Continuous-Time Probabilistic KLAIM

Alessandra Di Pierro [1]

Dipartimento di Informatica, University of Pisa, Italy

Chris Hankin [1]

Department of Computing, Imperial College London, United Kingdom

Herbert Wiklicky [1]

Department of Computing, Imperial College London, United Kingdom

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 [5] 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. In a security setting these issues are relevant e.g. for the analysis and prevention of denial of service attacks, which involve the delaying of time-critical operations [9].

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

[1] All three authors are partly funded by the EPSRC project S77066A "Quantitative Analysis of Computational Resources".

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

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

Table 1
Syntactic Categories

allocation environments which associate distributions on physical sites with logical localities. Furthermore, we associates a rate with each node; this determines how often a node is active. An alternative would be to associate a probability with each node, indicating the chance that a process at that node will be selected for execution. We have studied such a variant in detail in an earlier paper [6].

2 Syntax of pKLAIM

The syntax of pKLAIM is based essentially on the one given for KLAIM originally in [5] and refers to the syntactic categories in Table 1.

The process terms in pKLAIM are formed according to the rules in Table 2. The main difference with the original KLAIM language is the probabilistic weighting of choice and local parallelism.

At the network level transition probabilities are functions of the time; in particular they are governed by an exponential distribution of parameter given by some specified rate. The syntax is defined in Table 4. The difference between the network syntax of standard KLAIM is the *execution rate* λ, which is a positive real number determining the behaviour of the network according to a continuous time Markov chain model. We will formalise this model in Section 3.2, where we define an operational semantics for pKLAIM.

We make use of *probabilistic allocation environments* which 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 $l \in Loc$.

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

Table 2
Process Syntax

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

Table 3
KLAIM Tuples

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

Table 4
Network Syntax

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 $\varrho(\ell)$.

3 Operational Semantics of pKLAIM

The operational semantics for pKLAIM is defined via probabilistic versions of the two levels of the operational semantics of KLAIM. At the local level processes on each node behave essentially as discrete time Markov chains, while the global network updating is modelled as a chain which evolves according to transition probabilities determined by the rates on the nodes. Thus, at the global level each node behaves as a Poisson process with rate λ. This defines the operational semantics of pKLAIM according to the common view of a network as a system which is locally synchronous and globally asynchronous.

3.1 Local Semantics

The local transition relation

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

is defined in Table 5. 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 [5] 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 6, where $\mathcal{E}[\![e]\!]$ represents an evaluation mechanism for closed expressions. The matching of tuples (with templates) in Table 7 is defined exactly as in standard KLAIM. We will also write $X \simeq Y$ for $match(X, Y)$.

3.2 Global Semantics

We will define a continuous time semantics for pKLAIM which relies on the idea that state changes (transitions) do not occur at a regular space (like in

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

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

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

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

Table 5
The Local Structural Semantics

$$\begin{aligned} \mathcal{T}[\![e]\!]_\varrho &= \mathcal{E}[\![e]\!] & \mathcal{T}[\![!x]\!]_\varrho &= !x \\ \mathcal{T}[\![P]\!]_\varrho &= P\{\varrho\} & \mathcal{T}[\![!X]\!]_\varrho &= !X \\ \mathcal{T}[\![\ell]\!]_\varrho &= s \text{ with } p = \varrho(\ell)(s) & \mathcal{T}[\![!u]\!]_\varrho &= !u \\ \multicolumn{4}{c}{\mathcal{T}[\![t_1, t_2]\!]_\varrho = \mathcal{T}[\![t_1]\!]_\varrho, \mathcal{T}[\![t_2]\!]_\varrho} \end{aligned}$$

Table 6
Probabilistic Tuple Evaluation

$$match(v,v) \quad match(P,P) \quad match(s,s)$$

$$match(!x,v) \quad match(!X,P) \quad match(!u,s)$$

$$\frac{match(et_1, et_2) \quad match(et_3, et_4)}{match((et_1, et_3), (et_2, et_4))}$$

$$\frac{match(et_1, et_2)}{match(et_2, et_1)}$$

Table 7
Probabilistic Tuple Matching

a discrete time model); instead jumps from one state to another occur at rates specified by some real numbers. These rates determine an exponentially distributed time between transitions from one configuration of the network into another, according to a continuous time Markov chain model (cf. e.g.

[16,14,1]). In our model each node can initiate a network update independently at any time with a certain probability which is proportional to its rate. This parameter is specified by the superscript λ in the syntax of a node. We assume that these rates are independent on the time and therefore each node "fires", i.e. initiates an update, via a so called Poisson process (see e.g. [14, Sect 2.4]).

The global semantics for pKLAIM is defined in Table 8. The transition relation $\overset{t_{ij}}{\Longrightarrow}$ between two network configurations N_i and N_j is labelled by rates t_{ij} which are obtained as a product between the firing rate of the node which initiates the update and the *normalised* probabilities of the local transitions occurring in the nodes involved in the update. The normalisation of the local transition probabilities is needed in order to take into account the possibility of a node "inaction" and to ensure that only active processes are effectively selected on the node (see [6] for a formal definition of the normalisation procedure). The rates t_{ij} allow us to compute the probability of a global transition from the network configuration N_j at time t provided we started in state N_i at time $t = 0$ simply as:

$$P(N(t) = N_j \mid N(0) = N_i) = (\exp(t\mathbf{T}))_{ij}$$

where

$$t_{ij} = \mathbf{T}_{ij} = \mathbf{Q}_{ij}\mathbf{P}_{ij} \text{ for } i \neq j$$
$$t_{ii} = \mathbf{T}_{ii} = -\textstyle\sum_j \mathbf{Q}_{ij}\mathbf{P}_{ij}$$

and $\mathbf{P}_{ij}$ is the local discrete probability, and $\mathbf{Q}_{ij}$ is the rate associated to the node which initiates the global update from N_i to N_j.

Example 3.1 Consider the following simple three node pKLAIM network:

$$l_1 ::_\phi^{\lambda_1} (\frac{1}{3}\mathbf{out}(t)@l_2 + \frac{2}{3}\mathbf{out}(s)@l_2) \parallel l_2 ::_\phi^{\lambda_2} \mathbf{in}(x)@l_2.[x] \parallel l_3 ::_\phi^{\lambda_3} \mathbf{out}(u)@l_2.$$

The idea is that the node at location l_2 can consume a single token by executing the **in** action and by substituting this token for every occurrences of x in the 'body' $[x]$. We are not interested in the concrete form of the 'body' and therefore just indicate it by '$[x]$'; it could concretely be $[x] \equiv \mathbf{out}(x)@l_0$, $[x] \equiv \mathbf{read}(x)@\mathbf{self}$, etc.

The token to be consumed could originate in either of the two nodes at location l_1 or l_3. Intuitively, it is clear that it will depend on the rates λ_1 and λ_3 which token will be first placed at node l_2 and on the rate λ_2 how fast it will be consumed. In other words, there is a 'global competition' between l_1 and l_2 on which node has a higher chance to place its tokens at l_2. At the same time there is also a 'local competition' between the tokens s and t in the choice construct at l_1.

In order to illustrate how these different stochastic elements interact and what network configurations we may obtain after some time t we first have to enumerate all the reachable configurations as in Figure 1.

Based on this enumeration we can specify the matrix $\mathbf{Q}$ which contains all information on the transition rates between global network configurations and

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

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

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

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

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

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

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

$$\frac{P_1 \xrightarrow[\sigma_1]{e(Q)@\ell}_p P_1' \quad \langle s_2, p_{s_2}\rangle \in (\sigma_1 \bullet \varrho_1)(\ell)}{s_1 ::^{\lambda_1}_{\varrho_1} P_1 \parallel s_2 ::^{\lambda_2}_{\varrho_2} P_2 \rightarrowtail\!\!\!\!\Longrightarrow^{p_{s_2}\cdot\tilde{p}\cdot\lambda_1} s_1 ::^{\lambda_1}_{\varrho_1} P_1' \parallel s_2 ::^{\lambda_2}_{\varrho_2} P_2 \mid Q}$$

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

$$\frac{s ::^{\lambda}_{\varrho} P_1 \rightarrowtail\!\!\!\!\Longrightarrow^{p} s ::^{\lambda}_{\varrho} P_1'}{s ::^{\lambda}_{\varrho} P_1 \mid P_2 \rightarrowtail\!\!\!\!\Longrightarrow^{\tilde{p}\cdot\lambda} s ::^{\lambda}_{\varrho} P_1' \mid P_2}$$

Table 8
The Global Continuous Time Structural Semantics

$$
\begin{array}{llll}
(i) & l_1 ::^{\lambda_1}_{\phi} (\frac{1}{3}\mathbf{out}(s)@l_2 + \frac{2}{3}\mathbf{out}(t)@l_2) \parallel & l_2 ::^{\lambda_2}_{\phi} \mathbf{in}(x)@l_2.[x] & \parallel l_3 ::^{\lambda_3}_{\phi} \mathbf{out}(u)@l_2 \\
(ii) & l_1 ::^{\lambda_1}_{\phi} \mathbf{nil} \quad = & l_2 ::^{\lambda_2}_{\phi} \mathbf{in}(x)@l_2.[x]|\mathbf{out}(s) & \parallel l_3 ::^{\lambda_3}_{\phi} \mathbf{out}(u)@l_2 \\
(iii) & l_1 ::^{\lambda_1}_{\phi} \mathbf{nil} \quad = & l_2 ::^{\lambda_2}_{\phi} \mathbf{in}(x)@l_2.[x]|\mathbf{out}(t) & \parallel l_3 ::^{\lambda_3}_{\phi} \mathbf{out}(u)@l_2 \\
(iv) & l_1 ::^{\lambda_1}_{\phi} (\frac{1}{3}\mathbf{out}(s)@l_2 + \frac{2}{3}\mathbf{out}(t)@l_2) \parallel & l_2 ::^{\lambda_2}_{\phi} \mathbf{in}(x)@l_2.[x]|\mathbf{out}(u) & \parallel \quad l_3 ::^{\lambda_3}_{\phi} \mathbf{nil} \\
(v) & l_1 ::^{\lambda_1}_{\phi} \mathbf{nil} \quad = & l_2 ::^{\lambda_2}_{\phi} [s] & \parallel l_3 ::^{\lambda_3}_{\phi} \mathbf{out}(u)@l_2 \\
(vi) & l_1 ::^{\lambda_1}_{\phi} \mathbf{nil} \quad = & l_2 ::^{\lambda_2}_{\phi} [t] & \parallel l_3 ::^{\lambda_3}_{\phi} \mathbf{out}(u)@l_2 \\
(vii) & l_1 ::^{\lambda_1}_{\phi} (\frac{1}{3}\mathbf{out}(s)@l_2 + \frac{2}{3}\mathbf{out}(t)@l_2) \parallel & l_2 ::^{\lambda_2}_{\phi} [u] & \parallel \quad l_3 ::^{\lambda_3}_{\phi} \mathbf{nil} \\
(viii) & l_1 ::^{\lambda_1}_{\phi} \mathbf{nil} \quad \parallel & l_2 ::^{\lambda_2}_{\phi} \mathbf{in}(x)@l_2.[x]|\mathbf{out}(s)|\mathbf{out}(u) & \parallel \quad l_3 ::^{\lambda_3}_{\phi} \mathbf{nil} \\
(ix) & l_1 ::^{\lambda_1}_{\phi} \mathbf{nil} \quad \parallel & l_2 ::^{\lambda_2}_{\phi} \mathbf{in}(x)@l_2.[x]|\mathbf{out}(t)|\mathbf{out}(u) & \parallel \quad l_3 ::^{\lambda_3}_{\phi} \mathbf{nil} \\
(x) & l_1 ::^{\lambda_1}_{\phi} \mathbf{nil} \quad = & l_2 ::^{\lambda_2}_{\phi} [s]|\mathbf{out}(u) & = \quad l_3 ::^{\lambda_3}_{\phi} \mathbf{nil} \\
(xi) & l_1 ::^{\lambda_1}_{\phi} \mathbf{nil} \quad = & l_2 ::^{\lambda_2}_{\phi} [t]|\mathbf{out}(u) & = \quad l_3 ::^{\lambda_3}_{\phi} \mathbf{nil} \\
(xii) & l_1 ::^{\lambda_1}_{\phi} \mathbf{nil} \quad = & l_2 ::^{\lambda_2}_{\phi} [u]|\mathbf{out}(s) & = \quad l_3 ::^{\lambda_3}_{\phi} \mathbf{nil} \\
(xiii) & l_1 ::^{\lambda_1}_{\phi} \mathbf{nil} \quad = & l_2 ::^{\lambda_2}_{\phi} [u]|\mathbf{out}(t) & = \quad l_3 ::^{\lambda_3}_{\phi} \mathbf{nil}
\end{array}
$$

Fig. 1. Enumeration of Reachable Network Configurations

the matrix $\mathbf{P}$ containing the discrete transition probabilities. From $\mathbf{Q}$ and $\mathbf{P}$ we can directly construct the matrix $\mathbf{T}$ which is given in Figure 2.

If we specify concrete values for λ_1, λ_2, and λ_3 we can compute the probability that the network configuration $N(t)$ at any future time t is N_j if we start in a certain initial configuration N_i as $P(N(t) = N_j \mid N(0) = N_i) = (\exp(t\mathbf{T}))_{ij}$.

$$
\mathbf{T} = \begin{pmatrix}
-\lambda_1-\lambda_3 & \frac{1}{3}\lambda_1 & \frac{2}{3}\lambda_1 & \lambda_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -\lambda_2-\lambda_3 & 0 & 0 & \lambda_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -\lambda_2-\lambda_3 & 0 & 0 & \lambda_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\lambda_1-\lambda_2 & 0 & 0 & \lambda_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\lambda_3 & 0 & 0 & \lambda_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -\lambda_3 & 0 & 0 & \frac{1}{3}\lambda_1 & \frac{2}{3}\lambda_1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\lambda_3 & 0 & 0 & 0 & \lambda_3 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda_1 & 0 & 0 & 0 & \lambda_3 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda_2 & 0 & \frac{1}{2}\lambda_2 & 0 & \frac{1}{2}\lambda_2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda_2 & 0 & \frac{1}{2}\lambda_2 & 0 & \frac{1}{2}\lambda_2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda_2 & 0 & \frac{1}{2}\lambda_2 & 0 & \frac{1}{2}\lambda_2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda_2 & 0 & \frac{1}{2}\lambda_2 & 0 & \frac{1}{2}\lambda_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda_2 & 0 & \frac{1}{2}\lambda_2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda_2 & 0 & \frac{1}{2}\lambda_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda_2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda_2
\end{pmatrix}
$$

Fig. 2. The Combined Transition Rate Matrix $\mathbf{T}$

We can also look at the long run behaviour, i.e. what will happen to the network if we wait a (relatively) long time. For different values of λ_1, λ_2, and λ_3 we get different behaviours as in the following examples.

$\lambda_1 = 1, \lambda_2 = 1, \lambda_3 = 1$**.** In this case all nodes run at about the same speed. We reach one of the four last configurations with the following probabilities:

$$P(N(1000) = N_x \mid N(0) = N_i) = \frac{1}{6}$$

$$P(N(1000) = N_{xi} \mid N(0) = N_i) = \frac{1}{3}$$
$$P(N(1000) = N_{xii} \mid N(0) = N_i) = \frac{1}{6}$$
$$P(N(1000) = N_{xiii} \mid N(0) = N_i) = \frac{1}{3}$$

The probabilities in ending up in any other configuration is (effectively) zero.

$\lambda_1 = 10^{10}, \lambda_2 = 1, \lambda_3 = 10^{-10}$**.** Here the middle node runs at normal speed, while the first node is extremely fast and the third one is extremely slow. In this case we reach one of the intermediate nodes N_v and N_{vi} with probabilities

$$P(N(1000) = N_v \mid N(0) = N_i) = \frac{1}{3}$$
$$P(N(1000) = N_{vi} \mid N(0) = N_i) = \frac{2}{3}$$

The other probabilities are (effectively) zero. The reason why we do not reach a final configuration is that the third node is so extremely slow that its token u does not reach the middle node in 1000 time steps.

$\lambda_1 = 10^{10}, \lambda_2 = 1, \lambda_3 = 1$**.** Finally, we look at the situation in which the first node is very fast while the others run at a normal speed. The resulting final configurations are now the same as in the first case but with different probabilities

$$P(N(1000) = N_x \mid N(0) = N_i) = \frac{1}{4}$$
$$P(N(1000) = N_{xi} \mid N(0) = N_i) = \frac{1}{2}$$
$$P(N(1000) = N_{xii} \mid N(0) = N_i) = \frac{1}{12}$$
$$P(N(1000) = N_{xiii} \mid N(0) = N_i) = \frac{1}{6}$$

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).

4 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 use rates to determine how often a node is active. This information contributes to the probability of network updates.

The probabilistic version of KLAIM we have introduced in this paper is closely related to various *probabilistic programming languages* and *probabilistic process calculi* proposed in the recent literature. Among these we mention discrete time approaches — e.g. PCCS [8,12], PCCP [7], etc. — as well as continuous time approaches — e.g. PEPA [10], Stochastic π calculus [15]. Work in performance analysis is often based on probabilistic process calculi, for example, on Hillston's PEPA [11], or EMPA by Bernardo and Gorrieri [3]. 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 [13] and de Alfaro [4].

We have considered here a model based on Poisson processes which are some of the simplest examples of continuous-time Markov chains. More complicated continuous time behaviour could be considered, but this might require more parameters than just rate to describe the time distributions [1]. The language could also be extended so as to allow for a dynamic change of probabilities and rates, i.e. for rate and probability which depend on the time. These last two extensions require further work.

References

[1] Bause, F. and P. S. Kritzinger, "Stochastic Petri Nets – An Introduction to the Theory," Vieweg Verlag, 2002, second edition.

[2] Bergstra, J., A. Ponse and S. Smolka, editors, "Handbook of Process Algebra," Elsevier Science, Amsterdam, 2001.

[3] Bernardo, M. 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).

[4] de Alfaro, L., "Formal Verification of Probabilistic Systems," Ph.D. thesis, Stanford University, Department of Computer Science (1998).

[5] De Nicola, R., G. Ferrari and R. Pugliese, *KLAIM: A kernel language for agents interaction and mobility*, IEEE Transactions on Software Engineering **24** (1998), pp. 315–330.

[6] Di Pierro, A., C. Hankin and H. Wiklicky, *Probabilistic KLAIM*, in: R. D. Nicola, G. Ferrari and G. Meredith, editors, *Proceeddings of Coordination 2004*, number 2949 in Lecture Notes in Computer Science (2004), pp. 119–134.

[7] Di Pierro, A. 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 (2000).

[8] Giacalone, A., C.-C. Jou and S. Smolka, *Algebraic reasoning for probabilistic concurrent systems*, in: *Proceedings of the IFIP WG 2.2/2.3 Working Conference on Programming Concepts and Methods* (1990), pp. 443–458.

[9] Gollmann, D., "Computer Security," John Wiley & Sons, 1999.

[10] Hillston, J., *PEPA: Performance enhanced process algebra*, Technical Report CSR-24-93, University of Edinburgh, Edinburgh, Scotland (1993).

[11] Hillston, J., "A Compositional Approach to Performance Modelling," Cambridge University Press, 1996.

[12] Jonsson, B., W. Yi and K. Larsen, "Probabilistic Extentions of Process Algebras," Elsevier Science, Amsterdam, 2001 pp. 685–710, see [2].

[13] Kwiatkowska, M., 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*, Lecture Notes in Computer Science **2280** (2002), pp. 52–66.

[14] Norris, J., "Markov Chains," Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press, Cambridge, 1997.

[15] Priami, C., *Stochastic π-calculus*, Computer Journal **38** (1995), pp. 578–589.

[16] Tijms, H. C., "Stochastic Models – An Algorithmic Approach," John Wiley & Sons, Chichester, 1994.