SENSORIA
www.sensoria-ist.eu

Model-based Performance Analysis

Stephen Gilmore and Jane Hillston
LFCS, University of Edinburgh

SENSUS Summerschool
Keszthely, Hungary

30th June 2009

Information Society Technologies

Service-oriented Computing

SENSORIA
www.sensoria-ist.eu

What distinguishes *service-oriented computing* from *component based software development*?

Information Society Technologies

Service-Oriented Computing: Some differences

SENSORIA
www.sensoria-ist.eu

- 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

Information Society Technologies

Modelling Service-Oriented Computing: The challenges

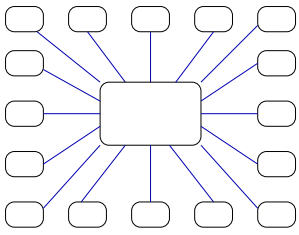
SENSORIA
www.sensoria-ist.eu

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?

Information Society Technologies

Quantitative Modelling: Motivation

SENSORIA
www.sensoria-ist.eu

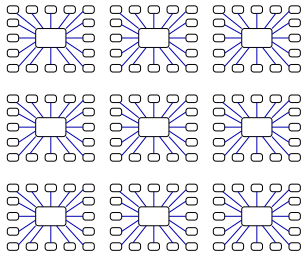


Quality of Service issues
- Can the server maintain reasonable response times?

Information Society Technologies

Quantitative Modelling: Motivation

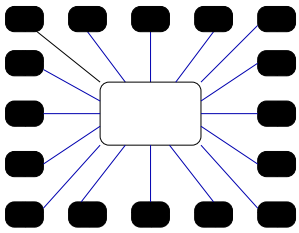
SENSORIA
www.sensoria-ist.eu



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

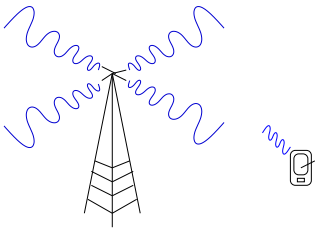
Information Society Technologies

Quantitative Modelling: Motivation



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

Quantitative Modelling: Motivation

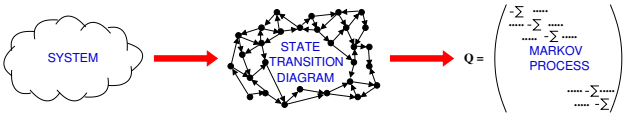


Service-level agreements
- What percentage of downloads do complete within the time we advertised?

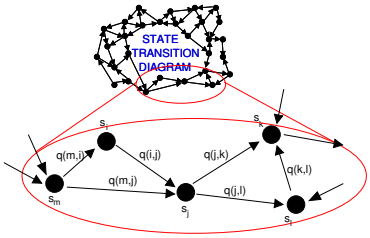
Outline

1. Introduction
2. Interplay: Process Algebra and Markov Process
3. Model-checking

Performance Modelling using CTMC

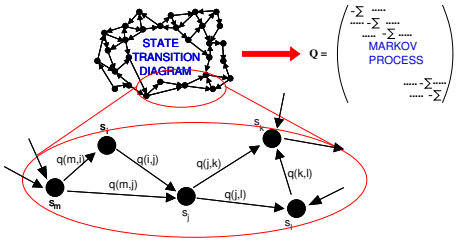


Performance Modelling using CTMC



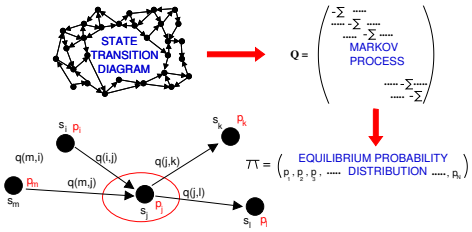
A negative exponentially distributed duration is associated with each transition.

Performance Modelling using CTMC



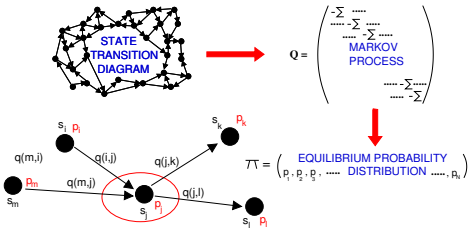
these parameters form the entries of the infinitesimal generator matrix Q

Performance Modelling using CTMC



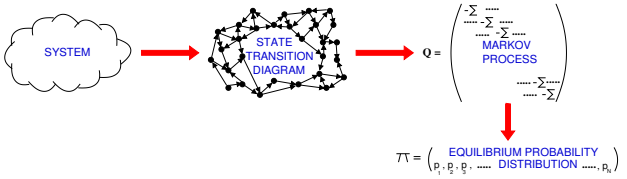
In steady state the probability flux out of a state is balanced by the flux in.

Performance Modelling using CTMC

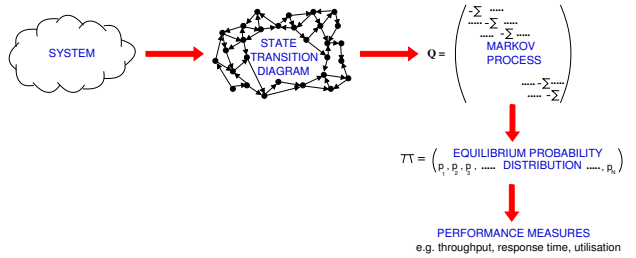


"Global balance equations" captured by $\pi Q = 0$ solved by linear algebra

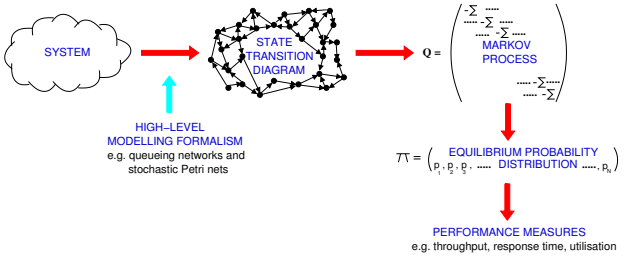
Performance Modelling using CTMC



Performance Modelling using CTMC



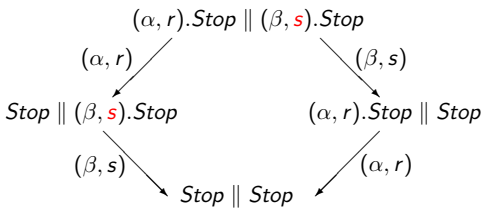
Performance Modelling using CTMC



Stochastic process algebras

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

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

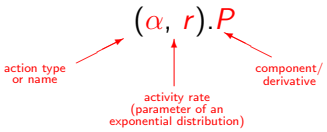
We retain the expansion law of classical process algebra:

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

only if the negative exponential distribution is used.

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.



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

PEPA

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

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.
- 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

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

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

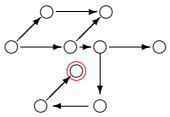
Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Integrated analysis: Reachability analysis

Reachability analysis

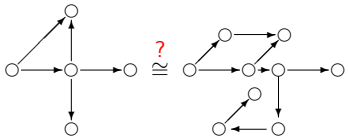
How long will it take for the system to arrive in a particular state?



Integrated analysis: Specification matching

Specification matching

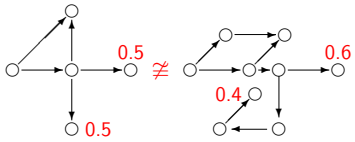
With what probability does system behaviour match its specification?



Integrated analysis: Specification matching

Specification matching

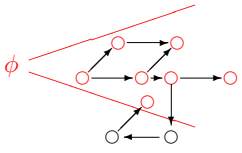
Does the "*frequency profile*" of the system match that of the specification?



Integrated analysis: Model checking

Model checking

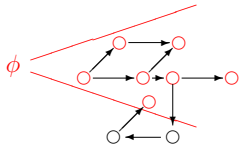
Does a given property ϕ hold within the system with a given probability?



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 — 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:

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

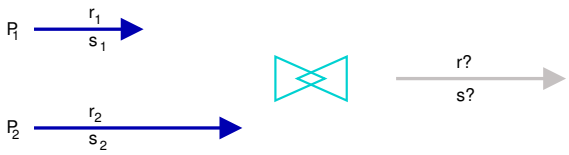
- No distinction between input and output actions.
- Communication or synchronisation takes place on the basis of shared names.
- The resulting action has the same name.

Most stochastic process algebras adopt CSP-style synchronisation.

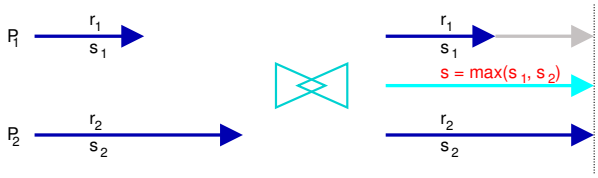
Timed Synchronisation

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

Timed Synchronisation

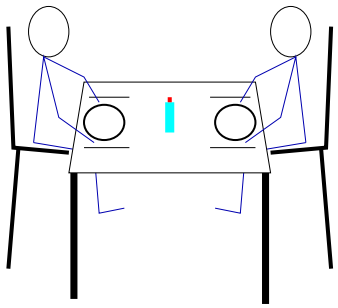


Timed Synchronisation

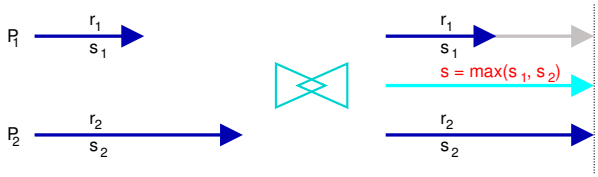


Barrier Synchronisation

Timed Synchronisation

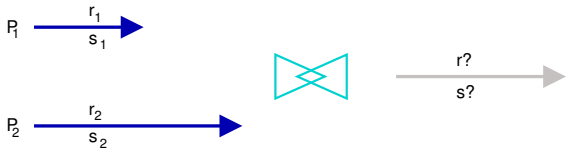


Timed Synchronisation



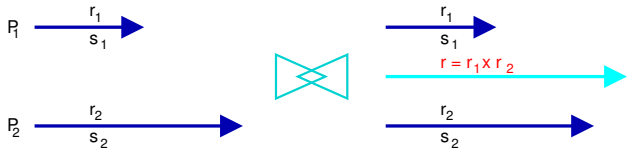
s is no longer exponentially distributed

Timed Synchronisation



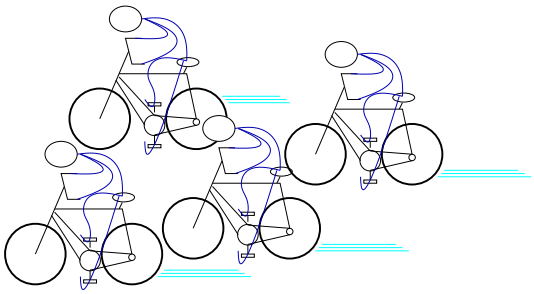
algebraic considerations limit choices

Timed Synchronisation

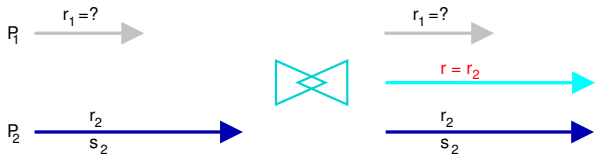


TIPP: new rate is product of individual rates

Timed Synchronisation

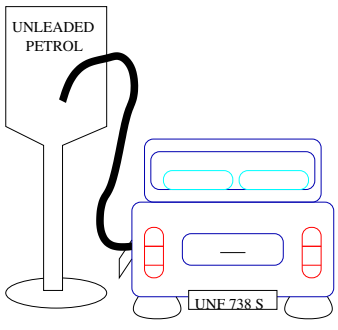


Timed Synchronisation

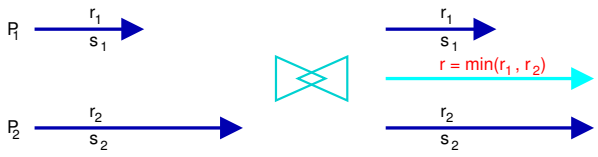


EMPA: one participant is passive

Timed Synchronisation

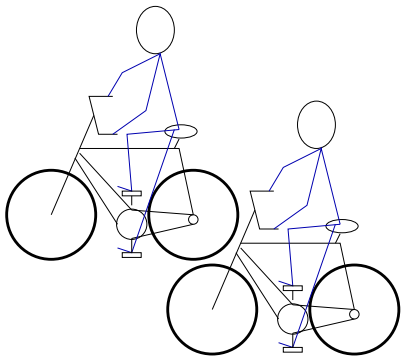


Timed Synchronisation



bounded capacity: new rate is the minimum of the rates

Timed Synchronisation



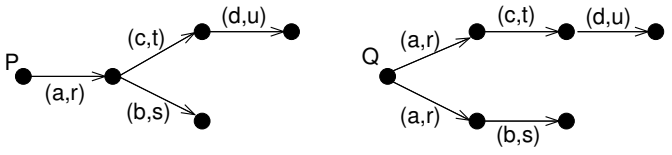
Cooperation in PEPA

- In PEPA each component has a bounded capacity to carry out activities of any particular type, determined by the apparent rate for that type.
- 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.

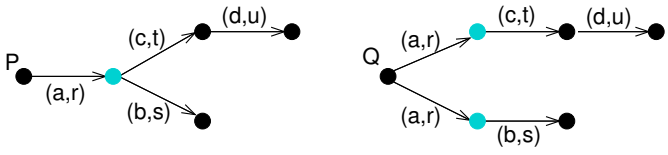
Equivalence Relations

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

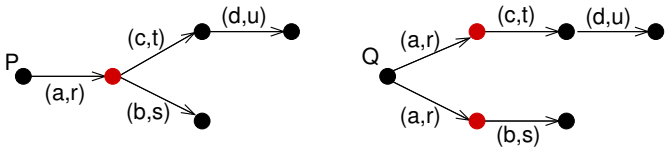
Equivalence Relations



Equivalence Relations



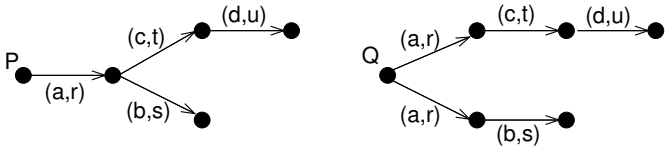
Equivalence Relations



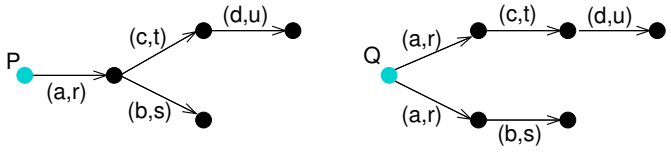
Equivalence Relations

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

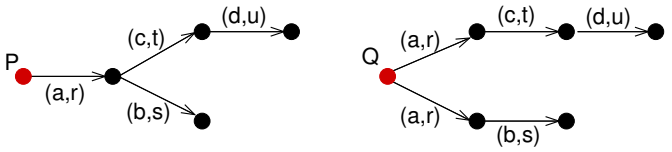
Equivalence Relations



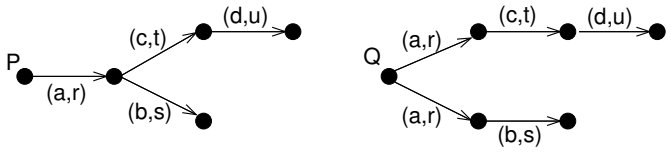
Equivalence Relations



Equivalence Relations



Equivalence Relations



Equivalence Relations

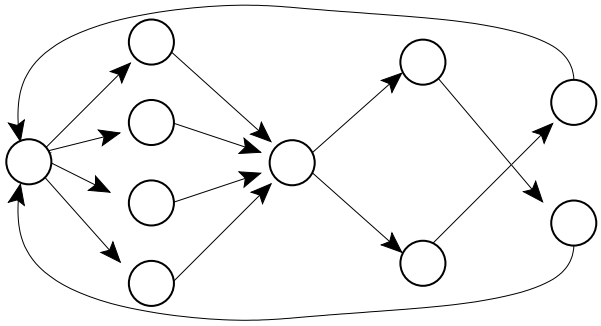
The resulting equivalence relation is a bisimulation in the style of Larsen and Skou, and coincides with the Markov process notion of lumpability.

Aggregation and lumpability

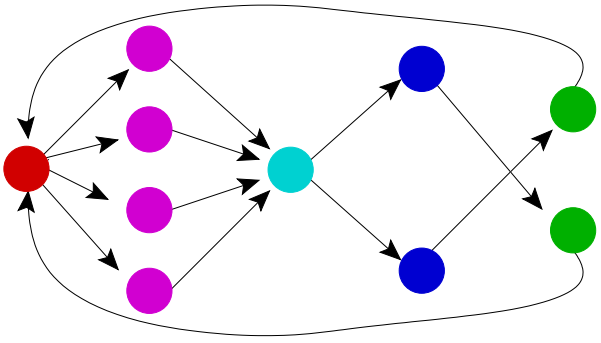
Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state, i.e. take a different stochastic representation of the same model.

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

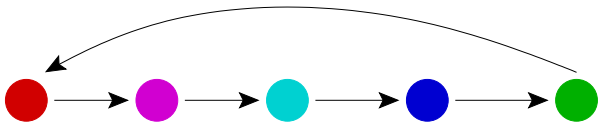
Aggregation and lumpability



Aggregation and lumpability



Aggregation and lumpability



Model-checking

A widely-used logic for model checking properties against continuous-time Markov chains is Continuous Stochastic Logic (CSL). The well-formed formulae of CSL are made up of *state formulae* ϕ and *path formulae* ψ.

Continuous Stochastic Logic (CSL)

State formulae

$\phi \quad ::= \quad \mathit{true} \mid \mathit{false} \mid a \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \mathcal{P}_{\star}[\psi] \mid \mathcal{S}_{\star}[\phi]$

Path formulae

$\psi \quad ::= \quad \mathrm{X}\,\phi \mid \phi \,\mathrm{U}^{I}\, \phi \mid \phi \,\mathrm{U}\, \phi$

where a is an atomic proposition, $\star$ is "$< p$", "$> p$", "$=?$" being a qualifying parameter, p is a probability, and I is an interval of $\mathbb{R}$. $\mathrm{X}\,\phi$ means "in the neXt state ϕ holds" and the meaning of the formula $\phi_1 \,\mathrm{U}\, \phi_2$ is "ϕ_1 holds Until ϕ_2 is satisfied".

CSL example

What is the probability that, going by any path, we reach a success state within 10 seconds?

$$\mathcal{P}_{=?}[\mathit{true}\ \mathrm{U}^{[\cdot,10]}\ \mathit{success}]$$

About the model-checker

Correctness of the model-checker:

- Validated against the PRISM model-checker.
- This implementation shares no code with PRISM.

Why not just use PRISM?

New features of the model-checker:

- Ability to abstract states to reduce the state-space overall.
  - Not found in PRISM
  - Analysis returns an *interval* with lower and upper bounds on the probability.

Interface of the model-checker

Objectives of the interface:

- Make it possible to perform CSL model-checking.
- Make it intuitive to add atomic propositions to states.
- Guide the user through building a well-formed CSL formula.
- Hide the term language of the process calculus used.