PEPA nets:
A structured performance modelling formalism

Stephen Gilmore [a], Jane Hillston [a], Leïla Kloul [a,1], Marina Ribaudo [b]

[a] *Laboratory for Foundations of Computer Science, The University of Edinburgh, Edinburgh EH9 3JZ, Scotland. Email:* {stg, jeh, leila} @inf.ed.ac.uk

[b] *Dipartimento di Informatica e Scienze dell'Informazione, Università di Genova, Via Dodecaneso 35, 16146 Genova, Italia. Email:* ribaudo@disi.unige.it

Abstract

In this paper we describe a formalism which uses the stochastic process algebra PEPA as the inscription language for labelled stochastic Petri nets. Viewed in another way, the net is used to provide a structure for linking related PEPA systems. The combined modelling language naturally represents such applications as mobile code systems where the PEPA terms are used to model the program code which moves between network hosts (the places in the net). We describe the implementation of a tool to support this modelling formalism and apply this to model a hierarchical cellular network.

1 Introduction

Variants of Petri nets have been widely used in the description and performance analysis of computer, telecommunications and manufacturing systems [1]. The appeal of Petri nets as a modelling formalism is easy to see. They provide a graphical presentation of a model which has an easily accessible interpretation and they also have the advantage of being supported by an unambiguous formal semantics.

In their use as performance modelling languages stochastic Petri nets have recently been joined by stochastic process algebras such as PEPA [20], EMPA [4] and IMC [18]. Stochastic process algebras lack the attractive graphical presentation of Petri nets and properties such as the depiction of causality and conflict in a model. In contrast though, in stochastic process algebras an explicit compositional structure is imposed on the model. This structure makes the model easy to

[1] L. Kloul is on leave from PRISM, Université de Versailles, 45, Av. des Etats-Unis 78000 Versailles, France.

understand, may alleviate problems of model construction and can be exploited for both qualitative and quantitative analysis. A comparison of these two modelling formalisms [11] concludes that "there is scope for future work incorporating the attractive characteristics of the formalisms, such as structural analysis or functional abstraction, from one paradigm into the other". Some work has been done in this area in beginning to develop a structural theory for process algebras [14] on the one hand and in importing composition operations from stochastic process algebras into net formalisms on the other [30,21,19]. The present work considers using both Petri nets and process algebras together as a single, structured performance modelling formalism. There is some reason to believe that these two formalisms complement each other.

Petri nets have previously been combined with other modelling formalisms such as the lazy functional programming language Haskell (used with non-stochastic Petri nets in [29]) and queueing models (used with generalised stochastic Petri nets in [3]). The combination of stochastic Petri nets with queueing networks in particular has been a source of inspiration to several authors. Earlier work in this area includes Bause's *Queueing Petri nets* [2] and Haverkort's *Dynamic Queueing Networks* [17]. An extension of (non-stochastic) Petri nets which provides modelling concepts similar to ours is Valk's *Elementary Object systems* [31]. The tokens in an elementary object system are themselves Petri nets having individual dynamic behaviour.

Coloured Petri nets are a high-level form of classical Petri nets. The plain (indistinguishable) tokens of a classical Petri net are replaced by arbitrary terms which are distinguishable. In stochastic Petri nets the evolution of the net from one marking to another is associated with a random variable drawn from an exponential distribution. Here we consider coloured stochastic Petri nets where the colours used as the tokens of the net are PEPA components. We refer to these as *PEPA nets* from here on.

Structure of this paper: Section 2 introduces the notation and terminology of PEPA nets, to give the reader an informal explanation of the ideas. However, PEPA is a formal language with a precise semantic definition and so in Section 3 we present the operational semantics of PEPA nets. In Section 4 we present two small examples, a simple mobile agent system and a Jini architecture, both modelled as PEPA nets. Having presented the reader with examples of modelling with PEPA nets we then compare them to the related modelling formalisms of Petri nets and the PEPA stochastic process algebra in Section 5. In each case we seek to show that PEPA nets offer some expressivity which is not directly offered by the other formalisms. Section 6 is a more detailed case study of a hierarchical cellular network. In Section 7 we discuss tool support for PEPA nets. Related work is discussed in Section 8. Concluding remarks and further work are presented in Section 9.

2 PEPA nets

In this section we present the concepts and definitions used in PEPA nets. In the following paragraphs we give a brief overview of PEPA. Readers are referred to [20] for a more detailed introduction.

2.1 Summary of the PEPA language

The PEPA language provides a small set of combinators. These allow language terms to be constructed defining the behaviour of components, via the activities they undertake and the interactions between them. The syntax may be formally introduced by means of the grammar shown in the lower part of Figure 1. In that grammar S denotes a *sequential component* and P denotes a *model component* which executes in parallel. I stands for a constant which denotes either a sequential or a model component, as defined by a defining equation. The component combinators, together with their names and interpretations, are presented informally below.

Prefix: The basic mechanism for describing the behaviour of a system is to give a component a designated first action using the prefix combinator, denoted by a full stop. For example, the component $(\alpha, r).S$ carries out activity (α, r), which has action type α and an exponentially distributed duration with parameter r, and it subsequently behaves as S.

Choice: The life cycle of a sequential component may be more complex than any behaviour which can be expressed using the prefix combinator alone. The choice combinator captures the possibility of competition between different possible activities. The component $P + Q$ represents a system which may behave either as P or as Q. The activities of both P and Q are enabled. The first activity to complete distinguishes one of them: the other is discarded. The system will behave as the derivative resulting from the evolution of the chosen component.

Constant: It is convenient to be able to assign names to patterns of behaviour associated with components. Constants are components whose meaning is given by a defining equation.

Hiding: The possibility to abstract away some aspects of a component's behaviour is provided by the hiding operator, denoted P/L. Here, the set L of visible action types identifies those activities which are to be considered internal or private to the component and which will appear as the unknown type τ.

Cooperation: In PEPA direct interaction, or *cooperation*, between components is the basis of compositionality. The set which is used as the subscript to the cooperation symbol, the *cooperation set* L, determines those activities on which the *cooperands* are forced to synchronise. For action types not in L, the components proceed independently and concurrently with their enabled activities. However, if a component enables an activity whose action type is in the cooperation set it will not be able to proceed with that activity until the other component also enables an activity of that type. The two components then proceed together to complete the *shared activity*. The rate of the shared activity may be altered to reflect the work carried out by both components to complete the activity (for details see [20]). We write $P \underset{L}{\bowtie} Q$ to denote cooperation between P and Q over L. We write $P \parallel Q$ as an abbreviation for $P \underset{L}{\bowtie} Q$ when L is empty.

In some cases, when an activity is known to be carried out in cooperation with another component, a component may be *passive* with respect to that activity. This means that the rate of the activity is left unspecified (denoted $\top$) and is determined upon cooperation, by the rate of the activity in the other component. All passive actions must be synchronised in the final model.

Model components capture the structure of the system in terms of its *static* components. The dynamic behaviour of the system is represented by the evolution of these components, either individually or in cooperation. The form of this evolution is governed by a set of formal rules which give an operational semantics of PEPA terms. The semantic rules, in the structured operational style, are presented in Figure A.1 in the Appendix without further comment.

The semantics of each term in PEPA is given via a labelled *multi-transition* system—the multiplicities of arcs are significant. In the transition system a state corresponds to each syntactic term of the language, or *derivative*, and an arc represents the activity which causes one derivative to evolve into another. The complete set of reachable states is termed the *derivative set* of a model and these form the nodes of the *derivation graph* which is formed by applying the semantic rules exhaustively.

The timing aspects of components' behaviour are represented on each arc as the parameter of the negative exponential distribution governing the duration of the corresponding activity. The interpretation is as follows: when enabled an activity $a = (\alpha, r)$ will delay for a period sampled from the negative exponential distribution which has parameter r. If several activities are enabled concurrently, either in competition or independently, we assume that a *race condition* exists between them. The evolution of the model will determine whether the other activities have been *aborted* or simply *interrupted* by the resulting state change. In either case the memoryless property of the distribution eliminates the need to record the previous execution time.

When two components carry out an activity in cooperation the rate of the shared activity will reflect the working capacity of the slower component. We assume that each component has a capacity for performing an activity type α, which cannot be enhanced by working in cooperation, unless the component is passive with respect to that activity type. For a component P and an action type α, this capacity is termed the *apparent rate* [20] of α in P. It is the sum of the rates of the α type activities enabled in P. The apparent rate of α in a cooperation between P and Q over α will be the minimum of the apparent rate of α in P and the apparent rate of α in Q.

The derivation graph is the basis of the underlying Continuous Time Markov Chain (CTMC) which is used to derive performance measures from a PEPA model. The graph is systematically reduced to a form where it can be treated as the state transition diagram of the underlying CTMC. Each derivative is then a state in the CTMC. The *transition rate* between two derivatives P and Q in the derivation graph is the rate at which the system changes from behaving as component P to behaving as Q. It is denoted by $q(P, Q)$ and is the sum of the activity rates labelling arcs connecting node P to node Q. In order for the CTMC to be *ergodic* its derivation graph must be strongly connected. Some necessary conditions for ergodicity, at the syntactic level of a PEPA model, have been defined [20]. These syntactic conditions are imposed by the grammar in Figure 1.

2.2 Introduction to PEPA nets

In PEPA, as in most performance modelling formalisms, there is a single modelling mechanism, *activities*, used to represent changes of state within a system. PEPA nets are motivated by the observation that in many systems we can identify distinct types of change of state. A PEPA net differentiates between two types of change of state. We refer to these as *firings* of the net and *transitions* of PEPA components. Each are special cases of PEPA activities. Transitions of PEPA components will typically be used to model small-scale (or *local*) changes of state as components undertake activities. Firings of the net will typically be used to model macro-step (or *global*) changes of state such as context switches, breakdowns and repairs, one thread yielding to another, or a mobile software agent moving from one network host to another.

A firing in a PEPA net causes the transfer of one token from one place to another. The token which is moved is a PEPA component, which causes a change in the subsequent evaluation both in the source (where existing cooperations with other components now can no longer take place) and in the target (where previously disabled cooperations are now enabled by the arrival of an incoming component which can participate in these interactions). Firings have global effect because they involve components at more than one place in the net.

A transition in a PEPA net takes place whenever a transition of a PEPA component can occur (either individually, or in cooperation with another component). Components can only cooperate if they are resident in the same place in the net. The PEPA net formalism does not allow components at different places in the net to cooperate on a shared activity. An analogy is with message-passing distributed systems without shared-memory where software components on the same host can exchange information without incurring a communication overhead but software components on different hosts cannot. Additionally we do not allow a firing to coincide with a transition which is shared, i.e. it is not possible for two components in one place to cooperate *and* transfer to another place as an atomic action. Thus transitions in a PEPA net have local effect because they involve only components at one place in the net. Maintaining this strict distinction between firings and transitions is essential in order to provide the separation into macro- and micro-step state changes that we are seeking to represent.

Each place has a distinct alphabet for transitions and firings, meaning that the same action type cannot be used for both. Thus there can be no ambiguity between such micro- and macro-scale transitions.

Within the set of firings offered at the net level of a PEPA net we allow the modeller to assign different priorities. Note that this does not mean that more than two time-scales of activity are being represented. This mechanism is offered as a modelling convenience to allow one macro-step transition to be fired in preference to another when both are enabled.

A PEPA net is made up of PEPA *contexts*, one at each place in the net. A context consists of a number of *static* components (possibly zero) and a number of *cells* (at least one). Like a memory location in an imperative program, a cell is a storage area to be filled by a datum of a particular type. In particular in a PEPA net, a cell is a storage area dedicated to storing a PEPA component. The components which fill cells can circulate as the tokens of the net. In contrast, the static components cannot move. Most variants of Petri nets do not include static tokens, the closest concept being "self loops" where a token is deleted from a place and then immediately replaced. Here static components provide the infrastructure of the place and act as cooperation partners in synchronisation activities with tokens. Contexts have previously been used in both classical process algebras [26], and in the stochastic process algebra PEPA [9].

We use the notation $Q[\_]$ to denote a context which could be filled by the PEPA component Q or one with the same alphabet. If Q has derivatives Q' and Q'' only and no other component has the same alphabet as Q then there are four possible values for such a context: $Q[\_]$, $Q[Q]$, $Q[Q']$ and $Q[Q'']$. $Q[\_]$ enables no transitions.

$Q[Q]$ enables the same transitions as Q. $Q[Q']$ enables the same transitions as Q'. $Q[Q'']$ enables the same transitions as Q''. As usual with PEPA components we require that the component has an ergodic definition so that it is always possible to return to a state which one has previously reached. This has as a consequence that if Q' is a derivative of Q then it is also the case that Q is a derivative of Q', for any Q and Q'.

For any token component its action type set can be partitioned in distinct subsets corresponding to transitions and firings respectively. For a component Q we will denote these sets by $\mathcal{A}_t(Q)$ and $\mathcal{A}_f(Q)$, where $\mathcal{A}_t(Q)$ is the set of local transitions currently enabled in Q and $\mathcal{A}_f(Q)$ is the set of firings currently enabled for Q. Note that for a firing to be enabled the token must enable the corresponding activity, it must be in a place connected to a net-level transition of the same type and there must be an empty cell at the output place of the transition of the correct token type.

We use capitalised names to denote PEPA components (such as P and Q) and lowercase for PEPA transitions (such as a and b). We use bold capitalised names for PEPA net places (such as $\mathbf{P_1}$ and $\mathbf{P_2}$) and bold lowercase for PEPA net firings (such as $\mathbf{a}$ and $\mathbf{b}$).

2.3 Markings in a PEPA net

The *marking* of a classical Petri net records the number of tokens which are resident at each place in the net. Since the tokens of a classical Petri net are indistinguishable it is sufficient to record their number and one could present the marking of a Petri net with places P_1, P_2 and P_3 as $(P_1 : 2, P_2 : 1, P_3 : 0)$. If an ordering is imposed on the places of the net a more compact representation of the marking can be used. Place names are omitted and the marking can be written using vector notation thus, $(2, 1, 0)$.

Consider now a PEPA net with places $\mathbf{P_1}$, $\mathbf{P_2}$ and $\mathbf{P_3}$ as shown below.

$$\mathbf{P_1}[Q] \stackrel{\mathit{def}}{=} Q[Q] \underset{L}{\bowtie} R$$

$$\mathbf{P_2}[Q] \stackrel{\mathit{def}}{=} Q[Q] \underset{K}{\bowtie} S$$

$$\mathbf{P_3}[Q] \stackrel{\mathit{def}}{=} Q[Q] \underset{K \cup L}{\bowtie} (R \parallel S)$$

From its use in the contexts at each place we see that Q is a component which can move as a token around the net whereas R and S are static components which cannot move. There is a copy of R at place $\mathbf{P_1}$ and another at $\mathbf{P_3}$. There is a copy of S at place $\mathbf{P_2}$ and another at $\mathbf{P_3}$.

Given the above definitions for the places in this PEPA net, we can denote a marking of this net by $(\mathbf{P_1}[Q], \mathbf{P_2}[\_], \mathbf{P_3}[\_])$. In general, a context may have more than one parameter, to be filled by PEPA components of different types. Where an ordering is imposed on places and each context has only a single cell to be filled we can abbreviate such a marking by $(Q, \_, \_)$.

Moreover, the local state captured by a place marking will also depend on the current state of the static components in the place. To identify these states we allow place definitions to specify a particular state of each of the static components. Thus, in the example above, if S can evolve to S' we can define $\mathbf{P'_2}[Q] \stackrel{\mathit{def}}{=} Q[Q] \underset{K}{\bowtie} S'$.

2.4 Net-level transitions in a PEPA net

Transitions at the net-level of a PEPA net are labelled in a similar way to the labelled multi-transition system which records the unfolding of the state space of a PEPA model. A labelling function ℓ maps transition names into pairs of names such as (α, r) where it is possible that $\ell(t_i) = \ell(t_j)$ but $t_i \neq t_j$. The first element of a pair (α, r) specifies an *activity* which must be performed in order for a component to move from the input place of the transition to the output place. The activity type records formally the activity which must be performed if the transition is to fire. The second element is an exponentially-distributed random variable which quantifies the *rate* at which the activity can progress in conjunction with the component which is performing it.

As an example, suppose that Q is a component which is currently at place $\mathbf{P_1}$ and that it can perform an activity α with rate r_1 to produce the derivative Q'. Further, say that the net has a transition between $\mathbf{P_1}$ and $\mathbf{P_2}$ labelled by (α, r_2). If Q performs activity α in this setting it will be removed from $\mathbf{P_1}$ (leaving behind an empty cell) and Q' will be deposited into $\mathbf{P_2}$ (filling an empty cell there).

A priority function π maps action types to the natural numbers, and can be used to eliminate some firings from the labelled multi-transition system: only enabled firings with the highest priority value are considered eligible to fire. For example, suppose that Q is a component which is currently at place $\mathbf{P_1}$ and that it can perform activities of types α, β and γ where $\pi(\alpha) = \pi(\beta) = 2$ whereas $\pi(\gamma) = 1$. Further, suppose that there are net transitions between $\mathbf{P_1}$ and each of $\mathbf{P_2}$, $\mathbf{P_3}$ and $\mathbf{P_4}$ labelled by α, β and γ respectively. Assuming that there are empty cells in all places, Q may perform activity α and be deposited in place $\mathbf{P_2}$, or activity β and be deposited in place $\mathbf{P_3}$ but it cannot perform activity γ and be deposited in place $\mathbf{P_4}$. Only if there are no empty cells in places $\mathbf{P_2}$ and $\mathbf{P_3}$ will activity γ become enabled.

2.5 Net structure of a PEPA net

From the preceding explanation it is clear that the expression of the macro-level structure of a PEPA net could be represented by any transition-based modelling formalism. Indeed it would be possible to use a PEPA component to control the possible "firings" (macro-steps) of the model. However, we feel that there are some advantages to using a Petri net in this role.

Firstly, using a different formalism gives a clearer separation of concerns within our model making it both easier to construct and to understand. Furthermore, this macro-level is often of a size that can benefit from graphical representation, to give an intuitive understanding of the coarse structure of the model. Finally, the *movement* of components—to a new host, to a new context, etc.—has resonance with the systems we study.

The class of nets that we currently use for modelling the net structure of a PEPA net is restricted to *structural state machines*, i.e. nets whose transitions can have only one input place and one output place. This means that we can represent conflicts at the net level, while synchronisations are not allowed. This is consistent with the fact that PEPA components cannot cooperate on a shared activity when they are resident in different places. However, we have imposed this restriction in the interests of developing a clear theory of PEPA nets incrementally; it is not in anyway inherent in the formalism. Indeed we hope to relax it in due course.

It is usual with coloured Petri nets to associate functions with arcs, offering a generalisation of the usual, basic "functions" offered by arc multiplicities. In PEPA nets the arc functions are implicit. The modification of a token which takes place when it is fired is wholly specified by the action type of the firing, the definition of the token and the semantics. Furthermore, although we allow multiple tokens within net places, only one token can move at each firing. Thus arc multiplicities greater than one are not allowed.

3 Semantics

The PEPA language is formally defined by a small-step operational semantics. In order to describe the firing rule for PEPA nets formally we need a relational operator which is to be used to express the fact that there exists a particular transition in the net superstructure. This operator must have the properties that it identifies the source and target of the transition and that it records the activity which is to be performed in order for a component to cross this transition, moving from the source to the target. We use the notation

$$\mathbf{P_1} \xrightarrow{(\alpha, r)} \!\!\!\! \boxed{} \!\!\!\!\longrightarrow \mathbf{P_2}$$

to capture the information that there is a transition connecting place $\mathbf{P_1}$ to place $\mathbf{P_2}$ labelled by (α, r). This relation captures static information about the structure of the net, not dynamic information about its behaviour. We could describe the net structure in a PEPA net using a list of such declarations but the more familiar graphical presentation of a net presents the same information in a more accessible way.

The introduction of contexts requires an extension to the syntax of PEPA. This extension is presented in Figure 1.

$N ::= D^+ M$ (net)

(definitions and marking)

$M ::= (M_{\mathbf{P}}, \ldots)$	(marking)	$D ::= I \stackrel{\textit{def}}{=} S$	(component defn)
$M_{\mathbf{P}} ::= \mathbf{P}[C, \ldots]$	(place marking)	$\mid \ \mathbf{P}[C] \stackrel{\textit{def}}{=} P[C]$	(place defn)
		$\mid \ \mathbf{P}[C, \ldots] \stackrel{\textit{def}}{=} P[C] \underset{L}{\bowtie} P$	(place defn)
(marking vectors)		(identifier declarations)	

$S ::= (\alpha, r).S$	(prefix)	$P ::= P \underset{L}{\bowtie} P$	(cooperation)	$C ::=$ '_'	(empty)
$\mid \ S + S$	(choice)	$\mid \ P/L$	(hiding)	$\mid \ S$	(full)
$\mid \ I$	(identifier)	$\mid \ P[C]$	(cell)		
		$\mid \ I$	(identifier)		
(sequential components)		(concurrent components)		(cell term expressions)	

Fig. 1. The syntax of PEPA extended with contexts

We assume that there is a set $\mathcal{A}$ of PEPA action types which can be partitioned into disjoint subsets $\mathcal{A}_f$ and $\mathcal{A}_t$ corresponding to firings and local transitions respectively.

Definition 1 *A PEPA net $\mathcal{N}$ is a tuple $\mathcal{N} = (\mathcal{P}, \mathcal{T}, I, O, \ell, \pi, \mathcal{C}, D, M_0)$ such that*

- *$\mathcal{P}$ is a finite set of places;*

- *$\mathcal{T}$ is a finite set of net transitions;*
- *$I : \mathcal{T} \to \mathcal{P}$ is the input function;*
- *$O : \mathcal{T} \to \mathcal{P}$ is the output function;*
- *$\ell : \mathcal{T} \to (\mathcal{A}_f, \mathbb{R}^+ \cup \{\top\})$ is the labelling function, which assigns a PEPA activity ((type, rate) pair) to each transition. The rate determines the negative exponential distribution governing the delay associated with the transition;*
- *$\pi : \mathcal{A}_f \to \mathbb{N}$ is the priority function which assigns priorities (represented by natural numbers) to firing action types;*
- *$\mathcal{C} : \mathcal{P} \to P$ is the place definition function which assigns a PEPA context, containing at least one cell, to each place;*
- *D is the set of token component definitions;*
- *M_0 is the initial marking of the net.*

The semantic rules for PEPA nets are provided in Figure 2. The Cell rule conservatively extends the PEPA semantics to define that a cell which is filled by a component Q has the same transitions as Q itself. A healthiness condition on the rule (also called a *typing judgement*) requires a context such as $Q[\_]$ to be filled with a component which has the same alphabet as Q. We write $Q =_a Q'$ to state that Q and Q' have the same alphabet. There are no rules to infer transitions for an empty cell because an empty cell enables no transitions.

The Transition rule states that the net has local transitions which change only a single component in the marking vector. This rule also states that these transitions agree with the transitions which are generated by the PEPA semantics (including the extension for contexts). Recall that the transition and firing alphabets of any place must be distinct. We do not give priority to one alphabet of actions over the other; the highest-priority firings and the transitions compete based on a race policy.

The Firing rule takes one marking of the net to another marking by performing a PEPA activity and moving a PEPA component from the input place to the output place. This has the effect that two entries in the marking vector change simultaneously. In order to take account of the priorities we define a number of supplementary transition relations, one for each priority level.

A net level transition's eligibility for firing depends on two conditions. Firstly there must be an empty cell in the destination place into which the token can be transferred. The Enabling rule ensures that this is the case, and defines a transition relation, decorated with the priority level of the corresponding activity type. The rate at which the activity is enabled is calculated as in the PEPA semantics of cooperation.

In order for a firing to take place it must also be the case that the type of the enabled firing has the highest priority level in the set of the enabled firings. This is imposed by the Firing rule in which we discard those enabled firings which do not have the highest priority. In other words for a firing to occur there must not be any other firing satisfying the Enabling rule (empty destination cell) which has a higher priority.

Cell:

$$\frac{Q' \xrightarrow{(\alpha,r)} Q''}{Q[Q'] \xrightarrow{(\alpha,r)} Q[Q'']} \quad (Q =_a Q')$$

Transition:

$$\frac{M_{\mathbf{P}} \xrightarrow{(\alpha,r)} M'_{\mathbf{P}}}{(\ldots, M_{\mathbf{P}}, \ldots) \xrightarrow{(\alpha,r)} (\ldots, M'_{\mathbf{P}}, \ldots)} (\alpha \in \mathcal{A}_t)$$

Enabling:

$$\frac{Q \xrightarrow{(\alpha,r_1)} Q' \qquad \mathbf{P_i} \xrightarrow{(\boldsymbol{\alpha},r_2)} \mathbf{P_j}}{(.., \mathbf{P_i}[.., Q, ..], .., \mathbf{P_j}[.., \_, ..], ..) \xrightarrow{(\boldsymbol{\alpha}, R)}_{\pi(\alpha)} (.., \mathbf{P_i}[.., \_, ..], .., \mathbf{P_j}[.., Q', ..], ..)} (\alpha \in \mathcal{A}_f)$$

Firing:

$$\frac{M \xrightarrow{(\alpha,r)}_n M' \qquad M \xrightarrow{(\beta,s)}_m M''}{M \xrightarrow{(\alpha,r)} M'} (n \geq m)$$

Fig. 2. Additional semantic rules for PEPA nets

3.1 The net bisimulation relation

In this section we define a bisimulation relation for PEPA nets called *net bisimulation*. This relation is important both in theory and in practice. In the evolution of the state space of a model by our tool we only store states up to net bisimulation, i.e. we carry out automatic aggregation over equivalent states. This provides a dramatic reduction in the state space of the model under certain conditions.

Our relation is defined in the style of Larsen and Skou [25], based on a conditional transition rate between *markings*, rather than the strong equivalence relation of PEPA which considers the transition rates between components. The *conditional transition rate* from marking M to marking M' via action type α, denoted $q(M, M', \alpha)$, is the sum of the activity rates labelling arcs connecting the corresponding nodes in the derivation graph which are labelled by the action type α. The *total conditional transition rate* from a marking M to a set of markings E is defined as

$$q[M, E, \alpha] = \sum_{M' \in E} q(M, M', \alpha)$$

Definition 2 *An equivalence relation over markings, $\mathcal{R} \subseteq M \times M$, is a* net bisimulation *if whenever $(M, M') \in \mathcal{R}$ then for all $\alpha \in \mathcal{A}$ and for all equivalence classes $E \in M/\mathcal{R}$,*

$$q[M, E, \alpha] = q[M', E, \alpha]$$

4 Examples

4.1 A mobile agent system

We present a small example to reinforce the reader's understanding of PEPA nets. In this example a roving agent visits three sites. It interacts with static software components at these sites and has two kinds of interactions. When visiting a site where a network probe is present it interrogates the probe for the data which it has gathered on recent patterns of network traffic. When it returns to the central co-ordinating site it dumps the data which it has harvested to the master probe. The master probe performs a computationally expensive statistical analysis of the data. The structure of the system allows this computation to be overlapped with the agent's communication and data gathering. The marshalling and unmarshalling costs for mobile code applications are a significant expense so overlapping this with data processing allows some of this expense to be offset.

The structure of the application is as represented by the PEPA net in Figure 3. This marking of the net shows the mobile agent resident at the central co-ordinating site. In this example the activities which can cause a firing of the net are $\mathbf{go}$ and $\mathbf{return}$.

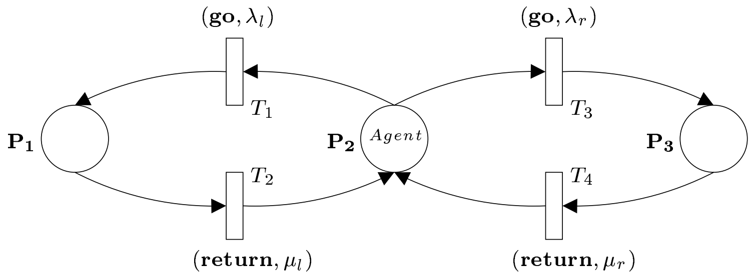


Fig. 3. A simple mobile agent system

Formally, we define the places of the net as shown in the PEPA context definitions below. We denote the local state of the context $\mathbf{P_2}$ by $\mathbf{P_2'}$. This local state is arrived at when the static component $Master$ has evolved to $Master'$.

$$\mathbf{P_1}[Agent] \stackrel{def}{=} Agent[Agent] \underset{\{interrogate\}}{\bowtie} Probe$$

$$\mathbf{P_2}[Agent] \stackrel{def}{=} Agent[Agent] \underset{\{dump\}}{\bowtie} Master$$

$$\mathbf{P'_2}[Agent] \stackrel{def}{=} Agent[Agent] \underset{\{dump\}}{\bowtie} Master'$$

$$\mathbf{P_3}[Agent] \stackrel{def}{=} Agent[Agent] \underset{\{interrogate\}}{\bowtie} Probe$$

The initial marking of the net is $(\_, Agent, \_)$. The behaviour of the components is given by the following PEPA definitions.

$$\begin{array}{rlrl}
Agent & \stackrel{def}{=} (\mathbf{go}, \lambda).Agent' & Master & \stackrel{def}{=} (dump, \top).Master' \\
Agent' & \stackrel{def}{=} (interrogate, r_i).Agent'' & Master' & \stackrel{def}{=} (analyse, r_a).Master \\
Agent'' & \stackrel{def}{=} (\mathbf{return}, \mu).Agent''' & Probe & \stackrel{def}{=} (monitor, r_m).Probe + \\
Agent''' & \stackrel{def}{=} (dump, r_d).Agent & & \quad (interrogate, \top).Probe
\end{array}$$

The derivation of the transition system underlying the model is the first step in the performance analysis of such a system. The transition system contains the specification of a Continuous-Time Markov Chain model of the system. This CTMC is solved for its stationary distribution and performance measures are calculated from that. For a model as simple as this one we can solve it simply with Gaussian elimination. We also have available solvers such as an efficient implementation of the preconditioned biconjugate gradient method.

4.2 A Jini Federation

In this example we present a model of a *Jini Federation*. The Jini architecture is designed to support spontaneous networking, allowing both the clients and the servers within a network to change dynamically. In this model we consider the discovery and use of servers by clients, and since the model is presented for illustrative purposes we consider only two distinct services, a printer and an information server.

The matching of clients and servers in Jini is managed by a *lookup server* which provides the discovery service. Servers wishing to accept service requests register with one or more lookup servers. A client with no current access to a lookup server sends a join message to a well-known IP port, and any lookup server which receives it will respond with its own address on which it receives discovery requests. The client can then access the discovery service and on discovery will be given a proxy to the service required. This allows the client to then make direct contact with the relevant server.

In order to cope with disconnection both proxies and registration are considered to be granted by the lookup server on a *leased* basis meaning that after some time the proxy or registration is assumed to have expired and subsequently must be explicitly re-established. This means that if a server has crashed, then after some period the client will no longer be granted a defunct proxy, and the proxies already held by clients will expire. The following model could be used to experiment with the trade-off created by the leasing mechanism. If the expiry rate is low, then clients can be granted proxies which will not work. However if the expiry rate is high, overhead is increased due to the delay incurred in re-registering a functioning server.

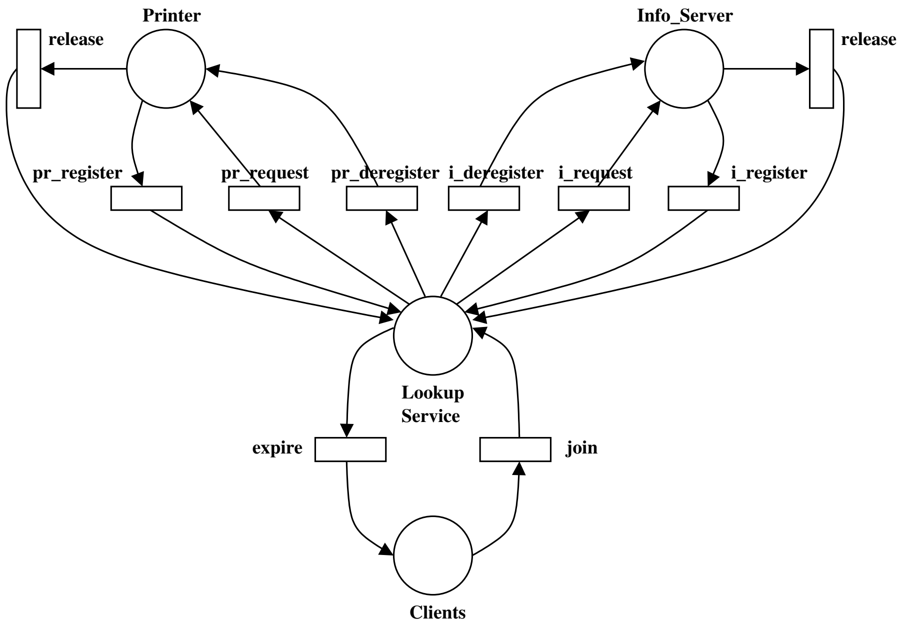


Fig. 4. The net-level description of the Jini Federation example

In the PEPA net model the net-level represents the different contexts of operation experienced by the clients. In the place $\mathbf{Clients}$, clients are assumed to be newly connected or in possession of expired leases to a discovery service, and so without access to servers. In the place $\mathbf{Lookup\_Service}$ the presence of a client indicates that the client has joined the service, while the presence of a server indicates that the corresponding service is currently registered with the lookup service. Thus the services that a client can access from the lookup service will depend on the servers which are currently registered (i.e. have tokens currently in that place). There are also places $\mathbf{Printer}$ and $\mathbf{Info\_Server}$ corresponding to each of the potential services, and the presence of a client token in these places corresponds to the client being in possession of a proxy for that service. In this model we assume that the leasing periods are set so that the probability of a client being in possession of two proxies simultaneously is negligible.

The tokens of our model are the PEPA components Cl, PS and IS. The Cl component represents the evolution of the client, containing both firings (net-level transitions) representing changes of context and transitions representing computational steps within a context.

$$\mathbf{Clients}[Cl, Cl, Cl] \stackrel{\mathit{def}}{=} (Cl[Cl] \parallel Cl[Cl] \parallel Cl[Cl])$$

$$\begin{aligned}
Cl &\stackrel{\mathit{def}}{=} (\mathbf{join}, r_1).Cl' \\
Cl' &\stackrel{\mathit{def}}{=} (\mathit{pr_lookup}, r_2).PProxy + (\mathit{i_lookup}, r_3).IProxy \\
&\quad + (\mathbf{expire}, r_4).Cl \\
PProxy &\stackrel{\mathit{def}}{=} (\mathbf{pr_request}, r_5).PProxy' \\
PProxy' &\stackrel{\mathit{def}}{=} (\mathit{print}, \top).PProxy'' \\
PProxy'' &\stackrel{\mathit{def}}{=} (\mathit{print}, \top).PProxy'' + (\mathbf{release}, r_6).Cl' \\
IProxy &\stackrel{\mathit{def}}{=} (\mathbf{i_request}, r_7).IProxy' \\
IProxy' &\stackrel{\mathit{def}}{=} (\mathit{info}, \top).IProxy'' \\
IProxy'' &\stackrel{\mathit{def}}{=} (\mathit{info}, \top).IProxy'' + (\mathbf{release}, r_6).Cl'
\end{aligned}$$

Each of the services $\mathbf{Printer}$ and $\mathbf{Info\_Server}$ is represented as the composition of a static service element ($PServe$ and $IServe$ respectively) and a token component, which acts as the service proxy with the lookup service. When a client is granted a proxy by the service proxy in the lookup service, it can make a request, represented by moving to the service context. Once there the client cooperates with the server directly until all its needs are satisfied or until the proxy expires. The role of the static service element is simply to satisfy service requests from proxied clients.

$$\mathbf{Printer}[PS, \_, \_, \_] \stackrel{\mathit{def}}{=} (PServe \parallel PS[PS]) \underset{\{print\}}{\bowtie} (PProxy[\_] \parallel PProxy[\_] \parallel PProxy[\_])$$

$$\begin{aligned}
PServe &\stackrel{\mathit{def}}{=} (\mathit{print}, \mathit{pr}).PServe \\
PS &\stackrel{\mathit{def}}{=} (\mathbf{pr_register}, r_8).PS' \\
PS' &\stackrel{\mathit{def}}{=} (\mathbf{pr_deregister}, r_9).PS + (\mathit{pr_lookup}, \top).PS'
\end{aligned}$$

$$\mathbf{Info\_Server}[IS, \_, \_, \_] \stackrel{\mathit{def}}{=} (IServe \parallel IS[IS]) \underset{\{info\}}{\bowtie} (IProxy[\_] \parallel IProxy[\_] \parallel IProxy[\_])$$

$$\begin{aligned}
IServe &\stackrel{\mathit{def}}{=} (\mathit{info}, \mathit{ir}).IServe \\
IS &\stackrel{\mathit{def}}{=} (\mathbf{i_register}, r_{10}).IS' \\
IS' &\stackrel{\mathit{def}}{=} (\mathbf{i_deregister}, r_{11}).IS + (\mathit{i_lookup}, \top).IS'
\end{aligned}$$

The lookup service itself is represented by the place $\mathbf{Lookup\_Service}$. This place has no static elements but a context for each potential server proxy and each potential client. The lookup activities (pr_lookup and i_lookup) are carried out in cooperation between a client and a server proxy, and thus require the presence of both in the $\mathbf{Lookup\_Service}$ place.

$$\mathbf{Lookup\_Service}[\_,\_,\_,\_,\_,] \stackrel{\mathit{def}}{=} (PS[\_] \parallel IS[\_]) \underset{\{pr\_lookup, i\_lookup\}}{\bowtie} (Cl[\_] \parallel Cl[\_] \parallel Cl[\_])$$

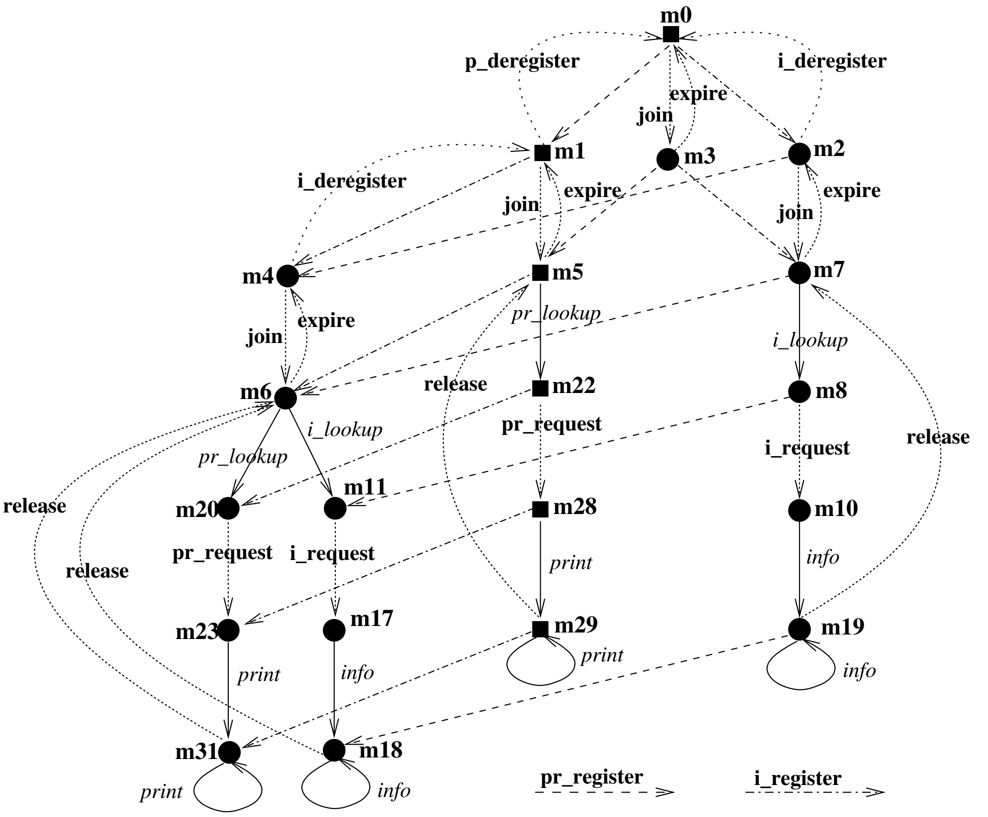
We have run this PEPA net model for different numbers of clients and servers. In the simplest case of one client, one printer server, one information server, and when considering the following ordering for the net places ($\mathbf{Lookup\_Service}$, $\mathbf{Printer}$, $\mathbf{Info\_Server}$, $\mathbf{Clients}$), the initial marking $\mathbf{m_0}$ is

$$\mathbf{m_0} = (\mathbf{Lookup\_Service}[\_,\_,\_], \mathbf{Printer}[PS,\_], \mathbf{Info\_Server}[IS,\_], \mathbf{Clients}[Cl])$$

The model has 32 states, 20 transitions and 88 firings; a portion of the transition system, describing some possible accesses to the printer server and some possible accesses to the information server is shown in Figure 5. For readability, many firings and transitions are omitted from the graph as well as activity rates; solid lines represent local transitions while dotted and dashed lines represent firings.

Figure 5 also describes the markings in the path whose states are drawn as black squares. This path shows one possible evolution of the client contacting the printer server. Starting from the initial marking $\mathbf{m_0}$, the printer server registers with the lookup server. This is obtained by moving the component PS from the place $\mathbf{Printer}$ to the place $\mathbf{Lookup\_Service}$, filling the appropriate empty cell there ($\mathbf{m_1}$). Afterwards, the client joins the lookup server, as shown in marking $\mathbf{m_5}$ where the component Cl has left place $\mathbf{Client}$ and has reached the appropriate empty cell into $\mathbf{Lookup\_Service}$. Now the internal transition pr_lookup can take place ($\mathbf{m_{22}}$). The client is granted the printer proxy and can move to the place $\mathbf{Printer}$ ($\mathbf{m_{28}}$). Finally, the request can be satisfied thanks to the cooperation on action $print$ performed by the two components resident in the same place ($\mathbf{m_{29}}$). A similar evolution is possible for the path in the right part of the graph connecting markings $\mathbf{m_0}, \mathbf{m_2}, \mathbf{m_7}, \mathbf{m_8}, \mathbf{m_{10}}, \mathbf{m_{19}}$ when considering the information server instead of the printer server.

The right part of the graph describes a situation in which both servers have registered with the lookup server and the client can choose among one of them (see marking $\mathbf{m_6}$ in Figure 5).



$$\mathbf{m_0} : (\mathbf{Lookup\_Service}[\_,\_,\_], \mathbf{Printer}[PS,\_], \mathbf{Info\_Server}[IS,\_], \mathbf{Clients}[Cl])$$

$$\mathbf{m_1} : (\mathbf{Lookup\_Service}[PS',\_,\_], \mathbf{Printer}[\_,\_], \mathbf{Info\_Server}[IS,\_], \mathbf{Clients}[Cl])$$

$$\mathbf{m_5} : (\mathbf{Lookup\_Service}[PS',\_,Cl'], \mathbf{Printer}[\_,\_], \mathbf{Info\_Server}[IS,\_], \mathbf{Clients}[\_])$$

$$\mathbf{m_{22}} : (\mathbf{Lookup\_Service}[PS',\_,PProxy], \mathbf{Printer}[\_,\_], \mathbf{Info\_Server}[IS,\_], \mathbf{Clients}[\_])$$

$$\mathbf{m_{28}} : (\mathbf{Lookup\_Service}[PS',\_,\_], \mathbf{Printer}[\_,PProxy'], \mathbf{Info\_Server}[IS,\_], \mathbf{Clients}[\_])$$

$$\mathbf{m_{29}} : (\mathbf{Lookup\_Service}[PS',\_,\_], \mathbf{Printer}[\_,PProxy''], \mathbf{Info\_Server}[IS,\_], \mathbf{Clients}[\_])$$

$$\mathbf{m_6} : (\mathbf{Lookup\_Service}[PS',IS',Cl'], \mathbf{Printer}[\_,\_], \mathbf{Info\_Server}[\_,\_], \mathbf{Clients}[\_])$$

Fig. 5. Partial transition system of the Jini Federation example and marking descriptions of selected states

5 Relating PEPA nets to Petri nets and PEPA

If they were to be viewed purely formally as high-level description languages for specifying continuous-time Markov chains, then PEPA nets, stochastic Petri nets and the PEPA stochastic process algebra would be considered to be equally expressive. That is to say, for a given CTMC C, it is possible to construct a high-level model in each of these three formalisms such that the underlying CTMC derived from the model is isomorphic to C.

In practice, the three languages present different sets of conceptual tools to the modeller. From the pragmatic perspective of a performance modeller who wishes to reliably encode a high-level model of a particular system then there might be reasons to select one of the languages instead of the others for this particular modelling study. In the remainder of this section we compare modelling with PEPA nets with modelling with Petri nets and the PEPA stochastic process algebra.

5.1 Relating PEPA nets to Petri nets

To illustrate the difference between PEPA nets and Petri nets we first show how to represent an ordinary k-safe stochastic Petri net as a PEPA net. In a classical stochastic Petri net tokens are indistinguishable. We can replicate this in a PEPA net by having only a single class of tokens which have only one (PEPA) state. The definition of such a token would also need to always permit firings of the net to take place. We define these tokens by summing over all of the transition activity names, for all of the transitions of the net ($tn_i \in T$).

$$\mathit{Token} \stackrel{\mathit{def}}{=} \sum_{tn_i \in T} (tn_i, \top).\mathit{Token}$$

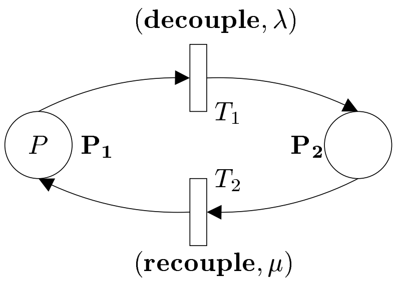
To define a k-safe stochastic Petri net with a PEPA net we then need simply to specify the places of the net as being capable of storing up to k of these tokens, and making no use of static components.

$$\mathbf{P_i}[\mathit{Token}, \ldots, \mathit{Token}] \stackrel{\mathit{def}}{=} \mathit{Token}[\mathit{Token}] \parallel \cdots \parallel \mathit{Token}[\mathit{Token}]$$

This reconstruction of ordinary k-safe stochastic Petri nets from PEPA nets points to the difference between the two formalisms. A PEPA net can be viewed as a Petri net where the tokens are *programmable*. The tokens of a PEPA net have state, can count, can observe activities, and can even refuse to be fired from the place where they reside. We believe that this gives the PEPA net modeller a novel conceptual modelling tool which can be used to express natural descriptions of systems with active, stateful mobile agents.

The relationship between PEPA nets and PEPA is straightforward. A PEPA net with only one place and no transitions is simply a PEPA stochastic process algebra model. To explain how a PEPA net can offer added expressive power we consider a PEPA net with more than one place, as in Figure 6.

$$\mathbf{P_1}[P] \stackrel{\mathit{def}}{=} P[P] \underset{L}{\bowtie} Q$$



$$\mathbf{P_2}[P] \stackrel{\mathit{def}}{=} P[P]$$

Fig. 6. Tokens in a PEPA net can decouple from a static component

In this model a token of type P moves between place $\mathbf{P_1}$ and place $\mathbf{P_2}$. In doing so it decouples itself from a static component Q, located at $\mathbf{P_1}$. The token thereby moves out of the scope of the cooperation set L. Cooperation sets are used to configure copies of components, coupling them to communication partners. In this way they restrict the behaviour of a component, requiring it to perform some activities (those in the cooperation set) only if they have a partner who is able to cooperate in performing them. In the example in Figure 6 if Q is unwilling to perform some of these activities then the behaviour of P will be restricted. Even if Q is willing to perform all of the activities in the cooperation set L then it can still influence the rate at which they are performed. In contrast when the token P is resident in $\mathbf{P_2}$ then it is subject to no such restriction and can perform all of its activities at the rates which it itself specifies.

This concept cannot be expressed in a PEPA stochastic process algebra model. The cooperation sets used in a PEPA model impose a static communication topology on the model. In contrast a PEPA net has dynamically varying communication structure and, in consequence, a given action in a component might sometimes be performed in isolation and sometimes be performed in cooperation. The ability to express the concept of dynamically varying communication structure offers an additional conceptual tool to the performance modeller which is not available when modelling in PEPA.

6 Case study: A hierarchical cellular network

Hierarchical cellular networks consist of two tiers of cells, a macrocellular level overlaying a microcellular one. This means that a geographical point is potentially covered by two levels of cells and a user can be assigned to one of these two levels. Generally, such a network architecture takes into account two user classes according to their speed: the pedestrians and the vehicles. Usually, macrocells are deployed in rural areas and have good properties for fast users, whereas the microcellular network concept has been developed to satisfy the high traffic demand in the dense urban regions and is better suited to providing for services requiring low mobility.

The objective of the hierarchical architecture is to take advantage of the wide coverage of macrocells and the traffic capacity of microcells. However, this architecture suffers from the major drawback of microcellular systems, which is the *handoff problem*.

The handoff is defined as the change of radio channel used by a wireless terminal. For example, if a subscriber crosses a cell boundary to move to an adjacent cell while the call is in progress, the call must be handed off to the new cell in order to provide uninterrupted service to the mobile subscriber. If the new cell does not have enough channels to support the handoff, the call is dropped. So, the handoff procedure has an important effect on the performance of the system and the probability of forced call termination must be limited because from the point of view of a mobile user, forced termination of an ongoing call is less acceptable than blocking a new call.

6.1 Topology and Assumptions

The topology of the hierarchical network we study is depicted in Figure 7. In this topology, each macrocell overlays a cluster of seven microcells.

As in an hexagonal model each microcell has six neighbouring cells, we consider a microcell cluster model composed of a central microcell surrounded by six peripheral cells (Figure 7). We consider the Fixed Channel Allocation scheme (FCA) [22], where a constant number B of channels is distributed among the two layers of cells.

Although hierarchical cellular networks are studied, we consider only one class of users. As we focus our study on dense urban regions, we consider only services which require low mobility such as slow-moving vehicles or pedestrians. However, we consider two types of customers within the network, the new calls and the handover calls. Thus, external arrivals to a microcell consist either of new calls

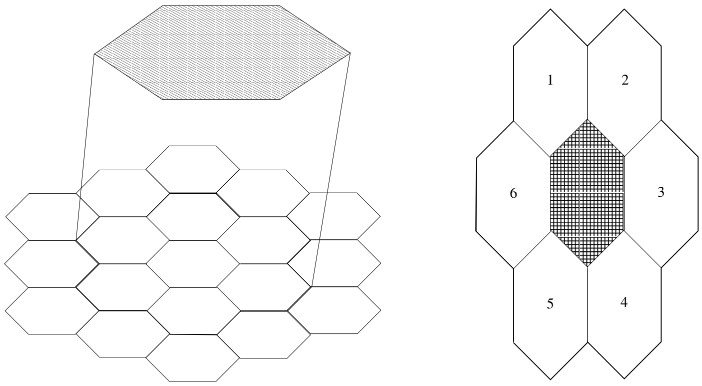


Fig. 7. The hierarchical cellular network and cluster model

or handover requests coming from its adjacent microcells or from the macrocell. Similarly, external arrivals to a macrocell consist either of new calls or handover requests coming from either its adjacent macrocells or from one of the microcells of the cluster. Thus, new calls can be assigned at either the microcell or the macrocell level.

For the microcell level, we consider the overflow strategy with reversible capability. Therefore, a request, which could be either a new call or a handover, initiated at the microcell level, is served in its originating microcell if a channel is available. Otherwise, according to the overflow strategy, the request is overflowed to the upper layer and is satisfied at this level if a channel is free. In the case where all channels are busy at both levels, the request is blocked (new call) or dropped (handover). Similarly, when a request is first initiated at the macrocellular level and there is no available channel, the request is transferred to the microcell level where it may be satisfied if a channel is available; otherwise, it is dropped.

We consider a homogeneous system in statistical equilibrium. Thus any microcell overlaid by a macrocell has statistically the same behaviour as any other microcell overlaid by a macrocell. We can then analyse the overall system by focusing on a given cell under the condition that the neighbouring cells exhibit their typical random behaviour independently. Moreover, we assume that any geographical point of this network is covered by both microcellular and macrocellular levels, and that the whole area is crossed randomly by mobile users, according to an uniform traffic matrix.

This system is studied under the usual Markovian assumptions. It is assumed that the average new call arrival rate and the handover rate in each cell in the network follow a Poisson distribution. The amount of time that a user remains within a coverage cell of a given base station (called *dwell-time*) is modelled by a service time which is exponentially distributed. In the next section, we present the PEPA net model corresponding to this system.

6.2 PEPA net model

In the hierarchical cellular network, all microcells of the network are assumed to have exactly the same behaviour. Moreover, all customers have the same behaviour and do not change behaviour according to the microcell in which they evolve. Therefore, the PEPA net model is based on the description of the behaviour of one microcell (here the central microcell) and its links with the cells surrounding it. This model is depicted in Figure 8 where each cell is represented by a place $\mathbf{MICRO}_j$, $1 \leq j \leq 7$, in which the wireless network customers evolve. Note that $\mathbf{MICRO_7}$ represents the central microcell. Similarly, the macrocell is modelled using a place denoted $\mathbf{MACRO}$. Moreover, we use a place denoted $\mathbf{NETENV}$ to model what we call *the network environment*. This part of the network is assumed to generate the new calls and absorb the dropped or terminated ones.

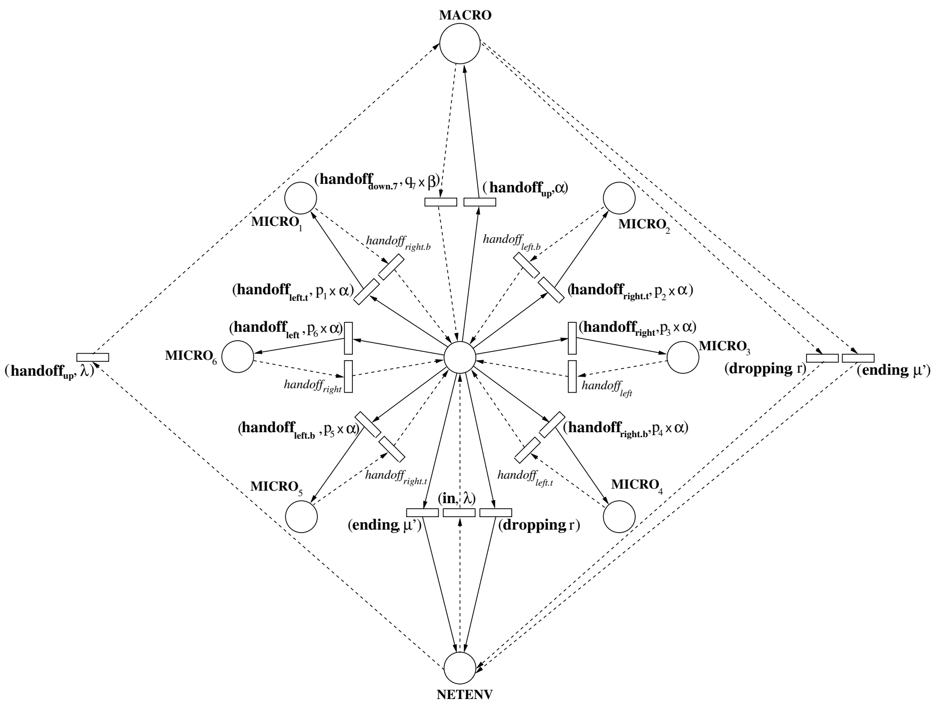


Fig. 8. The PEPA net of a basic cluster

The places of the PEPA net are defined as follows.

$$\mathbf{MICRO}_{j=1,\ldots,7}[\_,\ldots,\_] \stackrel{\text{def}}{=} (\mathit{Client}[\_] \parallel \ldots \parallel \mathit{Client}[\_]) \underset{\{service\}}{\bowtie} \mathit{Micro}_j$$

$$\mathbf{MACRO}[\_,\ldots,\_] \stackrel{\text{def}}{=} (\mathit{Client}[\_] \parallel \ldots \parallel \mathit{Client}[\_]) \underset{\{service\}}{\bowtie} \mathit{Macro}$$

$$\mathbf{NETENV}[C_1,\ldots,C_N] \stackrel{\text{def}}{=} \mathit{Client}[C_1] \parallel \cdots \parallel \mathit{Client}[C_N]$$

The behaviour of a network customer is modelled using a dynamic component $Client$. Formally, component $Client$ is defined as shown below.

$$Client \stackrel{\text{def}}{=} (\mathbf{in}, \lambda).Client_1 + (\mathbf{handoff_{up}}, \lambda).Client_1$$

$$\begin{aligned}
Client_1 \stackrel{\text{def}}{=} & (\mathbf{handoff_{right}}, p_3 \times \alpha).Client_1 + (\mathbf{handoff_{left}}, p_6 \times \alpha).Client_1 \\
+ & (\mathbf{handoff_{right.b}}, p_4 \times \alpha).Client_1 + (\mathbf{handoff_{left.b}}, p_5 \times \alpha).Client_1 \\
+ & (\mathbf{handoff_{right.t}}, p_2 \times \alpha).Client_1 + (\mathbf{handoff_{left.t}}, p_1 \times \alpha).Client_1 \\
+ & (\mathbf{handoff_{up}}, \alpha).Client_1 \\
+ & (\mathbf{handoff_{down.1}}, q_1 \times \beta).Client_1 + (\mathbf{handoff_{down.2}}, q_2 \times \beta).Client_1 \\
+ & (\mathbf{handoff_{down.3}}, q_3 \times \beta).Client_1 + (\mathbf{handoff_{down.4}}, q_4 \times \beta).Client_1 \\
+ & (\mathbf{handoff_{down.5}}, q_5 \times \beta).Client_1 + (\mathbf{handoff_{down.6}}, q_6 \times \beta).Client_1 \\
+ & (\mathbf{handoff_{down.7}}, q_7 \times \beta).Client_1 \\
+ & (service, \mu).Client_2 + (\mathbf{dropping}, r).Client
\end{aligned}$$

$$Client_2 \stackrel{\text{def}}{=} (\mathbf{ending}, \mu').Client$$

The new calls arrival process (external arrival) to a microcell is represented by an **in** activity. If all channels of the microcell, covering the area where the new customer is, are busy then the new call has to be transferred to the macrocell. This transfer is modelled using the activity $\mathbf{handoff_{up}}$. If all channels of the macrocell are busy too, this transfer fails and the new call is blocked. As shown in Figure 8, **in** and $\mathbf{handoff_{up}}$ are activities which can cause the firing of the net. The firing **in** has priority over the firing $\mathbf{handoff_{up}}$, capturing the fact that a call will be allocated a channel in the microcell if there is one available. The rate of these activities is the same and is equal to the external arrival rate to the microcell, λ.

The relative priorities of the firings of this model are listed below.

$$\left\{ \begin{array}{l} \mathbf{in}, \mathbf{handoff_{right}}, \mathbf{handoff_{left}}, \\ \mathbf{handoff_{down.3}}, \mathbf{handoff_{down.6}}, \mathbf{handoff_{down.7}} \end{array} \right\} > \mathbf{handoff_{up}} > \mathbf{dropping}$$

A customer may terminate its communication during its sojourn in a microcell. This is modelled using the activity $service$. The execution of this activity is followed by the execution of activity **ending** which specifies that the customer is leaving the network. The customer may also cross the cell boundary to move to an adjacent cell during its communication. The call must then be handed off to the new cell in order to provide uninterrupted service. For that a handoff request is generated to ask for another channel in the destination microcell. In our model, this aspect of the communication is modelled using the activity $\mathbf{handoff}_{dir}$ where dir may be one of the following values: $\mathbf{right}$, $\mathbf{left}$, $\mathbf{right.b}$, $\mathbf{left.b}$, $\mathbf{right.t}$ or $\mathbf{left.t}$, according to the direction taken by the customer during his communication. The rate of this activity is $p_i \times \alpha$ where p_i, $i = 1, \ldots, 6$, is the probability associated with each possible destination among the adjacent cells. If the handoff procedure does not succeed because all channels of the target microcell are busy, the network tries to transfer the call to the macrocell overlapping the area. This is modelled using activity $\mathbf{handoff_{up}}$. If once again all channels of the macrocell are busy, the transfer fails and the call is dropped. This is modelled using activity $\mathbf{dropping}$. As previously, $\mathbf{handoff_{up}}$ has lower priority, and here $\mathbf{dropping}$ has lowest priority.

For the sake of readability of Figure 8, the rates of the activities on the transitions from the adjacent microcells to the central microcell are omitted. These rates may be easily deduced from the transition with the same activity name from the central microcell to the opposite peripheral microcell. For example, the rate of activity $\mathbf{handoff_{right}}$ on the transition from $\mathbf{MICRO_6}$ to $\mathbf{MICRO_7}$, the central microcell, can be deduced from the activity on the transition from $\mathbf{MICRO_7}$ to $\mathbf{MICRO_3}$ and is therefore $p_3 \times \alpha$.

A customer in the macrocell has exactly the same behaviour except that activity $\mathbf{handoff_{up}}$ is no longer enabled. Instead, as the customer can be transferred to the microcellular level, we include an activity $\mathbf{handoff}_{\mathbf{down}.j}$ where j is the microcell number where the call is transferred. The rate of this activity is $q_j \times \beta$ where q_j, $j = 1, \ldots, 7$, is the probability to transfer the call to microcell j. Conversely, this activity cannot be executed by a customer in a microcell.

The services provided by a microcell j, $1 \leq j \leq 7$, and the macrocell to their customers are modelled using static components denoted $Micro_j$ and $Macro$ respectively, and are formally defined as follows:

$$Micro_j \stackrel{\text{def}}{=} (service, \top).Micro_j \qquad 1 \leq j \leq 7$$

$$Macro \stackrel{\text{def}}{=} (service, \top).Macro$$

Activity $service$ is the only activity on which components $Client$ and $Micro_j$, on one hand, and $Client$ and $Macro$, on the other hand, must synchronise.

We assume that in the initial state all macrocell and microcells channels are available and all potential network customers (N) are in place $\mathbf{NETENV}$.

As already remarked, in order to capture the exact behaviour of the network, we have assigned priorities to the firings. For example, in the net the transition labelled $\mathbf{in}$ has a higher priority than that labelled $\mathbf{handoff_{up}}$. Thus a customer can fire $\mathbf{handoff_{up}}$ only if the firing of activity $\mathbf{in}$ is not possible, i.e. there are no slots available in the place $MICRO_i$.

In the case where each cell has two channels ($B = 16$) and $N = 7$, the model, once aggregated, has 49416 states, 147274 transitions and 571362 firings.

This study has been inspired by the work presented in [12]. In that paper the authors investigate the performance of the hierarchical cellular network using PEPA. They assume a Manhattan model [23] in which the reuse pattern is composed of a macrocell overlying a cluster of five microcells.

The PEPA model obtained consists of six components, one for each cell. As the customers are not explicitly modelled, all the information about the activities they can perform in a cell are captured by the component representing the cell. This results in a model where the mobility of the customers is observable only from the point of view of the cell.

In our PEPA net model (Figure 8), a customer is explicitly modelled using a component which captures the activities that it can perform. So when a customer moves, which is explicitly represented by the firings, its activities “move” with it. This gives another dimension to the model as the mobility of a customer is represented from both the point of view of the cell (place) and the customer ($Client$). Moreover, the representation of the cell only changes in the sense that its slots are filled when channels are in use; it does not need to explicitly record the state of the customers. Thus in many ways the PEPA net model provides a more natural representation of the system.

Furthermore the PEPA net model offers more compositionality in the sense that it would be possible to extend the model to represent a number of interacting macrocells, simply by modifying the net-level description and the initial marking of the model. In the PEPA model the whole model would need to be re-written to achieve this.

7 Implementation

The PEPA stochastic process algebra is supported by a range of tools including the PEPA Workbench [13] and the Möbius Modelling Framework [10]. We have implemented the PEPA nets formalism as an extension of the PEPA Workbench. The PEPA modelling tools, together with user documentation and papers and example PEPA models are available from the PEPA Web page which is located at `http://www.dcs.ed.ac.uk/pepa`.

The PEPA Workbench exists in two distinct versions. The first version is an experimental research tool which is coded in the functional programming language Standard ML [28]. The second is a re-implementation of this in the Java programming language. These are known as “the ML edition” and “the Java edition” respectively.

Standard ML and Java have very different strengths. For a visual language such as the notation of Petri nets the Java language’s visualisation capabilities would suit the task much better than Standard ML. Further, there are existing Java tools for Petri nets which could be extended to provide an implementation of PEPA nets. After initial experimentation with the Standard ML version of the PEPA Workbench for PEPA nets, a graphical presentation of PEPA nets could be incorporated into the Java version of the Workbench. This implementation plan is ongoing, but at an early stage.

The Standard ML language is well suited to implementing symbolic processing applications and provides built-in support for describing datatypes such as those needed to present the abstract syntax of a formal language such as PEPA. These features made it possible to rapidly adapt the routines for generating PEPA derivation graphs to generate derivation graphs for PEPA nets. The compact transition rules presented in Figure 2 look simple on the page but they proved to be a challenge to implement efficiently. Here again, the higher-order features of the Standard ML programming language proved to be useful in allowing us to form function closures from higher-order functions which fixed some of their formal parameters. This allowed us to unroll the derivation graph for the PEPA nets model without suffering a performance penalty due to accumulated parameter information.

The use of the PEPA Workbench for PEPA nets is illustrated in Figure 9. The input language of the tool is an extension of the concrete syntax used for storing PEPA language models. The topology of the net is specified by providing a textual description of the places and the arcs connecting them.

```
PEPA Workbench for PEPA Nets Version 0.81.1 "Granby Road"
[ Setting model aggregation on ]
[ Use of priorities is enabled ]
Compiling the model
Generating the derivation graph
The model has 49416 states
The model has 147274 transitions
The model has 571362 firings
Writing the hash table file to model3.hash
Exiting PEPA Workbench.
```

Fig. 9. The PEPA Workbench for PEPA nets processing the hierarchical cellular network example for seven clients

7.1 Model aggregation in the PEPA Workbench

As firings or transitions occur in the exploration of a PEPA net model, new syntactic terms are generated for these one-step derivatives of the model according to the semantics of the language as presented earlier. Some of the one-step derivatives which are generated will be syntactically distinct but semantically identical. We have extended our previously published algorithm for computing canonical forms of these terms [15] from the PEPA stochastic process algebra to be used on PEPA nets. We have extended the PEPA Workbench for PEPA nets to apply this canonicalisation on-the-fly, replacing derivatives with their canonicalised equivalents. We quotient the state space of the system with respect to this canonicalisation and aggregate the rates at which transitions into aggregated states occur.

This aggregation gives a dramatic reduction in the number of states of a PEPA net model. For example, consider a PEPA net with two places and a token T which can fire to move between the places. The net has the following initial marking.

$$(T[\_] \parallel T[\_] \parallel T[\_] \parallel T[\_], \quad T[T] \parallel T[T] \parallel T[T] \parallel T[T])$$

This net has 70 states and 640 firings without applying our aggregation algorithm and 5 states and 8 firings if the algorithm is applied.

As a PEPA net evolves, according to the operational semantics presented in Section 3, there is no opportunity for it to add or take away places or transitions. As a consequence of this a PEPA net and its one-step derivatives will be structurally isomorphic, i.e. will have the same net structure. We have exploited this to include an optimisation in our implementation of aggregation based on the net bisimulation relation as defined in Section 3.1. At each step we simply canonicalise the representation within each place marking instead of canonicalising at the marking level. Applying this aggregation avoids the generation of terms which could not be derivatives of the current term according to the operational semantics.

8 Related work

As mentioned in the Introduction, Petri nets have previously been combined with several other modelling formalisms. In particular in the arena of performance modelling, several proposals have been made to integrate queues or queueing networks with stochastic Petri nets [3,2,17]. However, these proposals differ from our own in that the two formalisms may be regarded as adjuncts to one another, with one providing delays to the other, rather than integrated into a single formalism.

Somewhat closer to our own work is Valk's work on *Elementary Object Systems* [31].

In this work an extension of Petri nets is presented in which the tokens circulating in the net structure (called the *System net*) are themselves Petri nets (termed *Object nets*). Object nets move like ordinary tokens and they can change their markings but not their structure. Three different types of transitions are defined. Transitions occurring in the Object net (i.e. in the marking) are called *system autonomous* and represent the object internal behaviour. An *interaction* takes places when both the Object and the System net enable transitions with the same attached label. A third type of transition causes a change in the System net only and it is called *transport*. In PEPA nets we do not allow such transitions, since a firing cannot occur without modifying the state of a component.

Despite the superficial similarities there are some quite strong differences between the work on PEPA nets and that on Elementary Object Systems (EOS). Fundamentally, EOS are without any timing considerations, other than the relative timing imposed by the Petri net causality relation. In PEPA nets, in addition to this implicit timing information we have explicit time delays integrated into behaviour at both the net level and the token level. Moreover the motivations for the works are distinct. Valk's work is motivated by a desire to provide a fundamental model of object-oriented programming, and the development of EOS has been strongly influenced by this goal. For example, two different semantics are provided to characterise the dynamic behaviour of EOS, called *value* and *reference* semantics, corresponding to differing approaches in object-oriented programming languages. Our motivation has been to develop a convenient high-level modelling language for Markov processes, for systems in which state changes can be regarded as proceeding in two ways.

As mentioned earlier, one of the domains of application envisaged for PEPA nets is the domain of mobile computation. Several process calculi have been developed specifically for this domain, the most notable being the π-calculus [27] and the calculus of mobile ambients [8]. The π-calculus, and Priami's subsequent extension, the stochastic π-calculus, have a very different style of representing systems [5], which does not satisfy our criterion of clearly separating state changes into distinct types related, in the case of mobile computation, to concepts of location and mobility. In this respect our formalism is closer to the work on mobile ambients.

The calculus of mobile ambients is intended to capture notion of *locations*, *mobility* and *authority for movement*. This is achieved by introducing the concept of *ambient*, i.e. a bounded place where computation happens. An ambient is denoted $n[P]$, where n is the name of the ambient and P is the process running inside it. Ambients can be nested into other ambients and can be moved as a whole. Mobility primitives are provided by considering *capabilities*: it is possible to *enter* into another ambient, to *exit* from an ambient, to *open* an ambient. Processes are executed within ambients and a simple asynchronous communication mechanism that works within a single ambient is chosen. Communication across ambients is modelled as the movement of 'messenger' agents that must cross ambient boundaries.

The most pronounced differences between PEPA nets and the ambient calculus are the lack of timing information in the ambient calculus and the ability to nest ambients which gives a hierarchical structure to locations which cannot be matched by the places in PEPA nets. It is an area for future work to study the differences and similarities between these formalisms more closely.

In the performance arena our work has some resonances with earlier work by Buchholz [6,7]. In this work Buchholz considered solution techniques for Markov processes which were specified in an hierarchical fashion, meaning that a number of component Markov processes were connected via a higher-level model. The higher-level model determines how entities move between lower-level models. The lower-level models in Buchholz's work were principally intended to be queueing networks, so that the entities (customers) which move between lower-level models are themselves without state and have no power to evolve independently. The primary focus of the papers [6,7] is finding efficient solution techniques for hierarchical models. Thus it is anticipated that it will be a fruitful area for future work to investigate how Buchholz's framework may be modified to accommodate PEPA nets.

9 Conclusions and further work

The PEPA nets formalism is new and, as yet, relatively unproven. It is our belief that it can provide a suitable framework for the description of performance models of systems which have distinct notions of changes of state. Our experience with the PEPA formalism has been that the combination of a well-defined formal semantics for the language and the availability of a range of tools to implement the language has enabled us and others to use it effectively in the performance modelling and analysis of systems. By following a similar development path we would hope that the PEPA nets formalism could also prove to be useful.

The combination of a process algebra with a Petri net presents many opportunities to import developments from the Petri net community into the practices in the process algebra community. Further, it is to be hoped that these developments can be imported more directly through the use of a Petri net with algebraic terms as tokens than if one was to rework them and to re-apply them in the process algebra context.

We have defined a language which provides an extension to the PEPA stochastic process algebra by allowing a number of distinct PEPA models to be arranged into a net. These models communicate via the transfer of tokens from one place to another. We have implemented this new language and applied it to some case studies. In the light of additional experience gained from further case studies it could be possible that we would discover that other language constructs would be helpful to the modeller.

One possibility would be an independent evolution of the net system, akin to the *transport transitions* of Elementary Object Systems, where tokens are forcibly moved from one place to another without the option to refuse this or change state in transit. We have omitted this feature at present because it seems at odds with the process algebra notion of every component having behaviour. Additional language design decisions and extensions remain as future work.

Other future work includes the continued development of our implementation of the PEPA Workbench for PEPA nets. The additional of a graphical editor for PEPA nets is a likely next step with this tool.

Together with Norman and Parker at Birmingham we have recently extended the PRISM probabilistic symbolic model checker [24] to support the PEPA stochastic process algebra as an additional modelling language. An extension of that work to support PEPA nets would greatly enhance our ability to experiment with models on a large scale.

Acknowledgements

Stephen Gilmore, Jane Hillston and Leïla Kloul are supported by the DEGAS (Design Environments for Global ApplicationS) IST-2001-32072 project funded by the FET Proactive Initiative on Global Computing. Our thanks go to Giuliana Franceschinis for her advice on coloured Petri nets. The authors are grateful to the anonymous referees for their comments on the paper. An earlier version of this paper appeared as [16].

References

[1] M. Ajmone Marsan, A. Bobbio, and S. Donatelli. Petri nets in performance analysis: An introduction. In Reisig, W. and Rozenberg, G., editors, *Lectures on Petri Nets I: Basic Models*, volume 1491 of *LNCS*, pages 211–256. Springer-Verlag, 1998.

[2] F. Bause. Queueing Petri nets—a formalism for the combined qualitative and quantitative analysis of systems. In *5th International Workshop on Petri Nets and Performance Models*, pages 14–23, Toulouse, France, October 1993.

[3] M. Becker and H. Szczerbicka. PNiQ: Integration of queuing networks in generalized stochastic Petri nets. *IEE Proceedings—Software*, 146(1):27–33, February 1999. Special issue of the proceedings of the Fourteenth UK Performance Engineering Workshop.

[4] M. Bernardo and R. Gorrieri. A tutorial on EMPA: a theory of concurrent processes with nondeterminism, priorities, probabilities and time. *Theoretical Computer Science*, 202:1–54, 1998.

[5] L. Brodo, S. Gilmore, J. Hillston, and C. Priami. A stochastic π-calculus semantics for PEPA nets. In *Proc. of the Workshop on Process Algebras and Stochastically Timed Activities*, pages 1–17. LFCS, University of Edinburgh, June 2002.

[6] P. Buchholz. Hierarchical Markov Models — symmetries and aggregation. *Performance Evaluation*, 22:93–110, 1995.

[7] P. Buchholz. Multi-level solutions for structured Markov chains. *SIAM Journal on Matrix Analysis and Applications*, 22(2):342–357, 2000.

[8] L. Cardelli and A.D. Gordon. Mobile ambients. In M. Nivat, editor, *Foundations of Software Science and Computational Structures*, volume 1378 of *LNCS*, pages 140–155. Springer Verlag, 1998.

[9] G. Clark. *Techniques for the Construction and Analysis of Algebraic Performance Models*. PhD thesis, The University of Edinburgh, 2000.

[10] G. Clark, T. Courtney, D. Daly, D. Deavours, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. Webster. The Möbius modeling tool. In *Proceedings of the 9th International Workshop on Petri Nets and Performance Models*, pages 241–250, Aachen, Germany, September 2001.

[11] S. Donatelli, J. Hillston, and M. Ribaudo. A comparison of Performance Evaluation Process Algebra and Generalized Stochastic Petri Nets. In *Proc. 6th International Workshop on Petri Nets and Performance Models*, Durham, North Carolina, 1995.

[12] J-M. Fourneau, L. Kloul, and F. Valois. Performance modelling of hierarchical networks using PEPA. *Performance Evaluation*, 50:83–99, 2002.

[13] S. Gilmore and J. Hillston. The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling. In *Proceedings of the Seventh International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, number 794 in Lecture Notes in Computer Science, pages 353–368, Vienna, May 1994. Springer-Verlag.

[14] S. Gilmore, J. Hillston, and L. Recalde. Elementary structural analysis for PEPA. Technical Report ECS-LFCS-97-377, Laboratory for Foundations of Computer Science, Department of Computer Science, The University of Edinburgh, 1997.

[15] S. Gilmore, J. Hillston, and M. Ribaudo. An efficient algorithm for aggregating PEPA models. *IEEE Transactions on Software Engineering*, 27(5):449–464, May 2001.

[16] S. Gilmore, J. Hillston, and M. Ribaudo. PEPA-coloured stochastic Petri nets. In K. Djemame and M. Kara, editors, *Proceedings of the Seventeenth UK Performance Engineering Workshop*, pages 155–166, University of Leeds, July 2001.

[17] B.R. Haverkort, I.G. Niemegeers, and P Veldhuyzen van Zanten. DyQNtool—a performability modelling tool based on the dynamic queueing network concept. In G. Balbo and G. Serazzi, editors, *Modelling Techniques and Tools for Computer Performance Evaluation*, pages 181–195. North-Holland, 1992.

[18] H. Hermanns. *Interactive Markov Chains And the Quest for Quantified Quality*, volume 2428 of *LNCS*. Springer-Verlag, 2002.

[19] H. Hermanns, U. Herzog, V. Mertsiotakis, and M. Rettelbach. Exploiting stochastic process algebra achievements for generalized stochastic Petri nets. In *Proc. of 7th International Workshop on Petri Nets and Performance Models*, Saint Malo, June 1997. IEEE CS Press.

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

[21] R. Hopkins and P. King. A visual formalism for the composition of stochastic Petri nets. In T. Field, P.G. Harrison, J. Bradley, and U. Harder, editors, *Proceedings of the 12th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, volume 2324 of *LNCS*, pages 239–258, London, 2002. Springer.

[22] I. Katzela and M. Naghshineh. Channel assignment schemes for cellular mobile telecommunication systems: A comprehensive survey. *Proceedings of the IEEE*, 82(9):1398–1430, 1994.

[23] M.D. Kulavaratharasah and A.H. Aghvami. Teletraffic performance evaluation of microcellular personal communication networks PCN's with prioritized handoff procedures. *IEEE Transactions on Vehicular Technology*, 48(1):137–152, January 1999.

[24] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In T. Field, P.G. Harrison, J. Bradley, and U. Harder, editors, *Proceedings of the 12th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, volume 2324 of *LNCS*, London, 2002. Springer.

[25] K. Larsen and A. Skou. Bisimulation through Probabilistic Testing. *Information and Computation*, 94(1):1–28, September 1991.

[26] K.G. Larsen. Compositional theories based on an operational semantics of contexts. In *REX Workshop on Stepwise Refinement of Parallel Systems*, volume 430 of *LNCS*, pages 487–518. Springer-Verlag, May 1989.

[27] R. Milner. *Communicating and Mobile Systems: the π-calculus*. Cambridge University Press, 1999.

[28] R. Milner, M. Tofte, R. Harper, and D. MacQueen. *The Definition of Standard ML: Revised 1996*. The MIT Press, 1996.

[29] C. Reinke. Haskell-Coloured Petri Nets. In *Implementation of Functional Languages, 11th International Workshop*, volume 1868 of *LNCS*, pages 165–180, Lochem, The Netherlands, September 1999. Springer-Verlag.

[30] I. C. Rojas M. *Compositional Construction and Analysis of Petri net Systems*. PhD thesis, The University of Edinburgh, 1997.

[31] R. Valk. Petri nets as token objects—an introduction to Elementary Object Nets. In J. Desel and M. Silva, editors, *Proceedings of the 19th International Conference on Application and Theory of Petri Nets*, volume 1420 of *Lecture Notes in Computer Science*, pages 1–25, Lisbon, Portugal, 1998. Springer-Verlag.

A Semantics of PEPA

The semantic rules, in the structured operational style, are presented in Figure A.1; the interested reader is referred to [20] for more details. The rules are read as follows: if the transition(s) above the inference line can be inferred, then we can infer the transition below the line. The notation $r_\alpha(E)$ which is used in the third cooperation rule denotes the apparent rate of α in E.

A.1 Definition of PEPA nets equality on alphabets

The relation $=_a$ is used in the PEPA nets semantics. Its definition is straightforward but is included here for completeness.

$$P =_a Q \quad \text{if} \quad \mathrm{alph}\, P = \mathrm{alph}\, Q$$

The alphabet of a PEPA nets component is the least set satisfying the following equations.

$$\begin{aligned}
\mathrm{alph}\,(P \underset{L}{\bowtie} Q) &= ((\mathrm{alph}\, P) \setminus L) \cup ((\mathrm{alph}\, Q) \setminus L) \cup ((\mathrm{alph}\, P) \cap L \cap (\mathrm{alph}\, Q)) \\
\mathrm{alph}\,(P/L) &= (\mathrm{alph}\, P) \setminus L \\
\mathrm{alph}\,(P[C]) &= \mathrm{alph}\, P \\
\mathrm{alph}\, I &= \mathrm{alph}\, S \text{ where } I \overset{\mathit{def}}{=} S \\
\mathrm{alph}((\alpha, r).S) &= \{\, \alpha \,\} \cup \mathrm{alph}\, S \\
\mathrm{alph}(R + S) &= \mathrm{alph}\, R \cup \mathrm{alph}\, S
\end{aligned}$$

Prefix

$$\frac{}{(\alpha, r).E \xrightarrow{(\alpha,r)} E}$$

Cooperation

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E' \underset{L}{\bowtie} F} \; (\alpha \notin L) \qquad \frac{F \xrightarrow{(\alpha,r)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E \underset{L}{\bowtie} F'} \; (\alpha \notin L)$$

$$\frac{E \xrightarrow{(\alpha,r_1)} E' \quad F \xrightarrow{(\alpha,r_2)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,R)} E' \underset{L}{\bowtie} F'} \; (\alpha \in L) \text{where } R = \frac{r_1}{r_\alpha(E)} \frac{r_2}{r_\alpha(F)} \min(r_\alpha(E), r_\alpha(F))$$

Choice

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E + F \xrightarrow{(\alpha,r)} E'} \qquad \frac{F \xrightarrow{(\alpha,r)} F'}{E + F \xrightarrow{(\alpha,r)} F'}$$

Hiding

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\alpha,r)} E'/L} \; (\alpha \notin L) \qquad \frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\tau,r)} E'/L} \; (\alpha \in L)$$

Constant

$$\frac{E \xrightarrow{(\alpha,r)} E'}{A \xrightarrow{(\alpha,r)} E'} \; (A \stackrel{\mathit{def}}{=} E)$$

Fig. A.1. The operational semantics of PEPA