Advanced Topics in Software Engineering: Stochastic Process Algebras: PEPA

Prof. Michele Loreti

Advanced Topics in Software Engineering
Corso di Laurea in Informatica (L31)
Scuola di Scienze e Tecnologie

Stochastic Process Algebra — Introduction and Motivation

Stochastic process algebras . . .

Stochastic Process Algebra — Introduction and Motivation

Stochastic process algebras . . .

- are formalisms originally developed to model concurrency.

Stochastic Process Algebra — Introduction and Motivation

Stochastic process algebras . . .

- are formalisms originally developed to model concurrency.
- are 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 formalisms originally developed to model concurrency.
- are discrete event modelling formalisms and incorporate timing and probabilistic information with the events in the system.
- 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 formalisms originally developed to model concurrency.
- are discrete event modelling formalisms and incorporate timing and probabilistic information with the events in the system.
- 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;

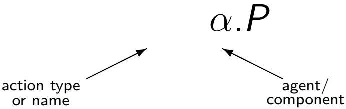
Advantages of compositionality

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.



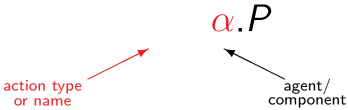
$$\alpha.P$$

action type
or name

agent/
component

Process Algebra

- Models consist of agents which engage in actions.



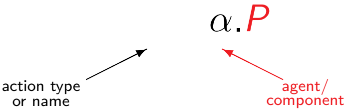
$$\alpha.P$$

action type or name

agent/ component

Process Algebra

- Models consist of agents which engage in actions.



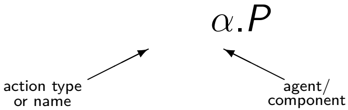
$$\alpha.P$$

action type
or name

agent/
component

Process Algebra

- Models consist of agents which engage in actions.



$$\alpha.P$$

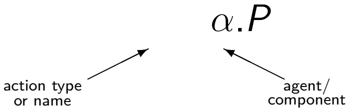
action type
or name

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

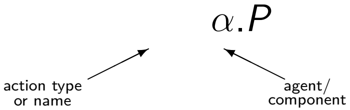
agent/
component

- 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

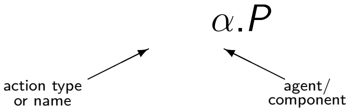
agent/
component

- 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

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

Dynamic behaviour

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

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

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

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

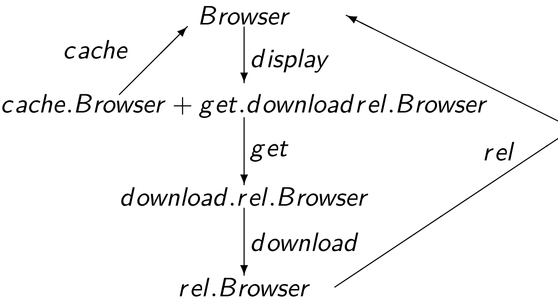
Dynamic behaviour

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

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

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

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



$Browser$

$cache$

$display$

$cache.Browser + get.downloadrel.Browser$

get

rel

$download.rel.Browser$

$download$

$rel.Browser$

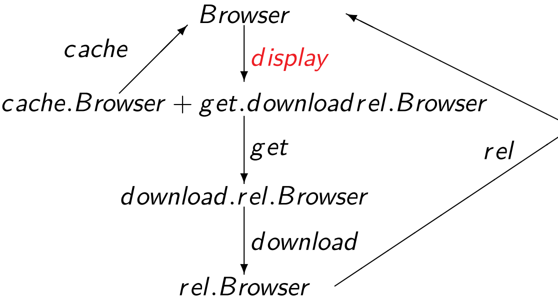
Dynamic behaviour

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

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

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

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



$Browser$

$cache$

$\textcolor{red}{display}$

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

get

rel

$download.rel.Browser$

$download$

$rel.Browser$

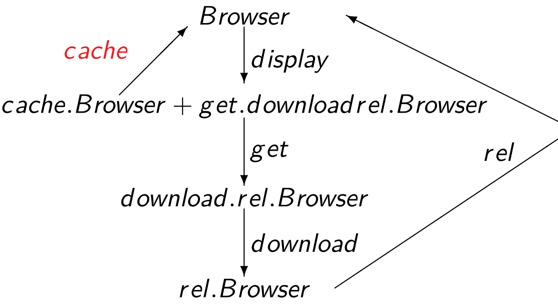
Dynamic behaviour

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

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

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

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



$Browser$

$cache$

$display$

$cache.Browser + get.downloadrel.Browser$

get

rel

$download.rel.Browser$

$download$

$rel.Browser$

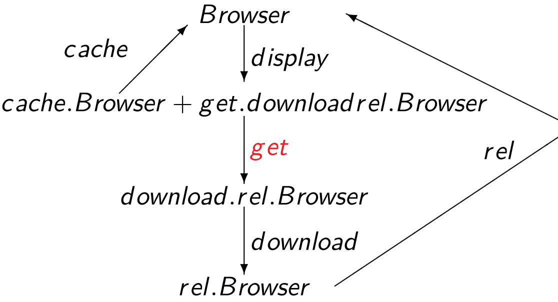
Dynamic behaviour

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

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

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

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



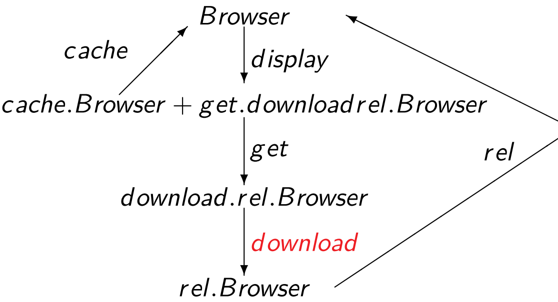
Dynamic behaviour

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

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

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

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



$Browser$

$cache$

$display$

$cache.Browser + get.downloadrel.Browser$

get

rel

$download.rel.Browser$

$download$

$rel.Browser$

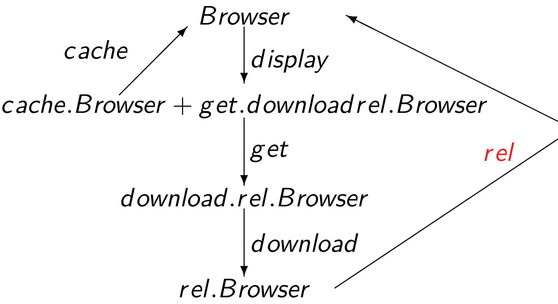
Dynamic behaviour

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

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

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

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



$Browser$

$cache$

$display$

$cache.Browser + get.downloadrel.Browser$

get

rel

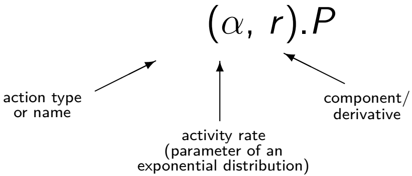
$download.rel.Browser$

$download$

$rel.Browser$

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

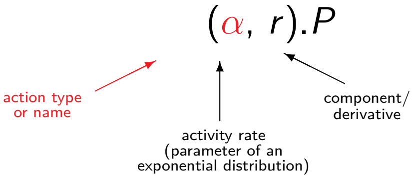
action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

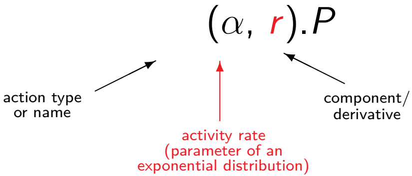
action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

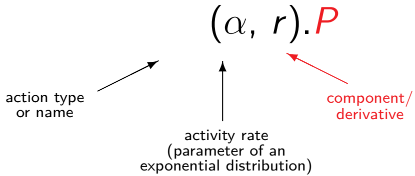
action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

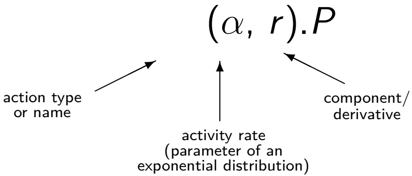
action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

action type or name

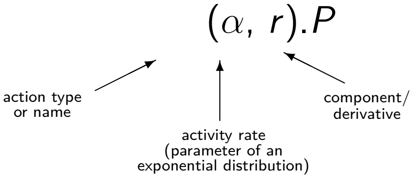
activity rate (parameter of an exponential distribution)

component/ derivative

- 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)

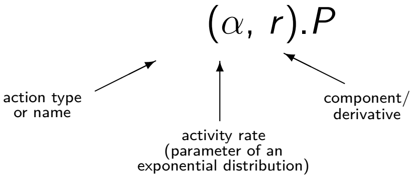
component/
derivative

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

SPA
MODEL

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

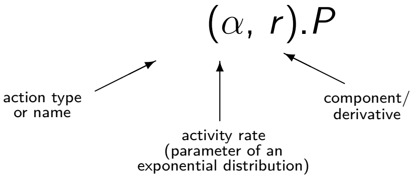
- The language is used to generate a CTMC for performance modelling.

SPA
MODEL

SOS rules

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



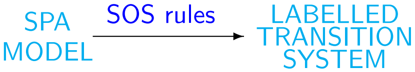
$$(\alpha, r).P$$

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

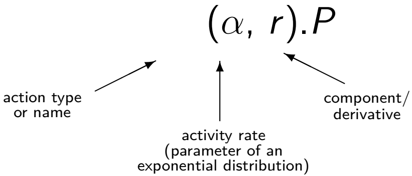
- The language is used to generate a CTMC for performance modelling.



SPA MODEL $\xrightarrow{\text{SOS rules}}$ LABELLED TRANSITION SYSTEM

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

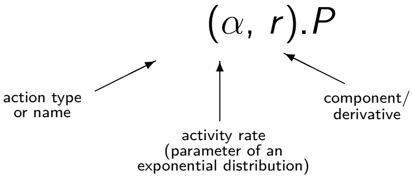
LABELLED
TRANSITION
SYSTEM

state transition

diagram

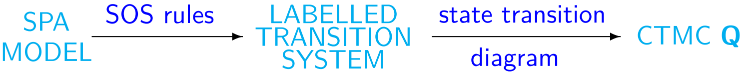
Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

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



PEPA syntax

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

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.

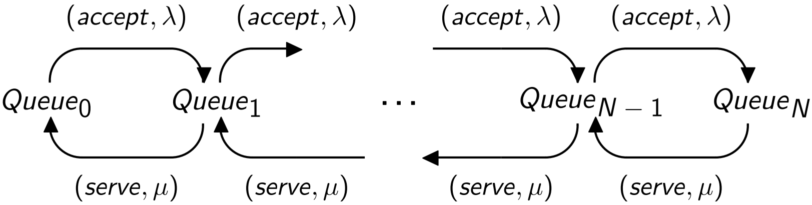
C / L

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

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

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



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

Example: Browsers, server and download

$$\begin{aligned}
Server &\overset{def}{=} (get, \top).(download, \mu).(rel, \top).Server \\
Browser &\overset{def}{=} (display, p\lambda).(get, g).(download, \top).(rel, r).Browser \\
&+ (display, (1-p)\lambda).(cache, m).Browser \\
WEB &\overset{def}{=} \big(Browser \parallel Browser\big) \underset{L}{\bowtie} Server
\end{aligned}$$

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

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

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

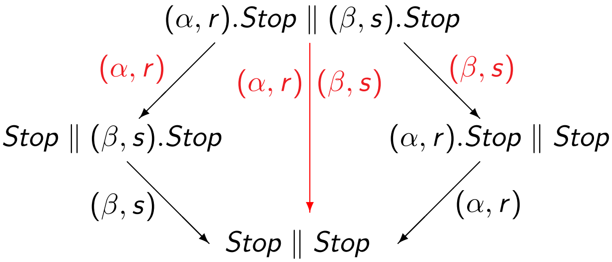
The Importance of Being Exponential

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

The Importance of Being Exponential

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

The Importance of Being Exponential



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

(α, r) (α, r) (β, s) (β, s)

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

(β, s) (α, r)

$Stop \parallel Stop$

The Importance of Being Exponential

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

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

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

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 $(\alpha \in L)$

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

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

Apparent Rate

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

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

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

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 by hand the transitions of a PEPA model and subsequently expressing these in a form which was suitable for solution was 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

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

$$P_1 \parallel P_1$$

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 $P_3 \parallel P_3$

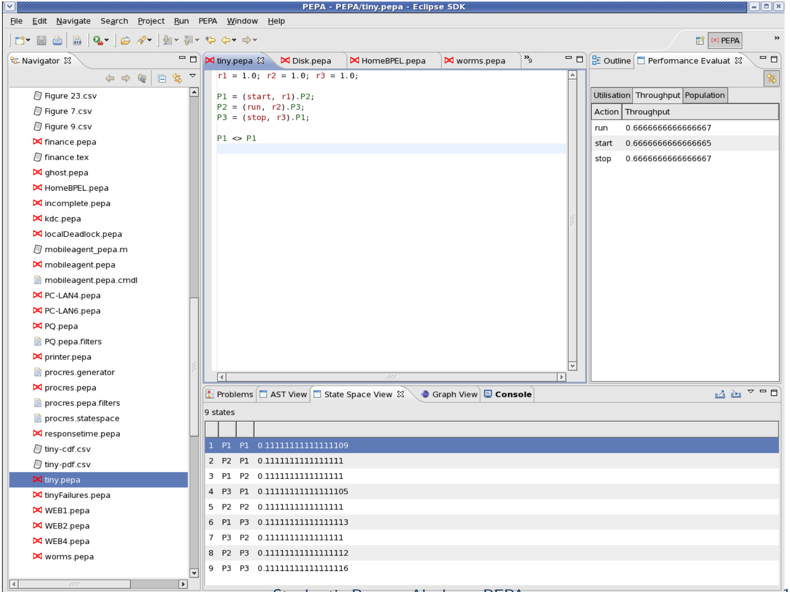
PEPA Eclipse Plug-In input

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

$$P_1 \parallel P_1$$

CTMC representation computed by the plug-in

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



```
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

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/

To be continued. . .

Advanced Topics in Software Engineering: Stochastic Process Algebras: PEPA

Prof. Michele Loreti

Advanced Topics in Software Engineering
Corso di Laurea in Informatica (L31)
Scuola di Scienze e Tecnologie

Stochastic Process Algebra — Introduction and Motivation

Stochastic process algebras . . .

Stochastic Process Algebra — Introduction and Motivation

Stochastic process algebras . . .

- are formalisms originally developed to model concurrency.

Stochastic Process Algebra — Introduction and Motivation

Stochastic process algebras . . .

- are formalisms originally developed to model concurrency.
- are 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 formalisms originally developed to model concurrency.
- are discrete event modelling formalisms and incorporate timing and probabilistic information with the events in the system.
- 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 formalisms originally developed to model concurrency.
- are discrete event modelling formalisms and incorporate timing and probabilistic information with the events in the system.
- 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;

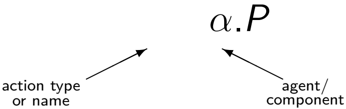
Advantages of compositionality

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.



$$\alpha.P$$

action type or name

agent/ component

Process Algebra

- Models consist of agents which engage in actions.

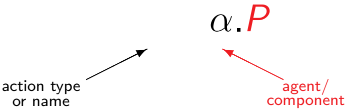
$$\alpha.P$$

action type
or name

agent/
component

Process Algebra

- Models consist of agents which engage in actions.



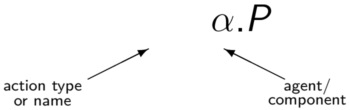
$$\alpha.P$$

action type
or name

agent/
component

Process Algebra

- Models consist of agents which engage in actions.



$$\alpha.P$$

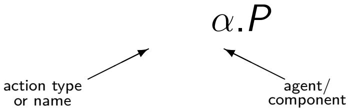
action type
or name

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

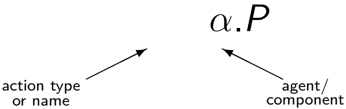
agent/
component

- 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.



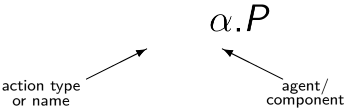
$$\alpha.P$$

- 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.



$$\alpha.P$$

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

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

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

Dynamic behaviour

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

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

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

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

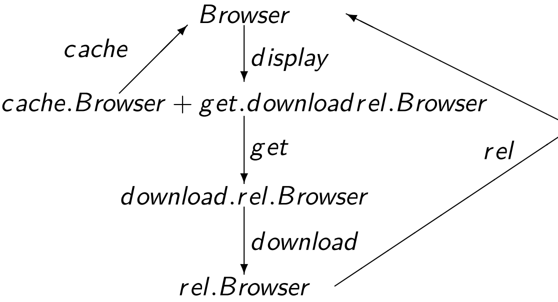
Dynamic behaviour

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

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

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

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



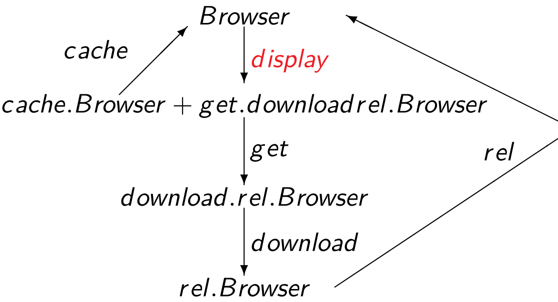
Dynamic behaviour

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

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

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

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



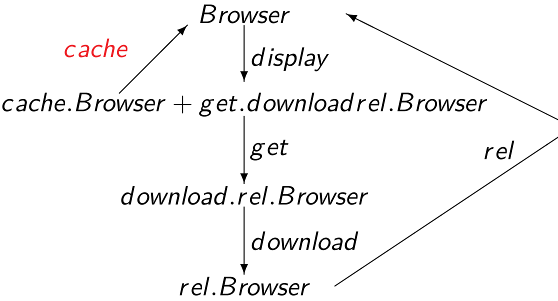
Dynamic behaviour

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

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

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

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


$Browser$

$cache$

$display$

$cache.Browser + get.downloadrel.Browser$

get

rel

$download.rel.Browser$

$download$

$rel.Browser$

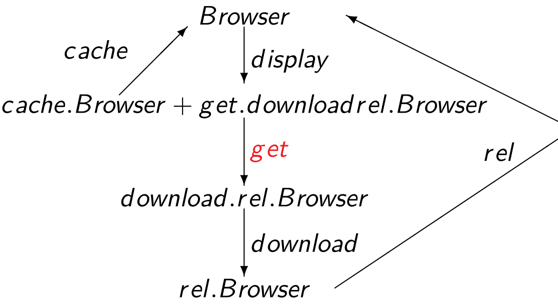
Dynamic behaviour

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

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

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

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



$Browser$

$cache$

$display$

$cache.Browser + get.downloadrel.Browser$

get

rel

$download.rel.Browser$

$download$

$rel.Browser$

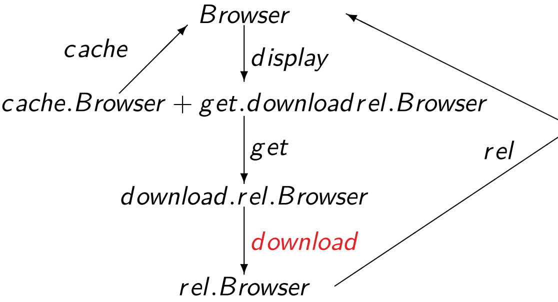
Dynamic behaviour

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

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

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

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



$Browser$

$cache$

$display$

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

get

rel

$download.rel.Browser$

$download$

$rel.Browser$

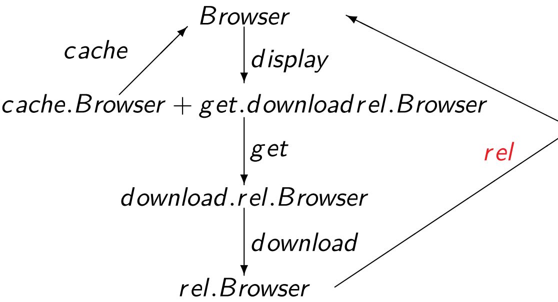
Dynamic behaviour

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

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

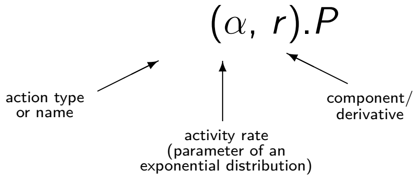
$$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

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



Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

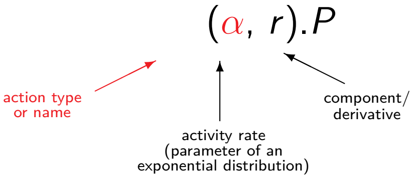
action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

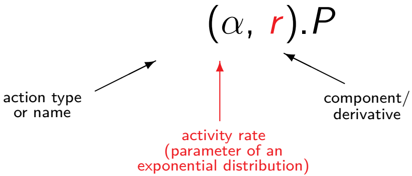
action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

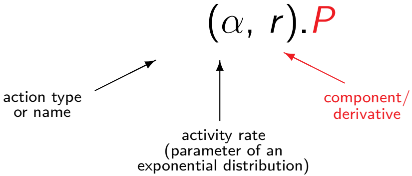
action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

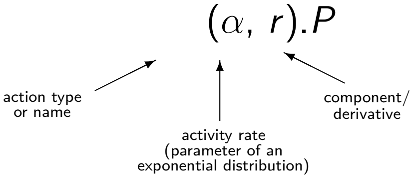
action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

action type
or name

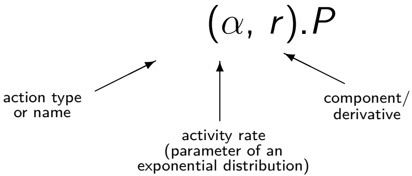
activity rate
(parameter of an
exponential distribution)

component/
derivative

- 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)

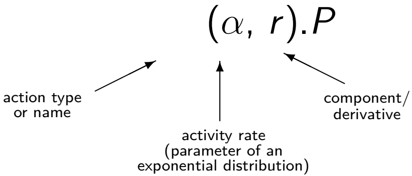
component/
derivative

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

SPA
MODEL

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)

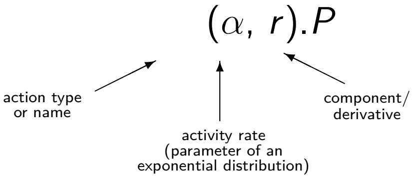
component/ derivative

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

SPA MODEL $\xrightarrow{\text{SOS rules}}$

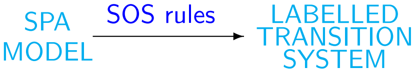
Stochastic Process Algebra

- Models are constructed from components which engage in activities.



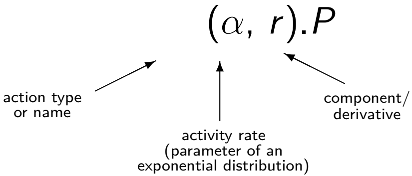
$$(\alpha, r).P$$

- 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

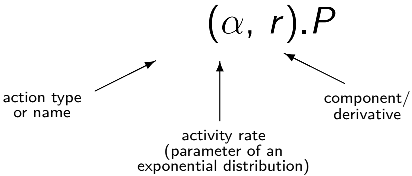
SOS rules

LABELLED TRANSITION SYSTEM

state transition diagram

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



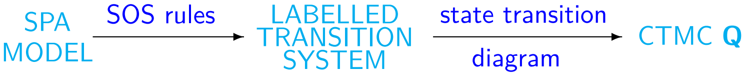
$$(\alpha, r).P$$

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 $\xrightarrow{\text{SOS rules}}$ LABELLED TRANSITION SYSTEM $\xrightarrow[\text{diagram}]{\text{state transition}}$ CTMC **Q**

PEPA syntax

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

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.

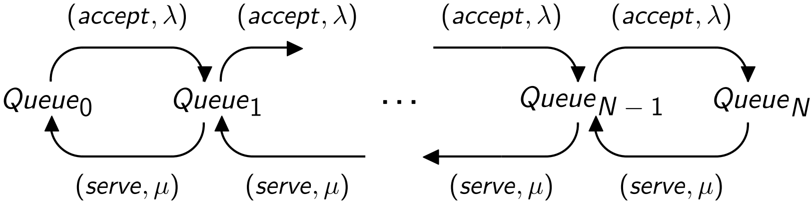
$C \,/\, L$

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

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

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



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

Example: Browsers, server and download

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

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

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

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

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

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

The Importance of Being Exponential

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

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

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

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

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

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

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

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.
- $\Big((\alpha, r).P \underset{\{\alpha\}}{\bowtie} (\alpha, s).Q\Big) \parallel (\alpha, t).R$
  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 by hand the transitions of a PEPA model and subsequently expressing these in a form which was suitable for solution was 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

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

$$P_1 \parallel P_1$$

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	$P_3 \parallel P_3$

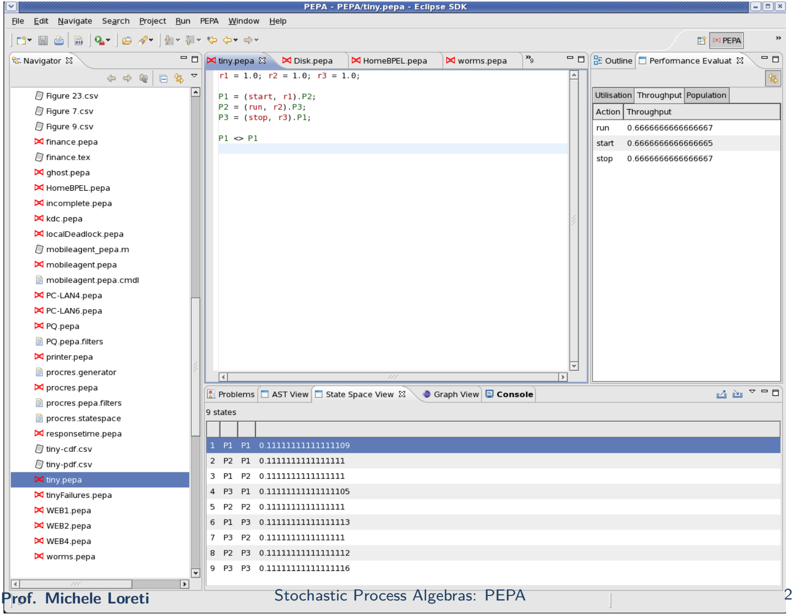
PEPA Eclipse Plug-In input

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

$$P_1 \parallel P_1$$

CTMC representation computed by the plug-in

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



```
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

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/

To be continued. . .

Advanced Topics in Software Engineering: Performance Modelling and Analysis with PEPA

Prof. Michele Loreti

Advanced Topics in Software Engineering
Corso di Laurea in Informatica (L31)
Scuola di Scienze e Tecnologie

Upgrading a PC LAN

Suppose we wish to determine the mean waiting time for data packets at a PC connected to a local area network, operating as a token ring.

Upgrading a PC LAN

Suppose we wish to determine the mean waiting time for data packets at a PC connected to a local area network, operating as a token ring.

The transmission medium supports no more than one transmission at any given time. To resolve conflicts, a token is passed round the network from one node to another in round robin order.

Token ring communication

A node has control of the medium, i.e. it can transmit, only whilst it holds the token.

Token ring communication

A node has control of the medium, i.e. it can transmit, only whilst it holds the token.

In a PC LAN every PC corresponds to a node on the network.

Token ring communication

A node has control of the medium, i.e. it can transmit, only whilst it holds the token.

In a PC LAN every PC corresponds to a node on the network.

Other nodes on the network might be peripheral devices such as printers or faxes but for the purposes of this study we make no distinction and assume that all nodes are PCs.

Upgrading a PC LAN

There are currently four PCs (or similar devices) connected to the LAN in a small office, but the company has recently recruited two new employees, each of whom will have a PC.

Upgrading a PC LAN

There are currently four PCs (or similar devices) connected to the LAN in a small office, but the company has recently recruited two new employees, each of whom will have a PC.

Our task is to find out how the delay experienced by data packets at each PC will be affected if another two PCs are added.

Modelling Assumptions

Each PC can only store one data packet waiting for transmission at a time, so at each visit of the token there is either one packet waiting or no packet waiting. The average rate at which each PC generates data packets for transmission is known to be λ.

Modelling Assumptions

Each PC can only store one data packet waiting for transmission at a time, so at each visit of the token there is either one packet waiting or no packet waiting. The average rate at which each PC generates data packets for transmission is known to be λ.

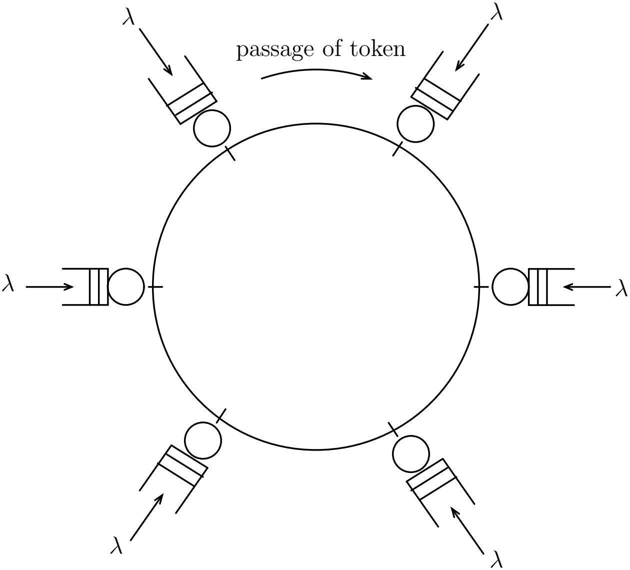
We also know the mean duration, d, of a data packet transmission, and the mean time, m, taken for the token to pass from one PC to the next.

Modelling Assumptions

Each PC can only store one data packet waiting for transmission at a time, so at each visit of the token there is either one packet waiting or no packet waiting. The average rate at which each PC generates data packets for transmission is known to be λ.

We also know the mean duration, d, of a data packet transmission, and the mean time, m, taken for the token to pass from one PC to the next.

It is assumed that if another data packet is generated, whilst the PC is transmitting, this second data packet must wait for the next visit of the token before it can be transmitted. In other words, each PC can transmit at most one data packet per visit of the token.


λ
λ
passage of token
λ
λ
λ
λ

Modelling the system: choosing components

The first stage in developing a model of the system in PEPA is to determine the components of the system and the actions which they can undertake.

Modelling the system: choosing components

The first stage in developing a model of the system in PEPA is to determine the components of the system and the actions which they can undertake.

It seems clear that one type of component should be used to represent the PCs. The components representing the four/six PCs with have essentially the same behaviour. But since token visits the nodes in order we will need to distinguish the components.

Modelling the system: choosing components

The first stage in developing a model of the system in PEPA is to determine the components of the system and the actions which they can undertake.

It seems clear that one type of component should be used to represent the PCs. The components representing the four/six PCs with have essentially the same behaviour. But since token visits the nodes in order we will need to distinguish the components.

We will need another component to represent the medium. As remarked previously, the medium can be represented solely by the token.

Modelling the system: choosing actvities

The description of the PC is very simple in this case. It only has two activities which it can undertake:

- generate a data packet;
- transmit a data packet.

Moreover we are told that it can only hold one data packet at a time and so these activities must be undertaken sequentially.

Modelling the system: choosing actvities

The description of the PC is very simple in this case. It only has two activities which it can undertake:

- generate a data packet;
- transmit a data packet.

Moreover we are told that it can only hold one data packet at a time and so these activities must be undertaken sequentially.

This suggests the following PEPA component for the *i*th PC:

$$\begin{aligned} PC_{i0} &\stackrel{def}{=} (arrive, \lambda).PC_{i1} \\ PC_{i1} &\stackrel{def}{=} (transmit_i, \mu).PC_{i0} \end{aligned}$$

Modelling the system: choosing actvities

The description of the PC is very simple in this case. It only has two activities which it can undertake:

- generate a data packet;
- transmit a data packet.

Moreover we are told that it can only hold one data packet at a time and so these activities must be undertaken sequentially.

This suggests the following PEPA component for the *i*th PC:

$$\begin{aligned} PC_{i0} &\stackrel{def}{=} (arrive, \lambda).PC_{i1} \\ PC_{i1} &\stackrel{def}{=} (transmit_i, \mu).PC_{i0} \end{aligned}$$

This will need some refinement when we consider interaction with the token.

Modelling the system: choosing activities

For the token we can think of its current state being characterised by its current position. Thus, if there are N PCs in the network the states of the token correspond to the values $\{1, 2, \ldots N\}$.

Modelling the system: choosing activities

For the token we can think of its current state being characterised by its current position. Thus, if there are N PCs in the network the states of the token correspond to the values $\{1, 2, \ldots N\}$.

When it is at the ith PC then the token may

- transmit a token if there is one to transmit and then walk on; or
- walk on at once if there is no token waiting.

Modelling the system: choosing activities

For the token we can think of its current state being characterised by its current position. Thus, if there are N PCs in the network the states of the token correspond to the values $\{1, 2, \ldots N\}$.

When it is at the ith PC then the token may

- transmit a token if there is one to transmit and then walk on; or
- walk on at once if there is no token waiting.

$$\mathit{Token}_i \overset{\mathit{def}}{=} (\mathit{walkon}_{i+1}, \omega).\mathit{Token}_{i+1} + \\ (\mathit{transmit}_i, \mu).(\mathit{walk}_{i+1}, \omega).\mathit{Token}_{i+1}$$

Refining the components

In order to ensure that the token's choice is made dependent on the state of PC being visited, we add a walkon action to the PC when it is empty, and impose a cooperation between the PC and the Token for both walkon and serve.

Refining the components

In order to ensure that the token's choice is made dependent on the state of PC being visited, we add a walkon action to the PC when it is empty, and impose a cooperation between the PC and the Token for both walkon and serve.

$$
\begin{array}{lcl}
PC_{i0} & \overset{def}{=} & (arrive, \lambda).PC_{i1} + (walkon_2, \omega).PC_{i0} \\
PC_{i1} & \overset{def}{=} & (transmit_i, \mu).PC_{i0}
\end{array}
$$

Complete model: four PC case

$$
\begin{aligned}
PC_{10} &\stackrel{def}{=} (arrive, \lambda).PC_{11} + (walkon_2, \omega).PC_{10} \\
PC_{11} &\stackrel{def}{=} (transmit_1, \mu).PC_{10} \\
\\
PC_{20} &\stackrel{def}{=} (arrive, \lambda).PC_{21} + (walkon_3, \omega).PC_{20} \\
PC_{21} &\stackrel{def}{=} (transmit_2, \mu).PC_{20} \\
\\
PC_{30} &\stackrel{def}{=} (arrive, \lambda).PC_{31} + (walkon_4, \omega).PC_{30} \\
PC_{31} &\stackrel{def}{=} (transmit_3, \mu).PC_{30} \\
\\
PC_{40} &\stackrel{def}{=} (arrive, \lambda).PC_{41} + (walkon_1, \omega).PC_{40} \\
PC_{41} &\stackrel{def}{=} (transmit_4, \mu).PC_{40}
\end{aligned}
$$

$$\begin{aligned}
Token_1 &\overset{def}{=} (walkon_2, \omega).Token_2 + (transmit_1, \mu).(walk_2, \omega).Token_2 \\
Token_2 &\overset{def}{=} (walkon_3, \omega).Token_3 + (transmit_2, \mu).(walk_3, \omega).Token_3 \\
Token_3 &\overset{def}{=} (walkon_4, \omega).Token_4 + (transmit_3, \mu).(walk_4, \omega).Token_4 \\
Token_4 &\overset{def}{=} (walkon_1, \omega).Token_1 + (transmit_4, \mu).(walk_1, \omega).Token_1
\end{aligned}$$

$$\begin{aligned}
LAN &\overset{def}{=} (PC_{10} \parallel PC_{20} \parallel PC_{30} \parallel PC_{40}) \underset{L}{\bowtie} Token_1 \\
&\text{where } L = \{walkon_1, walkon_2, walkon_3, walkon_4, \\
&\qquad\qquad serve_1, serve_2, serve_3, serve_4\}.
\end{aligned}$$

Here we have arbitrarily chosen a starting state in which all the PCs are empty and the Token is at PC1.

To be continued. . .