SPA for quantitative analysis: Lecture 2 — SPA languages

Jane Hillston

LFCS, School of Informatics
The University of Edinburgh
Scotland

4th March 2013

Outline

1 Process algebra and Markov processes

2 The nature of synchronisation

3 Equivalence Relations

Outline

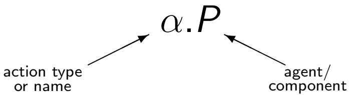
1 Process algebra and Markov processes

2 The nature of synchronisation

3 Equivalence Relations

Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

action type
or name

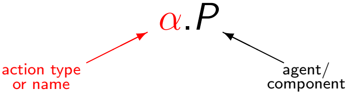
agent/
component

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

- Choices are non-deterministic and time is abstracted.

Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

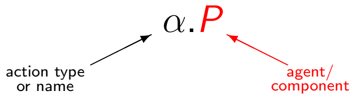
agent/ component

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

- Choices are non-deterministic and time is abstracted.

Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

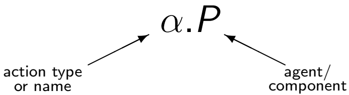
agent/ component

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

- Choices are non-deterministic and time is abstracted.

Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

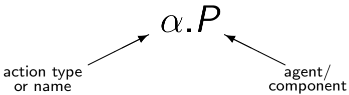
agent/ component

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

- Choices are non-deterministic and time is abstracted.

Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

agent/ component

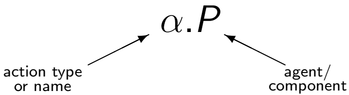
- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process algebra model

- Choices are non-deterministic and time is abstracted.

Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

agent/ component

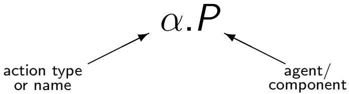
- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

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

- Choices are non-deterministic and time is abstracted.

Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

agent/ component

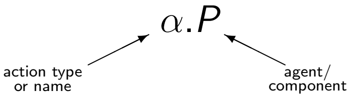
- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

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

- Choices are non-deterministic and time is abstracted.

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 $\xrightarrow{\text{SOS rules}}$ Labelled transition system

- Choices are non-deterministic and time is abstracted.

Example

Consider a web server which offers html pages for download:

$$Server \overset{def}{=} get.download.rel.Server$$

Example

Consider a web server which offers html pages for download:

$$Server \overset{def}{=} get.download.rel.Server$$

Its clients might be web browsers, in a domain with a local cache of frequently requested pages. Thus any display request might result in an access to the server or in a page being loaded from the cache.

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

Example

Consider a web server which offers html pages for download:

$$Server \overset{def}{=} get.download.rel.Server$$

Its clients might be web browsers, in a domain with a local cache of frequently requested pages. Thus any display request might result in an access to the server or in a page being loaded from the cache.

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

A simple version of the Web can be considered to be the interaction of these components:

$$WEB \overset{def}{=} (Browser \parallel Browser) \mid Server$$

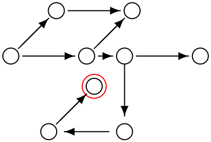
Qualitative Analysis

- The labelled transition system underlying a process algebra model can be used for functional verification e.g.: reachability analysis, specification matching and model checking.

Qualitative Analysis

- The labelled transition system underlying a process algebra model can be used for functional verification e.g.: reachability analysis, specification matching and model checking.

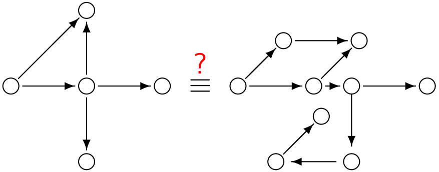
Will the system arrive in a particular state?



Qualitative Analysis

- The labelled transition system underlying a process algebra model can be used for functional verification e.g.: reachability analysis, specification matching and model checking.

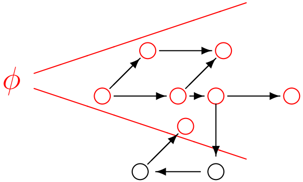
Does system behaviour match its specification?



Qualitative Analysis

- The labelled transition system underlying a process algebra model can be used for functional verification e.g.: reachability analysis, specification matching and model checking.

Does a given property ϕ hold within the system?



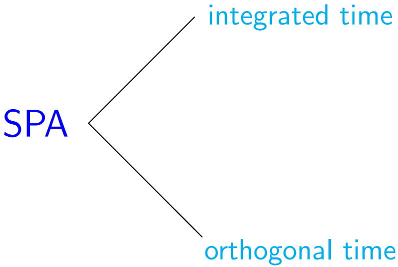
Stochastic process algebras

Process algebras where models are decorated with quantitative information used to generate a stochastic process are stochastic process algebras (SPA).

SPA Languages

SPA

SPA Languages

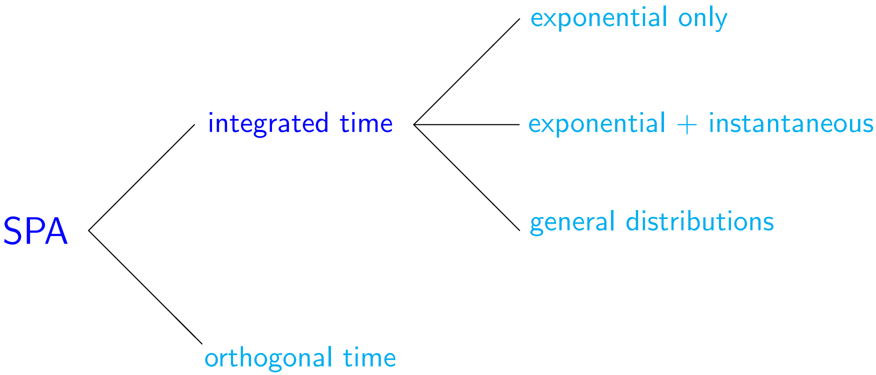


SPA

integrated time

orthogonal time

SPA Languages



SPA

integrated time

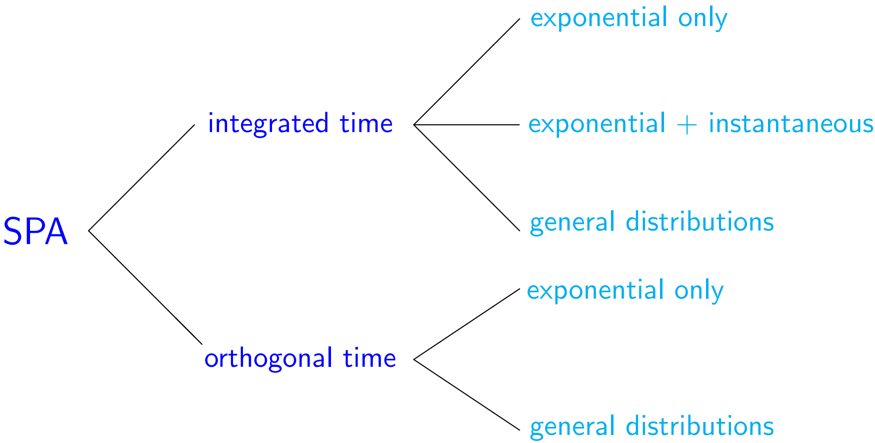
exponential only

exponential + instantaneous

general distributions

orthogonal time

SPA Languages



SPA

integrated time

exponential only

exponential + instantaneous

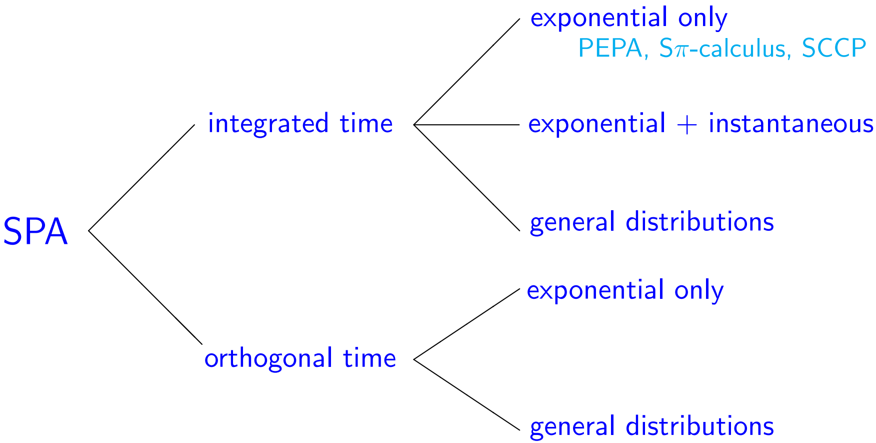
general distributions

orthogonal time

exponential only

general distributions

SPA Languages



SPA

integrated time

exponential only

PEPA, Sπ-calculus, SCCP

exponential + instantaneous

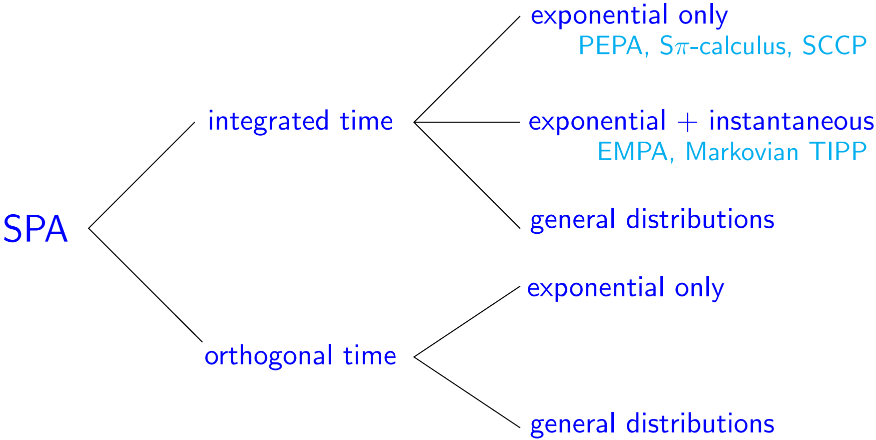
general distributions

orthogonal time

exponential only

general distributions

SPA Languages



SPA

integrated time

exponential only

PEPA, Sπ-calculus, SCCP

exponential + instantaneous

EMPA, Markovian TIPP

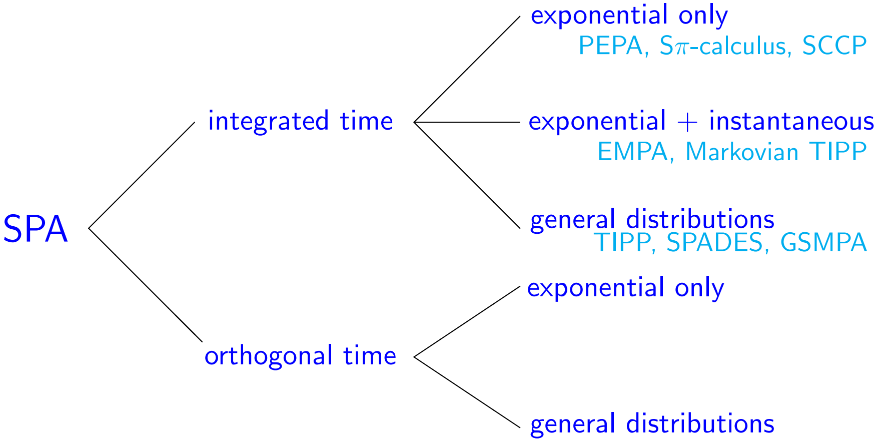
general distributions

orthogonal time

exponential only

general distributions

SPA Languages



SPA

integrated time

exponential only

PEPA, Sπ-calculus, SCCP

exponential + instantaneous

EMPA, Markovian TIPP

general distributions

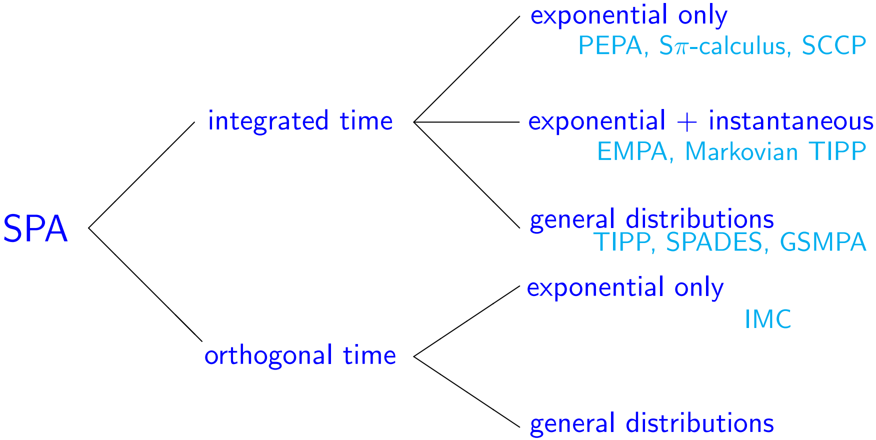
TIPP, SPADES, GSMPA

orthogonal time

exponential only

general distributions

SPA Languages



SPA

integrated time

exponential only

PEPA, Sπ-calculus, SCCP

exponential + instantaneous

EMPA, Markovian TIPP

general distributions

TIPP, SPADES, GSMPA

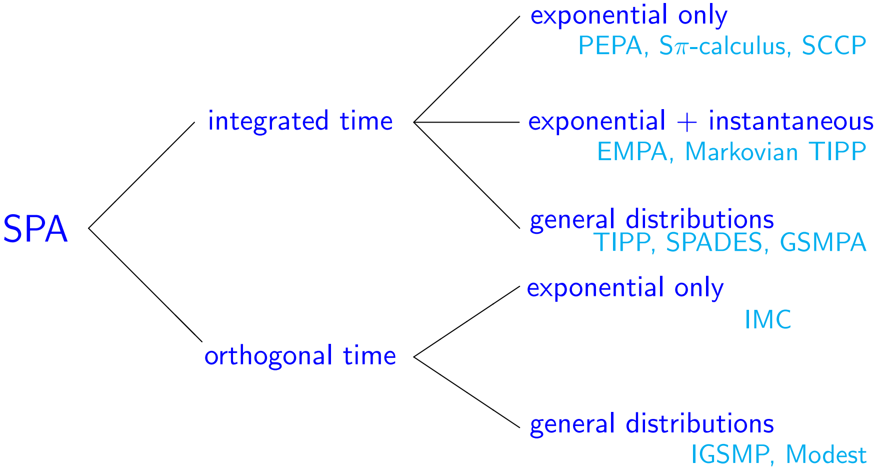
orthogonal time

exponential only

IMC

general distributions

SPA Languages



SPA

integrated time

exponential only

PEPA, Sπ-calculus, SCCP

exponential + instantaneous

EMPA, Markovian TIPP

general distributions

TIPP, SPADES, GSMPA

orthogonal time

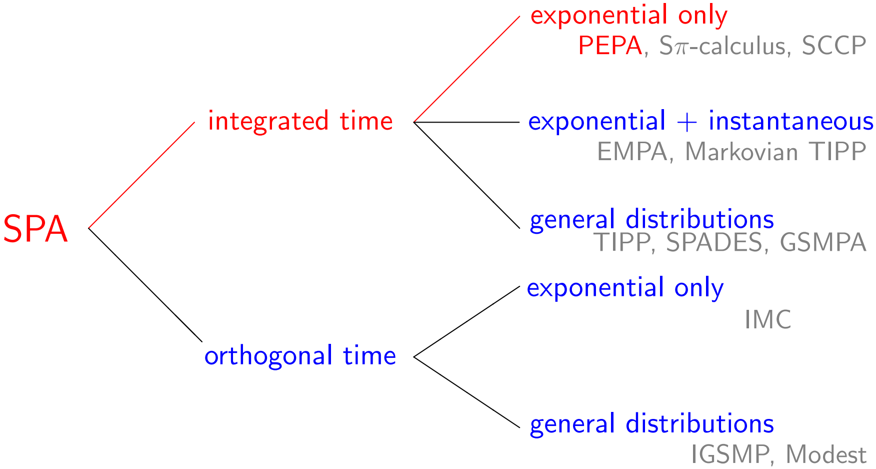
exponential only

IMC

general distributions

IGSMP, Modest

SPA Languages



SPA

integrated time

exponential only

PEPA, Sπ-calculus, SCCP

exponential + instantaneous

EMPA, Markovian TIPP

general distributions

TIPP, SPADES, GSMPA

orthogonal time

exponential only

IMC

general distributions

IGSMP, Modest

Interplay between process algebra and Markov process

- The theoretical development underpinning PEPA has focused on the interplay between the process algebra and the underlying mathematical structure, the Markov process.
- From the process algebra side the Markov chain had a profound influence on the design of the language and in particular on the interactions between components.
- From the Markov chain perspective the process algebra structure has been exploited to find aspects of independence even between interacting components.

Interplay between process algebra and Markov process

- The theoretical development underpinning PEPA has focused on the interplay between the process algebra and the underlying mathematical structure, the Markov process.
- From the process algebra side the Markov chain had a profound influence on the design of the language and in particular on the interactions between components.
- From the Markov chain perspective the process algebra structure has been exploited to find aspects of independence even between interacting components.

Interplay between process algebra and Markov process

- The theoretical development underpinning PEPA has focused on the interplay between the process algebra and the underlying mathematical structure, the Markov process.
- From the process algebra side the Markov chain had a profound influence on the design of the language and in particular on the interactions between components.
- From the Markov chain perspective the process algebra structure has been exploited to find aspects of independence even between interacting components.

Interplay with Performance Modelling

Model Construction: Compositionality leads to

- ease of construction
- reusable submodels
- easy to understand models

Model Manipulation: Equivalence relations lead to

- term rewriting/state space reduction techniques
- aggregation techniques based on lumpability

Model Solution: Formal semantics: lead to

- automatic identification of classes of models susceptible to efficient solution
- use of logics to express performance measures

Interplay with Performance Modelling

Model Construction: Compositionality leads to

- ease of construction
- reusable submodels
- easy to understand models

Model Manipulation: Equivalence relations lead to

- term rewriting/state space reduction techniques
- aggregation techniques based on lumpability

Model Solution: Formal semantics: lead to

- automatic identification of classes of models susceptible to efficient solution
- use of logics to express performance measures

Interplay with Performance Modelling

Model Construction: Compositionality leads to

- ease of construction
- reusable submodels
- easy to understand models

Model Manipulation: Equivalence relations lead to

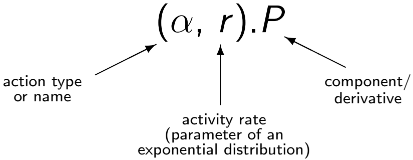
- term rewriting/state space reduction techniques
- aggregation techniques based on lumpability

Model Solution: Formal semantics: lead to

- automatic identification of classes of models susceptible to efficient solution
- use of logics to express performance measures

Integrated time stochastic process algebra

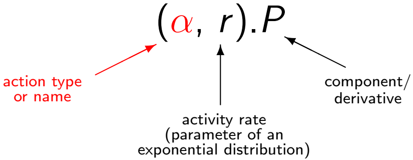
- Models are constructed from components which engage in durational activities.



- The language is used to generate a CTMC for performance modelling. The activity (α, r) will happen before time t with probability $1 - e^{-rt}$.

Integrated time stochastic process algebra

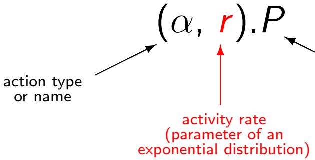
- Models are constructed from components which engage in durational activities.



- The language is used to generate a CTMC for performance modelling. The activity (α, r) will happen before time t with probability $1 - e^{-rt}$.

Integrated time stochastic process algebra

- Models are constructed from components which engage in durational activities.



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

action type
or name

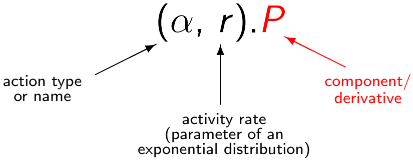
activity rate
(parameter of an
exponential distribution)

component/
derivative

- The language is used to generate a CTMC for performance modelling. The activity (α, r) will happen before time t with probability $1 - e^{-rt}$.

Integrated time stochastic process algebra

- Models are constructed from components which engage in durational activities.



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

action type or name

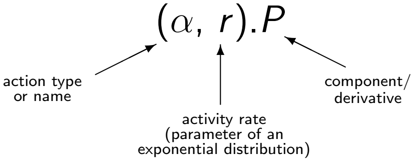
activity rate (parameter of an exponential distribution)

component/ derivative

- The language is used to generate a CTMC for performance modelling. The activity (α, r) will happen before time t with probability $1 - e^{-rt}$.

Integrated time stochastic process algebra

- Models are constructed from components which engage in durational activities.



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

action type
or name

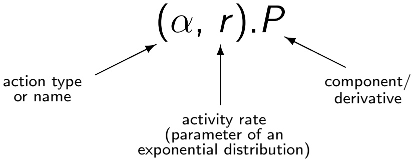
activity rate
(parameter of an
exponential distribution)

component/
derivative

- The language is used to generate a CTMC for performance modelling. The activity (α, r) will happen before time t with probability $1 - e^{-rt}$.

Integrated time stochastic process algebra

- Models are constructed from components which engage in durational activities.



$(\alpha, r).P$

action type or name

activity rate (parameter of an exponential distribution)

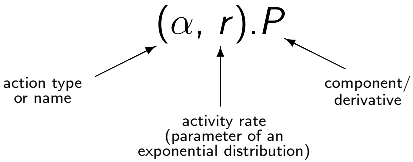
component/ derivative

- The language is used to generate a CTMC for performance modelling. The activity (α, r) will happen before time t with probability $1 - e^{-rt}$.

PEPA
MODEL

Integrated time stochastic process algebra

- Models are constructed from components which engage in durational activities.



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

action type or name

activity rate (parameter of an exponential distribution)

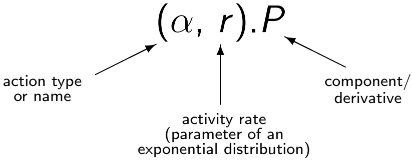
component/ derivative

- The language is used to generate a CTMC for performance modelling. The activity (α, r) will happen before time t with probability $1 - e^{-rt}$.

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

Integrated time stochastic process algebra

- Models are constructed from components which engage in durational activities.



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

action type or name

activity rate (parameter of an exponential distribution)

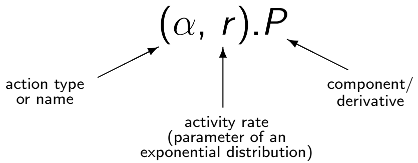
component/ derivative

- The language is used to generate a CTMC for performance modelling. The activity (α, r) will happen before time t with probability $1 - e^{-rt}$.

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

Integrated time stochastic process algebra

- Models are constructed from components which engage in durational activities.



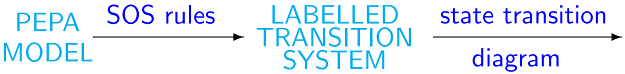
$$(\alpha, r).P$$

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

- The language is used to generate a CTMC for performance modelling. The activity (α, r) will happen before time t with probability $1 - e^{-rt}$.



PEPA MODEL

SOS rules

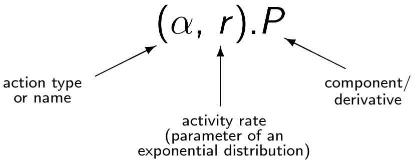
LABELLED TRANSITION SYSTEM

state transition

diagram

Integrated time stochastic process algebra

- Models are constructed from components which engage in durational activities.



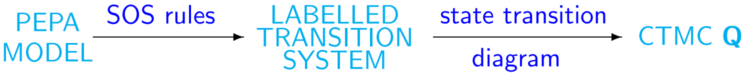
$(\alpha, r).P$

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

- The language is used to generate a CTMC for performance modelling. The activity (α, r) will happen before time t with probability $1 - e^{-rt}$.



PEPA MODEL

SOS rules

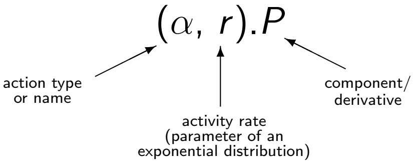
LABELLED TRANSITION SYSTEM

state transition diagram

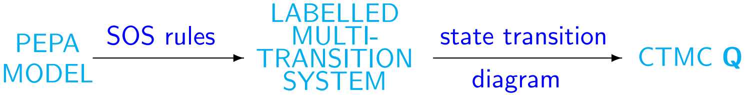
CTMC **Q**

Integrated time stochastic process algebra

- Models are constructed from components which engage in durational activities.



- The language is used to generate a CTMC for performance modelling. The activity (α, r) will happen before time t with probability $1 - e^{-rt}$.



PEPA

$$\begin{array}{rcl} S & ::= & (\alpha, r).S \mid S + S \mid A \\ P & ::= & S \mid P \underset{L}{\bowtie} P \mid P/L \end{array}$$

PEPA

$$
\begin{aligned}
S &::= (\alpha, r).S \mid S + S \mid A \\
P &::= S \mid P \bowtie_{L} P \mid P/L
\end{aligned}
$$

PREFIX: $(\alpha, r).S$ designated first action

PEPA

$$
\begin{aligned}
S &::= (\alpha, r).S \mid S + S \mid A \\
P &::= S \mid P \underset{L}{\bowtie} P \mid P/L
\end{aligned}
$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competing components

PEPA

$$
\begin{array}{rcl}
S & ::= & (\alpha, r).S \mid S + S \mid A \\
P & ::= & S \mid P \underset{L}{\bowtie} P \mid P/L
\end{array}
$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competing components
CONSTANT:	$A \overset{\mathit{def}}{=} S$	assigning names

PEPA

$$
\begin{array}{rcl}
S & ::= & (\alpha, r).S \mid S + S \mid A \\
P & ::= & S \mid P \underset{L}{\bowtie} P \mid P/L
\end{array}
$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competing components
CONSTANT:	$A \overset{def}{=} S$	assigning names
COOPERATION:	$P \underset{L}{\bowtie} P$	$\alpha \notin L$ individual actions
		$\alpha \in L$ shared actions

PEPA

$$
\begin{aligned}
S &::= (\alpha, r).S \mid S + S \mid A \\
P &::= S \mid P \underset{L}{\bowtie} P \mid P/L
\end{aligned}
$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competing components
CONSTANT:	$A \overset{def}{=} S$	assigning names
COOPERATION:	$P \underset{L}{\bowtie} P$	$\alpha \notin L$ individual actions $\alpha \in L$ shared actions
HIDING:	P/L	abstraction $\alpha \in L \Rightarrow \alpha \to \tau$

Example: Browsers, server and download

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

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

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

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

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

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

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

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

Cooperation

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

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

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

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

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

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

Apparent Rate

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

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: 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

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

Properties of the definition (1)

PEPA has no “nil” (a deadlocked process).

This is because the PEPA language is intended for modelling non-stop processes (such as Web servers, operating systems, or manufacturing processes) rather than for modelling terminating processes (a compilation, a sorting routine, and so forth).

Creating a deadlocked process

When we are interested in transient behaviour we use the deadlocked process *Stop* to signal a component which performs no further actions.

$$\mathit{Stop} \overset{\mathit{def}}{=} \Big(((a, r).\mathit{Stop}) \underset{\{a,b\}}{\bowtie} ((b, r).\mathit{Stop}) \Big) / \{ a, b \}$$

Properties of the definition (2)

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.

This comes from the fact that synchronisation has the form $a, a \rightarrow a$ (as in CSP) instead of $a, \bar{a} \rightarrow \tau$ (as in CCS and the π-calculus).

This is used to have "witnesses" to events (known as stochastic probes).

Properties of the definition (2)

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.

This comes from the fact that synchronisation has the form $a, a \rightarrow a$ (as in CSP) instead of $a, \bar{a} \rightarrow \tau$ (as in CCS and the π-calculus).

This is used to have "witnesses" to events (known as stochastic probes).

Properties of the definition (2)

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.

This comes from the fact that synchronisation has the form $a, a \rightarrow a$ (as in CSP) instead of $a, \bar{a} \rightarrow \tau$ (as in CCS and the π-calculus).

This is used to have “witnesses” to events (known as stochastic probes).

Properties of the definition (3)

- Because of its mapping onto a CTMC, PEPA has an interleaving semantics.
- Other modelling formalisms based on CTMCs are also based on an interleaving semantics (e.g. Generalised Stochastic Petri nets).
- 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.
- The resulting probability distribution is seldom the ultimate goal of performance analysis; a modeller derives performance measures from this distribution via a reward structure.

Properties of the definition (3)

- Because of its mapping onto a CTMC, PEPA has an interleaving semantics.
- Other modelling formalisms based on CTMCs are also based on an interleaving semantics (e.g. Generalised Stochastic Petri nets).
- 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.
- The resulting probability distribution is seldom the ultimate goal of performance analysis; a modeller derives performance measures from this distribution via a reward structure.

Properties of the definition (3)

- Because of its mapping onto a CTMC, PEPA has an interleaving semantics.
- Other modelling formalisms based on CTMCs are also based on an interleaving semantics (e.g. Generalised Stochastic Petri nets).
- 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.
- The resulting probability distribution is seldom the ultimate goal of performance analysis; a modeller derives performance measures from this distribution via a reward structure.

Properties of the definition (3)

- Because of its mapping onto a CTMC, PEPA has an interleaving semantics.
- Other modelling formalisms based on CTMCs are also based on an interleaving semantics (e.g. Generalised Stochastic Petri nets).
- 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.
- The resulting probability distribution is seldom the ultimate goal of performance analysis; a modeller derives performance measures from this distribution via a reward structure.

Properties of the definition (3)

- Because of its mapping onto a CTMC, PEPA has an interleaving semantics.
- Other modelling formalisms based on CTMCs are also based on an interleaving semantics (e.g. Generalised Stochastic Petri nets).
- 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.
- The resulting probability distribution is seldom the ultimate goal of performance analysis; a modeller derives performance measures from this distribution via a reward structure.

Dynamic behaviour

$$Comp \overset{def}{=} (first, f).((slow, r/3).Comp \; + \; (quick, r).(rapid, r).(fast, r).Comp)$$

Dynamic behaviour

$$Comp \overset{def}{=} (first, f).((slow, r/3).Comp \; + \; (quick, r).(rapid, r).(fast, r).Comp)$$

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

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

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

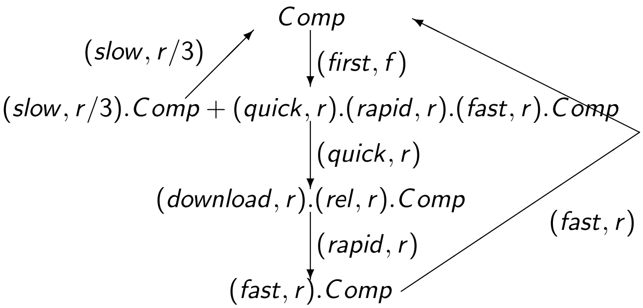
Dynamic behaviour

$$Comp \overset{def}{=} (first, f).((slow, r/3).Comp \ + \ (quick, r).(rapid, r).(fast, r).Comp)$$

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

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

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


$Comp$
$(slow, r/3)$
$(first, f)$
$(slow, r/3).Comp + (quick, r).(rapid, r).(fast, r).Comp$
$(quick, r)$
$(download, r).(rel, r).Comp$
$(fast, r)$
$(rapid, r)$
$(fast, r).Comp$

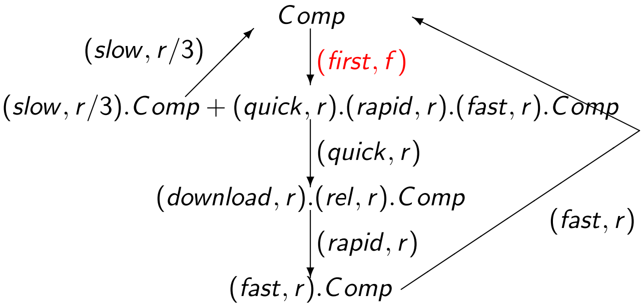
Dynamic behaviour

$$Comp \overset{def}{=} (first, f).((slow, r/3).Comp \; + \; (quick, r).(rapid, r).(fast, r).Comp)$$

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

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

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



$Comp$

$(slow, r/3)$

$(first, f)$

$(slow, r/3).Comp + (quick, r).(rapid, r).(fast, r).Comp$

$(quick, r)$

$(download, r).(rel, r).Comp$

$(rapid, r)$

$(fast, r).Comp$

$(fast, r)$

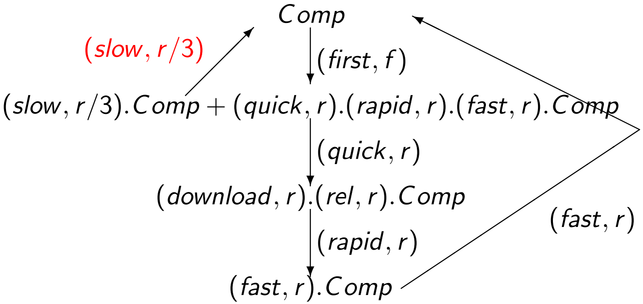
Dynamic behaviour

$$Comp \overset{def}{=} (first, f).((slow, r/3).Comp \; + \; (quick, r).(rapid, r).(fast, r).Comp)$$

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

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

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


$Comp$
$(slow, r/3)$
$(first, f)$
$(slow, r/3).Comp + (quick, r).(rapid, r).(fast, r).Comp$
$(quick, r)$
$(download, r).(rel, r).Comp$
$(fast, r)$
$(rapid, r)$
$(fast, r).Comp$

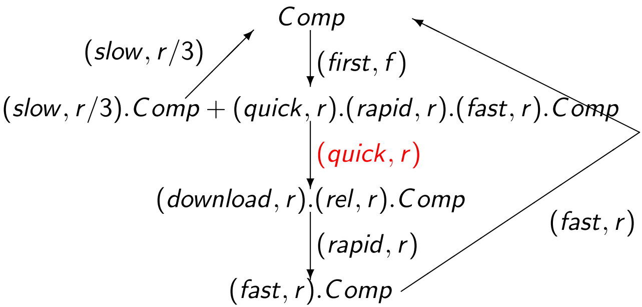
Dynamic behaviour

$$Comp \stackrel{\mathit{def}}{=} (first, f).((slow, r/3).Comp \; + \; (quick, r).(rapid, r).(fast, r).Comp)$$

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

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

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


$Comp$
$(slow, r/3)$
$(first, f)$
$(slow, r/3).Comp + (quick, r).(rapid, r).(fast, r).Comp$
$(quick, r)$
$(download, r).(rel, r).Comp$
$(rapid, r)$
$(fast, r)$
$(fast, r).Comp$

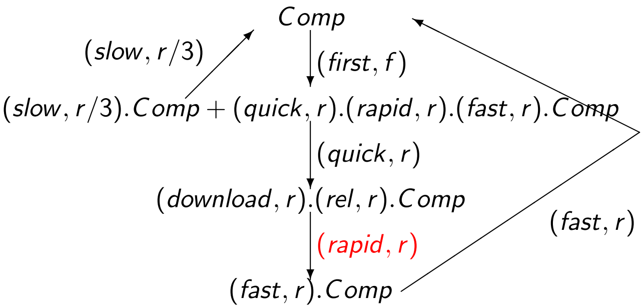
Dynamic behaviour

$$Comp \overset{def}{=} (first, f).((slow, r/3).Comp \ + \ (quick, r).(rapid, r).(fast, r).Comp)$$

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

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

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



$Comp$

$(first, f)$

$(slow, r/3)$

$(slow, r/3).Comp + (quick, r).(rapid, r).(fast, r).Comp$

$(quick, r)$

$(download, r).(rel, r).Comp$

$(rapid, r)$

$(fast, r)$

$(fast, r).Comp$

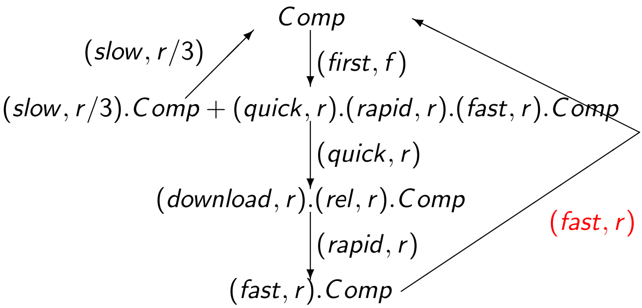
Dynamic behaviour

$$Comp \overset{def}{=} (first, f).((slow, r/3).Comp \ + \ (quick, r).(rapid, r).(fast, r).Comp)$$

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

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

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



$Comp$

$(first, f)$

$(slow, r/3)$

$(slow, r/3).Comp + (quick, r).(rapid, r).(fast, r).Comp$

$(quick, r)$

$(download, r).(rel, r).Comp$

$(rapid, r)$

$(fast, r).Comp$

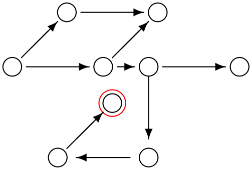
$(fast, r)$

Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

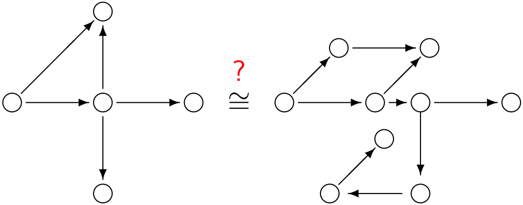
Integrated analysis: Reachability analysis

How long will it take for the system to arrive in a particular state?



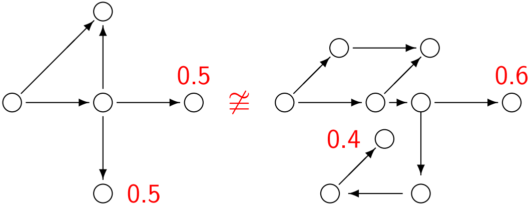
Integrated analysis: Specification matching

With what probability does system behaviour match its specification?



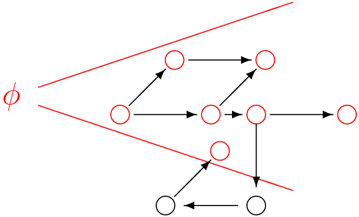
Integrated analysis: Specification matching

Does the “*frequency profile*” of the system match that of the specification?



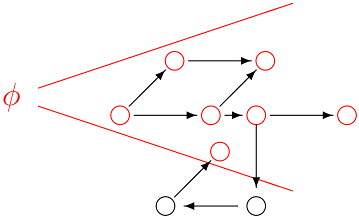
Integrated analysis: Model checking

Does a given property ϕ hold within the system with a given probability?



Integrated analysis: Model checking

For a given starting state
how long is it until
a given property ϕ holds?



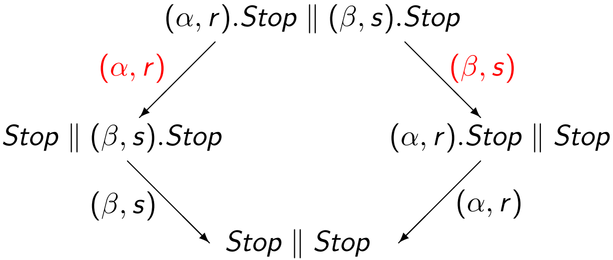
The Importance of Being Exponential

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

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)

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

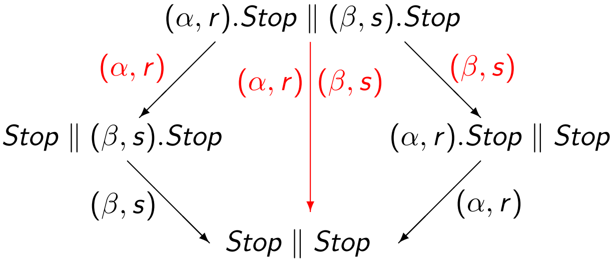
$(\alpha, r).Stop \parallel Stop$

(β, s)

(α, r)

$Stop \parallel Stop$

The Importance of Being Exponential



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

(α, r)

$(\alpha, r) \mid (\beta, s)$

(β, s)

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

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

(β, s)

(α, r)

$Stop \parallel Stop$

The Importance of Being Exponential

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

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

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

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

The exponential distribution and the expansion law

We retain the expansion law of classical process algebra:

$$
\begin{aligned}
&(\alpha, r).\mathit{Stop} \parallel (\beta, s).\mathit{Stop} = \\
&\qquad\qquad (\alpha, r).(\beta, s).(\mathit{Stop} \parallel \mathit{Stop}) + (\beta, s).(\alpha, r).(\mathit{Stop} \parallel \mathit{Stop})
\end{aligned}
$$

only if the negative exponential distribution is used.

Outline

1. Process algebra and Markov processes

2. The nature of synchronisation

3. Equivalence Relations

Parallel Composition

- Parallel composition is the basis of the compositionality in a process algebra
- In classical process algebra is it often associated with communication.
- When the activities of the process algebra have a duration the definition of parallel composition becomes more complex.
- The issue of what it means for two timed activities to synchronise is a vexed one....

Parallel Composition

- Parallel composition is the basis of the compositionality in a process algebra — it defines which components interact and how.
- In classical process algebra is it often associated with communication.
- When the activities of the process algebra have a duration the definition of parallel composition becomes more complex.
- The issue of what it means for two timed activities to synchronise is a vexed one....

Parallel Composition

- Parallel composition is the basis of the compositionality in a process algebra — it defines which components interact and how.
- In classical process algebra is it often associated with communication.
- When the activities of the process algebra have a duration the definition of parallel composition becomes more complex.
- The issue of what it means for two timed activities to synchronise is a vexed one....

Parallel Composition

- Parallel composition is the basis of the compositionality in a process algebra — it defines which components interact and how.
- In classical process algebra is it often associated with communication.
- When the activities of the process algebra have a duration the definition of parallel composition becomes more complex.
- The issue of what it means for two timed activities to synchronise is a vexed one....

Parallel Composition

- Parallel composition is the basis of the compositionality in a process algebra — it defines which components interact and how.
- In classical process algebra is it often associated with communication.
- When the activities of the process algebra have a duration the definition of parallel composition becomes more complex.
- The issue of what it means for two timed activities to synchronise is a vexed one....

Who Synchronises...?

Even within classical process algebras there is variation in the interpretation of parallel composition:

Who Synchronises...?

Even within classical process algebras there is variation in the interpretation of parallel composition:

CCS-style

- Actions are partitioned into input and output pairs.
- Communication or synchronisation takes places between conjugate pairs.
- The resulting action has silent type τ.

Who Synchronises...?

Even within classical process algebras there is variation in the interpretation of parallel composition:

CCS-style

- Actions are partitioned into input and output pairs.
- Communication or synchronisation takes places between conjugate pairs.
- The resulting action has silent type τ.

CSP-style

- No distinction between input and output actions.
- Communication or synchronisation takes place on the basis of shared names.
- The resulting action has the same name.

Who Synchronises...?

Even within classical process algebras there is variation in the interpretation of parallel composition:

CCS-style

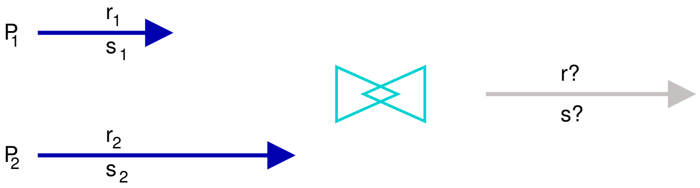
- Actions are partitioned into input and output pairs.
- Communication or synchronisation takes places between conjugate pairs.
- The resulting action has silent type τ.

CSP-style

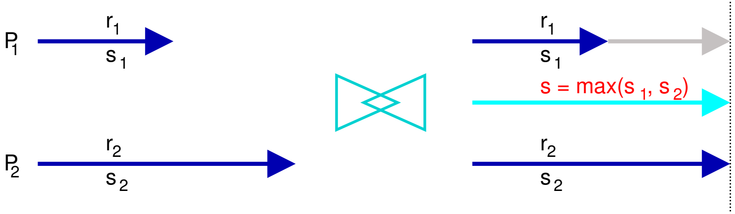
- No distinction between input and output actions.
- Communication or synchronisation takes place on the basis of shared names.
- The resulting action has the same name.

Most stochastic process algebras adopt CSP-style synchronisation.

Timed Synchronisation

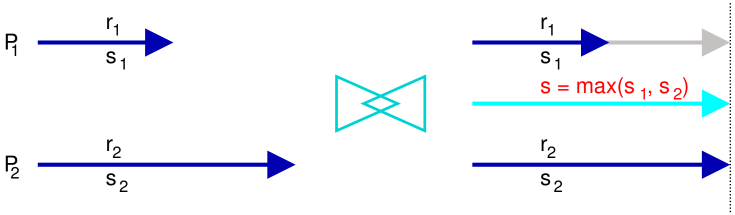


Timed Synchronisation



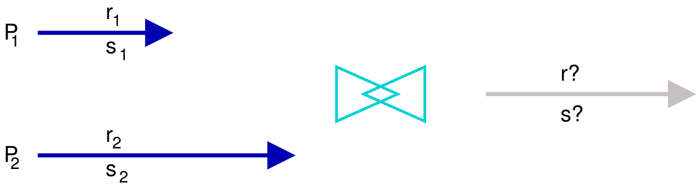
Barrier Synchronisation

Timed Synchronisation



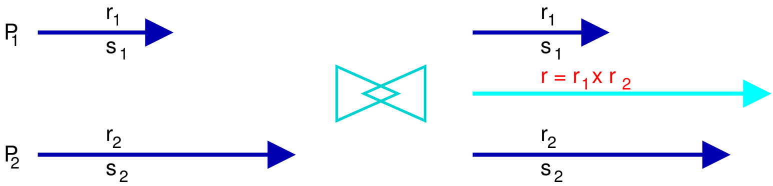
s is no longer exponentially distributed

Timed Synchronisation



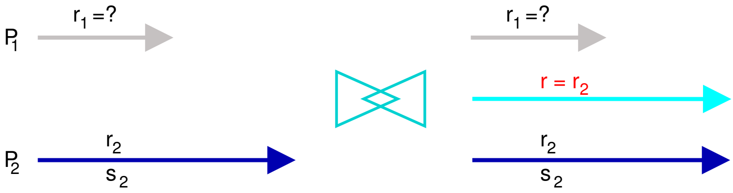
algebraic considerations limit choices

Timed Synchronisation



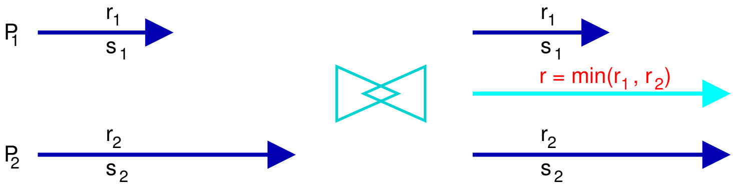
TIPP: new rate is product of individual rates

Timed Synchronisation



EMPA: one participant is passive

Timed Synchronisation



bounded capacity: new rate is the minimum of the rates

Cooperation in PEPA

- In PEPA each component has a bounded capacity to carry out activities of any particular type, determined by the apparent rate for that type.
- Synchronisation, or cooperation cannot make a component exceed its bounded capacity.
- Thus the apparent rate of a cooperation is the minimum of the apparent rates of the co-operands.

Cooperation in PEPA

- In PEPA each component has a bounded capacity to carry out activities of any particular type, determined by the apparent rate for that type.
- Synchronisation, or cooperation cannot make a component exceed its bounded capacity.
- Thus the apparent rate of a cooperation is the minimum of the apparent rates of the co-operands.

Cooperation in PEPA

- In PEPA each component has a bounded capacity to carry out activities of any particular type, determined by the apparent rate for that type.
- Synchronisation, or cooperation cannot make a component exceed its bounded capacity.
- Thus the apparent rate of a cooperation is the minimum of the apparent rates of the co-operands.

Apparent rate

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

It is defined as:

$$r_\alpha(P) = \sum_{P \xrightarrow{(\alpha,\lambda_i)}} \lambda_i$$

where $\lambda_i \in \mathbb{R}^+ \cup \{n\top \mid n \in \mathbb{Q}, n > 0\}$.

$n\top$ is shorthand for $n \times \top$ and $\top$ represents the passive action rate that inherits the rate of the coaction from the cooperating component.

Apparent rate

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

It is defined as:

$$r_\alpha(P) = \sum_{P \xrightarrow{(\alpha, \lambda_i)}} \lambda_i$$

where $\lambda_i \in \mathbb{R}^+ \cup \{n\top \mid n \in \mathbb{Q}, n > 0\}$.

$n\top$ is shorthand for $n \times \top$ and $\top$ represents the passive action rate that inherits the rate of the coaction from the cooperating component.

Apparent rate

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

It is defined as:

$$r_\alpha(P) = \sum_{P \xrightarrow{(\alpha,\lambda_i)}} \lambda_i$$

where $\lambda_i \in \mathbb{R}^+ \cup \{n\top \mid n \in \mathbb{Q}, n > 0\}$.

$n\top$ is shorthand for $n \times \top$ and $\top$ represents the passive action rate that inherits the rate of the coaction from the cooperating component.

Properties of $\top$ (the "unspecified" symbol)

$\top$ requires the following arithmetic rules:

$$\begin{aligned} m\top < n\top \quad &: \quad \text{for } m < n \text{ and } m, n \in \mathbb{Q} \\ r < n\top \quad &: \quad \text{for all } r \in \mathbb{R}, n \in \mathbb{Q} \\ m\top + n\top = (m+n)\top \quad &: \quad m, n \in \mathbb{Q} \\ \frac{m\top}{n\top} = \frac{m}{n} \quad &: \quad m, n \in \mathbb{Q} \end{aligned}$$

Note that $(r + n\top)$ is undefined for all $r \in \mathbb{R}$ in PEPA therefore disallowing components which enable both active and passive actions in the same action type at the same time, e.g. $(\alpha, \lambda).P + (\alpha, \top).P'$.

Properties of $\top$ (the “unspecified” symbol)

$\top$ requires the following arithmetic rules:

$$\begin{aligned} m\top < n\top &\quad : \quad \text{for } m < n \text{ and } m, n \in \mathbb{Q} \\ r < n\top &\quad : \quad \text{for all } r \in \mathbb{R}, n \in \mathbb{Q} \\ m\top + n\top = (m+n)\top &\quad : \quad m, n \in \mathbb{Q} \\ \frac{m\top}{n\top} = \frac{m}{n} &\quad : \quad m, n \in \mathbb{Q} \end{aligned}$$

Note that $(r + n\top)$ is undefined for all $r \in \mathbb{R}$ in PEPA therefore disallowing components which enable both active and passive actions in the same action type at the same time, e.g. $(\alpha, \lambda).P + (\alpha, \top).P'$.

Outline

1 Process algebra and Markov processes

2 The nature of synchronisation

3 Equivalence Relations

Equivalence relations in Performance Modelling

Equivalence relations are used, often informally, in performance modelling to manipulate models into an alternative form which is somehow easier to solve:

Model simplification: use a model-model equivalence to substitute one model by another which is more attractive from a solution point of view, e.g. smaller state space, special class of model, etc.

Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state, i.e. take a different stochastic representation of the same model.

Equivalence relations in Performance Modelling

Equivalence relations are used, often informally, in performance modelling to manipulate models into an alternative form which is somehow easier to solve:

Model simplification: use a model-model equivalence to substitute one model by another which is more attractive from a solution point of view, e.g. smaller state space, special class of model, etc.

Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state, i.e. take a different stochastic representation of the same model.

Aggregation and lumpability

- Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state.
- This is not as straightforward as it may seem if we wish the aggregated process to still be a Markov process — an arbitrary partition will not in general preserve the Markov property.
- A lumpable partition is the only partition of a Markov process which preserves the Markov property.

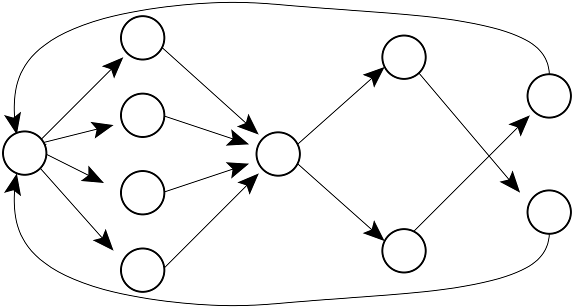
Aggregation and lumpability

- Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state.
- This is not as straightforward as it may seem if we wish the aggregated process to still be a Markov process — an arbitrary partition will not in general preserve the Markov property.
- A lumpable partition is the only partition of a Markov process which preserves the Markov property.

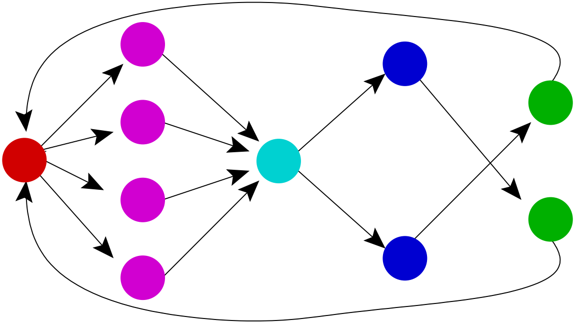
Aggregation and lumpability

- Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state.
- This is not as straightforward as it may seem if we wish the aggregated process to still be a Markov process — an arbitrary partition will not in general preserve the Markov property.
- A lumpable partition is the only partition of a Markov process which preserves the Markov property.

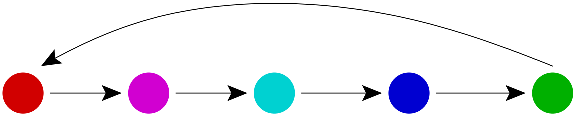
Reducing by lumpability



Reducing by lumpability



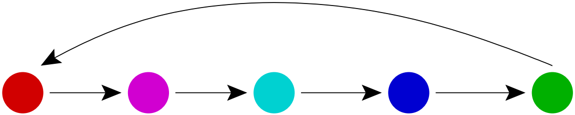
Reducing by lumpability



As appealling as this is, it is not the case that it is always mathematically legitimate.

In particular, arbitarily lumping the states of a Markov chain, will typically give rise to a stochastic process which no longer satisfies the Markov condition.

Reducing by lumpability



As appealling as this is, it is not the case that it is always mathematically legitimate.

In particular, arbitarily lumping the states of a Markov chain, will typically give rise to a stochastic process which no longer satisfies the Markov condition.

Equivalence Relations in Process Algebras

- It is standard for a process algebra to be equipped with an equivalence relation based on the semantics.
- Many different styles of equivalences have been defined, but the most fundamental is perhaps the bisimulation.
- Bisimulation is based on the notion of observability.
- An external observer should not be able to distinguish between two equivalent processes.

Bisimulation

In classical process algebras such as CCS a bisimulation has the following form:

A relation $\mathcal{R}$ is a strong bisimulation relation if $(P, Q) \in \mathcal{R}$ implies

1. whenever $P \xrightarrow{\alpha} P'$, then there exists Q' such that $Q \xrightarrow{\alpha} Q'$, and $(P', Q') \in \mathcal{R}$;
2. whenever $Q \xrightarrow{\alpha} Q'$, then there exists P' such that $P \xrightarrow{\alpha} P'$, and $(P', Q') \in \mathcal{R}$.

Strong bisimulation $\sim$ is the largest strong bisimulation, i.e.

$$\sim = \bigcup \mathcal{R}.$$

Bisimulation

In classical process algebras such as CCS a bisimulation has the following form:

A relation $\mathcal{R}$ is a strong bisimulation relation if $(P, Q) \in \mathcal{R}$ implies

1. whenever $P \xrightarrow{\alpha} P'$, then there exists Q' such that $Q \xrightarrow{\alpha} Q'$, and $(P', Q') \in \mathcal{R}$;
2. whenever $Q \xrightarrow{\alpha} Q'$, then there exists P' such that $P \xrightarrow{\alpha} P'$, and $(P', Q') \in \mathcal{R}$.

Strong bisimulation $\sim$ is the largest strong bisimulation, i.e.

$$\sim = \bigcup \mathcal{R}.$$

Bisimulation

In classical process algebras such as CCS a bisimulation has the following form:

A relation $\mathcal{R}$ is a strong bisimulation relation if $(P, Q) \in \mathcal{R}$ implies

1. whenever $P \xrightarrow{\alpha} P'$, then there exists Q' such that $Q \xrightarrow{\alpha} Q'$, and $(P', Q') \in \mathcal{R}$;
2. whenever $Q \xrightarrow{\alpha} Q'$, then there exists P' such that $P \xrightarrow{\alpha} P'$, and $(P', Q') \in \mathcal{R}$.

Strong bisimulation $\sim$ is the largest strong bisimulation, i.e.

$$\sim = \bigcup \mathcal{R}.$$

Bisimulation

In classical process algebras such as CCS a bisimulation has the following form:

A relation $\mathcal{R}$ is a strong bisimulation relation if $(P, Q) \in \mathcal{R}$ implies

1. whenever $P \xrightarrow{\alpha} P'$, then there exists Q' such that $Q \xrightarrow{\alpha} Q'$, and $(P', Q') \in \mathcal{R}$;
2. whenever $Q \xrightarrow{\alpha} Q'$, then there exists P' such that $P \xrightarrow{\alpha} P'$, and $(P', Q') \in \mathcal{R}$.

Strong bisimulation $\sim$ is the largest strong bisimulation, i.e.

$$\sim = \bigcup \mathcal{R}.$$

Bisimulation

In classical process algebras such as CCS a bisimulation has the following form:

A relation $\mathcal{R}$ is a strong bisimulation relation if $(P, Q) \in \mathcal{R}$ implies

1. whenever $P \xrightarrow{\alpha} P'$, then there exists Q' such that $Q \xrightarrow{\alpha} Q'$, and $(P', Q') \in \mathcal{R}$;
2. whenever $Q \xrightarrow{\alpha} Q'$, then there exists P' such that $P \xrightarrow{\alpha} P'$, and $(P', Q') \in \mathcal{R}$.

Strong bisimulation $\sim$ is the largest strong bisimulation, i.e.

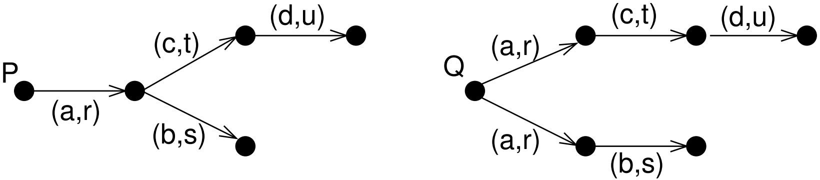
$$\sim = \bigcup \mathcal{R}.$$

Strong Equivalence in PEPA

Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.

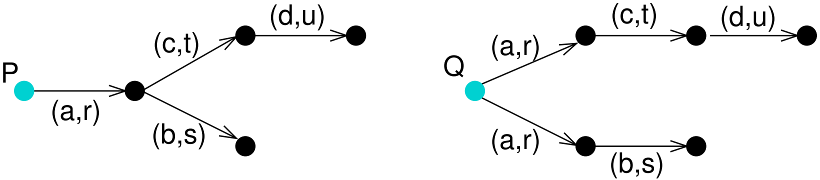
Strong Equivalence in PEPA

Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.



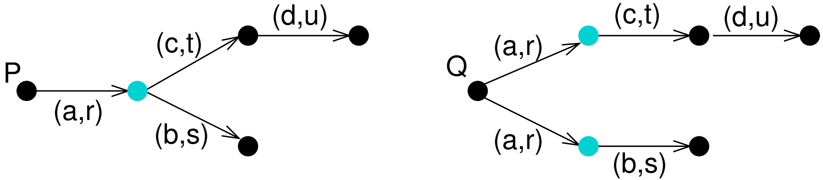
Strong Equivalence in PEPA

Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.



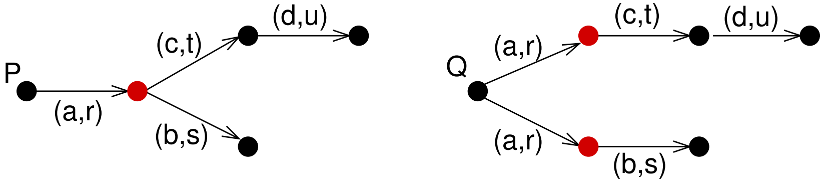
Strong Equivalence in PEPA

Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.



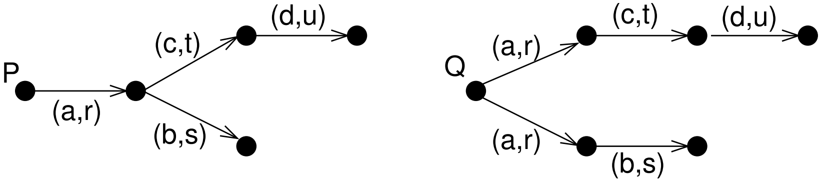
Strong Equivalence in PEPA

Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.



Strong Equivalence in PEPA

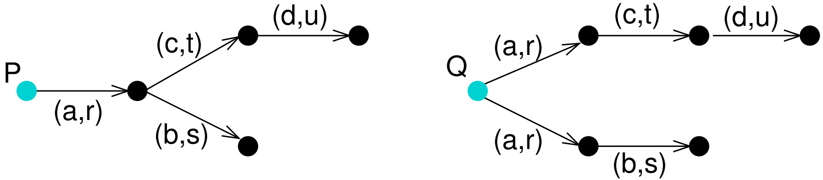
Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.



Observability is assumed to include the ability to record timing information over a number of runs.

Strong Equivalence in PEPA

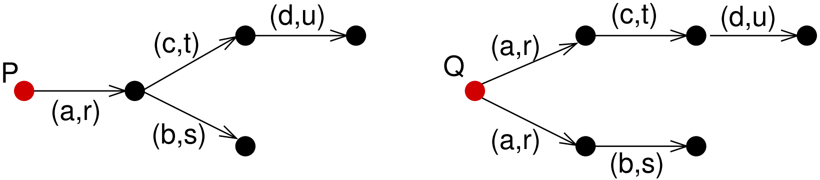
Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.



Observability is assumed to include the ability to record timing information over a number of runs.

Strong Equivalence in PEPA

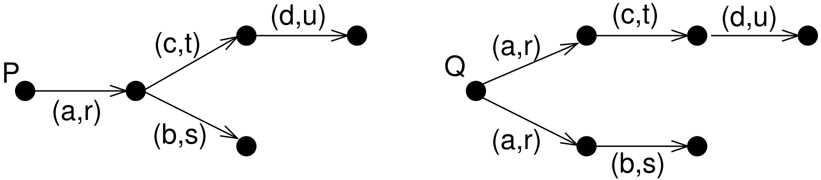
Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.



Observability is assumed to include the ability to record timing information over a number of runs.

Strong Equivalence in PEPA

Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.

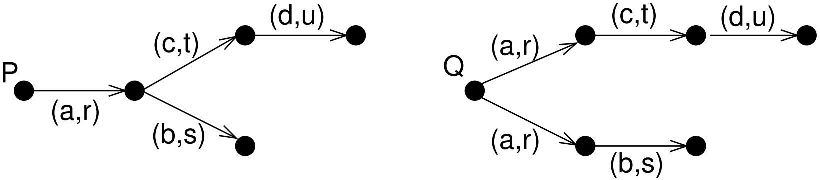


Observability is assumed to include the ability to record timing information over a number of runs.

Two processes are equivalent if they can undertake the same actions, at the same rate, and arrive at processes that are equivalent.

Strong Equivalence in PEPA

Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.



Observability is assumed to include the ability to record timing information over a number of runs.

Two processes are equivalent if they can undertake the same actions, at the same rate, and arrive at processes that are equivalent.

Expressed as rates to equivalence classes of processes

Strong Equivalence in PEPA (Markovian Bisimulation)

Definition

An equivalence relation $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$ is a *strong equivalence* if whenever $(P, Q) \in \mathcal{R}$ then for all $\alpha \in \mathcal{A}$ and for all $S \in \mathcal{C}/\mathcal{R}$

$$q[P, S, \alpha] = q[Q, S, \alpha].$$

where

$$q[C_i, S, \alpha] = \sum_{C_j \in S} q(C_i, C_j, \alpha)$$

Strong equivalence $\sim$ is $\sim = \bigcup \mathcal{R}$.

Strong Equivalence in PEPA (Markovian Bisimulation)

Definition

An equivalence relation $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$ is a *strong equivalence* if whenever $(P, Q) \in \mathcal{R}$ then for all $\alpha \in \mathcal{A}$ and for all $S \in \mathcal{C}/\mathcal{R}$

$$q[P, S, \alpha] = q[Q, S, \alpha].$$

where

$$q[C_i, S, \alpha] = \sum_{C_j \in S} q(C_i, C_j, \alpha)$$

Strong equivalence $\sim$ is $\sim = \bigcup \mathcal{R}$.

Axiomatization

$$
\begin{array}{rcl}
P_1 + P_2 & \sim & P_2 + P_1 \\
(P_1 + P_2) + P_3 & \sim & P_1 + (P_2 + P_3) \\
(\alpha, r_1).P + (\alpha, r_2).P & \sim & (\alpha, r_1 + r_2).P \\
P_1 \underset{L}{\bowtie} P_2 & \sim & P_2 \underset{L}{\bowtie} P_1 \\
(\alpha, r_1).P_1 \underset{L}{\bowtie} (\alpha, r_2).P_2 & \sim & \left\{ \begin{array}{ll} (\alpha, r_1).(P_1 \underset{L}{\bowtie} (\alpha, r_2).P_2) + & \\ \quad (\alpha, r_2).((\alpha, r_1).P_1 \underset{L}{\bowtie} P_2) & \text{if } \alpha \notin L \\ (\alpha, r_1).(P_1 \underset{L}{\bowtie} P_2) & \text{if } \alpha \in L \end{array} \right. \\
(P_1 + P_2)/L & \sim & P_1/L + P_2/L \\
((\alpha, r).P)/L & \sim & \left\{ \begin{array}{ll} (\alpha, r).(P/L) & \text{if } \alpha \notin L \\ (\tau, r).(P/L) & \text{if } \alpha \in L \end{array} \right.
\end{array}
$$

Note that there is no longer the idempotency law from classical process algebras, $P + P \sim P$.

Axiomatization

$$
\begin{array}{rcl}
P_1 + P_2 & \sim & P_2 + P_1 \\
(P_1 + P_2) + P_3 & \sim & P_1 + (P_2 + P_3) \\
(\alpha, r_1).P + (\alpha, r_2).P & \sim & (\alpha, r_1 + r_2).P \\
P_1 \underset{L}{\bowtie} P_2 & \sim & P_2 \underset{L}{\bowtie} P_1 \\
(\alpha, r_1).P_1 \underset{L}{\bowtie} (\alpha, r_2).P_2 & \sim & \left\{ \begin{array}{ll} (\alpha, r_1).(P_1 \underset{L}{\bowtie} (\alpha, r_2).P_2) + & \\ \quad (\alpha, r_2).((\alpha, r_1).P_1 \underset{L}{\bowtie} P_2) & \text{if } \alpha \notin L \\ (\alpha, r_1).(P_1 \underset{L}{\bowtie} P_2) & \text{if } \alpha \in L \end{array} \right. \\
(P_1 + P_2)/L & \sim & P_1/L + P_2/L \\
((\alpha, r).P)/L & \sim & \left\{ \begin{array}{ll} (\alpha, r).(P/L) & \text{if } \alpha \notin L \\ (\tau, r).(P/L) & \text{if } \alpha \in L \end{array} \right.
\end{array}
$$

Note that there is no longer the idempotency law from classical process algebras, $P + P \sim P$.

Strong Equivalence and Lumpability

- Given this definition it is fairly straightforward to show that if we consider strong equivalence of states within a single model, it induces an ordinarily lumpable partition on the state space of the underlying Markov chain.
- Moreover it can be shown that strong equivalence is a congruence.
- This means that aggregation based on lumpability can be applied component by component, avoiding the previous problem of having to construct the complete state space in order to find the lumpable partitions.

Strong Equivalence and Lumpability

- Given this definition it is fairly straightforward to show that if we consider strong equivalence of states within a single model, it induces an ordinarily lumpable partition on the state space of the underlying Markov chain.
- Moreover it can be shown that strong equivalence is a congruence.
- This means that aggregation based on lumpability can be applied component by component, avoiding the previous problem of having to construct the complete state space in order to find the lumpable partitions.

Strong Equivalence and Lumpability

- Given this definition it is fairly straightforward to show that if we consider strong equivalence of states within a single model, it induces an ordinarily lumpable partition on the state space of the underlying Markov chain.

- Moreover it can be shown that strong equivalence is a congruence.

- This means that aggregation based on lumpability can be applied component by component, avoiding the previous problem of having to construct the complete state space in order to find the lumpable partitions.

A logical foundation for the specification language

The expression, and testing for satisfaction of equilibrium properties, can be seen to be closely related to the specification, and model checking of a formula expressed in Larsen and Skou's probabilistic modal logic (PML). We give a modified interpretation of such formulae suitable for reasoning about PEPA's continuous time models.

We exploit the operators of modal logic to be more discriminating about which states contribute to the reward measure. In particular, we can select a state based on model behaviour which is not immediately local to the state.

A logical foundation for the specification language

The expression, and testing for satisfaction of equilibrium properties, can be seen to be closely related to the specification, and model checking of a formula expressed in Larsen and Skou's probabilistic modal logic (PML). We give a modified interpretation of such formulae suitable for reasoning about PEPA's continuous time models.

We exploit the operators of modal logic to be more discriminating about which states contribute to the reward measure. In particular, we can select a state based on model behaviour which is not immediately local to the state.

Larsen and Skou's PML

$$
\begin{array}{lcll}
\mathrm{F} & ::= & \mathrm{tt} & (\text{truth}) \\
 & \mid & \nabla\alpha & (\text{inability}) \\
 & \mid & \neg F & (\text{negation}) \\
 & \mid & F_1 \wedge F_2 & (\text{conjunction}) \\
 & \mid & \langle\alpha\rangle_\mu F & (\text{``at least''})
\end{array}
$$

Relation to PEPA

Defn. $P \xrightarrow{(\alpha,\nu)} S$ if for all $P' \in S$, $P \xrightarrow{\alpha} P'$ and

$$\sum\{r \mid P \xrightarrow{(\alpha,r)} P', P' \in S\} = \nu.$$

Let P be a model of a PEPA process.

$$\begin{aligned}
P \quad &\models \mathrm{tt} \\
P \quad &\models \neg F \text{ if } P \not\models F \\
P \quad &\models F_1 \wedge F_2 \text{ if } P \models F_1 \text{ and } P \models F_2 \\
P \quad &\models \nabla_\alpha \text{ if } P \overset{\alpha}{\not\rightarrow} \\
P \quad &\models \langle\alpha\rangle_\mu F \text{ if } P \xrightarrow{(\alpha,\nu)} S \text{ for some } \nu \geq \mu, \\
&\qquad \text{and for all } P' \in S, P' \models F
\end{aligned}$$

Relation to PEPA

Defn. $P \xrightarrow{(\alpha,\nu)} S$ if for all $P' \in S$, $P \xrightarrow{\alpha} P'$ and

$$\sum\{r \mid P \xrightarrow{(\alpha,r)} P', P' \in S\} = \nu.$$

Let P be a model of a PEPA process.

$P \quad \models \mathrm{tt}$

$P \quad \models \neg F$ if $P \not\models F$

$P \quad \models F_1 \wedge F_2$ if $P \models F_1$ and $P \models F_2$

$P \quad \models \nabla_\alpha$ if $P \stackrel{\alpha}{\not\longrightarrow}$

$P \quad \models \langle\alpha\rangle_\mu F$ if $P \xrightarrow{(\alpha,\nu)} S$ for some $\nu \geq \mu$,
and for all $P' \in S, P' \models F$

Relation to PEPA

Defn. $P \xrightarrow{(\alpha,\nu)} S$ if for all $P' \in S$, $P \xrightarrow{\alpha} P'$ and
$$\sum\{r \mid P \xrightarrow{(\alpha,r)} P', P' \in S\} = \nu.$$

Let P be a model of a PEPA process.

$P \models \mathrm{tt}$

$P \models \neg F$ if $P \not\models F$

$P \models F_1 \wedge F_2$ if $P \models F_1$ and $P \models F_2$

$P \models \nabla_\alpha$ if $P \overset{\alpha}{\not\longrightarrow}$

$P \models \langle\alpha\rangle_\mu F$ if $P \xrightarrow{(\alpha,\nu)} S$ for some $\nu \geq \mu$,
and for all $P' \in S$, $P' \models F$

Relation to PEPA

Defn. $P \xrightarrow{(\alpha,\nu)} S$ if for all $P' \in S$, $P \xrightarrow{\alpha} P'$ and

$$\sum\{r \mid P \xrightarrow{(\alpha,r)} P', P' \in S\} = \nu.$$

Let P be a model of a PEPA process.

$P \models \mathrm{tt}$

$P \models \neg F$ if $P \not\models F$

$P \models F_1 \wedge F_2$ if $P \models F_1$ and $P \models F_2$

$P \models \nabla_\alpha$ if $P \overset{\alpha}{\not\longrightarrow}$

$P \models \langle\alpha\rangle_\mu F$ if $P \xrightarrow{(\alpha,\nu)} S$ for some $\nu \geq \mu$,
and for all $P' \in S, P' \models F$

Relation to PEPA

Defn. $P \xrightarrow{(\alpha,\nu)} S$ if for all $P' \in S$, $P \xrightarrow{\alpha} P'$ and

$$\sum\{r \mid P \xrightarrow{(\alpha,r)} P', P' \in S\} = \nu.$$

Let P be a model of a PEPA process.

$P \quad \models \mathrm{tt}$

$P \quad \models \neg F$ if $P \not\models F$

$P \quad \models F_1 \wedge F_2$ if $P \models F_1$ and $P \models F_2$

$P \quad \models \nabla_\alpha$ if $P \overset{\alpha}{\not\rightarrow}$

$P \quad \models \langle\alpha\rangle_\mu F$ if $P \xrightarrow{(\alpha,\nu)} S$ for some $\nu \geq \mu$,
and for all $P' \in S, P' \models F$

Relation to PEPA

Defn. $P \xrightarrow{(\alpha,\nu)} S$ if for all $P' \in S$, $P \xrightarrow{\alpha} P'$ and
$$\sum\{r \mid P \xrightarrow{(\alpha,r)} P', P' \in S\} = \nu.$$

Let P be a model of a PEPA process.

$$P \models \mathrm{tt}$$

$$P \models \neg F \text{ if } P \not\models F$$

$$P \models F_1 \wedge F_2 \text{ if } P \models F_1 \text{ and } P \models F_2$$

$$P \models \nabla_\alpha \text{ if } P \overset{\alpha}{\not\rightarrow}$$

$$P \models \langle\alpha\rangle_\mu F \text{ if } P \xrightarrow{(\alpha,\nu)} S \text{ for some } \nu \geq \mu,$$
$$\text{and for all } P' \in S, P' \models F$$

Modal characterisation of strong equivalence

Let P be a model of a PEPA process. Then

$$P \cong Q \text{ iff for all } F, \;\; P \models F \text{ iff } Q \models F$$

i.e. two PEPA processes are strongly equivalent (in particular, their underlying Markov chains are lumpably equivalent) if and only if they both satisfy, in the setting where rates are quantified, the same set of PML formulae.

Modal characterisation of strong equivalence

Let P be a model of a PEPA process. Then

$$P \cong Q \text{ iff for all } F, \quad P \models F \text{ iff } Q \models F$$

i.e. two PEPA processes are strongly equivalent (in particular, their underlying Markov chains are lumpably equivalent) if and only if they both satisfy, in the setting where rates are quantified, the same set of PML formulae.

References

- U. Herzog, *Formal description, time and performance analysis: A framework*, Technical Report 15/90, IMMD VII, Friedrich-Alexander-Universität, Erlangen- Nürnberg, (Sept 1990)
- M. Bernardo and R. Gorrieri, *A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time*, in Theoretical Computer Science, 202(1–2), pp. 1–54, 1998.
- L. Bortolussi, *Stochastic Concurrent Constraint Programming*, in Proc. of 4th Intl. Workshop of Quantitative Aspects of Programming Languages, QAPL 2006, ENTCS 164-3, Wien, Austria, April 2006.
- N. Götz, U. Herzog, M. Rettelbach, *TIPP — a language for timed processes and performance evaluation.* Technical Report 4/92, IMMD7, University of Erlangen- Nürnberg, (Nov 1992)

References

- H. Hermanns, *Interactive Markov Chains: The Quest for Quantified Quality*, Volume 2428 of LNCS. Springer (2002)
- C. Priami, *Stochastic π-Calculus*, in The Computer Journal, 38(7), 578–589, 1995.
- R. Milner, *Communication and Concurrency*, Prentice-Hall (1989)
- C. Hoare, *Communicating Sequential Processes*, Prentice-Hall (1985)
- J. Hillston, *The Nature of Synchronisation*, in 2nd Workshop of Process Algebras and Performance Modelling Workshop, 1994.