Stochastic Modelling
Part I: Foundations

Stephen Gilmore
LFCS, University of Edinburgh

GLOBAN Summerschool
Lygby, Denmark

25th August 2006

Global Computing

What distinguishes *global* computing from *local* computing?

Global Computing: Some differences

- Physical distance
  - Network latency

Global Computing: Some differences

- Physical distance
  - Network latency
- Partial failures
  - Server may be down
  - Routers may be down

Global Computing: Some differences

- Physical distance
  - Network latency
- Partial failures
  - Server may be down
  - Routers may be down
- Scale
  - Workload characterisation

Global Computing: Some differences

- Physical distance
  - Network latency
- Partial failures
  - Server may be down
  - Routers may be down
- Scale
  - Workload characterisation
- Resource sharing
  - Network contention
  - CPU load

Global Computing: Some differences

- Physical distance — need to represent time
  - Network latency
- Partial failures
  - Server may be down
  - Routers may be down
- Scale
  - Workload characterisation
- Resource sharing
  - Network contention
  - CPU load

Global Computing: Some differences

- Physical distance — need to represent time
  - Network latency
- Partial failures — randomness and probability
  - Server may be down
  - Routers may be down
- Scale
  - Workload characterisation
- Resource sharing
  - Network contention
  - CPU load

Global Computing: Some differences

- Physical distance — need to represent time
  - Network latency
- Partial failures — randomness and probability
  - Server may be down
  - Routers may be down
- Scale — need to quantify population sizes
  - Workload characterisation
- Resource sharing
  - Network contention
  - CPU load

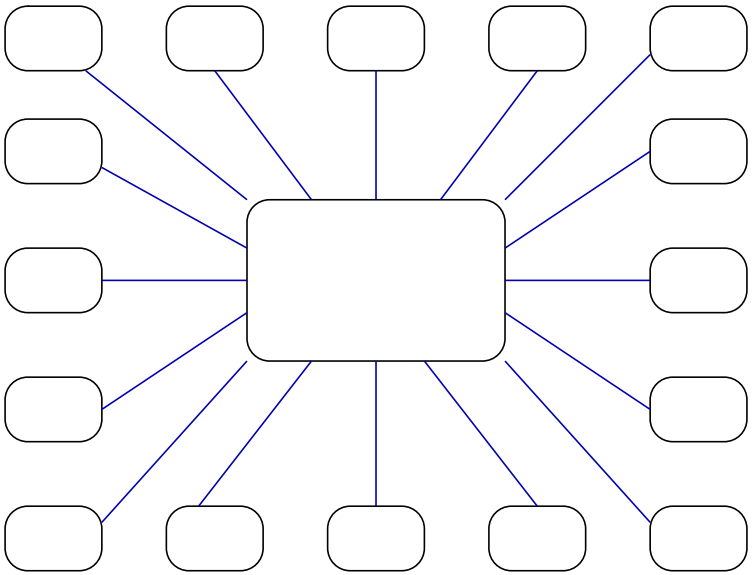
Global Computing: Some differences

- Physical distance — need to represent time
  - Network latency
- Partial failures — randomness and probability
  - Server may be down
  - Routers may be down
- Scale — need to quantify population sizes
  - Workload characterisation
- Resource sharing — need to express percentages
  - Network contention
  - CPU load

Modelling Global Computing: The challenges

Time	What representation of time will we use?
Randomness	What kind of random number distributions will we use?
Probability	How can we have probabilities in the model without uncertainty in the results?
Scale	How can we escape the state-space explosion problem?
Percentages	What can it mean to have a fraction of a process?

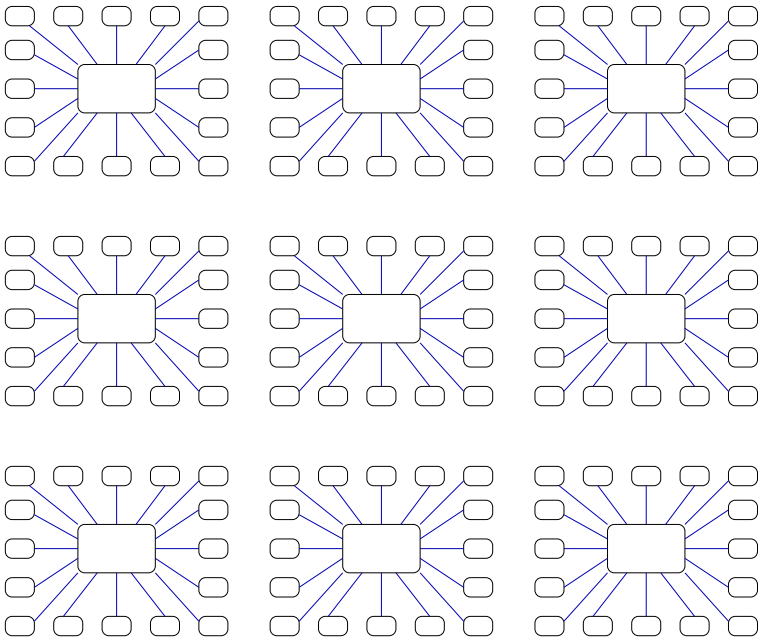
Quantitative Modelling: Motivation



Quality of Service issues

- Can the server maintain reasonable response times?

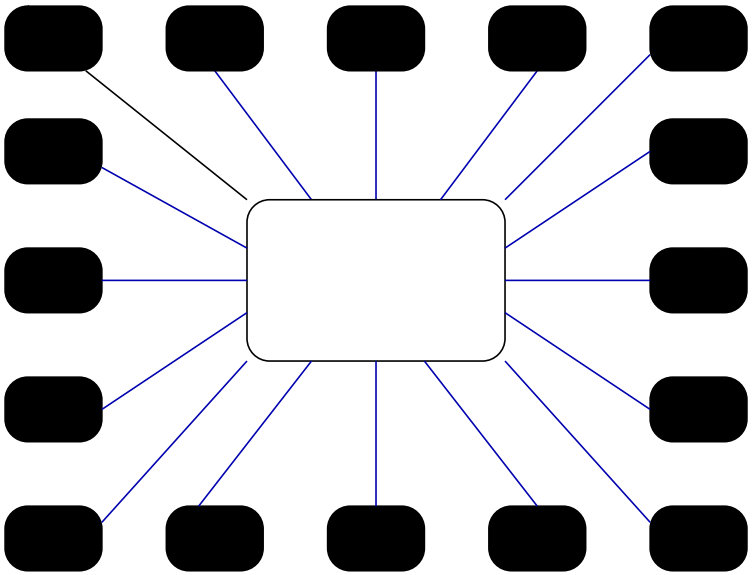
Quantitative Modelling: Motivation



Scalability issues

- How many times do we have to replicate this service to support all of the subscribers?

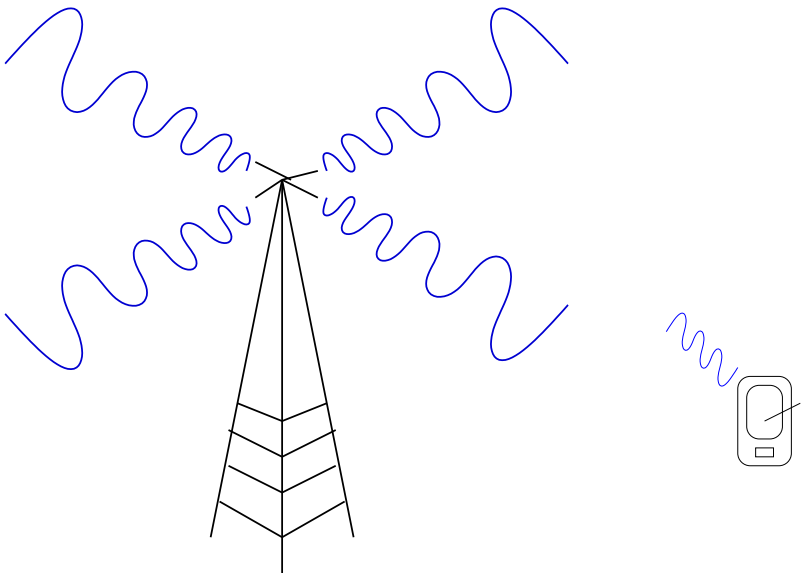
Quantitative Modelling: Motivation



Scalability issues

- Will the server withstand a distributed denial of service attack?

Quantitative Modelling: Motivation



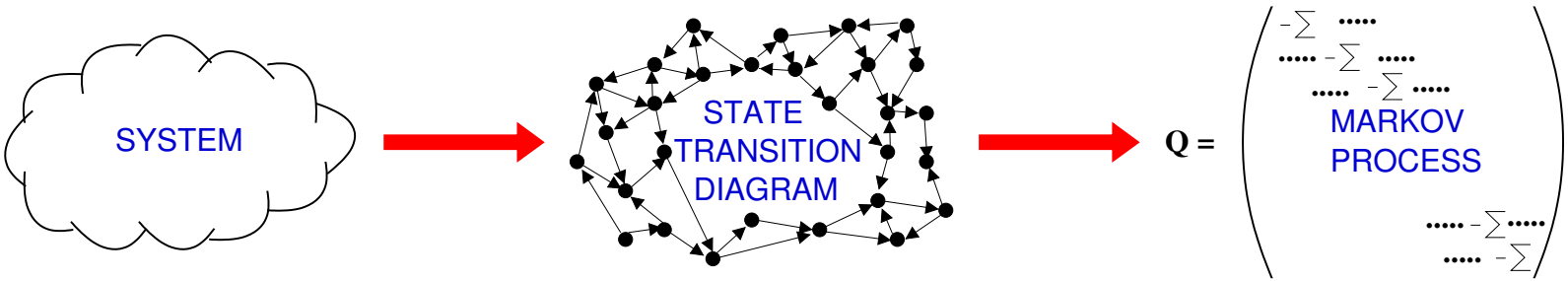
Service-level agreements

- What percentage of downloads do complete within the time we advertised?

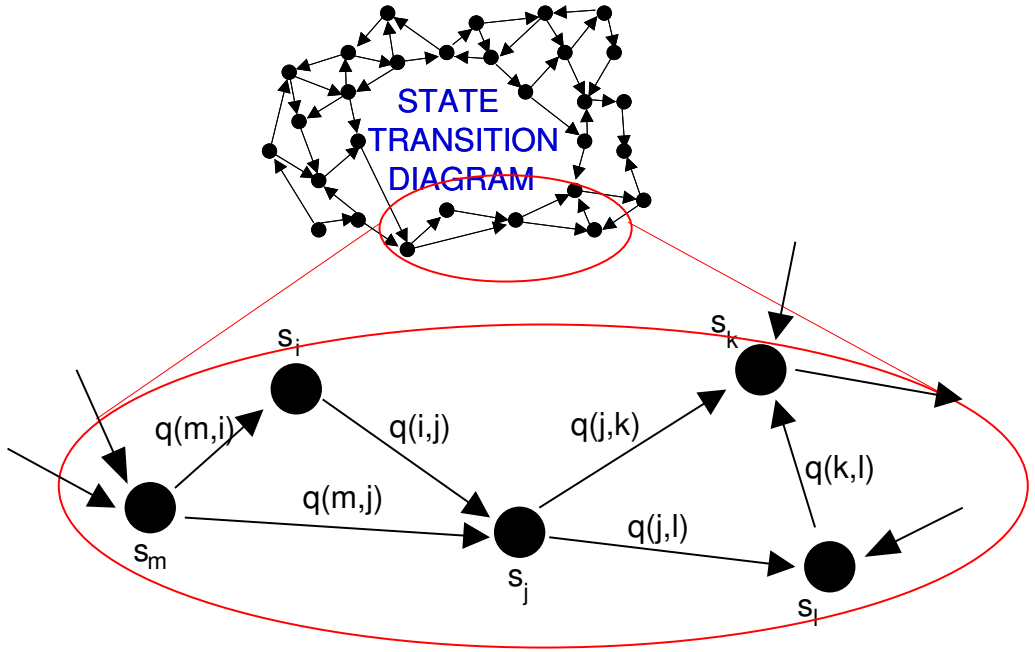
Outline

Outline

Performance Modelling using CTMC

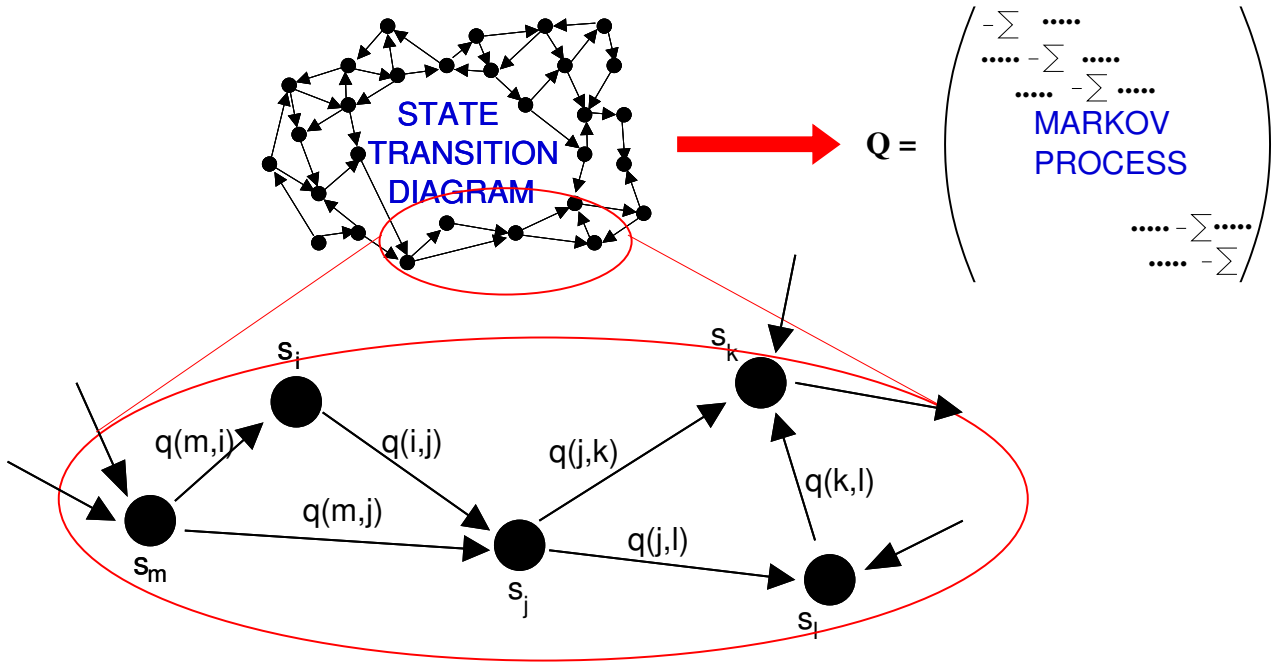


Performance Modelling using CTMC



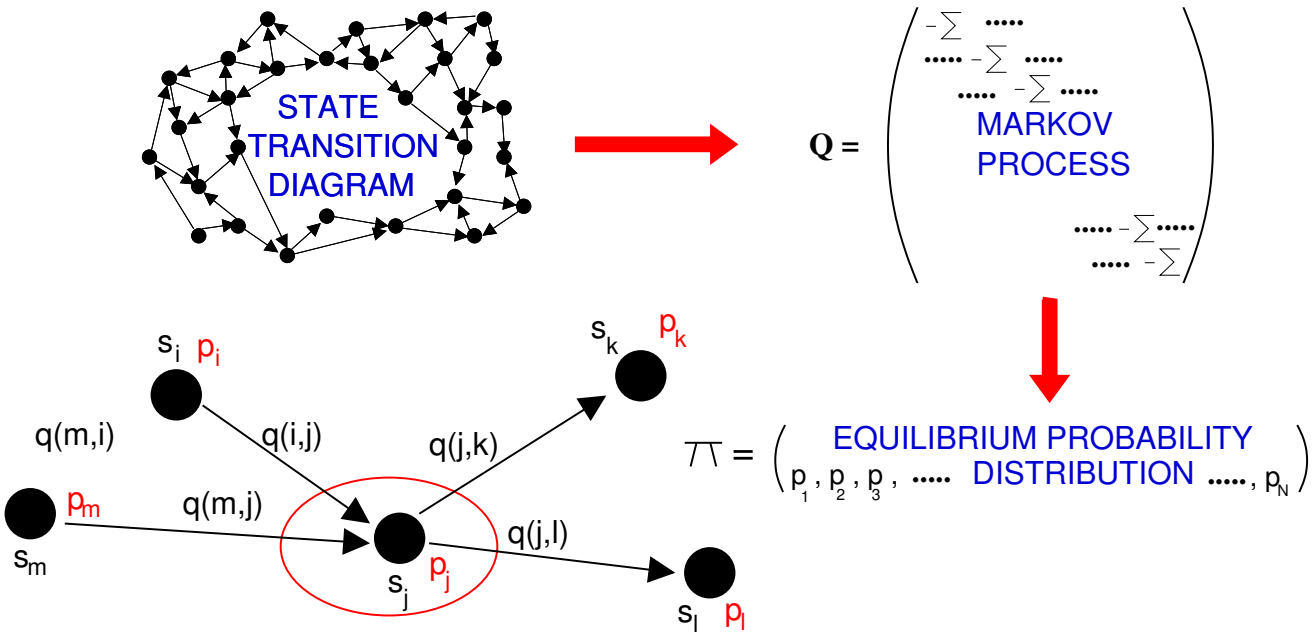
A negative exponentially distributed duration is associated with each transition.

Performance Modelling using CTMC



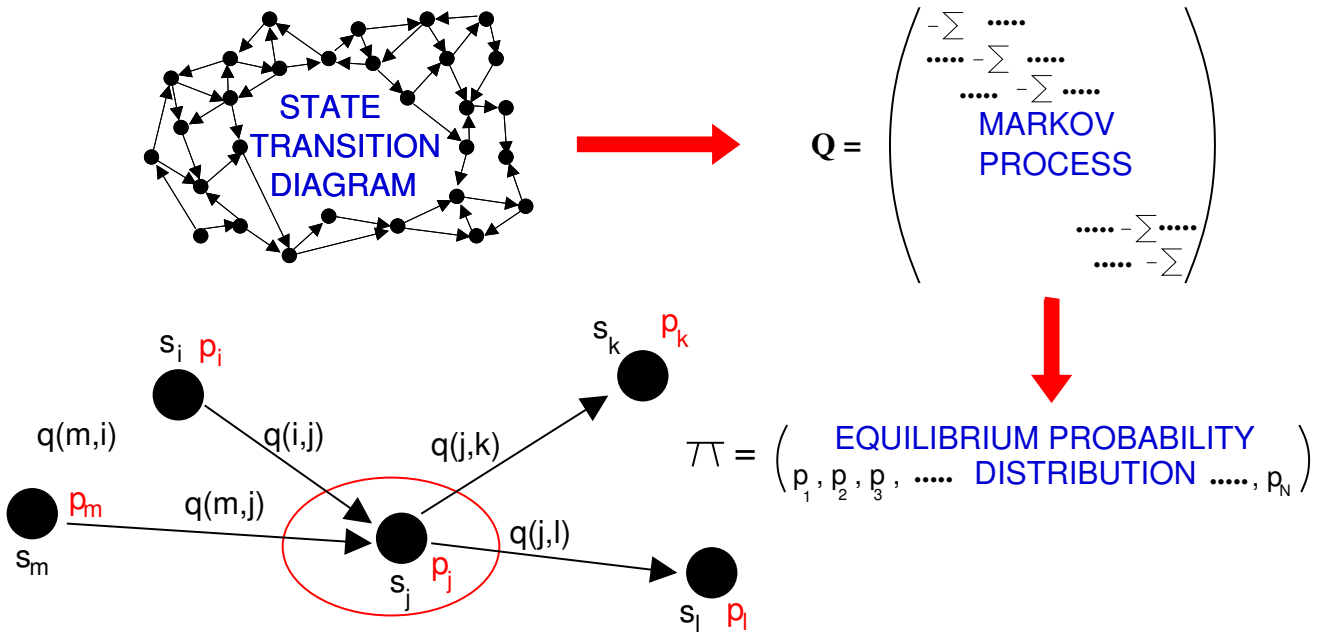
these parameters form the entries of the infinitesimal generator matrix Q

Performance Modelling using CTMC



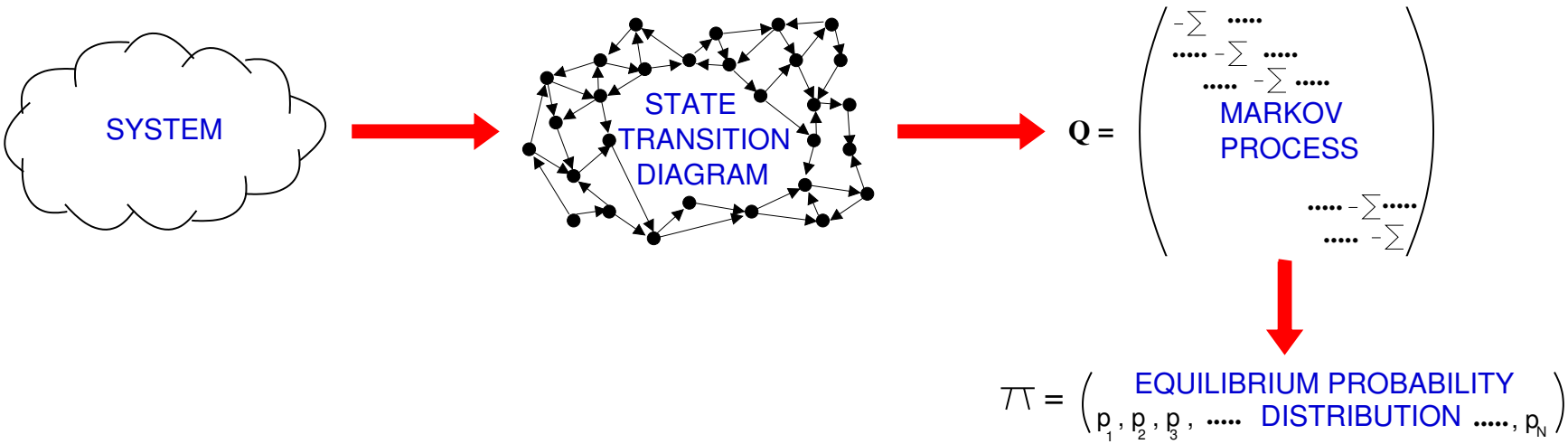
In steady state the probability flux out of a state is balanced by the flux in.

Performance Modelling using CTMC

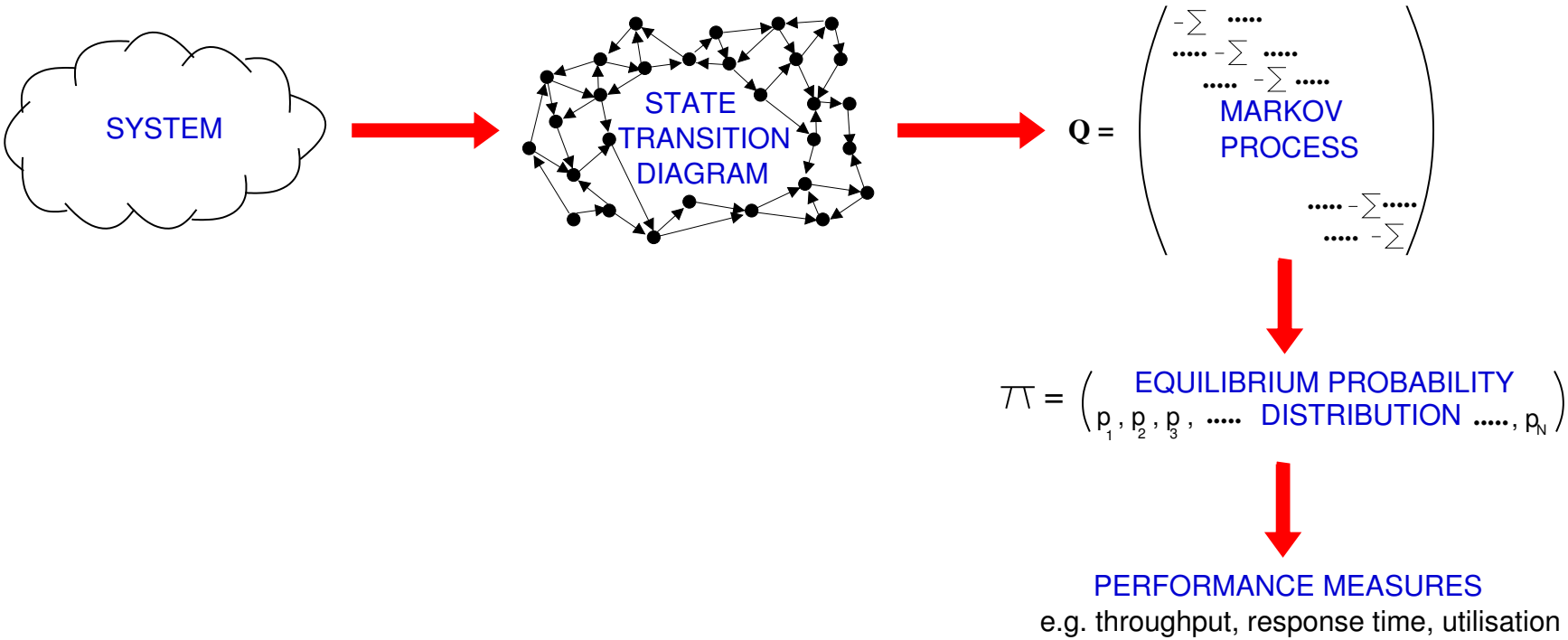


"Global balance equations" captured by $\pi Q = 0$ solved by linear algebra

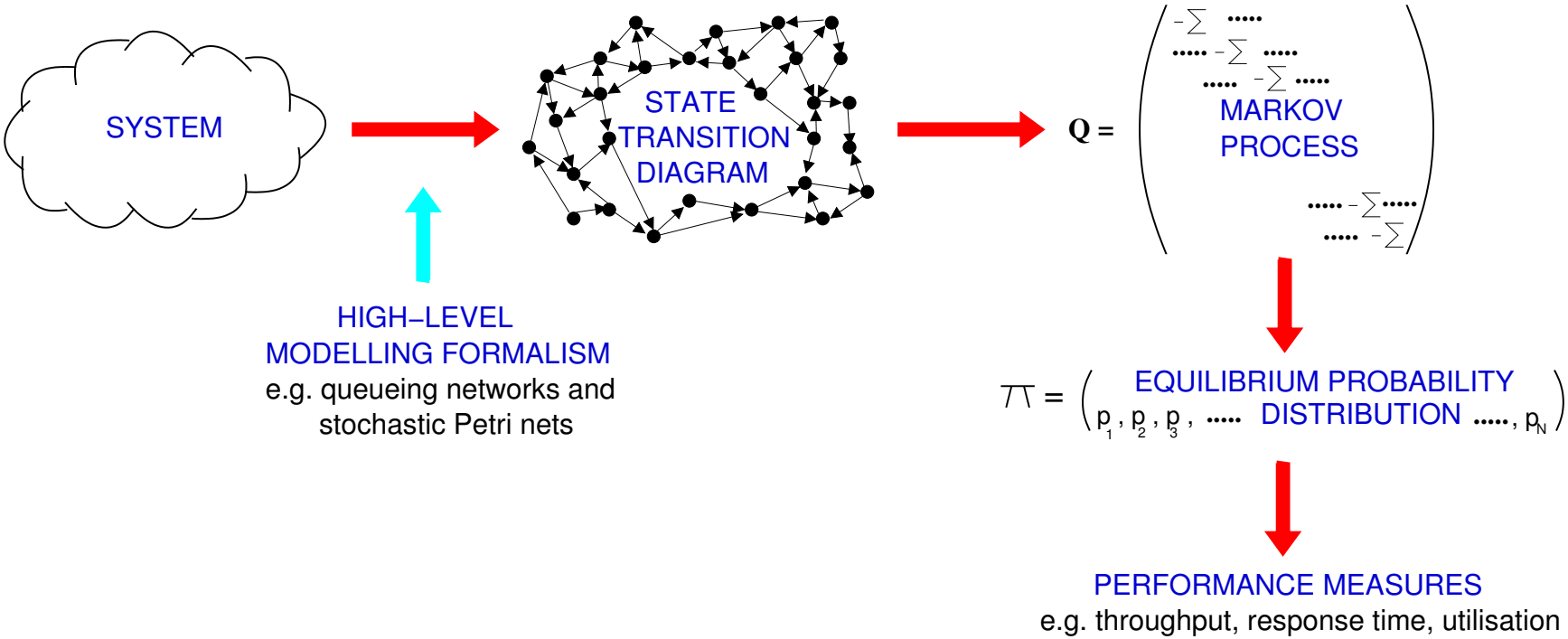
Performance Modelling using CTMC



Performance Modelling using CTMC



Performance Modelling using CTMC



Outline

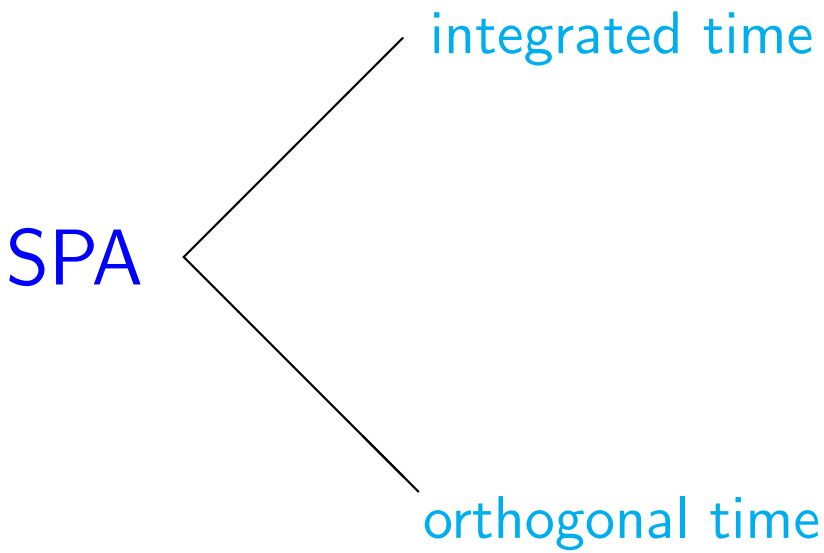
Stochastic process algebras

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

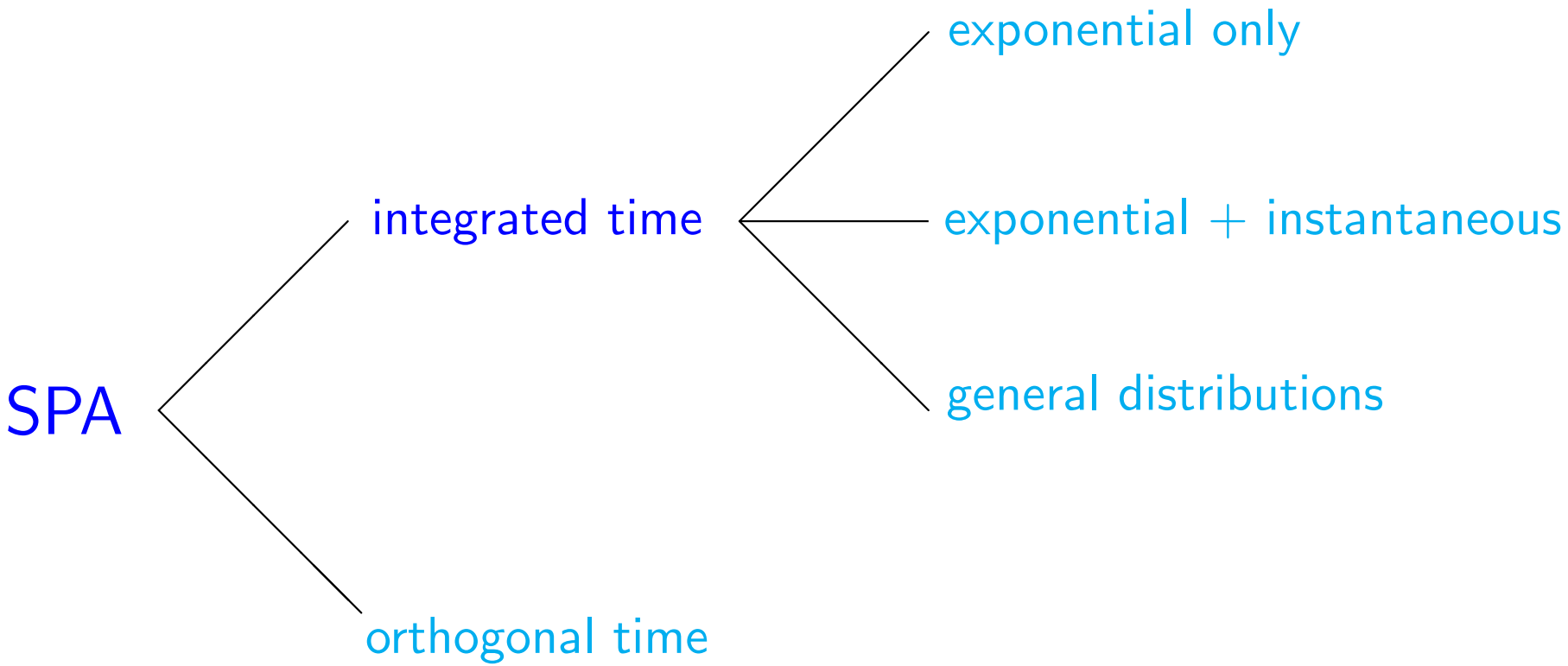
SPA Languages

SPA

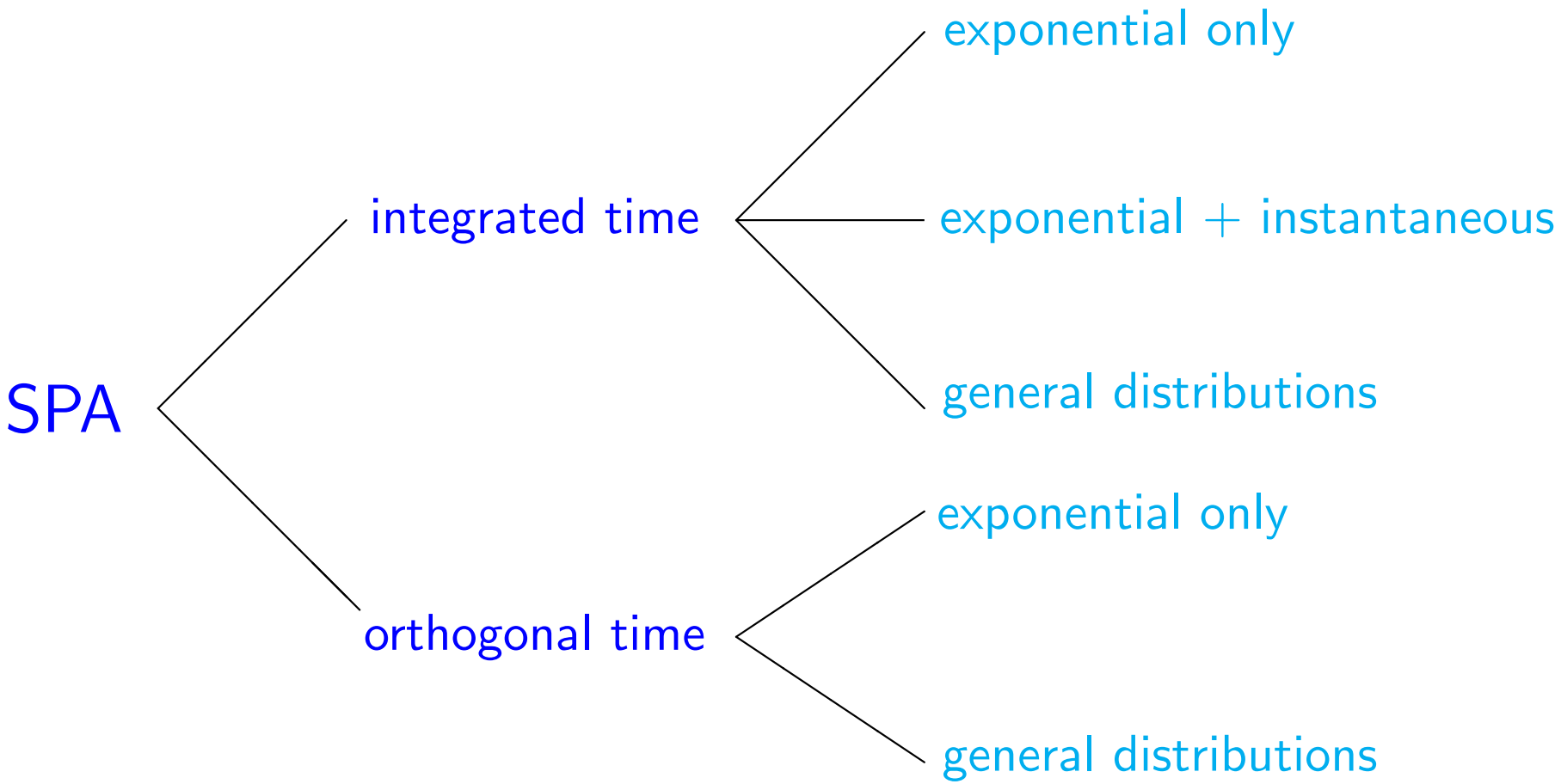
SPA Languages



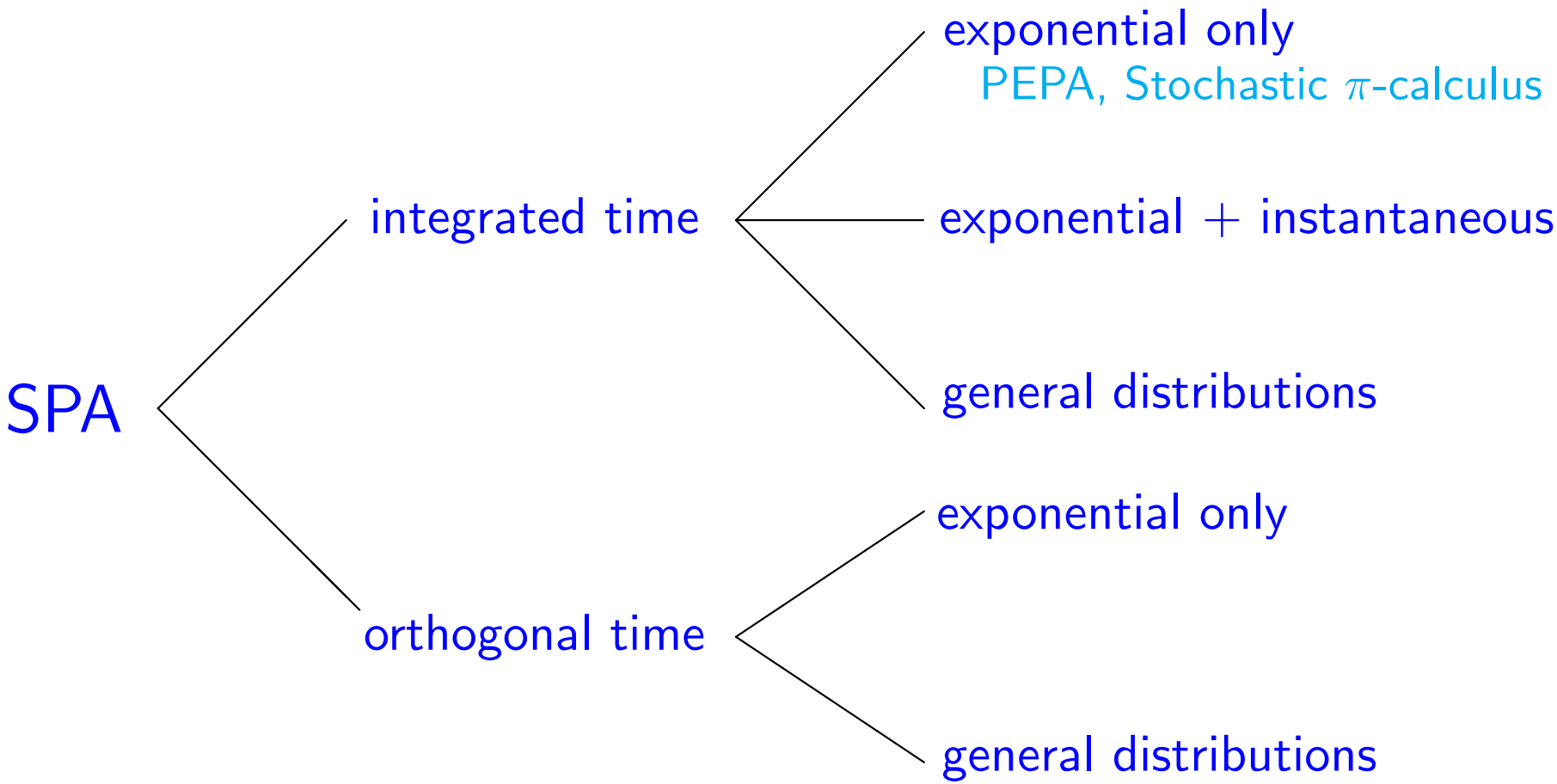
SPA Languages



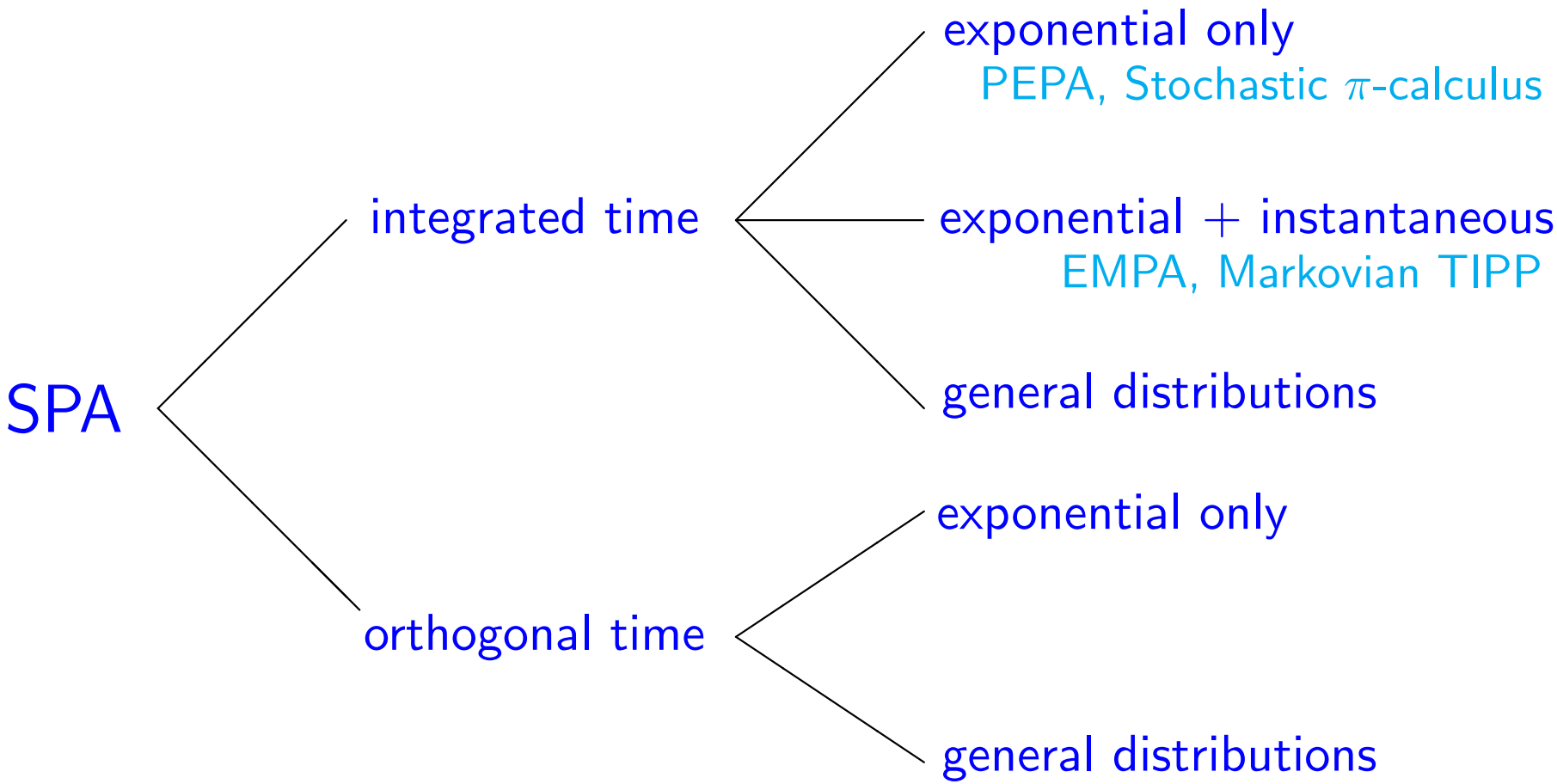
SPA Languages



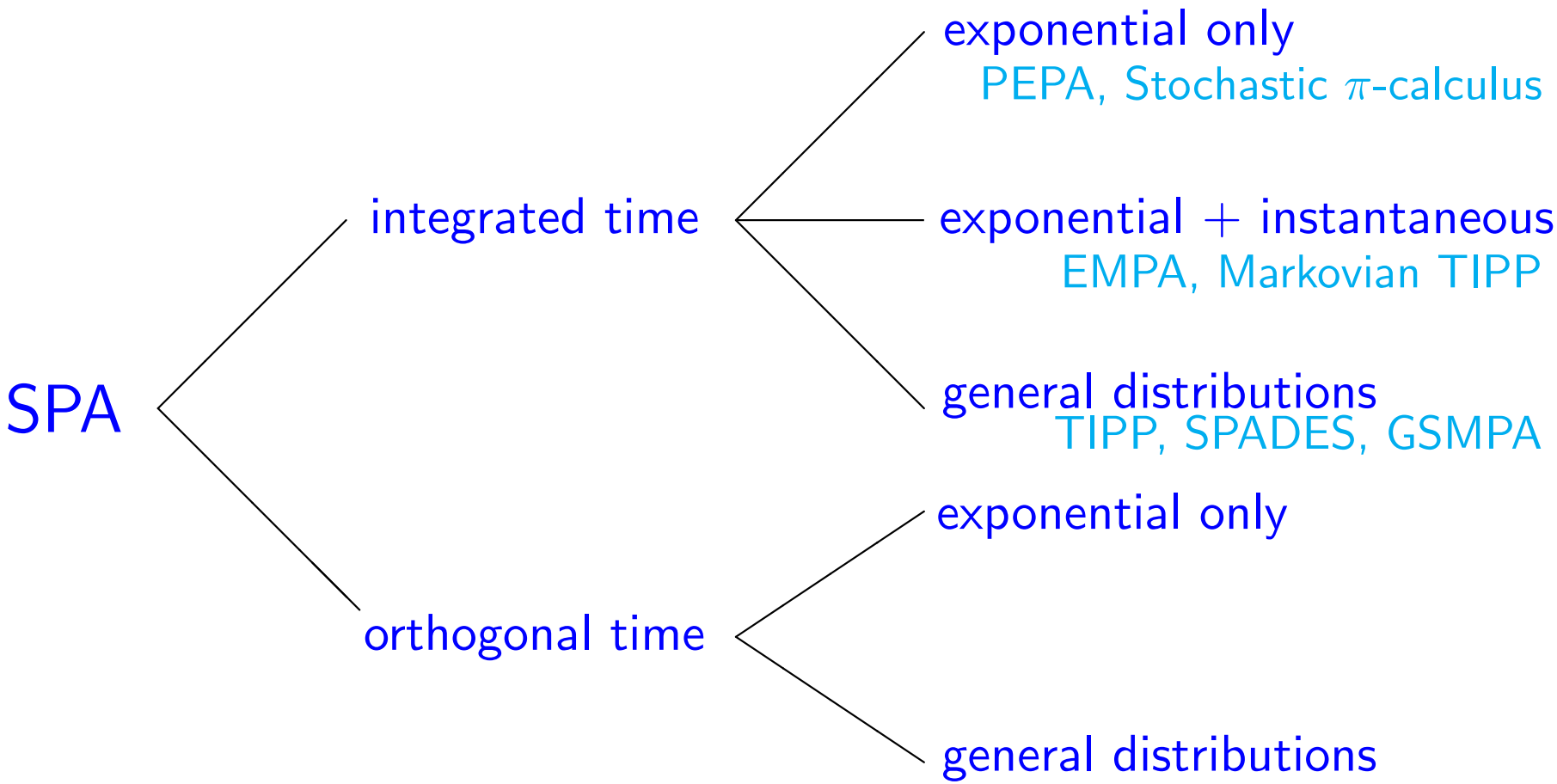
SPA Languages



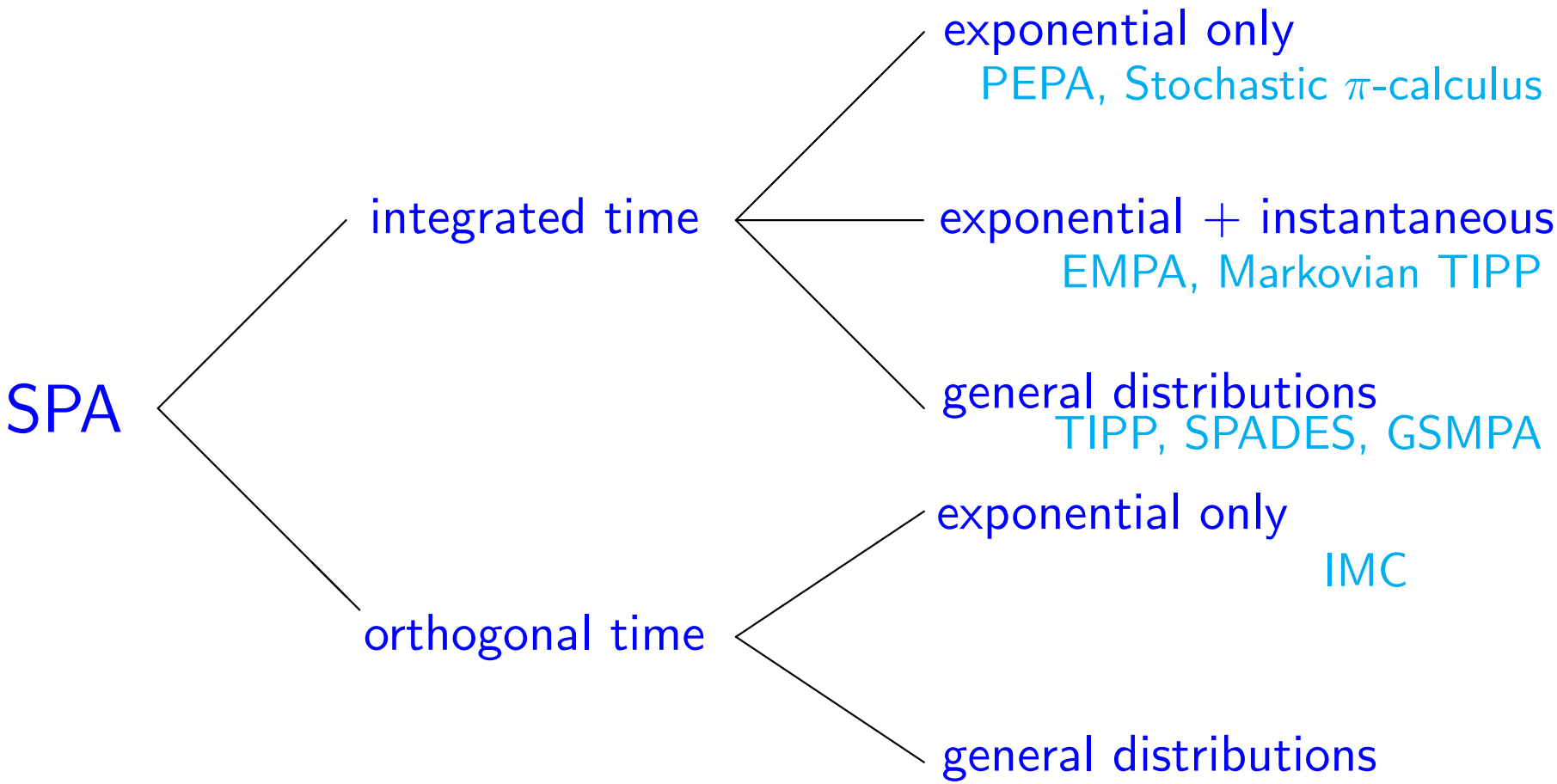
SPA Languages



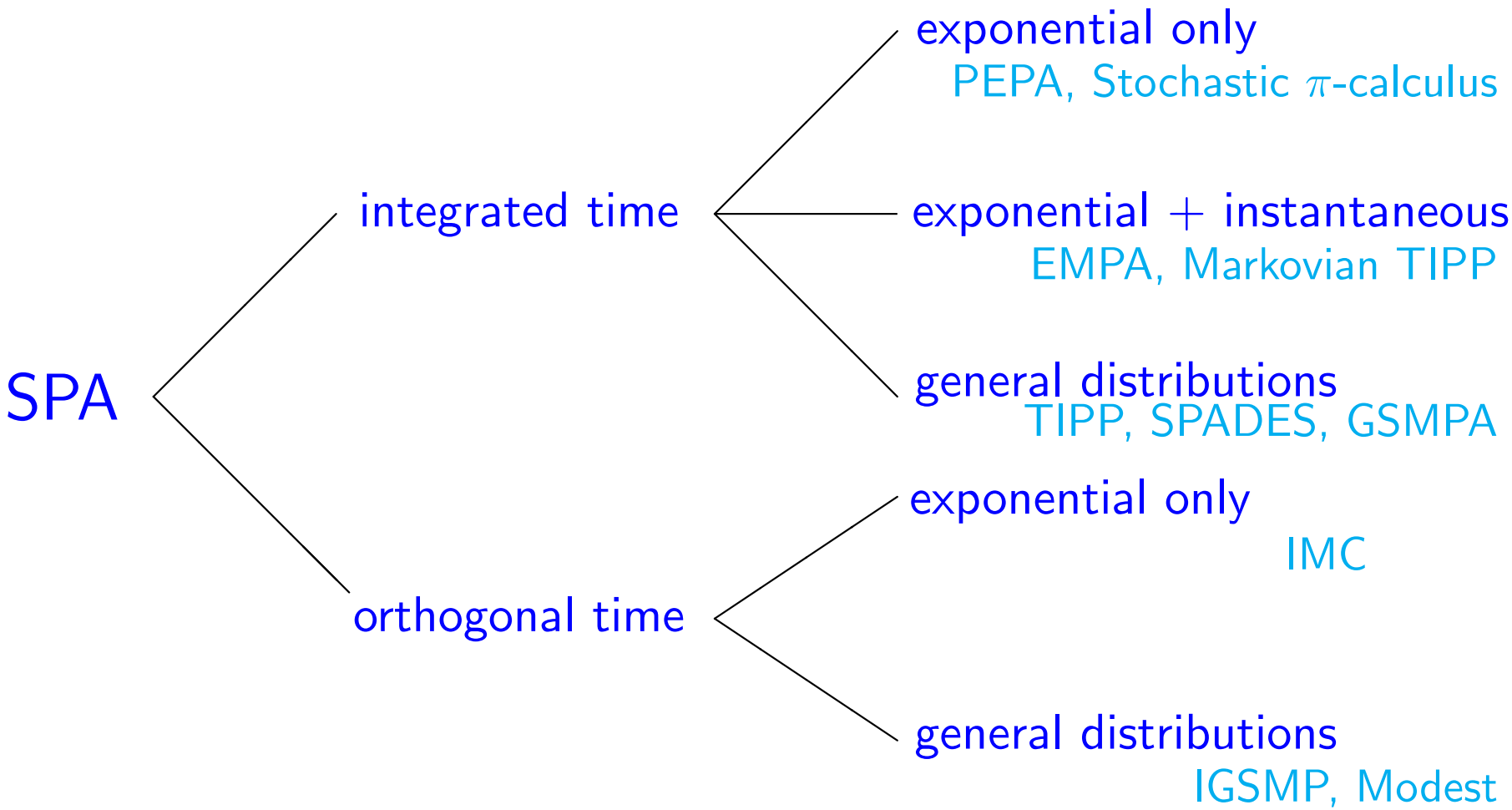
SPA Languages



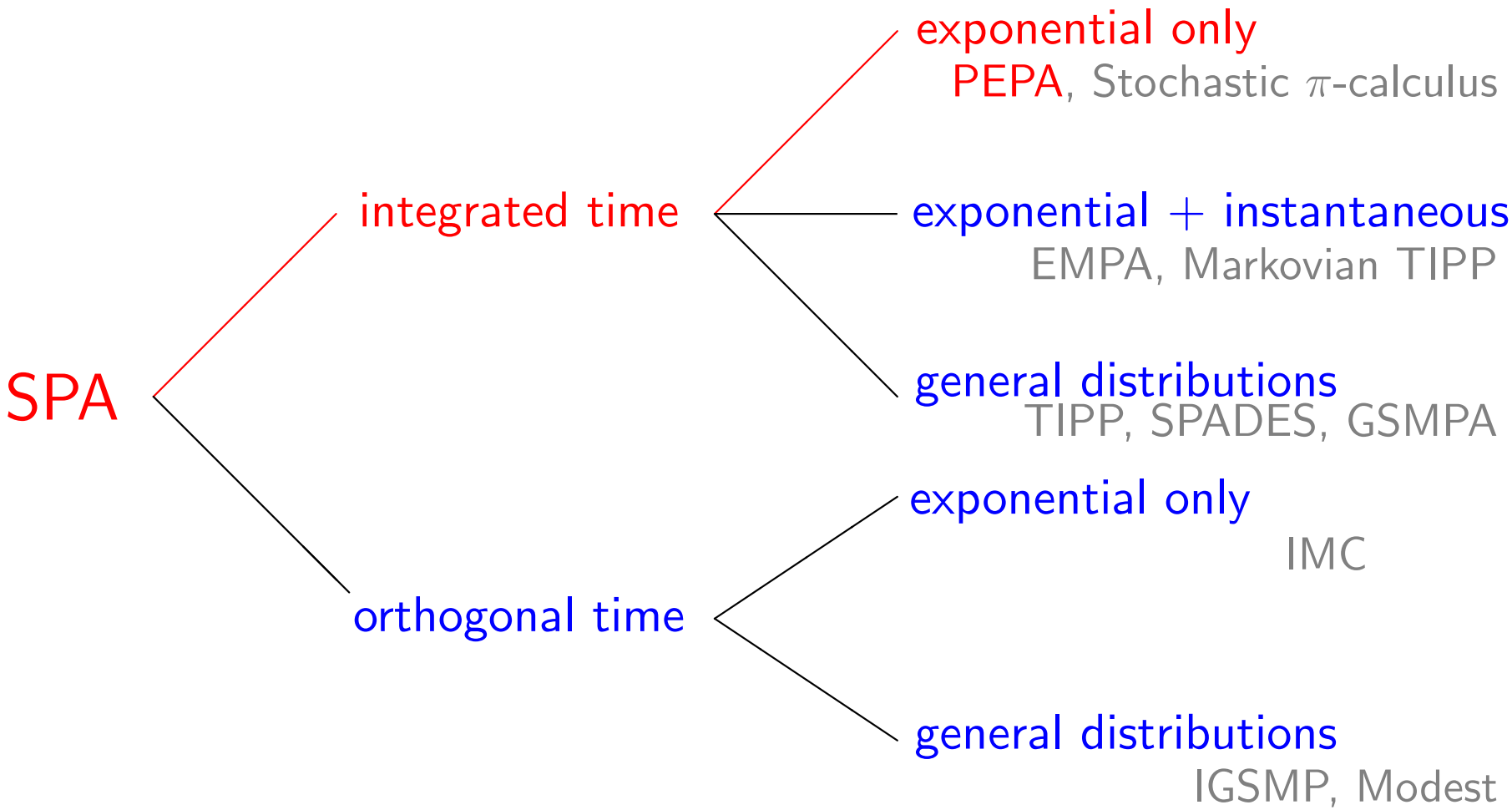
SPA Languages



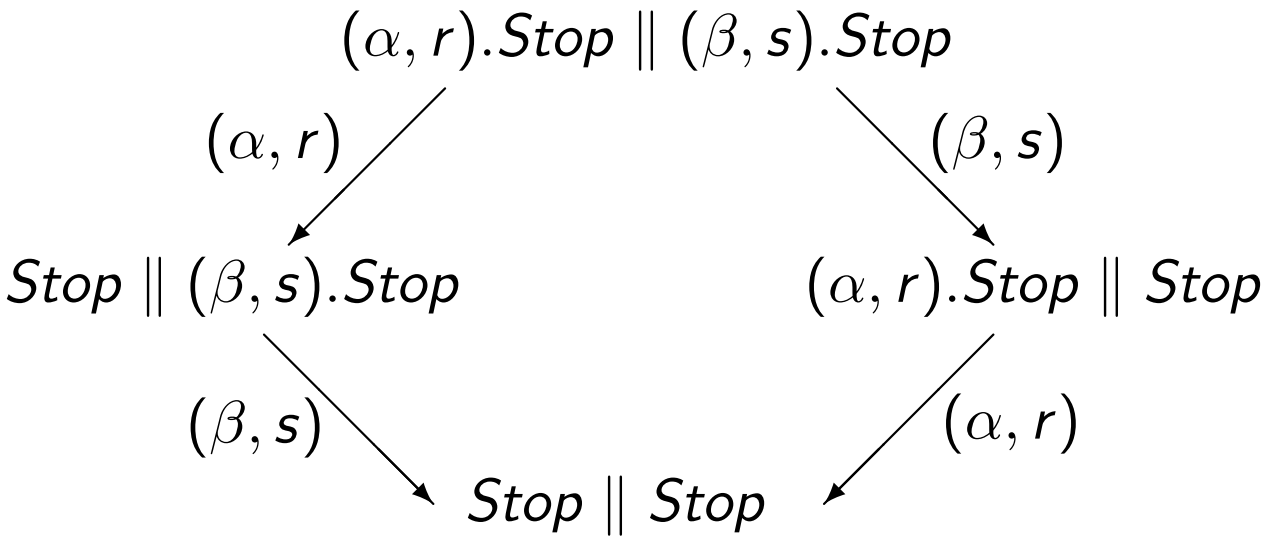
SPA Languages



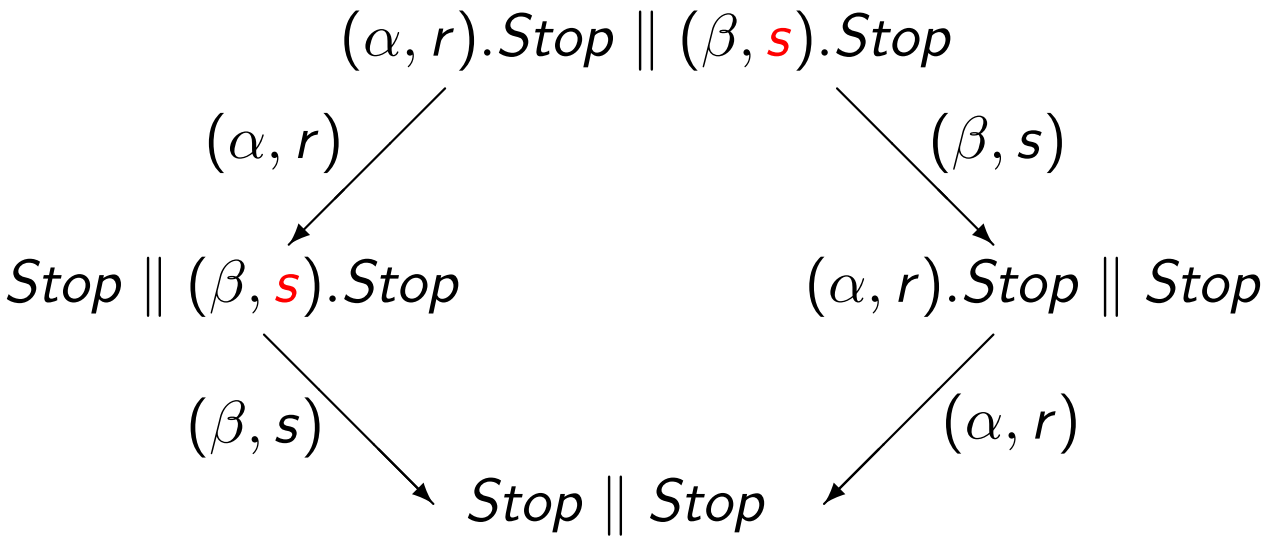
SPA Languages



The Importance of Being Exponential



The Importance of Being Exponential



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

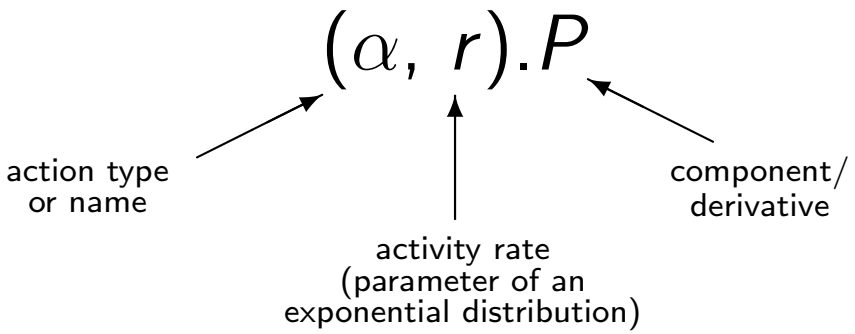
We retain the expansion law of classical process algebra:

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

only if the negative exponential distribution is used.

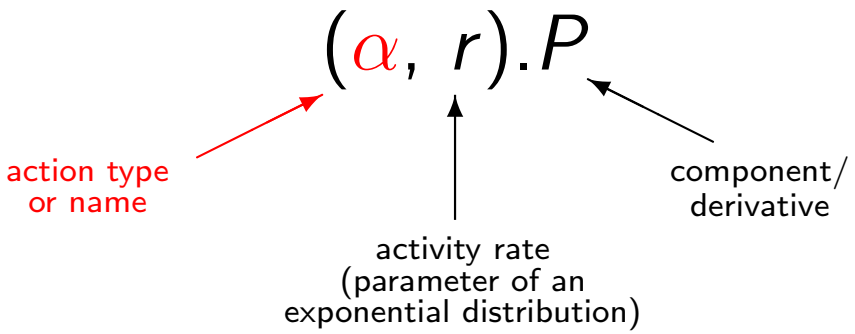
Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.



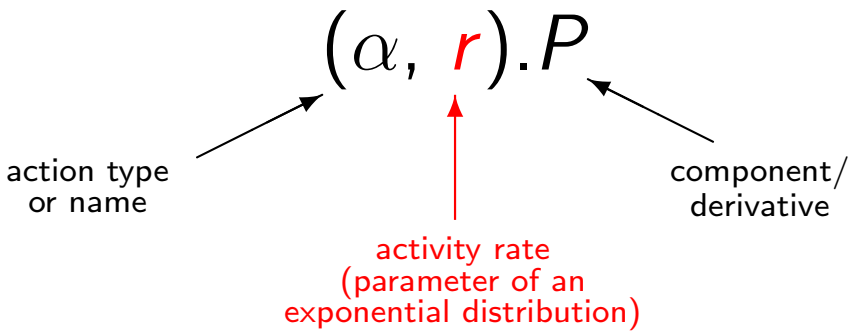
Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.



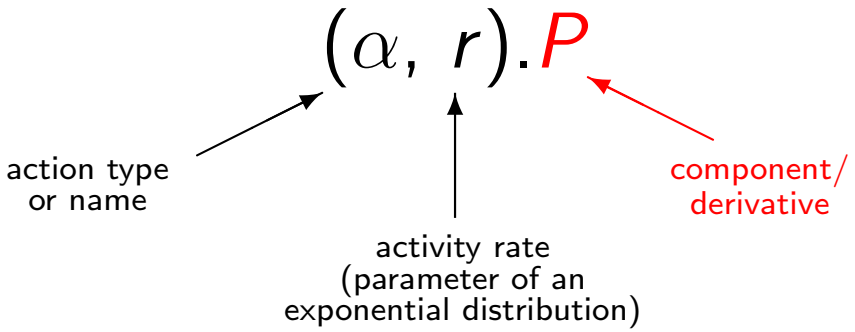
Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.



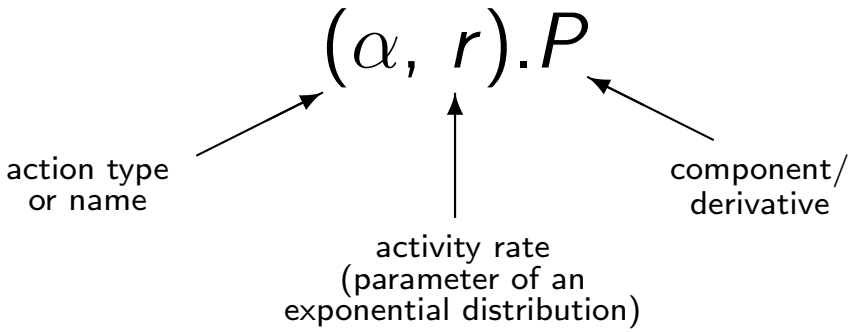
Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.



Performance Evaluation Process Algebra

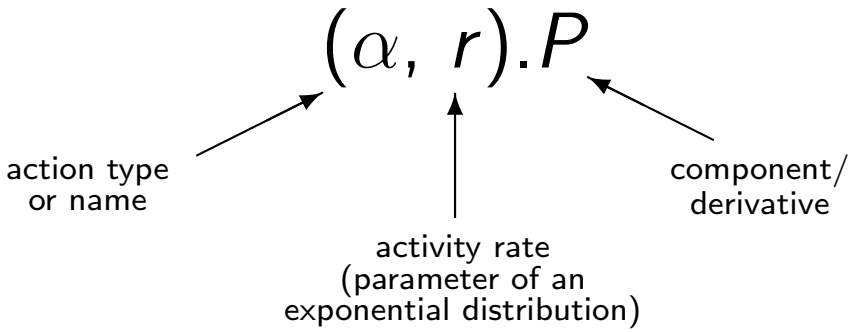
- Models are constructed from components which engage in activities.



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

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

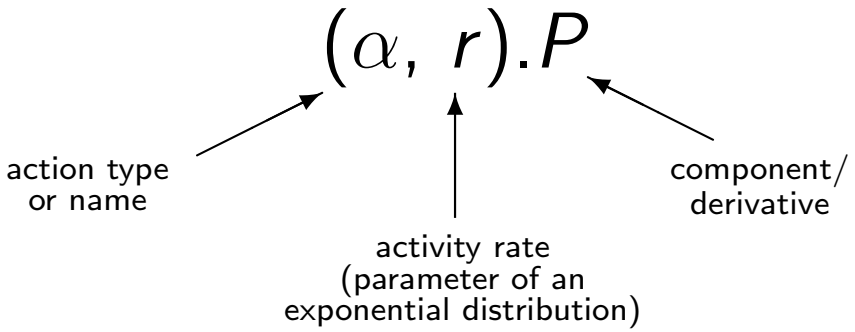


- The language is used to generate a CTMC for performance modelling.

PEPA
MODEL

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

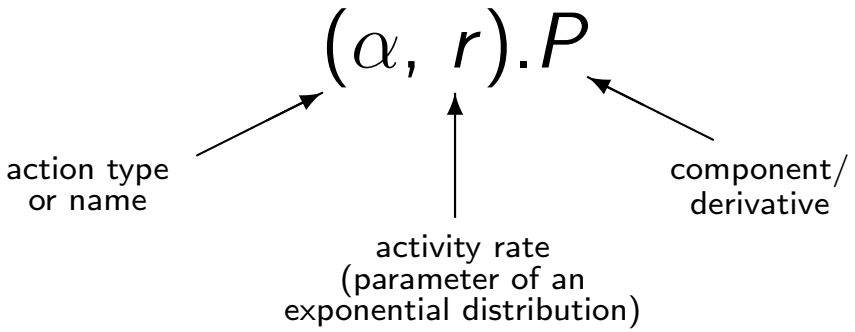


- The language is used to generate a CTMC for performance modelling.

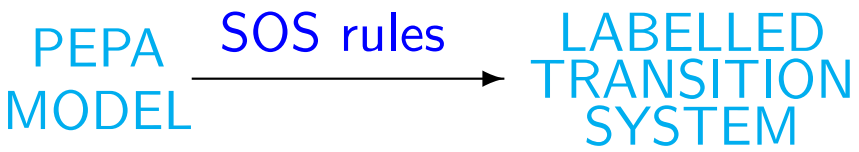
PEPA MODEL $\xrightarrow{\text{SOS rules}}$

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

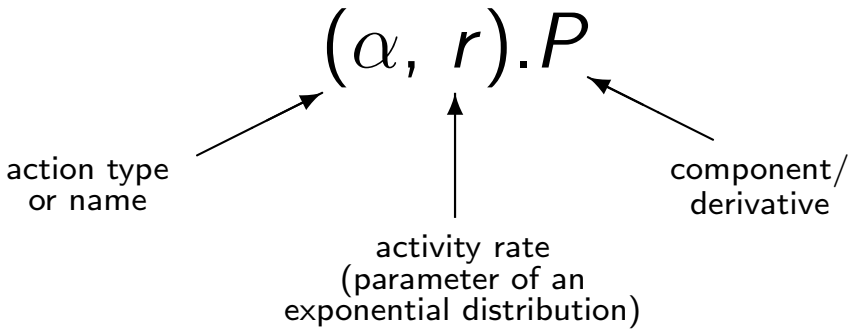


- The language is used to generate a CTMC for performance modelling.



Performance Evaluation Process Algebra

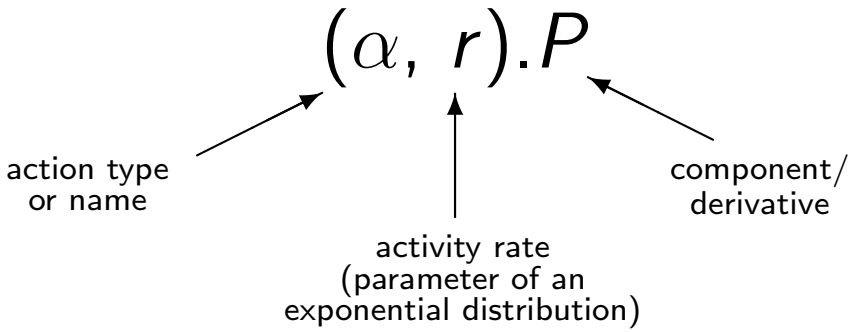
- Models are constructed from components which engage in activities.



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

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.



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

PEPA

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

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

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

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:	$A \stackrel{def}{=} S$	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:	$A \stackrel{def}{=} S$	assigning names
COOPERATION:	$P \underset{L}{\bowtie} P$	$\alpha \notin L$ individual actions
		$\alpha \in L$ shared actions

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:	$A \stackrel{def}{=} S$	assigning names
COOPERATION:	$P \underset{L}{\bowtie} P$	$\alpha \notin L$ individual actions $\alpha \in L$ shared actions
HIDING:	P/L	abstraction $\alpha \in L \Rightarrow \alpha \rightarrow \tau$

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.

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.

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.

Example: Browsers, server and download

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

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

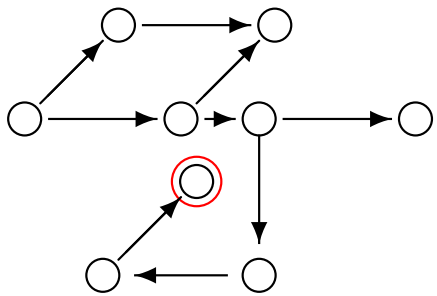
Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Integrated analysis: Reachability analysis

Reachability analysis

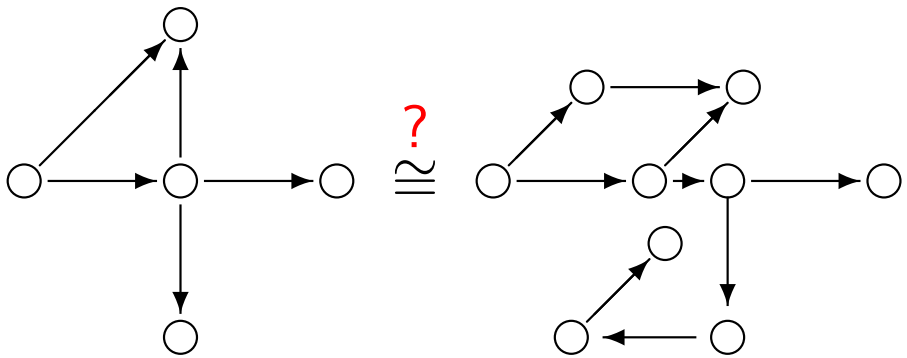
How long will it take for the system to arrive in a particular state?



Integrated analysis: Specification matching

Specification matching

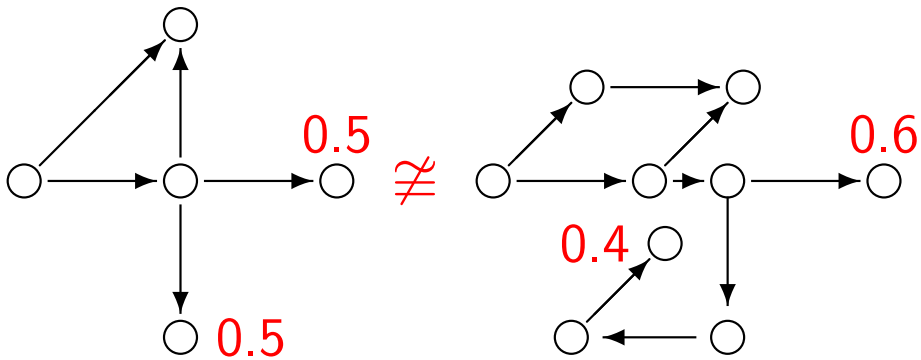
With what probability
does system behaviour
match its specification?



Integrated analysis: Specification matching

Specification matching

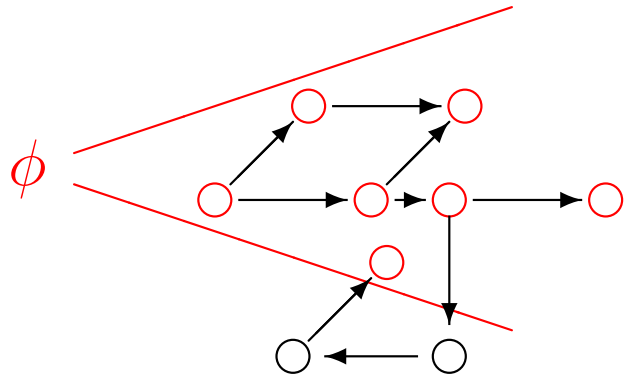
Does the "*frequency profile*" of the system match that of the specification?



Integrated analysis: Model checking

Model checking

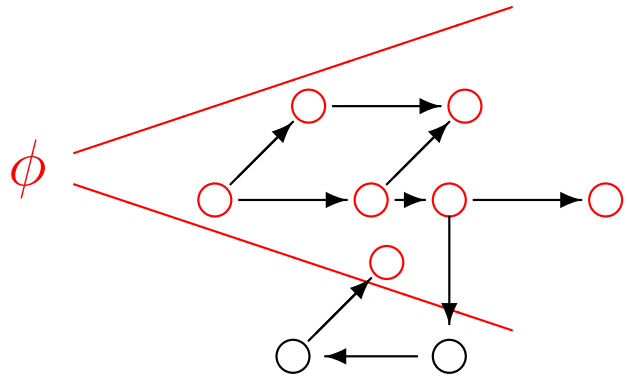
Does a given property ϕ hold within the system with a given probability?



Integrated analysis: Model checking

Model checking

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



Parallel Composition

- Parellel composition is the basis of the compositionality in a process algebra

Parallel Composition

- Parellel composition is the basis of the compositionality in a process algebra — it defines which components interact and how.

Parallel Composition

- Parellel 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.

Parallel Composition

- Parellel 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.

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.

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.

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

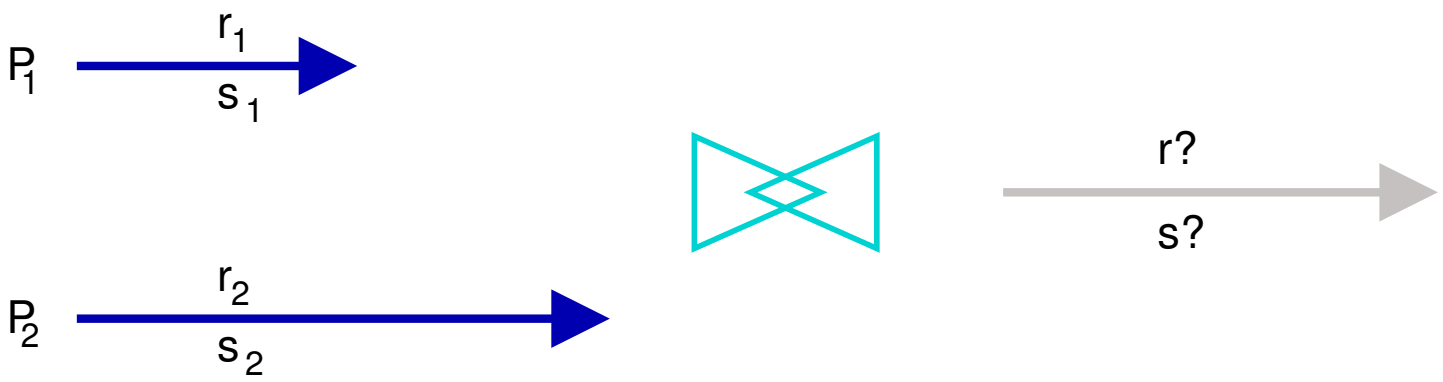
- No distinction between input and output actions.
- Communication or synchronisation takes place on the basis of shared names.
- The resulting action has the same name.

Most stochastic process algebras adopt CSP-style synchronisation.

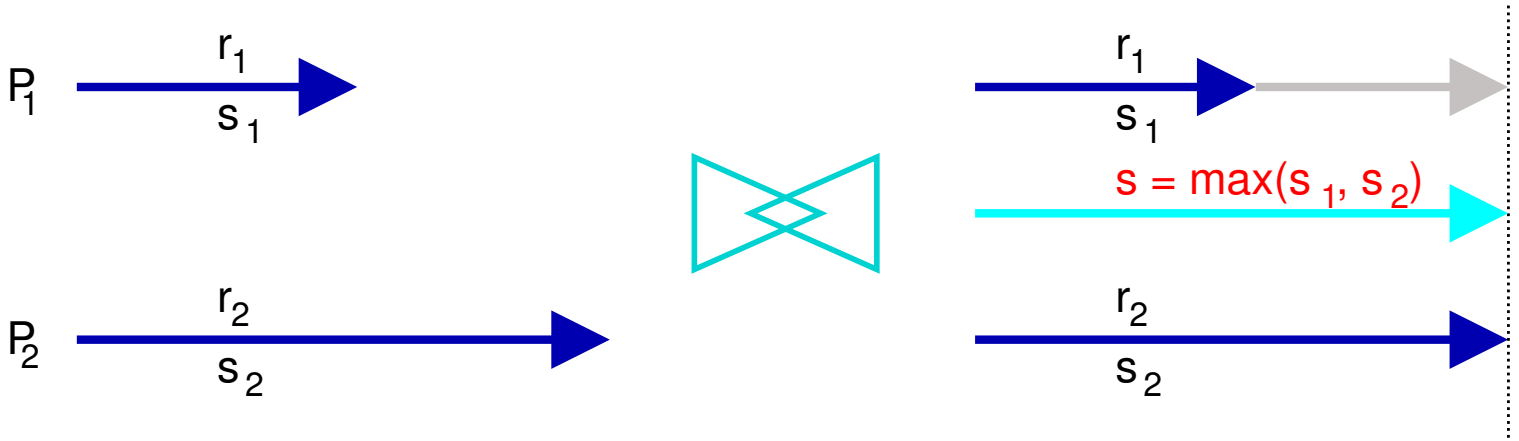
Timed Synchronisation

- The issue of what it means for two timed activities to synchronise is a vexed one....

Timed Synchronisation

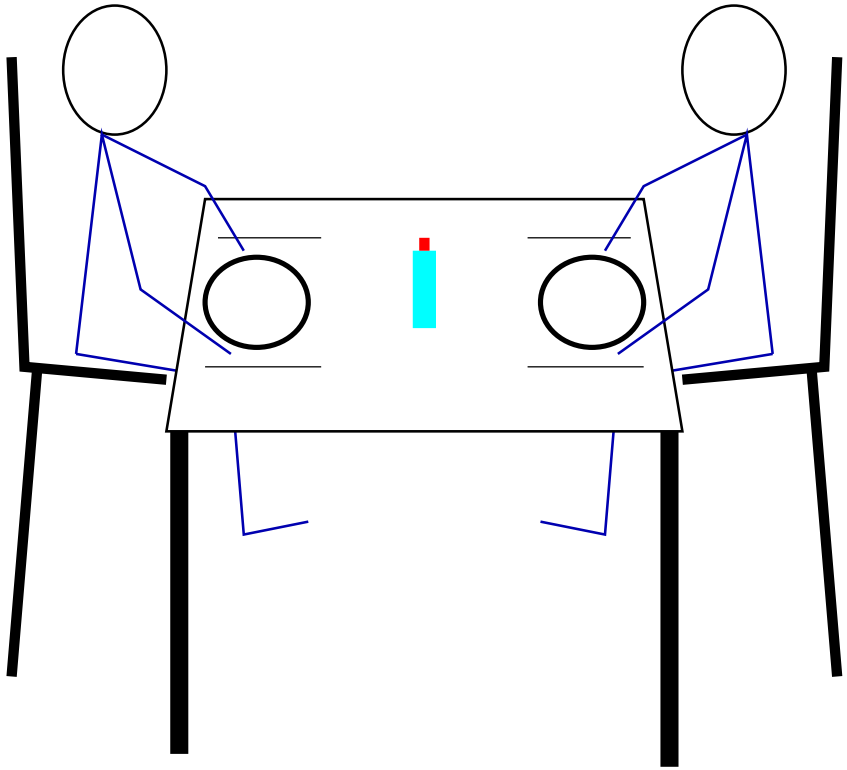


Timed Synchronisation

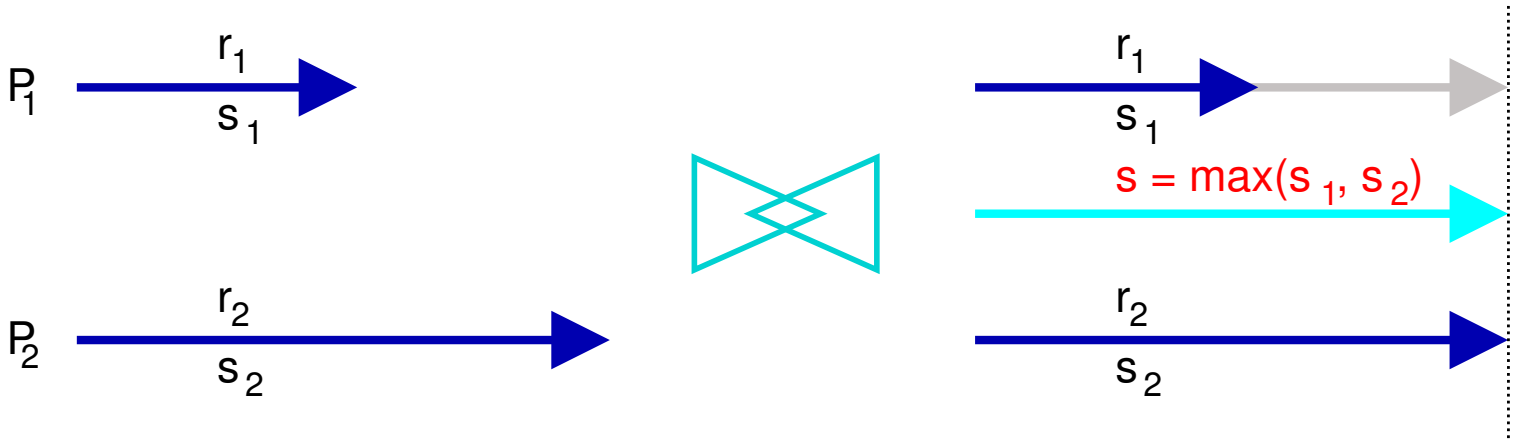


Barrier Synchronisation

Timed Synchronisation

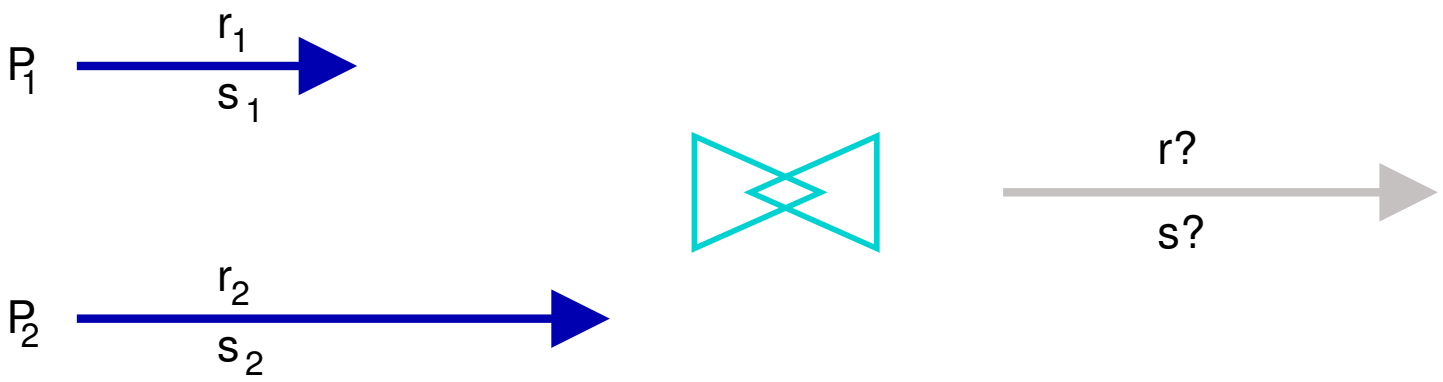


Timed Synchronisation



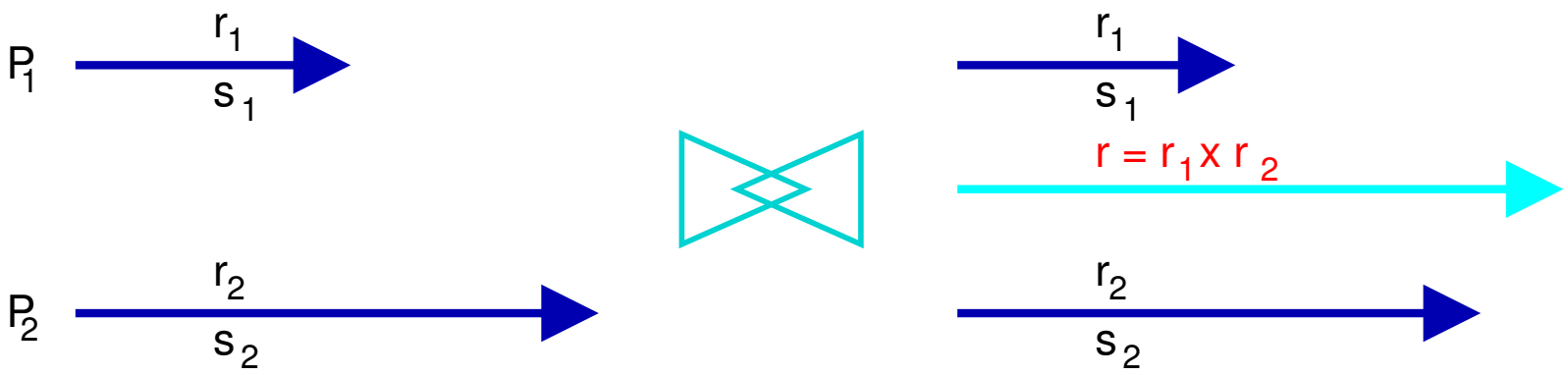
s is no longer exponentially distributed

Timed Synchronisation



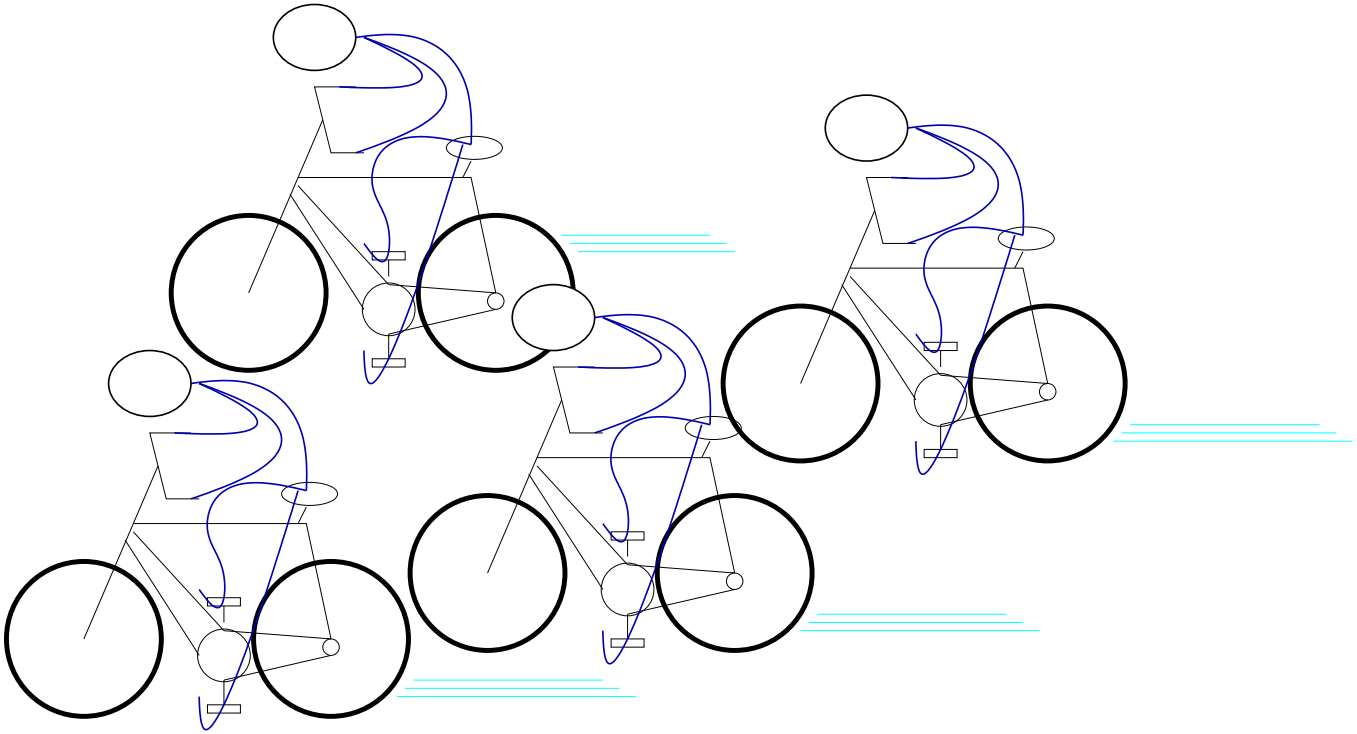
algebraic considerations limit choices

Timed Synchronisation

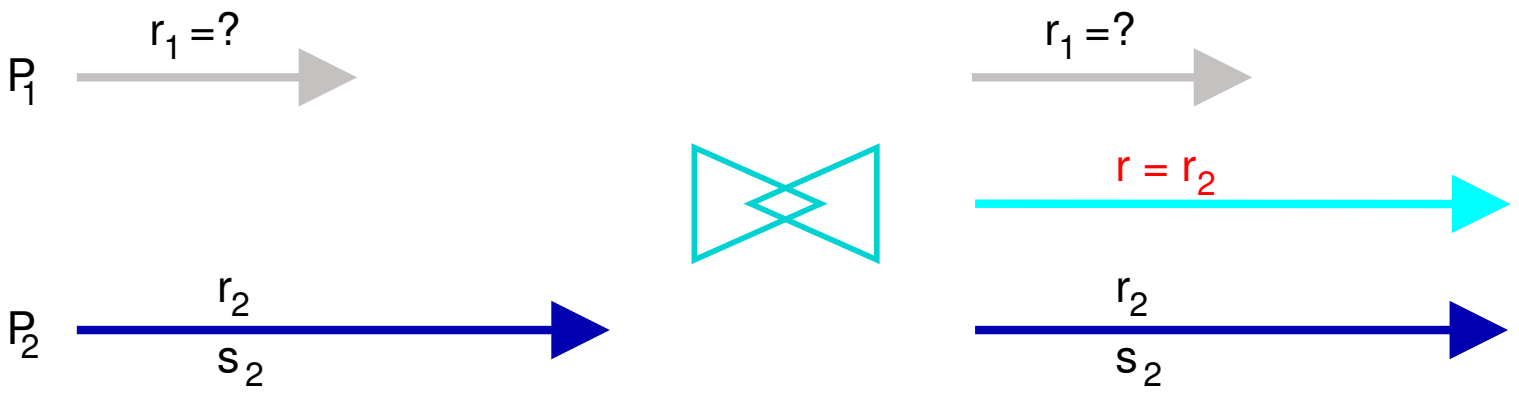


TIPP: new rate is product of individual rates

Timed Synchronisation

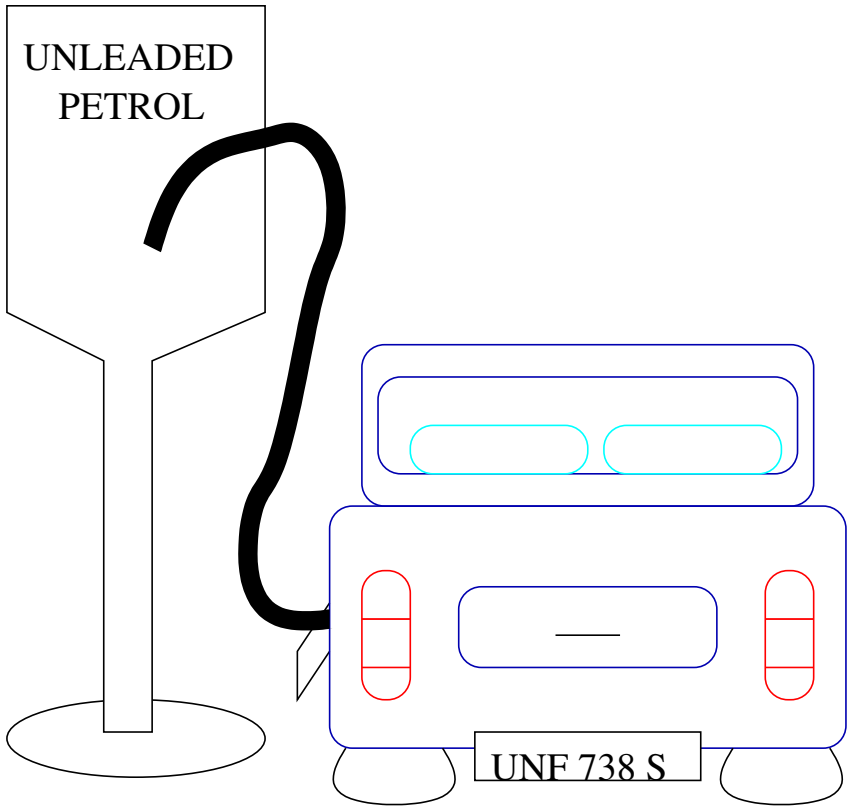


Timed Synchronisation

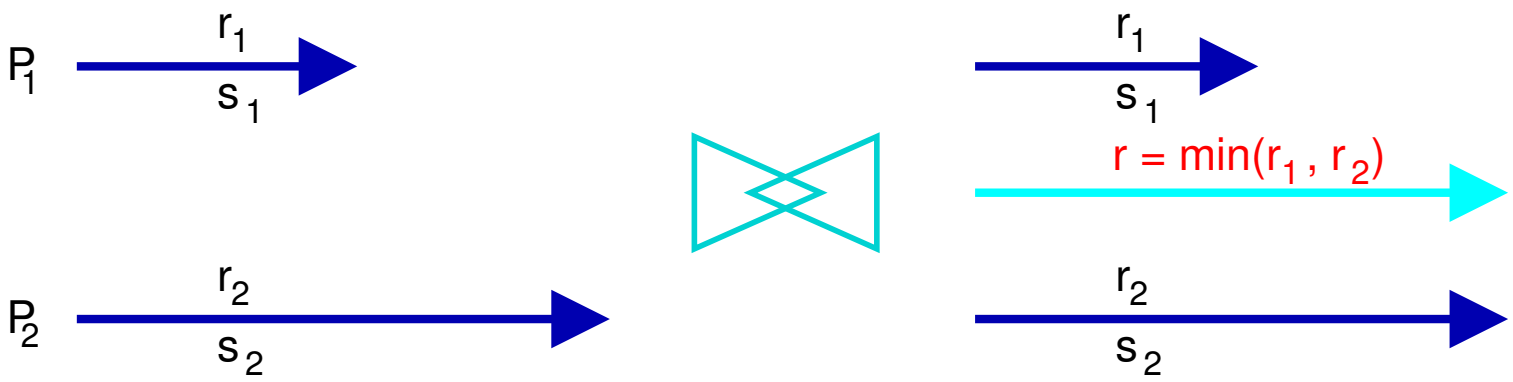


EMPA: one participant is passive

Timed Synchronisation

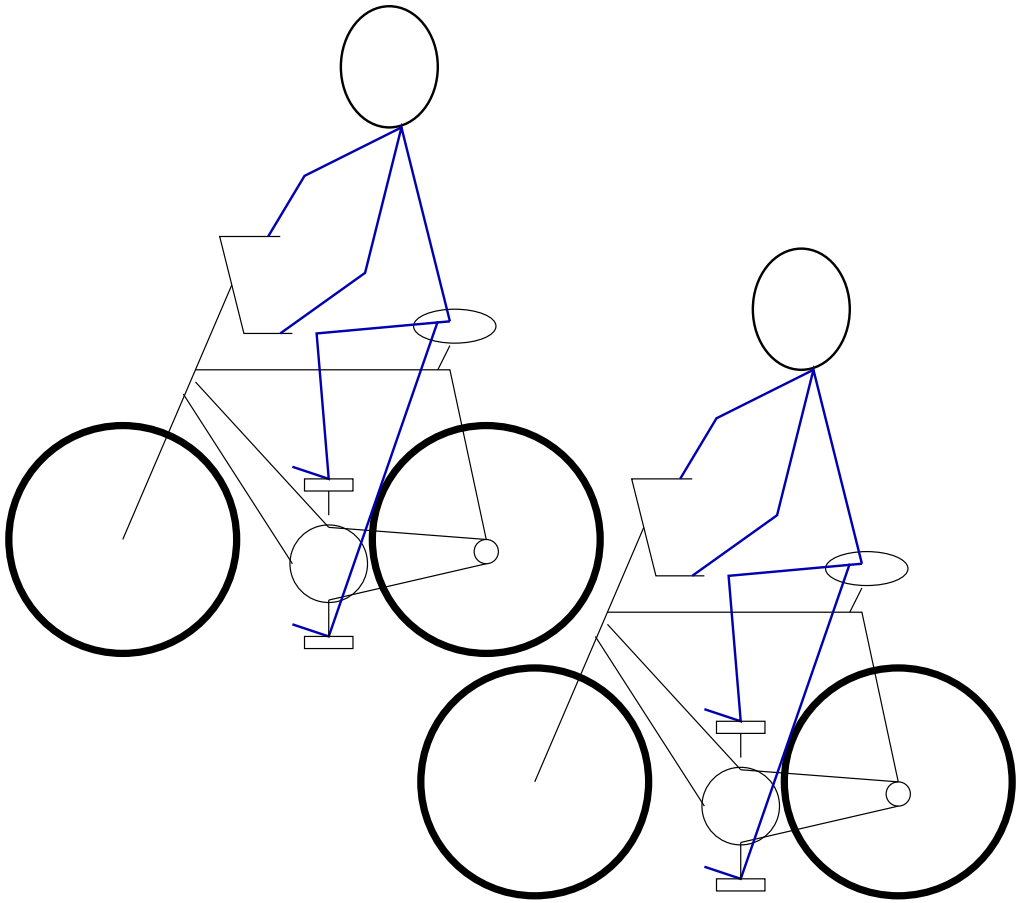


Timed Synchronisation



bounded capacity: new rate is the minimum of the rates

Timed Synchronisation



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.

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.

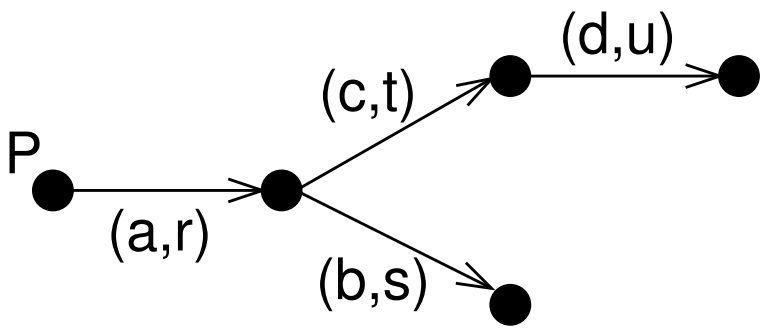
Cooperation in PEPA

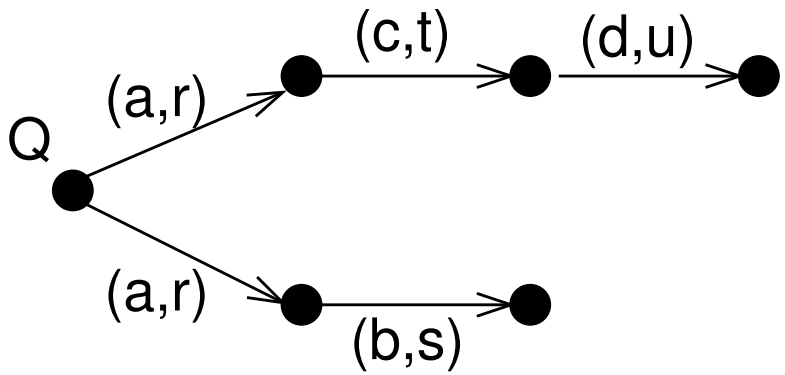
- In PEPA each component has a bounded capacity to carry out activities of any particular type, determined by the apparent rate for that type.
- Synchronisation, or cooperation cannot make a component exceed its bounded capacity.
- Thus the apparent rate of a cooperation is the minimum of the apparent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

Equivalence Relations

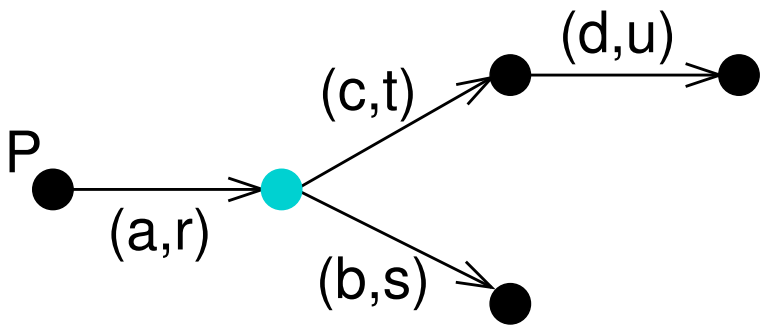
In process algebra equivalence relations are defined based on the notion of observability.

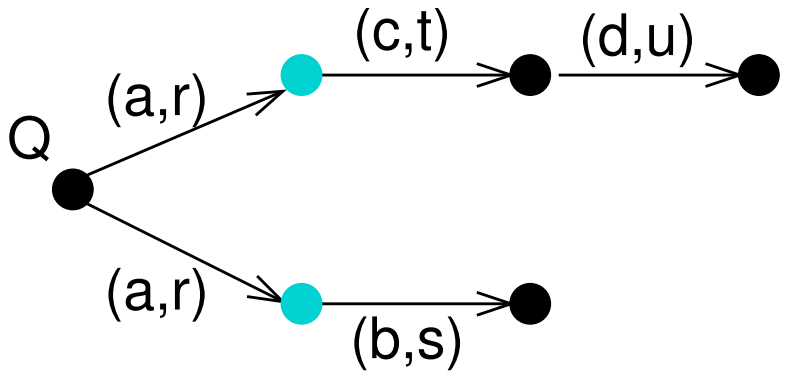
Equivalence Relations



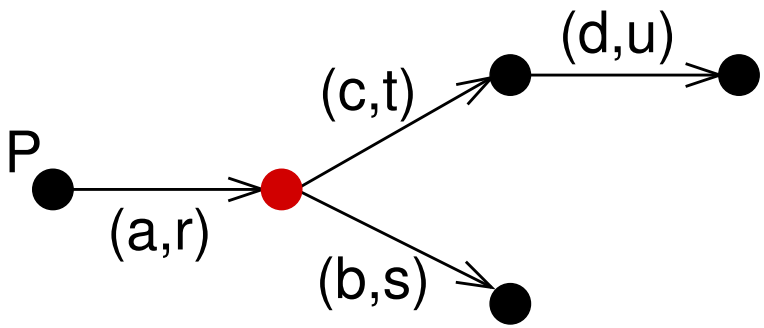


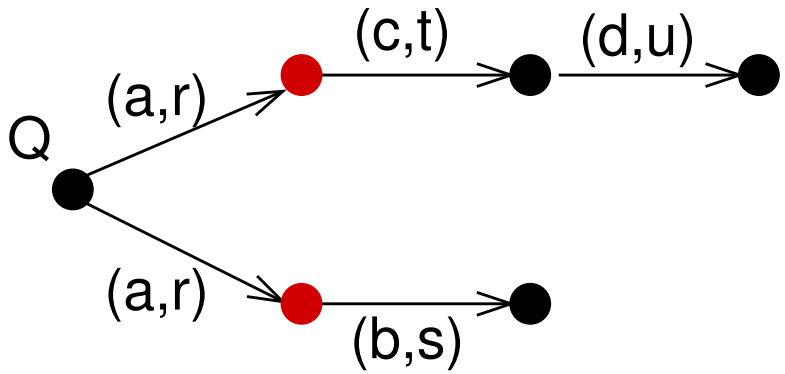
Equivalence Relations





Equivalence Relations

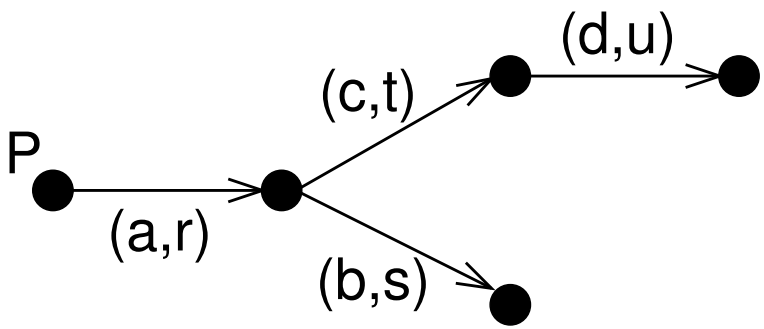


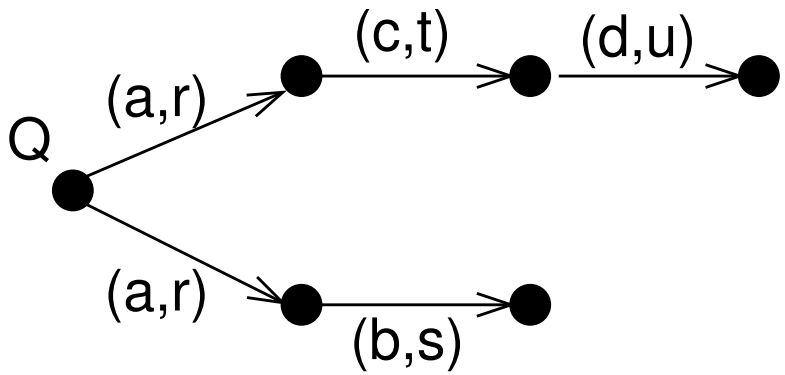


Equivalence Relations

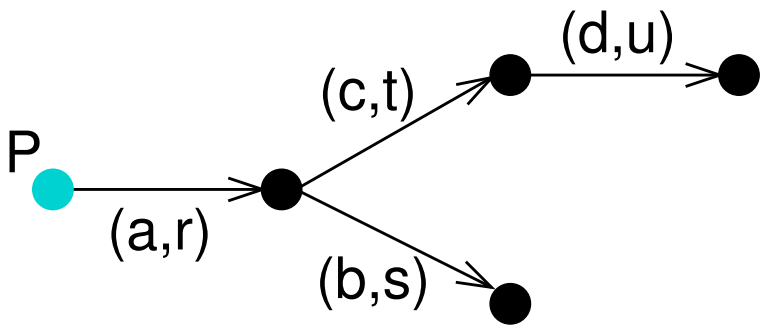
In PEPA observation is assumed to include the ability to record timing information over a number of runs.

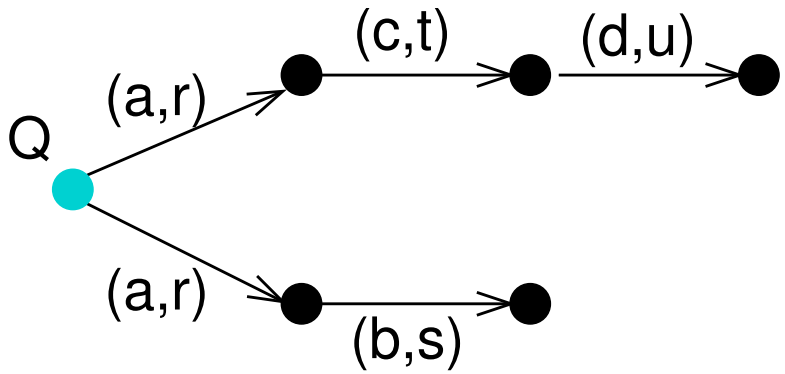
Equivalence Relations



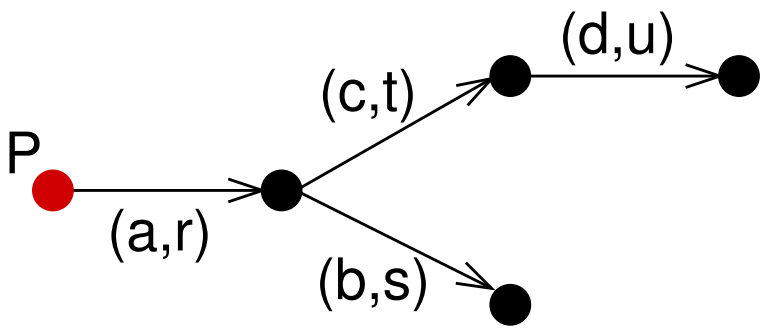


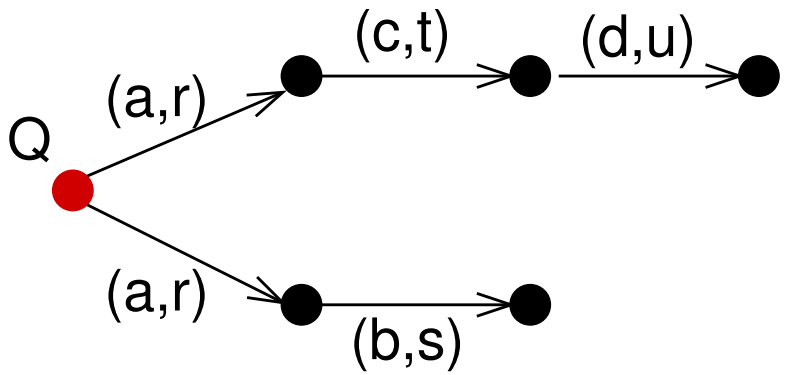
Equivalence Relations



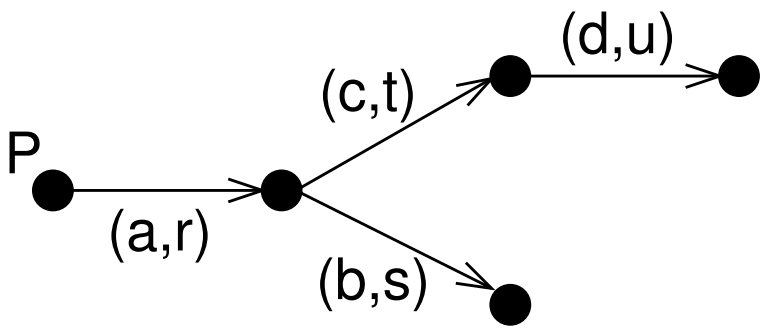


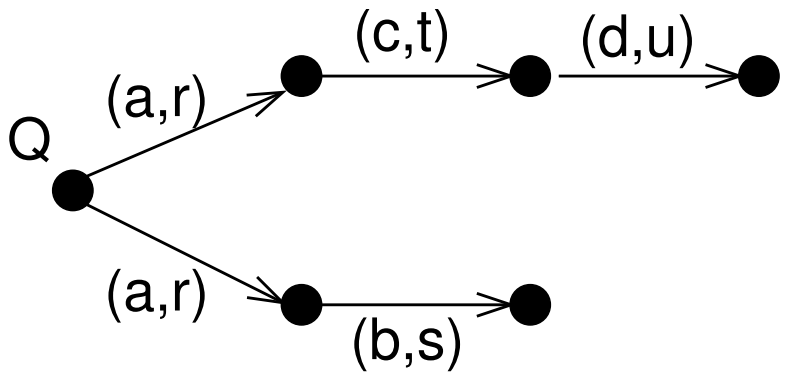
Equivalence Relations





Equivalence Relations





Equivalence Relations

The resulting equivalence relation is a bisimulation in the style of Larsen and Skou, and coincides with the Markov process notion of lumpability.

Aggregation and lumpability

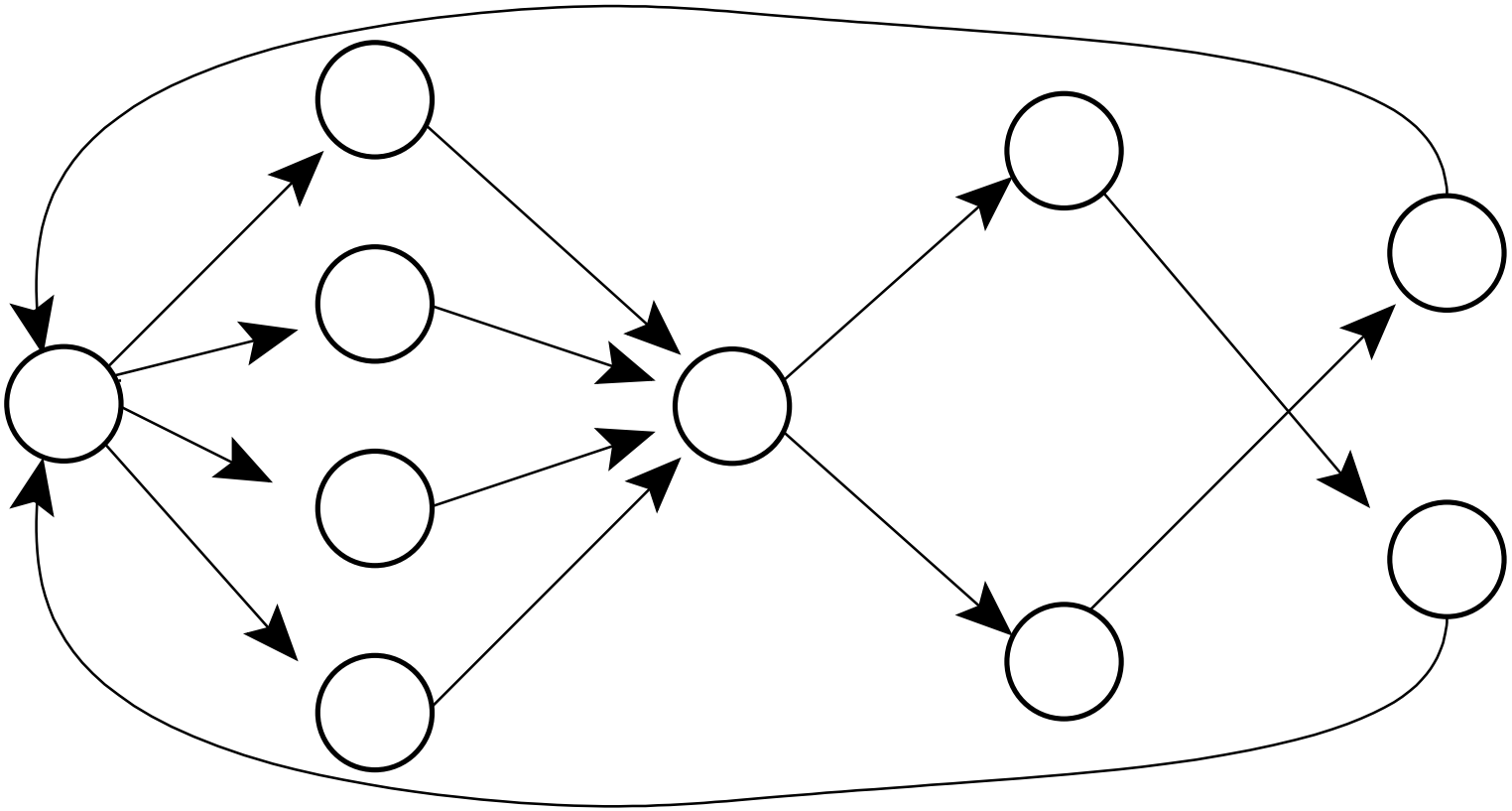
Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state, i.e. take a different stochastic representation of the same model.

Aggregation and lumpability

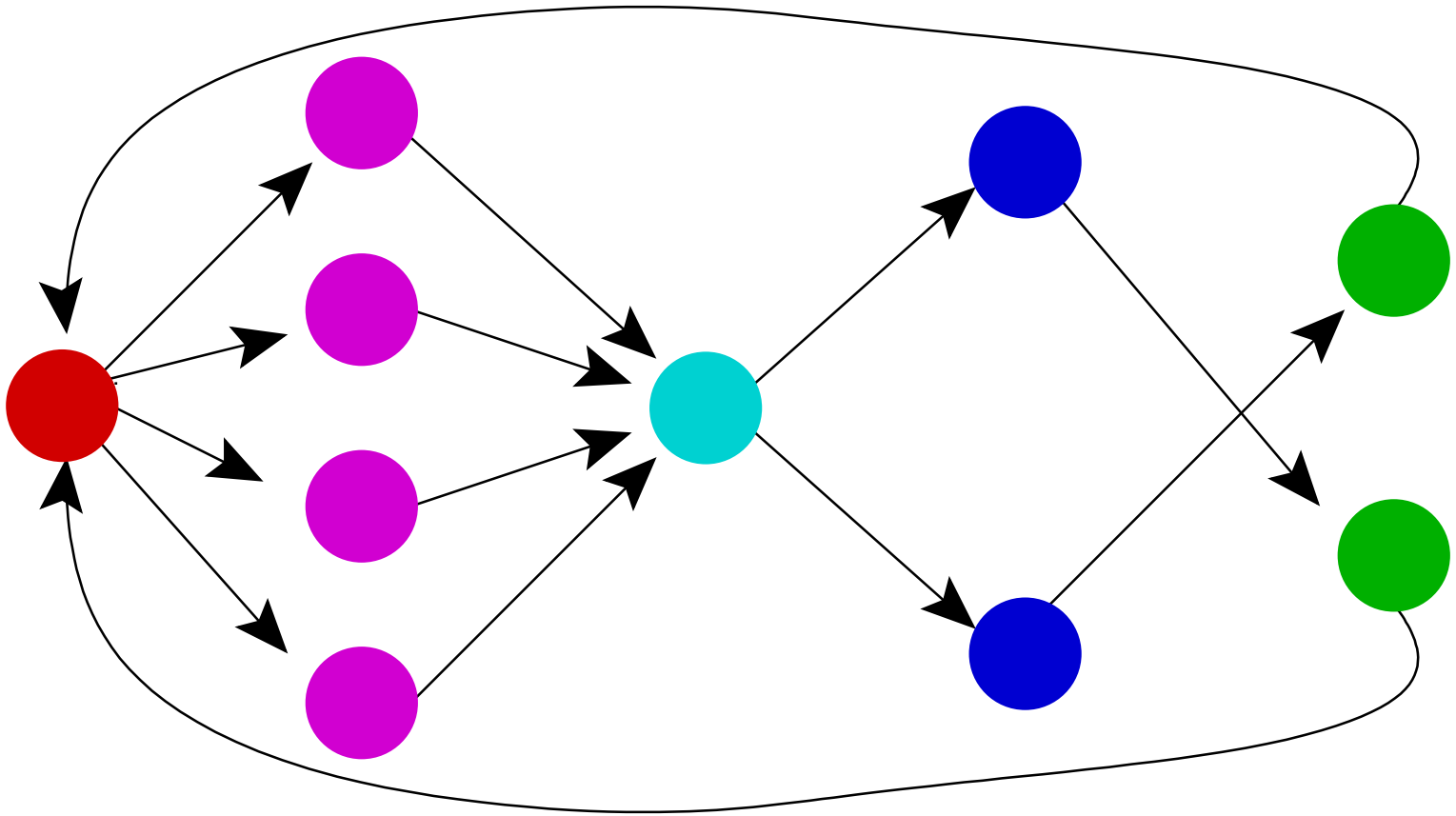
Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state, i.e. take a different stochastic representation of the same model.

A lumpable partition is the only partition of a Markov process which preserves the Markov property.

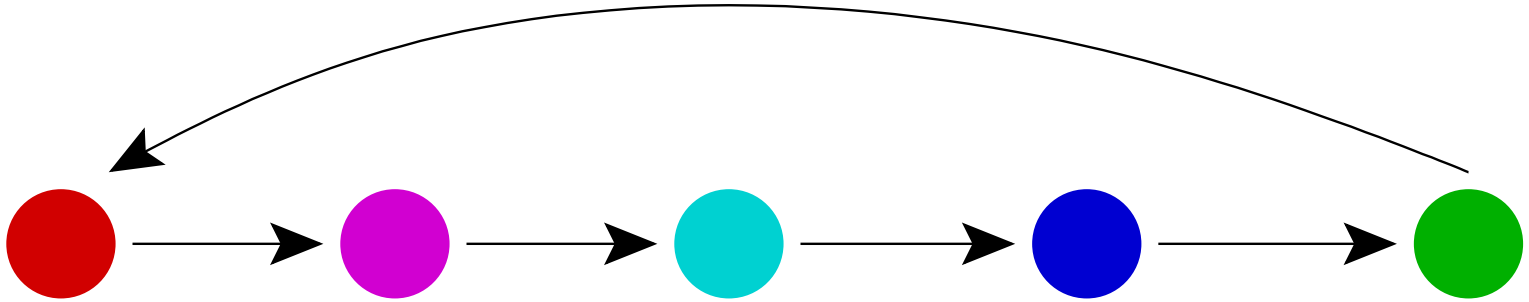
Aggregation and lumpability



Aggregation and lumpability



Aggregation and lumpability



Stochastic Modelling
Part II: Markovian methods

Stephen Gilmore
LFCS, University of Edinburgh

GLOBAN Summerschool
Lygby, Denmark

25th August 2006

Outline

Outline

Performance Evaluation Process Algebra

- PEPA (Performance Evaluation Process Algebra) is a high-level modelling language for distributed systems. It can be used to develop models of existing systems (abstraction) or designs for proposed ones (specification).
- PEPA can capture performance information in a process algebra setting. It is a stochastic process algebra.
- The definitive reference for PEPA is *A Compositional Approach to Performance Modelling*, Jane Hillston, Cambridge University Press, 1996.

Strengths of stochastic process algebras

SPAs have strengths in the areas of semantic definition, inherent compositionality and the existence of important equivalence relations (including bisimulation). This relation provides the basis for aggregation of PEPA models.

Terminology

The components in a PEPA model engage, cooperatively or individually, in activities. Each activity has an action type which corresponds to the actions of the system being modelled.

Terminology

The components in a PEPA model engage, cooperatively or individually, in activities. Each activity has an action type which corresponds to the actions of the system being modelled.

To represent unimportant or unknown actions there is a distinguished action type, τ.

Quantitative aspects

Every activity in PEPA has an associated activity rate which may be any positive real number, or the distinguished symbol "$\top$", meaning unspecified, read as 'top'.

Quantitative aspects

Every activity in PEPA has an associated activity rate which may be any positive real number, or the distinguished symbol "$\top$", meaning unspecified, read as 'top'.

Components and activities are primitives. PEPA also provides a small set of combinators.

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 (sequential sublanguage)

$(\alpha, r).S$

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 (combinators)

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

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.

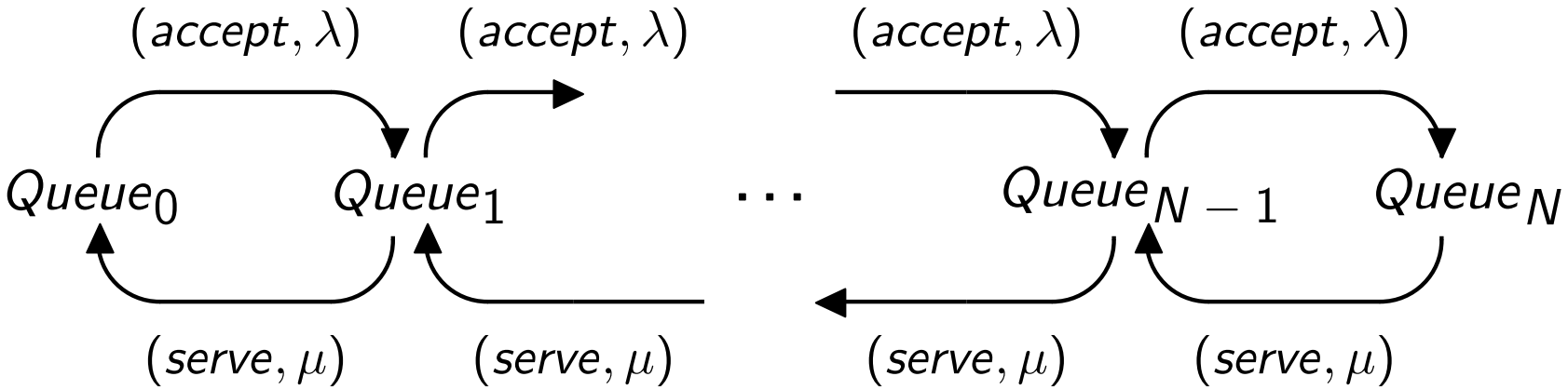
C / L

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

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

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



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

Expansion Law

$$P \underset{L}{\bowtie} Q =$$

Expansion Law

$$P \underset{L}{\bowtie} Q =$$

$$\sum\{(\alpha, r).(P' \underset{L}{\bowtie} Q) : \; P \xrightarrow{(\alpha,r)} P'; \; \alpha \notin L\} \; +$$

Expansion Law

$$P \underset{L}{\bowtie} Q =$$

$$\sum\{(\alpha, r).(P' \underset{L}{\bowtie} Q) : \; P \xrightarrow{(\alpha,r)} P'; \; \alpha \notin L\} \; +$$

$$\sum\{(\alpha, r).(P \underset{L}{\bowtie} Q') : \; Q \xrightarrow{(\alpha,r)} Q'; \; \alpha \notin L\} \; +$$

Expansion Law

$$P \underset{L}{\bowtie} Q =$$

$$\sum\{(\alpha, r).(P' \underset{L}{\bowtie} Q) : \; P \xrightarrow{(\alpha,r)} P';\; \alpha \notin L\} \; +$$

$$\sum\{(\alpha, r).(P \underset{L}{\bowtie} Q') : \; Q \xrightarrow{(\alpha,r)} Q';\; \alpha \notin L\} \; +$$

$$\sum\{(\alpha, r).(P' \underset{L}{\bowtie} Q') :$$

$$P \xrightarrow{(\alpha,r_1)} P';\; Q \xrightarrow{(\alpha,r_2)} Q';\; \alpha \in L\}$$

Synchronisation

What should be the impact of synchronisation on rate? There are many possibilities.

- Restrict synchronisations to have one active partner and one passive partner.

Synchronisation

What should be the impact of synchronisation on rate? There are many possibilities.

- Restrict synchronisations to have one active partner and one passive partner.
- Choose a function which satisfies a small number of algebraic properties.

Synchronisation

What should be the impact of synchronisation on rate? There are many possibilities.

- Restrict synchronisations to have one active partner and one passive partner.
- Choose a function which satisfies a small number of algebraic properties.
- Have the rate limited by the slowest participant in terms of apparent rate. This is the approach adopted by PEPA.

Bounded capacity

Within the cooperation framework, 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.

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 $r_\alpha(P)$.

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 used heavily when calculating the pairwise cooperation rate: when cooperating with another component, the bounded capacity principle ensures that the overall rate of cooperation does not exceed either of the consistuent apparent rates.

Apparent rate: definition

To summarise the original ruleset from [Hillston 96], the apparent rate function can be defined as:

$$r_\alpha(P) = \sum_{P \xrightarrow{(\alpha,\lambda_i)}} \lambda_i \tag{1}$$

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:

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

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

Random experiments and events

- To apply probability theory to the process under study, we view it as a random experiment.

Random experiments and events

- To apply probability theory to the process under study, we view it as a random experiment.
- The sample space of a random experiment is the set of all individual outcomes of the experiment.

Random experiments and events

- To apply probability theory to the process under study, we view it as a random experiment.
- The sample space of a random experiment is the set of all individual outcomes of the experiment.
- These individual outcomes are also called sample points or elementary events.

Random experiments and events

- To apply probability theory to the process under study, we view it as a random experiment.
- The sample space of a random experiment is the set of all individual outcomes of the experiment.
- These individual outcomes are also called sample points or elementary events.
- An event is a subset of a sample space.

Random variables

We are interested in the dynamics of a system as events happen over time. A function which associates a (real-valued) number with the outcome of an experiment is known as a random variable. Formally, a random variable X is a real-valued function defined on a sample space Ω.

Measurable functions

If X is a random variable, and x is a real number, we write $X \leq x$ for the event

$$\{\, \omega : \omega \in \Omega \text{ and } X(\omega) \leq x \,\}$$

Measurable functions

If X is a random variable, and x is a real number, we write $X \leq x$ for the event

$$\{\omega : \omega \in \Omega \text{ and } X(\omega) \leq x\}$$

and we write $X = x$ for the event

$$\{\omega : \omega \in \Omega \text{ and } X(\omega) = x\}$$

Measurable functions

If X is a random variable, and x is a real number, we write $X \leq x$ for the event

$$\{\omega : \omega \in \Omega \text{ and } X(\omega) \leq x\}$$

and we write $X = x$ for the event

$$\{\omega : \omega \in \Omega \text{ and } X(\omega) = x\}$$

Another property required of a random variable is that the set $X \leq x$ is an event for each real x. This is necessary so that probability calculations can be made. A function having this property is said to be a measurable function or measurable in the Borel sense.

Distribution function

For each random variable X we define its distribution function F for each real x by

$$F(x) = \Pr[X \leq x]$$

Distribution function

For each random variable X we define its distribution function F for each real x by

$$F(x) = \Pr[X \leq x]$$

We associate another function $p(\cdot)$, called the probability mass function of X (pmf), for each real x:

$$p(x) = \Pr[X = x]$$

Continuous random variables

A random variable X is continuous if $p(x) = 0$ for all real x.

Continuous random variables

A random variable X is continuous if $p(x) = 0$ for all real x.

(If X is a *continuous* random variable, then X can assume infinitely many values, and so it is reasonable that the probability of its assuming any *specific* value we choose beforehand is zero.)

Continuous random variables

A random variable X is continuous if $p(x) = 0$ for all real x.

(If X is a *continuous* random variable, then X can assume infinitely many values, and so it is reasonable that the probability of its assuming any *specific* value we choose beforehand is zero.)

The distribution function for a continuous random variable is a continuous function in the usual sense.

Exponential random variables, distribution function

The random variable X is said to be an *exponential random variable with parameter* λ ($\lambda > 0$) or to have an exponential distribution with parameter λ if it has the distribution function

$$F(x) = \begin{cases} 1 - e^{-\lambda x} & \text{for } x > 0 \\ 0 & \text{for } x \leq 0 \end{cases}$$

Exponential random variables, distribution function

The random variable X is said to be an *exponential random variable with parameter* λ ($\lambda > 0$) or to have an exponential distribution with parameter λ if it has the distribution function

$$F(x) = \begin{cases} 1 - e^{-\lambda x} & \text{for } x > 0 \\ 0 & \text{for } x \leq 0 \end{cases}$$

Some authors call this distribution the negative exponential distribution.

Exponential random variables, density function

The density function $f = dF/dx$ is given by

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

Mean, or expected value

If X is a continuous random variable with density function $f(\cdot)$, we define the mean or expected value of X, $\mu = E[X]$ by

$$\mu = E[X] = \int_{-\infty}^{\infty} xf(x)dx$$

Mean, or expected value, of the exponential distribution

Suppose X has an exponential distribution with parameter $\lambda > 0$. Then

$$\mu = E[X] = \int_{-\infty}^{\infty} x\lambda e^{-\lambda x} dx$$

Mean, or expected value, of the exponential distribution

Suppose X has an exponential distribution with parameter $\lambda > 0$. Then

$$\mu = E[X] = \int_{-\infty}^{\infty} x\lambda e^{-\lambda x} dx = \frac{1}{\lambda}$$

Exponential inter-event time distribution

The time interval between successive events can also be deduced. Let $F(t)$ be the distribution function of T, the time between events. Consider $\Pr(T > t) = 1 - F(t)$:

$$\Pr(T > t) \quad = \quad \Pr(\text{No events in an interval of length } t)$$

Exponential inter-event time distribution

The time interval between successive events can also be deduced. Let $F(t)$ be the distribution function of T, the time between events. Consider $\Pr(T > t) = 1 - F(t)$:

$$\begin{aligned} \Pr(T > t) &= \Pr(\text{No events in an interval of length } t) \\ &= 1 - F(t) \end{aligned}$$

Exponential inter-event time distribution

The time interval between successive events can also be deduced. Let $F(t)$ be the distribution function of T, the time between events. Consider $\Pr(T > t) = 1 - F(t)$:

$$\begin{aligned} \Pr(T > t) &= \Pr(\text{No events in an interval of length } t) \\ &= 1 - F(t) \\ &= 1 - (1 - e^{-\lambda t}) \end{aligned}$$

Exponential inter-event time distribution

The time interval between successive events can also be deduced. Let $F(t)$ be the distribution function of T, the time between events. Consider $\Pr(T > t) = 1 - F(t)$:

$$\begin{aligned}
\Pr(T > t) &= \Pr(\text{No events in an interval of length } t) \\
&= 1 - F(t) \\
&= 1 - (1 - e^{-\lambda t}) \\
&= e^{-\lambda t}
\end{aligned}$$

Memoryless property of the exponential distribution

The memoryless property of the exponential distribution is so called because the time to the next event is independent of when the last event occurred.

Memoryless property of the exponential distribution

Suppose that the last event was at time 0. What is the probability that the next event will be after $t + s$, given that time t has elapsed since the last event, and no events have occurred?

Memoryless property of the exponential distribution

Suppose that the last event was at time 0. What is the probability that the next event will be after $t + s$, given that time t has elapsed since the last event, and no events have occurred?

$$\Pr(T > t + s \mid T > t) = \frac{\Pr(T > t + s \text{ and } T > t)}{\Pr(T > t)}$$

Memoryless property of the exponential distribution

Suppose that the last event was at time 0. What is the probability that the next event will be after $t + s$, given that time t has elapsed since the last event, and no events have occurred?

$$\begin{aligned} \Pr(T > t+s \mid T > t) &= \frac{\Pr(T > t+s \text{ and } T > t)}{\Pr(T > t)} \\ &= \frac{e^{-\lambda(t+s)}}{e^{-\lambda t}} \end{aligned}$$

Memoryless property of the exponential distribution

Suppose that the last event was at time 0. What is the probability that the next event will be after $t + s$, given that time t has elapsed since the last event, and no events have occurred?

$$\begin{aligned} \Pr(T > t+s \mid T > t) &= \frac{\Pr(T > t+s \text{ and } T > t)}{\Pr(T > t)} \\ &= \frac{e^{-\lambda(t+s)}}{e^{-\lambda t}} \\ &= e^{-\lambda s} \end{aligned}$$

Memoryless property of the exponential distribution

Suppose that the last event was at time 0. What is the probability that the next event will be after $t + s$, given that time t has elapsed since the last event, and no events have occurred?

$$\begin{aligned} \Pr(T > t+s \mid T > t) &= \frac{\Pr(T > t+s \text{ and } T > t)}{\Pr(T > t)} \\ &= \frac{e^{-\lambda(t+s)}}{e^{-\lambda t}} \\ &= e^{-\lambda s} \end{aligned}$$

This value is independent of t (and so the time already spent has not been remembered).

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. The time allocated to the timer is determined by 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 an activity of type α. An activity may be preempted, or aborted, if another one completes first.

PEPA and Markov processes

In a PEPA model if we define the stochastic process $X(t)$, such that $X(t) = C_i$ indicates that the system behaves as component C_i at time t, then $X(t)$ is a Markov process which can be characterised by a matrix, $\boldsymbol{Q}$.

PEPA and Markov processes

In a PEPA model if we define the stochastic process $X(t)$, such that $X(t) = C_i$ indicates that the system behaves as component C_i at time t, then $X(t)$ is a Markov process which can be characterised by a matrix, $\boldsymbol{Q}$.

A stationary or equilibrium probability distribution, $\pi(\cdot)$, exists for every time-homogeneous irreducible Markov process whose states are all positive-recurrent.

PEPA and Markov processes

In a PEPA model if we define the stochastic process $X(t)$, such that $X(t) = C_i$ indicates that the system behaves as component C_i at time t, then $X(t)$ is a Markov process which can be characterised by a matrix, $\boldsymbol{Q}$.

A stationary or equilibrium probability distribution, $\pi(\cdot)$, exists for every time-homogeneous irreducible Markov process whose states are all positive-recurrent.

This distribution is found by solving the global balance equation

$$\pi \boldsymbol{Q} = \boldsymbol{0}$$

subject to the normalisation condition

$$\sum \pi(C_i) = 1.$$

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. 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.

Consequences

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.

Continuous-Time Markov Chains (CTMCs)

A Markov process with discrete state space and discrete index set is called a Markov chain. The future behaviour of a Markov chain depends only on its current state, and not on how that state was reached. This is the Markov, or memoryless, property.

$$\begin{aligned} & \Pr(X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n, \ldots, X(t_0) = x_0) \\ = \quad & \Pr(X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n) \end{aligned}$$

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

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

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

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).

Roll your own!

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

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

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.

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).

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.

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).

Model solution and analysis

A continuous time Markov chain (CTMC) is generated from a PEPA model via its structured operational semantics.

Model solution and analysis

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.

Model solution and analysis

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.

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.

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

$$
\begin{array}{lcll}
F & ::= & \mathsf{tt} & \text{(truth)} \\
 & | & \nabla_\alpha & \text{(inability)} \\
 & | & \neg F & \text{(negation)} \\
 & | & F_1 \wedge F_2 & \text{(conjunction)} \\
 & | & \langle\alpha\rangle_\mu F & \text{(“at least”)}
\end{array}
$$

Relation to PEPA

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

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.

$$P \models \mathsf{tt}$$

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.

$$P \models \mathsf{tt}$$

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

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.

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

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

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

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.

$$P \models \mathsf{tt}$$

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

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

$$P \models \nabla_\alpha \text{ if } P \xrightarrow{\alpha}\!\!\!\!\!\!/$$

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.

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

Modal characterisation of strong equivalence

Let P be a model of a PEPA process. Then

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

That is to say that 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.

Case study: active badges

We have used the PEPA modelling language and its accompanying specification language to analyse the configuration of a location tracking system based on active badges. Active badges transmit unique infra-red signals which are detected by networked sensors. These report locations back to a central database.

Case study: active badges

The badges are battery-powered and the tradeoff in the system is between the conservation of battery power and the accuracy of the information harvested from the sensors. When transmissions from badges collide, the badges sleep for a randomly determined time before retrying.

Active badges: the PEPA model

The PEPA model of this system tracks the progress of one badge-wearer around three connected corridors (numbered 14, 15 and 16). The activities which are performed in the system include the badge registering with a sensor (at rate r), the person moving to another corridor (at rate m) and a sensor reporting back to the central database (at rate s).

Active badges: the PEPA model

Person

$$
\begin{aligned}
P_{14} &\stackrel{def}{=} (reg_{14}, r).P_{14} + (move_{15}, m).P_{15} \\
P_{15} &\stackrel{def}{=} (reg_{15}, r).P_{15} + (move_{14}, m).P_{14} + (move_{16}, m).P_{16} \\
P_{16} &\stackrel{def}{=} (reg_{16}, r).P_{16} + (move_{15}, m).P_{15}
\end{aligned}
$$

Active badges: the PEPA model

Person

$$
\begin{aligned}
P_{14} &\stackrel{def}{=} (reg_{14}, r).P_{14} + (move_{15}, m).P_{15} \\
P_{15} &\stackrel{def}{=} (reg_{15}, r).P_{15} + (move_{14}, m).P_{14} + (move_{16}, m).P_{16} \\
P_{16} &\stackrel{def}{=} (reg_{16}, r).P_{16} + (move_{15}, m).P_{15}
\end{aligned}
$$

Sensor

$$
\begin{aligned}
S_{14} &\stackrel{def}{=} (reg_{14}, \top).(rep_{14}, s).S_{14} \\
S_{15} &\stackrel{def}{=} (reg_{15}, \top).(rep_{15}, s).S_{15} \\
S_{16} &\stackrel{def}{=} (reg_{16}, \top).(rep_{16}, s).S_{16}
\end{aligned}
$$

Active badges: the PEPA model

Database

$$
\begin{aligned}
DB_{14} &\stackrel{def}{=} (rep_{14}, \top).DB_{14} + (rep_{15}, \top).DB_{15} + (rep_{16}, \top).DB_{16} \\
DB_{15} &\stackrel{def}{=} (rep_{14}, \top).DB_{14} + (rep_{15}, \top).DB_{15} + (rep_{16}, \top).DB_{16} \\
DB_{16} &\stackrel{def}{=} (rep_{14}, \top).DB_{14} + (rep_{15}, \top).DB_{15} + (rep_{16}, \top).DB_{16}
\end{aligned}
$$

Active badges: the PEPA model

Database

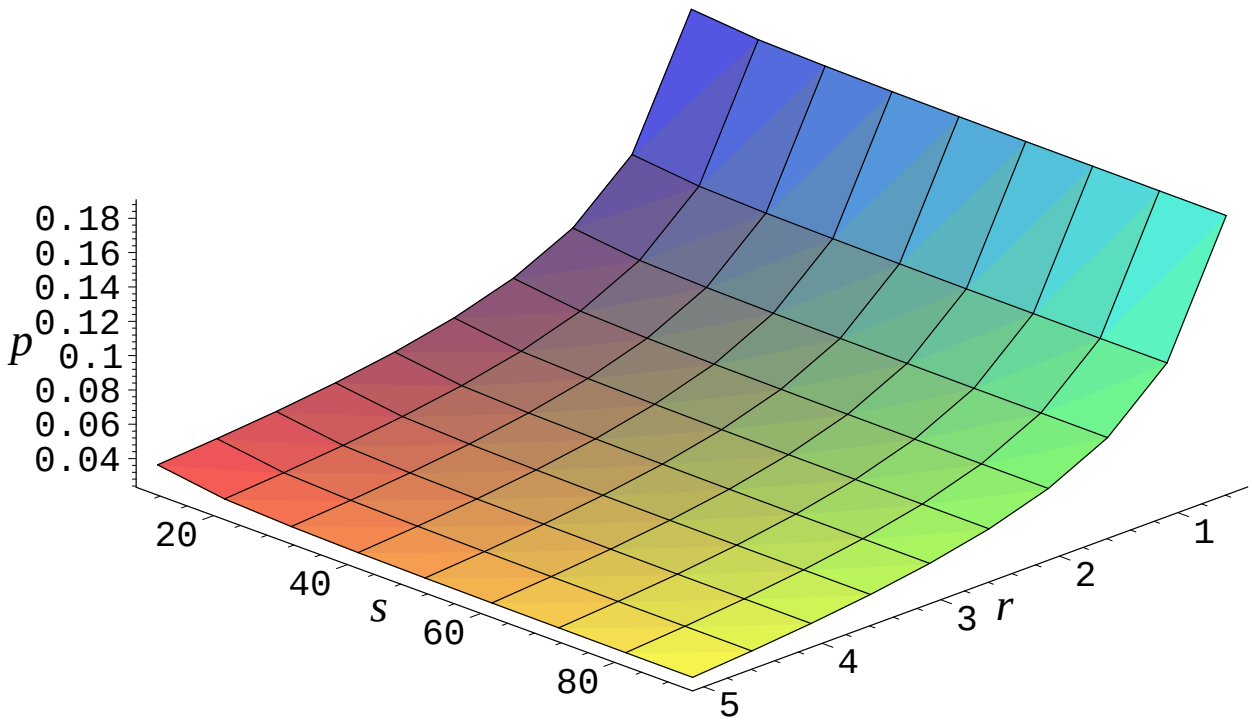
$$
\begin{aligned}
DB_{14} &\stackrel{def}{=} (rep_{14}, \top).DB_{14} + (rep_{15}, \top).DB_{15} + (rep_{16}, \top).DB_{16} \\
DB_{15} &\stackrel{def}{=} (rep_{14}, \top).DB_{14} + (rep_{15}, \top).DB_{15} + (rep_{16}, \top).DB_{16} \\
DB_{16} &\stackrel{def}{=} (rep_{14}, \top).DB_{14} + (rep_{15}, \top).DB_{15} + (rep_{16}, \top).DB_{16}
\end{aligned}
$$

System

$$P_{14} \underset{L}{\bowtie} (S_{14} \parallel S_{15} \parallel S_{16}) \underset{M}{\bowtie} DB_{14}$$

where $L = \{ reg_{14}, reg_{15}, reg_{16} \}$
$M = \{ rep_{14}, rep_{15}, rep_{16} \}$

Probability that the database holds inaccurate information



Outline

A high-level language for performance modelling

PEPA is a high-level language for performance modelling. PEPA models describe stochastic (in fact, Markovian) processes.

A high-level language for performance modelling

PEPA is a high-level language for performance modelling. PEPA models describe stochastic (in fact, Markovian) processes.

Without a high-level modelling language, the modeller would be forced to work with unstructured matrix representations of stochastic processes.

A high-level language for performance modelling

PEPA is a high-level language for performance modelling. PEPA models describe stochastic (in fact, Markovian) processes.

Without a high-level modelling language, the modeller would be forced to work with unstructured matrix representations of stochastic processes.

Process algebras are useful because they allow the definition of equivalence relations between model components and these relations may be used in model simplification.

The PEPA Workbench

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 workbench relieves the modeller of this work.

The PEPA Workbench: functionality

The workbench will report errors in the model function:

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

The workbench also generates the transition graph of the model, computes the number of states, formulates the Markov process matrix Q and communicates the matrix to a solver.

The PEPA Workbench: functionality

The workbench will report errors in the model function:

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

The workbench 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 workbench provides a simple pattern language for selecting states from the stationary distribution.

PEPA Workbench 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 Workbench 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 Workbench output

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

Stochastic Modelling
Part III: Scalability

Stephen Gilmore
LFCS, University of Edinburgh

GLOBAN Summerschool
Lygby, Denmark

25th August 2006

Outline

Performance Evaluation Process Algebra

PEPA components perform activities either independently or in co-operation with other components.

Performance Evaluation Process Algebra

PEPA components perform activities either independently or in co-operation with other components.

The rate at which an activity is performed is quantified by some component in each co-operation. The symbol $\top$ indicates that the rate value is quantified elsewhere (not in this component).

Performance Evaluation Process Algebra

PEPA components perform activities either independently or in co-operation with other components.

The rate at which an activity is performed is quantified by some component in each co-operation. The symbol $\top$ indicates that the rate value is quantified elsewhere (not in this component).

$(\alpha, r).P$	Prefix
$P_1 + P_2$	Choice
$P_1 \underset{L}{\bowtie} P_2$	Co-operation
P/L	Hiding
X	Variable

PEPA: informal semantics (sequential sublanguage)

$(\alpha, r).S$

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 (combinators)

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

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.

Derived forms and additional syntax

$P_1 \parallel P_2$ is a derived form for $P_1 \underset{\emptyset}{\bowtie} P_2$.

Derived forms and additional syntax

$P_1 \parallel P_2$ is a derived form for $P_1 \underset{\emptyset}{\bowtie} P_2$.

Because we are interested in transient behaviour we use the deadlocked process *Stop*.

Derived forms and additional syntax

$P_1 \parallel P_2$ is a derived form for $P_1 \underset{\emptyset}{\bowtie} P_2$.

Because we are interested in transient behaviour we use the deadlocked process *Stop*.

When working with large numbers of jobs and servers, we write $P[n]$ to denote an array of n copies of P executing in parallel.

Derived forms and additional syntax

$P_1 \parallel P_2$ is a derived form for $P_1 \underset{\emptyset}{\bowtie} P_2$.

Because we are interested in transient behaviour we use the deadlocked process *Stop*.

When working with large numbers of jobs and servers, we write $P[n]$ to denote an array of n copies of P executing in parallel.

$$P[5] \equiv (P \parallel P \parallel P \parallel P \parallel P)$$

Background: Deterministic processes

A process is a deterministic process if knowledge of its values up to and including time t allows us to unambiguously predict its value at any infinitesimally later time $t + dt$.

Background: ODEs are memoryless deterministic processes

A set of ordinary differential equations defines a memoryless deterministic process.

$$\begin{aligned} \mathbf{X}(t+dt) &= \mathbf{X}(t) + f(\mathbf{X}(t), t)dt \\ \frac{d\mathbf{X}}{dt} &= f(\mathbf{X}, t) \end{aligned}$$

Background: Stochastic processes

A process is a stochastic process if knowledge of its values up to and including time t allows us to probabilistically predict its value at any infinitesimally later time $t + dt$.

Background: Stochastic processes

A process is a stochastic process if knowledge of its values up to and including time t allows us to probabilistically predict its value at any infinitesimally later time $t + dt$.

Stochastic processes subsume deterministic processes.

Outline

Modelling with quantified process algebras

Tiny example

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

$System \stackrel{def}{=} (P_1 \parallel P_1)$

Modelling with quantified process algebras

Tiny example

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

$System \stackrel{def}{=} (P_1 \parallel P_1)$

This example defines a system with nine reachable states:

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

The transitions between states have quantified duration r which can be evaluated against a CTMC or ODE interpretation.

Analysis based on Continuous-time Markov Chains

Tiny example

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

$System \stackrel{def}{=} (P_1 \parallel P_1)$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 0$:

1. 1.0000
2. 0.0000
3. 0.0000
4. 0.0000
5. 0.0000
6. 0.0000
7. 0.0000
8. 0.0000
9. 0.0000

Analysis based on Continuous-time Markov Chains

Tiny example

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

$System \stackrel{def}{=} (P_1 \parallel P_1)$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 1$:

1. 0.1642
2. 0.1567
3. 0.0842
4. 0.1567
5. 0.1496
6. 0.0804
7. 0.0842
8. 0.0804
9. 0.0432

Analysis based on Continuous-time Markov Chains

Tiny example

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

$System \stackrel{def}{=} (P_1 \parallel P_1)$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 2$:

1. 0.1056
2. 0.1159
3. 0.1034
4. 0.1159
5. 0.1272
6. 0.1135
7. 0.1034
8. 0.1135
9. 0.1012

Analysis based on Continuous-time Markov Chains

Tiny example

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

$System \stackrel{def}{=} (P_1 \parallel P_1)$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 3$:

1. 0.1082
2. 0.1106
3. 0.1100
4. 0.1106
5. 0.1132
6. 0.1125
7. 0.1100
8. 0.1125
9. 0.1119

Analysis based on Continuous-time Markov Chains

Tiny example

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

$System \stackrel{def}{=} (P_1 \parallel P_1)$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 4$:

1. 0.1106
2. 0.1108
3. 0.1111
4. 0.1108
5. 0.1110
6. 0.1113
7. 0.1111
8. 0.1113
9. 0.1116

Analysis based on Continuous-time Markov Chains

Tiny example

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

$System \stackrel{def}{=} (P_1 \parallel P_1)$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 5$:

1. 0.1111
2. 0.1110
3. 0.1111
4. 0.1110
5. 0.1110
6. 0.1111
7. 0.1111
8. 0.1111
9. 0.1111

Analysis based on Continuous-time Markov Chains

Tiny example

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

$System \stackrel{def}{=} (P_1 \parallel P_1)$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 6$:

1. 0.1111
2. 0.1111
3. 0.1111
4. 0.1111
5. 0.1110
6. 0.1111
7. 0.1111
8. 0.1111
9. 0.1111

Analysis based on Continuous-time Markov Chains

Tiny example

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

$System \stackrel{def}{=} (P_1 \parallel P_1)$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 7$:

1. 0.1111
2. 0.1111
3. 0.1111
4. 0.1111
5. 0.1111
6. 0.1111
7. 0.1111
8. 0.1111
9. 0.1111

Analysis based on Ordinary Differential Equations

Tiny example

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

$System \stackrel{def}{=} (P_1 \parallel P_1)$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 0$:

P_1	2.0000
P_2	0.0000
P_3	0.0000

Analysis based on Ordinary Differential Equations

Tiny example

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

$System \stackrel{def}{=} (P_1 \parallel P_1)$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 1$:

P_1	0.8121
P_2	0.7734
P_3	0.4144

Analysis based on Ordinary Differential Equations

Tiny example

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

$System \stackrel{def}{=} (P_1 \parallel P_1)$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 2$:	P_1	0.6490
	P_2	0.7051
	P_3	0.6457

Analysis based on Ordinary Differential Equations

Tiny example

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

$System \stackrel{def}{=} (P_1 \parallel P_1)$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 3$:

P_1	0.6587
P_2	0.6719
P_3	0.6692

Analysis based on Ordinary Differential Equations

Tiny example

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

$System \stackrel{def}{=} (P_1 \parallel P_1)$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 4$:	P_1	0.6648
	P_2	0.6665
	P_3	0.6685

Analysis based on Ordinary Differential Equations

Tiny example

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

$System \stackrel{def}{=} (P_1 \parallel P_1)$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 5$:

P_1	0.6666
P_2	0.6663
P_3	0.6669

Analysis based on Ordinary Differential Equations

Tiny example

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

$System \stackrel{def}{=} (P_1 \parallel P_1)$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 6$:

P_1	0.6666
P_2	0.6666
P_3	0.6666

Analysis based on Ordinary Differential Equations

Tiny example

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

$System \stackrel{def}{=} (P_1 \parallel P_1)$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 7$:	P_1	0.6666
	P_2	0.6666
	P_3	0.6666

Analysis based on Ordinary Differential Equations

Slightly larger example

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

$System \stackrel{def}{=} (P_1 \parallel P_1 \parallel P_1)$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 0$:

P_1	3.0000
P_2	0.0000
P_3	0.0000

Analysis based on Ordinary Differential Equations

Slightly larger example

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

$System \stackrel{def}{=} (P_1 \parallel P_1 \parallel P_1)$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 1$:	P_1	1.1782
	P_2	1.1628
	P_3	0.6590

Analysis based on Ordinary Differential Equations

Slightly larger example

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

$System \stackrel{def}{=} (P_1 \parallel P_1 \parallel P_1)$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 2$:	P_1	0.9766
	P_2	1.0754
	P_3	0.9479

Analysis based on Ordinary Differential Equations

Slightly larger example

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

$System \stackrel{def}{=} (P_1 \parallel P_1 \parallel P_1)$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 3$:	P_1	0.9838
	P_2	1.0142
	P_3	1.0020

Analysis based on Ordinary Differential Equations

Slightly larger example

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

$System \stackrel{def}{=} (P_1 \parallel P_1 \parallel P_1)$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 4$:	P_1	0.9981
	P_2	0.9995
	P_3	1.0023

Analysis based on Ordinary Differential Equations

Slightly larger example

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

$System \stackrel{def}{=} (P_1 \parallel P_1 \parallel P_1)$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 5$:

P_1	1.0001
P_2	0.9996
P_3	1.0003

Analysis based on Ordinary Differential Equations

Slightly larger example

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

$System \stackrel{def}{=} (P_1 \parallel P_1 \parallel P_1)$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 6$:

P_1	1.0001
P_2	0.9999
P_3	1.0000

Analysis based on Ordinary Differential Equations

Slightly larger example

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

$System \stackrel{def}{=} (P_1 \parallel P_1 \parallel P_1)$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 7$:	P_1	1.0000
	P_2	0.9999
	P_3	0.9999

Analysis based on Ordinary Differential Equations

Slightly larger example

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

$System \stackrel{def}{=} (P_1 \parallel P_1 \parallel P_1)$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 8$:	P_1	1.0000
	P_2	1.0000
	P_3	1.0000

What just happened?

An ODE specifies how the value of some continuous variable varies over continuous time. For example, the temperature in a container may be modelled by an ODE describing how the temperature will change dependent on the current temperature and pressure. The pressure can be similarly modelled and the equations together form a system of ODEs describing the state of the container.

What just happened?

In a PEPA model the state at any current time is the local derivative or state of each component of the model. When we have large numbers of repeated components it can make sense to represent each component type as a continuous variable, and the state of the model as a whole as the set of such variables. The evolution of each such variable can then be described by an ODE.

Isn't this just the Chapman-Kolmogorov equations?

It is possible to perform transient analysis of a continuous-time Markov chain by solving the Chapman-Kolmogorov differential equations:

$$\frac{d\pi(t)}{dt} = \pi(t)Q$$

[Stewart, 1994]

Isn't this just the Chapman-Kolmogorov equations?

It is possible to perform transient analysis of a continuous-time Markov chain by solving the Chapman-Kolmogorov differential equations:

$$\frac{d\pi(t)}{dt} = \pi(t)Q$$

[Stewart, 1994]

That's not what we're doing. We go directly to ODEs.

What's the value proposition?

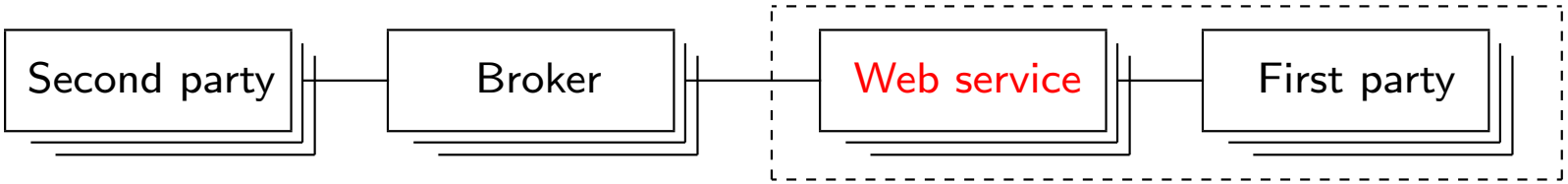
- The bottleneck for Markovian modelling of systems is the size of the solution vector, which is bounded by the product of the state-space sizes of the processes which are composed in parallel ("state-space explosion").

What's the value proposition?

- The bottleneck for Markovian modelling of systems is the size of the solution vector, which is bounded by the product of the state-space sizes of the processes which are composed in parallel ("state-space explosion").
- The size of the solution vector for the system of ODEs may be exponentially smaller.

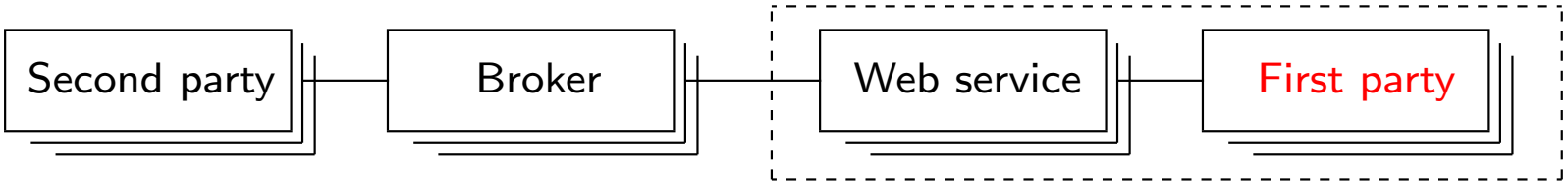
Outline

Example: Secure Web Service use



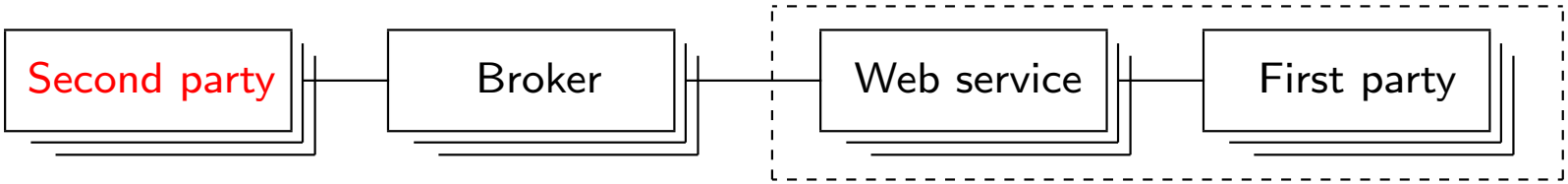
- The example which we consider is a Web service which has two types of clients:
  - first party application clients which access the web service across a secure intranet, and
  - second party browser clients which access the Web service across the Internet.
- Second party clients route their service requests via trusted brokers.

Example: Secure Web Service use



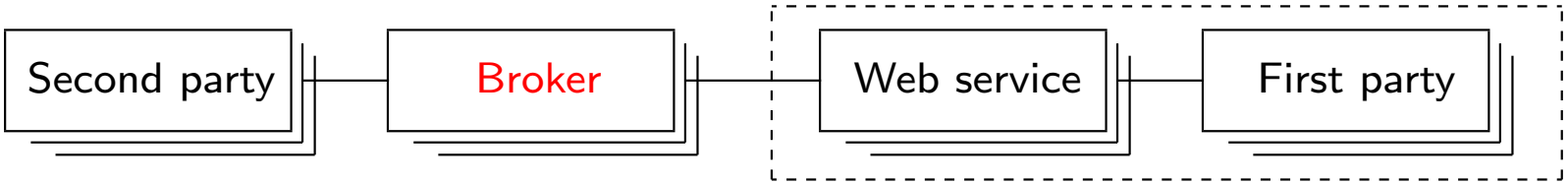
- The example which we consider is a Web service which has two types of clients:
  - first party application clients which access the web service across a secure intranet, and
  - second party browser clients which access the Web service across the Internet.
- Second party clients route their service requests via trusted brokers.

Example: Secure Web Service use



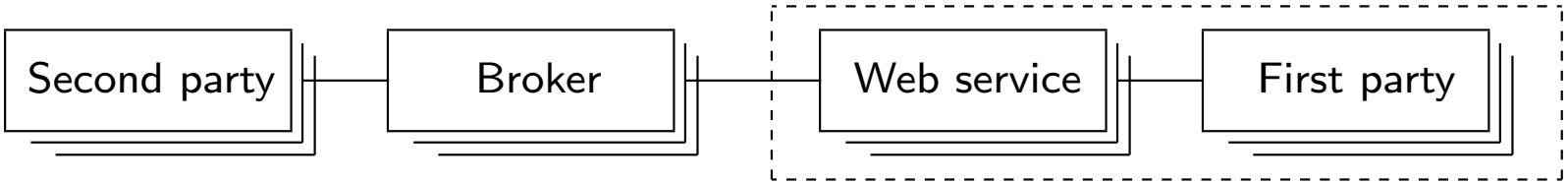
- The example which we consider is a Web service which has two types of clients:
  - first party application clients which access the web service across a secure intranet, and
  - second party browser clients which access the Web service across the Internet.
- Second party clients route their service requests via trusted brokers.

Example: Secure Web Service use



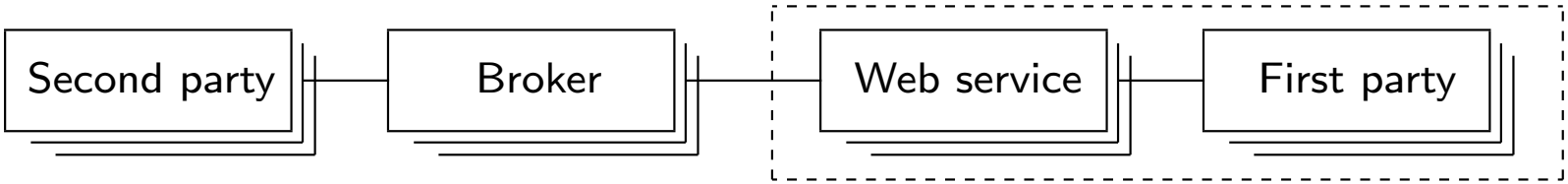
- The example which we consider is a Web service which has two types of clients:
  - first party application clients which access the web service across a secure intranet, and
  - second party browser clients which access the Web service across the Internet.
- Second party clients route their service requests via trusted brokers.

Scalability and replication



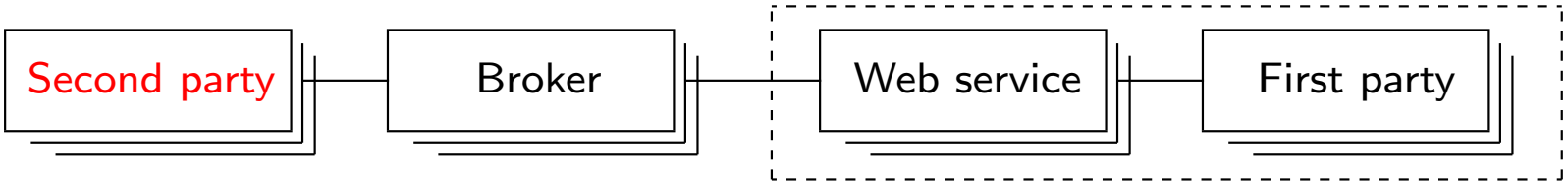
- To ensure scalability the Web service is replicated across multiple hosts.
- Multiple brokers are available.
- There are numerous first party clients behind the firewall using the service via remote method invocations across the secure intranet.
- There are numerous second party clients outside the firewall.

Security and use of encryption



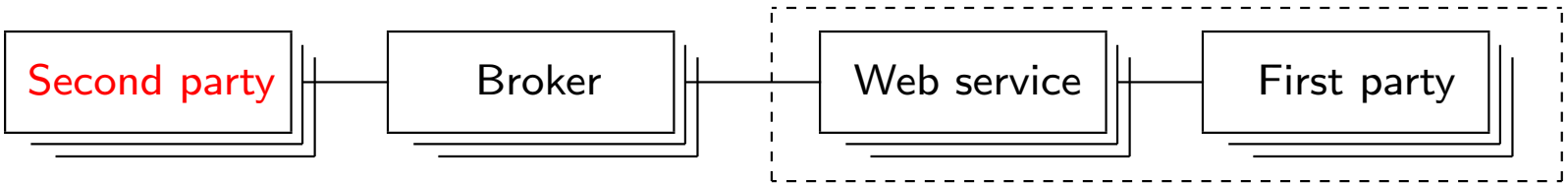
- Second party clients need to use encryption to ensure authenticity and confidentiality. First party clients do not.
- Brokers add decryption and encryption steps to build end-to-end security from point-to-point security.
  - When processing a request from a second party client brokers decrypt the request before re-encrypting it for the Web service.
  - When the response to a request is returned to the broker it decrypts the response before re-encrypting it for the client.

PEPA model: Second party clients



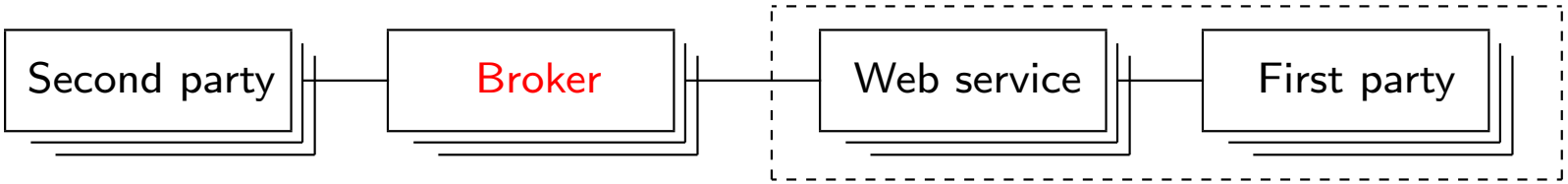
- A second party client composes service requests, encrypts these and sends them to its broker.
- It then waits for a response from the broker.
- The rate at which the first three activities happen is under the control of the client.
- The rate at which responses are produced is determined by the interaction of the broker and the service endpoint.

PEPA model: Second party clients



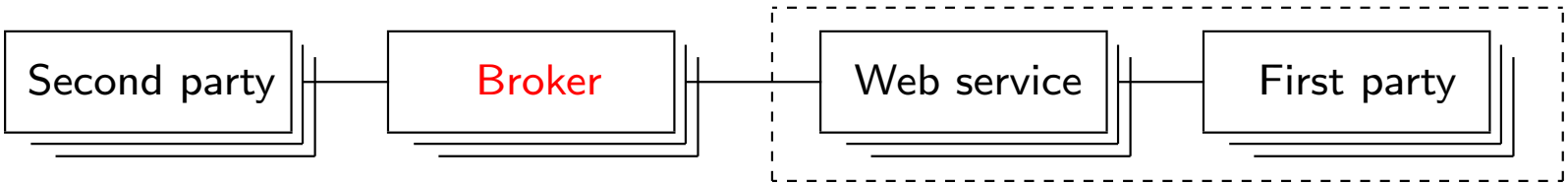
$$
\begin{aligned}
SPC_{idle} &\stackrel{def}{=} (compose_{sp}, r_{sp\_cmp}).SPC_{enc} \\
SPC_{enc} &\stackrel{def}{=} (encrypt_{b}, r_{sp\_encb}).SPC_{sending} \\
SPC_{sending} &\stackrel{def}{=} (request_{b}, r_{sp\_req}).SPC_{waiting} \\
SPC_{waiting} &\stackrel{def}{=} (response_{b}, \top).SPC_{dec} \\
SPC_{dec} &\stackrel{def}{=} (decrypt_{b}, r_{sp\_decb}).SPC_{idle}
\end{aligned}
$$

PEPA model: Brokers



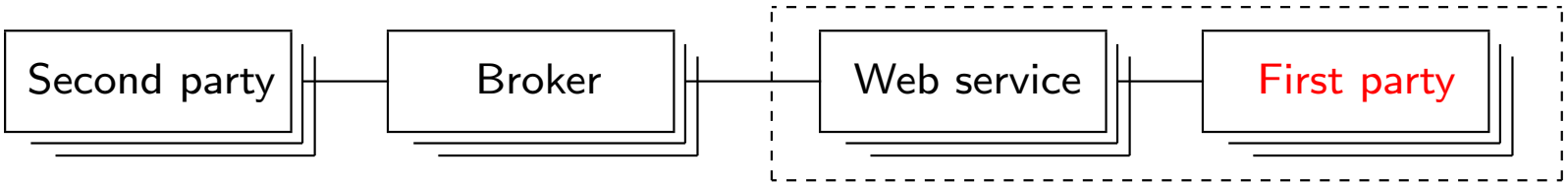
- The broker is inactive until it receives a request.
- It then decrypts the request before re-encrypting it for the Web service to ensure end-to-end security.
- It forwards the request to the Web service and then waits for a response.
- The corresponding decryption and re-encrytion are performed before returning the response to the client.

PEPA model: Brokers



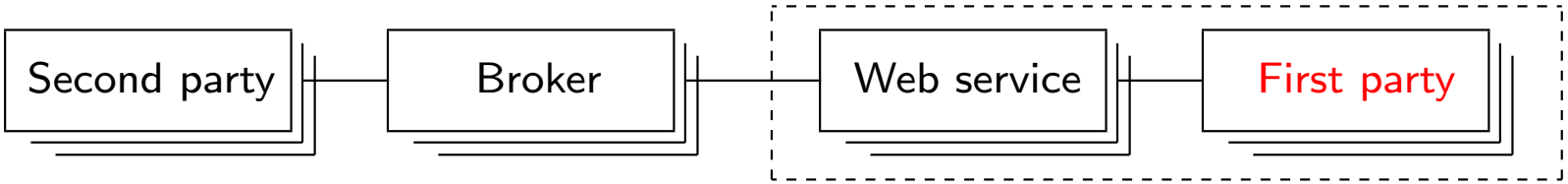
$$
\begin{aligned}
Broker_{idle} &\stackrel{def}{=} (request_b, \top).Broker_{dec\_input} \\
Broker_{dec\_input} &\stackrel{def}{=} (decrypt_{sp}, r_{b\_dec\_sp}).Broker_{enc\_input} \\
Broker_{enc\_input} &\stackrel{def}{=} (encrypt_{ws}, r_{b\_enc\_ws}).Broker_{sending} \\
Broker_{sending} &\stackrel{def}{=} (request_{ws}, r_{b\_req}).Broker_{waiting} \\
Broker_{waiting} &\stackrel{def}{=} (response_{ws}, \top).Broker_{dec\_resp} \\
Broker_{dec\_resp} &\stackrel{def}{=} (decrypt_{ws}, r_{b\_dec\_ws}).Broker_{enc\_resp} \\
Broker_{enc\_resp} &\stackrel{def}{=} (encrypt_{sp}, r_{b\_enc\_sp}).Broker_{replying} \\
Broker_{replying} &\stackrel{def}{=} (response_b, r_{b\_resp}).Broker_{idle}
\end{aligned}
$$

PEPA model: First party clients



- The lifetime of a first party client mirrors that of a second party client except that encryption need not be used when all of the communication is conducted across a secure intranet.
- Also the service may be invoked by a remote method invocation to the host machine instead of via HTTP.
- Thus the first party client experiences the Web service as a blocking remote method invocation.

PEPA model: First party clients

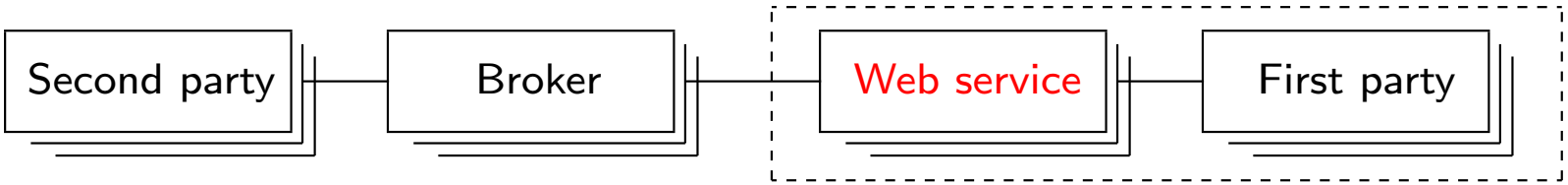


$$FPC_{idle} \stackrel{def}{=} (compose_{fp}, r_{fp\_cmp}).FPC_{calling}$$

$$FPC_{calling} \stackrel{def}{=} (invoke_{ws}, r_{fp\_inv}).FPC_{blocked}$$

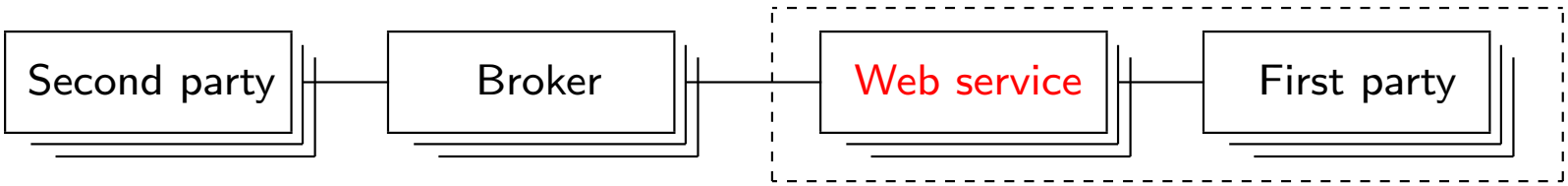
$$FPC_{blocked} \stackrel{def}{=} (result_{ws}, \top).FPC_{idle}$$

PEPA model: Web service



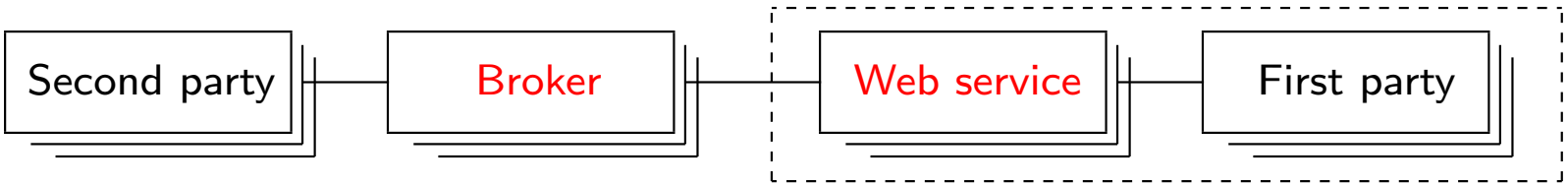
- There are two ways in which the service is executed, leading to a choice in the process algebra model taking the service process into one or other of its two modes of execution.
- In either case, the duration of the execution of the service itself is unchanged.
- The difference is only in whether encryption is needed and whether the result is delivered via HTTP or not.

PEPA model: Web service



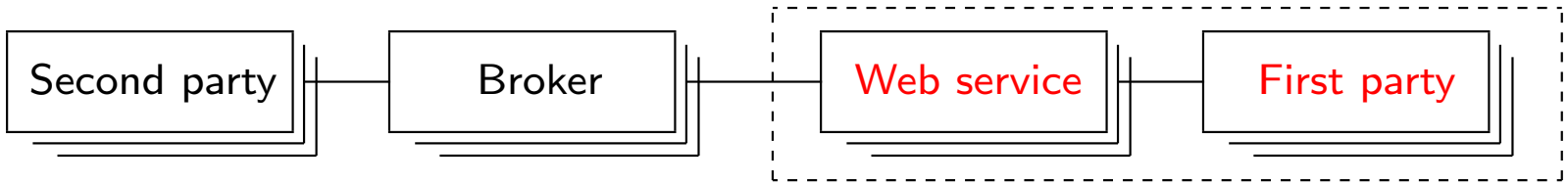
$$
\begin{aligned}
WS_{idle} &\stackrel{def}{=} (request_{ws}, \top).WS_{decoding} \\
&\quad + (invoke_{ws}, \top).WS_{method} \\
WS_{decoding} &\stackrel{def}{=} (decryptReq_{ws}, r_{ws\_dec\_b}).WS_{execution} \\
WS_{execution} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{securing} \\
WS_{securing} &\stackrel{def}{=} (encryptResp_{ws}, r_{ws\_enc\_b}).WS_{responding} \\
WS_{responding} &\stackrel{def}{=} (response_{ws}, r_{ws\_resp\_b}).WS_{idle} \\
WS_{method} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{returning} \\
WS_{returning} &\stackrel{def}{=} (result_{ws}, r_{ws\_res}).WS_{idle}
\end{aligned}
$$

PEPA model: Web service



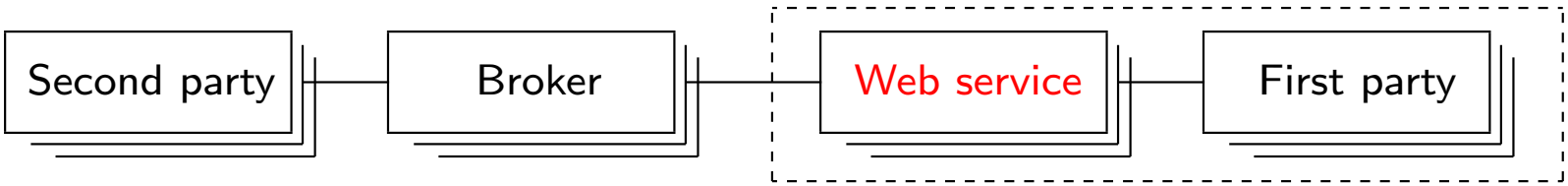
$$
\begin{aligned}
WS_{idle} &\stackrel{def}{=} (request_{ws}, \top).WS_{decoding} \\
&\quad + (invoke_{ws}, \top).WS_{method} \\
WS_{decoding} &\stackrel{def}{=} (decryptReq_{ws}, r_{ws\_dec\_b}).WS_{execution} \\
WS_{execution} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{securing} \\
WS_{securing} &\stackrel{def}{=} (encryptResp_{ws}, r_{ws\_enc\_b}).WS_{responding} \\
WS_{responding} &\stackrel{def}{=} (response_{ws}, r_{ws\_resp\_b}).WS_{idle} \\
WS_{method} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{returning} \\
WS_{returning} &\stackrel{def}{=} (result_{ws}, r_{ws\_res}).WS_{idle}
\end{aligned}
$$

PEPA model: Web service



$$
\begin{aligned}
WS_{idle} &\stackrel{def}{=} (request_{ws}, \top).WS_{decoding} \\
&\quad + (invoke_{ws}, \top).WS_{method} \\
WS_{decoding} &\stackrel{def}{=} (decryptReq_{ws}, r_{ws\_dec\_b}).WS_{execution} \\
WS_{execution} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{securing} \\
WS_{securing} &\stackrel{def}{=} (encryptResp_{ws}, r_{ws\_enc\_b}).WS_{responding} \\
WS_{responding} &\stackrel{def}{=} (response_{ws}, r_{ws\_resp\_b}).WS_{idle} \\
WS_{method} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{returning} \\
WS_{returning} &\stackrel{def}{=} (result_{ws}, r_{ws\_res}).WS_{idle}
\end{aligned}
$$

PEPA model: Web service



$$
\begin{aligned}
WS_{idle} &\stackrel{def}{=} (request_{ws}, \top).WS_{decoding} \\
&\quad + (invoke_{ws}, \top).WS_{method} \\
WS_{decoding} &\stackrel{def}{=} (decryptReq_{ws}, r_{ws\_dec\_b}).WS_{execution} \\
WS_{execution} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{securing} \\
WS_{securing} &\stackrel{def}{=} (encryptResp_{ws}, r_{ws\_enc\_b}).WS_{responding} \\
WS_{responding} &\stackrel{def}{=} (response_{ws}, r_{ws\_resp\_b}).WS_{idle} \\
WS_{method} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{returning} \\
WS_{returning} &\stackrel{def}{=} (result_{ws}, r_{ws\_res}).WS_{idle}
\end{aligned}
$$

PEPA model: System composition

In the initial state of the system model we represent each of the four component types being initially in their idle state.

$$System \stackrel{def}{=} (SPC_{idle} \underset{\mathcal{K}}{\bowtie} Broker_{idle}) \underset{\mathcal{L}}{\bowtie} (WS_{idle} \underset{\mathcal{M}}{\bowtie} FPC_{idle})$$

$$\begin{array}{lrcl} \text{where} & \mathcal{K} & = & \{ request_b, response_b \} \\ & \mathcal{L} & = & \{ request_{ws}, response_{ws} \} \\ & \mathcal{M} & = & \{ invoke_{ws}, result_{ws} \} \end{array}$$

PEPA model: System composition

In the initial state of the system model we represent each of the four component types being initially in their idle state.

$$System \stackrel{def}{=} (SPC_{idle} \underset{\mathcal{K}}{\bowtie} Broker_{idle}) \underset{\mathcal{L}}{\bowtie} (WS_{idle} \underset{\mathcal{M}}{\bowtie} FPC_{idle})$$

$$\begin{array}{lrcl} \text{where} & \mathcal{K} & = & \{ request_b, response_b \} \\ & \mathcal{L} & = & \{ request_{ws}, response_{ws} \} \\ & \mathcal{M} & = & \{ invoke_{ws}, result_{ws} \} \end{array}$$

This model represents the smallest possible instance of the system, where there is one instance of each component type. We evaluate the system as the number of clients, brokers, and copies of the service increase.

Cost of analysis

- Performance models admit many different types of analysis. Some have lower evaluation cost, but are less informative, such as steady-state analysis. Others have higher evaluation cost, but are more informative, such as transient analysis.
- We compare ODE-based evaluation against other techniques which could be used to analyse the model.
- We compare against steady-state and transient analysis as implemented by the PRISM probabilistic model-checker, which provides PEPA as one of its input languages. We also compare against Monte Carlo Markov Chain simulation.

Comparison of analysis types

- We report only a single run of the transient analysis and simulation. In practice, due to the stochastic nature of the analyses, these would need to be re-run multiple times to produce results comparable to the ODE-based analysis.
- Moreover, note that the number of ODEs is constant regardless of the number of components in the system, whilst the state space grows dramatically.

Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81

Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81

Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81
3	3	3	3	1,130,496	161,296	172.48	255.48	588.80	2.48	2.83

Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81
3	3	3	3	1,130,496	161,296	172.48	255.48	588.80	2.48	2.83
4	4	4	4	>234M	–	–	–	–	2.44	2.85

Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81
3	3	3	3	1,130,496	161,296	172.48	255.48	588.80	2.48	2.83
4	4	4	4	>234M	–	–	–	–	2.44	2.85
100	100	100	100	–	–	–	–	–	2.78	2.78

Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81
3	3	3	3	1,130,496	161,296	172.48	255.48	588.80	2.48	2.83
4	4	4	4	>234M	–	–	–	–	2.44	2.85
100	100	100	100	–	–	–	–	–	2.78	2.78
1000	100	500	1000	–	–	–	–	–	3.72	2.77

Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81
3	3	3	3	1,130,496	161,296	172.48	255.48	588.80	2.48	2.83
4	4	4	4	>234M	–	–	–	–	2.44	2.85
100	100	100	100	–	–	–	–	–	2.78	2.78
1000	100	500	1000	–	–	–	–	–	3.72	2.77
1000	1000	1000	1000	–	–	–	–	–	5.44	2.77

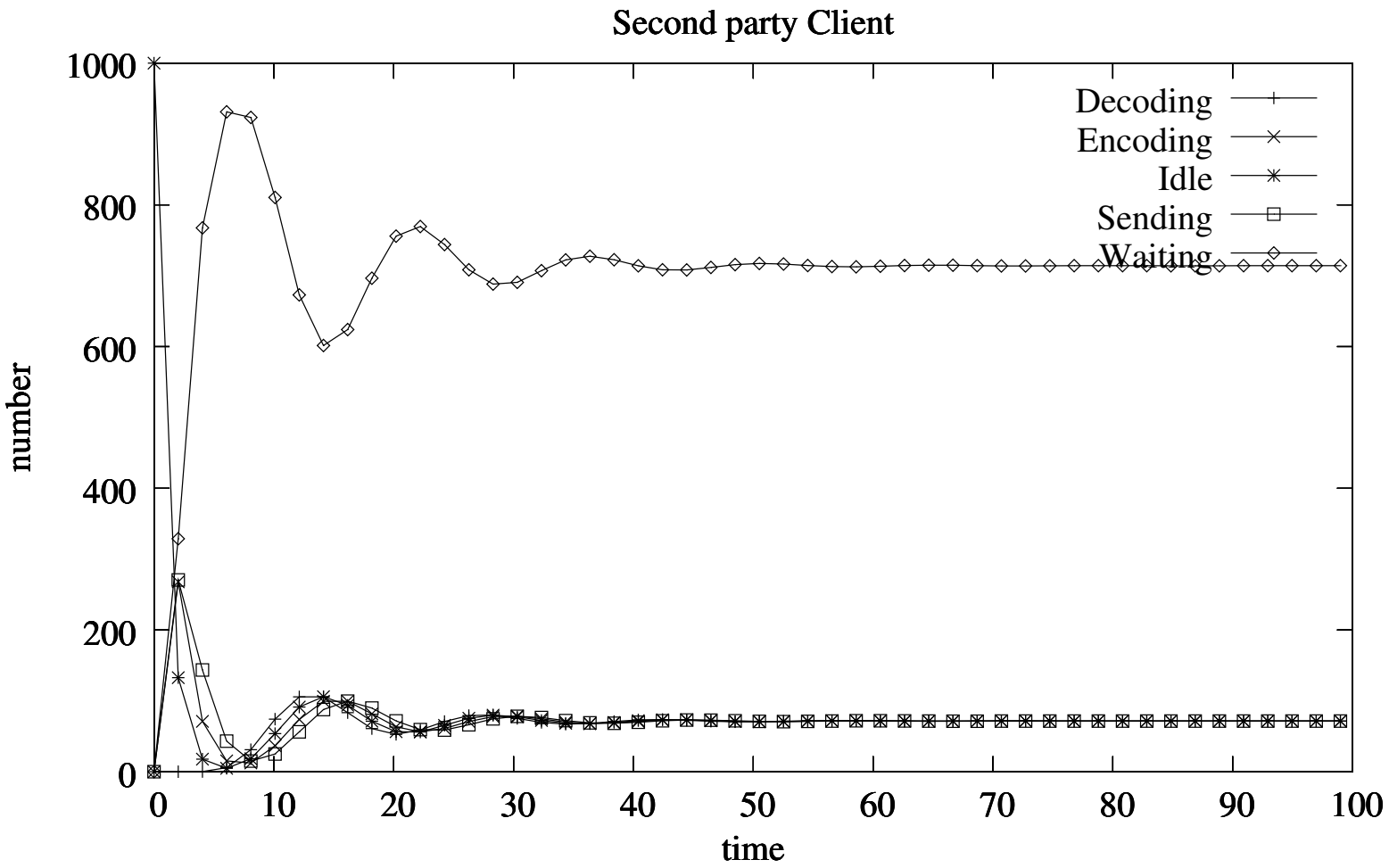
Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81
3	3	3	3	1,130,496	161,296	172.48	255.48	588.80	2.48	2.83
4	4	4	4	>234M	–	–	–	–	2.44	2.85
100	100	100	100	–	–	–	–	–	2.78	2.78
1000	100	500	1000	–	–	–	–	–	3.72	2.77
1000	1000	1000	1000	–	–	–	–	–	5.44	2.77

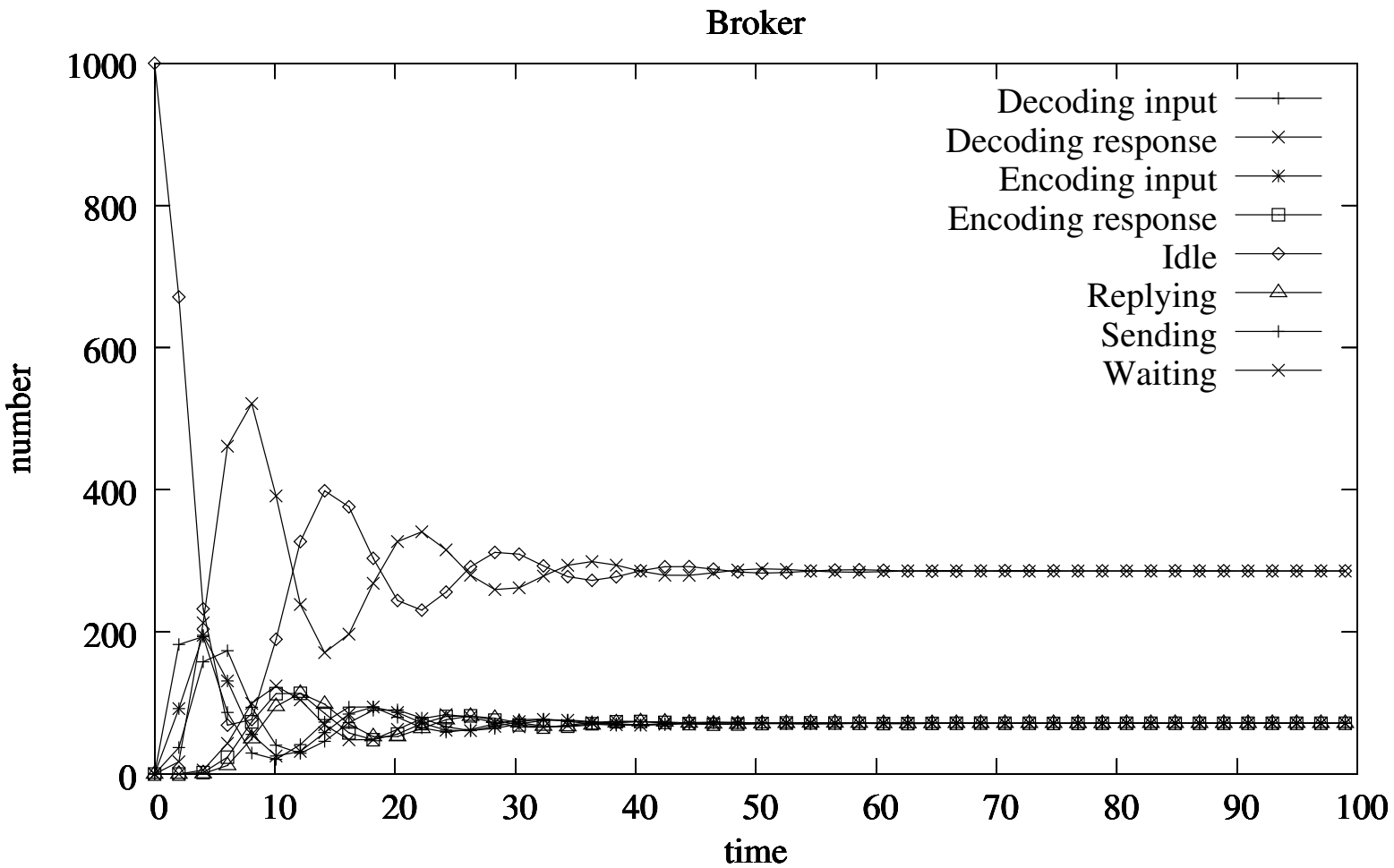
Time series analysis via ODEs

- We now consider the results from our solution of the PEPA Web Service model as a system of ODEs with the number of clients of both kinds, brokers, and web service instances all 1000.
- The results as presented from our ODE integrator are time-series plots of the number of each type of component behaviour as a function of time.
- The graphs show fluctuations in the numbers of components with respect to time from $t = 0$ to $t = 100$ for estimated values of rates for the activities of the system. We can observe an initial flurry of activity until the system stabilises into its steady-state equilibrium at time (around) $t = 50$.

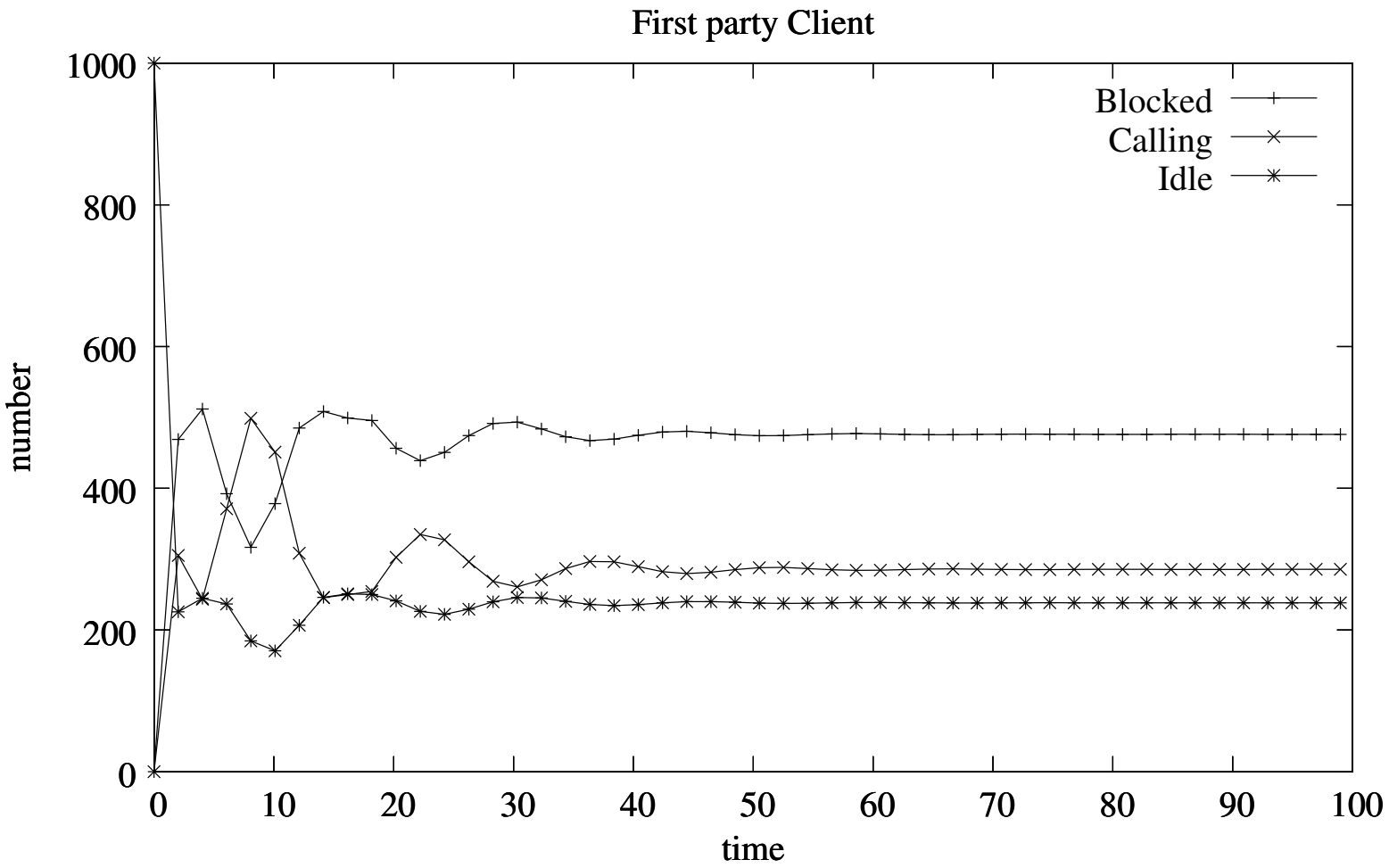
Second party clients



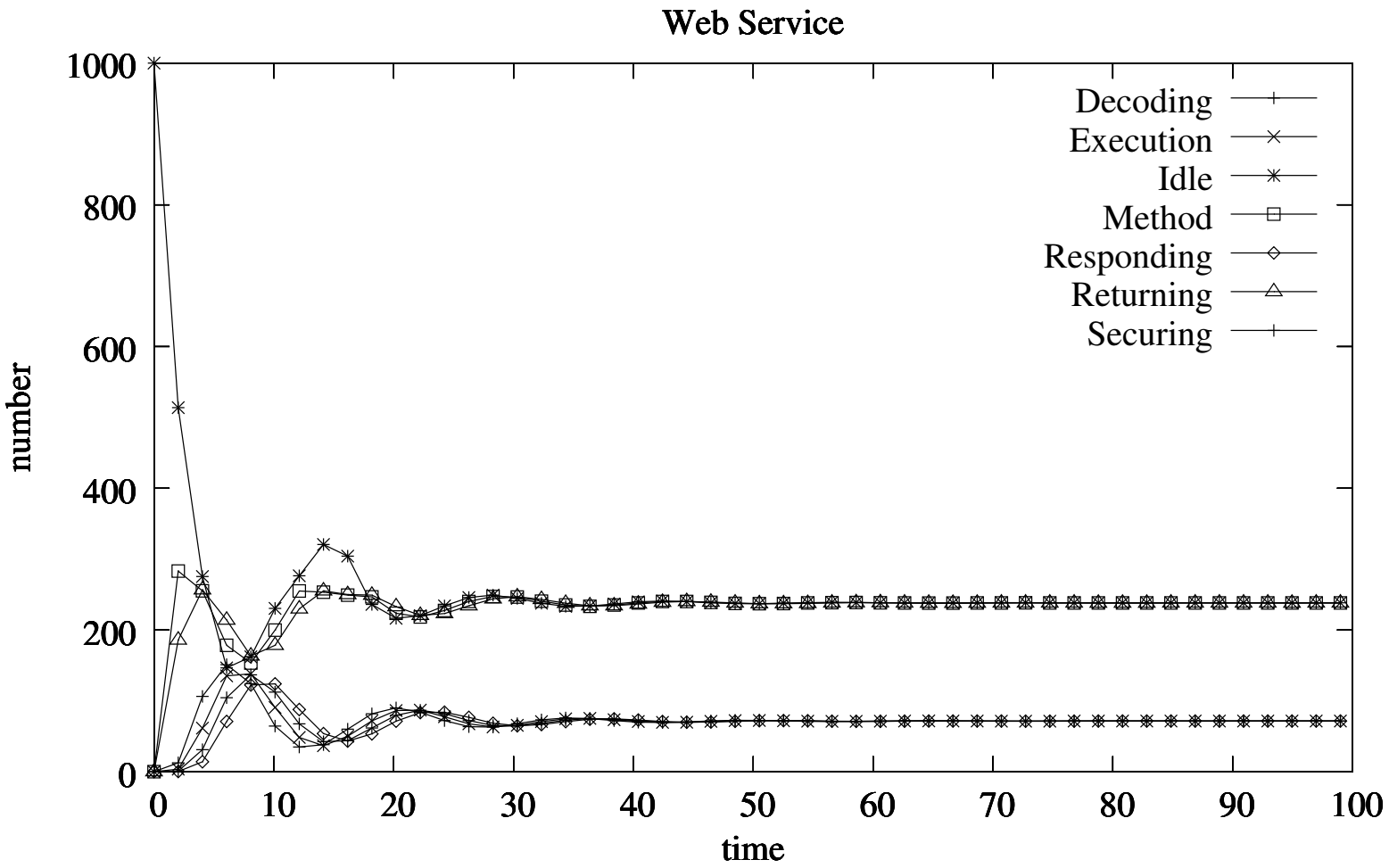
Brokers


Broker
number
time
0
200
400
600
800
1000
0
10
20
30
40
50
60
70
80
90
100
Decoding input
Decoding response
Encoding input
Encoding response
Idle
Replying
Sending
Waiting

First party clients



Web service


Web Service
1000
800
600
400
200
0
number
0
10
20
30
40
50
60
70
80
90
100
time
Decoding
Execution
Idle
Method
Responding
Returning
Securing

Outline

Commentary and comparison

- Previous performance modelling with PEPA used continuous-time Markov chains (CTMCs). These admit *steady-state* and *transient* analysis (by solving the CTMC).

Commentary and comparison

- Previous performance modelling with PEPA used continuous-time Markov chains (CTMCs). These admit *steady-state* and *transient* analysis (by solving the CTMC).
- Steady-state is cheaper but less informative. Transient is more informative but more expensive.

Commentary and comparison

- Previous performance modelling with PEPA used continuous-time Markov chains (CTMCs). These admit *steady-state* and *transient* analysis (by solving the CTMC).
- Steady-state is cheaper but less informative. Transient is more informative but more expensive.
- Major drawback: state-space explosion. Generating the state-space is slow. Solving the CTMC is slow.

Commentary and comparison

- Previous performance modelling with PEPA used continuous-time Markov chains (CTMCs). These admit *steady-state* and *transient* analysis (by solving the CTMC).
- Steady-state is cheaper but less informative. Transient is more informative but more expensive.
- Major drawback: state-space explosion. Generating the state-space is slow. Solving the CTMC is slow.
- In practice effective only to systems of size 10^6 states, even when using clever storage representations.

Commentary and comparison

- Mapping PEPA to ODEs admits *course-of-values* analysis by solving the ODE (akin to transient analysis).

Commentary and comparison

- Mapping PEPA to ODEs admits *course-of-values* analysis by solving the ODE (akin to transient analysis).
- Major benefit: avoids state-space generation entirely.

Commentary and comparison

- Mapping PEPA to ODEs admits *course-of-values* analysis by solving the ODE (akin to transient analysis).
- Major benefit: avoids state-space generation entirely.
- Major benefit: ODE solving is effective in practice, leaning towards suitability for interactive experimentation. Good for modellers, gaining more insights into the system behaviour.

Commentary and comparison

- Mapping PEPA to ODEs admits *course-of-values* analysis by solving the ODE (akin to transient analysis).
- Major benefit: avoids state-space generation entirely.
- Major benefit: ODE solving is effective in practice, leaning towards suitability for interactive experimentation. Good for modellers, gaining more insights into the system behaviour.
- Effective for systems of size 10^{10^6} states and beyond.

Discussion: process algebras and ODEs

- Models in the PEPA stochastic process algebra are concise, and in direct style they generate a system of ODEs the number of which is linear in the number of distinct component types in the PEPA model.

Discussion: process algebras and ODEs

- Models in the PEPA stochastic process algebra are concise, and in direct style they generate a system of ODEs the number of which is linear in the number of distinct component types in the PEPA model.
- Thus there is no hidden cost in the use of the high-level language but there are many advantages.

Discussion: process algebras and ODEs

- Models in the PEPA stochastic process algebra are concise, and in direct style they generate a system of ODEs the number of which is linear in the number of distinct component types in the PEPA model.
- Thus there is no hidden cost in the use of the high-level language but there are many advantages.
  - PEPA models can be checked for freedom from deadlock, satisfaction of logical properties, or compared using relations such as bisimulation.

Discussion: process algebras and ODEs

- Models in the PEPA stochastic process algebra are concise, and in direct style they generate a system of ODEs the number of which is linear in the number of distinct component types in the PEPA model.
- Thus there is no hidden cost in the use of the high-level language but there are many advantages.
  - PEPA models can be checked for freedom from deadlock, satisfaction of logical properties, or compared using relations such as bisimulation.
  - As a compositional modelling language PEPA components can be re-used in other models, promoting best practice.

False and true concurrency

In the process algebra world, algebras with an interleaving semantics are termed false concurrency. PEPA [Hillston 1994] was the first timed process algebra to have an interleaving semantics allowing it to generate a CTMC. The interleaving semantics gives rise to the state-space explosion problem.

Process algebras without an interleaving semantics are termed true concurrency process algebras. The search for a true concurrency timed process algebra has been a ten-year open problem.

PEPA [Hillston 2005] is the first timed process algebra to have a true concurrency semantics via the mapping to ODEs. The true concurrency semantics avoids the state-space explosion problem.

Stochastic Modelling
Part IV: Variance

Stephen Gilmore
LFCS, University of Edinburgh

GLOBAN Summerschool
Lygby, Denmark

25th August 2006

Review: Problems of scale

- Global computing systems are typically ubiquitous, may exhibit mobility, and are distributed. There are many places where computations can happen, and many replications of services.

Review: Problems of scale

- Global computing systems are typically ubiquitous, may exhibit mobility, and are distributed. There are many places where computations can happen, and many replications of services.
- In many types of stochastic modelling one runs into problems of scale.

Review: Problems of scale

- Global computing systems are typically ubiquitous, may exhibit mobility, and are distributed. There are many places where computations can happen, and many replications of services.
- In many types of stochastic modelling one runs into problems of scale.
- Continuous-Time Markov Chain

Review: Problems of scale

- Global computing systems are typically ubiquitous, may exhibit mobility, and are distributed. There are many places where computations can happen, and many replications of services.
- In many types of stochastic modelling one runs into problems of scale.
- Continuous-Time Markov Chain
  - 6 copies of a process with 10 states $\Rightarrow 10^6$ states

Review: Problems of scale

- Global computing systems are typically ubiquitous, may exhibit mobility, and are distributed. There are many places where computations can happen, and many replications of services.
- In many types of stochastic modelling one runs into problems of scale.
- Continuous-Time Markov Chain
  - 6 copies of a process with 10 states $\Rightarrow 10^6$ states
  - The state-space explosion problem

Review: Markovian methods and differential equations

- Markovian methods allow us to apply a thorough stochastic treatment in system modelling, giving an exact representation of a stochastic process.

Review: Markovian methods and differential equations

- Markovian methods allow us to apply a thorough stochastic treatment in system modelling, giving an exact representation of a stochastic process.
- Differential equation methods allow us to scale this analysis to large problem sizes but rest on the assumption of continuity for component populations.

Modelling challenges: multi-scale systems

- What about systems in which some components are present in large numbers and others are not?

Modelling challenges: multi-scale systems

- What about systems in which some components are present in large numbers and others are not?
- The system will exhibit a high degree of variance.

Modelling challenges: multi-scale systems

- What about systems in which some components are present in large numbers and others are not?
- The system will exhibit a high degree of variance.
- Simulation allows us to investigate this.

Outline

Outline

A simple example: processors and resources

$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, \top).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_1).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, s).Res_0
\end{aligned}
$$

$$
Proc_0[P] \underset{\{task1\}}{\bowtie} Res_0[R]
$$

A simple example: processors and resources

$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, \top).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_1).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, s).Res_0
\end{aligned}
$$

$$
Proc_0[P] \underset{\{task1\}}{\bowtie} Res_0[R]
$$

CTMC interpretation

Processors (P)	Resources (R)	States (2^{P+R})
1	1	4
2	1	8
2	2	16
3	2	32
3	3	64
4	3	128
4	4	256
5	4	512
5	5	1024
6	5	2048
6	6	4096
7	6	8192
7	7	16384
8	7	32768
8	8	65536
9	8	131072
9	9	262144
10	9	524288
10	10	1048576

A simple example: processors and resources

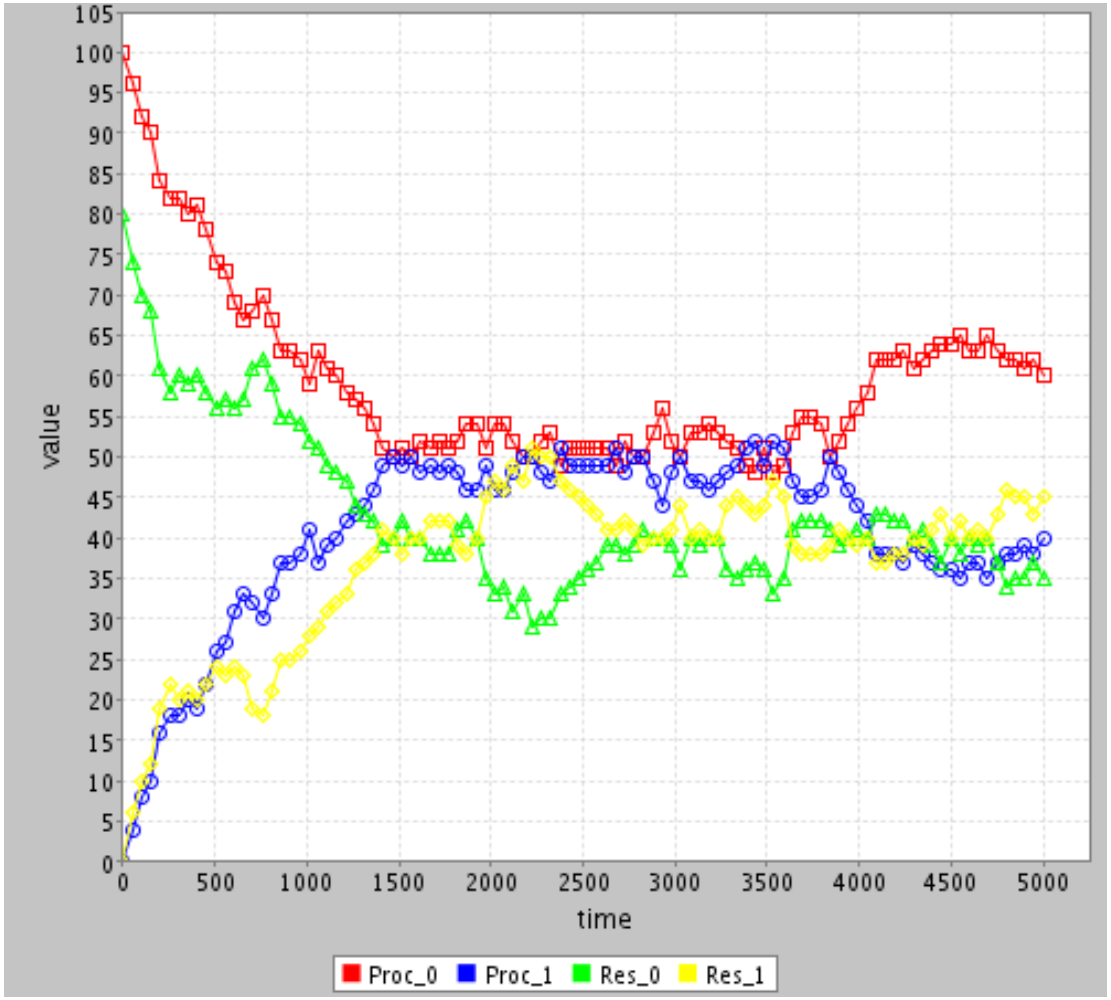
$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, \top).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_1).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, s).Res_0
\end{aligned}
$$

$$
Proc_0[P] \underset{\{task1\}}{\bowtie} Res_0[R]
$$

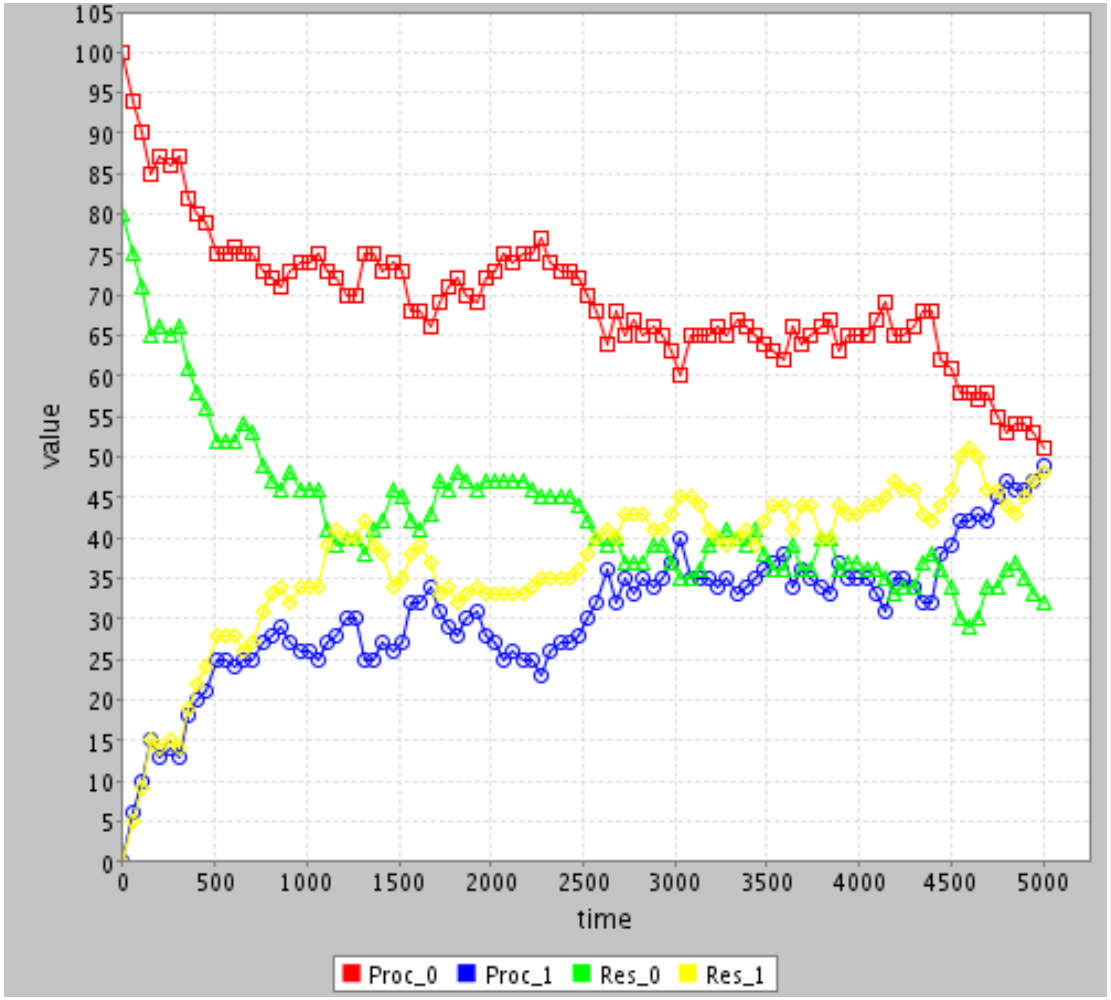
ODE interpretation

$$
\begin{aligned}
\frac{\mathrm{d}Proc_0}{\mathrm{d}t} &= -r_1 \ \min(Proc_0, Res_0) \\
&\quad +r_2 \ Proc_1 \\
\frac{\mathrm{d}Proc_1}{\mathrm{d}t} &= r_1 \ \min(Proc_0, Res_0) \\
&\quad -r_2 \ Proc_1 \\
\frac{\mathrm{d}Res_0}{\mathrm{d}t} &= -r_1 \ \min(Proc_0, Res_0) \\
&\quad +s \ Res_1 \\
\frac{\mathrm{d}Res_1}{\mathrm{d}t} &= r_1 \ \min(Proc_0, Res_0) \\
&\quad -s \ Res_1
\end{aligned}
$$

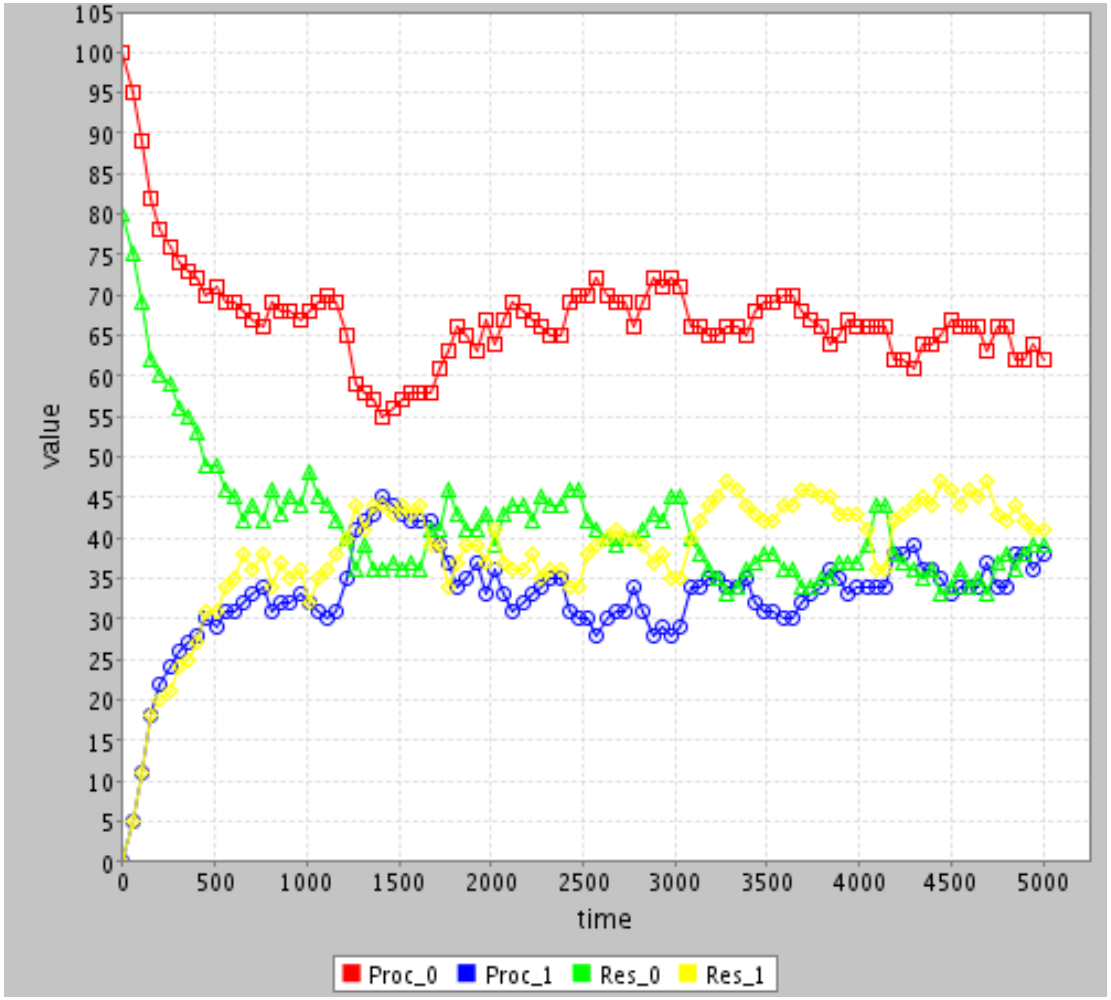
Processors and resources (SSA run A)



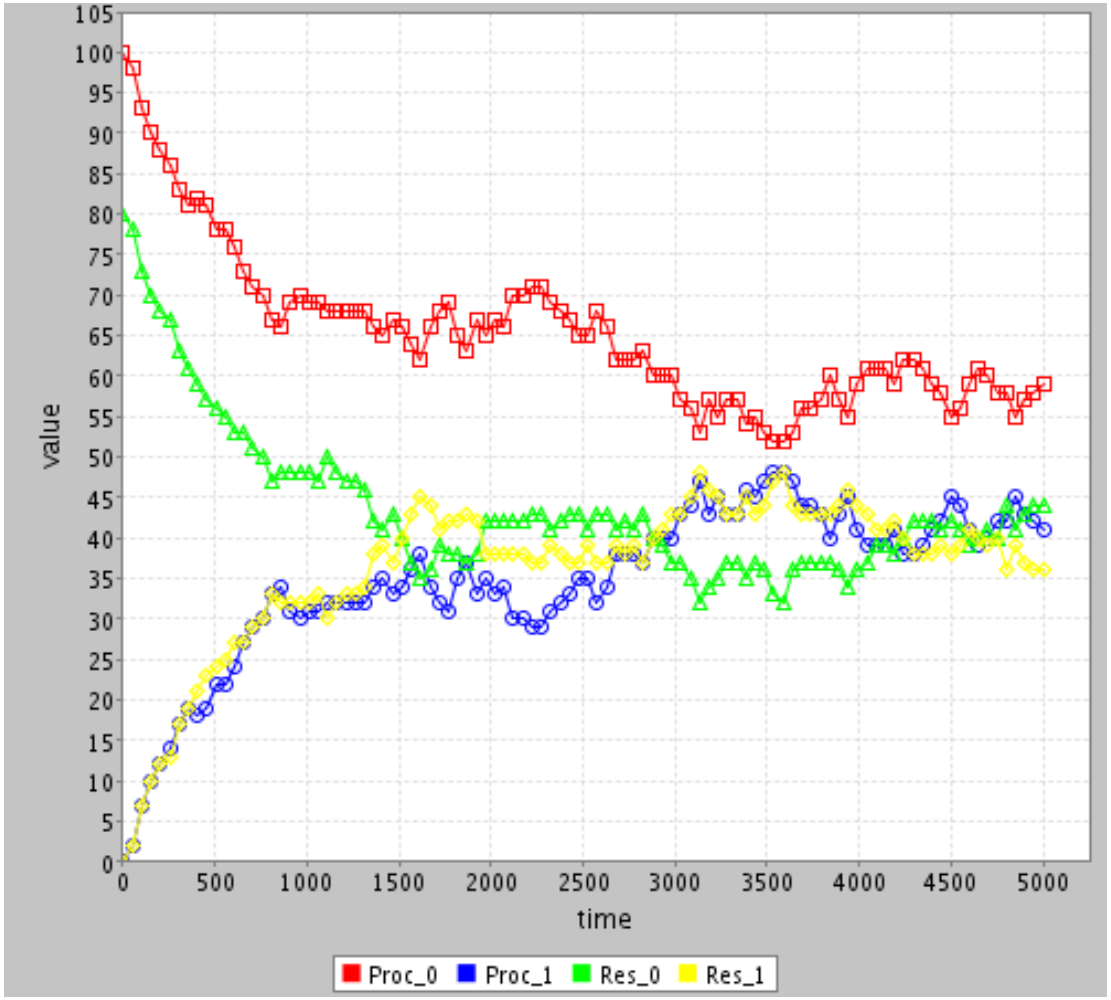
Processors and resources (SSA run B)



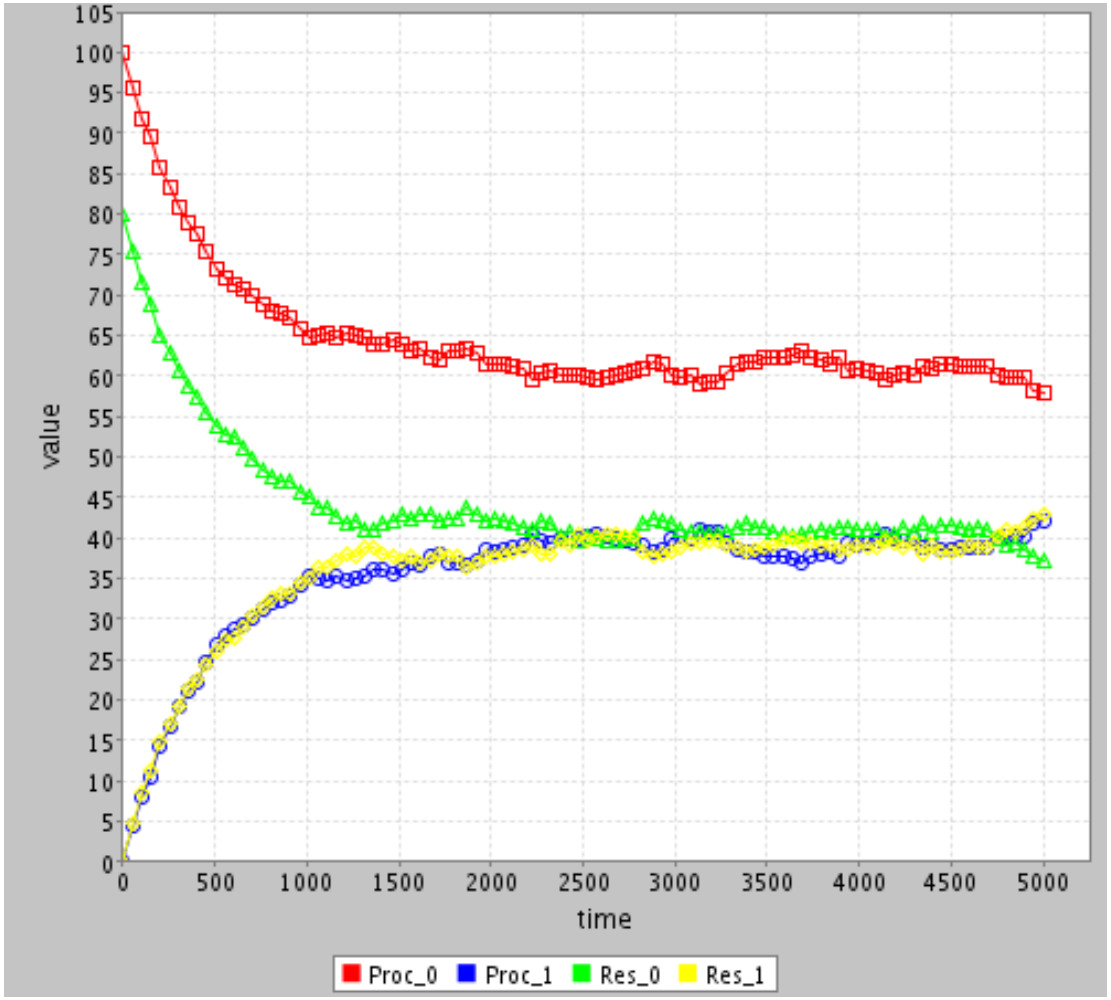
Processors and resources (SSA run C)



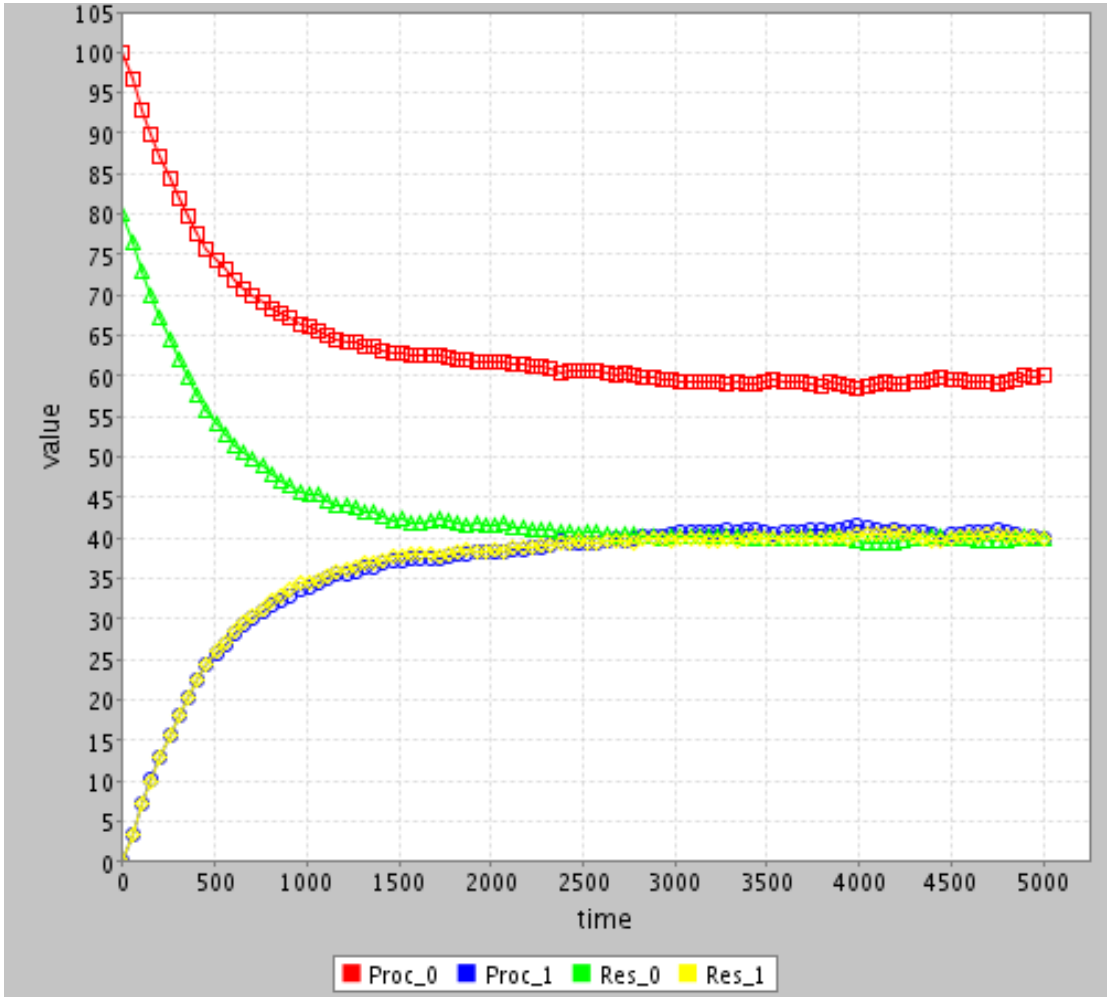
Processors and resources (SSA run D)



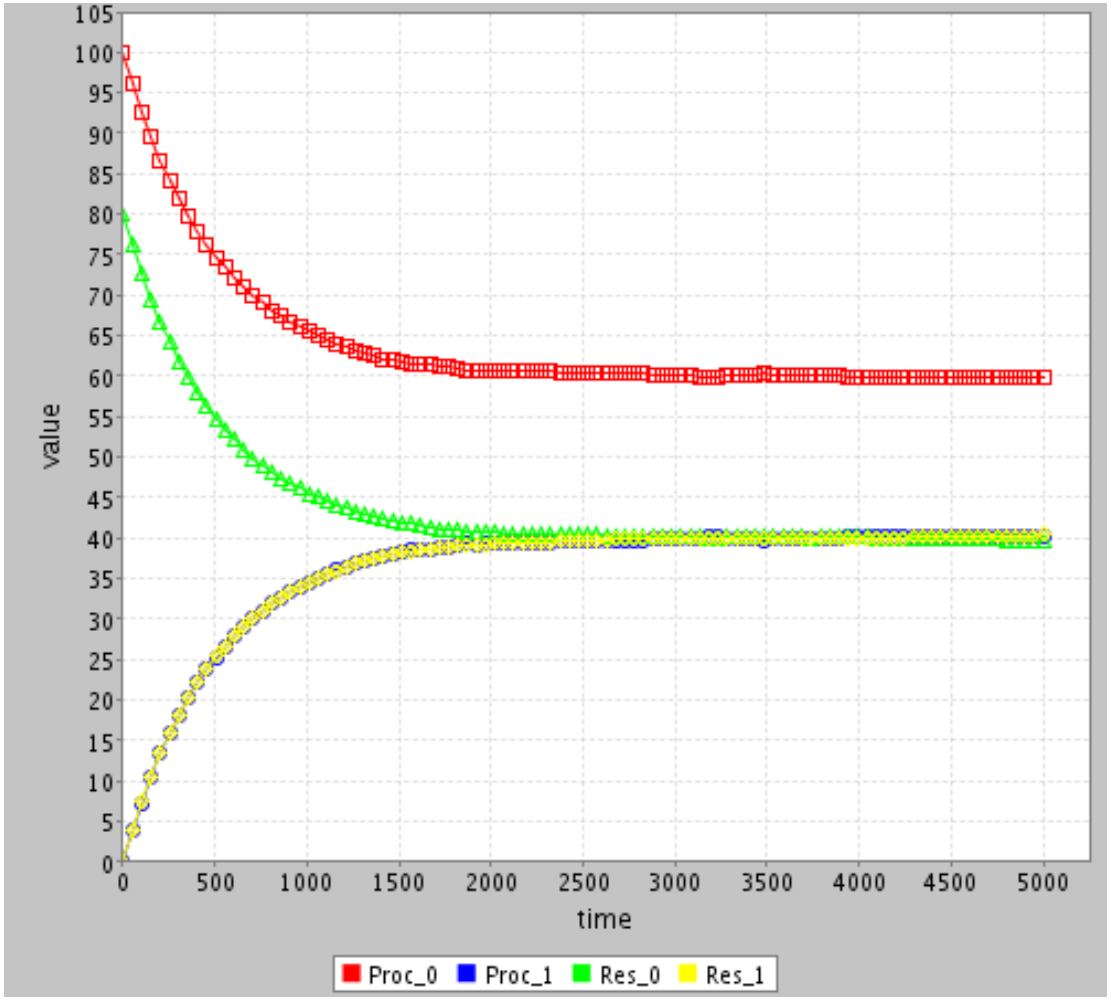
Processors and resources (average of 10 runs)



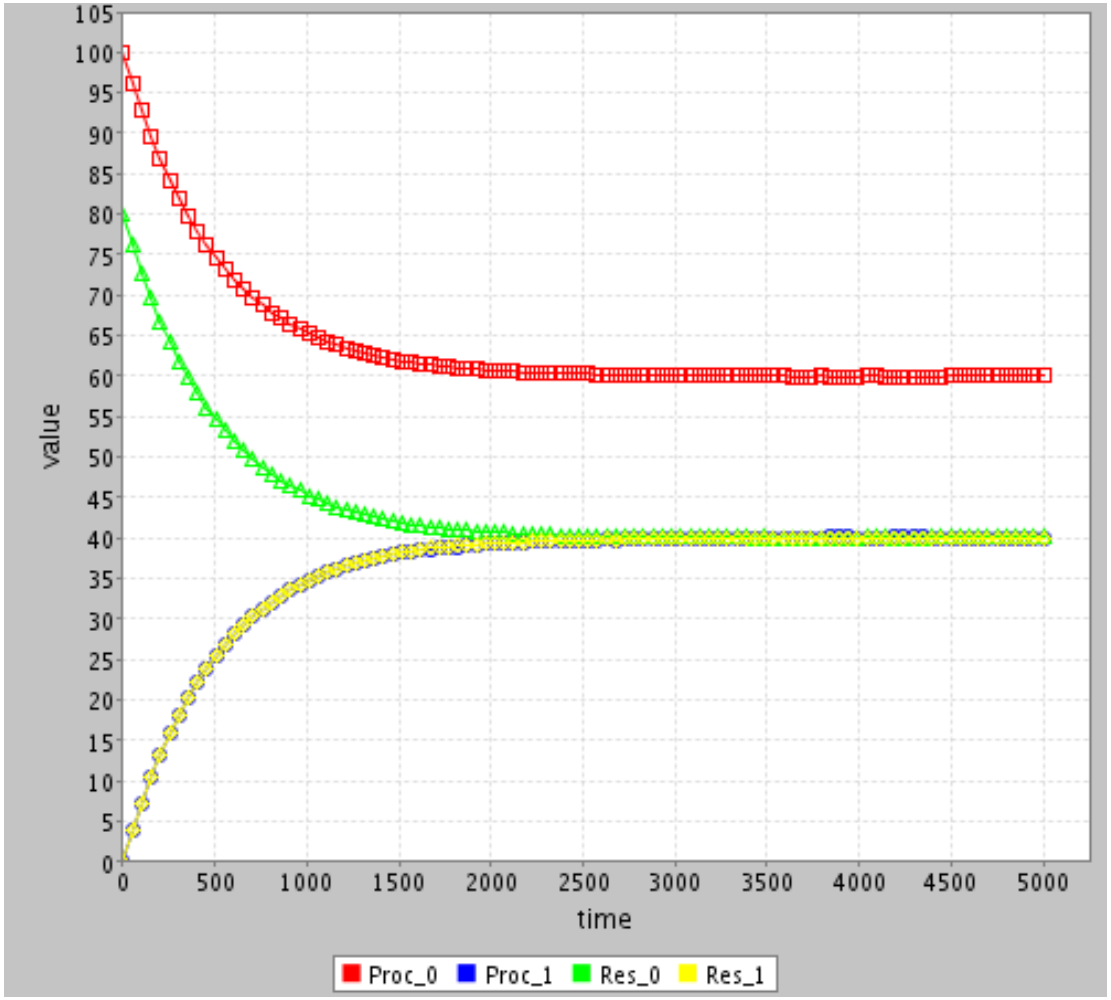
Processors and resources (average of 100 runs)



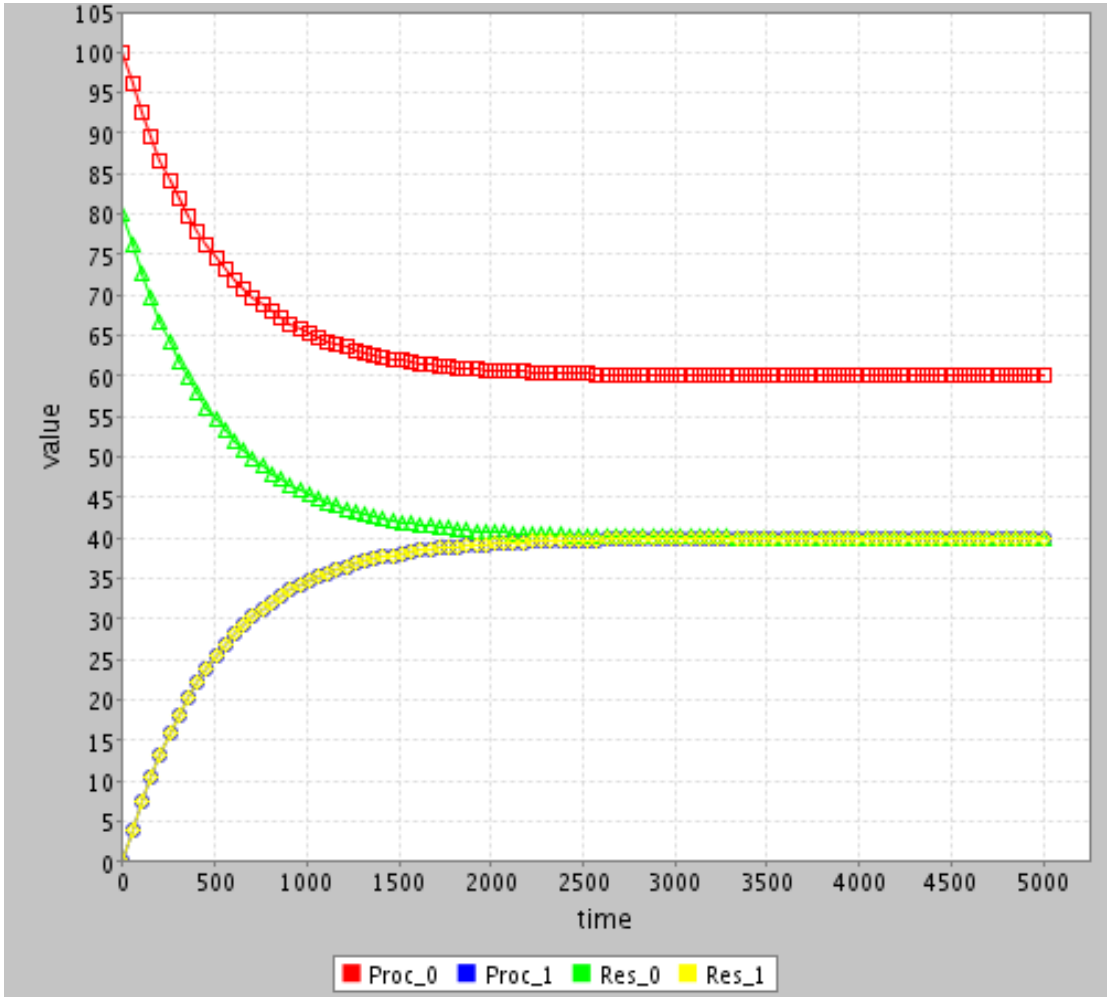
Processors and resources (average of 1000 runs)



Processors and resources (average of 10000 runs)



Processors and resources (ODE solution)



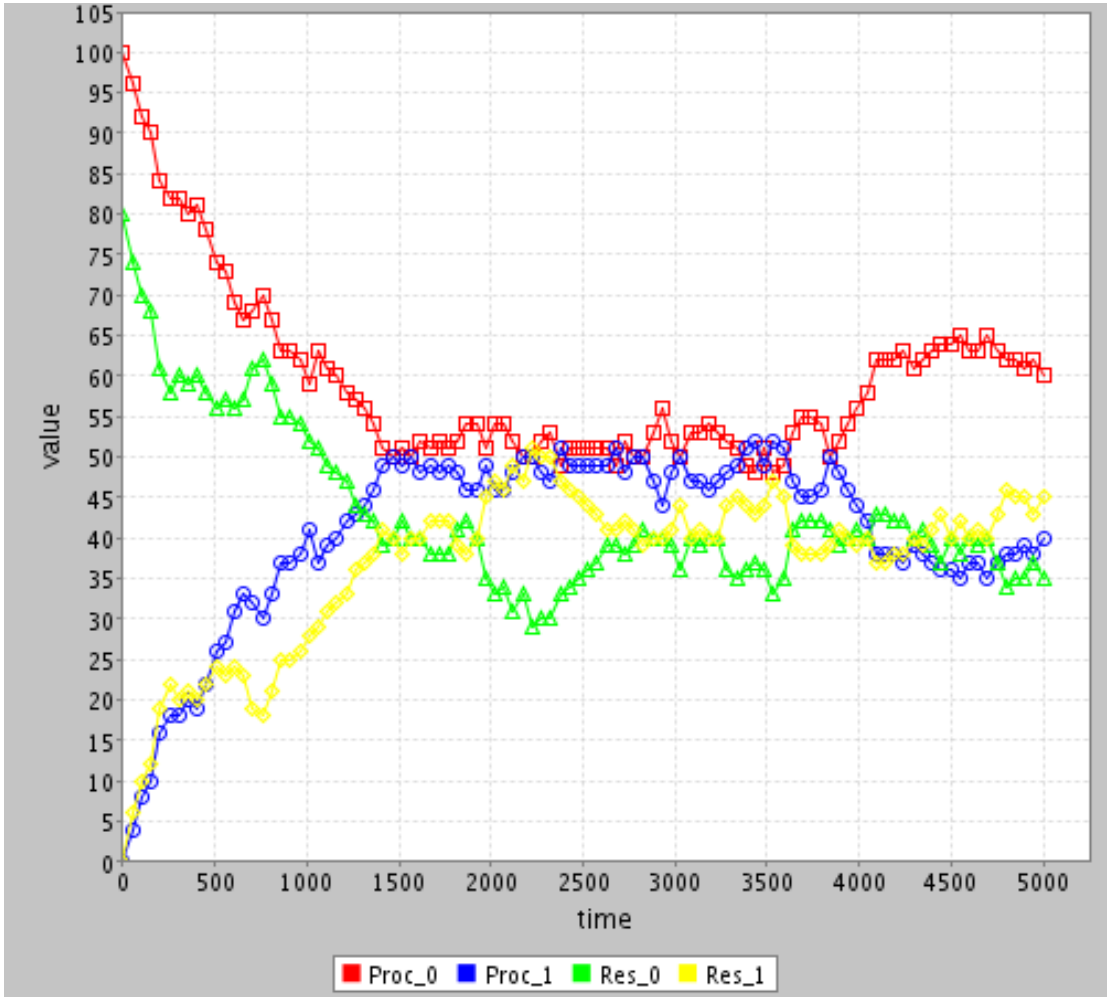
From realisations to ensembles

As we repeatedly sample from the underlying random number distributions the average of the samples tends to the mean.

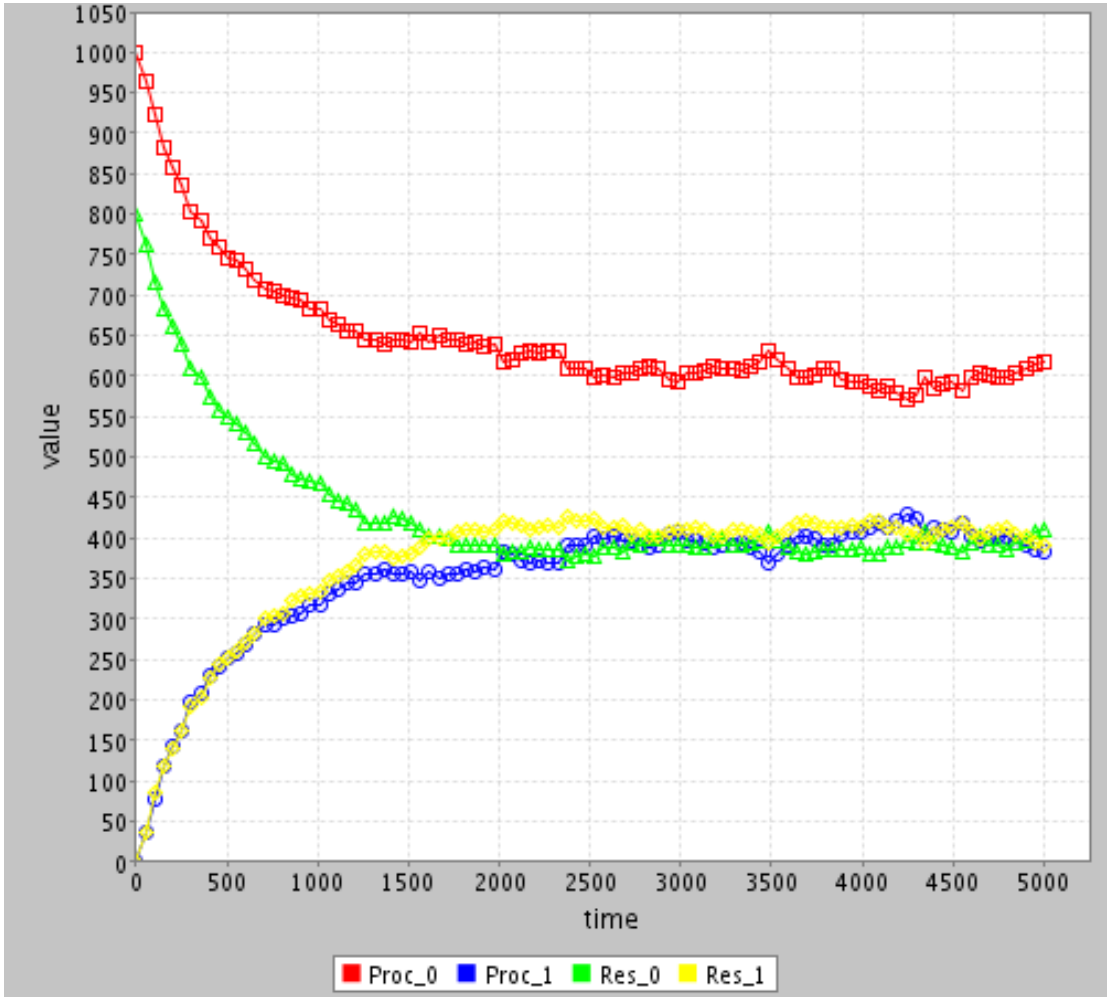
Processors and resources: scaling out

What effect does increasing the number of copies have?

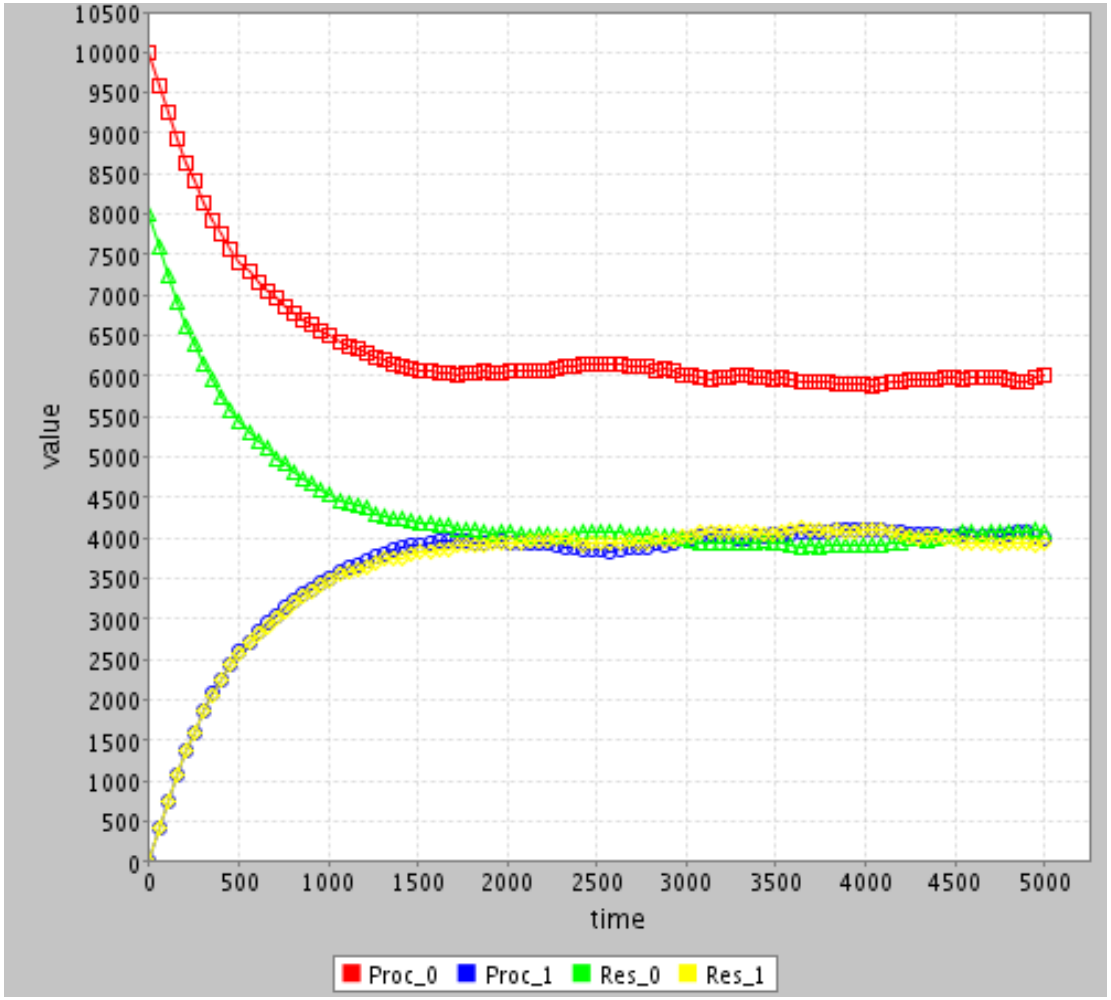
Processors and resources (single SSA run, 100/80)



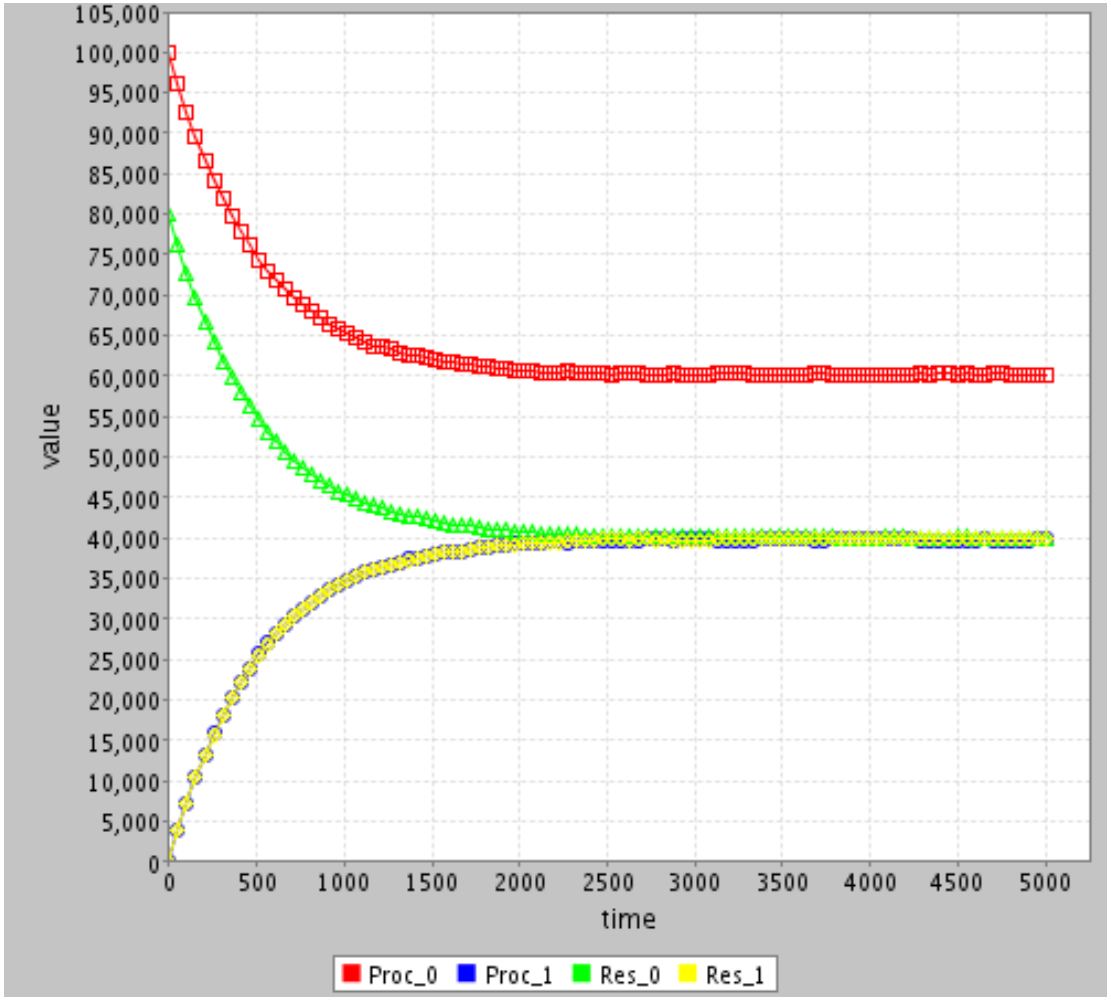
Processors and resources (single SSA run, 1,000/800)



Processors and resources (single SSA run, 10,000/8,000)



Processors and resources (single SSA run, 100,000/80,000)



Systems of many components

Each specific run is individually in closer and closer agreement with the differential equation approach as the number of components in the system increases.

Outline

Stochastic simulation: Propensity functions

As explicitly derived by Gillespie, the stochastic model uses a form often termed the propensity function that gives the probability a_μ of activity μ occurring in time interval $(t, t + \mathrm{d}t)$.

$$a_\mu \mathrm{d}t = h_\mu c_\mu \mathrm{d}t$$

where the M possible activities are given an arbitrary index μ $(1 \leq \mu \leq M)$, h_μ denotes the number of possible combinations of components involved in μ, and c_μ is a stochastic rate constant.

Stochastic: Grand probability function

The stochastic formulation proceeds by considering the grand probability function $\Pr(\mathbf{X}; t) \equiv$ probability that there will be at time t, X_i copies of component S_i, where $\mathbf{X} \equiv (X_1, X_2, \ldots X_N)$ is a vector of populations.

Stochastic: Grand probability function

The stochastic formulation proceeds by considering the grand probability function $\Pr(\mathbf{X}; t) \equiv$ probability that there will be at time t, X_i copies of component S_i, where $\mathbf{X} \equiv (X_1, X_2, \ldots X_N)$ is a vector of populations.

Evidently, knowledge of this function provides a complete understanding of the probability distribution of all possible states at all times.

Stochastic: Infinitesimal time interval

By considering a discrete infinitesimal time interval $(t, t + \mathrm{d}t)$ in which either no activities occur or exactly one does we see that there exist only $M + 1$ distinct configurations at time t that can lead to the state $\mathbf{X}$ at time $t + \mathrm{d}t$.

Stochastic: Infinitesimal time interval

By considering a discrete infinitesimal time interval $(t, t + \mathrm{d}t)$ in which either no activities occur or exactly one does we see that there exist only $M + 1$ distinct configurations at time t that can lead to the state $\mathbf{X}$ at time $t + \mathrm{d}t$.

$$\Pr(\mathbf{X}; t + \mathrm{d}t)$$
$$= \Pr(\mathbf{X}; t) \Pr(\text{no state change over } \mathrm{d}t)$$

Stochastic: Infinitesimal time interval

By considering a discrete infinitesimal time interval $(t, t + \mathrm{d}t)$ in which either no activities occur or exactly one does we see that there exist only $M + 1$ distinct configurations at time t that can lead to the state $\mathbf{X}$ at time $t + \mathrm{d}t$.

$$\begin{aligned}
&\Pr(\mathbf{X}; t + \mathrm{d}t) \\
&\quad = \Pr(\mathbf{X}; t) \Pr(\text{no state change over } \mathrm{d}t) \\
&\quad + \textstyle\sum_{\mu=1}^{M} \Pr(\mathbf{X} - \mathbf{v}_\mu; t) \Pr(\text{state change to } \mathbf{X} \text{ over } \mathrm{d}t)
\end{aligned}$$

Stochastic: Infinitesimal time interval

By considering a discrete infinitesimal time interval $(t, t + \mathrm{d}t)$ in which either no activities occur or exactly one does we see that there exist only $M + 1$ distinct configurations at time t that can lead to the state $\mathbf{X}$ at time $t + \mathrm{d}t$.

$$\begin{aligned}
&\Pr(\mathbf{X}; t + \mathrm{d}t) \\
&\quad = \Pr(\mathbf{X}; t) \Pr(\text{no state change over } \mathrm{d}t) \\
&\quad + \textstyle\sum_{\mu=1}^{M} \Pr(\mathbf{X} - \mathbf{v}_\mu; t) \Pr(\text{state change to } \mathbf{X} \text{ over } \mathrm{d}t)
\end{aligned}$$

where $\mathbf{v}_\mu$ is a vector defining the result of reaction μ on state vector $\mathbf{X}$, i.e. $\mathbf{X} \rightarrow \mathbf{X} + \mathbf{v}_\mu$ after an occurrence of reaction μ.

Stochastic: State change probabilities

Pr(no state change over dt)

$$1 - \sum_{\mu=1}^{M} a_\mu(\mathbf{X})\mathrm{d}t$$

Stochastic: State change probabilities

Pr(no state change over dt)

$$1 - \sum_{\mu=1}^{M} a_\mu(\mathbf{X})\mathrm{d}t$$

Pr(state change to $\mathbf{X}$ over dt)

$$\sum_{\mu=1}^{M} \Pr(\mathbf{X} - \mathbf{v}_\mu; t) a_\mu(\mathbf{X} - \mathbf{v}_\mu)\mathrm{d}t$$

Learning from other disciplines

Computing is not the only discipline where problems of scale and multi-scale occur. A recent trend has been to borrow simulation methods from computational biology and computational chemistry.

Stochastic simulation algorithms

Gillespie's Stochastic Simulation Algorithm (SSA) is essentially an exact procedure for numerically simulating the time evolution of a well-stirred chemically reacting system by taking proper account of the randomness inherent in such a system.

Stochastic simulation algorithms

Gillespie's Stochastic Simulation Algorithm (SSA) is essentially an exact procedure for numerically simulating the time evolution of a well-stirred chemically reacting system by taking proper account of the randomness inherent in such a system.

It is rigorously based on the same microphysical premise that underlies the chemical master equation and gives a more realistic representation of a system's evolution than the deterministic reaction rate equation (RRE) represented mathematically by ODEs.

Gillespie's exact SSA (1977)

- The algorithm takes time steps of variable length, based on the rate constants and population size of each chemical species.

Gillespie's exact SSA (1977)

- The algorithm takes time steps of variable length, based on the rate constants and population size of each chemical species.
- The probability of one reaction occurring relative to another is dictated by their relative propensity functions.

Gillespie's exact SSA (1977)

- The algorithm takes time steps of variable length, based on the rate constants and population size of each chemical species.
- The probability of one reaction occurring relative to another is dictated by their relative propensity functions.
- One random variable is then used to choose which reaction will occur, and another random variable determines how long the step will last.

Gillespie's exact SSA (1977)

- The algorithm takes time steps of variable length, based on the rate constants and population size of each chemical species.
- The probability of one reaction occurring relative to another is dictated by their relative propensity functions.
- One random variable is then used to choose which reaction will occur, and another random variable determines how long the step will last.
- The chemical populations are altered according to the definition of the reaction and the process is repeated.

Gillespie's SSA is a Monte Carlo Markov Chain simulation

The SSA is a Monte Carlo type method. With the SSA one may approximate any variable of interest by generating many trajectories and observing the statistics of the values of the variable. Since many trajectories are needed to obtain a reasonable approximation, the efficiency of the SSA is of critical importance.

Computational cost of Gillespie's exact algorithm

The cost of this detailed stochastic simulation algorithm is the likely large amounts of computing time. The key issue is that the time step for the next reaction can be very small indeed if we are to guarantee that only one reaction can take place in a given time interval.

Computational cost of Gillespie's exact algorithm

The cost of this detailed stochastic simulation algorithm is the likely large amounts of computing time. The key issue is that the time step for the next reaction can be very small indeed if we are to guarantee that only one reaction can take place in a given time interval.

Increasing the molecular population or number of reaction mechanisms necessarily requires a corresponding decrease in the time interval. The SSA can be very computationally inefficient especially when there are large numbers of molecules or the propensity functions are large.

Enhanced stochastic simulation techniques

If the system under study possesses a macroscopically infinitesimal timescale so that during any dt all of the reaction channels can fire many times, yet none of the propensity functions change appreciably, then the discrete Markov process as described by the SSA can be approximated by a continuous Markov process.

Enhanced stochastic simulation techniques

If the system under study possesses a macroscopically infinitesimal timescale so that during any dt all of the reaction channels can fire many times, yet none of the propensity functions change appreciably, then the discrete Markov process as described by the SSA can be approximated by a continuous Markov process.

Enhanced stochastic simulation techniques

If the system under study possesses a macroscopically infinitesimal timescale so that during any dt all of the reaction channels can fire many times, yet none of the propensity functions change appreciably, then the discrete Markov process as described by the SSA can be approximated by a continuous Markov process.

Enhanced stochastic simulation techniques

If the system under study possesses a macroscopically infinitesimal timescale so that during any dt all of the reaction channels can fire many times, yet none of the propensity functions change appreciably, then the discrete Markov process as described by the SSA can be approximated by a continuous Markov process.

Gillespie's tau-leap method (2001)

Gillespie proposed two new methods, namely the τ-leap method and the midpoint τ-leap method in order to improve the efficiency of the SSA while maintaining acceptable losses in accuracy.

Daniel T. Gillespie.

Approximate accelerated stochastic simulation of chemically reacting systems.

J. Comp. Phys., 115(4):1716–1733, 2001.

Gillespie's tau-leap method (2001)

Gillespie proposed two new methods, namely the τ-leap method and the midpoint τ-leap method in order to improve the efficiency of the SSA while maintaining acceptable losses in accuracy.

Daniel T. Gillespie.

Approximate accelerated stochastic simulation of chemically reacting systems.

J. Comp. Phys., 115(4):1716–1733, 2001.

The key idea here is to take a larger time step and allow for more reactions to take place in that step, but under the proviso that the propensity functions do not change too much in that interval. By means of a Poisson approximation, the tau-leaping method can "leap over" many reactions.

Gillespie's tau-leap method (drawback)

The use of approximation in Poisson methods leads to the possibility of negative molecular numbers being predicted — something with no physical explanation.

Gillespie's Modified Poisson tau-leap methods (2005)

Gillespie's modified Poisson tau-leaping method introduces a second control parameter whose value dials the procedure from the original Poisson tau-leaping method at one extreme to the exact SSA at the other.

Gillespie's Modified Poisson tau-leap methods (2005)

Gillespie's modified Poisson tau-leaping method introduces a second control parameter whose value dials the procedure from the original Poisson tau-leaping method at one extreme to the exact SSA at the other.

Any reaction channel with a positive propensity function which is within n_c firings of exhausting its reactants is termed a *critical* reaction.

Y. Cao, D. Gillespie, and L. Petzold.

Avoiding negative populations in explicit tau leaping.

J. Chem. Phys, 123(054104), 2005.

Gillespie's Modified Poisson tau-leap methods (2006)

The modified algorithm chooses τ in such a way that no more than *one* firing of *all* the critical reactions can occur during the leap. The probability of producing a negative population is reduced to nearly zero.

Gillespie's Modified Poisson tau-leap methods (2006)

The modified algorithm chooses τ in such a way that no more than *one* firing of *all* the critical reactions can occur during the leap. The probability of producing a negative population is reduced to nearly zero.

If a negative population *does* occur the leap can simply be rejected and repeated with τ reduced by half, or the entire simulation can be abandoned and repeated for larger n_c.

Y. Cao, D. Gillespie, and L. Petzold.

Efficient stepsize selection for the tau-leaping simulation method.

J. Chem. Phys, 2006, 124(4):044109.

Family of stochastic simulation algorithms

FASTEST, BEST	
Discrete, exact	**Continuous, approximate**
	Modified Poisson τ leap (2005)
	τ leap (2001)
Logarithmic Direct Method (2006)	
Sorting Direct Method (2005)	
Optimised Direct Method (2004)	
Next Reaction Method (2000)	
Direct Method (1977)	
First Reaction Method (1977)	
SLOWEST, WORST	

Conclusions

- Stochastic methods can be used to investigate the temporal and behavioural nature of global computing systems.
- Numerical solution of Markov chains gives a thorough stochastic treatment of models of modest size.
- Differential equation methods scale well to increasing population sizes allowing global computing systems to be addressed.
- Stochastic simulation algorithms make a bridge between the two approaches: there has been a recent explosion of interest in the subject with many new variants of Gillespie's algorithm.