Extensions to PEPA to simplify the creation of component libraries

P R Mitton * D R W Holton †

June 18, 1999

Abstract

PEPA is a stochastic process algebra (SPA) used for performance modelling. SPAs allow both functional and performance measures to be derived from the same model. In addition, SPAs are compositional in that they allow complex components to be created from simpler ones. For wider use and acceptance of SPAs it is essential that it is possible to create a library of reusable components. This paper proposes a number of extensions to the PEPA language that simplifies the creation of a library. Extensions are proposed that allow renaming of components, a distinguished name for probabilistic choices and a minor extension to the basic language that allows the use of general probability distribution functions.

1 Introduction

Accurate models of processes are increasingly important, as the cost of implementing many technological solutions is rapidly increasing. Techniques that allow the performance and functional characteristics of a proposed system to be evaluated prior to their construction can reduce the probability of systems being deployed that do not meet their design goals.

Stochastic Process Algebras (SPA) have the advantage over other modelling paradigms in that a single model can provide both performance measures and functional characteristics. PEPA [1] is a compositional SPA that allows the creation of complex components from a number of simpler components. PEPA has a formal definition that allows comparison between models and components based on equivalence relations defined by semantic rules. A design goal of PEPA was to create a parsimonious language that was simple to learn and easy to reason about. However, this does result in PEPA descriptions being verbose unless the compositional properties of the language are fully deployed. This paper suggests additions to the language, which, while not greatly expanding the size and complexity of the language, assists the modeller in creating a library of reusable components. Additionally, the proposed extensions improve the readability of PEPA descriptions.

PEPA currently makes no distinction between activities with a rate that models an actual process, and activities that denote the outcome of a probabilistic choice. This presents a problem for the designer of tools to assist the modeller, in that it is difficult to differentiate, purely from the language description, whether a rate corresponds to an actual process or not. As a result, it is difficult to allocate an appropriate rate to probabilistic choices. The probabilistic choice should take place instantaneously and this is achieved by setting the rate of a probabilistic choice to a value that is much greater than any non-probabilistic choice. To address this, it is proposed that a distinguished rate is created that by allowing probabilistic choices to be recognised assists the creation of modelling tools for use with reusable libraries of components. In conjunction a renaming operator is proposed which can separate the process of creating a generic component and the naming of activities in that component.

To maximise the utility of PEPA it should be capable of modelling as wide a range of problems as possible. Currently PEPA models are restricted to processes with a negative exponential probability

*School of Computing and Mathematics, School of Informatics The University of Bradford, Bradford, BD7 1DP. Email: P.R.Mitton@scm.brad.ac.uk

†School of Computing and Mathematics, School of Informatics, The University of Bradford, Bradford, BD7 1DP. Email: D.R.W.Holton@scm.brad.ac.uk

distribution function. This paper proposes an extension to the language that allows the use of generalized probability distributions by the introduction of additional parameters to an activity.

1.1 PEPA

PEPA is a compositional language based on extensions to process algebras such as CCS [2]. The design objectives of PEPA were to create a formally defined, parsimonious, compositional language.

The extensions to process algebras are to associate a random variable with each activity that may occur. This variable represents the time taken for an activity to complete. The randomness of the variable allows temporal uncertainty to be included in the model. Non-deterministic branching is replaced by probabilistic branching. A race condition between enabled activities determines which takes place; this represents the spatial uncertainty within the system.

Full details of the PEPA language are found in [1] here we concentrate on the formal definition of the operational semantics. The following conventions are observed in this section:

- Every activity in PEPA has an associated random variable which is a duration with exponential distribution.
- Every activity is uniquely typed and part of the countable set $\mathcal{A}$.
- $\vec{\mathcal{A}}\,(P)$ denotes the action type set of component P.
- The distinguished action type τ is the unknown type. These activities are private and not visible outside the component.
- Components names start with a large roman letter.
- Activity names are single roman letters from the start of the alphabet..
- Action type names are single greek letters, or by names beginning with small roman letters.
- Activity rate names are single roman letter from the end of the alphabet; when the activity rate is unspecified (a passive component) the distinguished rate $\top$ is used.

Each activity a is a pair (α, r) where $\alpha \in \mathcal{A}$ is the action type and r is the rate.

1.2 Syntax

A prefix operator, ., is provided. This indicates that given $(\alpha, r).P$ the system will evolve into P at a rate given by an exponentially distributed probability function with parameter r if the action type α is enabled.

The choice operator is $+$. Given a system $P + Q$, the system may behave either as component P or component Q. The behaviour will be determined by a race condition between the rates of components P and Q. As they are continuous probability distributions one component must complete first and the system will then behave either as P' or Q' where P' and Q' are the components that result from P and Q completing their respective activities.

The cooperation operator is $\underset{L}{\bowtie}$. Given a system $P \underset{L}{\bowtie} Q$, where L is a set of action types ($L \subseteq \mathcal{A}$). L is the cooperation set and indicates a synchronization between P and Q on the action types in L. For this to occur, the action type must be enabled in both P and Q. Both active and passive synchronizations can be modelled. If both action types have specified rates the component is active, when one is unspecified (has a rate of $\top$), that component is passive. When $L = \emptyset$ there is no synchronization and the components evolve in parallel.

The hiding operator is $/$. Given a system P/L, the hiding operator ensures that any action type within the set L are not visible outside the component. This produces the effect of an internal delay of the unknown action type.

The constant operator is $\stackrel{def}{=}$. Given a system $A \stackrel{def}{=} P$, the component A will behave as the component P.

1.3 Operational Semantics

1.3.1 Prefix

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

1.3.2 Choice

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

1.3.3 Cooperation

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

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

1.3.4 Hiding

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

1.3.5 Constant

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

2 Renaming

Our first proposal is the introduction of a renaming operator that allows a component to be given a different name. This is useful both in terms of allowing re-use of agents and also in defining new combinators. The notation used follows CCS [2]; $E[f]$ means apply the *renaming function* f to the names of actions and rates in E. The operational semantics are:

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E[f] \xrightarrow{(f(\alpha),f(r))} E'[f]}$$

f is a union of a total function from $\vec{\mathcal{A}}\ (E)$ to $\mathcal{A}$, with a function from rate names to $R^+ \cup \{\top\}$. It is clear that $f(\tau) = \tau$ and $f(\top) = \top$. In cases where few actions or names are to be renamed then it will be convenient to write $E[a \mapsto b, \dots, y \mapsto z]$, where it is understood that the term in square brackets corresponds to $Id \oplus \{a \mapsto b, \dots, y \mapsto z\}$, with Id being the identity function over $\vec{\mathcal{A}}\ (E)$, and $\oplus$ being the function overriding operator. Renaming of action names and rate names is used in the following sections.

$$A \stackrel{def}{=} (action1, r1).A' + (action2, r2).A'' \quad (1)$$
$$A' \stackrel{def}{=} (tossCoin, r3).A + (tossCoin, r4).A'' \quad (2)$$
$$A'' \stackrel{def}{=} (something1, r5).A + (something2, r6).A' \quad (3)$$

Figure 1: PEPA choice operator example

3 Probabilistic Choice

In all modelling, the names of objects within the model should correspond to the names of objects in the modelled system. This greatly simplifies the job of a new modeller being introduced to an existing system or alternatively the original modeller returning to the model after a period of time. Carefully chosen object names can greatly improve the efficiency of maintaining models.

In PEPA the choice operator indicates that a system may evolve into one of a set of derivatives. Which choice is taken is determined by a race condition between the set of all enabled activities of each of the components involved in the choice.

In definition 1 of Figure 1 the name of the action type suggests that a real activity is being modelled. $r1$ and $r2$ probably correspond to the rate of an activity in the modelled system. Conversely, the definition in 2 suggests that a probabilistic decision is being taken and $r3$ and $r4$ are the relative probabilities of the two events. In definition 3 the naming of the action types provides no clue as to whether the choice is made probabilistically or not. Whether $r5$ and $r6$ have a real world counterpart is difficult to determine.

For the majority of modelling situations, the performance measures are obtained by using a computer program to generate and solve the underlying Markov process to find the steady state probability distribution for the states. Only very simple problems are reliably solvable using "by hand" methods.

Any tool that is used to find the steady state probability distribution should produce a steady state solution where the probability of being in states such as A', which occur only to allow a probabilistic choice to occur, should be insignificant.

Generalized Stochastic Petri Nets (GSPN) [3] adress this problem by having instantaneous transitions, thereby preventing probabilistic choices distorting the steady state probability distribution of the model. An alternative approach is to ensure that the rates allocated to probabilistic choices are significantly greater than the rates of all non-probabilistic choices. One disadvantage of the second approach is that it can lead to a "stiff" Markov process that may be less amenable to numerical solution [4].

In simple examples such as Figure 1 the modeller can allocate appropriate values to the rates of all the transitions. However, if the component A is to be incorporated into a larger model the modeller has the responsibility of ensuring that the relative rates of all probabilistic choices should be larger than all non-probabilistic choices.

If the modeller is responsible for allocating all the rates within the model this condition can be satisfied. When a component is to be packaged as part of a library of reusable components, however, the condition cannot be satisfied. The creator of a library component has no knowledge about the maximum rate of modelled processes and is thereby unable to determine an appropriate rate for probabilistic rates.

The problem is compounded by the fact that often the probabilities in a choice may differ by several orders of magnitude. This may occur, for example, when we are modelling system failures.

In the end it is up to the designer of the modelling tool to use the most appropriate method to reduce distortion of the steady state probability distribution. However, the modelling language should provide assistance to the modelling tool designer in selecting such a method.

With this in mind, it is proposed that a distinguished rate $\perp$ should be introduced into the PEPA language. A distinguished rate already exists in PEPA ($\top$) and the introduction of a new distinguished rate does not alter the foundations of the language. However, an additional variable is needed to describe the probability of the action. Again, this does not alter the formal basis of the language as the extension can be translated into the basic language, however it is a change

$$A \stackrel{def}{=} (action1, r1).A' + (action2, r2).A'' \quad (4)$$

$$A' \stackrel{def}{=} (tossCoin, \perp, r3).A + (tossCoin, \perp, r4).A'' \quad (5)$$

$$A'' \stackrel{def}{=} (something1, r5).A + (something2, r6).A' \quad (6)$$

Figure 2: Extended PEPA choice operator example

that aids the modeller. This variable should always be in the range $0 \leq r \leq 1$. The modelling tool can check that this condition is satisfied.

If $\perp$ and an actual rate occur in the same choice then the $\perp$ choice should always be taken. This represents the ideal, instantaneous behaviour of the probabilistic activity.

Figure 2 is how the example given in Figure 1 would be expressed using $\perp$. The computer based modelling tool can now take advantage of the additional information provided by the extended language to handle the probabilistic choices in state A'. The tool can check that the total probability of the choices is one, and assign appropriate rates to the probabilistic action types.

It is also essential that any user of the library components should be aware which rates within the component are probabilistic. It is proposed that an additional operator ($\oslash$) is provided for convenience that comes after the hiding ($/$) operator that indicates that what follows is a set of all the probabilistic rates in the component.

For example a component library version of A might be described as:

$$\begin{aligned} LibA &\stackrel{def}{=} A[tossCoin \mapsto choose, r3 \mapsto head, r4 \mapsto tail] \\ &\quad /\{action1, action2, something1, something2\} \oslash \{r3, r4\} \end{aligned}$$

This is purely a convenience that distinguishes to the library user that these names may only have a value (v) where $0 \leq v \leq 1$. Otherwise the library user has no indication that $r3$ and $r4$ are probabilistic.

A consequence of the semantics of the $\perp$ action type greatly expands its usefulness. If a $\perp$ rate, with probability one, is used in conjunction with other action types it can inform the modelling tool that the action is to be considered as an instantaneous transition. An example of this use is given in Section 5. Again this does not alter the formal foundations of the language or the difficulty of reasoning about the model. It is merely a convenient interpretation of the existing language that assists the construction and solution of the underlying Markov process.

4 Example of renaming and choice

Modelling Multiple-Cell Manufacturing systems is one area to which SPAs have been applied [5]. A single cell could be described in PEPA as:

$$Cell \stackrel{def}{=} (enterA, \top).(processA, pRateA).(leaveA, lRateA).Cell$$

The three parts of the definition are a synchronization with components entering the work cell, the actual processing, and the preparation for passing the component to the next workcell. The first part is synchronized with the preceding part, and the last with the following part of the system.

A useful item within a library of components would be a number of identical cells combined into a single component. One way of describing that component in PEPA would be:

$$CellA5 \stackrel{def}{=} Cell \bowtie Cell \bowtie Cell \bowtie Cell \bowtie Cell$$

Unfortunately, even when simply combined with a single input cell and a single output cell, this leads to a large state space (244 states in the example). By aggregating the states a different definition can be found:

$$\begin{aligned}
CellB5_0 &\stackrel{def}{=} (enterB, \top).(processB, pRateB).CellB5_1 \\
CellB5_i &\stackrel{def}{=} (enterB, \top).(processB, pRateB).CellB5_{i+1} + \\
&\quad (leaveB, lRateB).CellB5_{i-1} \qquad 0 < i < 5 \\
CellB5_5 &\stackrel{def}{=} (leaveB, lRateB).CellB5_4
\end{aligned}$$

This has a greatly reduced state space (12 states in the example) compared to the first definition. The second definition of the component is preferable when creating a library component, because when components are combined the resultant number of states is of the order of the product of the states of the individual components, handling very large state spaces is a recurring problem in SPAs. However, given a more complicated example the current "best" definition may be superceded. When this occurs it should be possible to alter the definition of the component without affecting any models which make use of that component. In other words the component must display a consistent interface to the modeller even when the underlying definition is changed.

For example, assume we have a library where all the components are preceded by the capital letter C. Then we can create a $CCells5$ component by making use of the renaming operator introduced in Section 2:

$$CCell5 \stackrel{def}{=} CellA5[enterA \mapsto cellsEnter,\ leaveA \mapsto cellsLeave\ pRateA \mapsto pRate]/\{processA\}$$

This makes use of the first definition. However, a version using the second definition can easily be created:

$$CCell5 \stackrel{def}{=} CellB5_0[enterB \mapsto cellsEnter,\ leaveB \mapsto cellsLeave,\ pRateB \mapsto pRate]/\{processB\}$$

If a work cell model was to be created of five polish cells the component could be created by:

$$Polish \stackrel{def}{=} CCell5[cellsEnter \mapsto polishEnter,\ cellsLeave \mapsto polishLeave,\ pRate \mapsto polishRate]$$

The modeller making use of $Polish$ has no need to know anything about the construction of component $CCell5$ other than its signature. It could have been implemented using the $CellA5$ or the $CellB5$ component. At a later stage, if a "better" implementation is found it could replace the current definition.

5 General Probability Distributions

The PEPA description of a component makes no assumptions as to the probability distribution function (pdf) of the transition rate. However, for a solution to be found for the model the solution of the underlying Markov process must be found. The transitions within a Markov process are independant of the time of the transition. This memoryless property is an essential element of Markov processes. However, the memoryless property only holds for negative exponential pdfs. If PEPA is to be able to consider components where the transition is a different pdf then changes are required to the language.

As the intention of these extensions is to allow non-mathematician to use the PEPA language it is proposed that any extension to the language should use terms that should be understood by a general population. The is already true of the rate parameter within an activity. Typically, when a system is to be modelled sample measurements of the components of the system will be taken in isolation. From these sample measurements it is possible to find the mean and standard deviation of the activity.

It is proposed that an optional third parameter should be added to each activity that represents the standard deviation of the activity. For backward compatibility, if the third parameter is omitted it should be assumed that the standard deviation is that of a negative exponential pdf.

Generally, the mean and standard deviation will provide enough information to accurately model a process. However, it is possible that more detailed information may be required, in that case there must be a way for adding this additional information into the PEPA description.

The current rate parameter is the inverse of the first moment (mean or first expectation). The standard deviation (σ)is a function of the first ($E[X]$) and second moments ($E[X^2]$):

$$\sigma = \sqrt{E[X^2] - E[X]^2}$$

It is not proposed at this stage to suggest possible ways of including information gained from moments higher than the second into PEPA. However, it seems reasonable that any further extensions should make use of the corresponding higher moment, i.e. the next extension should be a function of the first three moments.

The designer of the modelling tool can now use the most appropriate technique for generating an equivalent Markov process for the pdf. A typical technique would be to make use of Cox's Method of Stages [6].

Currently PEPA activities have the descriptive form:

$$(action\ type, actionrate) = \left(action\ type, \frac{1}{E[X]}\right)$$

The proposed extension would directly use the standard deviation of the pdf as an additional parameter in the activity description

$$(action\ type, mean, standard\ deviation) = \left(action\ type, \frac{1}{E[X]}, \sqrt{E[X^2] - E[X]^2}\right)$$

As an example consider the case of a uniform distribution function. Here the random variable will be a randomly distributed value in the range $a \leq x \leq b$. Thinking in terms of transitions times, the transition will occur after time a and before time b.

It can be shown ([7]) that the mean is:

$$E[X] = \frac{1}{2}(a + b)$$

and the variance is:

$$\sigma^2 = \frac{(b-a)^2}{12}$$

If the time taken for an activity is randomly distributed between 10 and 22 seconds ($a = 10, b = 22$) this gives $E[X] = 16$ and $Var[X] = 12$. Thus, a PEPA description of this activity would be:

$$(actiontype, 16, \sqrt{12})$$

Similarly, it is possible to describe deterministic transitions by setting the third parameter to 0, although the method of stages is not appropriate when the coefficient of variation ($C_x = \frac{\sigma}{E[X]}$) is small [8].

Again, it is stressed that it is the responsibility of the modelling tool designer to determine appropriate translations from the non-exponential pdfs to an equivalent Markov process.

6 Renaming Revisited

The proposed extensions require changes to the renaming operatational semantics to account for the additional parameter, using a similar notation to Section 2. Here, $E[f]$ means apply f to actions, rates, and standard deviations, $f(sd)$ is a total function from the standard deviation name to a non-negative real number, and the semantics are:

$$\frac{E \xrightarrow{(\alpha,r,sd)} E'}{E[f] \xrightarrow{(f(\alpha),f(r),f(sd))} E'[f]}$$

where f is now a union of a total function from $\vec{\mathcal{A}}\ (E)$ to $\mathcal{A}$, a function from rate names to $R^{+} \cup \{\top\}$ and a function from standard deviation names to a non-negative real number.

There is a special case to consider. If the rate is $\perp$, the "standard deviation" parameter is a probability, but $f(sd)$ is a total function as before. It is clear that $f(\perp) = \perp$.

7 Using the PEPA extensions

7.1 Generalized Exponential Distributions

A useful PEPA library component would model the generalized exponential pdf. This is simply modelled using the extended semantics as:

$$\begin{aligned} GE &\stackrel{def}{=} (chooseImm, \perp, rate1).GE + (chooseDel, \perp, rate2).GE' \\ GE' &\stackrel{def}{=} (arrive, rate3).GE \end{aligned}$$

where $rate1 = 1 - rate2$ and $rate1$ is the probability of an instantaneous transition to the derivative. Notice, that this is an example of a non-negative exponential function which is modelled by use of the $\perp$ distinguished rate rather than the inclusion of a standard deviation term.

To create a specific instance of the GE component we only have to rename the activities to reflect any synchronizations that we wish to use and to set appropriate rates.

$$\begin{aligned} LibGE \stackrel{def}{=}\ & GE[chooseImm \mapsto delay, rate1 \mapsto libRate1,\ chooseDel \mapsto delay, rate2 \mapsto libRate2, \\ & arrive \mapsto output, rate3 \mapsto libRate3] \end{aligned}$$

It is good practice to always rename objects within a library component to prevent inadvertant name clashes.

If we are modelling a system that accepts customer orders to manufacture a batch of components on a just-in-time basis we can make use of our library component. Assuming that the orders arrive in a Poisson stream at rate ($arrivalRate$) and, after checking their validity, the orders are manufactured:

$$\begin{aligned} Entrance &\stackrel{def}{=} (ordersArrive, arrivalRate).(check, checkRate).Entrance \\ MGE &\stackrel{def}{=} LibGE\langle enter \mapsto check, rate1 \mapsto mRate1, \\ &\quad rate2 \mapsto mRate2, \\ &\quad arrive \mapsto mArriveo, rate3 \mapsto \top\rangle \\ Manufacture &\stackrel{def}{=} (mArrive, \top).(manuf, mRate4).Manufacture \\ Entrance \underset{check}{\bowtie}\ & MGE \underset{mArrive}{\bowtie} Manufacture \end{aligned}$$

$$
\begin{aligned}
CPU &\stackrel{def}{=} (request, rr).CPU \\
Cache &\stackrel{def}{=} (request, \top).((cService, \bot, pch).(access, rca).Cache \\
&\quad +(mmRequest, \bot, pcm).Cache) \\
CPU &\underset{request}{\bowtie} Cache
\end{aligned}
$$

Figure 3: CPU/Cache Subsystem

7.2 Shared Memory Multiple Processor Systems

In modelling shared processor systems it is possible to create a component that represents a CPU and its local cache. The CPU produces a stream of requests for memory access that will either be met by the local cache or be serviced by the shared memory via the system bus. The percentage of requests met by the local cache is high due to the principle of locality of code. The time taken to service requests to main memory is dependant on the speed of the memory and the amount of contention for the bus, which is a function of the number of CPUs connected to the bus. Figure 3 is a PEPA description of a CPU and cache memory. The CPU makes a memory request at rate rr, the cache then either services the request with a probability of pch or sends the request to shared memory with probability pcm. If the cache is able to service the request a further time period elapses before the contents of the memory location is returned to the CPU.

A library component can be created with the behaviour of this system:

$$
\begin{aligned}
CPUCache &\stackrel{def}{=} CPU \underset{request}{\bowtie} Cache \\
LibCPUCache &\stackrel{def}{=} CPUCache[request \mapsto req, \\
&\quad rr \mapsto rMemRequests, \\
&\quad pch \mapsto probHit, \\
&\quad pcm \mapsto probMiss, \\
&\quad rca \mapsto cacheAccessTime] \\
&\quad /\{cService, access, mmRequest\} \oslash \{probHit, probMiss\}
\end{aligned}
$$

8 Conclusions and Suggestions for further work

PEPA is a compositional process algebra. To take full advantage of this important characteristic of the language, it should be able to create a library of reusable components that modellers can use, or modify, to simplify the creation of complex system models. Currently, PEPA does not have all the language features required to accomplish this.

This paper has described how the addition of operators to allow the renaming of components enables different definitions of systems to be substituted within such a library component. Renaming allows more than one instance of a library component to be used, and individual synchronizations and rates to be established for each of the instances.

PEPA transitions are determined by a race condition between enabled states. Some of these transitions are caused by probabilistic choices. The distinguished rate introduced in this paper allows a modelling tool to allocate appropriate values to such rates.

Not all systems can be modelled by negative exponential probability distribution functions. To extend the usefullness of PEPA a parameter is added that represents the standard deviation of the probability distribution function. This allows the modelling tool to use the most appropriate means of generating an equivalent Markov process, based upon the relationship between the mean and standard deviation. One such method is Cox's Method of Stages.

Appropriate extensions were also made to the operational semantics of the PEPA language. At all times the authors were reluctant to add features to the language, aware that its parsimonious nature is one of the attractions of PEPA that simplifies learning and reasoning about the language.

However, the proposed extensions make very little difference to the simplicity of reasoning about the language whilst simplifying the creation of a reusable library of components. It is suggested that this feature more than compensates for the additional operators.

The obvious next stage is to create a modelling tool that implements all the features described in this paper. This tool should have a library of reusable components including, but not limited to, fixed size buffers, multiple servers, batch sources and servers with different probability distribution functions. Each of the implementations of the components should be easily replaced if a more efficient implementation is discovered.

References

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

[2] R Milner. *Communications and Concurrency.* Prentice Hall International, 1989.

[3] M Ajmone Marsan, G Balbo, G Conte, S Donatelli, and G Franceschinis. *Modelling with Generalized Stochastic Petri Nets.* Wiley, 1995.

[4] William J Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, 1994.

[5] J A Buzacott and J G Shantikumar. *Stochastic Models of Manufacturing Systems.* Industrial and Systems Engineering. Prentice Hall International, 1993.

[6] D R Cox and H D Miller. *The Theory of Stochastic Processes.* Methuen and Co, 1965.

[7] Howard M Taylor and Samuel Karlin. *An Introduction to Stochastic Modeling.* Academic Press, third edition, 1998.

[8] Charles H Sauer and K Mani Chandy. *Computer Systems Performance Modeling.* Prentice Hall International, 1981.