# Notes on non-Markovian Extension of Value Passing CCS[⋆]

D. Latella[1], M. Loreti[2], and M. Massink[1]

(1) CNR-ISTI - (2) Univ. di Firenze
EU Integrated Project Sensoria
{Diego.Latella,Mieke.Massink}@isti.cnr.it, loreti@dsi.unifi.it

## 1 Introduction

We define a non-Markovian extension of Value Passing CCS, based on PH-distributions. The approach consists in describing the execution time of each action with a PH-distribution, i.e. by associating a finite CTMC with a terminal, absorbing, state to *each* action of a process. Such CTMC characterises a random variable, namely the time needed to go from the initial state to the final one. It is well known that such distributions can be used for approximating, ideally, any general distribution and are closed under max. The choice of a CCS-like pattern of interaction is motivated by the fact that we are interested in service-oriented computing models and many process calculi for service-oriented computing are based on such a pattern of interaction (see e.g. [8, 1, 2, 6]).

Proposals for non-Markovian extensions of stochastic process algebras are available in the literature (see, e.g. [5, 3]). They are typically based on a CSP-like pattern of interaction and not a CCS-like one. This situation motivates our work.

After a brief introduction on PH-distributions, the language extension is presented. Input/output rendez-vous (r.v.) is modeled by a synchronised *start* of the input and output actions pair forming the r.v., followed by the composition (namely interleaving) of the CTMC modeling the execution duration of the input action with the CTMC modeling the execution duration of the output action, all followed by the synchronised *termination* of the r.v. Synchronised start and termination are currently represented by $\tau$-transitions, while the transitions of the CTMCs are labeled by rates, as usual. All such transitions can be interleaved with those of other concurrent processes, actually r.v's; in fact we present a very restricted version of the language, since this is sufficient for showing our approach. The language includes only input/output actions and parallel composition.

Each process is thus associated to a LTS whose labels are either $\tau$ or rates of exponential distributions which contribute to the construction of the relevant PH-distributions. Several ways for dealing with $\tau$ transitions are briefly discussed.

[⋆] Research supported by the Project FET-GC II IST-2005-16004 Sensoria and MIUR Project TOCAI.IT.

## 2 PH-CCS

### 2.1 PH-distributions

The following brief introduction to PH-distributions is essentially taken from [4]. For more information on PH-distributions the reader is referred to [7]. Intuitively, a PH-distribution is characterised by the time until absorption in a finite state continuous time Markov process with a single absorption state, i.e. a state with no outgoing transitions. Formally, let $\mathcal{M}$ be a CTMC with transient states $\{1, \ldots, m\}$ and absorbing state $m+1$, initial probability vector $(\widetilde{\pi}, \pi_{m+1})$ and infinitesimal generator matrix

$$\mathbf{Q} = \begin{pmatrix} \mathbf{T} & \mathbf{T}_0 \\ \mathbf{0} & 0 \end{pmatrix}$$

where $\mathbf{T}$ is a square matrix of order $m$ such that $\mathbf{T}_{ii} < 0$ and $\mathbf{T}_{ij} \geq 0$ for $i \neq j$. Moreover the row sums of $\mathbf{Q}$ must equal zero, i.e. $\mathbf{T}\widetilde{\mathbf{1}} + \mathbf{T}_0 = \widetilde{\mathbf{0}}$ (see Fig. 1)

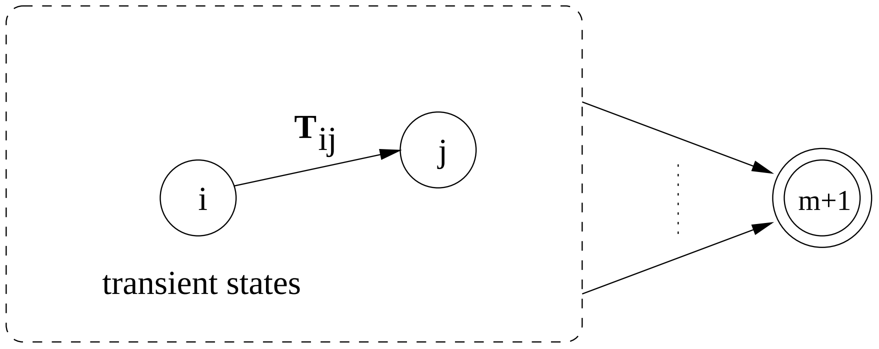

**Fig. 1.** A CTMC with unique absorbing state $m+1$

The probability distribution $F(x)$ of the time until absorption in state $m+1$ is given by:

$$F(x) = \begin{cases} 1 - \widetilde{\pi} e^{\mathbf{T}x} \widetilde{\mathbf{1}}, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases}$$

where $e^{\mathbf{T}x}$ is defined as follows:

$$e^{\mathbf{T}x} = \mathbf{I}_m + \mathbf{T}x + \mathbf{T}^2 \frac{x^2}{2!} + \mathbf{T}^3 \frac{x^3}{3!} + \ldots$$

with $\mathbf{I}_m$ being the identity matrix of order $m$ and $\mathbf{T}^n \frac{x^n}{n!}$ the matrix $\mathbf{T}^n$ with each element multiplied by $\frac{x^n}{n!}$. The corresponding probability density function is given below:

$$F'(x) = \begin{cases} \widetilde{\pi} e^{\mathbf{T}x} \mathbf{T}_0, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases}$$

**Definition 1.** *A continuous distribution function $F$ on $[0, \infty)$ is called a* phase-type *distribution (PH-distribution) if and only if it is the distribution of time of absorption in a CTMC with unique absorbing state as above.*

PH-distributions can be used for approximating general distributions and some of them are quite well known. For instance, the Erlang distribution, with rate $\lambda$ and $m$ phases, and probability density function $f_{\lambda,m}(x) = \frac{\lambda^m x^{m-1} e^{-\lambda x}}{(m-1)!}$, for $x > 0$, is just a special case of PH-distributions (see Fig. 2) and is normally used for approximating the *constant* random variable $\frac{m}{\lambda}$ since its mean value is $\frac{m}{\lambda}$ while its variance is $\frac{m}{\lambda^2}$. Notice that the simple rate $\lambda$ exponential distribution is in turn a special case of Erlang, namely when $m = 1$.

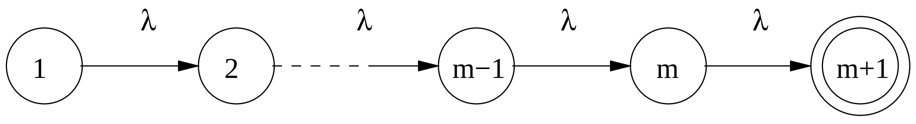

**Fig. 2.** The $\lambda, m$ Erlang distribution

## 2.2 Syntax and informal semantics

Since the purpose of this note is purely demonstrative, we restrict our attention to a very simple process language, namely one composed only of the null process $\mathbf{0}$, Value Passing CCS-like output and input action prefix, and process parallel composition. Output action prefix takes the form $a\,!_M\, v.P$, while input action prefix looks like $a\,?_M\, x.P$, where $M$ specifies a CTMC with the above properties. Intuitively, the execution of $a\,!_M\, v$ in $a\,!_M\, v.P$ has a random duration which is given by the time until absorption in the final state of $M$, with initial distribution assigning probability 1 to the initial state of $M$. Similarly for $a\,?_M\, x.P$. In order for an action to actually execute, its process must be active in a context where a complementary action is ready to execute. In this case the two actions *synchronise* their *start*. After that, and when *both* CTMCs associated to the actions have reached their final state, the complementary actions *synchronise* their *termination*. In the present notes, such start and termination synchronisation is modeled by means of $\tau$-actions. A discussion on possible alternative options can be found later on in these notes. The formal Syntax is shown in Fig. 3. We assume disjoint countable sets $\mathcal{N}$, ranged over by $a, b, \ldots$ of names, $\mathcal{B}$ and $\mathcal{X}$, ranged over by $v, v', \ldots$ and $x, y, \ldots$, of basic values and variables. Syntactic category $M$ defines a simple language for the specification of finite CTMC; it includes summation of rate prefixes, where each $\lambda_i$ is a positive real number, the terminated CTMC $\mathcal{T}$ and variable terms $X$ for which a proper defining equation $X \stackrel{\triangle}{=} M$ is required.

$$P, Q \ ::= \ \mathbf{0} \mid a\,!_M\, v.P \mid a\,?_M\, x.P \mid P|Q$$

$$M \ ::= \ \textstyle\sum_{i=1}^{k} \lambda_i.N_i$$

$$N \ ::= \ \mathcal{T} \mid M \mid X$$

**Fig. 3.** Syntax of processes.

Notice that, in order to characterise a sensible timing specification, a CTMC term $M$ should enjoy the property that the terminated state $\mathcal{T}$ is reachable, in one or more steps, from each other state of $M$. Given that $M$ can denote only a finite CTMC the above requirement can be effectively and easily checked.

*Example 1.* Consider $P1 = a\,!_{\lambda.\mathcal{T}}\, v.\mathbf{0}$ and $P2 = a\,?_{\mu.\mathcal{T}}\, x.\mathbf{0}$. The LTS of $P1|P2$ defined by the operational semantics is shown in Fig. 4 where states are as follows:

| | |
|---|---|
| 1 | $a\,!_{\lambda.\mathcal{T}}\, v.\mathbf{0} \vert a\,?_{\mu.\mathcal{T}}\, x.\mathbf{0}$ |
| 2 | $(\mathcal{RV}\, r)(a\,!^r_{\lambda.\mathcal{T}}\, v.\mathbf{0} \vert a\,?^r_{\mu.\mathcal{T}}\, v.\mathbf{0})$ |
| 3 | $(\mathcal{RV}\, r)(a\,!^r_{\mathcal{T}}\, v.\mathbf{0} \vert a\,?^r_{\mu.\mathcal{T}}\, v.\mathbf{0})$ |
| 4 | $(\mathcal{RV}\, r)(a\,!^r_{\lambda.\mathcal{T}}\, v.\mathbf{0} \vert a\,?^r_{\mathcal{T}}\, v.\mathbf{0})$ |
| 5 | $(\mathcal{RV}\, r)(a\,!^r_{\mathcal{T}}\, v.\mathbf{0} \vert a\,?^r_{\mathcal{T}}\, v.\mathbf{0})$ |
| 6 | $\mathbf{0} \vert \mathbf{0}$ |

Notice that the terms corresponding to states 2 to 6 are not part of the process language, as defined by the grammar of Fig. 3. In fact, for the purpose of the operational semantics definition, the grammar needs to be extended with the addition of the following process terms: $a\,!^r_M\, v.P, a\,?^r_M\, v.P, (\mathcal{RV}\, r)P$, where $r \in \mathcal{N}$. As we will see, terms of the form $a\,!^r_M\, v.P$ and $a\,?^r_M\, v.P$ are used to model the execution of $a\,!_M\, v.P$ and $a\,?_M\, x.P$ during r.v. $r$, which is in turn modeled by the r.v. operator $(\mathcal{RV}\, r)$.

The LTS of Fig. 4 corresponds, apart from the two $\tau$-transitions, to the PH-distribution modeling the *max* of the random variables with exponential distribution with rates $\lambda$ and $\mu$.

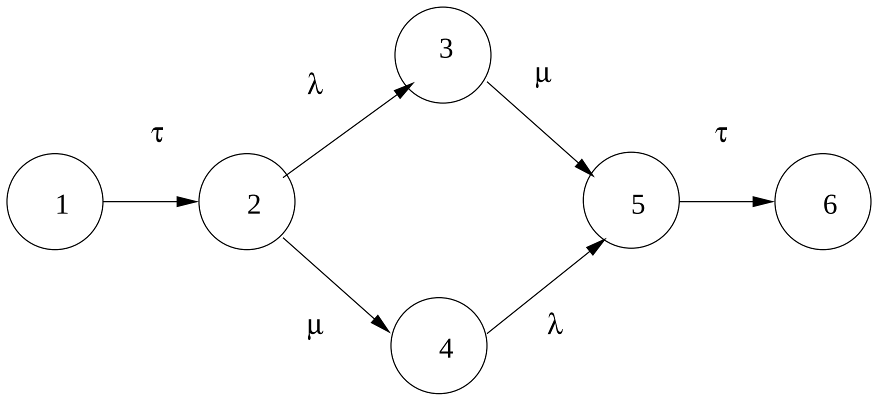

**Fig. 4.** LTS

### 2.3 Semantics

The operational semantics definition associates a LTS to each process $P$. In these notes we abstract from methods for defining multi-relations. Usual techniques, like e.g. proved LTS, can be used to that purpose and the choice for the most appropriate technique is postponed. The structural congruence laws are given below:

$$(\mathcal{RV}\, r)P \equiv (\mathcal{RV}\, r')P[r'/r], \text{if } r \notin fn(P)$$

We say that a process term is *well formed* if it contains *zero* or *exactly two* occurrences of any $r \in \mathcal{N}$. In the sequel all process terms are assumed well formed. We let $\xrightarrow{\Lambda}$ be the labelled transition relation induced by the rules of figures from 5 to 9, where $fn(P)$ denotes the set of the free names of $P$. Transition labels are defined by the following grammar:

$$\Lambda ::= \tau \,|\, \lambda \,|\, r : \alpha \,|\, r : \alpha, \lambda \,|\, r : \dagger \,|\, r : \alpha, \dagger$$
$$\alpha ::= a!v \,|a?v$$

We use the notation $\bar{\alpha}$, defined as follows:

$$\bar{\alpha} \stackrel{def}{=} \begin{cases} a?v, \text{if } \alpha = a!v \\ a!v, \text{if } \alpha = a?v \end{cases}$$

In the sequel we will often use brackets for labels, e.g. $(r : \alpha, \dagger)$, for avoiding confusion. Moreover, we call a $\Lambda$-transition a transition labeled by $\Lambda$. $\lambda$-transitions are generated according to the CTMC specification part of the language and they are used in turn for generating the $\lambda$-transitions of the LTS of the process where such CTMC specifications are used, as we will see in a moment. A $(r : \alpha)$-transition of a process models the *offer* of the process to start a r.v. for $\alpha$, where $r$ is the unique name of the r.v.'s to be started; r.v.'s are uniquely named, in a way which closely resembles Caspis sessions [1, 2]. The r.v. $r$ is started if there is a complementary $(r : \bar{\alpha})$-transition offered by another process, in which case both transitions synchronise, this resulting in a $\tau$-transition. A $(r : \alpha, \lambda)$-transition is generated for each $\lambda$-transition of the CTMC associated with $\alpha$ in r.v. $r$. The offer to terminate r.v. $r$ for $\alpha$ is modeled by a $(r : \alpha, \dagger)$-transition; when such a transition synchronises with a complementary $(r : \bar{\alpha}, \dagger)$-transition the r.v. actually terminates, which is signalled by a $(r : \dagger)$-transition. As we will see, this last transition will be turned into a $\tau$-transition by the (innermost embracing) $(\mathcal{RV}\, r)$ operator.

The transition relation for the PH-type distribution specification part $M$ of the language is given in Fig. 5.

(CHO) $$\sum_{i=1}^{k} \lambda_i.N_i \xrightarrow{\lambda_i} N_i$$

(CON) $$\frac{M \xrightarrow{\lambda} N,\ X \overset{\triangle}{=} M}{X \xrightarrow{\lambda} N}$$

**Fig. 5.** Rules for PH-type distribution specification

The *termination* term $\mathcal{T}$ is used for denoting the absorbing state of the CTMC for the PH-Type distribution. Its presence in input/output actions generates a transition labeled by the special symbol † which is used for synchronising termination of the execution of such complementary actions, as we will see. The CTMC associated to a term $M$ can be easily computed from the derivatives of $M$, in the usual way[1]. Notice, moreover, that term $M$ implicitly characterises the initial distribution of the associated CTMC, namely the probability distribution which assigns 1 to the state corresponding to the term $M$ itself and 0 to all the others.

(OAP) $$\frac{r \notin \mathrm{fn}(P)}{a\,!_M\, v.P \xrightarrow{r:a!v} a\,!_M^r\, v.P}$$

(IAP) $$\frac{r \notin \mathrm{fn}(P), v \in \mathcal{B}}{a\,?_M\, x.P \xrightarrow{r:a?v} a\,?_M^r\, v.P[v/x]}$$

**Fig. 6.** Rules for processes (output and input) action prefix

As we already mentioned, r.v. is modeled using a notion similar to that of Caspis session, where the special operator $(\mathcal{RV}\, r)$ is used, instead of restriction.

[1] As we will see in the sequel, we will never compute the CTMC of a term $M$ in isolation; such CTMC is actually embedded in the LTS associated to the process term using $M$ and is computed together with such LTS.

The reason why restriction is not used is the fact that in our case the transitions of a r.v. $r$ are not synchronisation transitions like in Caspis sessions, but just $(r : \alpha, \lambda)$-transitions, interleaved with $(r : \bar{\alpha}, \lambda')$-transitions, all generated from the $\lambda/\lambda'$-transitions of the CTMCs associated with $\alpha$ and $\bar{\alpha}$; such $(r : \alpha, \lambda)/(r : \bar{\alpha}, \lambda')$-transitions must simply "pass through" the boundary of the $(\mathcal{RV}\, r)$ operator with all the label information discarded, except the rate; notice that hiding $r$, $\alpha$ and $\bar{\alpha}$ avoids interference between different r.v.'s. A more detailed description of r.v. modeling is given below.

(RVTO) $$a\,!^r_{\mathcal{T}}\, v.P \xrightarrow{r:a!v,\dagger} P$$

(RVTI) $$a\,?^r_{\mathcal{T}}\, v.P \xrightarrow{r:a?v,\dagger} P$$

(RVEO) $$\frac{M \xrightarrow{\lambda} N}{a\,!^r_M\, v.P \xrightarrow{r:a!v,\lambda} a\,!^r_N\, v.P}$$

(RVEI) $$\frac{M \xrightarrow{\lambda} N}{a\,?^r_M\, v.P \xrightarrow{r:a?v,\lambda} a\,?^r_N\, v.P}$$

**Fig. 7.** Rules for *rendezvous* execution and termination offer

R.v. start synchronisation is similar to Caspis session opening and it is taken care of by rules (OAP) and (IAP) of Fig. 6, together with rule (SS) of Fig. 8. Essentially, a process which is ready to start an interaction via $a\,!_M\, v.P$ generates a $(r : a!v)$-transition offering $(r : a!v)$, with fresh $r$, and becomes $a\,!^r_M\, v.P$; this latter process will generate the transitions of the CTMC $M$, with their labels enriched with $r : a!v$ (see rule (RVEO) of Fig. 7). Similarly, $a\,?_M\, x.Q$ may generate a transition labelled by $r : a?v$ for $v \in \mathcal{B}$ and $r$ fresh; an early semantics is used.

Rule (SS) defines r.v. start synchronisation and the result is a term of the form $(\mathcal{RV}\, r)R$. Any $(r : \alpha, \lambda)$-transition of $R$ , i.e. related to r.v. $r$ gives rise to a $\lambda$-labeled transition of $(\mathcal{RV}\, r)R$ (see rule (HDE) of Fig. 9). In other words, these transitions correspond to transitions of the CTMCs modeling the execution of the complementary actions of the r.v. Axioms (RVTO) and (RVTI) of Fig. 7 are related to r.v. termination. In particular, they generate r.v. $r$ termination offer (by either party). When both parties are ready to terminate r.v. $r$, i.e. they offer $(r : \alpha, \dagger)$ and $(r : \bar{\alpha}, \dagger)$ respectively, the r.v. itself is terminated; r.v. $r$ *termination completion* is modeled by a $(r : \dagger)$-transition (rule (TS) of Fig.8). Rules (INT1) and (INT2) of Fig.8 take care of interleaving in the usual way. R.v. $r$ termination is turned into a $\tau$-transition by the $(\mathcal{RV}\, r)$ operator (rule (HDT) of Fig. 9), which otherwise hides r.v. $r$ identification information (rule (HDE)) for CTMC transitions and discards all other transitions concerning $r$ (in this case, behaving like a restriction operator), leaving all other transitions (i.e.

(SS) $$\frac{P \xrightarrow{r:\alpha} P',\; Q \xrightarrow{r:\bar{\alpha}} Q'}{P|Q \xrightarrow{\tau} (\mathcal{RV}\, r)(P'|Q')}$$

(TS) $$\frac{P \xrightarrow{r:\alpha,\dagger} P',\; Q \xrightarrow{r:\bar{\alpha},\dagger} Q'}{P|Q \xrightarrow{r:\dagger} P'|Q'}$$

(INT1) $$\frac{P \xrightarrow{\Lambda} P'}{P|Q \xrightarrow{\Lambda} P'|Q}$$

(INT2) $$\frac{Q \xrightarrow{\Lambda} Q'}{P|Q \xrightarrow{\Lambda} P|Q'}$$

**Fig. 8.** Rules for process parallel composition

those which are *not* related to r.v. $r$) untouched. Finally notice that a complete sequentialisation of different r.v. can be obtained by omitting rule (HDo).

(HDE) $$\frac{P \xrightarrow{r:\alpha,\lambda} P'}{(\mathcal{RV}\, r)P \xrightarrow{\lambda} (\mathcal{RV}\, r)P'}$$

(HDT) $$\frac{P \xrightarrow{r:\dagger} P'}{(\mathcal{RV}\, r)P \xrightarrow{\tau} P'}$$

(HDo) $$\frac{P \xrightarrow{\Lambda} P',\; r \notin \Lambda}{(\mathcal{RV}\, r)P \xrightarrow{\Lambda} (\mathcal{RV}\, r)P'}$$

**Fig. 9.** Rules for hiding

For the purpose of CTMC construction, we define the LTS of a process $P$ in the usual way:

**Definition 2.** *The set of Rate and $\tau$ Derivatives of well-formed process $P$, $De_{R\tau}(P)$, is the smallest set such that both the following two conditions hold:*

- $P \in De_{R\tau}(P)$
- *if $Q \in De_{R\tau}(P)$ and $Q \xrightarrow{\lambda} Q'$ for positive real number $\lambda$ or $Q \xrightarrow{\tau} Q'$, then also $Q' \in De_{R\tau}(P)$.*

**Definition 3.** *The LTS of process $P$, $LTS(P)$ is the tuple $(S, ACT, s0, \longrightarrow)$ s.t.*

- $S \overset{def}{=} De_{R\tau}(P)$
- *$ACT \subseteq \{\tau\} \cup \mathbb{R}_{\geq 0}$ with*
  *$ACT \overset{def}{=} \{x | \exists s, s' \in S$ s.t. $s \xrightarrow{x} s'$ can be derived using the rules of Fig.5 to 9$\}$*

– $s0 = P$
– $\longrightarrow \subseteq S \times ACT \times S$ *with* $(s, x, s') \in \longrightarrow$ *iff* $s \xrightarrow{x} s'$ *can be derived using the rules of Fig.5 to 9.*

## 2.4 Dealing with $\tau$-transitions

For term $P$, $LTS(P)$ is not a CTMC only because of $\tau$ transitions modeling start/end r.v. synchronisation. There are several options for dealing with such $\tau$s:

1. Treating them as *immediate* actions. *Immediate* actions in the context of SPAs have received wide attention and we can use one of the solutions already proposed in the literature.
2. Treating them as normal, rated, transitions; the simplest way is to force any CTMC term start with a single rate prefix (i.e. $\lambda.M$, and then combine such rates in the synchronised r.v. start using total input/output rates, as in Markovian Caspis. More sophisticated solutions may relax the above single rate start restriction. Similarly for termination synchronisation.
3. Investigate if one can synchronise initial alternatives of CTMC, without forcing to initial single CTMC start rate.
4. Brute force $\tau$ elimination, using some form of behavioural equivalence.

## References

1. M. Boreale, R. Bruni, L. Caires, R. De Nicola, I. Lanese, M. Loreti, F. Martins, U. Montanari, A. Ravara, D. Sangiorgi, V. Vasconcelos, and G. Zavattaro. SCC: a Service Centered Calculus. In M. Bravetti and G. Zavattaro, editors, *3rd International Workshop on Web Services and Formal Methods. WS-FM 2006*, volume 4184 of *LNCS*, pages 38–57. Springer-Verlag, 2006.
2. M. Boreale, R. Bruni, R. De Nicola, and M. Loreti. Sessions and pipelines for structured service oriented programming. In *Formal Methods for Open Object-Based Distributed Systems*, LNCS. Springer-Verlag, 2008. To appear.
3. M. Bravetti. Specification and analysis of stochastic real-time systems, 2002. Ph.D Thesis. Universita' di Bologna.
4. E. Brinksma, J. Katoen, R. Langerak, and D. Latella. A stochastic causality-based process algebra. *The Computer Journal. Oxford University Press.*, 38(7):552–565, 1995.
5. H. Hermanns and J. Katoen. Automated compositional Markov chain generation for a plain-old telephone system. *Science of Computer Programming. Elsevier*, 36(1):97–127, 2000.
6. A. Lapadula, R. Pugliese, and F. Tiezzi. A calculus for orchestration of web services. In R. De Nicola, editor, *Programming Languages and Systems, 16th European Symposium on Programming (ESOP)*, volume 4421 of *LNCS*, pages 33–47. Springer-Verlag, 2007.
7. M. Neuts. *Matrix-geometric Solutions in Stochastic Models—An Algorithmic Approach.* The Johns Hopkins University Press, Baltimore, 1981.
8. SENSORIA WP2. D2.1a: Core calculi for Service Oriented Computing, 2006. http://www.pst.ifi.lmu.de:8080/Sensoria/DOWNLOAD/wp2