(w)$ depends only on the infinite part of w. We are interested in evaluating a system with respect to its average-case behavior, which corresponds to the expected value of the function $\mathcal{R}_{\frac{c}{r}}$ over all possible behaviors of the systems in a given probabilistic environment [CHJS10].

Our contributions can be summarized as follows:

1. We developed a framework to automatically construct a system that has an efficient average-case behavior with respect to a reward and a cost model in a probabilistic environment. To the best of our knowledge, this is the first approach that allows to automatically synthesis efficient systems. In our framework, finding an optimal system corresponds to finding an optimal strategy in an MDP with ratio objective.

2. We introduced and study MDPs with ratio objectives .We presented several algorithms to compute optimal strategies in MDPs under ratio objectives. All algorithms are based on decomposing the MDP into end-components [dA97]. The algorithms differ in the way they compute an optimal strategy for a single end-component. One algorithm uses fractional linear programming. The second one, a simple adaption of an algorithm presented in [dA97], is based on a reduction to linear programming. The third algorithm is based on policy iteration and a sequence of reductions to MDPs with long-run average-reward objective. This novel algorithm based on policy iteration is particularly interesting, since it can readily be applied to symbolically encoded MDPs and to large structures [WBB$^+$10]. We compared our framework based on MDPs with ratio objectives to related work and discuss the need for separating the cost and reward model.

3. We have implemented all algorithms in a stand-alone tool and compare them on our examples.In order to increase the scope of our approach, we also integrated the best-performing algorithm into the explicit-state version of PRISM [KNP09], a well-known probabilistic model checker.

For a detailed description of the work, we refer the reader to [vEJ11, vEJ12] .

2.2 Statistical Model-Checking

The association between an expressive modeling formalisms, with sound semantical basis, and efficient analysis techniques and tools is essential for successful model-based development for autonomic systems. While expressivity is needed for obvious reasons (such as mastering heterogeneity and complexity), sound and rigorous models are mandatory in order to establish and reason meaningfully about systems' correctness and performance at design time. Additionally, efficient tools are equally needed to provide, as rapidly as possible, feedback on the models and consequently to assist designers and increase their productivity, during the whole design process.

The BIP [BIP] (Behavior-Interaction-Priority) formalism is such an example of a highly expressive, component-based framework with rigorous semantical basis, developed for supporting development of embedded systems. BIP allows the construction of complex, hierarchically structured models from atomic components characterized by their behavior and their interfaces. Atomic components are composed by layered application of interactions and of priorities. Interactions express synchronization constraints between actions of the composed components while priorities are used to filter amongst possible interactions and to steer system evolution so as to meet performance requirements e.g. to express scheduling policies.

BIP is supported by an extensible tool-set which includes tools for checking correctness, various model transformation and code generation. Correctness can be either formally proven using invariants, that is, assertions automatically generated from BIP models that hold on all executions, or tested using simulation. For the latter case, simulation is driven by a specific middleware, *the BIP engine*, which allows to generate, explore and inspect execution traces corresponding to BIP models. Model transformations allow to realize static optimizations as well as specific transformations towards distributed implementation of models. Finally, code generation targets both simulation and implementation models, for different platforms and operating systems support (e.g., distributed, multi-threaded, real-time, etc.).

We developed a stochastic extension of the BIP formalism and toolset, namely SMC-BIP. Adding stochastic aspects permits to model uncertainty in the design e.g., by including faults or execution platform assumptions. Another advantage is that it allows to combine the simulation engine of BIP with statistical inference algorithms in order to reason on properties in a quantitative manner.

SMC-BIP relies on two key features. The first is a stochastic extension of the syntax and the semantics of the BIP formalism. This extension allows us to specify stochastic aspects of individual components and to produce execution traces of the designed system in a random manner. The second feature is a *Statistical Model Checking (SMC) engine* that, given a randomly sampled finite set of executions/simulations of the stochastic system, can decide with some confidence[1] whether the system satisfies a given property. The decision is taken through either a Monte Carlo (that estimates the probability), or an hypothesis testing algorithm [Wal45, You05] (that compares the probability to a threshold). Due to SMC restrictions, these properties shall be evaluated on bounded executions. Here, we restrict to Bounded Linear Temporal Logic (BLTL), but we also allow the user to plug her own trace analyzer that shall respect a C-Interface. As it relies on sampling executions of a unique distribution, SMC can only be applied to pure stochastic systems i.e., systems without non-determinism. The problem is that most of components-based design approaches exhibit non-determinism due to interleaving semantics, usually adopted for parallel execution of components and their interactions. SMC-BIP allows specifying systems with both non-deterministic and stochastic aspects. However, the semantics of such systems will be purely stochastic, as explained hereafter.

Syntactically, we add stochastic behavior to atomic components in BIP by randomizing individ-

[1]By reasoning on a finite st of executions there is always a probability to make a mistake; the confidence can be given by the user.

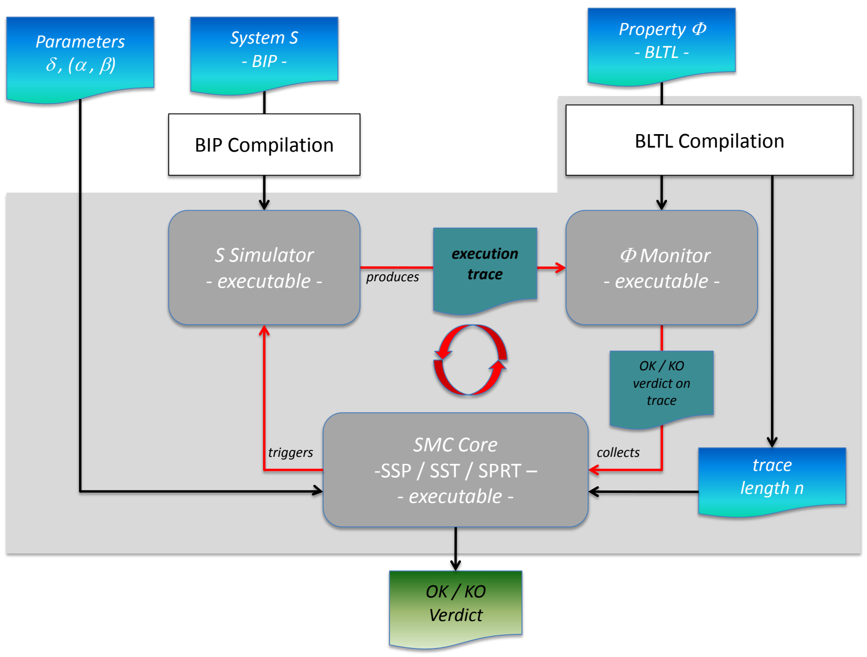

Figure 1: SMC-BIP tool architecture and work flow

ual transitions. Indeed, it suffices to randomize the assignments of variables, which can be done in the C functions used on transition. Hence, from the user point of view, dealing with SMC-BIP is as easy as dealing with BIP (the parser checking that all the assignations are indeed randomized or deterministic. As explained above, this extension leads to the description of stochastic systems with non-deterministic aspects. As an example, an atomic component with two such randomized transitions is nothing more than a Markov Decision Process. BIP allows such a syntactical specification, however the resulting semantics is the one of a pure stochastic system, which is automatically guaranteed as follows. First, observe that synchronizing individual transitions will not add non-determinism. Indeed, to perform such an interaction (synchronization in BIP), we simply compute the product of assignments of individual transitions to get a probability distribution on the resulting interaction. This is allowed as there is no notion of shared variables in BIP, which means that the distributions associated to transitions in separate components are independent. Second, when several such randomized interactions are enabled, the non-determinism is (or can be) partially solved by using priorities. Only interactions with maximal priority are legal for execution, while the other ones are filtered. Third, the remaining non-deterministic choices amongst maximal interactions is solved using a uniform distribution. That is, for example, if two maximal interactions are enabled, each one of them will be selected for execution with probability $\frac{1}{2}$. As a summary, stochastic BIP allows designing systems with both stochastic and non-deterministic aspects (as an example an atomic component with several randomized transitions), but their semantics is purely stochastic.

The structure is given in Figure 1. The tool takes as inputs a system written in the stochastic extension of BIP, a property, and a series of parameters needed by the statistical test (see [You05]). Then the tool creates an executable model and a monitor (the user may also add her own monitor instead of a BLTL formula) for the property. From there, the SMC core engine will generate and monitor executions until a decision can be taken by SMC. As our approach relies on SMC, we are guaranteed that the execution will eventually terminate.

While still at the prototype level, SMC-BIP has been already applied to several case studies coming from serious industrial applications. As an example, in [BBB+10a], we have applied SMC-BIP to estimate the precision of clock synchronization achieved by using the PTP protocol embedded in an aircraft communication network. The full system has more than 2^{3000} states which makes it inaccessible to classical model checking techniques. Using the tool, we have been able to show that the original precision expected by the system designers only holds over 10 percents of the behaviors. We latter estimated the correct value of the synchronization precision, again through SMC, hence correcting the original design. SMC-BIP has been also applied to the analysis of *Avionics Full Duplex Ethernet* (AFDX) [Inc05] networks, that is a network standard developed by Airbus for building highly reliable, time deterministic aircraft data networks based on commercial, off-the shelf Ethernet technology. AFDX provides an abstraction over the communication media as a set of *virtual links*, that is, uni-directional communication channels (from one transmitter towards one or several receivers) with given quality of service in terms of bandwidth and latency constraints. We used the SMC-BIP tool for finding estimates on latencies for particular virtual links in complex AFDX networks. Contrary to existing approaches, ours is capable to retrieve stochastic informations regarding the model [BBB+10b].

2.3 Model Checking Systems with Dynamic Allocation of Resources

One of the long term goals of the sub-task is to "reconcile the denotational and operational views" for specification languages, such as nominal calculi, that can adapt to model systems with an ever-changing topology. The aim of this foundational work is to support the development of novel verification tools and techniques for such languages.

After our initial work in [GMM06], which traces the correspondence between presheaves categories and History Dependent automata [MP05], we aimed for denotational semantics of calculi that feature resource-allocating constructs and efficient verification techniques for these calculi.

In [CKM10] we show how to use symmetries to obtain efficient models of resource binding, thus produce smaller and more manageable automata to be used in verification. In parallel to this work, we analyzed calculi where the handling of the topology is more complex, and system states are equipped with constraints expressing the identity of some of the allocated names (i.e., resources). The research presenting a denotational model for these fusion calculi is coming to fruition now [BBCG11]. Our current work is addressing calculi with an even more complex handling of names, possibly with a hierarchical structure over them, also stimulated by novel calculi proposing a white-box approach to the network management (meaning that each system is aware of the topology of the network it is embedded in), as in [MS11].

On the more operational side, several formalisms for modeling systems with a dynamic handling (including creation or deletion) of resources have been proposed. Notable examples are History Dependent Automata [MP05], High-level Allocational Büchi Automata [DRK02], Graph Transition Systems (e.g. [Ren06]) and Counterpart models (e.g. [GLLV10]). Reasoning about such systems requires to rely on extensions of traditional temporal logics with quantifiers. Several semantics for such logics exist, but there is not a widely accepted agreement on the right semantics. In [GLLV10] we propose a semantics for quantified modal logics for counterpart models, i.e. transition systems where states are enriched with information about their internal structure (many-sorted unary algebras), and transitions are labeled with (partial) morphisms, explicitly correlating entities of the source and target states (counterpart relations).

To tackle the inherent complexity of automated verification in such scenarios we are working on a state space reduction technique [LMV11] for rule-based specifications, mainly inspired by symmetry reduction and abstract interpretation. The main idea is to use canonizer functions mapping each state into a (not necessarily unique) canonical representative of its equivalence class modulo a bisimulation equivalence relation, capturing some specific system regularities. The approach is very flexible and subsumes in an uniform way symmetry reduction as well as other kinds of reductions like name reuse and name abstraction.

Our current efforts are devoted to bring these two lines of research together, i.e. to extend the c-reduction technique to exploit name-reuse when reasoning about counterpart models. Similar issues have been already studied by various authors [DRK02, MP05, Ren06] and we have also taken some preliminary steps into this direction [LV11].

2.4 Behavioral Types and Logics for Abstract Modeling and Verification

In the setting of process calculi, type systems have traditionally been employed to statically enforce safety or liveness communication properties, from simple ones, such as arity mismatch avoidance [Mil91], to more sophisticated ones, such as deadlock freedom [Kob02] and responsiveness [AB08a]. A recent trend is the use of behavioral type systems [IK01, CRR02, AB10], where behavioral abstractions (types) are extracted out of pi-like process specifications and then model-checked against given logical requirements. In order to make this analysis feasible, behavioral types belong to a process calculus (e.g. CCS) more tractable than the original one. The properties of interest are described in a modal logic expressive enough to capture not only behavior-, but also the spatiality-related aspects of processes. In this respect, a natural choice is Caires and Cardelli's Spatial logic [CC03]. In a recent work [AB09], decidability of an interesting fragment of this logic has been proved. We aim at adapting the approach outlined above to the calculi and languages considered in WP1 and the BIP algebra.

During this first year of the project, we have finalized our research on decidability of behavioral type system for the pi-calculus, by extending [AB09] with additional results on complexity and decidability [AB11]. More precisely, we have proved that, under standard complexity-theoretic assumptions, the worst-case complexity of model checking for any arguably interesting fragment of the considered logic is exponential; also, we have identified classes of properties for which decidability holds, including safety ones.

During this year, we have also started investigating the possible relationships between a variant of SCEL and BIP and we have obtained the prototype translation introduced in the following subsection.

From SCEL to BIP—Links with WP1

As a first step towards the bridging of WP1 and WP5 we propose a prototype translation from a variant of SCEL into BIP [BIP]. This translation would make it possible to exploit the BIP framework, especially to take advantage of the existing analysis tools and instruments.

A key point of our translation is modeling the knowledge of each SCEL component (and ensemble) as a BIP component. Any access to Knowledge is then translated into an interaction with this component. This will result in several dyadic interactions between BIP components: one representing the knowledge and the other representing the actual SCEL term. The lower level of BIP components, the behavior, will be described in terms of automata and Petri nets where arcs and transitions, respectively, represent those interactions.

As an example, Figure 2 reports a knowledge component that is receiving a get invocation (via the arc labeled $kget_n$) with retrieval pattern g. In the transition from state i, a built-in function **get** accesses the actual memory and updates the content of variable o, which is then sent back to the invoker by the subsequent transition labeled $kout_n$.

Any component asking for a get from the knowledge should first interact with $kget_n$, by sending a retrieval pattern T, and then wait for the reply by interacting with $kout_n$. This piece of behavior can be described by the Petri net if Figure 3, where the last interaction is highlighted in blue. In the picture, $\mathcal{N}(\llbracket P \rrbracket)$ stands for the translation of the behavior of the SCEL component after the get operation. The other primitives can be translated in the same vein.

Clearly, in order to guarantee the correctness of the analysis with the tools provided by the BIP framework, this translation should be proved to be correct and complete. As a next step and as future works, we plan to finalize the translation, together with any related technical result, for the latest version of SCEL introduced in [NFLP11].

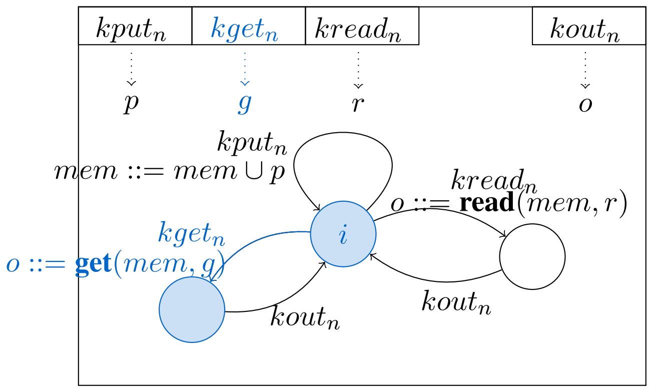

Figure 2: Knowledge component.

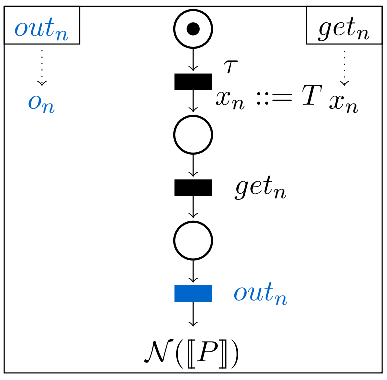

Figure 3: Get from Knowledge.

2.5 On Quantitative Security Policies

In the last few years a new trend is emerging, that exploits the network for computing in a different manner. Applications are no longer built as monolithic entities, rather they are constructed by plugging together computational facilities and resources offered by (possibly) untrusted providers. Illustrative examples of this approach are the Service Oriented, GRID and CLOUD paradigms. Since applications have little or no control of network facilities, security issues became even more acute. The literature has several proposals that address these problems. They can be roughly divided into dynamic, that monitor executions possibly stopping them when insecure; and static, that analyze at binding time the published behavioral interfaces to avoid risky executions.

A language based approach supporting the static analysis of security has been developed in [BDFZ09a, BDFZ08a, BDF09, BDF06]. Its main ingredients are: local policies, call-by-contract invocation, type-effect systems, model checking and secure orchestration. However, this approach only takes into account qualitative aspects of behavior, neglecting quantitative ones, typically the rates at which the different activities are performed. The importance of describing also quantitative aspects of systems is witnessed by several quantitative models and analysis tools that have recently been put forward in the literature. To cite only a few, the stochastic process algebras PEPA[Hil96], the Stochastic π-calculus [Pri95], EMPA [BG96], the stochastic model checker PRISM[KNP02].

We extended the approach of [BDF09] to also deal with quantitative aspects. Our starting point is the abstraction of system behavior, called *history expression*, that are processes of a suitable process calculus.

We extended history expressions by associating a *rate* with actions, so landing in the world of stochastic process calculi. In this way, we obtained *stochastic history expressions* (HEμ). Our first goal is to give them a quantitative semantics in terms of continuous-time Markov chains (CTMC), so making usable well-known techniques for quantitative analysis [BHHK03, KNP07, CGH]. We used a variation of the stochastic kernels over measurable spaces [BDEP97, Pan09] to represent CTMC in the style of [CM10b, CM10a]. To overcome the difficulties with recursion, we restricted stochastic history expressions to a disciplined iteration, namely *binary Kleene star* (for a different approach, see [CM10b]). As a matter of fact, HEμ turn out to be a stochastic extension of BPA$^*_\delta$ [FZ94, BW90].

Our second main contribution is sharpening security policies with quantitative constraints. Roughly, quantitative security policies are safety properties that enforce bounds on the speed at which actions have to be performed. These policies are first class operators inside HEμ, so that security can be taken into account from the very beginning of application development. To express policies we consider CSL$_\mathcal{S}$, a linear subset of CSL [BHHK03, ASSB00].

Because of the inherent stochasticity of our programming model, policies are to be controlled in two complementary modalities: *potential* or *actual*. The first one applies to the CTMC semantics, hence the check is on the *expected* behavior — rates in the CTMC associated with the an HEμ expression e represent the *average* speed of the actions in e. Potential analysis then measures the probability of policy violations. This kind of verification can be carried out through a probabilistic model checker, e.g. PRISM [KNP02].

The actual control can only be done dynamically, because in a specific, unlikely computation, the actual speed of an action can greatly deviate from its rate. Security is then enforced during the execution through an *execution monitor* aborting such a unlikely, unsafe computation.

Potential verification enables a user to accept/discard an application when the probability of a security violation is below/above a certain threshold he feels acceptable. Complementary, actual monitoring will stop the unwanted execution, so guaranteeing security.

To clarify our formal development, we introduce below a simple example. We want to analyze a system, the behavior of which is specified by the following process. The system starts a race between actions a and c. In the case a is the first to complete, b is performed and then the whole process restarts.

$$\begin{tikzcd}
\text{start} \arrow[r] & h \arrow[loop above, "{(c,1)}"] \arrow[r, bend left, "{(a,0.5)}"] & {(b,1)\cdot h} \arrow[l, bend left, "{(b,1)}"]
\end{tikzcd}$$

Table 1: CTMC associated with h

Otherwise, if c wins the race, the process restarts right after c completion. We model the expected execution speed by associating to each action a rate, used then as the parameter of an exponentially distributed random variable. For simplicity, suppose that here the action a has rate 0.5, while the actions b and c have both rates 1.

We model the system above through the following HEμ expression $h = (((a, 0.5) \cdot (b, 1)) + (c, 1)) \,^*\, \delta$. The operator * is the binary Kleene star, that expresses the iteration of the process (the δ is the deadlock process preventing the iteration to terminate). As said, the long term behavior of HEμ expressions is conveniently specified by a CTMC. In our case, we give a graphical representation of the semantics of h in Figure 1. The syntax and the semantics of the stochastic history expressions are formally defined in [DFM11].

Assume now that the system has to respect a quantitative actual policy ϕ saying that "action a must never last more than 1 second". This policy is to be reflected into a potential requirement, expressing that, in the long-run, the system will violate ϕ with low probability. A suitable CSL$_S$ formula that represents this potential quantitative policy is $\psi = \mathcal{C}_{\leq 1\%}(\phi)$. We omit here the details and only read ψ as: the computations violating ϕ are less than 1% of the total.

We now verify whether the expression h respects the potential policy ψ or not. To this purpose, we compute the vector of the steady state distribution of the CTMC associated with h. Each entry of the vector expresses the portion of time spent in each part of the computation. The steady distribution of the CTMC in Figure 1 is $[0.\bar{6}, 0.\bar{3}]$. The first entry is related with the part where a and c are racing, the second one with the part when b is executing. By standard reasoning on the properties of exponential random variables, the probability that a lasts longer than 1 second is $p = 0.36$, the probability that the action a wins the race is $q = 0.\bar{3}$. Hence, we obtain that ψ is violated because the probability that ϕ is violated is about 8%. Indeed, multiplying $0.\bar{6}$, the first entry of the vector (when action a is executing), by q (the probability that a is the one that completes) and by p (the probability that the duration of a violates ϕ) we get about 0.08.

This analysis suggests to deploy the system equipped with a monitoring mechanism that abort an execution when it is about to violate ϕ.

3 Conclusion and Summary of the Main Achievements

During the first year, we have worked in the following research directions incorporating quantitative aspects of a SC or SCE:

- We developed a framework to (i) evaluate the efficiency of a controller and (ii) construct a controller that behaves most-efficient in a given probabilistic environment. Efficiency is defined in terms of a cost model (e.g., energy consumption) and a reward model (e.g., reliability). The controller aims to find an optimal trade-off between costs and rewards.
- We developed SMC-BIP, a statistical model checking method and tool for the BIP framework. SMC-BIP permits to model uncertainty in the design and can verify quantitative properties. Moreover, it can handle large models that cannot be verified using classical model checking techniques, as shown by an avionics case study for which we successfully applied SMC-BIP.
- We used the SMC-BIP tool to estimate (1) the precision of clock synchronization in an aircraft communication network using the PTP protocol and (2) the latencies for particular virtual links in complex AFDX (Avionics Full Duplex Ethernet) networks.
- We have presented a denotational model (based on a presheaf category) for fusion calculi, calculi with complex topologies, where system states are equipped with constraints expressing the identity of some of the allocated names (i.e., resources).
- We have developed a state-space reduction technique for rule-based specifications that allows us to verify larger models.
- We have finalized our research on behavioral type systems for the pi-calculus. We have identified a decidable fragment of Spatial logic [CC03] and we have proven that any model checking algorithm for this logic is characterized by exponential complexity.
- We have started investigating the relationship between a variant of SCEL and BIP and proposed a prototype translation of the former into the latter.
- We have introduced stochastic history expressions (HEμ), a formalism to statically check quantitative aspects of security issues in a distributed context.

4 Next Steps and Long-Term Technical Goals

For the rest of the project, we planed to work on the directions including the following:

- We will develop a symbolic version of our synthesis algorithm for constructing efficient controllers. This will allow us to control systems of large size and complexity. We plan to integrate this algorithm into the probabilistic model checker PRISM in order to combine our synthesis technique with more traditional verification techniques to enable the construction of correct and efficient systems. Our notion of efficiency also allows us to construct systems that are "robust" with respect to given a specification, because robustness can be seen as a trade-off between violations of the desired system constraints (the costs) and violations of the assumed environment constraints (the rewards). We envision to evaluate our symbolic implementation by constructing efficient (robust) controllers for models of SC and SCE.
- For verification of BIP models, we will work on the following topics:
  - a new technique based on stochastic abstraction to address the challenge of combinatorial complexity in verification of heterogeneous systems,
  - the design of new and efficient statistical model checking algorithms that exploit the structure of the system/property through Bayesian and rare event simulation theories, and
  - the integration of our methods in an industrial context.
- We plan to investigate suitable classes of nominal automata such as HD-automata to specify and verify resource usage policies of service components. The first step of our research will consists on extending nominal techniques to manage general resources rather than simple names. Roughly, we will represent resources as nominal entities: structures that can bind names. Usage policies will be abstractly represented as sets of words with multiple data values, namely the language recognized by nominal automata. In this setting, checking correctness of usage policies will result into a model checking problem. To this purpose, we plan to extend the classical automata-like model checking techniques to the case of nominal automata. Finally, we intend to apply these nominal techniques to manage SCEL policies.
- During the next years of the project, we plan to start investigating whether behavioral-type systems for the SCEL language can be defined. The first step of our investigation will consist in defining how to extract an abstract operational model of the components for the occurrence of the run-time events of interest. Such a model should give a quantitative/probabilistic behavioral representation of the whole system. The second step will consist in the definition of a logic, suitable to define qualitative and quantitative properties of interest. Logical correspondence results between the abstractions and the original systems will guarantee correctness and completeness of our proposal.

References

[AB08a] Lucia Acciai and Michele Boreale. Responsiveness in process calculi. *Theor. Comput. Sci.*, 409(1):59–93, 2008.

[AB08b] Lucia Acciai and Michele Boreale. Spatial and behavioral types in the pi-calculus. In Franck van Breugel and Marsha Chechik, editors, *CONCUR*, volume 5201 of *Lecture Notes in Computer Science*, pages 372–386, 2008.

[AB09] Lucia Acciai and Michele Boreale. Deciding safety properties in infinite-state pi-calculus via behavioural types. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *ICALP (2)*, volume 5556 of *Lecture Notes in Computer Science*, pages 31–42, 2009.

[AB10] Lucia Acciai and Michele Boreale. Spatial and behavioral types in the pi-calculus. *Inf. Comput.*, 208(10):1118–1153, 2010.

[AB11] Lucia Acciai and Michele Boreale. Deciding safety properties in infinite-state pi-calculus via behavioural types, 2011. Submitted for publication, Information and Computation.

[ASSB00] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model-checking continuous-time Markov chains. *ACM Transactions on Computational Logic (TOCL)*, 1(1):170, 2000.

[BBB$^+$10a] A. Basu, S. Bensalem, M. Bozga, B. Caillaud, B. Delahaye, and A. Legay. Statistical abstraction and model-checking of large heterogeneous systems. In *FORTE*, volume 6117 of *LNCS*, pages 32–46. Springer, 2010.

[BBB$^+$10b] A. Basu, S. Bensalem, M. Bozga, B. Delahaye, A. Legay, and E. Siffakis. Verification of an afdx infrastructure using simulations and probabilities. volume 6418 of *LNCS*. Springer, 2010.

[BBCG11] Filippo Bonchi, Marzia Buscemi, Vincenzo Ciancia, and Fabio Gadducci. A presheaf environment for the explicit fusion calculus. *Journal of Automated Reasoning*, 2011. to appear.

[BBSN08] Saddek Bensalem, Marius Bozga, Joseph Sifakis, and Thanh-Hung Nguyen. Compositional Verification for Component-Based Systems and Application. In *ATVA*, pages 64–79, 2008.

[BBSN09] Saddek Bensalem, Marius Bozga, Joseph Sifakis, and Thanh-Hung Nguyen. D-Finder: A Tool for Compositional Deadlock Detection and Verification. In *CAV*, 2009.

[BCHJ09a] R. Bloem, K. Chatterjee, T. A. Henzinger, and B. Jobstmann. Better quality in synthesis through quantitative objectives. In *CAV*, pages 140–156, 2009.

[BCHJ09b] Roderick Bloem, Krishnendu Chatterjee, Thomas Henzinger, and Barbara Jobstmann. Better Quality in Synthesis through Quantitative Objectives. In *Computer Aided Verification (CAV)*, 2009.

[BDEP97] R. Blute, J. Desharnais, A. Edalat, and P. Panangaden. Bisimulation for labelled Markov processes. 1997.

[BDF06] M. Bartoletti, P. Degano, and G.L. Ferrari. Types and effects for secure service orchestration. In *CSFW*, pages 57–69, 2006.

[BDF09] M. Bartoletti, P. Degano, and G.L. Ferrari. Planning and verifying service composition. *Journal of Computer Security*, 17(5):799–837, 2009.

[BDFZ08a] M. Bartoletti, P. Degano, G.L. Ferrari, and R. Zunino. Semantics-based design for secure web services. *IEEE Trans. Software Eng.*, 34(1):33–49, 2008.

[BDFZ08b] Massimo Bartoletti, Pierpaolo Degano, Gian Luigi Ferrari, and Roberto Zunino. Semantics-Based Design for Secure Web Services. *IEEE Trans. Software Eng.*, 34(1):33–49, 2008.

[BDFZ09a] M. Bartoletti, P. Degano, G.L. Ferrari, and R. Zunino. Local policies for resource usage analysis. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 31(6):23, 2009.

[BDFZ09b] Massimo Bartoletti, Pierpaolo Degano, Gian Luigi Ferrari, and Roberto Zunino. Local policies for resource usage analysis. *ACM Trans. Program. Lang. Syst.*, 31(6), 2009.

[BG96] M. Bernardo and R. Gorrieri. Extended markovian process algebra. *CONCUR'96: Concurrency Theory*, pages 315–330, 1996.

[BGHJ09a] R. Bloem, K. Greimel, T. A. Henzinger, and B. Jobstmann. Synthesizing robust systems. In *FMCAD*, pages 85–92, 2009.

[BGHJ09b] Roderick Bloem, Karin Greimel, Thomas Henzinger, and Barbara Jobstmann. Synthesizing Robust Systems. In *Conference on Formal Methods in Computer Aided Design (FMCAD'09)*, 2009. Accepted for publication.

[BHHK03] C. Baier, B. Haverkort, H. Hermanns, and J.P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on software engineering*, 29(6):524–541, 2003.

[BIP] `http://www-verimag.imag.fr/Rigorous-Design-of-Component-Based.html?lang=en`.

[BK08a] C. Baier and J.-P. Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[BK08b] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[BM07] Maria Grazia Buscemi and Ugo Montanari. Cc-pi: A constraint-based language for specifying service level agreements. In Rocco De Nicola, editor, *ESOP*, volume 4421 of *Lecture Notes in Computer Science*, pages 18–32. Springer, 2007.

[BMR97] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Semiring-based constraint satisfaction and optimization. *J. ACM*, 44(2):201–236, 1997.

[Bor09] Michele Boreale. Quantifying information leakage in process calculi. *Inf. Comput.*, 207(6):699–725, 2009.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process algebra*. Cambridge University Press Cambridge, 1990.

[CC03] Luís Caires and Luca Cardelli. A spatial logic for concurrency (part i). *Inf. Comput.*, 186(2):194–235, 2003.

[CGH] G. Clark, S. Gilmore, and J. Hillston. Specifying performance measures for PEPA. *Formal Methods for Real-Time and Probabilistic Systems*, pages 211–227.

[CHJ08] K. Chatterjee, T. Henzinger, and B. Jobstmann. Environment Assumptions for Synthesis. In *International Conference on Concurrency Theory (CONCUR)*, pages 147–161, 2008.

[CHJS10] K. Chatterjee, T. A. Henzinger, B. Jobstmann, and R. Singh. Measuring and synthesizing systems in probabilistic environments. In *CAV*, pages 380–395, 2010.

[CKM10] Vincenzo Ciancia, Alexander Kurz, and Ugo Montanari. Families of symmetries as efficient models of resource binding. In Bart Jacobs, Milad Niqui, Jan J. M. M. Rutten, and Alexandra Silva, editors, *CMCS 2010*, volume 264(2) of *ENTCS*, pages 63–81. Elsevier, 2010. Journal version submitted.

[CM10a] Luca Cardelli and Radu Mardare. The measurable space of stochastic processes. In *QEST*, pages 171–180, 2010.

[CM10b] Luca Cardelli and Radu Mardare. Stochastic pi-calculus revisited. Unpublished, 2010.

[CRR02] Sagar Chaki, Sriram K. Rajamani, and Jakob Rehof. Types as models: model checking message-passing programs. In *POPL*, pages 45–57, 2002.

[dA97] L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997.

[DFM11] Pierpaolo Degano, Gian Luigi Ferrari, and Gianluca Mezzetti. On quantitative security policies. In *PaCT*, pages 23–39, 2011.

[DFPV00] Rocco De Nicola, Gian Luigi Ferrari, Rosario Pugliese, and Betti Venneri. Types for access control. *Theor. Comput. Sci.*, 240(1):215–254, 2000.

[DRK02] Dino Distefano, Arend Rensink, and Joost-Pieter Katoen. Model checking birth and death. In Ricardo A. Baeza-Yates, Ugo Montanari, and Nicola Santoro, editors, *IFIP TCS*, volume 223 of *IFIP Conference Proceedings*, pages 435–447. Kluwer, 2002.

[ENLT] Computer Engineering and Switzerland Networks Laboratory (TIK), ETH Zurich. Modular Performance Analysis with Real-Time Calculus. http://www.mpa.ethz.ch/Rtctoolbox/Overview/.

[FMP04] Gian Luigi Ferrari, Eugenio Moggi, and Rosario Pugliese. MetaKlaim: a type safe multi-stage language for global computing. *Mathematical Structures in Computer Science*, 14(3):367–395, 2004.

[FMS02] Marcelo P. Fiore, Eugenio Moggi, and Davide Sangiorgi. A Fully Abstract Model for the π-calculus. *Inf. Comput.*, 179(1):76–117, 2002.

[FZ94] W. Fokkink and H. Zantema. Basic process algebra with iteration: Completeness of its equational axioms. *The Computer Journal*, 37(4):259, 1994.

[GLLV10] Fabio Gadducci, Alberto Lluch-Lafuente, and Andrea Vandin. Counterpart semantics for a second-order μ-calculus. In Hartmut Ehrig, Arend Rensink, Grzegorz Rozenberg, and Andy Schuerr, editors, *ICGT 2010*, volume 6372 of *LNCS*, pages 282–297. Springer, 2010. journal version conditionally accepted in Fundamenta Informaticae.

[GMM06] Fabio Gadducci, Marino Miculan, and Ugo Montanari. About permutation algebras, (pre)sheaves and named sets. *Higher-Order and Symbolic Computation*, 19(2-3):283–304, 2006.

[GP09] Daniele Gorla and Rosario Pugliese. Dynamic management of capabilities in a network aware coordination language. *Journal of Logic and Algebraic Programming*, 2009.

[GS05] Gregor Gößler and Joseph Sifakis. Composition for component-based modeling. *Sci. Comput. Program.*, 55(1-3):161–183, 2005.

[Hav98a] B. R. Haverkort. *Performance of Computer Communication Systems: A Model-Based Approach*. John Wiley & Sons, Inc., New York, NY, USA, 1998.

[Hav98b] Boudewijn R. Haverkort. *Performance of Computer Communication Systems: A Model-Based Approach*. Wiley, 1998.

[Hil96] J. Hillston. *A compositional approach to performance modelling*. Cambridge Univ Pr, 1996.

[HKNP] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. http://www.prismmodelchecker.org/.

[HMRT01] Boudewijn R. Haverkort, Raymond Marie, Gerardo Rubino, and Kishor S. Trivedi, editors. *Performability Modelling : Techniques and Tools*. Wiley, 2001.

[IK01] Atsushi Igarashi and Naoki Kobayashi. A generic type system for the pi-calculus. In *POPL*, pages 128–141, 2001.

[Inc05] Aeronautical Radio Inc. *ARINC 664, Aircraft Data Network, Part 7: Avionics Full Duplex Switched Ethernet (AFDX) Network*. 2005.

[KNP02] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. *Computer Performance Evaluation: Modelling Techniques and Tools*, 2002.

[KNP07] M. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. *Formal Methods for Performance Evaluation*, pages 220–270, 2007.

[KNP09] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic model checking for performance and reliability analysis. *ACM SIGMETRICS Perform. Evaluation Review*, 36(4):40–45, 2009.

[Kob02] Naoki Kobayashi. A type system for lock-free processes. *Inf. Comput.*, 177(2):122–159, 2002.

[LMV11] Alberto Lluch Lafuente, José Meseguer, and Andrea Vandin. State space c-reductions of concurrent systems in rewriting logic. Conference version submitted, 2011.

[LV11] Alberto Lluch Lafuente and Andrea Vandin. Towards a Maude tool for model checking temporal graph properties. In Fabio Gadducci and Leonardo Mariani, editors, *GT-VMT 2011*, ECEASST. EAASST, 2011. to appear.

[Mil91] Robin Milner. The polyadic π-calculus: a tutorial. Technical Report ECS-LFCS-91-180, LFCS, 1991.

[MP95] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, 1995.

[MP05] Ugo Montanari and Marco Pistore. Structured coalgebras and minimal HD-automata for the π-calculus. *Theoretical Computer Science*, 340(3):539–576, 2005.

[MS11] Ugo Montanari and Matteo Sammartino. Network conscious π-calculus. Conference version submitted, 2011.

[NFLP11] Rocco De Nicola, Gianluigi Ferrari, Michele Loreti, and Rosario Pugliese. SCEL – A Language for Context Aware Programming, 2011. To Appear in Proc. of Software Technologies Concertation on Formal Methods for Components and Objects (FMCO).

[NKS07] Mogens Nielsen, Karl Krukow, and Vladimiro Sassone. A Bayesian Model for Event-based Trust. *Electr. Notes Theor. Comput. Sci.*, 172:499–521, 2007.

[Pan09] P. Panangaden. *Labelled Markov Processes*. Imperial College Press, 2009.

[Pnu77] A. Pnueli. The temporal logic of programs. In *IEEE Symposium on Foundations of Computer Science*, pages 46–57, Providence, RI, 1977.

[Pri95] C. Priami. Stochastic π-calculus. *The Computer Journal*, 38(7):578, 1995.

[Ren06] Arend Rensink. Isomorphism checking in GROOVE. In *GRABAT 2006*, volume 1 of *ECEASST*. EASST, 2006.

[vEJ11] C. von Essen and B. Jobstmann. Synthesizing systems with optimal average-case behavior for ratio objectives. In *Workshop on Interactions, Games and Protocols*. Electronic Proceedings in Theoretical Computer Science, 2011. http://arxiv.org/abs/1102.4118v1.

[vEJ12] C. von Essen and B. Jobstmann. Synthesizing efficient controllers. In *13th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI)*, Lecture Notes in Computer Science. Springer, 2012. To appear.

[Wal45] A. Wald. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16(2):117–186, 1945.

[WBB$^+$10] R. Wimmer, B. Braitling, B. Becker, E. M. Hahn, P. Crouzen, H. Hermanns, A. Dhama, and O. Theel. Symblicit calculation of long-run averages for concurrent probabilistic systems. In *QEST*, 2010.

[YBK10] H. Yue, H. C. Bohnenkamp, and J.-P. Katoen. Analyzing energy consumption in a gossiping mac protocol. In *MMB&DFT 2010*, pages 107–119, 2010.

[You05] H. L. S. Younes. *Verification and Planning for Stochastic Processes with Asynchronous Events*. PhD thesis, Carnegie Mellon, 2005.