A Framework for Performance Evaluation and Functional Verification in Stochastic Process Algebras

Hossein Hojjat
IPM and University of Tehran,
Tehran, Iran

MohammadReza Mousavi*
TU/Eindhoven, Eindhoven,
The Netherlands, and
Reykjavík University,
Reykjavík, Iceland

Marjan Sirjani
IPM and University of Tehran,
Tehran, Iran

ABSTRACT

Despite its relatively short history, a wealth of formalisms exist for algebraic specification of stochastic systems. The goal of this paper is to give such formalisms a unifying framework for performance evaluation and functional verification. To this end, we propose an approach enabling a provably sound transformation from some existing stochastic process algebras, e.g., PEPA and MTIPP, to a generic form in the mCRL2 language. This way, we resolve the semantic differences among different stochastic process algebras themselves, on one hand, and between stochastic process algebras and classic ones, such as mCRL2, on the other hand. From the generic form, one can generate a state space and perform various functional and performance-related analyses, as we illustrate in this paper.

1. INTRODUCTION

Compositionality is a very much sought feature in the analysis of systems which is inherent to process algebras [2, 15, 16]. Stochastic process algebras (SPAs) [4, 6, 11, 14] bring about this useful feature to the field of performance evaluation. Several SPAs appeared as a result of different design decisions in merging stochastic aspects of processes with their behavioral aspects: some decided to follow the tradition in many timed process algebras and keep the semantics of stochastic rates orthogonal to the behavioral semantics while others decided to merge the two into a single semantic model, i.e., a single transition (multi-)relation. Communication and synchronization are central to the semantics of process algebras and defining the semantics of synchronization for SPAs is yet another point of dispute among them. Furthermore, different SPAs interpret nondeterminism differently.

*The work of M.R. Mousavi has been partially supported by the projects "Unifying Framework for Operational Semantics" (nr. 070030041).

SAC'08 March 16-20, 2008, Fortaleza, Ceará, Brazil

To make our goals concrete, we would like to translate a number of SPAs which have similar semantic frameworks (e.g., PEPA [14], MTIPP [13], EMPA [4] and IMC [11]) to a common form in the mCRL2 process algebra (which is a classic process algebra [2] enhanced with abstract data types). Our next milestone is to generate state space using the mCRL2 tool-set. Finally, we wish to be able to perform various functional- as well as performance-related analyses on the generated state space using available tools, as well as tools that we develop for this purpose.

For the ease of presentation, we shall focus on one of these process algebras, namely PEPA (due to its popularity), in the remainder of this paper and only touch upon the aspects in which our approach has to be adapted to fit the other process algebras listed above. Our prototype implementation currently supports specifications in PEPA and MTIPP and we are currently extending it to support EMPA and IMC. [1]

Related Work. A proposal for unifying performance and functional analysis is put forward in [9] which is closest to ours. There, the authors use the process algebra LOTOS together with stochastic gates (basic actions); the LOTOS specification is then translated, using CADP toolset, into IMC and various existing as well as newly developed analysis techniques are introduced on the generated IMC specification. Our work can benefit from the latter techniques after generating the LTS semantics of stochastic decision processes. As for the former translation (from LOTOS to IMC), our work can be considered complementary to that of [9]. In [12], in order to combine functional and performance analysis, a new operator, called *elapse*, is added to the LOTOS process algebra. We decided to use the syntax of existing stochastic process algebras rather than adding syntactic sugar to mCRL2 for modeling stochastic processes.

Our approach relies essentially on using the equational theory of SPAs in order to resolve their semantic differences with mCRL2. Sound and complete equational theories are already developed for MTIPP [13], EMPA [4] and IMC [11]. We are not aware of an existing complete axiomatization for PEPA; in [14], however, a number of sound axioms of PEPA are introduced. All existing axiomatizations of SPAs make use of an axiom scheme called *Expansion Theorem* which stands for an infinite number of axioms (one for each number of summands as arguments of parallel composition). However, we exploit the well-known technique of using auxiliary operators [3] to give PEPA a finite (sound and complete)

[1]The implementation is available from `http://www.win.tue.nl/~mousavi/spa/`.

axiomatization. In [4, Section 3], the authors define an interleaving semantics for EMPA. The method used in [4] for defining the interleaving semantics of EMPA resembles our approach to resolving the semantics of synchronization and choice in our implementation.

The rest of this paper is organized as follows. Sections 2 and 3 give, respectively, an overview of our source and target domains, i.e., stochastic process algebras (with a focus on PEPA) and mCRL2. Section 4 presents our general approach and gives a detailed account of its application to PEPA by means of a few examples. Section 5 applies our method to a case-study and provides a practical view to the application of our approach. Section 6 concludes the paper and presents directions for ongoing and future research.

2. STOCHASTIC PROCESS ALGEBRAS

2.1 PEPA

2.1.1 Syntax and Semantics

The syntax of PEPA processes is given below.

$$p \quad ::= \quad 0 \mid (\alpha, r).p \mid p + p \mid p \underset{L}{\bowtie} p \mid p/L \mid A$$

In the above syntax, 0 stands for the deadlocking process. [2] $(\alpha, r).p$ stands for an action of type $\alpha \in \mathcal{A}$ whose duration is determined by an exponentially distributed random variable with rate $r \in \mathbb{R}^{>0}$ followed by process p. There are *weighted passive* actions in PEPA which have rates in $\{n\top \mid n \in I\!N \setminus \{0\}\}$ ($1\top$ is denoted by $\top$). It is assumed that $r < n\top$ for each $r \in \mathbb{R}^{>0}$ and $n \in I\!N \setminus \{0\}$. Passive actions are important in practical applications: they represent actions whose rates are unknown or determined by their synchronizing partner. Although we fully treat weighted passive actions in our tool implementation, we do not clutter the presentation in this section by treating them formally. They can be incorporated smoothly without any substantial change in the theory. Nondeterministic choice is represented by $+$ and stands for its general process-algebraic intuition. Parallel composition with mandatory (CSP-style) synchronization among actions in $L \subseteq \mathcal{A}$ is denoted by $p \underset{L}{\bowtie} p$. Expression p/L hides actions in $L \subseteq \mathcal{A}$, i.e., renames them to the unobservable action $\tau \in \mathcal{A}$. A denotes a recursive variable defined by an equation $A := t$.

The semantics of PEPA processes is given by the following SOS-style deduction rules.

$$(\mathbf{pre})\frac{}{(\alpha, r).x \overset{(\alpha,r)}{\rightarrow} x} \qquad (\mathbf{c0})\frac{x_0 \overset{(\alpha,r)}{\rightarrow} y_0}{x_0 + x_1 \overset{(\alpha,r)}{\rightarrow} y_0}$$

$$(\mathbf{c1})\frac{x_1 \overset{(\alpha,r)}{\rightarrow} y_1}{x_0 + x_1 \overset{(\alpha,r)}{\rightarrow} y_1} \qquad (\mathbf{par0})\frac{x_0 \overset{(\alpha,r)}{\rightarrow} y_0}{x_0 \underset{L}{\bowtie} x_1 \overset{(\alpha,r)}{\rightarrow} y_0 \underset{L}{\bowtie} x_1} \ \alpha \notin L$$

$$(\mathbf{par1})\frac{x_1 \overset{(\alpha,r)}{\rightarrow} y_1}{x_0 \underset{L}{\bowtie} x_1 \overset{(\alpha,r)}{\rightarrow} x_0 \underset{L}{\bowtie} y_1} \ \alpha \notin L$$

$$(\mathbf{par2})\frac{x_0 \overset{(\alpha,r_0)}{\rightarrow} y_0 \quad x_1 \overset{(\alpha,r_1)}{\rightarrow} y_1}{x_0 \underset{L}{\bowtie} x_1 \overset{(\alpha,R)}{\rightarrow} y_0 \underset{L}{\bowtie} y_1} \ \alpha \in L$$

$$\text{where } R = \frac{r_0}{r_\alpha(x_0)} \frac{r_1}{r_\alpha(x_1)} min(r_\alpha(x_0), r_\alpha(x_1))$$

$$(\mathbf{hid0})\frac{x \overset{(\alpha,r)}{\rightarrow} y}{x/L \overset{(\alpha,r)}{\rightarrow} y} \ \alpha \notin L \qquad (\mathbf{hid1})\frac{x \overset{(\alpha,r)}{\rightarrow} y}{x/L \overset{(\tau,r)}{\rightarrow} y} \ \alpha \in L$$

$$(\mathbf{rec})\frac{t \overset{(\alpha,r)}{\rightarrow} y}{A \overset{(\alpha,r)}{\rightarrow} y} \ A := t$$

Most deduction rules are standard, i.e., the same as their PA counterparts. The most notable exception is deduction rule (**par2**) which defines synchronization. In this rule, the rate of the synchronizing action is $\frac{r_0}{r_\alpha(x_0)} \frac{r_1}{r_\alpha(x_1)} min(r_\alpha(x_0), r_\alpha(x_1))$, where $r_\alpha(x_i)$, for each $i \in \{0, 1\}$ is the sum of the rates of all enabled α-transitions at the parallel component x_i formally defined below [14].

$$\begin{array}{lcll}
r_\alpha(0) & = & 0 & \\
r_\alpha((\alpha, r).x) & = & r & \\
r_\alpha((\beta, r).x) & = & 0 & \\
r_\alpha(x + y) & = & r_\alpha(x) + r_\alpha(y) & \\
r_\alpha(x \underset{L}{\bowtie} y) & = & r_\alpha(x) + r_\alpha(y) & \text{if } \alpha \notin L \\
r_\alpha(x \underset{L}{\bowtie} y) & = & min(r_\alpha(x), r_\alpha(y)) & \text{if } \alpha \in L \\
r_\alpha(x/L) & = & r_\alpha(x) & \text{if } \alpha \notin L \\
r_\alpha(x/L) & = & 0 & \text{if } \alpha \in L
\end{array}$$

2.1.2 Axiomatization

Equational theories are powerful tools in process algebraic formalisms which allow for sound transformations of processes (w.r.t. a particular notion of equality) without resorting to their semantics and generating their state space. Given a notion of behavior equality, a set of equations is called an *axiomatization* of a model, or is *sound and complete*, when one can prove all sound (and only sound) equalities from them. We present an axiomatization of the recursion-free subset of PEPA w.r.t. Markovian bisimulation [14, 11] in this section. To our knowledge, this is the first finite axiomatization of PEPA (most of the ingredients were already present in [14], though).[3] Similar axiomatizations exist for other SPAs; apart from semantic difference reflected by the axioms, the main subtle difference between our axiomatization and those given before is that previous axiomatizations of SPAs [13, 4, 11] were infinite (i.e., used an expansion theorem) but our axiomatization has the advantage of being finite, thanks to the use of auxiliary left-merge and communication merge operators. We later use this axiomatization

[2] In the original syntax of PEPA, 0 is not included but 0 can be expressed using the enforced synchronization, e.g, $(\alpha, r).P \underset{\alpha,\beta}{\bowtie} (\beta, r').P'$, for each $\alpha \neq \beta$, is a deadlocking process. We included it as a syntactic construct to facilitate axiomatization.

[3] Our axiomatization is also complete for the linear subset of PEPA (with a proof similar to that of the original one for CCS [17] and later adapted for SPAs [11]) but we used the recursion-free subset to simplify our proofs.

to resolve the semantics of synchronization and choice and transform PEPA processes to our *common stochastic form*.

First we give the finite axiomatization for sequential nondeterministic PEPA processes, i.e., processes comprising prefixing and choice.

$$\begin{array}{rcl} x+y & = & y+x \\ (x+y)+z & = & x+(y+z) \\ (\alpha,r).x+(\alpha,r).x & = & (\alpha,2r).x \\ x+0 & = & x \end{array}$$

Next, we axiomatize the hiding operator and recursion, using the following four axioms.

$$\begin{array}{rcll} (\alpha,r).x/L & = & (\tau,r).(x/L) & (\alpha \in L) \\ (\alpha,r).x/L & = & (\alpha,r).(x/L) & (\alpha \notin L) \\ (x+y)/L & = & (x/L)+(y/L) & \\ A & = & t & (A := t) \end{array}$$

Following the tradition in other process algebras, we feel the need to add two auxiliary operators, $\underset{L}{\rhd}$ and $\underset{L}{\diamond}$, called left-merge and communication-merge, respectively, in order to finitely axiomatize parallel composition in PEPA. The semantics of these two operators are defined below.

$$\frac{x_0 \overset{(\alpha,r)}{\rightarrow} y_0}{x_0 \underset{L}{\rhd} x_1 \overset{(\alpha,r)}{\rightarrow} y_0 \underset{L}{\bowtie} x_1} \alpha \notin L \qquad \frac{x_0 \overset{(\alpha,r_0)}{\rightarrow} y_0 \quad x_1 \overset{(\alpha,r_1)}{\rightarrow} y_1}{x_0 \underset{L}{\diamond} x_1 \overset{(\alpha,R)}{\rightarrow} y_0 \underset{L}{\bowtie} y_1} \alpha \in L,$$

where R is defined in the semantics of parallel composition. Intuitively, the left-merge $\underset{L}{\rhd}$ behaves the same as parallel composition $\underset{L}{\bowtie}$ but it forces the first transition to be of type $\alpha \notin L$ and to be taken from its left-hand-side argument (or otherwise deadlocks). Communication merge $\underset{L}{\diamond}$ forces the first action to be of type $\alpha \in L$ and to be the result of a synchronization in which both the left- and the right-hand-side parameters take part. After performing such an action, communication merge continues as a parallel composition. We believe, following the well-known results for untimed [18] and timed [1] PAs, that PEPA (without the above-mentioned auxiliary operators) is not finitely axiomatizable. The following equations complete the axiomatization of PEPA by axiomatizing parallel composition.

$$\begin{array}{rcll} x \underset{L}{\bowtie} y & = & x \underset{L}{\diamond} y + x \underset{L}{\rhd} y + y \underset{L}{\rhd} x & \\ x \underset{L}{\bowtie} (y \underset{L}{\bowtie} z) & = & (x \underset{L}{\bowtie} y) \underset{L}{\bowtie} z & \\ (\alpha,r_0).x \underset{L}{\diamond} (\alpha,r_1).y & = & (\alpha,R).(x \underset{L}{\bowtie} y) & (\alpha \in L) \\ (\alpha,r).x \underset{L}{\diamond} y & = & 0 & (\alpha \notin L) \\ x \underset{L}{\diamond} y & = & y \underset{L}{\diamond} x & \\ x \underset{L}{\diamond} (y \underset{L}{\diamond} z) & = & (x \underset{L}{\diamond} y) \underset{L}{\diamond} z & \\ (\alpha,r).x \underset{L}{\rhd} y & = & (\alpha,r).(x \underset{L}{\bowtie} y) & (\alpha \notin L) \\ (\alpha,r).x \underset{L}{\rhd} y & = & 0 & (\alpha \in L) \\ (x+y) \underset{L}{\rhd} z & = & (x \underset{L}{\rhd} z)+(y \underset{L}{\rhd} z) & \\ 0 \underset{L}{\diamond} x & = & 0 & \\ x \underset{L}{\rhd} 0 & = & x & \\ 0 \underset{L}{\rhd} x & = & 0 & \end{array}$$

where, R is defined in the semantics of parallel composition and $r_\alpha(x)$ (in the definition of R) for the newly introduced left-merge and communication merge is given by the following equations.

$$\begin{array}{rcll} r_\alpha(x \underset{L}{\rhd} y) & = & r_a(x) & \text{if } \alpha \notin L \\ r_\alpha(x \underset{L}{\rhd} y) & = & 0 & \text{if } \alpha \in L \\ r_\alpha(x \underset{L}{\diamond} y) & = & 0 & \text{if } \alpha \notin L \\ r_\alpha(x \underset{L}{\diamond} y) & = & min(r_\alpha(x), r_\alpha(y)) & \text{if } \alpha \in L \end{array}$$

The following lemma states that our definition of rate (as well as the old definition in [14] given before) are in keeping with our axiomatization of PEPA.

lemma 1 *If $E \vdash p = q$, then $r_\alpha(p) = r_\alpha(q)$ for each processes p and q and each basic action α.*

Moreover, we prove that our axiomatization is indeed sound and complete.

theorem 2 *Let E be the union of the three sets of equations given above. E is a finite axiomatization for PEPA.*

Proof Sketch. Soundness proof is straightforward yet laborious and proceeds by giving, for each axiom, a bisimulation relation, relating left- and right-hand-side of all instances of equations.

For the completeness proof, we first focus on the nondeterministic sequential subset of PEPA. Then, for processes containing hiding and parallel composition, we show that using axioms as rewrite rules, one can eliminate these operators. Thus, for each two PEPA processes p and q such that $p \leftrightarrows q$, there exists PEPA processes p' and q' such that $E \vdash p = p'$, $E \vdash q = q'$ and p' and q' are nondeterministic sequential processes, i.e., do not contain hiding and parallel composition. Once we prove completeness for the nondeterministic sequential subset, it follows (from soundness) that $p \leftrightarrows p' \leftrightarrows q' \leftrightarrows q$ and (from the completeness of the subset) that $E \vdash p' = q'$. Further, from $E \vdash p = p'$ and $E \vdash q' = q$, we conclude that $E \vdash p = q$ which was to be proven.

For the completeness proof of the nondeterministic sequential subset, we show that each process p in this subset is provably equal (using E) to a process of the form $\sum_{i \in I}(a_i, r_i).p_i$ in which for all $i, j \in I$ if $i \neq j$ then $a_i \neq a_j$ or it does not hold that $p_i \leftrightarrows p_j$. This holds by a simple induction on the structure of p.

Then, for each two processes p and q such that $p \leftrightarrows q$, there exists two processes $p' \equiv \sum_{i \in I}(a_i, r_i).p_i$ and $q' \equiv \sum_{j \in J}(a_j, r_j).q_j$ such that $E \vdash p = p'$ and $E \vdash q = q'$ and thus, by soundness, $p \leftrightarrows p' \leftrightarrows q' \leftrightarrows q$. It follows from the semantics of nondeterministic choice that for each process p' of the form $\sum_{i \in I}(a_i, r_i).p_i$, if $p \overset{(a,r)}{\rightarrow} p'$, then $a = a_j$, $r = r_j$ and $p' \equiv p_j$ for some $j \in I$. We have that $p' \leftrightarrows q'$ and furthermore for each $i, i' \in I$ (and $j, j' \in J$), if $a_i = a_{i'}$, then it does not hold that $p_i \leftrightarrows p_{i'}$ ($q_j \leftrightarrows q_{j'}$); hence $\{(a_i, r_i).p_i \mid i \in I\} = \{(a_j, r_j).q_j \mid j \in J\}$ and thus, $E \vdash p' = q'$. It follows thus from $E \vdash p = p'$ and $E \vdash q' = q$ that $E \vdash p = q$. ⊠

As already mentioned in the proof sketch, using the equations in E as rewrite rules (from left to right), we can rewrite PEPA processes into a head normal form of the form $\sum_{i \in I}(a_i, r_i).P_i$ in which for all $i, j \in I$ if $i \neq j$ then $a_i \neq a_j$ or $P_i \not\equiv P_j$ where $\not\equiv$ denotes syntactic inequality. This will form the basis of our common stochastic form presented in Section 4.2.

2.2 MTIPP

Similar to PEPA, Markovian Processes for Timed Interaction (MTIPP) basic actions are represented with (a, λ). The main difference between MTIPP and PEPA is in the semantics of synchronization; namely, in MTIPP the rate of the result of synchronization (which is, as well, in the CSP-style) is the product of the rates. Our approach is applicable, and

currently implemented, to MTIPP with a marginal change concerning the semantics of synchronization.

2.3 IMC

IMC is a conservative extension of classical process algebras which separates action transitions from rate transitions. Consequently, IMC simplifies the semantics of synchronization by allowing only for plain actions to synchronize (rate transitions interleave). Another major semantic difference between IMC and other SPAs is that in IMC actions are nondeterministic just like in classical process algebras and the internal action is considered instantaneous and thus is given priority when composed nondeterministically with a stochastic rate (i.e., the stochastic rate will never get the chance to spend time since the internal action will be taken immediately). Both these semantic differences can be taken care of by using our common form for SPAs and rewriting IMC into this form using its axioms as rewrite rules.

2.4 EMPA

Extended Markovian Process Algebra (EMPA) [4] differs from PEPA in the following points.

1. EMPA has prioritized weighted immediate actions, denoted by $(a, \infty_{l,w})$ and passive actions, denoted by $(a, *)$.

2. The semantics of synchronization in EMPA is different. EMPA only allows for synchronization when at most one party is active and then the rate of a synchronization is defined as the maximum of the normalized rates of synchronizing actions.

3. EMPA has a class of relabeling operators, denoted by $P[\phi]$, which is parameterized by a relabeling function $\phi : \mathcal{A} \to \mathcal{A}$.

For our approach to be applicable to EMPA, we need to resolve the different types of choice as defined by priorities. Moreover, we have to adapt our semantics of synchronization to fit that of EMPA. Based on the axiom systems and the recipe given in [4, Section 3], we expect no theoretical challenges in this regard. There, the authors define a procedure which resolves the semantics of choice and parallelism and defines a multiset of pairs of actions and process: the action to be performed and the trailing process (which is basically the same as our common stochastic form for PEPA). There is a well-known problem concerning the congruence of Markovian bisimulation for EMPA which can be solved by restricting its syntax or focusing on a semantic subset.

3. MCRL2

mCRL2 is a successor of μCRL, and extends it by including features such as true concurrency (in terms of multi-actions), real time, higher-order functions and concrete data types. The specification formalism μCRL, in turn, extends the Algebra of Communicating Processes (ACP) [2] with abstract data types. We refer to [10] for a more elaborate description of mCRL2 features. The choice of mCRL2 as our target framework is motivated by the presence of abstract data types as a first-class entity in mCRL2. As we show in the next section, to our understanding, the only general and plausible solution for the unifying framework is to model stochastic processes as data types; the approach which models stochastic processes as classic processes is either bound for failure or has to resort to restrictions on / manual manipulations of the stochastic process as we illustrate later.

The summarized syntax of mCRL2 processes is given below.

$$p ::= a(d_1, \ldots, d_n) \mid \tau \mid \delta \mid p{+}p \mid p{\cdot}p \mid p \parallel p \mid \tau_I(p) \mid \partial_H(p) \mid \nabla_V(p) \mid \Gamma_C(p) \mid \textstyle\sum_{d:D} p \mid c \to p \diamond p$$

A basic action a of a process may have a number of arguments $d_1, \ldots, d_n$. These arguments correspond to the data elements. Action τ (which does not take any parameter) denotes an internal action. Process δ denotes the deadlock process in which no further transition is possible. Nondeterministic choice between two processes is denoted by the "+" operator. Processes can be composed sequentially and in parallel by means of "$\cdot$" and "$\parallel$". The abstraction operator $\tau_I(p)$ renames actions in I into τ and thus makes them invisible. To enforce synchronization, the encapsulation operator $\partial_H(p)$ specifies the set of actions H which are not allowed to occur. Conversely, the allow operator $\nabla_V(p)$ indicates the actions that are only allowed to occur. To show possible communications in a system and the resulting actions, communication operator $\Gamma_C(p)$ is used. The elements of set C are of the form $a_1 \mid a_2 \mid \cdots \mid a_n \to c$ (for $n \geq 0$), which intuitively means that action c is the result of the multi-party synchronization of actions a_1, a_2, ..., and a_n.

There are a number of built-in data types in mCRL2, such as (unbounded) integers, (uncountable) reals, lists, sets and functions which are quite useful for our implementation.

The mCRL2 tool-set contains tools for state space generation, reduction, analysis and visualization. Furthermore, it can be smoothly integrated with the CADP tool-set [8]. For our case-study, we used the state space generation and visualization facilities of mCRL2. For performance analysis, we integrated mCRL2 with Matlab and for functional analysis, we used CADP. (CADP provides some tools for state space reduction and steady-state and transient-state analysis of stochastic systems, as well.)

4. UNIFYING FRAMEWORK

4.1 General Approach

Our goal is to transform stochastic processes into a common form which can in turn be used for state space generation and verification as well as performance evaluation. The main obstacles in achieving this goal are listed below.

1. Classic PAs use transition relations, i.e., for each label l and each two states s and s', there is at most one transition labeled l from s to s'. However, some SPAs [4, 11] allow for multiple transitions from s to s' labeled with l.

   Moreover, different SPAs use different semantic frameworks. Some use labels combining actions and rates [14, 4], others separate action-labeled transitions from rate- (probability-)labeled transitions [11].

2. The semantics of synchronization in classic PAs is totally different from their SPA counterparts. When taking the relevant notions of behavioral equality into account, even nondeterministic choice gets a different interpretation in SPAs as opposed to PAs. For example,

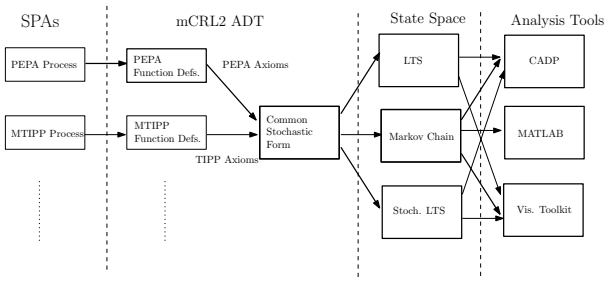


Figure 1: A Schematic View of Our Approach

in PAs for all relevant notions of behavioral equality we have $p+p=p$, while in SPAs this equality is usually unsound.

Furthermore, different SPAs use different semantics for synchronization; some only allow for hand-shaking synchronization among action-based transitions (thus, rates play no role in synchronization), others define synchronization only between active and passive processes and thus the rate of the outcome is determined by the rate of the active party, the third class allow for synchronization among (several) active processes. Even within the third category, there is no consensus as for the rate of the synchronized action; some define it as the product of the rates, others use more complicated calculations essentially taking the minimum of the rates [5].

To deal with these challenges, we define SPAs as Abstract Data Types in mCRL2. Then, we use sound axioms in each SPA to resolve parallel composition (and nondeterministic choice to the extent needed). Thus, we implement the axioms of concurrency in each SPA as rewrite theories which turn parallel processes into nondeterministic sequential ones. Furthermore, we resolve nondeterminism among identical processes by summing up the rates of corresponding identical actions. Finally, for each SPA, we write a simple interpreter in mCRL2 that takes the SPA process and a linear mCRL2 process (i.e., nondeterministic choice among different sequential processes). This last step enables us to generate a state space from the SPA process automatically. Further, we can use several existing tools as well as the tools presented in this paper, to analyze the generated state space. A schematic view of our approach is given in Figure 1.

An alternative approach to ours would be to translate PEPA processes into mCRL2 processes. This approach is taken in [9] for the process algebra LOTOS (as the target of the translation). Due to the semantic differences mentioned above, a structural and semantic preserving transformation from SPAs to PAs is very challenging, if not impossible. For the very reason, in [9], the authors had to restrict the syntax of their source stochastic specifications in order to prevent multi-transitions. Furthermore in [9], the synchronization of rates is not considered which is inherited from the IMC setting. The same comment holds for implementing PEPA processes as real-time mCRL2 processes.

Another possible approach, which we tried, is to define a PA process (for each SPA) which can interact with stochastic processes and interpret their behavior (according to the semantics of SPA). Consider PEPA, for example; the goal of this approach is to write an mCRL2 process context $C_{PEPA}[\_]$ such that for each PEPA process p, when $C_{PEPA}[p]$ is fed into the mCRL2 tool-set generates the state space as of p according to the PEPA semantics is generated. This way, by writing different process contexts $C_{TIPP}[\_]$, $C_{IMC}[\_]$, etc., one can perform functional and performance verification on different SPA processes using the state space generated by the mCRL2 tool-set.

However, this approach is also bound for failure since in many process algebras, such as PEPA and TIPP, $C[(\alpha,r).0 \underset{\{\alpha\}}{\bowtie} (\alpha,r).0]$ has a totally different state space from $C[(\alpha,r).0 \underset{\{\alpha\}}{\bowtie} ((\alpha,r).0+(\alpha,r).0)]$, i.e., C can distinguish between two strongly bisimilar processes. Thus, C is not implementable in mCRL2 (and in general in classic PAs).

4.2 Common Stochastic Form (CSF)

In this section, we define the Common Stochastic Form (CSF) which can accommodate semantics of different SPAs despite their very different semantics. A term in the CSF is of the form $P = \sum_{i\in I}(a_i, r_i).P_i$ where $a_i \in \{\alpha, \tau, Nil \mid \alpha \in \mathcal{A}\}$ and $r_i \in \mathbb{R}^{>0} \cup B$, where $B = \{n\top \mid n \in \mathbb{N}\}$. Furthermore, it should satisfy the following constraints.

1. For each $i, j \in I$ with $i \neq j$, it should hold that $a_i \neq a_j$ or $P_i \not\equiv P_j$.
2. If for some $i \in I$, $a_i = \tau$, then for no $j \in J$, $a_j = Nil$.
3. If for some $i \in I$, $a_i = Nil$, then for all $j \in I$ such that $a_j \in \mathcal{A} \cup \{\tau\}$, $r_j \in B$.

Apart from α and τ which represent basic and internal action types, respectively, Nil represents the empty action type which is used to model rate transitions, such as those in IMC. If the rate is set to $\top$, then the action is passive/instantaneous. Weighted passive actions with rates $n\top$ are used to give different probabilities to the choices among different instantaneous actions.

The first constraint given above forces the rewriting step to aggregate all rates among identical processes. In particular, it disallows terms of the form $(a,r).P + (a,r).P$ which is deemed equal to $(a,r).P$ by mCRL2 (and all other classic process algebras for that matter) while they are interpreted as $(a,2r).P$ by stochastic process algebras. The second constraint disallows implicit race condition among internal actions and stochastic transitions and it forces this race to be resolved in the rewriting step. Finally, the third constraint prevents the strange mixture of pure action and rate transitions, on one hand, and pairs of action and rates, on the other hand.

4.3 Application to PEPA

Using ANTLR compiler generator, we wrote a parsers for PEPA which automatically generates an mCRL2 specification in which PEPA processes (including their recursive definitions) are defined as Abstract Data Types. Furthermore, axiomatization of PEPA is included in the generated code as a set of rewrite rules (defined in terms of a function named `hnfrewrite`). This function calculates the rates of synchronizing actions and eliminates multi-transitions with the same labels to equivalent processes. Moreover, an mCRL2 process is defined which given a finite PEPA process (with only guarded recursion) in as its arguments, generates its state space. Thus, using mCRL2 tool-set one can generate the state space of the PEPA model.

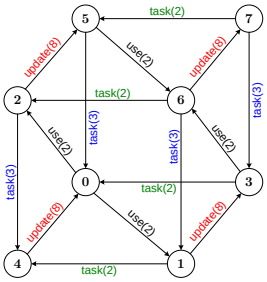


Figure 2: State space of a simple PEPA process

For example, consider the following simple PEPA process (an extended version of the example in [14, Section 3.5.7]).

$$\begin{array}{lcl} Process_1 & = & (use, 2).(task, 2).Process_1; \\ Process_2 & = & (use, 2).(task, 3).Process_2; \\ Resource & = & (use, 6).(update, 8).Resource; \\ Start & = & (Process_1 \bowtie Process_2) \underset{\{use\}}{\bowtie} Resource; \end{array}$$

Process *Start* (which is by default assumed to be the starting recursive definition in our implementation) models two processes, called $Process_1$ and $Process_2$ which compete with each other to get hold of the only available resource by synchronizing on action type *use*. The only difference among the two processes is that $Process_1$ is somewhat faster and after obtaining the resource usually waits for a shorter period before it requests the resource again. A summary of the mCRL2 code generated for this process is given in Figure 3. In this code, the PEPA processes are defined using the sort `PepaProcess`. Using the map `definition`, each recursive variable of the original program is mapped to a PEPA Process. The aim of the map `hnfrew` is to rewrite each PEPA process into its head normal form (`hnf`). The mCRL2 process `Exec`, executes an `hnf` (i.e., a PEPA process rewritten into its head normal form) by producing the actions of the form `mcrl_act` parameterized by the corresponding PEPA action type and rate, which are visible in the output state space. In the initial process, `Exec` is called by the `hnf` of the `p_start` process.

We implemented a simple program in Matlab which takes the generated state space, produces its rate matrix (by taking the underlying Markov Chain) and calculates steady-state probabilities of states based from the generated matrix.

The mCRL2 tool-set generated the state space depicted in Figure 2 for this code. Furthermore, we analyzed the steady-state probabilities of this state space using our Matlab program which resulted in the following probabilities.

State	0	1	2	3	4	5	6	7
Prob.	0.30	0.08	0.07	0.22	0.04	0.14	0.06	0.09

One can use our implementation to project on the functional aspects of the processes (thus, generate an LTS with action types and no rates) and use the CADP tool-set for functional analysis. Furthermore, one could abstract from different actions, reduce the state space and use the visualization toolkit (e.g., FSM View, ltsview, ltsgraph and DiaGraphica tools) to scrutinize different functional aspects of the system. We illustrate some of these possibilities on the case study presented in the next section.

```
sort
  Action = struct p_tau | p_update | p_task | p_use;
  RecursiveVars = struct p_Resource |
                  p_Process1 | p_Process2 | p_start;
  DelayAction = struct
        delayaction( p_act : Action, p_rate : Rate);
  Rate = struct num( r:Nat) |
                infty( w:Nat) | infinity;
  PepaProcess = struct p_nil?is_nil |
              p_recvar( rv:RecursiveVars)?is_recvar |
              p_prefix( d:DelayAction,
              p:PepaProcess)?is_prefix |
              ...
map
  definition      :   RecursiveVars -> PepaProcess;
eqn
  definition( p_Process1) =
         p_prefix( delayaction(p_use,num(2)),
           p_prefix( delayaction(p_task,num(2)),
                        p_recvar( p_Process1)));
  ...
map
  apparentrate    :   PepaProcess # Action -> Rate;

eqn
  apparentrate( p_prefix( a, p), b) =
      if( p_act(a) == b, p_rate(a), num(0));
  ...
sort
  Summand = struct summand( p_delayaction : DelayAction,
                            p_process : PepaProcess);
map
  hnfrew          :   PepaProcess -> Hnf;
eqn
  hnfrew( p_nil) = [];
  hnfrew( p_prefix( a, p)) = [summand( a, p)];
  hnfrew( p_choice( p, p_nil)) = hnfrew(p);
  hnfrew( p_choice( p1, p2)) =
       p_union( hnfrew( p1), hnfrew( p2));
  ...
act
  mcrl_act      :   Action # Rate;
proc
  Exec( hnf : Hnf) = sum i : Nat. ( i < #hnf) ->
      mcrl_act( p_act(p_delayaction( hnf.i)),
         p_rate(p_delayaction( hnf.i))).
          Exec( hnfrew( p_process( hnf.i)));
init
  Exec( hnfrew(  p_recvar(p_start) ));
```

Figure 3: mCRL2 spec. of a PEPA process

5. CASE STUDY

5.1 Informal Explanation

To illustrate our method, we specify an abstract PEPA model of a multichannel random access scheme, in the spirit of OFDMA, as described in [7]. We perform various functional and performance-related analyses on this specification. In OFDMA a given channel is divided into many parallel subchannels and hence, multiple symbols are sent in parallel. When two users simultaneously send packets to a subchannel their packets collide and get lost. After experiencing a collision in a subchannel, the clients have to back-off. Several collision resolution schemes exists for such networks. In the approach proposed in [7, Section III.A], after experiencing a collision, the client chooses another subchannel and retries the transmission immediately. Next, we give an abstract PEPA model of this protocol.

5.2 The PEPA Model

In our PEPA specification, we have two main subsystems: subchannel and client. The former is the description of a single subchannel in the system, and the latter specifies the behavior of a user.

Client. The clients request permission for sending their packet with rate r_{snd}. If no collision is detected they transmit their data with the rate specified by the subchannel. Upon detecting a collision, they nondeterministically choose a subchannel and retry the transmission. The PEPA model for each client i is defined as follows.

$$\begin{array}{lcl} Send_i & = & \sum_{j\in J}(req_j, r_{snd}).Chan_{i,j}; \\ Chan_{i,j} & = & (srv_j, \top).Send_i + (col_j, r_{inst}).Back_{i,j}; \\ Back_{i,j} & = & \sum_{j'\in J-\{j\}}(req_{j'}, r_{inst}).Chan_{i,j'}; \end{array}$$

Apart from nondeterministic back-off mechanism specified above, we specified and analyzed a deterministic back-off mechanism in which the next subchannel (using a modulo counter) is chosen for retransmission.
Since PEPA does not support immediate actions, we use the rate r_{inst} which represents a huge number (2000 in our implementation) to model the rate of immediate actions for retrying another subchannel.

Subchannel. Subchannels (frequencies) receive transmission requests from clients and either transmit the packets successfully or upon receiving another, i.e., a colliding, transmission request report a collision to clients.

$$\begin{array}{lcl} Freq_j & = & (req_j, \top).((srv_j, r_{srv}).Freq_j + \\ & & (req_j, \top).(col_j, r_{inst}).(col_j, r_{inst}).Freq_j); \end{array}$$

System. The system comprises the parallel composition of n subchannels and m clients and is defined as follows.

$$\begin{array}{lcl} start & = & (Freq_0 \underset{\emptyset}{\bowtie} \ldots \underset{\emptyset}{\bowtie} Freq_{n-1}) \\ & & \underset{req_0,\ldots,req_{n-1},srv_0,\ldots,srv_{n-1},col_0,\ldots,col_{n-1}}{\bowtie} \\ & & (Send_0 \underset{\emptyset}{\bowtie} \ldots \underset{\emptyset}{\bowtie} Send_{m-1}) \end{array}$$

5.3 Transformations and Analysis

We generated the state space of this case-study by varying several parameters such as the number of clients and subchannels and the rates of sending packets (by clients) and service (by subchannels). The following table shows the size of the generated state space for various number of clients and subchannels (the size of state space is oblivious to the stochastic rates).

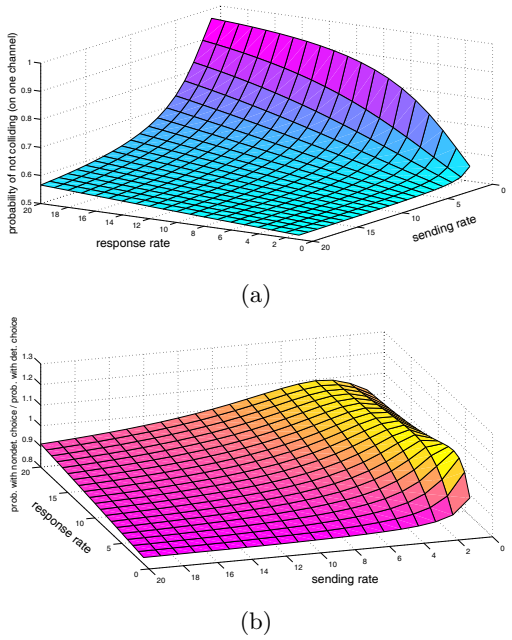
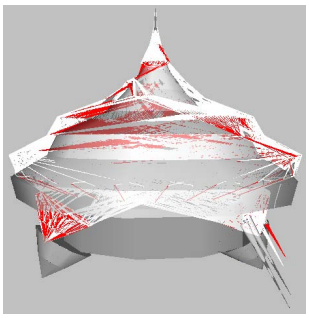


Figure 4: (a) Collision probabilities and (b) Nondeterministic vs. deterministic back-off mechanisms.

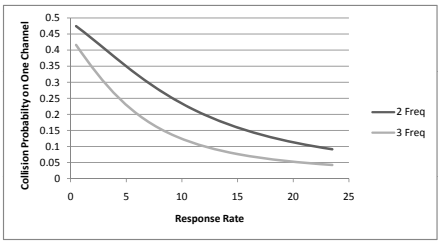
Subch. / Clients	1	2	3	4	5	6
1	2	9	30	83	218	553
2	3	35	231	1191	5713	25971
3	4	67	712	5866	42784	295330

Figure 5.(a) depicts the state space for 5 clients and 2 subchannels as visualized by the FSM View tool. The red transitions denote a collision.

Functional Analysis. The specification is checked automatically to be deadlock free while generating the state space by the mCRL2 tool-set. Furthermore, we use the CADP toolkit to model check the underlying LTS for the following properties specified in the μ-calculus. The first property asserts that after each request, client 1 either receives an acknowledgement that the transmission is successful or is informed about a collision. $[\top^*.req_{1,1}] < (not\ req_{1,1})^*.(srv_1\ or\ col_1) > \top$ The result of model-checking shows that the aforementioned property indeed holds. However, the following property does not hold in the system with 2 subchannels and 4 clients; it asserts that each path starting with a request will lead to a service action. $[\top^*.req_{1,1}]\mu X.(< srv_1 > true\ and < not\ srv_1 > X)$ The reason for the above property to be invalid is that there is an infinite path in which all requests on channel 1 collide and thus no successful transmission happens on this channel. Next, we analyze the probability of collision by varying different parameters of the system.



(a)



(b)

Figure 5: (a) Visualized state space and (b) Collision probabilities with 2 vs. 3 subchannels

Performance Evaluation. In the setting with 4 clients and 2 subchannels, we calculated the sum of steady state probabilities of states in which no collision occurs in one subchannel (i.e., a reward model in which all states with at least one collision in a specific subchannel are assigned reward value 0 and all other states are assigned 1); the result is shown in the diagram of Figure 4.(a). This diagram shows that the probability of not colliding on a subchannel increases exponentially with the increase in the response rates and decrease in sending rates. Figure 4.(b) shows the ratio of the probabilities of collision avoidance in a subchannel for the nondeterministic back-off scheme vs. the deterministic (choose next) one. It shows that in networks with a high-load (with greater sending rates), the deterministic scheme works better while the probability is higher with the nondeterministic scheme when there is little traffic in the network (with lesser sending rates). Figure 5 gives a comparison of collision probabilities between the cases in which the service capability is divided into two or three subchannels. In this experiment, the system comprises 4 clients, $r_{snd} = 2$ and $r_{inst} = 200$.

6. CONCLUSIONS

In this paper, we introduced a general approach for translating a number of Stochastic Process Algebras (SPAs) into the process algebra mCRL2. We implemented this approach for two typical examples of such SPAs, namely PEPA and MTIPP. We further implemented Matlab scripts to calculate steady-state probabilities and reward models for the underlying Markov chains of generated state spaces.

We are currently extending the implementation to SPAs such as EMPA and IMC. We plan to investigate the application of visualization techniques that, by incorporating rates, give us more insight about stochastic LTSs.

Acknowledgements. Useful comments of Holger Hermanns, Michel Reniers and Jan Friso Groote are acknowledged.

7. REFERENCES

[1] L. Aceto, A. Ingolfsdottir, and M. Mousavi. Impossibility results for the equational theory of timed ccs. In *Proceedings of CALCO'07*, LNCS, Springer, 2007.

[2] J.C.M. Baeten and W. P. Weijland. *Process Algebra*. Cambrdige, 1990.

[3] J. A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *I & C*, 60(1-3):109–137, 1984.

[4] M. Bernardo and R. Gorrieri. A tutorial on EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time. *TCS*, 202(1-2):1–54, 1998. (and Corigendum, *TCS*, 254(1-2):691–694, 2001.)

[5] E. Brinksma, H. Hermanns. Process Algebra and Markov Chains. vol. 2090 of LNCS, pages 183–231, Springer, 2001.

[6] P. Buchholz. On a markovian process algebra. Technical Report 500, Fachbereich Informatik, Universität Dortmund, 1994.

[7] Y.-J. Choi, S. Park and S. Bahk Multichannel random access in OFDMA wireless networks. *IEEE J. on Selected Areas in Communications*, 24(3):603–613, 2006.

[8] J.-C. Fernandez, H. Garavel, A. Kerbrat, L. Mounier, R. Mateescu, and M. Sighireanu. CADP - a protocol validation and verification toolbox. In *Proceedings of CAV'96*, vol. 1102 of *LNCS*, pages 437–440. Springer, 1996.

[9] H. Garavel and H. Hermanns. On combining functional verification and performance evaluation using CADP. In *Proceedings of FME'02*, vol. 2391 of *LNCS*, pages 410–429. Springer, 2002.

[10] J. F. Groote, A. Mathijssen, M. Reniers, Y. Usenko, and M. van Weerdenburg. The formal specification language mCRL2. In *Proceedings of the Dagstuhl Seminar*, 2007. Avilable from www.mcrl2.org.

[11] H. Hermanns. *Interactive Markov Chains: The Quest for Quantified Quality*, vol. 2428 of *LNCS*. Springer, 2002.

[12] H. Hermanns and J.-P. Katoen. Automated compositional Markov chain generation for a plain-old telephone system, *SCP*. 36:97–127, 2000.

[13] H. Hermanns and M. Rettelbach. Syntax, semantics, equivalence, and axioms for MTIPP. Technical Report 10/94, Friedrich-Alexender-Universität, Erlangen-Nürnberg, 1994.

[14] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge, 1996.

[15] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[16] R. Milner. *A Calculus of Communicating Systems*, vol. 92 of *LNCS*. Springer, 1980.

[17] R. Milner. A complete inference system for a class of regular behaviours. *JCSS*, 28:439–466, 1984.

[18] F. Moller The Importance of the Left Merge Operator in Process Algebras. In *Proceedings of ICALP'90*, vol. 443 of *LNCS*, pages 752–764, Springer, 1990.