Ten Years of Performance Evaluation for Concurrent Systems Using CADP

Nicolas Coste[1], Hubert Garavel[2], Holger Hermanns[2,3], Frédéric Lang[2], Radu Mateescu[2], and Wendelin Serwe[2]

[1] STMicroelectronics, 12, rue Jules Horowitz, BP 217, 38019 Grenoble, France
[2] Inria Grenoble – Rhône-Alpes, Inovallée, 655, av. de l'Europe, Montbonnot, 38334 Saint Ismier, France
{Hubert.Garavel,Frederic.Lang,Radu.Mateescu,Wendelin.Serwe}@inria.fr
[3] Saarland University, Dept. of Computer Science, 66123 Saarbrücken, Germany
hermanns@cs.uni-saarland.de

Abstract. This article comprehensively surveys the work accomplished during the past decade on an approach to analyze concurrent systems qualitatively and quantitatively, by combining functional verification and performance evaluation. This approach lays its foundations on semantic models, such as Imc (*Interactive Markov Chain*) and Ipc (*Interactive Probabilistic Chain*), at the crossroads of concurrency theory and mathematical statistics. To support the approach, a number of software tools have been devised and integrated within the Cadp (*Construction and Analysis of Distributed Processes*) toolbox. These tools provide various functionalities, ranging from state space generation (Cæsar and Exp.Open), state space minimization (Bcg_Min and Determinator), numerical analysis (Bcg_Steady and Bcg_Transient), to simulation (Cunctator). Several applications of increasing complexity have been successfully handled using these tools, namely the Hubble telescope lifetime prediction, performance comparison of mutual exclusion protocols, the Scsi-2 bus arbitration protocol, the Send/Receive and Barrier primitives of Mpi (*Message Passing Interface*) implemented on a cache-coherent multiprocessor architecture, and the xStream multiprocessor data-flow architecture for embedded multimedia streaming applications.

1 Introduction

The design of models suited for performance and reliability analysis is challenging due to complexity and size of the modeled systems, in particular for those with a high degree of irregularity. Traditional performance models like Markov chains and queueing networks are not easy to apply for large-sized systems, mainly because they lack hierarchical composition and abstraction means.

Therefore, various specification formalisms have been proposed, which enable systems to be modeled in a compositional, hierarchical manner. A prominent example of such specification formalisms is the class of process algebras, which provide abstraction mechanisms to treat system components as black boxes, making their internal implementation details invisible. Among the many process

T. Margaria and B. Steffen (Eds.): ISoLA 2010, Part II, LNCS 6416, pp. 128–142, 2010.

algebras proposed in the literature, LOTOS [1] has received much attention, due to its technical merits and its status of ISO/IEC International Standard. CADP (*Construction and Analysis of Distributed Processes*) is a widespread tool set for the design and verification of complex systems. CADP supports, among others, the process algebra LOTOS for specification, and offers various tools for simulation and formal verification, including equivalence checkers (bisimulations) and model checkers (temporal logics and modal μ-calculus). About a decade ago, CADP has been extended with performance evaluation capabilities, based on the IMC (*Interactive Markov Chain*) theory [2,3]. More recently, the IMC theory has been transposed into a discrete-time setting, leading to the IPC (*Interactive Probabilistic Chain*) theory [4]. These theories combine well with the approach behind CADP by integrating both, on the one hand, Markov chains and, on the other hand, classical process algebra and the underlying standard notion of LTS (*Labeled Transition System*). Over the years, the performance evaluation branch of CADP has gained maturity, new tools have been added, and many applications have been carried out with the toolbox. This paper provides a survey of the principal modeling and analysis ingredients, and applications of performance evaluation with CADP.

2 The Interactive Markov Chain Model

An IMC (*Interactive Markov Chain*) [3] is a state-transition graph with a denumerable state space, action-labeled transitions, as well as stochastic transitions (also called Markovian transitions). The latter are labeled with rates of exponential distributions. Actions are ranged over by a and b; the particular action τ models internal, i.e., unobservable activity, whereas all other actions model observable activities.

Definition 1 (Interactive Markov Chain). *An* IMC *is a tuple* $\mathcal{I} = (S, \mathcal{A}, \longrightarrow, \Longrightarrow, s_0)$ *where:*

- S *is a nonempty set of states with* initial state $s_0 \in S$,
- $\mathcal{A}$ *is a set of actions,*
- $\longrightarrow \subseteq S \times \mathcal{A} \times S$ *is a set of* interactive *transitions, and*
- $\Longrightarrow \subseteq S \times \mathbb{R}_{>0} \times S$ *is a set of* stochastic *transitions.*

An IMC is a natural extension of a standard LTS (*Labeled Transition System*), as well as of a CTMC (*Continuous-Time Markov Chain*): a standard LTS is an IMC with $\Longrightarrow = \emptyset$, while a CTMC is an IMC with $\longrightarrow = \emptyset$.

Behavioral interpretation. Roughly speaking, the interpretation of a stochastic transition $s \overset{\lambda}{\Longrightarrow} s'$ is that the IMC can switch from state s to s' within d time units with probability $1 - e^{-\lambda \cdot d}$. The positive real value λ thus uniquely identifies a negative exponential distribution. For state s, let $\mathbf{R}(s, s') = \sum\{\lambda \mid s \overset{\lambda}{\Longrightarrow} s'\}$ be the *rate* to move from s to state s'. If $\mathbf{R}(s, s') > 0$ for more than one state s', a competition between the transitions of s exists, known as the *race condition.*

The probability to move from such state s to a particular state s' within d time units, i.e., the stochastic transition $s \Longrightarrow s'$ wins the race, is given by:

$$\frac{\mathbf{R}(s,s')}{E(s)} \cdot \left(1 - e^{-E(s)\cdot d}\right),$$

where $E(s) = \sum_{s' \in S} \mathbf{R}(s,s')$ denotes the *exit rate* of state s. Intuitively, it states that after a delay of at most d time units (second term), the IMC moves probabilistically to a direct successor state s' with discrete branching probability $\mathbf{P}(s,s') = \mathbf{R}(s,s')/E(s)$.

An *internal* interactive transition is a τ-labeled interactive transition, also called τ-transition for short in the sequel, which plays a special role in an IMC. As a τ-transition is not subject to any interaction, it cannot be delayed. Thus, τ-transitions can be assumed to take place immediately. Now consider a state s with both a τ-transition and a stochastic transition. At the precise instant, when the IMC moves to s, the τ-transition can be taken immediately, but the probability that the stochastic transition executes immediately is zero. This justifies that τ-transitions take precedence over stochastic transitions, a property called the *maximal progress assumption*.

Definition 2 (IMC parallel composition). *Let* $\mathcal{I}_1 = (S_1, \mathcal{A}_1, \longrightarrow_1, \Longrightarrow_1, s_{0,1})$ *and* $\mathcal{I}_2 = (S_2, \mathcal{A}_2, \longrightarrow_2, \Longrightarrow_2, s_{0,2})$ *be* IMC*s. For a set of actions* A *such that* $\tau \notin A$*, the* parallel composition *of* $\mathcal{I}_1$ *and* $\mathcal{I}_2$ *wrt.* A *is defined by:*

$$\mathcal{I}_1 \,||_A\, \mathcal{I}_2 = (S_1 \times S_2, \mathcal{A}_1 \cup \mathcal{A}_2, \longrightarrow, \Longrightarrow, (s_{0,1}, s_{0,2}))$$

where $\longrightarrow$ *and* $\Longrightarrow$ *are defined as the smallest relations satisfying:*

1. $s_1 \xrightarrow{a}_1 s_1'$ *and* $s_2 \xrightarrow{a}_2 s_2'$ *and* $a \in A$ *implies* $(s_1, s_2) \xrightarrow{a} (s_1', s_2')$
2. $s_1 \xrightarrow{a}_1 s_1'$ *and* $a \notin A$ *implies* $(s_1, s_2) \xrightarrow{a} (s_1', s_2)$ *for any* $s_2 \in S_2$
3. $s_2 \xrightarrow{a}_2 s_2'$ *and* $a \notin A$ *implies* $(s_1, s_2) \xrightarrow{a} (s_1, s_2')$ *for any* $s_1 \in S_1$
4. $s_1 \overset{\lambda}{\Longrightarrow}_1 s_1'$ *implies* $(s_1, s_2) \overset{\lambda}{\Longrightarrow} (s_1', s_2)$ *for any* $s_2 \in S_2$
5. $s_2 \overset{\lambda}{\Longrightarrow}_2 s_2'$ *implies* $(s_1, s_2) \overset{\lambda}{\Longrightarrow} (s_1, s_2')$ *for any* $s_1 \in S_1$.

The first three constraints define a LOTOS-like parallel composition [1]: actions in A need to be performed by both IMCs simultaneously (first constraint), whereas actions not in A are performed autonomously (second and third constraint). According to the last two constraints, an IMC can delay independently. This differs from timed models such as timed automata, in which individual processes typically need to synchronize on the advance of time. Independent delaying is justified, because whenever two stochastic transitions with rates λ and μ are competing to be executed, then the remaining delay of the μ-transition after the λ-transition has been taken is exponentially distributed with rate μ, due to the memoryless property of exponential distributions.

To compare IMCs, notions of strong and branching bisimulation are used, which extend the notions defined on standard LTS in a conservative fashion, and also extend Markov chain lumpability [3]. Both relations are congruences for

parallel composition and other process algebraic operators. Therefore an IMC in a parallel composition can be replaced by an equivalent, but possibly smaller one, while preserving semantics.

Constraint-oriented specification of performance aspects. To evaluate the performance of a system using the IMC approach, one can insert delays — i.e., a probability distribution approximated arbitrarily closely by a terminating CTMC — into the standard LTS of the system. This insertion can be achieved following the constraint-oriented specification style [5], originally developed to support the early phases of system design. Put in a nutshell, constraints are viewed as separate processes (in our case, IMCs), and parallel composition is used to combine these constraints much in the same vein as logical conjunction.

To illustrate the constraint-oriented specification style applied to an IMC, consider an IMC $\mathcal{I}$ and let a and b be two successive actions in $\mathcal{I}$. To insert a delay approximated by a terminating CTMC Δ with a single final state, construct an IMC $\mathcal{I}_\Delta$ whose initial state contains a single outgoing transition labeled a to the initial state of Δ, and whose final state can only be reached from the final state of Δ by a transition labeled b. The resulting system is then obtained as $\mathcal{I} \,||_{\{a,b\}}\, \mathcal{I}_\Delta$.

3 The Interactive Probabilistic Chain Model

An IPC (*Interactive Probabilistic Chain*) [4] can be seen as a transposition of the IMC approach to a discrete time setting. Thus, in this section, we focus on the differences between the two models. An IPC is essentially a state-transition graph with a denumerable state space, action labeled transitions, and probabilistic transitions. As for an IMC, actions are ranged over by a and b, and the internal action is denoted τ.

Definition 3 (Interactive Probabilistic Chain). *An* IPC *is a tuple* $\mathcal{D} = (S, \mathcal{A}, \rightarrow, \rightsquigarrow, s_0)$ *where:*

- S *is a nonempty set of states with* initial state $s_0 \in S$,
- $\mathcal{A}$ *is a set of actions,*
- $\rightarrow \subseteq S \times \mathcal{A} \times S$ *is a set of* interactive *transitions, and*
- $\rightsquigarrow \subseteq S \times \,]0..1] \times S$ *is a set of* probabilistic *transitions, satisfying for each state* $s \in S$ *that the sum of probabilities of outgoing probabilistic transitions is equal to one, i.e.,* $(\forall s \in S)\ \sum_{s' \in S} \sum \{p \mid s \overset{p}{\rightsquigarrow} s'\} = 1$

Notice that the constraint on probabilistic transitions implies that each state has at least one outgoing probabilistic transition, which may be a self-loop with probability one, i.e., a transition of the form $s \overset{1}{\rightsquigarrow} s$.

An IPC is a natural extension of a standard LTS as well as of a DTMC (*Discrete-Time Markov Chain*): a standard LTS is an IPC with $\rightsquigarrow = \emptyset$, while a DTMC is an IPC with $\rightarrow = \emptyset$.

Behavioral interpretation. Executing a probabilistic transition takes exactly one time step. Any choice between probabilistic transitions is solved according

to the probability distribution. Self-loops with probability one express the *arbitrary waiting* property [6]: an IPC may be blocked waiting for a synchronization that is arbitrarily long (even infinitely) while still letting time advance. As for an IMC, internal interactive transitions take precedence over probabilistic transitions following the maximal progress assumption.

The parallel composition of the IPC model differs from the composition of the IMC model, because the memoryless property of exponential distributions does not apply to the IPC model.

Definition 4 (IPC parallel composition). *Let* $\mathcal{D}_1 = (S_1, \mathcal{A}_1, \longrightarrow_1, \rightsquigarrow_1, s_{0,1})$ *and* $\mathcal{D}_2 = (S_2, \mathcal{A}_2, \longrightarrow_2, \rightsquigarrow_2, s_{0,2})$ *be* IPC*s. For a set of actions* A *such that* $\tau \notin A$*, the* parallel composition *of* $\mathcal{D}_1$ *and* $\mathcal{D}_2$ *wrt.* A *is defined by:*

$$\mathcal{D}_1 \,||_A\, \mathcal{D}_2 = (S_1 \times S_2, \mathcal{A}_1 \cup \mathcal{A}_2, \longrightarrow, \rightsquigarrow, (s_{0,1}, s_{0,2}))$$

where $\longrightarrow$ *and* $\rightsquigarrow$ *are defined as the smallest relations satisfying:*

1. $s_1 \xrightarrow{a}_1 s_1'$ *and* $s_2 \xrightarrow{a}_2 s_2'$ *and* $a \in A$ *implies* $(s_1, s_2) \xrightarrow{a} (s_1', s_2')$
2. $s_1 \xrightarrow{a}_1 s_1'$ *and* $a \notin A$ *implies* $(s_1, s_2) \xrightarrow{a} (s_1', s_2)$ *for any* $s_2 \in S_2$
3. $s_2 \xrightarrow{a}_2 s_2'$ *and* $a \notin A$ *implies* $(s_1, s_2) \xrightarrow{a} (s_1, s_2')$ *for any* $s_1 \in S_1$
4. $s_1 \overset{p_1}{\rightsquigarrow}_1 s_1'$ *and* $s_2 \overset{p_2}{\rightsquigarrow}_2 s_2'$ *implies* $(s_1, s_2) \overset{p_1 p_2}{\rightsquigarrow} (s_1', s_2')$.

Again, the first three constraints define a LOTOS-like parallel composition [1]: actions in A need to be performed by both IPCs simultaneously (first constraint), whereas actions not in A are performed autonomously (second and third constraint). Contrary to the IMC model, the last constraint forces IPCs to synchronize on the advance of time, similar to discrete-timed models. Notice that the probability of the synchronized transition is precisely the product of the probabilities to take each transition separately.

Strong and branching bisimulations for the IPC model are defined similar to the corresponding bisimulations for the IMC model. As for the IMC model, strong and branching probabilistic bisimulations are congruences wrt. parallel composition and other operators.

4 CADP Tools for Extended Markovian Models

A key benefit of the IMC/IPC approach is its compatibility with most existing process calculi, without requiring syntactic and semantic extensions to handle stochastic/probabilistic features. Consequently, it is possible to reuse or extend tools already available in the CADP toolbox, rather than developing a whole set of new tools.

In this section, we present the various CADP tools handling extended Markovian models, i.e., state-transition models that combine features from standard LTS and discrete-time and continuous-time Markov chains. A number of these tools (namely BCG_STEADY, BCG_TRANSIENT, and DETERMINATOR [7]) have been developed specifically to support performance evaluation. Other tools already existed but were already compatible or have been extended to be compatible with the proposed approach. Contrary to most CADP tools that operate

on standard LTSs, these tools operate on extended Markovian models, encoded as probabilistic/stochastic extensions of LTSs. Precisely, an extended Markovian model is an LTS, where all transition labels must be one of the following, a representing either an observable action or the internal action τ:

- a rate "λ", called a *stochastic transition*, or
- a pair "$a; \lambda$" of an action and a rate, called a *labeled stochastic transition*, or
- a probability "p" with $p \in [0, 1]$, called a *probabilistic transition*, or
- a pair "$a; p$" of an action and a probability, called a *labeled probabilistic transition*, or
- an action "a", called an *interactive transition* (also called *ordinary transition* in the CADP documentation).

Note that extended Markovian models are sufficiently general to include IMC, IPC, and various other probabilistic[1] and stochastic models[2]. In CADP, extended Markovian models are represented explicitly in the BCG format, or implicitly using the OPEN/CÆSAR environment [8].

4.1 State Space Generation Using CÆSAR.ADT and CÆSAR

CÆSAR.ADT [9] and CÆSAR [10,11] are two complementary LOTOS to C compilers, the former for the data part, the latter for the behavior part of LOTOS. The C code generated by these compilers is then used by other CADP tools for various purposes: simulation, random execution, on-the-fly verification, test generation, etc. Additionally, CÆSAR can generate the LTS corresponding to a LOTOS specification, if of finite size. This LTS is encoded in the BCG format and can be verified using bisimulations and/or model checking of μ-calculus or temporal logic formulas.

A LOTOS specification, whose functional correctness has been already verified, can be enriched with stochastic information as follows: the user must insert in the LOTOS specification, at each place where a Markov delay or a probabilistic transition should occur, a new LOTOS gate λ_i.

After all gates λ_i have been inserted in the LOTOS specification, CÆSAR and CÆSAR.ADT are invoked as usual to generate the corresponding LTS. This LTS is then turned into an extended Markovian model (still encoded in the BCG format) by replacing each transition labeled with λ_i by a stochastic or probabilistic transition of known numerical value. This can be achieved using the BCG_LABELS tool of CADP, which performs hiding and/or renaming on the labels attached to the transitions of a BCG file, according to a set of regular expression and substitution patterns specified by the user.

[1] Discrete Time Markov Chains, Discrete Time Markov Reward Models, Alternating Probabilistic LTS, Discrete Time Markov Decision Processes, Generative Probabilistic LTS, Reactive Probabilistic LTS, Stratified probabilistic LTS.

[2] Continuous Time Markov Chains, Continuous Time Markov Reward Models, Continuous Time Markov Decision Processes, Timed Processes for Performance Models, Performance Evaluation Process Algebra Models, Extended Markovian Process Algebra Models.

4.2 Compositional Verification Using EXP.OPEN

Exp.Open [12] is a compositional verification tool for on-the-fly exploration of a graph corresponding to a *network of communicating automata* (represented as a set of Bcg files). These automata are composed together in parallel using either algebraic operators (as in the Ccs, Csp, Lotos, and μCrl process algebras), "*graphical*" operators (as in E-Lotos and Lotos NT), or synchronization vectors (as in the Mec and Fc2 tools). Additional operators are available to hide and/or rename labels (using regular expressions), to cut certain transitions, and to give priority of certain transitions over others.

To address state explosion, Exp.Open is equipped with partial order reduction techniques that preserve either deadlocks, weak traces, or branching bisimulation. For an extended Markovian model, Exp.Open uses the maximal progress property to cut all stochastic transitions in choice with τ-transitions (see Section 2).

Branching bisimulation (and both its stochastic and probabilistic variants) as well as trace equivalence, weak trace equivalence, safety equivalence, observational equivalence, and strong bisimulation are congruences for all Exp.Open operators except priorities. This is a key property for compositional verification, which extends the congruence property mentioned in Section 2 to the more general network of communicating automata model.

4.3 Bisimulation Reduction Using BCG_MIN

The Bcg_Min tool enables graph minimization modulo strong bisimulation or branching bisimulation, extended to the probabilistic and stochastic cases. Thus, it can be used for functional verification and performance evaluation. Bcg_Min accepts as input three kinds of extended Markovian models, all encoded in the Bcg graph format:

- either a *standard* Lts, containing only interactive transitions,
- or a *probabilistic* model, containing only interactive, probabilistic, or labeled probabilistic transitions (e.g., an Ipc),
- or a *stochastic* model, containing only interactive, stochastic, or labeled stochastic transitions (e.g., an Imc).

A new version 2.0 of Bcg_Min implementing a *signature-based partition refinement* algorithm [13] generalized to stochastic and probabilistic bisimulations has been released in 2010. This new version brings spectacular performance improvements with respect to the previous version. In particular, the reduction modulo stochastic and probabilistic bisimulations of a test base consisting of 1335 probabilistic and stochastic models was 540 times faster, and up to 8500 times faster for one particular model.

4.4 Nondeterminism Elimination Using DETERMINATOR

The Determinator tool [7] eliminates stochastic nondeterminism in extended Markovian models on the fly. It takes as input an extended Markovian model $\mathcal{M}$

(encoded in the BCG graph format) containing probabilistic and/or stochastic transitions and attempts at translating $\mathcal{M}$ to a CTMC (*Continuous Time Markov Chain*), i.e., an LTS (encoded in the BCG graph format) that contains (labeled) stochastic transitions only. The aim is therefore to eliminate the interactive and probabilistic transitions, while keeping the stochastic information present in $\mathcal{M}$.

Because the translation is impossible in the general case, DETERMINATOR only handles models verifying a sufficient condition (*well-specified check*) [14], which guarantees that the resulting CTMC is unique, independently of the way nondeterministic choices are resolved. This enables to eliminate the nondeterminism without modifying the stochastic behavior of the system.

The translation algorithm implemented in DETERMINATOR is a variant of the one presented in [14]. It works on the fly using the functionalities of the OPEN/CÆSAR environment [8].

Although the BCG_MIN tool also enables, in some way, to eliminate nondeterminism (by doing more general reductions based on the concept of *lumpability*), it differs from DETERMINATOR: BCG_MIN does not handle the case of LTS containing both probabilistic and stochastic transitions and it does not eliminate interactive transitions.

4.5 Numerical Analysis Using BCG_STEADY and BCG_TRANSIENT

The BCG_STEADY and BCG_TRANSIENT tools take as input a CTMC (*Continuous Time Markov Chain*), a DTMC (*Discrete Time Markov Chain*), or even any extended Markovian model without interactive transitions meeting the restrictions detailed on the respective manual pages[3]. The input model is represented internally as a (sparse) matrix indexed by states that is used by numerical algorithms for performance evaluation:

- BCG_STEADY computes, for each state s, the probability to be in s on the long run, i.e., in the equilibrium or "steady state". These probabilities are computed iteratively using a Gauss-Seidel algorithm [15].
- BCG_TRANSIENT computes, for each state s and for each time instant t in a discrete set provided by the user, the probability to be in s at instant t. This computation uses the uniformization algorithm [16,15] and the Fox-Glynn [17] method to approximate Poisson probabilities.

Based on the computed probabilities, BCG_STEADY and BCG_TRANSIENT can also compute the corresponding transition throughputs, i.e., the average number of transition executions per time unit. These measures can provide important high-level information to assess the system performance, reliability or productivity, such as operation latencies (see Section 6).

BCG_STEADY and BCG_TRANSIENT generate output in the standard CSV (*Comma Separated Values*) format used by mainstream data processing tools, including EXCEL and GNUPLOT.

[3] Available at `http://vasy.inria.fr/cadp/man/bcg_steady.html` and `http://vasy.inria.fr/cadp/man/bcg_transient.html`

The BCG_MIN (see Section 4.3) and DETERMINATOR (see Section 4.4) tools might be required to determinize an extended Markovian model to be given as input to BCG_STEADY or BCG_TRANSIENT.

4.6 On-the-Fly Steady-State Simulation Using CUNCTATOR

CUNCTATOR is an on-the-fly steady-state simulator for stochastic models. It takes an extended Markovian model (represented using the OPEN/CÆSAR environment) as input, applies any user-defined hiding and renaming operations, and explores a random execution sequence on the fly. Exploration is aborted whenever an observable interactive transition or a probabilistic transition is encountered. During the exploration, the tool sums up the virtual time elapsed in the states, determined according to the rates of their outgoing stochastic transitions. The simulation terminates when either the virtual time, or the length of the simulation sequence (number of transitions) reaches a maximum value specified by the user. Upon termination, the throughputs of the labeled stochastic transitions of interest are displayed, together with additional information (number of τ-transitions encountered, presence of nondeterminism, etc.). The context reached at the end of a simulation can be saved in order to restart subsequent simulations from this context. This mechanism is useful for implementing convergence criteria (e.g., based on confidence intervals) by allowing to perform in linear time a series of increasingly long simulations, each one being the prefix of the subsequent ones.

When a nondeterministic state (with at least two outgoing τ-transitions) is reached, CUNCTATOR explores one of the outgoing τ-transitions. The choice of this transition can be made currently according to three scheduling policies: the first τ-transition encountered, the last one, or a randomly chosen one. When a simulation has encountered nondeterministic states, the user has the possibility of launching other simulations using different scheduling policies in order to obtain more insight about the stochastic behavior of the model.

CUNCTATOR stores in memory only the last state of the simulation sequence, thus consuming only a small amount of memory, independent from the length of the simulation sequence and from the size of the CTMC. Compared to BCG_STEADY, which computes exact throughputs in a CTMC represented as a BCG file, CUNCTATOR consumes less memory but may require a longer execution time in order to achieve the same accuracy.

5 Additional Tools for Interactive Probabilistic Chains

To support the IPC model, we took advantage of the open architecture of CADP and prototyped additional tools (4,900 lines of C code) that are not yet integrated into CADP. Together with CADP, these tools support the compositional construction of an IPC (following a constraint oriented style) and the computation of latency distributions.

Parallel composition of IPCs. Because the parallel composition of LOTOS and those supported by EXP.OPEN are incompatible with the parallel composition of the IPC model, a different parallel composition is required.

The IPC_COMPOSE tool takes as input a network of communicating IPCs (represented as a set of BCG files, composed in parallel using LOTOS parallel compositions) and produces the corresponding IPC.

Constraint oriented delay insertion. Ideally, one would like to use IPC_COMPOSE to insert delays into an LTS, following an approach similar to the constraint-oriented style supported by CADP for the IMC model. Unfortunately, this approach may lead to non-determinism in the case a choice between two actions depends whether a delay has elapsed or not, because in the IPC model, delays in two concurrent processes may expire at exactly the same time instant. Notice that this cannot happen in an IMC model, because the probability for two exponential distributions to expire at the same time is zero.

Thus, we developed the IPC_INSERT tool, which takes as input an LTS (represented as a BCG file) and a probabilistic distribution for a single delay (also represented as a BCG file) and produces the corresponding IPC.

In practice, the IPC of a complete system is obtained by first generating the LTS of each sequential interacting subcomponent, then inserting delays into these components (using IPC_INSERT), and finally computing their parallel composition (using IPC_COMPOSE).

Computation of latency distributions. The IPC_DISTRIBUTION tool takes as input a deterministic IPC and two actions a and b and computes the long-run average probability distribution of the latency between a and b [4].

6 Applications

In this Section, we report about five case studies that have been tackled using the proposed methodology and its associated tools.

6.1 The Hubble Telescope Lifetime

The first case study with the performance evaluation tools provided by CADP was the Hubble space telescope example described in [18].

A 50-line LOTOS specification was developed for this example; it consists of seven concurrent processes: one controller process and one process for each of the six Hubble stabilizing units (i.e., gyroscopes that may fail as time elapses). This LOTOS specification is parameterized by three constants λ, μ, and ν representing the average lifetime of a gyroscope, the time needed to stop all Hubble equipments and the time needed to replace all gyroscopes.

Using the CÆSAR and CÆSAR.ADT compilers, an LTS (877 states, 3341 transitions) was generated; this LTS was then turned into an IMC by replacing by their actual values the λ, μ, and ν parameters present in the transition labels. Then, the BCG_MIN tool was used to minimize this IMC modulo stochastic branching

minimization, leading to a CTMC (9 states, 12 transitions) that is small enough to be verified visually. Finally, the BCG_TRANSIENT tool was used to compute failure probabilities at various time instants, thus giving an estimation of the Hubble telescope lifetime.

6.2 Mutual Exclusion Protocols

Recently, several mutual exclusion protocols for shared memory computers have been analyzed using CADP [19]. These protocols are an essential building block of concurrent systems, required whenever a shared resource has to be protected against concurrent non-atomic accesses.

For a system with two processes communicating through up to seven shared variables, 24 mutual exclusion protocols have been described in LOTOS NT and translated automatically into LOTOS. The LTS of each protocol was transformed into an IMC (from 89 states and 130 transitions up to 31,222 states and 43,196 transitions), which was then reduced using BCG_MIN. Finally, the throughput of the accesses to the shared resource was computed using BCG_STEADY.

Besides comparing the performance of the various protocols, this study gave insight about the performance impact of changing the rates for accessing the shared resource. The results corroborate functional properties, in particular asymmetric behavior, i.e., overtaking of one process by the other.

6.3 The SCSI-2 Bus Arbitration Protocol

Another case study [20] with the performance evaluation tools of CADP was a storage system developed by Bull in the early 90's. This system consisted of a disk controller and (at most) seven disks connected by a SCSI-2 (*Small Computer System Interface*) bus. During the testing phase, Bull engineers discovered potential starvation problems for disks having a smaller SCSI number than the disk controller.

The storage system was formally described in LOTOS, and it was found that the multiway rendezvous of LOTOS was most appropriate to model the SCSI-2 bus arbitration protocol concisely. This LOTOS description was submitted to model checking verification using CADP enabling to reproduce the starvation problem automatically.

Then, the LOTOS description was turned into a performance model by inserting at various places two stochastic delays λ (load stress imposed on the disk controller) and μ (average time for a disk to service a transfer request) and by adding an auxiliary LOTOS process modeling a phase-type distribution (Erlang law with parameter ν) between two SCSI-2 bus arbitration periods.

The corresponding LTS was generated using CÆSAR.ADT and CÆSAR, and then minimized using BCG_MIN modulo branching bisimulation after hiding and/or renaming actions unrelated to performance. Due to the use of compositional state space generation techniques, state space explosion does not occur (the largest automaton produced has 56,169 states and 154,752 transitions only).

Then, the λ, μ, and ν parameters were replaced with a series of numerical constants; for each instantiation, the IMC obtained was minimized using BCG_MIN modulo stochastic branching bisimulation, yielding a CTMC in which nondeterminism had vanished; finally, the BCG_STEADY tool was applied to each such CTMC to compute the equilibrium (steady-state) probabilities for each state, as well as throughputs for relevant actions, enabling a precise study of unfairness in the SCSI-2 system under heavy load.

6.4 The MPI Send/Receive and Barrier Primitives

In the context of the MULTIVAL project together with Bull, we studied an implementation of MPI (*Message Passing Interface*) to be run on FAME2 (*Flexible Architecture for Multiple Environments*), a CC-NUMA multiprocessor architecture developed at Bull for teraflop mainframes and petaflop computing.

In a first study [21] we focused on the *ping-pong* MPI benchmark, with the goal of estimating the latency of *send/receive* primitives on FAME2 machines. Several configurations of the benchmark were specified in LOTOS, by considering three interconnection topologies, two implementations of the *send/receive* primitives SR_1 and SR_2 (based on linked lists with locks and on lock-free buffers) and two cache coherency protocols A and B (in which a variable written by a process becomes either owned by that process, or shared between that process and the previous owner). The performance analysis was carried out by extending the LOTOS specification with exponential distributions and applying the BCG_MIN, DETERMINATOR, and BCG_STEADY tools of CADP to compute the *send/receive* latency. The computed latencies were close (down to 9% of difference) to the experimental measures, even for the relatively simple model considered. This analysis also enabled to estimate the number of cache misses corresponding to each instruction, showing that the most efficient configuration is given by the SR_2 *send/receive* implementation and the cache coherency protocol A.

We applied the same approach to study five protocols implementing the *barrier* primitive of MPI (*centralized*, *combining*, *tournament*, *dissemination*, and *tree-based*). Using EXP.OPEN, the final Markov chain was generated compositionally for the centralized barrier with six processes and the tree-based barrier with four processes, and computed the latencies of barrier traversals using BCG_STEADY. The remaining protocols, which have prohibitively large state spaces, were analyzed by simulation using CUNCTATOR with the confidence interval criterion for convergence (automated using the save/restore mechanism), for configurations containing up to four processes. The throughputs obtained by simulation were close (less than 5%) to those computed by BCG_STEADY.

6.5 The xSTream Data-Flow Architecture

In the context of the MULTIVAL project together with STMicroelectronics, we studied XSTREAM, a multiprocessor data-flow architecture for high performance embedded multimedia streaming applications. In this architecture, computation nodes (e.g., filters) communicate using XSTREAM queues connected by a NOC

(*Network on Chip*). An xSTREAM queue generalizes a bounded FIFO queue in two ways: it provides additional primitives (such as *peek* to consult items in the middle of the queue, which is not possible with the standard *push*/*pop* primitives of FIFO queues), and a *backlog* (extra memory) to allow the increase of the queue size when the queue overflows.

Our performance evaluation study [4] aimed at predicting throughput and latency of communication between xSTREAM queues. For us, a key challenge is to combine probabilistic/stochastic information (e.g., the rates at which xSTREAM applications push and pop elements in and out of the queues) with precise timing information (e.g., memory access time). We studied the performance impact of the flow-control protocol, which ensures that every message emitted into the NOC can be received, i.e., leave the NOC. By enriching a functional LOTOS model with probabilistic delays, we compositionally constructed an IPC of a system of two data streams sharing the NOC (3205 states and 4630 transitions; before hiding and minimization, the IPC contained 539,302 states and 1,412,168 transitions, and the largest intermediate IPC contained 46,940,161 states and 198,490,980 transitions). The performance measures obtained with the prototype tools presented in Section 5 justified the relevance of the flow-control protocol. On the one hand, without the flow-control protocol, increasing the latency of one stream also increases the latency of the other stream, because the slow stream might fill the buffers in the NOC (functional verification even showed the possibility of a deadlock for particular kinds of applications). On the other hand, with the flow-control protocol, increasing the latency of one stream even reduces the latency of the other stream.

7 Conclusion and Future Work

This paper has given a survey of foundations, methodology, tool components, and applications of the CADP approach to compositional performance evaluation. The IPC and the IMC models have very close conceptual roots, and one can view an IPC as a clock-ticked version of an IMC and, vice versa, one can view an IMC as the continuous time limit of an IPC, with clock intervals tending to zero. A recent proposal [22] introduces a model that integrates both worlds in one, and develops the basic compositional theory, along the lines of IMC for this model. It is interesting to see how the available tool support can be extended to this setting.

Recent applications of CADP in large industrial projects are very promising, but are also fostering the development of new and improved analysis support. This opens two challenging directions. First of all, the CUNCTATOR tool opens an analysis avenue, based on discrete-event simulation, that does not suffer from the state space explosion, is straightforward to parallelize, and can support distributions that are not restricted by the Markov property. We are exploring this avenue in relation to discrete-event simulation activities revolving around the MODEST language and tool [23].

The analysis performed with CADP is an instance of the general theme of combining performance evaluation and model checking [24]. An interesting research direction concerns recent advances in model checking a general IMC. So far, IMC analysis with CADP is limited to cases where the branching bisimulation quotient — obtained with BCG_MIN — is free of nondeterminism. With the advances reported in [25] this restriction is — at least in principle — obsolete, but an implementation of this technique inside CADP is still to be done.

References

1. ISO/IEC: LOTOS — a formal description technique based on the temporal ordering of observational behaviour. International Standard 8807, International Organization for Standardization — Information Processing Systems — Open Systems Interconnection, Genève (September 1989)
2. Hermanns, H., Katoen, J.P.: Automated compositional Markov chain generation for a plain-old telephone system. Science of Computer Programming 36(1), 97–127 (2000)
3. Hermanns, H.: Interactive Markov Chains. LNCS, vol. 2428. Springer, Heidelberg (2002)
4. Coste, N., Hermanns, H., Lantreibecq, E., Serwe, W.: Towards performance prediction of compositional models in industrial gals designs. In: Bouajjani, A., Maler, O. (eds.) Computer Aided Verification. LNCS, vol. 5643, pp. 204–218. Springer, Heidelberg (2009)
5. Vissers, C.A., Scollo, G., van Sinderen, M., Brinksma, E.: Specification styles in distributed systems design and verification. Theoretical Computer Science 89(1), 179–206 (1991)
6. Hansson, H.A.: Time and Probability in Formal Design of Distributed Systems. Elsevier Science Inc., Amsterdam (1994)
7. Hermanns, H., Joubert, C.: A set of performance and dependability analysis components for CADP. In: Garavel, H., Hatcliff, J. (eds.) TACAS 2003. LNCS, vol. 2619, pp. 425–430. Springer, Heidelberg (2003)
8. Garavel, H.: Open/Cæsar: An open software architecture for verification, simulation, and testing. In: Steffen, B. (ed.) TACAS 1998. LNCS, vol. 1384, pp. 68–84. Springer, Heidelberg (1998)
9. Garavel, H.: Compilation of LOTOS abstract data types. In: Vuong, S.T. (ed.) Proceedings of the 2nd International Conference on Formal Description Techniques FORTE 1989, Vancouver B.C., Canada, pp. 147–162. North Holland, Amsterdam (December 1989)
10. Garavel, H., Sifakis, J.: Compilation and verification of LOTOS specifications. In: Logrippo, L., Probert, R.L., Ural, H. (eds.) Proceedings of the 10th International Symposium on Protocol Specification, Testing and Verification, Ottawa, Canada. IFIP, pp. 379–394. North Holland, Amsterdam (June 1990)
11. Garavel, H., Serwe, W.: State space reduction for process algebra specifications. Theoretical Computer Science 351(2), 131–145 (2006)
12. Lang, F.: Exp.open 2.0: A flexible tool integrating partial order, compositional, and on-the-fly verification methods. In: Romijn, J.M.T., Smith, G.P., van de Pol, J. (eds.) IFM 2005. LNCS, vol. 3771, pp. 70–88. Springer, Heidelberg (2005)
13. Blom, S., Orzan, S.: Distributed state space minimization. Springer International Journal on Software Tools for Technology Transfer (STTT) 7(3), 280–291 (2005)

14. Deavours, D.D., Sanders, W.H.: An efficient well-specified check. In: Proceedings of the 8th International Workshop on Petri Nets and Performance Models PNPM 1999, Zaragoza, Spain, pp. 124–133. IEEE Press, Los Alamitos (1999)
15. Stewart, W.J.: Introduction to the Numerical Solution of Markov Chains. Princeton University Press, Princeton (1994)
16. Jensen, A.: Markov chains as an aid in the study of markov processes. Skand. Aktuarietidskrift 3, 87–91 (1953)
17. Fox, B.L., Glynn, P.W.: Computing Poisson probabilities. Communications of the ACM 31(4), 440–445 (1987)
18. Hermanns, H.: Construction and verification of performance and reliability models. Bulletin of the EATCS 74, 135–154 (2001)
19. Mateescu, R., Serwe, W.: A study of shared-memory mutual exclusion protocols using CADP. In: Kowalewski, S., Roveri, M. (eds.) FMICS 2010. LNCS, vol. 6371, pp. 180–197. Springer, Heidelberg (2010)
20. Garavel, H., Hermanns, H.: On combining functional verification and performance evaluation using CADP. In: Eriksson, L.-H., Lindsay, P.A. (eds.) FME 2002. LNCS, vol. 2391, pp. 410–429. Springer, Heidelberg (2002)
21. Chehaibar, G., Zidouni, M., Mateescu, R.: Modeling multiprocessor cache protocol impact on mpi performance. In: Proceedings of the 2009 IEEE International Workshop on Quantitative Evaluation of Large-Scale Systems and Technologies QuEST 2009, Bradford, UK. IEEE Computer Society Press, Los Alamitos (2009)
22. Eisentraut, C., Hermanns, H., Zhang, L.: On probabilistic automata in continuous time. In: Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010. IEEE Computer Society, Los Alamitos (2010)
23. Bohnenkamp, H.C., D'Argenio, P.R., Hermanns, H., Katoen, J.P.: MoDeST: A compositional modeling formalism for hard and softly timed systems. IEEE Transactions on Software Engineering 32(10), 812–830 (2006)
24. Baier, C., Haverkort, B., Hermanns, H., Katoen, J.P.: Performance evaluation and model checking join forces. Communications of the ACM 53(9), 76–85 (2010)
25. Zhang, L., Neuhäußer, M.R.: Model checking interactive markov chains. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 53–68. Springer, Heidelberg (2010)