Probabilistic KLAIM

Alessandra Di Pierro[1], Chris Hankin[2], and Herbert Wiklicky[2]

[1] Dipartimento di Informatica, Universitá di Pisa, Italy
[2] Department of Computing, Imperial College London, UK

Abstract. We introduce a probabilistic extension of KLAIM, where the behaviour of networks and individual nodes is determined by a probabilistic scheduler for processes and probabilistic allocation environments which describe the logical neighbourhood of each node. The resulting language has two variants which are modelled respectively as discrete and continuous time Markov processes. We suggest that Poisson processes are a natural probabilistic model for the coordination of discrete processes asynchronously communicating in continuous time and we use them to define the operational semantics of the continuous time variant. This framework allows for the implementation of networks with independent clocks on each site.

1 Introduction

The design of languages supporting network programming is a necessary step towards the formalisation of distributed and mobile computing. The existence of an abstract semantic framework constitutes the basis for a formal analysis of such systems. The KLAIM paradigm [6] provides such a semantic framework by introducing basic concepts and primitives addressing the key aspects of the coordination of interacting located processes. We extend this basic paradigm with probabilistic constructs with the aim of introducing a semantic basis for a quantitative analysis of networks. A quantitative analysis allows in general for the consideration of more "realistic" situations. For example, a probabilistic analysis allows for establishing the security of a system up to a given tolerance factor expressing how much the system is actually vulnerable. This is in contrast to a qualitative analysis which typically might be used to validate the absolute security of a given system. In a distributed environment quantitative analysis is also of a great practical use in the consideration of timing issues which involve the asynchronous communications among processes running with different clocks.

Specifically in our probabilistic version of KLAIM, which we call pKLAIM, we introduce probabilities in a number of ways. At the local level, we introduce probabilistic parallel and choice operators. In addition we use probabilistic allocation environments which associate distributions on physical sites with logical localities. We propose two variants of pKLAIM which differ in the way in which probabilities are introduced at the network level. In the *discrete time* variant each node also has a probability associated with it; this indicates the chance that the process at that node will be selected for execution. The *continuous*

$s \in S$	(physical) sites	$v \in Val$	(basic) values
$l \in Loc$	(logical) locality	$e \in Exp$	(basic) expressions
$\ell \in Loc \cup S$	(general) locality	$A \in Proc$	(predefined) processes
$x \in Var$	(value) variable		
$u \in LVar$	(locality) variable	$\varrho \in View$	allocation environment
$X \in PVar$	(process) variable		

Table 1. Syntactic Categories

time variant instead associates a "rate" with each node; this determines how often a node is active.

In the next section we introduce the language in its two variants. Sections 3 through to 5 present the operational semantics of pKLAIM; we present the local semantics, which is common to both variants, in Section 3, the discrete time model in Section 4 and the continuous time model in Section 5. Section 5 ends with an example which highlights the differences between the two variants. Our main motivation is to provide a basis for quantitative analysis; one example of this kind of analysis might be to look at the rate at which a virus can spread through a network. We sketch the way in which such a scenario can be modelled in the final main section.

2 Syntax of pKLAIM

The syntax of pKLAIM is based essentially on the one given for KLAIM originally in [6]. It is based on the syntactic categories in Table 1.

The process terms in pKLAIM are formed according to rules which are very similar to the ones in the original KLAIM language [6]. The main difference is the probabilistic weighting of choice and local parallelism in Table 2.

The syntax of pKLAIM networks can be defined according to two different models: in the *discrete time* model the transition from one network configuration to another is done with a certain probability in one single step which determines the clock for all nodes in the network; in the *continuous time* execution model the notion of a "single step transition probability" is replaced by the notion of "transition rate" which describes how probability is distributed in the time between state changes. Thus, transition probabilities are now functions of the time; in particular they are governed by an exponential distribution of parameter given by the specified rate. The syntax of the discrete time variant of pKLAIM is given on the left hand side of Table 3 while the continuous time variant is defined on the right hand side of Table 3. The difference between the network syntax of standard KLAIM is the *scheduling probability* q, which is a real number in the interval $[0, 1]$ used in the discrete time version of pKLAIM, and the *execution rate* λ, which is a positive real number determining the behaviour of the network for the continuous time execution model of pKLAIM.

$P ::= \mathbf{nil}$	null process		$a ::= \mathbf{out}(t)@\ell$	sending tuples
$\mid a.P$	action prefix		$\mid \mathbf{in}(t)@\ell$	receiving tuples
$\mid \vert_{i=1}^{n}\ p_i : P_i$	probabilistic parallelism		$\mid \mathbf{read}(t)@\ell$	inspecting tuples
$\mid +_{i=1}^{n}\ p_i : P_i$	probabilistic choice		$\mid \mathbf{eval}(P)@\ell$	remote evaluation
$\mid X$	process variable		$\mid \mathbf{newloc}(u)$	new location
$\mid A(P, \ell, e)$	process call			

$t ::= e$	expression tuple
$\mid P$	process tuple
$\mid \ell$	locality tuple
$\mid !x$	expression template
$\mid !u$	locality template
$\mid !X$	process template
$\mid t_1, t_2$	tuple

Table 2. Process Syntax

$N ::= s ::_{\varrho}^{q} P$	node		$N ::= s ::_{\varrho}^{\lambda} P$	node
$\mid N_1 \parallel N_2$	composition		$\mid N_1 \parallel N_2$	composition

Table 3. Network Syntax (Discrete and Continuous Time Variants)

In both the discrete and the continuous time execution model of pKLAIM we make use of *probabilistic allocation environments* ϱ. These allow us to associate logical localities with a probability distribution on the set of physical sites. A probabilistic allocation environment is formally defined as a partial map:

$$\varrho : Loc \to Dist(S),$$

where $Dist(S)$ is the set of all distributions on physical sites. We extend this definition to the set S of physical sites, by defining $\varrho(s)$ as the distribution which assigns probability 1 to s and 0 to all the other $s' \in S$. We denote by ϕ the probabilistic allocation environment which is undefined on all $l \in Loc$.

The stratification procedure, which allows to extend an inner allocation environment σ with an outer one ϱ (as in standard KLAIM), is defined for probabilistic allocation environments by:

$$(\sigma \bullet \varrho) = \begin{cases} \sigma \text{ iff } \sigma \neq \phi \\ \varrho \text{ otherwise} \end{cases}$$

As long as there is an inner allocation σ for ℓ we randomly identify ℓ with one of the possible sites according to the distribution $\sigma(\ell)$; only if $\sigma = \phi$ do we allocate ℓ according to $\varrho(\ell)$.

$$\mathbf{out}(t)@\ell.P \xrightarrow[\phi]{o(t)@\ell}_1 P \qquad \mathbf{in}(t)@\ell.P \xrightarrow[\phi]{i(t)@\ell}_1 P \qquad \mathbf{read}(t)@\ell.P \xrightarrow[\phi]{r(t)@\ell}_1 P$$

$$\mathbf{eval}(Q)@\ell.P \xrightarrow[\phi]{e(Q)@\ell}_1 P \qquad \mathbf{newloc}(u)@\ell.P \xrightarrow[\phi]{n(u)@\ell}_1 P$$

$$\frac{P_j \xrightarrow[\varrho]{\mu}_p P_j'}{+_{i=1}^n p_i : P_i \xrightarrow[\varrho]{\mu}_{p \cdot p_j} P_j'} \qquad \frac{P_j \xrightarrow[\varrho]{\mu}_p P_j'}{|_{i=1}^n p_i : P_i \xrightarrow[\varrho]{\mu}_{p \cdot p_j} |_{j \neq i=1}^n P_i \mid P_j'} \qquad \frac{P \xrightarrow[\varrho]{\mu}_p P'}{P\{\sigma\} \xrightarrow[\varrho \bullet \sigma]{\mu}_p P'\{\sigma\}}$$

$$\frac{P[Q/X, \ell/u, e/x] \xrightarrow[\varrho]{\mu}_p P'}{A(Q, \ell, e) \xrightarrow[\varrho]{\mu}_p P'} \quad \text{with } P \equiv A(X, u, x)$$

Table 4. The Local Structural Semantics

3 Operational Semantics of pKLAIM: Local Semantics

The operational semantics for pKLAIM is defined via probabilistic versions of the two levels of the operational semantics of KLAIM. The local transition relation

$$P_1 \xrightarrow[\varrho]{action}_p P_2$$

is the same for the discrete and continuous time versions of pKLAIM and is defined in Table 4. As in the original semantics for KLAIM, we use the label *action* to describe the activities performed in the evolution; thus, for example $o(t)@\ell$ refers to the action of sending the tuple t in the tuple space specified by ℓ, and $r(t)@\ell$ is the action of consuming the tuple t in the tuple space specified by ℓ. Following the conventions in [6] we use additional "book-keeping terms" — which are not part of the pKLAIM syntax — in the description of the operational semantics; for example we use "$P\{\rho\}$" in order to indicate the encapsulation of mobile processes.

Tuple evaluation is slightly different from standard KLAIM, as we have to take into account that allocation environments are probabilistic identifications of localities with sites. Each time we evaluate a locality ℓ we might obtain another physical site s. If $\langle s, p\rangle \in \varrho(\ell)$, we denote the probability p by $\varrho(\ell)(s)$.

The evaluation function for tuples is defined in Table 5, where $\mathcal{E}[\![e]\!]$ represents an evaluation mechanism for closed expressions. The matching of tuples (with templates) in Table 5 is defined exactly as in standard KLAIM. We will also write $X \simeq Y$ for $match(X, Y)$.

$\mathcal{T}[\![e]\!]_\varrho = \mathcal{E}[\![e]\!]$ $\mathcal{T}[\![P]\!]_\varrho = P\{\varrho\}$ $\mathcal{T}[\![\ell]\!]_\varrho = s \text{ with } p = \varrho(\ell)(s)$ $\mathcal{T}[\![!x]\!]_\varrho = !x$ $\mathcal{T}[\![!X]\!]_\varrho = !X$ $\mathcal{T}[\![!u]\!]_\varrho = !u$ $\mathcal{T}[\![t_1, t_2]\!]_\varrho = \mathcal{T}[\![t_1]\!]_\varrho, \mathcal{T}[\![t_2]\!]_\varrho$	$match(v, v) \quad match(P, P) \quad match(s, s)$ $match(!x, v) \quad match(!X, P) \quad match(!u, s)$ $\dfrac{match(et_1, et_2) \quad match(et_3, et_4)}{match((et_1, et_3), (et_2, et_4))}$ $\dfrac{match(et_1, et_2)}{match(et_2, et_1)}$

Table 5. Probabilistic Tuple Evaluation and Tuple Matching

4 Operational Semantics of pKLAIM: Discrete Time

For the discrete time execution model the update principle simulates a global scheduler: At every time step one node is selected according to the scheduling probabilities and executed. The result is a discrete time Markov chain (DTMC), cf. e.g. [15, 2].

In order to update a network (configuration) the global scheduler has to perform at each time step the following tasks:

1. select a node which could initiate a global update,according to the scheduling probabilities q_i,
2. check if the selected node can indeed cause a global transition (e.g. it can make an **out** transition or it has an **in** pending with a matching **out** somewhere),
3. execute one of the possible updates, as defined by the local semantics of the node in question.

If the first node selected is unable to perform a local transition an alternative one has to be selected. This means that the scheduler is effectively choosing only among the "active" nodes.

We will implement this scheduling principle by normalising the scheduling probabilities q_i; this is also needed to accommodate for dynamically created nodes (which inherit their scheduling probabilities from their ancestor). We will also normalise the probabilities p of active processes at a single node and the selection probabilities p_s we have encoded in the probabilistic allocation environments for **in** and **read** actions. These normalisation procedures, which we define below, guarantee that the probabilities of all global updates add up to one.

Based on the local semantics, we define the global semantics for discrete time pKLAIM in Table 6. We assume the usual structural equivalences as in KLAIM [6], so for example we have the identification:

$$s ::^q_\varrho P' \mid p : \mathbf{out}(et) \equiv s ::^q_\varrho P' \mid p : \mathbf{out}(et).\mathbf{nil}.$$

Moreover, we omit the probability associated to a process when it is 1.

In order to keep the presentation of the semantical rules as simple as possible, we will define the involved normalisation procedures — i.e. the "side computations" of $\tilde{p}$, etc. — outside the table. Furthermore, in the rules in Table 6 we omit the context of the entire network and only describe the transitions of at most two nodes involved in a global transition. This means that, for example, the first rule in Table 6 (for an n node network) should actually be read as:

$$\frac{P_n \xrightarrow[\sigma]{o(t)@\ell}_p P'_n \quad \langle s_n, p_{s_n}\rangle \in (\sigma \bullet \varrho)(\ell) \quad et = \mathcal{T}[\![t]\!]_{\sigma\bullet\varrho}}{N_{n-1} \parallel s_n ::^{q_n}_{\varrho_n} P_n \xRightarrow{\tilde{q}_n \cdot p_{s_n} \cdot \tilde{p}} N_{n-1} \parallel s_n ::^{q_n}_{\varrho_n} P'_n \mid \mathbf{out}(et)}$$

with $N_{n-1} \equiv s_1 ::^{q_1}_{\varrho_1} P_1 \parallel s_2 ::^{q_2}_{\varrho_2} P_2 \parallel \ldots \parallel s_{n-1} ::^{q_{n-1}}_{\varrho_{n-1}} P_{n-1}$, and that the normalisation of q_n involves all probabilities in N_{n-1}.

The probabilities attached to each global transitions in Table 6 denote the probability that this particular transition will be executed at a certain time step. Each of these involves typically a normalised scheduling probability $\tilde{q}$ — reflecting the fact that a particular node was selected by the scheduler — a normalised local probability $\tilde{p}$ — indicating the chances that a certain process at the selected node gets executed — and a selection probability p_s which stands for the probability that a logical locality is associated to a certain physical site.

For the global transitions initiated by **in** and **read** actions we need to normalise the selection probabilities $\tilde{p}_s$ as only those nodes can be selected which have indeed a data item, i.e. an $\mathbf{out}(t)$, available. This is not necessary for proper **out** and **eval** action as they will always succeed. For **in** and **read** actions we also need to determine the probabilities that — in case there are several possible data items available — a particular one is consumed or inspected by the **in** and **read** action respectively.

The normalisation of the local probabilities involves only either all the "competing" **out**s at a certain node or all the "competing" **in**s and **read**s. It is defined as $\tilde{p} = \frac{p}{P}$, where P is either the sum of all the probabilities associated to $o(t)@\mathbf{self}$ transitions or the sum of the probabilities attached to transitions on $i(t)@\ell$ or $r(t)@\ell$ actions at a particular node.

In the following we will often use the abbreviation $\mathbf{out}(t)$ for the action $\mathbf{out}(t)@\ell$, and the analogous abbreviations for **in**, **read** and **eval**. We denote by p_a the probability attached to a local transition where the action label corresponds to the prefix a (e.g. $p_{\mathbf{out}(t)} = p$ if there is a transition $\mathbf{out}(t)@\ell.P \xrightarrow[\phi]{o(t)@\ell}_p P$). Moreover, we say that an action a_1 *matches* an action a_2 if $a_1 = \mathbf{out}(t_1)$ and $a_2 = \mathbf{in}(t_2)$ or $a_2 = \mathbf{read}(t_2)$ and $match(et_1, et_2)$, or vice versa. In order to formalise these normalisation procedures we define **Top** to extract top-level actions:

$$\begin{aligned}
\mathbf{Top}(\mathbf{nil}) &= \emptyset \\
\mathbf{Top}(a.P) &= \{a\} \\
\mathbf{Top}(\,|_{i=1}^{n}\; p_i : P_i) &= \textstyle\bigcup_{i=1}^{n} \mathbf{Top}(P_i) \\
\mathbf{Top}(+_{i=1}^{n}\; p_i : P_i) &= \textstyle\bigcup_{i=1}^{n} \mathbf{Top}(P_i)
\end{aligned}$$

$$\frac{P \xrightarrow[\sigma]{o(t)@\ell}_{p} P' \quad \langle s, p_s\rangle \in (\sigma \bullet \varrho)(\ell) \quad et = \mathcal{T}[\![t]\!]_{\sigma\bullet\varrho}}{s ::^{q}_{\varrho} P \xRightarrow{\tilde{q}\cdot p_s\cdot\tilde{p}} s ::^{q}_{\varrho} P' \mid \mathbf{out}(et)}$$

$$\frac{P_1 \xrightarrow[\sigma_1]{o(t)@\ell}_{p} P_1' \quad \langle s_2, p_{s_2}\rangle \in (\sigma_1 \bullet \varrho_1)(\ell) \quad et = \mathcal{T}[\![t]\!]_{\sigma_1\bullet\varrho_1}}{s_1 ::^{q_1}_{\varrho_1} P_1 \parallel s_2 ::^{q_2}_{\varrho_2} P_2 \xRightarrow{\tilde{q}_1\cdot p_{s_2}\cdot\tilde{p}} s_1 ::^{q_1}_{\varrho_1} P_1' \parallel s_2 ::^{q_2}_{\varrho_2} P_2 \mid \mathbf{out}(et)}$$

$$\frac{P_1 \xrightarrow[\sigma]{i(t)@\ell}_{p_1} P_1' \quad \langle s, p_s\rangle \in (\sigma \bullet \varrho)(\ell) \quad P_2 \xrightarrow[\phi]{o(et)@\mathbf{self}}_{p_2} P_2' \quad et \simeq \mathcal{T}[\![t]\!]_{\sigma\bullet\varrho}}{s ::^{q}_{\varrho} P_1 \mid P_2 \xRightarrow{\tilde{q}\cdot\tilde{p}_1\cdot\tilde{p}_s\cdot\tilde{p}_2} s ::^{q}_{\varrho} P_1'[et/\mathcal{T}[\![t]\!]_{\sigma\bullet\varrho}] \mid P_2'}$$

$$\frac{P_1 \xrightarrow[\sigma_1]{i(t)@\ell}_{p_1} P_1' \quad \langle s_2, p_{s_2}\rangle \in (\sigma_1 \bullet \varrho_1)(\ell) \quad P_2 \xrightarrow[\phi]{o(et)@\mathbf{self}}_{p_2} P_2' \quad et \simeq \mathcal{T}[\![t]\!]_{\sigma_1\bullet\varrho_1}}{s_1 ::^{q_1}_{\varrho_1} P_1 \parallel s_2 ::^{q_2}_{\varrho_2} P_2 \xRightarrow{\tilde{q}_1\cdot\tilde{p}_1\cdot\tilde{p}_{s_2}\cdot\tilde{p}_2} s_1 ::^{q_1}_{\varrho_1} P_1'[et/\mathcal{T}[\![t]\!]_{\sigma\bullet\varrho}] \parallel s_2 ::^{q_2}_{\varrho_2} P_2'}$$

$$\frac{P_1 \xrightarrow[\sigma]{r(t)@\ell}_{p_1} P_1' \quad \langle s, p_s\rangle \in (\sigma \bullet \varrho)(\ell) \quad P_2 \xrightarrow[\phi]{o(et)@\mathbf{self}}_{p_2} P_2' \quad et \simeq \mathcal{T}[\![t]\!]_{\sigma\bullet\varrho}}{s ::^{q}_{\varrho} P_1 \mid P_2 \xRightarrow{\tilde{q}\cdot\tilde{p}_1\cdot\tilde{p}_s\cdot\tilde{p}_2} s ::^{q}_{\varrho} P_1'[et/\mathcal{T}[\![t]\!]_{\sigma\bullet\varrho}] \mid P_2}$$

$$\frac{P_1 \xrightarrow[\sigma_1]{r(t)@\ell}_{p_1} P_1' \quad \langle s_2, p_{s_2}\rangle \in (\sigma_1 \bullet \varrho_1)(\ell) \quad P_2 \xrightarrow[\phi]{o(et)@\mathbf{self}}_{p_2} P_2' \quad et \simeq \mathcal{T}[\![t]\!]_{\sigma_1\bullet\varrho_1}}{s_1 ::^{q_1}_{\varrho_1} P_1 \parallel s_2 ::^{q_2}_{\varrho_2} P_2 \xRightarrow{\tilde{q}_1\cdot\tilde{p}_1\cdot\tilde{p}_{s_2}\cdot\tilde{p}_2} s_1 ::^{q_1}_{\varrho_1} P_1'[et/\mathcal{T}[\![t]\!]_{\sigma\bullet\varrho}] \parallel s_2 ::^{q_2}_{\varrho_2} P_2}$$

$$\frac{P \xrightarrow[\sigma]{e(Q)@\ell}_{p} P' \quad \langle s, p_s\rangle \in (\sigma \bullet \varrho)(\ell)}{s ::^{q}_{\varrho} P \xRightarrow{\tilde{q}\cdot p_s\cdot\tilde{p}} s ::^{q}_{\varrho} P' \mid Q}$$

$$\frac{P_1 \xrightarrow[\sigma_1]{e(Q)@\ell}_{p} P_1' \quad \langle s_2, p_{s_2}\rangle \in (\sigma_1 \bullet \varrho_1)(\ell)}{s_1 ::^{q_1}_{\varrho_1} P_1 \parallel s_2 ::^{q_2}_{\varrho_2} P_2 \xRightarrow{\tilde{q}_1\cdot p_{s_2}\cdot\tilde{p}} s_1 ::^{q_1}_{\varrho_1} P_1' \parallel s_2 ::^{q_2}_{\varrho_2} P_2 \mid Q}$$

$$\frac{P \xrightarrow[\sigma_1]{n(u)@\mathbf{self}}_{p} P' \quad s' \in S \quad s \neq s'}{s ::^{q}_{\varrho} P \xRightarrow{\tilde{q}\cdot\tilde{p}} s ::^{q}_{\varrho} P'[s'/u] \parallel s' ::^{q}_{[s'/\mathbf{self}]\bullet\varrho} \mathbf{nil}}$$

$$\frac{s ::^{q}_{\varrho} P_1 \xRightarrow{p} s ::_{\varrho} P_1'}{s ::^{q}_{\varrho} P_1 \mid P_2 \xRightarrow{\tilde{q}\cdot\tilde{p}} s ::^{q}_{\varrho} P_1' \mid P_2}$$

Table 6. The Global Structural Discrete Time Semantics

We next define the *active* **in**/**read**s and **out**s:

$$\begin{aligned}
&\mathbf{Act}_{out}(s_1 :: P_1) = \\
&= \{\mathbf{out}(t) \mid \mathbf{out}(t)@\ell \in \mathbf{Top}(P_1)\}
\end{aligned}$$

$$\begin{aligned}
&\mathbf{Act}_{in}(s_1 :: P_1, s_2 :: P_2) = \\
&= \{\mathbf{in}(t) \mid \mathbf{in}(t) \in \mathbf{Top}(P_1) \text{ and there exists} \\
&\quad \mathbf{out}(t') \in \mathbf{Top}(P_2) \text{ such that } \mathbf{out}(t') \text{ matches } \mathbf{in}(t)\} \cup \\
&\quad \{\mathbf{read}(t) \mid \mathbf{read}(t) \in \mathbf{Top}(P_1) \text{ and there exists} \\
&\quad \mathbf{out}(t') \in \mathbf{Top}(P_2) \text{ such that } \mathbf{out}(t') \text{ matches } \mathbf{read}(t)\},
\end{aligned}$$

and then compute

$$P_o = \sum_{a \in \mathbf{Act}_{out}(s_1::P_1)} p_a,$$

so that for the active **out**s we get the normalised probabilities as

$$\tilde{p}_i = \frac{p_i}{P_o},$$

and

$$P_i = \sum_{a \in \mathbf{Act}_{in}(s_1::P_1, s_2::P_2)} p_a,$$

so that for the active **in**s we get the normalised probabilities as

$$\tilde{p}_j = \frac{p_j}{P_i}.$$

A normalisation is needed also for the probability of the selected site. Given a site $s_1 ::_{\varrho_1} P_1$, suppose that $\mathbf{in}(t)@\ell \in \mathbf{Top}(P_1)$. Then a site s_2 is *active* (willing to communicate) if there is a matching **out** action possible at s_2. We define

$$P_s = \sum (\sigma_1 \bullet \varrho_1)(l)(s_2)$$

where the sum is over all selection probabilities of nodes which are willing to communicate, and σ_1 is the environment of the local transition of P, and then we normalise:

$$\tilde{p}_{s_2} = \frac{p_{s_2}}{P_s}.$$

Example 1. Consider the two node network:

$$N \equiv l_1 ::_{\phi}^{\frac{1}{4}} P_1 \parallel l_2 ::_{\phi}^{\frac{3}{4}} P_2$$

with

$$\begin{aligned}
P_1 &\equiv \mathbf{out}(t)@l_1.P_1 \\
P_2 &\equiv \frac{1}{3} : \mathbf{out}(t_1)@l_2.P_2 + \frac{2}{3} : \mathbf{out}(t_2)@l_2.P_2.
\end{aligned}$$

If we look at the possible global transitions $N \overset{p}{\rightarrowtail\!\!\Rightarrow} N'$ in one time unit/step we see that there are two cases: The network update can be initiated by node $n_1 \equiv l_1 ::_{\phi}^{\frac{1}{4}} P_1$ or by node $n_2 \equiv l_2 ::_{\phi}^{\frac{3}{4}} P_2$. The chances that each of these updates is happening is $\frac{1}{4}$ and $\frac{3}{4}$, respectively.

In the case the update is initiated the by n_1 there is only one possible global transition, namely:

$$l_1 ::_{\phi}^{\frac{1}{4}} P_1 \parallel l_2 ::_{\phi}^{\frac{3}{4}} P_2 \overset{\frac{1}{4}}{\rightarrowtail\!\!\Rightarrow} l_1 ::_{\phi}^{\frac{1}{4}} P_1 \mid \mathbf{out}(t) \parallel l_2 ::_{\phi}^{\frac{3}{4}} P_2$$

For a global transition initiated by n_2 we get two possibilities, each of which have probabilities $\frac{3}{4}\frac{1}{3} = \frac{1}{4}$ and $\frac{3}{4}\frac{2}{3} = \frac{1}{2}$, respectively:

$$l_1 ::_{\phi}^{\frac{1}{4}} P_1 \parallel l_2 ::_{\phi}^{\frac{3}{4}} P_2 \overset{\frac{1}{4}}{\rightarrowtail\!\!\Rightarrow} l_1 ::_{\phi}^{\frac{1}{4}} P_1 \parallel l_2 ::_{\phi}^{\frac{3}{4}} P_2 \mid \mathbf{out}(t_1)$$

$$l_1 ::_{\phi}^{\frac{1}{4}} P_1 \parallel l_2 ::_{\phi}^{\frac{3}{4}} P_2 \overset{\frac{1}{2}}{\rightarrowtail\!\!\Rightarrow} l_1 ::_{\phi}^{\frac{1}{4}} P_1 \parallel l_2 ::_{\phi}^{\frac{3}{4}} P_2 \mid \mathbf{out}(t_2)$$

In other words the one step global transitions $N \overset{p}{\rightarrowtail\!\!\Rightarrow} N'$ lead to a network in which either n_1 or n_2 is changed.

5 Operational Semantics of pKLAIM: Continuous Time

The continuous time semantics of pKLAIM relies on the idea that state changes (transitions) do not occur at regularly spaced intervals like in the previously defined discrete time semantics; instead the time between state transitions is exponentially distributed. The resulting model is therefore a continuous time Markov chain (CTMC), cf. e.g. [17, 15, 2].

In the discrete case, the global scheduler determines which node (according to the global scheduling priorities) can initiate the change. In the continuous case each node can initiate a network update independently at any moment in time with a certain probability.

In this paper we will consider a very basic mechanism describing the chance of a certain node initiating an update. We assume that each node "fires", i.e. initiates an update, via a so called Poisson process, where the chances are governed by an exponential distribution. Such processes can be specified by a single real parameter, the *rate*, which essentially describes a probability over the time between two updates (see e.g. [15, Sect 2.4]). This parameter is specified by the superscript λ in the syntax of a node.

The global semantics for the continuous time pKLAIM is defined in Table 7. The transition relation is now labelled by a probability $p(t)$ which is a continuous function of the time t.

The following examples illustrate how the local probabilities, in a choice or a local parallel construct and the global update probabilities interact with each other.

$$\frac{P \xrightarrow[\sigma]{o(t)@\ell}_p P' \quad \langle s, p_s\rangle \in (\sigma \bullet \varrho)(\ell) \quad et = \mathcal{T}[\![t]\!]_{\sigma\bullet\varrho}}{s ::_\varrho^\lambda P \xRightarrow{p_s \cdot \tilde{p} \cdot e^{-\lambda t}} s ::_\varrho^\lambda P' \mid \mathbf{out}(et)}$$

$$\frac{P_1 \xrightarrow[\sigma_1]{o(t)@\ell}_p P_1' \quad \langle s_2, p_{s_2}\rangle \in (\sigma_1 \bullet \varrho_1)(\ell) \quad et = \mathcal{T}[\![t]\!]_{\sigma_1\bullet\varrho_1}}{s_1 ::_{\varrho_1}^{\lambda_1} P_1 \parallel s_2 ::_{\varrho_2}^{\lambda_2} P_2 \xRightarrow{p_{s_2} \cdot \tilde{p} \cdot e^{-\lambda_1 t}} s_1 ::_{\varrho_1}^{\lambda_1} P_1' \parallel s_2 ::_{\varrho_2}^{\lambda_2} P_2 \mid \mathbf{out}(et)}$$

$$\frac{P_1 \xrightarrow[\sigma]{i(t)@\ell}_{p_1} P_1' \quad \langle s, p_s\rangle \in (\sigma \bullet \varrho)(\ell) \quad P_2 \xrightarrow[\phi]{o(et)@\mathbf{self}}_{p_2} P_2' \quad et \simeq \mathcal{T}[\![t]\!]_{\sigma\bullet\varrho}}{s ::_\varrho^\lambda P_1 \mid P_2 \xRightarrow{\tilde{p}_1 \cdot \tilde{p}_s \cdot \tilde{p}_2 \cdot e^{-\lambda t}} s ::_\varrho^\lambda P_1'[et/\mathcal{T}[\![t]\!]_{\sigma\bullet\varrho}] \mid P_2'}$$

$$\frac{P_1 \xrightarrow[\sigma_1]{i(t)@\ell}_{p_1} P_1' \quad \langle s_2, p_{s_2}\rangle \in (\sigma_1 \bullet \varrho_1)(\ell) \quad P_2 \xrightarrow[\phi]{o(et)@\mathbf{self}}_{p_2} P_2' \quad et \simeq \mathcal{T}[\![t]\!]_{\sigma_1\bullet\varrho_1}}{s_1 ::_{\varrho_1}^{\lambda_1} P_1 \parallel s_2 ::_{\varrho_2}^{\lambda_2} P_2 \xRightarrow{\tilde{p}_1 \cdot \tilde{p}_{s_2} \cdot \tilde{p}_2 \cdot e^{-\lambda_1 t}} s_1 ::_{\varrho_1}^{\lambda_1} P_1'[et/\mathcal{T}[\![t]\!]_{\sigma\bullet\varrho}] \parallel s_2 ::_{\varrho_2}^{\lambda_2} P_2'}$$

$$\frac{P_1 \xrightarrow[\sigma]{r(t)@\ell}_{p_1} P_1' \quad \langle s, p_s\rangle \in (\sigma \bullet \varrho)(\ell) \quad P_2 \xrightarrow[\phi]{o(et)@\mathbf{self}}_{p_2} P_2' \quad et \simeq \mathcal{T}[\![t]\!]_{\sigma\bullet\varrho}}{s ::_\varrho^\lambda P_1 \mid P_2 \xRightarrow{\tilde{p}_1 \cdot \tilde{p}_s \cdot \tilde{p}_2 \cdot e^{-\lambda t}} s ::_\varrho^\lambda P_1'[et/\mathcal{T}[\![t]\!]_{\sigma\bullet\varrho}] \mid P_2}$$

$$\frac{P_1 \xrightarrow[\sigma_1]{r(t)@\ell}_{p_1} P_1' \quad \langle s_2, p_{s_2}\rangle \in (\sigma_1 \bullet \varrho_1)(\ell) \quad P_2 \xrightarrow[\phi]{o(et)@\mathbf{self}}_{p_2} P_2' \quad et \simeq \mathcal{T}[\![t]\!]_{\sigma_1\bullet\varrho_1}}{s_1 ::_{\varrho_1}^{\lambda_1} P_1 \parallel s_2 ::_{\varrho_2}^{\lambda_2} P_2 \xRightarrow{\tilde{p}_1 \cdot \tilde{p}_{s_2} \cdot \tilde{p}_2 \cdot e^{-\lambda_1 t}} s_1 ::_{\varrho_1}^{\lambda_1} P_1'[et/\mathcal{T}[\![t]\!]_{\sigma\bullet\varrho}] \parallel s_2 ::_{\varrho_2}^{\lambda_2} P_2}$$

$$\frac{P \xrightarrow[\sigma]{e(Q)@\ell}_p P' \quad \langle s, p_s\rangle \in (\sigma \bullet \varrho)(\ell)}{s ::_\varrho^\lambda P \xRightarrow{p_s \cdot \tilde{p} \cdot e^{-\lambda t}} s ::_\varrho^\lambda P' \mid Q}$$

$$\frac{P_1 \xrightarrow[\sigma_1]{e(Q)@\ell}_p P_1' \quad \langle s_2, p_{s_2}\rangle \in (\sigma_1 \bullet \varrho_1)(\ell)}{s_1 ::_{\varrho_1}^{\lambda_1} P_1 \parallel s_2 ::_{\varrho_2}^{\lambda_2} P_2 \xRightarrow{p_{s_2} \cdot \tilde{p} \cdot e^{-\lambda_1 t}} s_1 ::_{\varrho_1}^{\lambda_1} P_1' \parallel s_2 ::_{\varrho_2}^{\lambda_2} P_2 \mid Q}$$

$$\frac{P \xrightarrow[\sigma_1]{n(u)@\mathbf{self}}_p P' \quad s' \in S \quad s \neq s'}{s ::_\varrho^\lambda P \xRightarrow{\tilde{p} \cdot e^{-\lambda t}} s ::_\varrho^\lambda P'[s'/u] \parallel s' ::_{[s'/\mathbf{self}]\bullet\varrho}^\lambda \mathbf{nil}}$$

$$\frac{s ::_\varrho^\lambda P_1 \xRightarrow{p} s ::_\varrho^\lambda P_1'}{s ::_\varrho^\lambda P_1 \mid P_2 \xRightarrow{\tilde{p} \cdot e^{-\lambda t}} s ::_\varrho^\lambda P_1' \mid P_2}$$

Table 7. The Global Continuous Time Structural Semantics

Example 2. Consider the following network which is the same as the one in Example 1 but for the scheduling probabilities which are replaced by rates:

$$N \equiv l_1 ::_\phi^1 P_1 \parallel l_2 ::_\phi^2 P_2$$

with

$$P_1 \equiv \mathbf{out}(t)@l_1.P_1$$
$$P_2 \equiv \frac{1}{3} : \mathbf{out}(t_1)@l_2.P_2 + \frac{2}{3} : \mathbf{out}(t_2)@l_2.P_2$$

In a similar way as in Example 1 we can look at the possible transitions of N in a single time unit, i.e. the probabilities for $N \overset{p(1)}{\rightarrowtail\!\!\!\Longrightarrow} N'$.

Again the global transitions of N can be initiated by two events: both $n_1 \equiv l_1 ::_\phi^1 P_1$ and $n_2 \equiv l_2 ::_\phi^2 P_2$ can initiate a global update. The difference with Example 1 is the fact that the statement that either of the two happens in one transition step does no longer make sense in the continuous time setting. Instead we observe that each of them, both together or none of them might happen in one time unit. In order to express this situation we introduce the notation $P^n = P \mid P \mid \ldots \mid P$ to stand for the n-fold parallel composition of copies of P, and define

$$P_{1k} \equiv P_1 \mid \mathbf{out}(t)^k$$
$$P_{2nm} \equiv P_2 \mid \mathbf{out}(t_1)^n \mid \mathbf{out}(t_2)^m.$$

Among the possible transitions we have the following:

$$l_1 ::_\phi^1 P_1 \parallel l_2 ::_\phi^2 P_2 \overset{\exp(-1)}{\rightarrowtail\!\!\!\Longrightarrow} l_1 ::_\phi^1 P_{1,1} \parallel l_2 ::_\phi^2 P_{2,0,0}$$
$$l_1 ::_\phi^1 P_1 \parallel l_2 ::_\phi^2 P_2 \overset{\exp(-1)\frac{1}{3}}{\rightarrowtail\!\!\!\Longrightarrow} l_1 ::_\phi^1 P_{1,0} \parallel l_2 ::_\phi^2 P_{2,1,0}$$
$$l_1 ::_\phi^1 P_1 \parallel l_2 ::_\phi^2 P_2 \overset{\exp(-2)\frac{2}{3}}{\rightarrowtail\!\!\!\Longrightarrow} l_1 ::_\phi^1 P_{1,0} \parallel l_2 ::_\phi^2 P_{2,0,1}$$

Note that the cases on the right hand side of these transitions are not mutually exclusive and that thus the probabilities do not sum up to one. Moreover, contrary to the discrete case, a number of other transitions are also possible since each of the two processes can be executed any number i of times in an interval of length t. From the theory of Poisson processes the following formula expresses the probability $p_i(t)$ that exactly i events occur in an interval of length t:

$$p_i(t) = \frac{(\lambda t)^i e^{-\lambda t}}{i!},$$

where λ is is the parameter of the Poisson distribution, e.g. [17, Thm 1.2.1].

We can then calculate the probability p of transition

$$l_1 ::_\phi^1 P_1 \parallel l_2 ::_\phi^2 P_2 \overset{p(1)}{\rightarrowtail\!\!\!\Longrightarrow} l_1 ::_\phi^1 P_{1,n} \parallel l_2 ::_\phi^2 P_{2,0,0}$$

by simply instantiating the above formula with $\lambda = 1$, $i = n$ and $t = 1$: $p(1) = \frac{e^{-1}}{n!}$. Analogously, we get the transition:

$$l_1 ::_\phi^1 P_1 \parallel l_2 ::_\phi^2 P_2 \overset{p(1)}{\rightarrowtail\!\!\Longrightarrow} l_1 ::_\phi^1 P_{1,0} \parallel l_2 ::_\phi^2 P_{2,n,m}$$

with probability $p(1) = \frac{2^k e^{-2}}{k!}$, where $k = n + m$, and the transition:

$$l_1 ::_\phi^1 P_1 \parallel l_2 ::_\phi^2 P_2 \overset{p(1)}{\rightarrowtail\!\!\Longrightarrow} l_1 ::_\phi^1 P_{1,s} \parallel l_2 ::_\phi^2 P_{2,n,m}$$

with probability $p(1) = \frac{e^{-1}}{s!} \cdot \frac{2^k e^{-2}}{k!}$, where $k = n + m$.

Comparing Example 1 and Example 2 we see that in the latter case we obtain a much richer set of possible transitions in one time unit. In the continuous case a single transition might happen in a time unit, none at all or more than one, while in the discrete case exactly one transition happens (provided that no deadlock occurs).

Furthermore, the continuous time model realises true concurrency as several transitions *seem* to happen in "parallel". In fact, two transitions are actually never happening at exactly the same moment, as the probability for this is zero. However, after a single time unit we can observe that two or more transitions have happened.

This allows us to avoid considering "clashes" like for example two $\mathbf{in}(t)$ actions trying to access the same token: the probability of this happening vanishes. We can however ask for the probability that either of the two **in**'s is executed first and in this way determine the chances that the token in question has been consumed by the first or the second **in** after a given time (or, as also could be the case, that neither of them has already consumed the token).

6 Example: Computer Viruses

We now present a more sophisticated example, motivated by the security literature. Computer viruses and worms have been very much in the news recently with the SoBig.F virus and the Blaster worm both making the headlines over the summer (of 2003). We have recently been looking at how probabilistic program analysis techniques can be used to predict the rate of spread of such viruses [7]. The techniques used in *op. cit.* were applied to a much smaller language than that treated here and they are beyond the scope of this paper. However, we will give a flavour of how such phenomena may be modelled in pKLAIM.

6.1 pKLAIM Program

Consider the following program implementing a "computer virus":

$$V \equiv \mathbf{out}(V)@\ell.V.\mathbf{nil}$$

which transmits itself to another location ℓ and executes some "malicious code" V.

At each node in the network we assume that there is an "operating system" running which either accepts any process it receives and executes it:

$$O \equiv \mathbf{in}(!P)@\mathbf{self}.\mathit{Seq}(P,O).\mathbf{nil}$$

i.e. an open system, or ignores processes it receives:

$$C \equiv \mathbf{in}(!P)@\mathbf{self}.C$$

i.e. a closed system. Note that O and C actually do nothing else but recursively receiving processes. The *Seq* process is defined as follows:

$$\mathit{Seq}(A,B) \equiv \mathbf{eval}(A)@\mathbf{self}.\mathbf{eval}(B)@\mathbf{self}.\mathbf{nil}$$

6.2 Discrete Time

Consider the following set of ('top', 'middle' and 'bottom') sites:

$$S = \{t_1, \dots, t_9, m_1, m_2, m_3, b_1, \dots, b_9\}$$

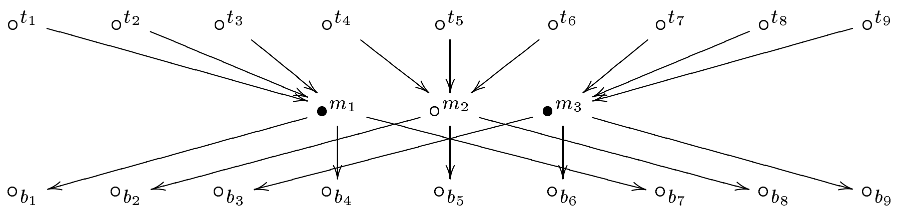
and define the following (probabilistic) allocation environments:

$$\tau = \left\{ \begin{array}{l} \langle [\ell/t_1], 1/12\rangle, \dots, \langle [\ell/t_9], 1/12\rangle, \\ \langle [\ell/m_1], 1/12\rangle, \langle [\ell/m_2], 1/12\rangle, \langle [\ell/m_3], 1/12\rangle \end{array} \right\}$$

$$\mu = \left\{ \begin{array}{l} \langle [\ell/m_1], 1/12\rangle, \langle [\ell/m_2], 1/12\rangle, \langle [\ell/m_3], 1/12\rangle, \\ \langle [\ell/b_1], 1/12\rangle, \dots, \langle [\ell/b_9], 1/12\rangle \end{array} \right\}$$

$$\lambda = \{\langle [\ell/b_1], 1/9\rangle, \dots, \langle [\ell/b_9], 1/9\rangle\}$$

Then a network which can informally be described by the following diagram:



is formally specified by the following pKLAIM specification:

$$\begin{array}{c} t_1 ::_{\tau}^{\frac{1}{21}} O \parallel \dots \parallel t_9 ::_{\tau}^{\frac{1}{21}} O \\ \parallel m_1 ::_{\mu}^{\frac{1}{21}} C \parallel m_2 ::_{\mu}^{\frac{1}{21}} O \parallel m_3 ::_{\mu}^{\frac{1}{21}} C \parallel \\ b_1 ::_{\lambda}^{\frac{1}{21}} O \parallel \dots \parallel b_9 ::_{\lambda}^{\frac{1}{21}} O \end{array}$$

We have assumed a uniform distribution — each node has probability $\frac{1}{21}$ associated with it. In the following we will omitted these probabilities on nodes.

Note that in the formal specification the nodes in each layer are fully connected and that each node on the top level is connected with all nodes in the middle layer (not all of these connections had been depicted in the above diagram). The idea is that at site m_1 and m_3 virus resistant processes are situated while all other nodes are vulnerable to “infections”. Now assume that one top node (e.g. t_1) gets infected by the virus, i.e. consider the execution of:

$$\begin{array}{c} t_1 ::_\tau V \parallel \ldots \parallel t_9 ::_\tau O \\ \parallel m_1 ::_\mu C \parallel m_2 ::_\mu O \parallel m_3 ::_\mu C \parallel \\ b_1 ::_\lambda O \parallel \ldots \parallel b_9 ::_\lambda O \end{array}$$

Then there are two prototypical changes possible: Either V spreads to an open node, like m_2, or to a closed one, like m_1. The probability for each of the neighbours that this might happen is $\frac{1}{12}$. Whilst in the former case the virus gets activated, in the later case the virus stays untouched and encapsulated.

$$\begin{array}{c} t_1 ::_\tau V \parallel \ldots \parallel t_9 ::_\tau O \\ \parallel m_1 ::_\mu C \parallel m_2 ::_\mu O \parallel m_3 ::_\mu C \parallel \\ b_1 ::_\lambda O \parallel \ldots \parallel b_9 ::_\lambda O \\ \xRightarrow{1/12} \\ t_1 ::_\tau V \parallel \ldots \parallel t_9 ::_\tau O \\ \parallel m_1 ::_\mu C \parallel m_2 ::_\mu O \mid \mathbf{out}(V) \parallel m_3 ::_\mu C \parallel \\ b_1 ::_\lambda O \parallel \ldots \parallel b_9 ::_\lambda O \\ \xRightarrow{1} \\ t_1 ::_\tau V \parallel \ldots \parallel t_9 ::_\tau O \\ \parallel m_1 ::_\mu C \parallel m_2 ::_\mu V.O \parallel m_3 ::_\mu C \parallel \\ b_1 ::_\lambda O \parallel \ldots \parallel b_9 ::_\lambda O \end{array}$$

and

$$\begin{array}{c} t_1 ::_\tau V \parallel \ldots \parallel t_9 ::_\tau O \\ \parallel m_1 ::_\mu C \parallel m_2 ::_\mu O \parallel m_3 ::_\mu C \parallel \\ b_1 ::_\lambda O \parallel \ldots \parallel b_9 ::_\lambda O \\ \xRightarrow{1/12} \\ t_1 ::_\tau V \parallel \ldots \parallel t_9 ::_\tau O \\ \parallel m_1 ::_\mu C \mid \mathbf{out}(V) \parallel m_2 ::_\mu O \parallel m_3 ::_\mu C \parallel \\ b_1 ::_\lambda O \parallel \ldots \parallel b_9 ::_\lambda O \end{array}$$

Analysing all possible network transitions together with their probabilities allows us to answer questions like: What is the chance that a certain node is infected after n steps? In other words, we can *formally* study and analyse the patterns of how a virus might spread through a network with a certain topology; in our simple example above we can determine, for example, how the number of closed nodes in the middle layer slows down the spread of a virus from the top to the bottom layer.

6.3 Continuous Time and Analysis

The continuous time version of the network is based upon the same process definitions. The only difference in the formal description is that probabilities associated with nodes are replaced by rates. We can then answer questions about the probability that a node becomes infected at a certain point given some initial configuration. As we have observed above, the continuous time model captures some aspects of true concurrency and, therefore in some senses, gives a more accurate model of virus/worm propagation in a network.

The preceding subsections constitute a work programme rather than a catalogue of results. Ideally we would like to produce automatic tools that could assist systems administrators in assessing the vulnerability of their networks to such infections. The approach we advocate in [7] is to model the network by a Markov process and then to use linear algebraic techniques to "abstract" the network by an *infection matrix* which enables us to study the propagation of viruses. The work we have done to date has been focused on discrete time models for a first-order restriction of pKLAIM. The analysis of full pKLAIM and the ultimate production of automated tools remains for future work.

7 Conclusions

We have presented an approach to introducing probabilities into coordination languages. Our proposals has been presented in the context of the KLAIM language where we introduced probabilities both at the local (or process) level and at the network level. The natural role of probabilities at the process level is in scheduling parallel threads and deciding choice operators. We have proposed two alternative ways of introducing probabilities at the network level: discrete time and continuous time. In the former case, we have used probabilities to schedule network parallelism. In the latter case we use rates to determine how often a node is active. We have illustrated the difference between these two approaches with a simple example and also sketched a more complex example of virus propagation. We have shown that the continuous time model captures some aspects of true concurrency.

The probabilistic version(s) of KLAIM we have introduced in this paper are closely related to various *probabilistic programming languages* and *probabilistic process calculi* proposed in the recent literature. Among these one can find discrete time approaches — e.g. PCCS [10, 13], PCCP [9], etc. — as well as continuous time approaches — e.g. PEPA [11], Stochastic π calculus [16], to name just a few. Work in performance analysis is often based on probabilistic process calculi, for example, on Hillston's PEPA [12], or EMPA by Bernardo and Gorrieri [4]. One of the long term aims of the work presented in this paper is the development of semantics based approaches towards *performance analysis* along similar lines as in classical program analysis. We also aim to investigate more closely the relation of our work to recent work on probabilistic verification and model checking, such as PRISM [14] and de Alfaro [5]. The virus example discussed in Section 6 is part of a more general probabilistic approach towards *security analysis* which attempts to investigate more "realistic" measures for confidentiality, security, etc., see e.g. [8, 1].

In our continuous time model we only consider the simplest time processes, the so called Poisson processes. More complicated continuous time behaviour could be considered, but this might require more parameters than just rate to describe the time distributions. The language could also be extended to allow for dynamic change of probabilities and rates, i.e. for rate and probability variables. This would require some care in the choice of range of e.g. p; taking all the reals might lead to paradoxes regarding computability. These last two extensions require further work as does a detailed analysis of the virus example presented in Section 6.

References

1. A. Aldini, M. Bravetti, and R. Gorrieri. A process algebraic approach for the analysis of probabilistic non-inetrference. *Journal of Computer Security*, 2004.
2. Falko Bause and Pieter S. Kritzinger. *Stochastic Petri Nets – An Introduction to the Theory*. Vieweg Verlag, second edition, 2002.
3. J.A. Bergstra, A. Ponse, and S.A. Smolka, editors. *Handbook of Process Algebra*. Elsevier Science, Amsterdam, 2001.
4. M. Bernardo and R. Gorrieri. A tutorial on empa: A theory of concurrent processes with nondeterminism, priorities, probabilities and time. Technical Report UBLCS-96-17, Department of Computer Science, University of Bologna, 1997.
5. L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, Department of Computer Science, 1998.
6. R. De Nicola, G.L. Ferrari, and R. Pugliese. KLAIM: A kernel language for agents interaction and mobility. *IEEE Transactions on Software Engineering*, 24(5):315–330, 1998.
7. A. Di Pierro, C. Hankin, and H. Wiklicky. Analysing the propagation of computer viruses. *Journal of Functional Programming*, 2003. (submitted).
8. A. Di Pierro, C. Hankin, and H. Wiklicky. Approximate Non-Interference. *Journal of Computer Security*, 12(1):37–81, 2004.
9. A. Di Pierro and H. Wiklicky. Quantitative observables and averages in Probabilistic Concurrent Constraint Programming. In *New Trends in Constraints*, number 1865 in Lecture Notes in Computer Science. Springer Verlag, 2000.
10. A. Giacalone, C.-C. Jou, and S.A. Smolka. Algebraic reasoning for probabilistic concurrent systems. In *Proceedings of the IFIP WG 2.2/2.3 Working Conference on Programming Concepts and Methods*, pages 443–458, Sea of Gallilee, April 1990. North-Holland.
11. J. Hillston. PEPA: Performance enhanced process algebra. Technical Report CSR-24-93, University of Edinburgh, Edinburgh, Scotland, 1993.
12. J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.
13. B. Jonsson, W. Yi, and K.G. Larsen. *Probabilistic Extentions of Process Algebras*, chapter 11, pages 685–710. Elsevier Science, Amsterdam, 2001. see [3].
14. M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. In J.-P. Katoen and P. Stevens, editors, *Proceedings of TACAS'02*, volume 2280 of *Lecture Notes in Computer Science*, pages 52–66. Springer Verlag, 2002.

15. J.R. Norris. *Markov Chains.* Cambidge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, 1997.
16. C. Priami. Stochastic π-calculus. *Computer Journal*, 38(7):578–589, 1995.
17. Henk C. Tijms. *Stochastic Models – An Algorithmic Approach.* John Wiley & Sons, Chichester, 1994.