Performance Modelling — Lecture 10
PEPA

Jane Hillston
School of Informatics
The University of Edinburgh
Scotland

20th February 2017

Stochastic Process Algebra — Introduction and Motivation

Stochastic process algebras are similar to Stochastic Petri nets in several ways:

Stochastic Process Algebra — Introduction and Motivation

Stochastic process algebras are similar to Stochastic Petri nets in several ways:

- Both are based on formalisms originally developed to model concurrency.

Stochastic Process Algebra — Introduction and Motivation

Stochastic process algebras are similar to Stochastic Petri nets in several ways:

- Both are based on formalisms originally developed to model concurrency.
- Both fall within the broad class of discrete event modelling formalisms and incorporate timing and probabilistic information with the events in the system.

Stochastic Process Algebra — Introduction and Motivation

Stochastic process algebras are similar to Stochastic Petri nets in several ways:

- Both are based on formalisms originally developed to model concurrency.
- Both fall within the broad class of discrete event modelling formalisms and incorporate timing and probabilistic information with the events in the system.
- Both have formal semantics which can be used to automatically derive an underlying Markov process (when durations are assumed to be exponentially distributed)

Stochastic Process Algebra — Introduction and Motivation

Stochastic process algebras are similar to Stochastic Petri nets in several ways:

- Both are based on formalisms originally developed to model concurrency.
- Both fall within the broad class of discrete event modelling formalisms and incorporate timing and probabilistic information with the events in the system.
- Both have formal semantics which can be used to automatically derive an underlying Markov process (when durations are assumed to be exponentially distributed)

The major difference between them is compositionality.

Advantages of compositionality

For model construction:

- when a system consists of interacting components, the components, and the interaction, can each be modelled separately;

Advantages of compositionality

For model construction:

- when a system consists of interacting components, the components, and the interaction, can each be modelled separately;
- models have a clear structure and are easy to understand;

Advantages of compositionality

For model construction:

- when a system consists of interacting components, the components, and the interaction, can each be modelled separately;
- models have a clear structure and are easy to understand;
- models can be constructed systematically, by either elaboration or refinement;

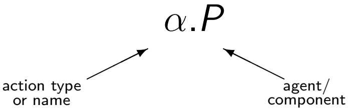
Advantages of compositionality

For model construction:

- when a system consists of interacting components, the components, and the interaction, can each be modelled separately;
- models have a clear structure and are easy to understand;
- models can be constructed systematically, by either elaboration or refinement;
- the possibility of maintaining a library of model components, supporting model reusability, is introduced.

Process Algebra

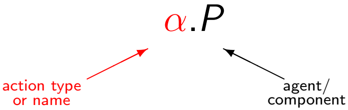
- Models consist of agents which engage in actions.



$$\alpha.P$$

Process Algebra

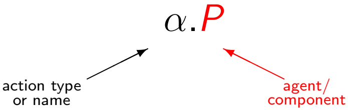
- Models consist of agents which engage in actions.



$\alpha.P$

Process Algebra

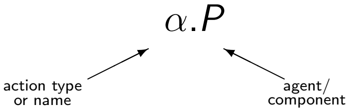
- Models consist of agents which engage in actions.



$$\alpha.P$$

Process Algebra

- Models consist of agents which engage in actions.



$$\alpha.P$$

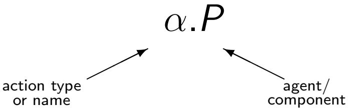
action type
or name

agent/
component

- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process Algebra

- Models consist of agents which engage in actions.



$$\alpha.P$$

action type
or name

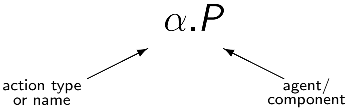
agent/
component

- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process algebra model

Process Algebra

- Models consist of agents which engage in actions.



$$\alpha.P$$

action type or name

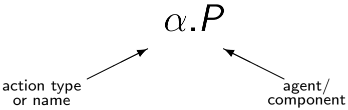
agent/ component

- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process algebra model $\xrightarrow{\text{SOS rules}}$

Process Algebra

- Models consist of agents which engage in actions.



$$\alpha.P$$

action type
or name

agent/
component

- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

$$\text{Process algebra model} \xrightarrow{\text{SOS rules}} \text{Labelled transition system}$$

Dynamic behaviour

- The behaviour of a model is dictated by the semantic rules governing the combinators of the language.

Dynamic behaviour

- The behaviour of a model is dictated by the semantic rules governing the combinators of the language.
- The possible evolutions of a model are captured by applying these rules exhaustively, generating a labelled transition system.

Dynamic behaviour

- The behaviour of a model is dictated by the semantic rules governing the combinators of the language.
- The possible evolutions of a model are captured by applying these rules exhaustively, generating a labelled transition system.
- This can be viewed as a graph in which each node is a state of the model (comprised of the local states of each of the components) and the arcs represent the actions which can cause the move from one state to another.

Dynamic behaviour

$$Browser \stackrel{def}{=} display.(cache.Browser \;+\; get.download.rel.Browser)$$

Dynamic behaviour

$$Browser \overset{def}{=} display.(cache.Browser \; + \; get.download.rel.Browser)$$

$$\frac{}{\alpha.P \xrightarrow{\alpha} P}$$

$$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

$$\frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$

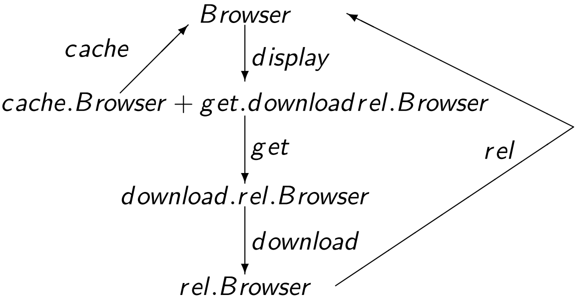
Dynamic behaviour

$$Browser \overset{def}{=} display.(cache.Browser \; + \; get.download.rel.Browser)$$

$$\frac{}{\alpha.P \xrightarrow{\alpha} P}$$

$$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

$$\frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$



$Browser$

$cache$

$display$

$cache.Browser + get.download rel.Browser$

get

rel

$download.rel.Browser$

$download$

$rel.Browser$

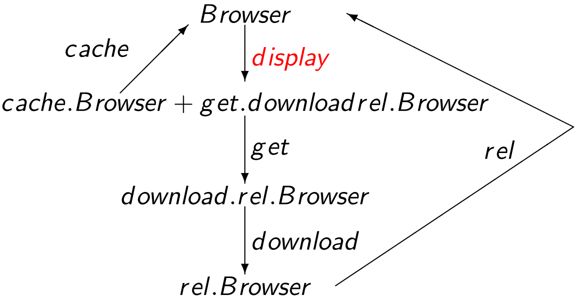
Dynamic behaviour

$$Browser \overset{def}{=} \color{red}{display.}(cache.Browser \;+\; get.download.rel.Browser)$$

$$\color{red}{\frac{}{\alpha.P \xrightarrow{\alpha} P}}$$

$$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

$$\frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$



$Browser$

$cache$

$display$

$cache.Browser + get.downloadrel.Browser$

get

rel

$download.rel.Browser$

$download$

$rel.Browser$

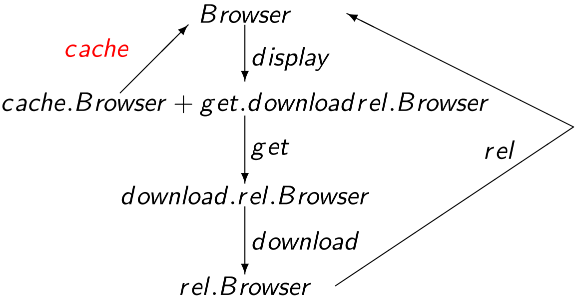
Dynamic behaviour

$$Browser \overset{def}{=} display.(cache.Browser \ + \ get.download.rel.Browser)$$

$$\frac{}{\alpha.P \xrightarrow{\alpha} P}$$

$$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

$$\frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$



$Browser$

$cache$

$display$

$cache.Browser + get.downloadrel.Browser$

get

rel

$download.rel.Browser$

$download$

$rel.Browser$

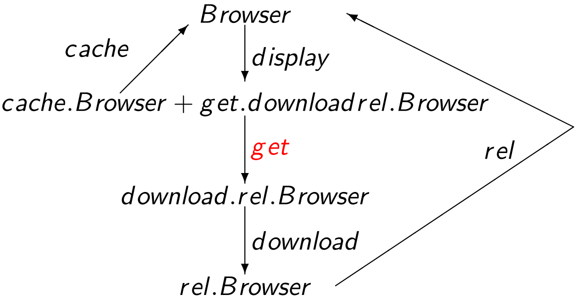
Dynamic behaviour

$$Browser \overset{def}{=} display.(cache.Browser \;+\; get.download.rel.Browser)$$

$$\frac{}{\alpha.P \xrightarrow{\alpha} P}$$

$$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

$$\frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$



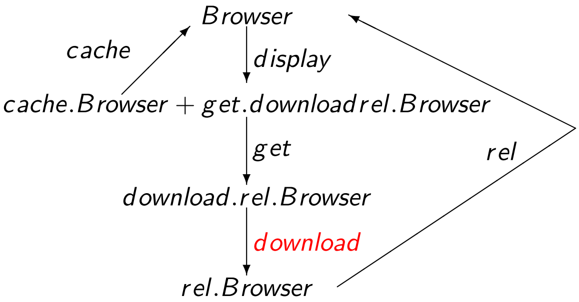
Dynamic behaviour

$$Browser \overset{def}{=} display.(cache.Browser \; + \; get.{\color{red}download.}rel.Browser)$$

$${\color{red}\frac{}{\alpha.P \xrightarrow{\alpha} P}}$$

$$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

$$\frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$



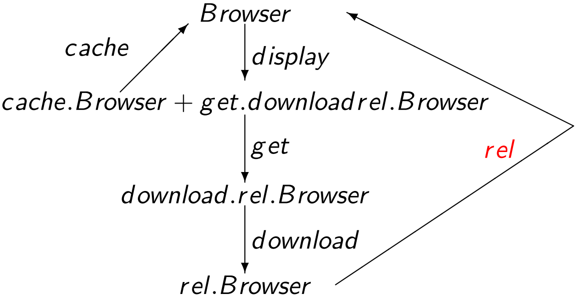
Dynamic behaviour

$$Browser \overset{def}{=} display.(cache.Browser \; + \; get.download.\color{red}{rel.}\color{black}Browser)$$

$$\frac{}{\color{red}{\alpha.P \xrightarrow{\alpha} P}}$$

$$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

$$\frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$



$Browser$

$cache$

$display$

$cache.Browser + get.downloadrel.Browser$

get

$\color{red}{rel}$

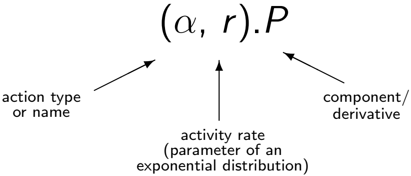
$download.rel.Browser$

$download$

$rel.Browser$

Stochastic Process Algebra

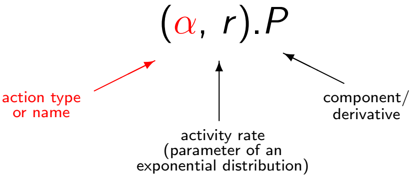
- Models are constructed from components which engage in activities.



$$(\alpha,\ r).P$$

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



$$(\alpha, r).P$$

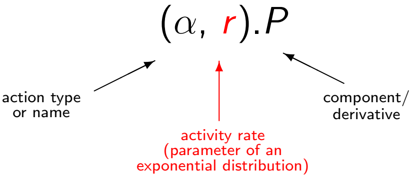
action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



$$(\alpha, r).P$$

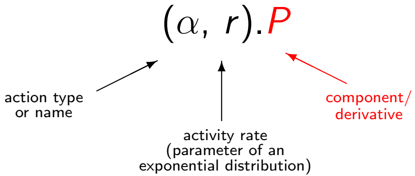
action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

Stochastic Process Algebra

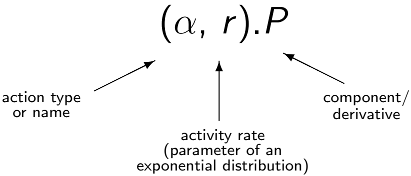
- Models are constructed from components which engage in activities.



$$(\alpha, r).P$$

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

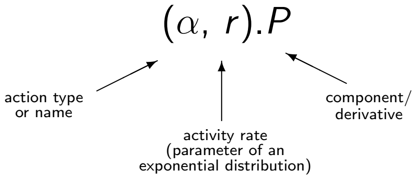


$$(\alpha, r).P$$

- The language is used to generate a CTMC for performance modelling.

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



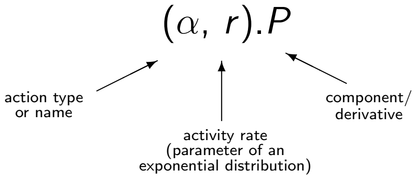
$$(\alpha, r).P$$

- The language is used to generate a CTMC for performance modelling.

SPA
MODEL

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

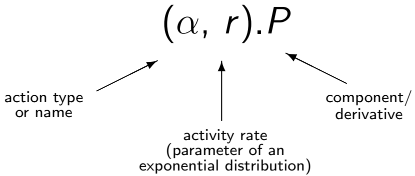


$$(\alpha, r).P$$

- The language is used to generate a CTMC for performance modelling.

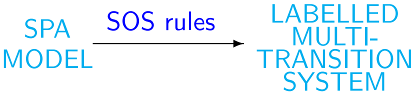
Stochastic Process Algebra

- Models are constructed from components which engage in activities.



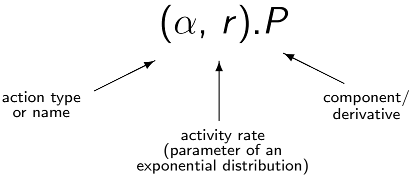
$$(\alpha, r).P$$

- The language is used to generate a CTMC for performance modelling.



Stochastic Process Algebra

- Models are constructed from components which engage in activities.



$$(\alpha, r).P$$

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

- The language is used to generate a CTMC for performance modelling.

SPA MODEL

SOS rules

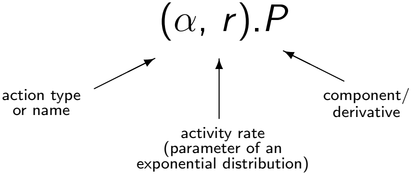
LABELLED MULTI-TRANSITION SYSTEM

state transition

diagram

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



$(\alpha, r).P$

action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

- The language is used to generate a CTMC for performance modelling.

SPA
MODEL

SOS rules

LABELLED
MULTI-
TRANSITION
SYSTEM

state transition

diagram

CTMC **Q**

PEPA syntax

$$
\begin{array}{lcll}
S & ::= & (\alpha, r).S & \text{(prefix)} \\
  & \mid & S_1 + S_2 & \text{(choice)} \\
  & \mid & X & \text{(variable)} \\
C & ::= & C_1 \underset{L}{\bowtie} C_2 & \text{(cooperation)} \\
  & \mid & C \,/\, L & \text{(hiding)} \\
  & \mid & S & \text{(sequential)}
\end{array}
$$

PEPA: informal semantics

$(\alpha, r).S$

The activity (α, r) takes time Δt (drawn from the exponential distribution with parameter r).

$S_1 + S_2$

In this choice either S_1 or S_2 will complete an activity first. The other is discarded.

PEPA: informal semantics

$C_1 \underset{L}{\bowtie} C_2$

All activities of C_1 and C_2 with types in L are shared: others remain individual.
NOTATION: write $C_1 \parallel C_2$ if L is empty.

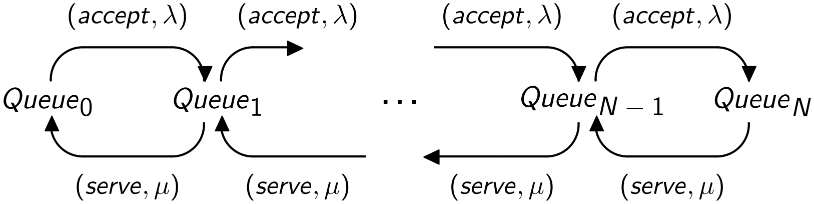
$C \,/\, L$

Activities of C with types in L are hidden (τ type activities) to be thought of as internal delays.

Example: M/M/1/N/N queue

$$\begin{array}{rcl} Arrival_0 & \stackrel{def}{=} & (accept, \lambda).Arrival_1 \\ Arrival_i & \stackrel{def}{=} & (accept, \lambda).Arrival_{i+1} + (serve, \top).Arrival_{i-1} \\ & & \qquad (0 < i < N) \\ Arrival_N & \stackrel{def}{=} & (serve, \top).Arrival_{N-1} \\ Server & \stackrel{def}{=} & (serve, \mu).Server \end{array}$$

Example: M/M/1/N/N queue



$$Queue_i \equiv Arrival_i \underset{\{serve\}}{\bowtie} Server$$

Example: Browsers, server and download

$$
\begin{array}{rcl}
\mathit{Server} & \stackrel{\mathit{def}}{=} & (\mathit{get}, \top).(\mathit{download}, \mu).(\mathit{rel}, \top).\mathit{Server} \\
\\
\mathit{Browser} & \stackrel{\mathit{def}}{=} & (\mathit{display}, p\lambda).(\mathit{get}, g).(\mathit{download}, \top).(\mathit{rel}, r).\mathit{Browser} \\
 & + & (\mathit{display}, (1-p)\lambda).(\mathit{cache}, m).\mathit{Browser} \\
\\
\mathit{WEB} & \stackrel{\mathit{def}}{=} & \left(\mathit{Browser} \parallel \mathit{Browser}\right) \underset{L}{\bowtie} \mathit{Server}
\end{array}
$$

where $L = \{\mathit{get}, \mathit{download}, \mathit{rel}\}$

Synchronisation

What should be the impact of synchronisation on rate?

Synchronisation

What should be the impact of synchronisation on rate?

PEPA assumes bounded capacity: that is, a component cannot be made to perform an activity faster by cooperation, so the rate of a shared activity is the minimum of the apparent rates of the activity in the cooperating components.

Synchronisation

What should be the impact of synchronisation on rate?

PEPA assumes bounded capacity: that is, a component cannot be made to perform an activity faster by cooperation, so the rate of a shared activity is the minimum of the apparent rates of the activity in the cooperating components.

The apparent rate of a component P with respect to action type α, is the total capacity of component P to carry out activities of type α, denoted $r_\alpha(P)$.

PEPA activities and rates

When enabled an activity, $a = (\alpha, \lambda)$, will delay for a period determined by its associated distribution function, i.e. the probability that the activity a happens within a period of time of length t is $F_a(t) = 1 - e^{-\lambda t}$.

PEPA activities and rates

- We can think of this as the activity setting a timer whenever it becomes enabled.

PEPA activities and rates

- We can think of this as the activity setting a timer whenever it becomes enabled.
- The time allocated to the timer is determined by the exponential distribution via the rate of the activity.

PEPA activities and rates

- We can think of this as the activity setting a timer whenever it becomes enabled.
- The time allocated to the timer is determined by the exponential distribution via the rate of the activity.
- If several activities are enabled at the same time each will have its own associated timer.

PEPA activities and rates

- We can think of this as the activity setting a timer whenever it becomes enabled.
- The time allocated to the timer is determined by the exponential distribution via the rate of the activity.
- If several activities are enabled at the same time each will have its own associated timer.
- When the first timer finishes that activity takes place—the activity is said to complete or succeed.

PEPA activities and rates

- We can think of this as the activity setting a timer whenever it becomes enabled.
- The time allocated to the timer is determined by the exponential distribution via the rate of the activity.
- If several activities are enabled at the same time each will have its own associated timer.
- When the first timer finishes that activity takes place—the activity is said to complete or succeed.
- This means that the activity is considered to "happen": an external observer will witness the event of activity of type α.

PEPA activities and rates

- We can think of this as the activity setting a timer whenever it becomes enabled.
- The time allocated to the timer is determined by the exponential distribution via the rate of the activity.
- If several activities are enabled at the same time each will have its own associated timer.
- When the first timer finishes that activity takes place—the activity is said to complete or succeed.
- This means that the activity is considered to “happen”: an external observer will witness the event of activity of type α.
- An activity may be preempted, or aborted, if another one completes first.

PEPA and time

All PEPA models are time-homogeneous since all activities are time-homogeneous: the rate and type of activities enabled by a component are independent of time.

PEPA and irreducibility and positive-recurrence

The other conditions, irreducibility and positive-recurrent states, are easily expressed in terms of the derivation graph of the PEPA model.

PEPA and irreducibility and positive-recurrence

The other conditions, irreducibility and positive-recurrent states, are easily expressed in terms of the derivation graph of the PEPA model.

We only consider PEPA models with a finite number of states so if the model is irreducible then all states must be positive-recurrent i.e. the derivation graph is strongly connected.

PEPA and irreducibility and positive-recurrence

The other conditions, irreducibility and positive-recurrent states, are easily expressed in terms of the derivation graph of the PEPA model.

We only consider PEPA models with a finite number of states so if the model is irreducible then all states must be positive-recurrent i.e. the derivation graph is strongly connected.

In terms of the PEPA model this means that all behaviours of the system must be recurrent; in particular, for every choice, whichever path is chosen it must eventually return to the point where the choice can be made again, possibly with a different outcome.

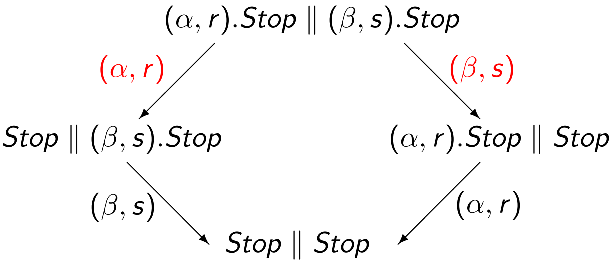
The Importance of Being Exponential

$$\begin{tikzcd}
 & (\alpha, r).Stop \parallel (\beta, s).Stop \arrow[dl, "{(\alpha, r)}"'] \arrow[dr, "{(\beta, s)}"] & \\
Stop \parallel (\beta, s).Stop \arrow[dr, "{(\beta, s)}"'] & & (\alpha, r).Stop \parallel Stop \arrow[dl, "{(\alpha, r)}"] \\
 & Stop \parallel Stop &
\end{tikzcd}$$

The Importance of Being Exponential

$$\begin{tikzcd}
 & (\alpha, r).Stop \parallel (\beta, s).Stop \arrow[dl, "{(\alpha, r)}"'] \arrow[dr, "{(\beta, s)}"] & \\
Stop \parallel (\beta, s).Stop \arrow[dr, "{(\beta, s)}"'] & & (\alpha, r).Stop \parallel Stop \arrow[dl, "{(\alpha, r)}"] \\
 & Stop \parallel Stop &
\end{tikzcd}$$

The Importance of Being Exponential



$(\alpha, r).Stop \parallel (\beta, s).Stop$

(α, r)

(β, s)

$Stop \parallel (\beta, s).Stop$

$(\alpha, r).Stop \parallel Stop$

(β, s)

(α, r)

$Stop \parallel Stop$

The Importance of Being Exponential

$$\begin{tikzcd}
 & (\alpha, r).Stop \parallel (\beta, s).Stop \arrow[dl, "{(\alpha, r)}"'] \arrow[dr, "{(\beta, s)}"] \arrow[dd, "{(\alpha, r) \mid (\beta, s)}"] & \\
Stop \parallel (\beta, s).Stop \arrow[dr, "{(\beta, s)}"'] & & (\alpha, r).Stop \parallel Stop \arrow[dl, "{(\alpha, r)}"] \\
 & Stop \parallel Stop &
\end{tikzcd}$$

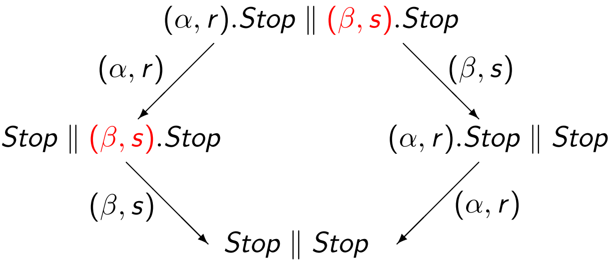
The Importance of Being Exponential

$$\begin{tikzcd}
& (\alpha, r).\mathit{Stop} \parallel (\beta, s).\mathit{Stop} \arrow[dl, "(\alpha, r)"'] \arrow[dr, "(\beta, s)"] & \\
\mathit{Stop} \parallel (\beta, s).\mathit{Stop} \arrow[dr, "(\beta, s)"'] & & (\alpha, r).\mathit{Stop} \parallel \mathit{Stop} \arrow[dl, "(\alpha, r)"] \\
& \mathit{Stop} \parallel \mathit{Stop} &
\end{tikzcd}$$

The Importance of Being Exponential

$$\begin{tikzcd}
 & (\alpha, r).Stop \parallel (\beta, s).Stop \arrow[dl, "(\alpha, r)"'] \arrow[dr, "(\beta, s)"] & \\
Stop \parallel (\beta, s).Stop \arrow[dr, "(\beta, s)"'] & & (\alpha, r).Stop \parallel Stop \arrow[dl, "(\alpha, r)"] \\
 & Stop \parallel Stop &
\end{tikzcd}$$

The Importance of Being Exponential



$$\begin{tikzcd}
 & (\alpha, r).Stop \parallel (\beta, s).Stop \arrow[dl, "{(\alpha, r)}"'] \arrow[dr, "{(\beta, s)}"] & \\
Stop \parallel (\beta, s).Stop \arrow[dr, "{(\beta, s)}"'] & & (\alpha, r).Stop \parallel Stop \arrow[dl, "{(\alpha, r)}"] \\
 & Stop \parallel Stop &
\end{tikzcd}$$

The Importance of Being Exponential

$$\begin{tikzcd}
 & (\alpha, r).Stop \parallel (\beta, s).Stop \arrow[dl, "{(\alpha, r)}"'] \arrow[dr, "{(\beta, s)}"] & \\
Stop \parallel (\beta, s).Stop \arrow[dr, "{(\beta, s)}"'] & & (\alpha, r).Stop \parallel Stop \arrow[dl, "{(\alpha, r)}"] \\
 & Stop \parallel Stop &
\end{tikzcd}$$

The memoryless property of the negative exponential distribution means that residual times do not need to be recorded.

Structured Operational Semantics

PEPA is defined using a Plotkin-style structured operational semantics (a “small step” semantics).

Structured Operational Semantics

PEPA is defined using a Plotkin-style structured operational semantics (a “small step” semantics).

Prefix

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

Structured Operational Semantics

PEPA is defined using a Plotkin-style structured operational semantics (a "small step" semantics).

Prefix

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

Choice

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

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

Structured Operational Semantics: Cooperation ($\alpha \notin L$)

Cooperation

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

Structured Operational Semantics: Cooperation ($\alpha \notin L$)

Cooperation

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

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

Structured Operational Semantics: Cooperation ($\alpha \in L$)

Cooperation

$$\frac{E \xrightarrow{(\alpha,r_1)} E' \qquad F \xrightarrow{(\alpha,r_2)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,R)} E' \underset{L}{\bowtie} F'} \; (\alpha \in L)$$

Structured Operational Semantics: Cooperation ($\alpha \in L$)

Cooperation $$\frac{E \xrightarrow{(\alpha,r_1)} E' \qquad F \xrightarrow{(\alpha,r_2)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,R)} E' \underset{L}{\bowtie} F'} \; (\alpha \in L)$$

where $$R = \frac{r_1}{r_\alpha(E)} \frac{r_2}{r_\alpha(F)} min(r_\alpha(E), r_\alpha(F))$$

Apparent Rate

$$
\begin{array}{rcl}
r_\alpha((\beta, r).P) & = & \left\{ \begin{array}{ll} r & \qquad\qquad\qquad \beta = \alpha \\ 0 & \qquad\qquad\qquad \beta \neq \alpha \end{array} \right. \\
r_\alpha(P + Q) & = & r_\alpha(P) + r_\alpha(Q) \\
r_\alpha(A) & = & r_\alpha(P) \qquad\qquad\qquad \text{where } A \overset{def}{=} P \\
r_\alpha(P \underset{L}{\bowtie} Q) & = & \left\{ \begin{array}{ll} r_\alpha(P) + r_\alpha(Q) & \alpha \notin L \\ min(r_\alpha(P), r_\alpha(Q)) & \alpha \in L \end{array} \right. \\
r_\alpha(P/L) & = & \left\{ \begin{array}{ll} r_\alpha(P) & \qquad\qquad \alpha \notin L \\ 0 & \qquad\qquad \alpha \in L \end{array} \right.
\end{array}
$$

Structured Operational Semantics: Hiding

Hiding

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

Structured Operational Semantics: Hiding

Hiding

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

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

Structured Operational Semantics: Constants

Constant

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

Multiway synchronisation

Cooperation in PEPA is multi-way. Two, three, four or more partners may cooperate, and they all need to synchronise for the activity to happen.

Multiway synchronisation

Cooperation in PEPA is multi-way. Two, three, four or more partners may cooperate, and they all need to synchronise for the activity to happen.

For example, the system

$$\left((\alpha, r).P \underset{\{\alpha\}}{\bowtie} (\alpha, s).Q\right) \underset{\{\alpha\}}{\bowtie} (\alpha, t).R$$

will have a three-way synchronisation between P, Q and R on the activity of type α

Multiway synchronisation

The cooperation sets can make a big difference in the behaviour.

Multiway synchronisation

The cooperation sets can make a big difference in the behaviour.

If we consider again the example from the previous slide but with a small change to the cooperation sets we get different possibilities.

- $((\alpha, r).P \parallel (\alpha, s).Q) \underset{\{\alpha\}}{\bowtie} (\alpha, t).R$
  will have P and Q competing to cooperate with R giving rise to two possible α type activities, only one of which can proceed.

Multiway synchronisation

The cooperation sets can make a big difference in the behaviour.

If we consider again the example from the previous slide but with a small change to the cooperation sets we get different possibilities.

- $((\alpha, r).P \parallel (\alpha, s).Q) \underset{\{\alpha\}}{\bowtie} (\alpha, t).R$
  will have P and Q competing to cooperate with R giving rise to two possible α type activities, only one of which can proceed.
- $\left((\alpha, r).P \underset{\{\alpha\}}{\bowtie} (\alpha, s).Q\right) \parallel (\alpha, t).R$
  will have two α type activities: one synchronising P and Q and one in R alone, both of which can proceed.

Solving PEPA models

- As we have seen a continuous time Markov chain (CTMC) is generated from a PEPA model via its structured operational semantics.

Solving PEPA models

- As we have seen a continuous time Markov chain (CTMC) is generated from a PEPA model via its structured operational semantics.
- Linear algebra is used to solve the model in terms of equilibrium behaviour.

Solving PEPA models

- As we have seen a continuous time Markov chain (CTMC) is generated from a PEPA model via its structured operational semantics.
- Linear algebra is used to solve the model in terms of equilibrium behaviour.
- As we seen previously, the probability distribution can be used to derive performance measures via a reward structure.

The PEPA Eclipse Plug-in

Calculating the transitions of a PEPA model by hand and expressing these in a form which was suitable for solution would be a tedious task prone to errors. The PEPA Eclipse Plug-in relieves the modeller of this work.

The PEPA Eclipse Plug-in: functionality

The plug-in will report errors in the model function:

- deadlock,
- absorbing states,
- static synchronisation mismatch (cooperations which do not involve active participants).

The PEPA Eclipse Plug-in: functionality

The plug-in will report errors in the model function:

- deadlock,
- absorbing states,
- static synchronisation mismatch (cooperations which do not involve active participants).

The plug-in also generates the transition graph of the model, computes the number of states, formulates the Markov process matrix $\boldsymbol{Q}$ and communicates the matrix to a solver.

The PEPA Eclipse Plug-in: functionality

The plug-in will report errors in the model function:

- deadlock,
- absorbing states,
- static synchronisation mismatch (cooperations which do not involve active participants).

The plug-in also generates the transition graph of the model, computes the number of states, formulates the Markov process matrix $\boldsymbol{Q}$ and communicates the matrix to a solver.

The plug-in provides a simple pattern language for selecting states from the stationary distribution.

PEPA Eclipse Plug-In input

$$P_1 \overset{def}{=} (start, r_1).P_2 \qquad P_2 \overset{def}{=} (run, r_2).P_3 \qquad P_3 \overset{def}{=} (stop, r_3).P_1$$

$$P_1 \parallel P_1$$

PEPA Eclipse Plug-In input

$$P_1 \overset{def}{=} (start, r_1).P_2 \qquad P_2 \overset{def}{=} (run, r_2).P_3 \qquad P_3 \overset{def}{=} (stop, r_3).P_1$$

$$P_1 \parallel P_1$$

State space

1	$P_1 \parallel P_1$
2	$P_1 \parallel P_2$
3	$P_2 \parallel P_1$
4	$P_1 \parallel P_3$
5	$P_2 \parallel P_2$
6	$P_3 \parallel P_1$
7	$P_3 \parallel P_2$
8	$P_3 \parallel P_2$
9	$P_3 \parallel P_3$

PEPA Eclipse Plug-In input

$$P_1 \stackrel{def}{=} (start, r_1).P_2 \qquad P_2 \stackrel{def}{=} (run, r_2).P_3 \qquad P_3 \stackrel{def}{=} (stop, r_3).P_1$$

$$P_1 \parallel P_1$$

CTMC representation computed by the plug-in

$$\begin{pmatrix}
-2r_1 & r_1 & r_1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -r_1 - r_2 & 0 & r_2 & r_1 & 0 & 0 & 0 & 0 \\
0 & 0 & -r_1 - r_2 & 0 & r_1 & r_2 & 0 & 0 & 0 \\
r_3 & 0 & 0 & -r_1 - r_3 & 0 & 0 & 0 & r_1 & 0 \\
0 & 0 & 0 & 0 & -2r_2 & 0 & r_2 & r_2 & 0 \\
r_3 & 0 & 0 & 0 & 0 & -r_1 - r_3 & r_1 & 0 & 0 \\
0 & r_3 & 0 & 0 & 0 & 0 & -r_2 - r_3 & 0 & r_2 \\
0 & 0 & r_3 & 0 & 0 & 0 & 0 & -r_2 - r_3 & r_2 \\
0 & 0 & 0 & r_3 & 0 & r_3 & 0 & 0 & -2r_3
\end{pmatrix}$$

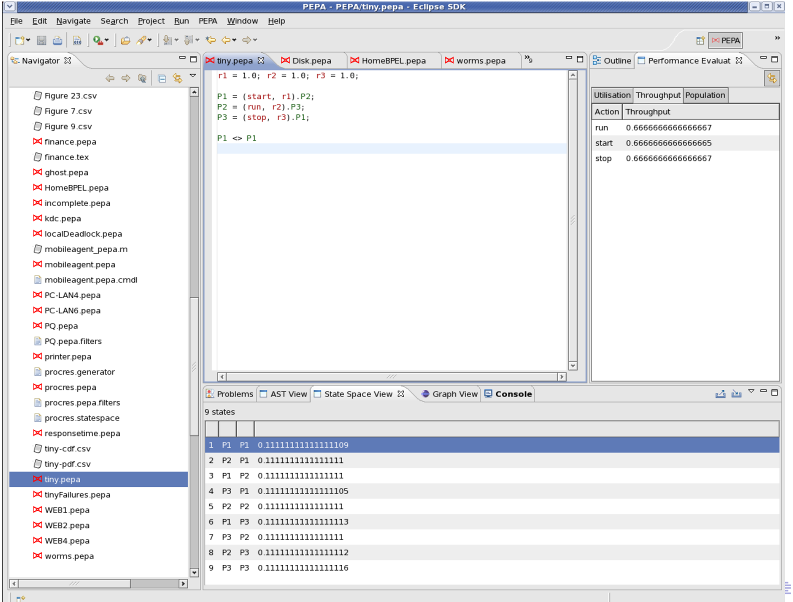


Figure 23.csv
Figure 7.csv
Figure 9.csv
finance.pepa
finance.tex
ghost.pepa
HomeBPEL.pepa
incomplete.pepa
kdc.pepa
localDeadlock.pepa
mobileagent_pepa.m
mobileagent.pepa
mobileagent.pepa.cmdl
PC-LAN4.pepa
PC-LAN6.pepa
PQ.pepa
PQ.pepa.filters
printer.pepa
procres.generator
procres.pepa
procres.pepa.filters
procres.statespace
responsetime.pepa
tiny-cdf.csv
tiny-pdf.csv
tiny.pepa
tinyFailures.pepa
WEB1.pepa
WEB2.pepa
WEB4.pepa
worms.pepa

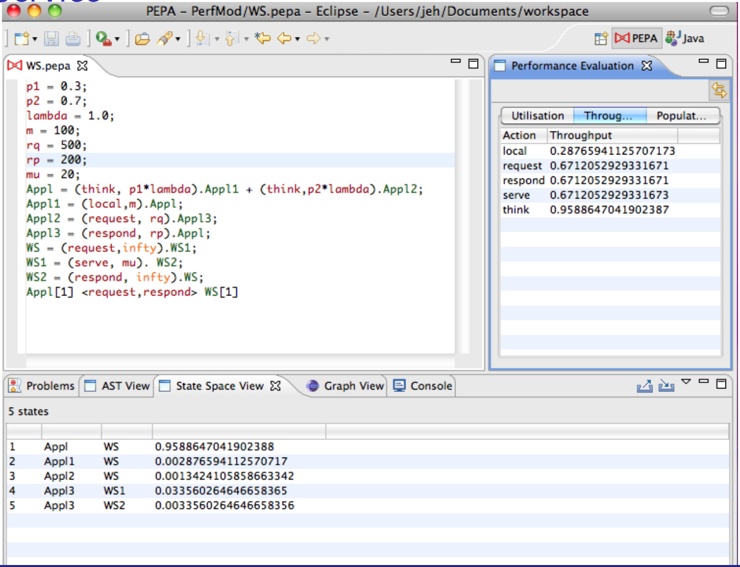
```
r1 = 1.0; r2 = 1.0; r3 = 1.0;

P1 = (start, r1).P2;
P2 = (run, r2).P3;
P3 = (stop, r3).P1;

P1 <> P1
```

Action	Throughput
run	0.6666666666666667
start	0.6666666666666665
stop	0.6666666666666667

9 states

1	P1	P1	0.11111111111111109
2	P2	P1	0.1111111111111111
3	P1	P2	0.1111111111111111
4	P3	P1	0.11111111111111105
5	P2	P2	0.1111111111111111
6	P1	P3	0.11111111111111113
7	P3	P2	0.1111111111111111
8	P2	P3	0.11111111111111112
9	P3	P3	0.11111111111111116

Web Service



```
p1 = 0.3;
p2 = 0.7;
lambda = 1.0;
m = 100;
rq = 500;
rp = 200;
mu = 20;
Appl = (think, p1*lambda).Appl1 + (think,p2*lambda).Appl2;
Appl1 = (local,m).Appl;
Appl2 = (request, rq).Appl3;
Appl3 = (respond, rp).Appl;
WS = (request,infty).WS1;
WS1 = (serve, mu). WS2;
WS2 = (respond, infty).WS;
Appl[1] <request,respond> WS[1]
```

Action	Throughput
local	0.28765941125707173
request	0.6712052929331671
respond	0.6712052929331671
serve	0.6712052929331673
think	0.9588647041902387

5 states

1	Appl	WS	0.9588647041902388
2	Appl1	WS	0.002876594112570717
3	Appl2	WS	0.0013424105858663342
4	Appl3	WS1	0.033560264646658365
5	Appl3	WS2	0.0033560264646658356

The PEPA website

`http://www.dcs.ed.ac.uk/pepa`

From the website the PEPA Eclipse Plug-in is available for download (as well as some other tools).

In particular you will find the plug-in and further instructions at
`http://www.dcs.ed.ac.uk/pepa/tools/plugin/download.html`

There is a short movie which may help you with installing the PEPA Plug-in for Eclipse at
`http://homepages.inf.ed.ac.uk/stg/pepa_eclipse/installing_pepa/`