UNE IMP EHE CWI ★QMIPS★ INR UER ZAR UTO

Quantitative
Modeling
In
Parallel
Systems

Stochastic Process Algebras — Constructive Specification Techniques Integrating Functional, Performance and Dependability Aspects

N. Götz[1], H. Hermanns, U. Herzog, V. Mertsiotakis, and M. Rettelbach

University of Erlangen-Nürnberg, IMMD VII
[1] IBM ENC Heidelberg

D W1.T2.2b-1.v1

22 September 1995

Distribution Level: Public

Approved by: U. Herzog

Abstract We introduce *Stochastic Process Algebras* as a novel approach for the structured design and analysis of both the functional behaviour and performability (i.e performance and dependability) characteristics of parallel and distributed systems. This is achieved by integrating stochastic modelling and analysis into the powerful and well investigated formal

The QMIPS Project: The QMIPS project is a collaborative research project supported by the CEC as ESPRIT-BRA project no 7269. It is being carried out by the following organisations: CWI (Amsterdam), EHEI (University of Paris V), Imperial College (London), INRIA (Sophia-Antipolis), University of Erlangen, University of Newcastle, University of Torino and University of Zaragoza.

description techniques of process algebras.
After advocating the use of Stochastic Process Algebras as a modelling technique we recapitulate the foundations of classical process algebras and – using our Stochastic Process Algebra TIPP – present extensions taking into account the requirements of performability analysis. We briefly survey the state-of-the-art by summarizing various applications, existing tool support and recent advances.

1 Introduction. The Indivisibility of Functional and Temporal Behaviour

Designing parallel and distributed systems we have to consider both functional specification and temporal aspects (Performance, Dependability). Usually these aspects are separated from each other. System designers use distinct hardware and software specification techniques while performance and dependability assurance is the task of modelling specialists. Such a separation was reasonable for uniprocessors and classical communication systems because it offers several advantages, mainly simplicity and understandability. Unfortunately, however, performance evaluation is often ignored and considered only when a complete misdesign is obvious. The same is true and even worse when regarding dependability aspects.

The situation is changing since distributed and parallel systems became the focus of general attention and a clear trend is observable: specification techniques have been extended by time attributes and performability evaluation techniques have been enhanced in order to capture the impact of functional dependencies. Such a combination of qualitative and quantitative modelling aspects is needed in many situations. Three typical examples may illustrate this:

- Specifying communication protocols often requires timeout mechanisms. Dependent on the selected time value the protocol may deadlock or not.
- Analysing the performance of parallel processors and distributed systems, we definitely have to consider functional dependencies and synchronisation of communicating subtasks. Program execution times, resource utilisation and speedup values significantly depend on these functional properties.
- Investigating systems, which are prone to hardware or software failures, has to deal with a strong correlation between the functionality, failure rates and performability measures.

The need for combined specification methods was already recognised in the seventies. The most successful examples are Stochastic Petri Nets (SPN) [1], Stochastic Graph Models [41, 42] and Stochastic Automata Networks [34]. We propose and recommend Stochastic Process Algebras [23, 12, 11, 26, 27, 24] mainly because of four important reasons:

1. Languages are the principal means for the design of hardware and software components: process algebras are abstract languages tailored to the description of parallel and distributed systems.[1]
2. Design automation and mechanisation are only possible, if we use formal techniques with a solid theoretical basis: process algebras are well founded, permit a detailed formal specification and support the process of implementation and verification.

[1] To be specific: we do not expect process algebras to be the specification languages of the future. However, we do expect such languages to overtake their exciting features.

3. Structuring is the only way to deal with complex systems: process algebras offer a design methodology, called constructivity, that allows to systematically build complex systems from smaller ones: there are operators available for composition as well as mechanisms for abstraction. And very important, process algebras offer an algebraic characterisation of equivalent system behaviour.

4. Combining stochastic modelling and analysis with the well investigated formal description technique of process algebras provides a rich source of further developments. The comprehensive theories of process algebras, which have been developed mainly during the last decade, bear an enormous potential also for performance and dependability modelling in general, for the modelling of distributed and parallel systems including high speed networks, mobile communication systems, for the modelling of manufacturing lines, etc.

This paper is organised as follows: in Section 2 we will give an overview on how everything began. We will introduce TIPP as an example of a Stochastic Process Algebra in Section 3, where we especially highlight the nice and important algebraic features of the approach. Section 4 illuminates the concept by means of a small example. Section 5 gives references to various case studies and Section 6 discusses tool support that is actually available. In Section 7 we will give an overview on recently obtained results. Section 8 concludes with a brief summary indicating also further challenges.

2 The Roots of Stochastic Process Algebra

Classical process algebras such as CCS [30], CSP [28] or ACP [3] are purely functional, i.e. the relative ordering of events is modelled, and alternative behaviour is expressed by non-determinism. From the very beginning their basic intention was to systematically build complex systems from smaller ones. The foundation is comprehensive and solid, excellent introductions may be found in the above mentioned books of Hoare [28] and Milner [30].

In the late eighties deterministic time descriptions were integrated in order to attack the "time-out" problem common to most communication protocols. Reed and Roscoe [36] may have initiated these research activities; a good overview on various approaches can be found in [32]. At about the same time, the first publications appeared dealing with both the functional and probabilistic behaviour of systems. The next step followed immediately and all the aspects - functional, probabilistic and deterministic temporal behaviour - were unified in one approach. To our knowledge, the dissertation of H. Hansson [14] treated all fundamental aspects for the first time extensively.

It is quite astonishing that Nounou and Yemini [33] have presented already in 1984 an approach containing most ideas of Stochastic Process Algebras. They began with a functional description and then mapped probabilistic and stochastic temporal features onto this basic process. They also show some performance results under exponential assumptions. We do not know why they did not continue research in this direction.

Attracted by lectures on classical process algebras and thrilled by the idea of systematic hierarchical modelling techniques, we started with a framework on Stochastic Process Algebras [23, 22] in the late eighties. We steadily extended our knowledge and got more and more involved in this exiting research area [37, 13, 12, 15, 11, 20]. Being sure that Stochastic Process Algebras offer some fundamental advantages, we have shown their basic ideas and concepts at many occasions, and have tried to demonstrate that this is an attractive alternative approach and a promising research area for many research groups.

Today there is a lively cooperation and interchange of ideas between the different research groups [26, 24] and several Stochastic Process Algebras have recently been published: Hillston's PEPA [27], Buchholz's MPA [6], Gorrieri et al's MPA [4]; our own language TIPP together with a tutorial introduction of its fundamental ideas has been published in [12]. Prototype tools are available for both PEPA [9] and TIPP [18]. The theoretical basis of TIPP is summarized in [11, 20].

3 The Stochastic Process Algebra TIPP

In this section we will present the main features of our Stochastic Process Algebra TIPP. The reader is expected to be familiar with the basic concepts of performance evaluation, reliability theory, and process algebras. A summary of the fundamental ideas and applicability of Stochastic Process Algebras can be found in [12].

The claim of Stochastic Process Algebras is to support the systematic description and construction of complex systems. Important features are therefore:

- the composition of components,
- the abstraction of details,
- the possibility to exchange subsystems, and
- the verification of functional and temporal properties.

In order to achieve these objectives one has to provide all elements of a complete process theory (cf. Fig. 1). The syntax of the language allows a modular and compact description of components and their interrelations. The semantic model (usually a labelled transition system) describes precisely and unambiguously the meaning of each language expression. The definition of equivalent behaviour allows to compare the behaviour of different components and systems with each other. Most important, the axiomatisation allows to provide algebraic laws permitting the comparison of systems (and components) already on the syntactic level. Last not least the semantic model, containing all information, can be analysed in order to obtain functional properties (e.g. deadlock, liveness), temporal properties (performance and dependability measures) as well as mixed properties (e.g. probability of deadlock, probability distribution functions of tagged event sequences).

It is important to note that usually the modelleris only concerned with the syntactic description and the resulting characteristics, while all other elements and transformations can be hidden by automatic tool support.

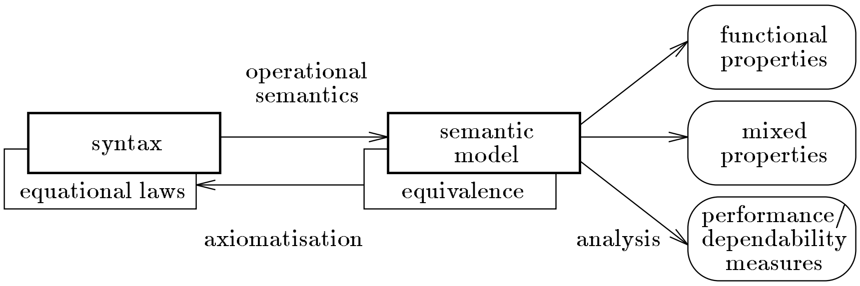


Figure 1: Overview on the Stochastic Process Algebra approach

3.1 Syntax

The process algebra TIPP – *ti*med *p*rocesses and *p*erformability evaluation – is an extension of the classical abstract languages CSP [28] and CCS [30] including random time variables. The basic elements to describe processes are actions and operators. Actions describe relevant activities of the system; they are considered to be atomic. By means of the operators, process descriptions can be composed from smaller ones in order to yield descriptions of more complex systems. Formally, the ways how descriptions can be built are defined by a grammar.

We assume a fixed set of action names $Act := Vis \cup \{\tau\}$, where we use τ as a distinguished symbol for internal, invisible actions and Vis as the set of regular, visible activities. An action named a can either be exponentially distributed or passive. Exponentially distributed actions – denoted by pairs of action name and rate (a, λ) – happen instantaneously after a duration that is exponentially distributed with rate λ. Passivity is indicated using the rate[2] 1.

Definition 3.1 *The set $\mathcal{L}$ of valid system descriptions of TIPP is given by the following grammar, where $a \in Act$, $\lambda \in I\!R^+$; $S \subseteq Vis$, $X \in Var$, Var is a set of process variables.*
$P \ ::= \ Stop \ \mid \ (a,\lambda).P \ \mid \ P+P \ \mid \ P \,\|_S\, P \ \mid \ P \backslash a \ \mid rec\ X : P \ \mid \ X$

To summarise the intuitive meaning of these basic elements: *Stop* denotes the halting process, the *prefix*ed process $(a, \lambda).P$ behaves as P after the action a has happened. The *choice* operator '+' allows to model alternative behaviour. By means of the *parallel* operator '$\|_S$' two processes are modelled to proceed independently, but they have to synchronise on actions within the set of synchronizing actions S. If no synchronisation is forced, we may omit the empty set index $\emptyset$. The *hiding* operator '\' provides an abstraction mechanism for actions that are internal at a certain level of specification. Infinite behaviour is formally expressed by means of the *recursion* operator '*rec*'. This can be regarded as a compact notation for the recursive equation $X := P$, where X may appear in P. In the remainder of this paper we will frequently use the latter notation, instead of using the *rec*-operator.

[2]We investigated several ideas to represent passive actions. The above presented way leads to correct rates and posseses nice algebraic properties.

3.2 Semantics

The behaviour of a process of this language is formalized in a structural operational style (cf. [35]). The deduction rules in Figure 2 define a labelled transition system (LTS) as a domain for the semantic model, consisting of nodes representing states or process behaviours and arcs representing transitions between them. These arcs are labelled with pairs of actions and rates[3]. Each of these arcs ($P \xrightarrow{a,\lambda} Q$) symbolises that P can evolve to Q by performing the action a with rate λ.

Definition 3.2 *The transition relation* $\longrightarrow \;\subseteq\; \mathcal{L} \times (Act \times I\!R^{+} \times Lab) \times \mathcal{L}$ *is defined as the least relation satisfying the rules in Figure 2, where* $Lab := \{\epsilon, +_l, +_r, \|_l, \|_r, (,)\}$.

$$\frac{}{(a,\lambda).P \xrightarrow{a,\lambda,\,\varepsilon} P} \qquad \frac{P \xrightarrow{a,\lambda,\,w} P'}{P + Q \xrightarrow{a,\lambda,\,+_l w} P'} \qquad \frac{Q \xrightarrow{a,\lambda,\,w} Q'}{P + Q \xrightarrow{a,\lambda,\,+_r w} Q'}$$

$$\frac{P \xrightarrow{a,\lambda,\,w} P'}{P \|_S Q \xrightarrow{a,\lambda,\,\|_l w} P' \|_S Q}\ (a \notin S) \qquad \frac{Q \xrightarrow{a,\lambda,\,w} Q'}{P \|_S Q \xrightarrow{a,\lambda,\,\|_r w} P \|_S Q'}\ (a \notin S)$$

$$\frac{P \xrightarrow{a,\lambda,\,v} P' \quad Q \xrightarrow{a,\mu,\,w} Q'}{P \|_S Q \xrightarrow{a,\lambda\mu,\,(v,w)} P' \|_S Q'}\ (a \in S) \qquad \frac{P\{(recX : P)/X\} \xrightarrow{a,\lambda,\,w} P'}{recX : P \xrightarrow{a,\lambda,\,w} P'}$$

$$\frac{P \xrightarrow{a,\lambda,\,w} P'}{P\backslash a \xrightarrow{\tau,\lambda,\,w} P'\backslash a} \qquad \frac{P \xrightarrow{b,\lambda,\,w} P'}{P\backslash a \xrightarrow{b,\lambda,\,w} P'\backslash a}\ (a \neq b)$$

Figure 2: Semantic rules of TIPP

3.3 Equivalences

Due to the strict formal semantics of TIPP it is possible to reason about the behaviour of a system in an algebraic setting. For example it is possible to characterise systems with the same functionality, but potentially different temporal characteristics. This is formally captured by an equivalence relation called *functional bisimulation* defined in [20], based on the definition of strong bisimulation in [30]. The idea is that the two systems can simulate each other in any of the states they are going through.

From the performance point of view it is more interesting to classify systems by means of their functional and temporal behaviour, as well. This classification is formally carried out by *Markovian bisimulation* ($\sim_M$) [20], an equivalence relation that abstracts away from different syntactic possibilities to describe the same behaviour. Hillston has pointed out that an equivalence of this kind can be exploited for state space reduction based on lumpability [27].

Functional and Markovian bisimulation are congruences, i.e. they allow the exchange of equivalent components that may be specified fairly different *without* changing the overall

[3]An additional third label is used for reasons of semantic consistency, cf. [12]. For the purpose of this paper we can neglect this label.

behaviour of the complex system. This embodies one of the main advantages of the SPA approach, *constructivity*: a complex system can be constructed step by step of smaller components. These components may be subject of some optimisation or manipulation effort – as long as the optimal candidate exhibits equivalent behaviour, the behaviour of the whole system is *not* affected.

Strictly speaking, these equivalences are defined on TIPP's *semantic* model. They interrelate transition systems rather than syntactic descriptions. Every equivalence class consists of equivalent, but not necessarily identical transition systems. Some of them are smaller, others are larger. This is due to additional information that is irrelevant under the behavioural view of the equivalence.

3.4 Axiomatisation

As the states of our transition systems are labelled with the corresponding process descriptions of $\mathcal{L}$, the bisimulations mentioned above also induce equalities on the *syntactic* level. For Markovian bisimulation this equality can be completely characterised by a set of equational laws (cf. Fig. 3). All these laws, together with four additional laws dealing with recursion, form a sound and complete axiomatisation of $\sim_M$, as long as no recursion over hiding and parallel composition occurs. The proof is carried out in [20].

$$
\begin{array}{rcl@{\qquad}rcll}
P + \mathit{Stop} &=& P & \mathit{Stop}\backslash a &=& \mathit{Stop} & \\
P + Q &=& Q + P & (P+Q)\backslash a &=& P\backslash a + Q\backslash a & \\
(P+Q)+R &=& P+(Q+R) & ((b\,,\,\lambda).P)\backslash a &=& (b\,,\,\lambda).(P\backslash a) & a \neq b \\
(a\,,\,\lambda).P + (a\,,\,\mu).P &=& (a\,,\,\lambda+\mu).P & ((a\,,\,\lambda).P)\backslash a &=& (\tau\,,\,\lambda).(P\backslash a) &
\end{array}
$$

$$
\begin{array}{rcl@{\qquad}rcl}
P \parallel_{\{\}} \mathit{Stop} &=& P & (P \parallel_S Q) \parallel_S R &=& P \parallel_S (Q \parallel_S R) \\
P \parallel_S Q &=& Q \parallel_S P & & &
\end{array}
$$

$$
\begin{array}{rcl}
P \parallel_S Q &=& \sum\limits_{a_i \notin S} (a_i\,,\,\lambda_i).\,(P_i \parallel_S Q) \;+\; \sum\limits_{b_j \notin S} (b_j\,,\,\mu_j).\,(P \parallel_S Q_j) \;+ \\
&& \sum\limits_{a_i \in S} \; \sum\limits_{b_j = a_i} (a_i\,,\,\lambda_i\mu_j).\,(P_i \parallel_S Q_j)
\end{array}
$$

$$
\text{where } P \equiv \sum_i (a_i\,,\,\lambda_i).P_i \text{ and } Q \equiv \sum_j (b_j\,,\,\mu_j).Q_j
$$

Figure 3: Axiomatisation of Markovian bisimulation

4 Example. A Multiprocessor with $MMPP$ Arrival Stream

In order to demonstrate the expressiveness of Stochastic Process Algebras and to show how some basic but frequently used modules like arrival streams, processors or queueing and scheduling mechanisms can be described and analyzed with TIPP, we present as a simple example a $MMPP^2|M|4$ queueing system with two types of customers and non-preemptive priorities.

One of the main advantages of process algebras is that they allow the creation of highly modular model descriptions. In our case, we can consider our system as the parallel composition of two components or processes:

$$System := Load \parallel_A Machine \qquad \text{with} \qquad A := \{hp\_job, lp\_job\}$$

The process *Load* represents the $MMPP$-arrival stream, the server is modelled by the process *Machine* and A contains the interfacing actions.

4.1 Load modelling

The arrival stream is supposed to be a Markov Modulated Poisson Process with two phases. During a normal phase jobs arrive with the rates λ_1 and μ_1, while in a busy phase arrivals occur with the rates λ_2 and μ_2. The process LP_i models the low priority job stream and HP_i represents the high priority job stream. The action c that is performed with rate ϕ leads to a change in the phase of the arrival processes and is performed synchronously by the two arrival processes.

$$\begin{array}{rcl}
Load & := & HP_1 \parallel_{\{c\}} LP_1 \\
HP_1 & := & (hp_job, \mu_1).HP_1 \\
 & + & (c, \phi).HP_2 \\
HP_2 & := & (hp_job, \mu_2).HP_2 \\
 & + & (c, \phi).HP_1
\end{array}
\qquad
\begin{array}{rcl}
 & & \\
LP_1 & := & (lp_job, \lambda_1).LP_1 \\
 & + & (c, \phi).LP_2 \\
LP_2 & := & (lp_job, \lambda_2).LP_2 \\
 & + & (c, \phi).LP_1
\end{array}$$

4.2 The machine model

The server consists of six processes that are composed in parallel. Two of them represent the queueing component and four model the service unit

$$Machine := (Q_0 \parallel_{\{get\_lp\_job\}} R_0) \parallel_B (P \parallel P \parallel P \parallel P)$$

where $B = \{get\_hp\_job, get\_lp\_job\}$.

Jobs are processed according to a FIFO strategy with priorities. To realize this scheduling mechanism we introduce for each class of jobs a separate queue process with maximum capacity l. The queue Q_0 stores the low priority jobs while the queue R_0 accepts the high priority jobs.

$$\begin{array}{rcl}
Q_0 & := & (lp_job, 1).Q_1 \\
Q_i & := & (lp_job, 1).Q_{i+1} \\
 & + & (get_lp_job, \alpha).Q_{i-1} \\
Q_l & := & (get_lp_job, \alpha).Q_{l-1} \\
 & + & (lp_job, 1).Q_l
\end{array}
\qquad
\begin{array}{rcl}
R_0 & := & (hp_job, 1).R_1 \\
 & + & (get_lp_job, 1).R_0 \\
R_i & := & (hp_job, 1).R_{i+1} \\
 & + & (get_hp_job, \alpha).R_{i-1} \\
R_l & := & (get_hp_job, \alpha).R_{l-1} \\
 & + & (hp_job, 1).R_l
\end{array}$$

The priority mechanism is realized by appropriate synchronization of the two queue processes. The process Q_i for instance is only able to deliver a job to a processor, if the other queue is in state R_0, i.e. if no high priority job is waiting to be processed. Otherwise, the action *get_lp_job* is not enabled.

Each processor waits until it can carry out a *get_hp_job*-action or a *get_lp_job*-action. The service time of high priority jobs and low priority jobs is exponentially distributed with rate ν and ξ, respectively.

$$P := (get\_hp\_job, 1).P_0 + (get\_lp\_job, 1).P_1$$

$$P_0 := (hp\_job\_ready, \nu).P \qquad\qquad P_1 := (lp\_job\_ready, \xi).P$$

4.3 Model simplification

To evaluate the above specification, we derive the semantic model based on the semantic rules of TIPP first (see Figure 2). By doing so we may observe that the transition system contains much too detailed information. This is due to the modular representation of the processing component where each processor is modelled individually. The number of states of the model would be clearly smaller if we do not distinguish the individuals and represent the whole processing component by a single process. However, to do this by hand makes model creation more difficult and error prone. With process algebras like TIPP this simplification can be done systematically using the equational laws of Markovian bisimulation. Moreover, an automation of this reduction step appears to be feasible and is currently under investigation.

In the above case, a simplified version of the server can be obtained as follows. Using the defining equation for each processor P the description expands to

$$\begin{aligned}
&P \parallel P \parallel P \parallel P \;= \\
&\quad (get_hp_job, 1).(P_0 \parallel P \parallel P \parallel P) \;+\; (get_lp_job, 1).(P_1 \parallel P \parallel P \parallel P) + \\
&\quad (get_hp_job, 1).(P \parallel P_0 \parallel P \parallel P) \;+\; (get_lp_job, 1).(P \parallel P_1 \parallel P \parallel P) + \\
&\quad (get_hp_job, 1).(P \parallel P \parallel P_0 \parallel P) \;+\; (get_lp_job, 1).(P \parallel P \parallel P_1 \parallel P) + \\
&\quad (get_hp_job, 1).(P \parallel P \parallel P \parallel P_0) \;+\; (get_lp_job, 1).(P \parallel P \parallel P \parallel P_1)
\end{aligned}$$

We may notice that the various subprocesses differ only by the position of the active processor. This information is redundant if we do not want to distinguish the individual processors. Markovian bisimulation takes this into account by comparing the behaviour of processes instead of their internal state. Clearly, the behaviour of $(P_1 \parallel P \parallel P \parallel P)$ and $(P \parallel P \parallel P \parallel P_1)$ is not distinguishable for an external observer. Consequently, applying the axioms we can reduce the expression above and obtain

$$P \parallel P \parallel P \parallel P \;=\; (get\_hp\_job, 4).(P_0 \parallel P \parallel P \parallel P) \;+\; (get\_lp\_job, 4).(P_1 \parallel P \parallel P \parallel P).$$

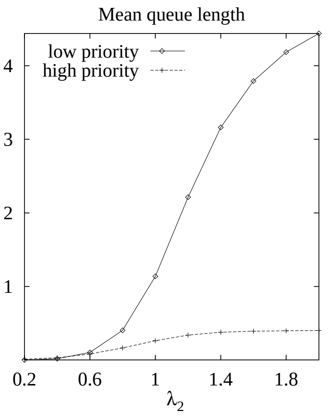
The same reduction is repeated for the processes $(P_0 \parallel P \parallel P \parallel P)$ and $(P_1 \parallel P \parallel P \parallel P)$. Again, some subprocesses appear that are amenable for subsequent reduction. This iterative procedure terminates, because the whole number of subprocesses of our description is limited. Finally, we substitute the parallel components by arbitrary process variables and replace the original server $(P \parallel P \parallel P \parallel P)$ by the simplified version, named (for example) P_{simple0}, where

$$\begin{aligned}
P_{\text{simple0}} &:= (get_hp_job, 4).P_{\text{simple1}} + (get_lp_job, 4).P_{\text{simple2}} \\
P_{\text{simple1}} &:= \ldots \\
&\;\;\vdots
\end{aligned}$$

The decrease of the state space size is immense. Instead of $O(e^p)$ states the simplified model complexity is of order $O(p^2)$, where p is the number of processors.

4.4 Numerical results

To investigate the performance of this model we analysed numerically the underlying Markov chain using the Gauß-Seidel method. We derived some performance measures that are presented in Figure 4. In these graphs the mean queue length as well as the throughput for high priority jobs and low priority jobs are shown dependent on the arrival rate for low priority jobs during the busy phase (λ_2).



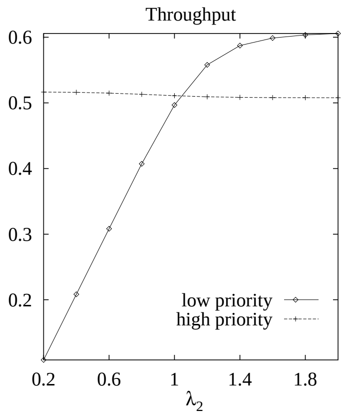


Figure 4: Performance measures

5 Case Studies

During the last years we have applied TIPP to construct and analyze systems of various types. These practical experiences demonstrate that this approach is a suitable way to construct large system in an elegant way, under integral consideration of performance and dependability aspects. The case studies include

- data transfer with multiple send-and-wait communications [13]
- multiprocessor and task graph modelling [12]
- electronic mail system (with J. Hillston) [16]
- alternating bit protocol [19]
- robot control system (with S. Gilmore, J. Hillston and R. Holton) [10]
- multiprocessor mainframe with software failures [17]

All these models were built up out of small modules that were composed to obtain the entire system. The application of the semantic rules (generation of the state space) was done automatically in most cases. Restrictions to the complexitiy (number of states) of the models are only due to the numerical solution methods. Reduction of the state space by means of our notion of equivalence was partially performed. The multiprocessor mainframe example additionally contains concepts for the decomposition of the generated state space along the structure given by the modular process description.

There were various types of performability measures derived such as response time, availability of components, failure probabilities, mean execution times and many more.

6 Tool Support

At the moment there are two tools available, the *PEPA Workbench* [9] and our *TIPP-tool* [18]. TIPP-Tool is a prototype modelling tool for creating and evaluating stochastic process algebra models of parallel and distributed systems. It supports a LOTOS-oriented input language ([5]) and apart from facilities to apply functional analysis based on reachability analysis, it comes up with numerical solution modules for the stationary as well as transient analysis of the Continuous Time Markov Chain (CTMC), underlying a TIPP-specification. The input language and the emphasis on efficient numerical evaluation techniques represent the main difference to the *PEPA Workbench*, another modelling tool for SPAs relying on symbolic evaluation of the underlying Markov chain [9].

The derived labelled transition system serves as a base for further reduction into a CTMC. For the steady state analysis of the underlying CTMC the TIPP-tool uses a variety of numerical algorithms: LU-factorization, power method, and Gauss-Seidel iteration scheme. It supports also transient analysis by providing methods to compute the mean time to absorption of an absorbing Markov chain or the transient state probabilities. For the latter, a refined randomization scheme is provided, similar to the method adopted by Lindemann in [29].

The result of numerical analysis is usually a vector with state probabilities. In order to obtain more sophisticated and expressive results the user can specify measures. This is done via rewards beeing assigned to states that match a certain regular expression the user has to specify. Experiment series are also supported by allowing rates to be symbolic variables. The specification of the model as well as the measures and experiments is supported through a graphical user interface.

7 Recent Extensions

With respect to the elements of a complete theory (cf. Fig. 1) extensions of the approach are possible on each described level, i.e. the syntax of the language, the semantic concepts and the evaluation strategies. Recent research has lead to extensions of the language expressiveness that have been shown to fit into the concept of SPA sketched above.

The introduction of immediate and probabilistic actions to achieve the expressiveness and the modelling power comparable with Generalised SPN is described in [21, 38]. Apart from adapting the semantic rules, we achieved new notions of equivalent behaviour that are conservative extensions of classical notions. They allow to eliminate immediate actions inside of arbitrary components of a complex system as soon as these actions do not have an observable impact on the environment. This feature, known as *congruence property* can also be carried out on the specification level, because an extension of the set of equational laws has been established.

The application of these laws on the syntactic level can avoid state space explosion. For this purpose, an on the fly reduction algorithm based on term rewriting has been proposed in [21]. Its efficient implementation is currently under development. Starting from a highly modular syntactic system description this algorithm will automatically produce a syntactic representation for a compact semantic model *before* the state space is generated.

The structural operational style of the semantics described in Section 3 is the usual approach to define the behaviour of process algebraic system descriptions. A completely different semantic concept has been published in [39]. In this paper a denotational semantics is presented for a restricted set of operators directly mapping syntactic descriptions onto transition matrices. The benefit of this concept is that it is paid attention to get a minimal semantic model in each step of model creation. Another denotational semantics has been established in [40]. The semantic domain of this approach is a class of Stochastic Petri Nets. This link between SPN and SPA has lead to fruitfull cooperation within the QMIPS project, cf. [8, 7] and provides interesting insight into properties of different SPAs.

Another positive outcome of the QMIPS project concerns the performance analysis of certain classes of systems. In [31] the evaluation of an infinite number of states is made possible by connecting the process description language to the Spectral Expansion solution method by means of an appropriate semantic concept.

The solution of large models for performability analysis usually suffers from the stiffness problem. This difficulty has been tackled by developing a *time scale decomposition* technique for SPA that uses structural properties of the system description [25]. Currently considerable work is going on aiming at characterizing classes of SPA descriptions that are amenable to product form solutions.

8 Summary and Outlook

Stochastic Process Algebras have been proven to provide a valuable concept to systematically construct complex systems including both functional and temporal aspects.

A complete theory is available for Stochastic Process Algebras with Markovian, i.e. exponentially distributed durations of actions. Only recently immediate actions have been included allowing elegant modelling and analysis of management functions (e.g. scheduling) and dependable systems (stiffness).

Ongoing and future research will cover mainly three different directions:

- extension of the theoretical basis,
- efficient and structured evaluation techniques (including tool support), and
- widespread application to complex system modelling and analysis.

The Workshop on *Process Algebras and Performance Modelling* has been established as an annual forum to exchange ideas and research results, where the scientific community dealing with SPA meets regularly. The participation in this workshop as well as valuable discussions with various external research groups indicate that the general interest in SPA is steadily increasing.

References

[1] M. Ajmone Marsan, G. Balbo, and G. Conte. *Performance Models of Multiprocessor Systems*. MIT Press, 1986.

[2] F. Baccelli and I. Mitrani, eds. *Quantitative Modelling in Parallel Systems*. Springer, 1995.

[3] J. A. Bergrstra and J.W.Klop. Algebra of Communicating Processes with Abstractions. *TCS*, 37(1):77–121, 1985.

[4] M. Bernardo, L. Donatiello, and R. Gorrieri. MPA: A Stochastic Process Algebra. Technical report tr94-10, University of Bologna, 1994.

[5] T. Bolognesi and E. Brinksma. Introduction to the ISO Specification Language LOTOS. In P.H.J. van Eijk, C.A. Vissers, and M. Diaz, eds., *The Formal Description Technique LOTOS*, p. 23–73, Amsterdam, 1989. North-Holland.

[6] P. Buchholz. Markovian Process Algebra: Composition and Equivalence. In U. Herzog and M. Rettelbach, eds., *Proc. of the 2nd Workshop on Process Algebras and Performance Modelling*, p. 11–30, Regensberg/Erlangen, July 1994. Arbeitsberichte des IMMD, Universität Erlangen-Nürnberg.

[7] S. Donatelli, H. Hermanns, U. Herzog, J. Hillston, V. Mertsiotakis, M. Rettelbach, and M. Ribaudo. Process Algebras and/or Petri Nets for Integrated Performance Modelling? Tutorial at the SIGMETRICS/PERFORMANCE'95, 1995.

[8] S. Donatelli, H. Hermanns, J. Hillston, and Marina Ribaudo. GSPN and SPA Compared in Practice: Modelling a Distributed Mail System. In Baccelli and Mitrani [2].

[9] S. Gilmore and J. Hillston. The PEPA Workbench: A Tool to Support a Process Algebra-Based Approach to Performance Modelling. In G. Haring and G. Kotsis, eds., *7th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, p. 353–368, Wien, May 1994.

[10] S. Gilmore, J. Hillston, R. Holton, and M. Rettelbach. Specifications in Stochastic Process Algebra for a Robot Control Problem. CSG Report Series ESG-CSG-7-94, Computer Systems Group, Dept. of Computer Science, The University of Edinburgh, The King's Buildings, Edinburgh EH9 3JZ, December 1994.

[11] N. Götz. *Stochastische Prozeßalgebren – Integration von funktionalem Entwurf und Leistungsbewertung Verteilter Systeme*. PhD thesis, Universität Erlangen–Nürnberg, Martensstraße 3, 91058 Erlangen, April 1994.

[12] N. Götz, U. Herzog, and M. Rettelbach. Multiprocessor and Distributed System Design: The Integration of Functional Specification and Performance Analysis Using Stochastic Process Algebras. In *Proc. of the 16th Int. Symposium on Computer Performance Modelling, Measurement and Evaluation, PERFORMANCE 1993, Tutorial*. Springer LNCS 729, 1993.

[13] N. Götz, U. Herzog, and M. Rettelbach. TIPP – Introduction and Application to Protocol Performance Analysis. In H. König, editor, *Formale Beschreibungstechniken für verteilte Systeme*. Saur Verlag, Reihe: FOKUS, 1993.

[14] Hans A. Hansson. *Time and Probability in Formal Design of Distributed Systems*. PhD thesis, University of Uppsala, 1991.

[15] H. Hermanns. Semantik für Prozeßsprachen zur Leistungsbewertung. Master's thesis, Universität Erlangen–Nürnberg, IMMD VII, November 1993.

[16] H. Hermanns, U. Herzog, J. Hillston, V. Mertsiotakis, and M. Rettelbach. Stochastic Process Algebras: Integrating Qualitative and Quantitative Modelling. Technical Report 11/94, Universität Erlangen–Nürnberg, IMMD VII, Martensstr. 3, 91058 Erlangen, May 1994.

[17] H. Hermanns, U. Herzog, and V. Mertsiotakis. Stochastic Process Algebras as a Tool for Performance and Dependability Modelling. In *Proc. of IEEE Int. Computer Performance and Dependability Symposium*, p. 102–111, Erlangen, April 1995. IEEE Computer Society Press.

[18] H. Hermanns and V. Mertsiotakis. A Stochastic Process Algebra Based Modelling Tool. submitted to 11th U.K. Performance Engineering Workshop for Computer and Telecommunication Systems, 1995.

[19] H. Hermanns, V. Mertsiotakis, and M. Rettelbach. Performance Analysis of Distributed Systems Using TIPP – a Case Study. In J. Hillston and R. Pooley, eds., *Proc. of the 10th U.K. Performance Engineering Workshop for Computer and Telecommunication Systems*, p. 131–144. Dept. of Computer Science, University of Edinburgh, 1994.

[20] H. Hermanns and M. Rettelbach. Syntax, Semantics, Equivalences, and Axioms for MTIPP. In U. Herzog and M. Rettelbach, eds., *Proc. of the 2nd Workshop on Process Algebras and Performance Modelling*, p. 71–88, Erlangen-Regensberg, July 1994. IMMD, Universität Erlangen-Nürnberg.

[21] H. Hermanns, M. Rettelbach, and T. Weiß. Formal characterisation of immediate actions in SPA with nondeterministic branching. *The Computer Journal*, special issue: Proc. of the 3rd Workshop on Process Algebras and Performance Modelling, 1995.

[22] U. Herzog. EXL — Syntax, Semantics and Examples. Technical Report 16/90, Friedrich-Alexander University Erlangen-Nürnberg, IMMD VII, 1990.

[23] U. Herzog. Formal Description, Time and Performance Analysis. A Framework. In T. Härder, H. Wedekind, and G. Zimmermann, eds., *Entwurf und Betrieb Verteilter Systeme*, p. 172–190. Springer Verlag, Berlin, IFB 264, 1990.

[24] U. Herzog and M. Rettelbach, eds. *Proc. of the 2nd Workshop on Process Algebras and Performance Modelling*, volume 27-4. Arbeitsberichte des IMMD (Informatik), Universität Erlangen–Nürnberg, 1994.

[25] J. Hillston and V. Mertsiotakis. A simple Time Scale Decomposition technique for SPA. *The Computer Journal*, special issue: Proc. of the 3rd Workshop on Process Algebras and Performance Modelling, 1995.

[26] J. Hillston and F. Moller, eds. *Proc. of the 1st Workshop on Process Algebras and Performance Modelling*. Dept. of Computer Science, University of Edinburgh, 1993.

[27] Jane Hillston. *A Compositional Approach to Performance Modelling*. PhD thesis, University of Edinburgh, 1994.

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

[29] C. Lindemann. DSPNexpress: A Software Package for the Efficient Solution of Deterministic and Stochastic Petri Nets. In *Proc. of the 6th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, p. 15–29, Edinburgh, September 1992.

[30] R. Milner. *A Calculus of Communicating Systems*. Prentice Hall, London, 1989.

[31] I. Mitrani, A. Ost, and M. Rettelbach. Tipp and the spectral expansion method. In Baccelli and Mitrani [2].

[32] X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. In Springer LNCS 600, editor, *Real-Time: Theory in Practice*, 1991.

[33] N. Nounou and Y. Yemini. Algebraic specification-based performance analysis of communication protocols. In *Proc. Int. Workshop on Protocol Specification, Testing and Verification, 4th*, p. 541–560, 1985.

[34] B. Plateau and K. Atif. Stochastic Automata Network for Modeling Parallel Systems. *IEEE Transactions on Software Engineering*, 17(10):1093–1108, 1991.

[35] Gordon Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Computer Science Dept. Aarhus University, September 1981.

[36] G.M. Reed and A.W. Roscoe. A Timed Model for Communication Sequential Processes. *Theoretical Computer Science*, 58:249–261, 1988.

[37] M. Rettelbach. Leistungsbewertung mit Prozeßalgebren. Master's thesis, Universität Erlangen–Nürnberg, IMMD VII, Februar 1991.

[38] M. Rettelbach. Towards a theory of generalised SPA. *The Computer Journal*, special issue: Proc. of the 3rd Workshop on Process Algebras and Performance Modelling, 1995.

[39] M. Rettelbach and M. Siegle. Compositional Minimal Semantics for the Stochastic Process Algebra TIPP. In U. Herzog and M. Rettelbach, editors, *Proc. of the 2nd Workshop on Process Algebras and Performance Modelling*. University of Erlangen-Nürnberg, IMMD, 1994.

[40] Marina Ribaudo. *On the Relationship between Stochastic Petri Nets and Stochastic Process Algebras*. PhD thesis, University of Torino, 1995.

[41] R. Sahner and K. Trivedi. Performance Analysis and Reliability Analysis Using Directed Acyclic Graphs. *IEEE Transactions on Software Engineering*, SE-13(10), October 1987.

[42] F. Sötz. A Method for Performance Prediction of Parallel Programs. In H. Burkhart, editor, Proc. of CONPAR 90–VAPP IV, Joint Int. Conf. on Vector and Parallel Processing, p. 98–107, Zürich, Switzerland, September 1990. Springer–Verlag, Berlin, LNCS 457.