SPAs for performance modelling: Lecture 3 — Model Manipulations

Jane Hillston

LFCS, School of Informatics
The University of Edinburgh
Scotland

10th April 2013

THE UNIVERSITY
of EDINBURGH

Outline

Outline

Dynamic behaviour

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

Dynamic behaviour

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

Dynamic behaviour

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

Dynamic behaviour

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

PEPA Eclipse Plug-In input

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

$$P_1 \parallel P_1$$

PEPA Eclipse Plug-In input

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

$$P_1 \parallel P_1$$

State space

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

PEPA Eclipse Plug-In input

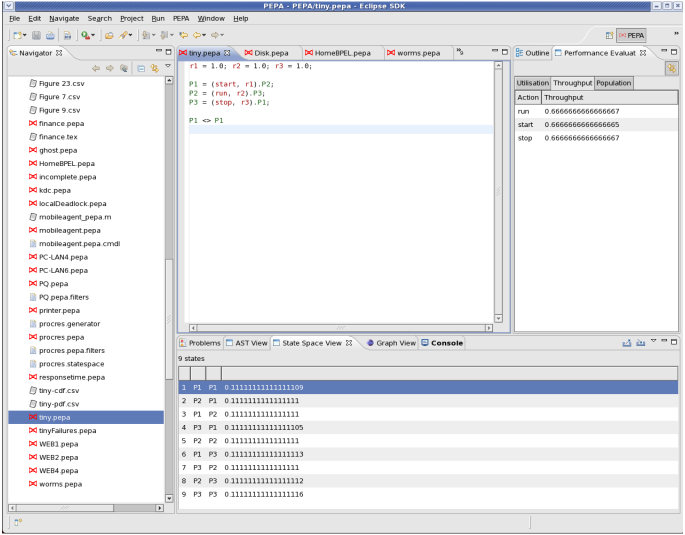
$$P_1 \stackrel{\mathit{def}}{=} (\mathit{start}, r_1).P_2 \qquad P_2 \stackrel{\mathit{def}}{=} (\mathit{run}, r_2).P_3 \qquad P_3 \stackrel{\mathit{def}}{=} (\mathit{stop}, r_3).P_1$$

$$P_1 \parallel P_1$$

CTMC representation computed by the plug-in

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

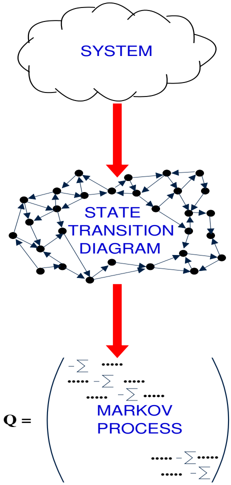
The PEPA Eclipse Plug-in processing the model



Performance Modelling using CTMC

Model Construction

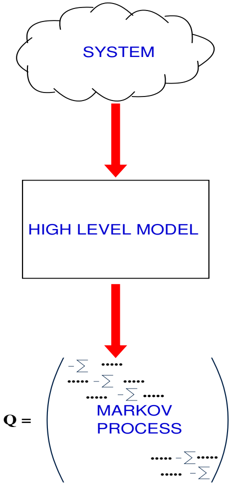
- describing the system using a high level modelling formalism
- generating the underlying CTMC



Performance Modelling using CTMC

Model Construction

- describing the system using a high level modelling formalism
- generating the underlying CTMC



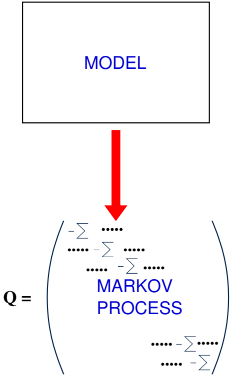
Performance Modelling using CTMC

Model Construction

- describing the system using a high level modelling formalism
- generating the underlying CTMC

Model Manipulation

- model simplification
- model aggregation



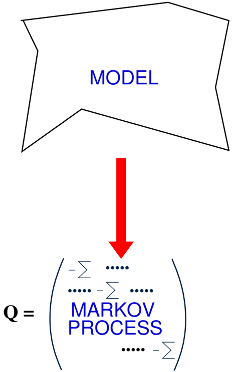
Performance Modelling using CTMC

Model Construction

- describing the system using a high level modelling formalism
- generating the underlying CTMC

Model Manipulation

- model simplification
- model aggregation



Performance Modelling using CTMC

Model Construction

- describing the system using a high level modelling formalism
- generating the underlying CTMC

Model Manipulation

- model simplification
- model aggregation

Model Solution

- solving the CTMC to find steady state probability distribution
- deriving performance measures

Outline

Equivalence relations in Performance Modelling

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

Equivalence relations in Performance Modelling

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

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

Equivalence relations in Performance Modelling

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

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

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

Aggregation and lumpability

- Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state.

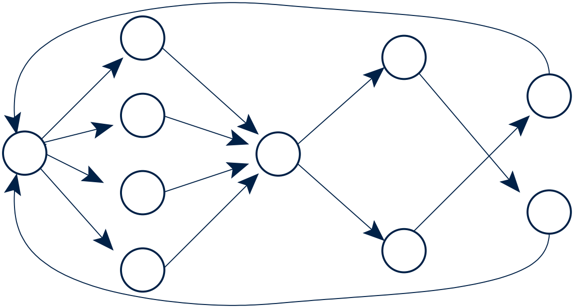
Aggregation and lumpability

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

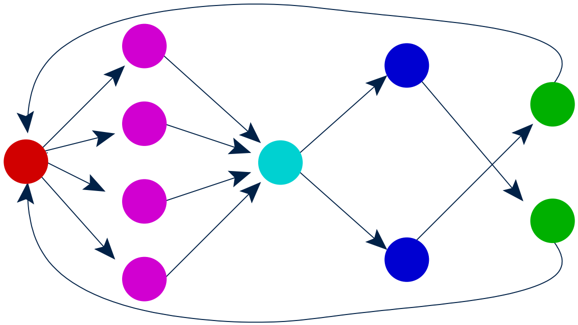
Aggregation and lumpability

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

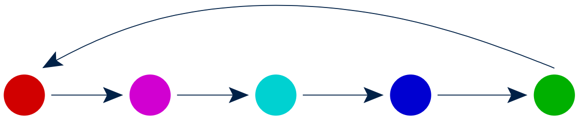
Reducing by lumpability



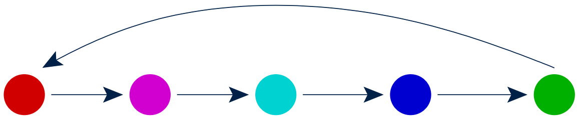
Reducing by lumpability



Reducing by lumpability



Reducing by lumpability



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

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

Lumpability

- In the early 1960's Kemeny and Snell established the conditions under which it was possible to lump a Markov chain and still have a Markov chain afterwards.

Lumpability

- In the early 1960's Kemeny and Snell established the conditions under which it was possible to lump a Markov chain and still have a Markov chain afterwards.
- In particular these conditions were characterised by conditions on the rates which are straightforward to check.

Lumpability

- In the early 1960's Kemeny and Snell established the conditions under which it was possible to lump a Markov chain and still have a Markov chain afterwards.
- In particular these conditions were characterised by conditions on the rates which are straightforward to check.
- However checking the conditions did involve constructing the complete Markov chain first.

Lumpability

- In the early 1960's Kemeny and Snell established the conditions under which it was possible to lump a Markov chain and still have a Markov chain afterwards.
- In particular these conditions were characterised by conditions on the rates which are straightforward to check.
- However checking the conditions did involve constructing the complete Markov chain first.
- This is something of a catch-22 situation when the problem is that the state space of the Markov chain is too large to handle.

Lumpability

If the original state space is $\{X_1, X_2, \ldots, X_n\}$ then the aggregated state space is some $\{X_{[1]}, X_{[2]}, \ldots, X_{[N]}\}$ where $N < n$ and ideally $N << n$.

Lumpability

If the original state space is $\{X_1, X_2, \ldots, X_n\}$ then the aggregated state space is some $\{X_{[1]}, X_{[2]}, \ldots, X_{[N]}\}$ where $N < n$ and ideally $N << n$.

In order to define a Markov chain in terms of the aggregated states we first need to work out the transition rates between these macro-states.

Lumped transition rates

If the transition rates of the original process are $q(X_i, X_k)$ then the transition rates into any partition from a state is

$$q(X_i, X_{[j]}) = \sum_{k \in [j]} q(X_i, X_j)$$

Lumped transition rates

If the transition rates of the original process are $q(X_i, X_k)$ then the transition rates into any partition from a state is

$$q(X_i, X_{[j]}) = \sum_{k \in [j]} q(X_i, X_j)$$

Transition rates between partitions are the weighted sum of the transition rates of each state in the first partition to the second partition, weighted by the conditional steady state probability of that state in the partition, $\overline{\pi}_j(\cdot)$

$$q(X_{[j]}, X_{[i]}) = \sum_{k \in [j]} \overline{\pi}_j(X_k) q(X_k, X_{[i]})$$

Ordinary, Exact and Strict Lumpability

- A Markov process is ordinarily lumpable with respect to a partition $\chi = \{X_{[i]}\}$ iff, for any $X_{[k]}, X_{[l]} \in \chi, X_i, X_j \in X_{[k]}$

$$q(X_i, X_{[l]}) = q(X_j, X_{[l]})$$

Ordinary, Exact and Strict Lumpability

- A Markov process is ordinarily lumpable with respect to a partition $\chi = \{X_{[i]}\}$ iff, for any $X_{[k]}, X_{[l]} \in \chi, X_i, X_j \in X_{[k]}$

$$q(X_i, X_{[l]}) = q(X_j, X_{[l]})$$

- χ is an exactly lumpable partition iff, for any $X_{[k]}, X_{[l]} \in \chi, X_i, X_j \in X_{[l]}$

$$q(X_{[k]}, X_i) = q(X_{[k]}, X_j)$$

Ordinary, Exact and Strict Lumpability

- A Markov process is ordinarily lumpable with respect to a partition $\chi = \{X_{[i]}\}$ iff, for any $X_{[k]}, X_{[l]} \in \chi, X_i, X_j \in X_{[k]}$

$$q(X_i, X_{[l]}) = q(X_j, X_{[l]})$$

- χ is an exactly lumpable partition iff, for any $X_{[k]}, X_{[l]} \in \chi, X_i, X_j \in X_{[l]}$

$$q(X_{[k]}, X_i) = q(X_{[k]}, X_j)$$

- χ is a strictly lumpable partition iff it is ordinarily lumpable and exactly lumpable.

Outline

Equivalence Relations

It is standard for a process algebra to be equipped with a semantic equivalence — a notion of equivalence related to the operational semantics of the language.

Equivalence Relations

It is standard for a process algebra to be equipped with a semantic equivalence — a notion of equivalence related to the operational semantics of the language.

In CCS-style process algebras, equivalence relations are defined based on the notion of observability.

Equivalence Relations

It is standard for a process algebra to be equipped with a semantic equivalence — a notion of equivalence related to the operational semantics of the language.

In CCS-style process algebras, equivalence relations are defined based on the notion of observability.

The idea is that each process should be able to mimic the behaviour of the other process sufficiently that an external observer cannot distinguish them via observation.

Equivalence Relations

It is standard for a process algebra to be equipped with a semantic equivalence — a notion of equivalence related to the operational semantics of the language.

In CCS-style process algebras, equivalence relations are defined based on the notion of observability.

The idea is that each process should be able to mimic the behaviour of the other process sufficiently that an external observer cannot distinguish them via observation.

This symmetric relation is known as bisimulation.

Congruence relation

In process algebras we are particularly interested in relations which are congruences with respect to the operators of the language.

Congruence relation

In process algebras we are particularly interested in relations which are congruences with respect to the operators of the language.

For example,

- if we have two process terms P and P' which are related by a congruence relation $\mathcal{R}$, i.e. $P \mathcal{R} P'$ or $(P, P') \in \mathcal{R}$,
- then in any expression $\mathcal{E}$ which includes P, we can substitute P' to get an expression $\mathcal{E}'$
- and know that $(\mathcal{E}, \mathcal{E}') \in \mathcal{R}$.

Congruence relation

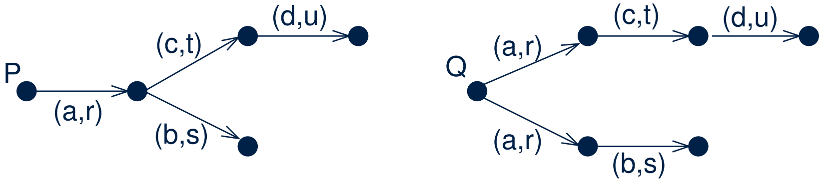
In process algebras we are particularly interested in relations which are congruences with respect to the operators of the language.

For example,

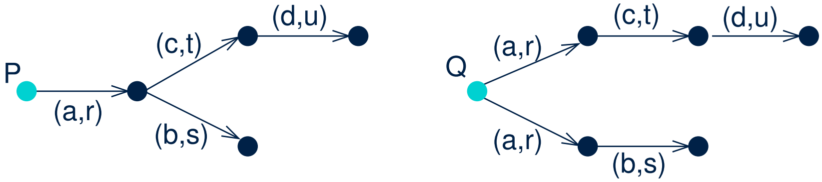
- if we have two process terms P and P' which are related by a congruence relation $\mathcal{R}$, i.e. $P \mathcal{R} P'$ or $(P, P') \in \mathcal{R}$,
- then in any expression $\mathcal{E}$ which includes P, we can substitute P' to get an expression $\mathcal{E}'$
- and know that $(\mathcal{E}, \mathcal{E}') \in \mathcal{R}$.

To prove that a relation is a congruence we need to show that this substitutivity for equivalent processes holds for each operator of the language.

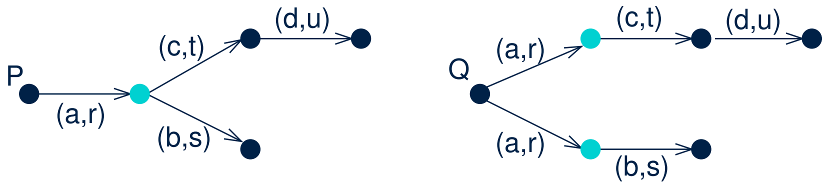
Classic Bisimulation



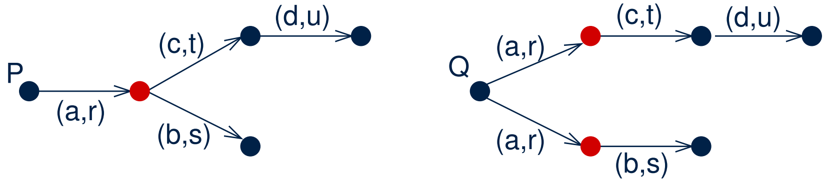
Classic Bisimulation



Classic Bisimulation



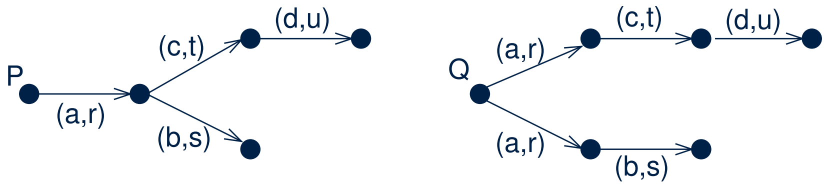
Classic Bisimulation



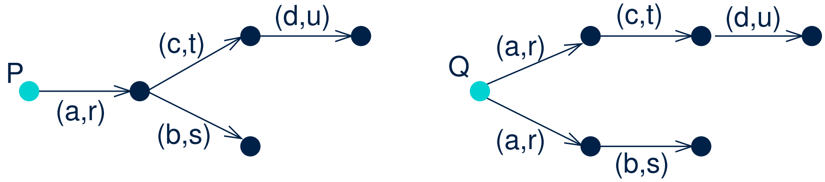
Markovian bisimulation

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

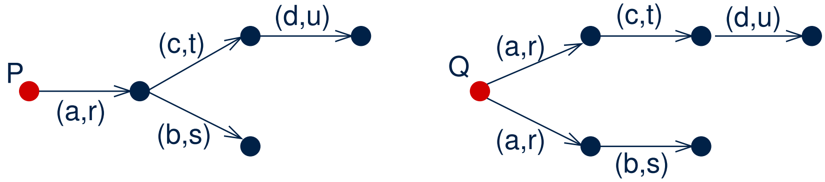
Markovian bisimulation



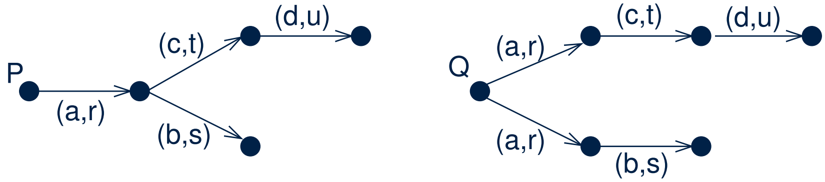
Markovian bisimulation



Markovian bisimulation



Equivalence Relations



Markovian bisimulation

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

Markovian bisimulation

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

Moreover this bisimulation is a congruence for all the combinators of PEPA.

Strong Equivalence in PEPA

Definition

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

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

where

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

Weak equivalence relations

In classic process algebras it is usual to consider weak equivalence relations in addition to the strong style of relations already discussed.

Weak equivalence relations

In classic process algebras it is usual to consider weak equivalence relations in addition to the strong style of relations already discussed.

In a weak relation the internal τ actions are abstracted so that the observer does not see internal actions.

Weak equivalence relations

In classic process algebras it is usual to consider weak equivalence relations in addition to the strong style of relations already discussed.

In a weak relation the internal τ actions are abstracted so that the observer does not see internal actions.

However weak equivalence relations are difficult to obtain for stochastic process algebra which have integrated time and action, although there some results from Bernardo *et al*.

Weak equivalence relations

The problem is that whilst $\tau.\alpha.P$ has the same observable behaviour as $\alpha.P$ it is not true that $(\tau, s).(\alpha, r).P$ has the same observable behaviour as $(\alpha, r).P$.

Weak equivalence relations

The problem is that whilst $\tau.\alpha.P$ has the same observable behaviour as $\alpha.P$ it is not true that $(\tau, s).(\alpha, r).P$ has the same observable behaviour as $(\alpha, r).P$.

The issue is that in the first process there is a delay (exponentially distributed with mean s) before the activity of type α commences, whereas in the second process the α activity starts immediately.

Weak equivalence relations

The problem is that whilst $\tau.\alpha.P$ has the same observable behaviour as $\alpha.P$ it is not true that $(\tau, s).(\alpha, r).P$ has the same observable behaviour as $(\alpha, r).P$.

The issue is that in the first process there is a delay (exponentially distributed with mean s) before the activity of type α commences, whereas in the second process the α activity starts immediately.

There seems to be no possibility of eliminating single τ type activities in a stochastically timed process algebra.

Weak equivalence relations

There is, however, the possibility of eliminating sequences of τ activities, e.g. reducing $(\tau, s).(\tau, r).P$ to $(\tau, t).P$ for an appropriate t.

Weak equivalence relations

There is, however, the possibility of eliminating sequences of τ activities, e.g. reducing $(\tau, s).(\tau, r).P$ to $(\tau, t).P$ for an appropriate t.

The difficulty is that the convolution of two exponentially distributed delays is no longer exponentially delayed.

Weak equivalence relations

There is, however, the possibility of eliminating sequences of τ activities, e.g. reducing $(\tau, s).(\tau, r).P$ to $(\tau, t).P$ for an appropriate t.

The difficulty is that the convolution of two exponentially distributed delays is no longer exponentially delayed.

Nevertheless the usual decision is to maintain the exponential distribution and the same mean duration: i.e. t is chosen to be $\frac{rs}{r+s}$.

Weak equivalence relations

For PEPA the closest has been the definition of weak isomorphism which collapses a sequence of τ actions to a single τ action with the same mean duration.

Weak equivalence relations

For PEPA the closest has been the definition of weak isomorphism which collapses a sequence of τ actions to a single τ action with the same mean duration.

In general any reduction based on this will be an approximation of the original model due to the use of the exponential distribution for the aggregate action.

Weak equivalence relations

For PEPA the closest has been the definition of weak isomorphism which collapses a sequence of τ actions to a single τ action with the same mean duration.

In general any reduction based on this will be an approximation of the original model due to the use of the exponential distribution for the aggregate action.

However, syntactic conditions have been identified for when this will be exact due to insensitivity.

Weak equivalence relations

For PEPA the closest has been the definition of weak isomorphism which collapses a sequence of τ actions to a single τ action with the same mean duration.

In general any reduction based on this will be an approximation of the original model due to the use of the exponential distribution for the aggregate action.

However, syntactic conditions have been identified for when this will be exact due to insensitivity.

Weak bisimulation relations are not typically preserved by the choice operator although they can be congruence relations with respect to the other operators.

Other equivalences

Further results on weak bisimulation have been obtained for SPA with orthogonal time and action such as IMC but even in this case there are some subtleties and it is not straightforward.

Other equivalences

Further results on weak bisimulation have been obtained for SPA with orthogonal time and action such as IMC but even in this case there are some subtleties and it is not straightforward.

Whilst most work on equivalences in SPAs have focussed on bisimulation style equivalences, Marco Bernardo and co-authors have developed branching and testing equivalences in the context of the stochastic process algebra EMPA, and consequently for CTMCs.

Outline

Querying models

So far we have focussed on the construction of the model and demonstrated the use of some particular examples to derive quantitative measures.

Querying models

So far we have focussed on the construction of the model and demonstrated the use of some particular examples to derive quantitative measures.

PEPA is complemented by a couple of formal approaches to query models.

- stochastic model checking based on a stochastic logic; and
- (eXtended) stochastic probes within the PEPA model.

Model checking

Model checking requires two inputs:

- a description of the system, usually given in some high-level modelling formalism such as a process algebra description, or a Petri net;

Model checking

Model checking requires two inputs:

- a description of the system, usually given in some high-level modelling formalism such as a process algebra description, or a Petri net;
- a specification of one or more desired properties of the system, normally using termporal logics such as CTL (Computational Tree Logic) or LTL (Linear-time Temporal Logic).

Model checking

From the high-level description the model checker constructs a labelled transition system which captures all possible behaviours of the system.

Model checking

From the high-level description the model checker constructs a labelled transition system which captures all possible behaviours of the system.

The model checking algorithms then automatically verify whether or not each property is satisfied in the system.

Stochastic model checking

In stochastic model checking it is assumed that the labelled transition system is a Continuous Time Markov Chain (CTMC).

Stochastic model checking

In stochastic model checking it is assumed that the labelled transition system is a Continuous Time Markov Chain (CTMC).

This makes stochastic process algebras suitable high-level language for stochastic model checking.

Stochastic model checking

In stochastic model checking it is assumed that the labelled transition system is a Continuous Time Markov Chain (CTMC).

This makes stochastic process algebras suitable high-level language for stochastic model checking.

The logic is also enhanced to query not just logical behaviour (whether some property is satisfied or not) but also quantified behaviour (e.g. the probability that a property is satisfied at a particular time).

Model checking

There are two broad approaches to model checking:

- Explicit state model checking (exhaustive exploration for all possible states/executions): exact results obtained via numerical computation.
- Statistical model-checking (discrete event simulation and sampling over multiple runs): approximate results.

A logical foundation for a 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).

A logical foundation for a 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 a 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.

A logical foundation for a 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

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

Transition rates to set of processes

Definition

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

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

Interpreting PML over PEPA processes

Interpreting PML over PEPA processes

Let P be a model of a PEPA process.

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

Interpreting PML over PEPA processes

Let P be a model of a PEPA process.

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

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

Interpreting PML over PEPA processes

Let P be a model of a PEPA process.

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

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

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

Interpreting PML over PEPA processes

Let P be a model of a PEPA process.

$$
\begin{array}{lll}
P & \models \mathrm{tt} & \\
P & \models \neg F & \text{if } P \not\models F \\
P & \models F_1 \wedge F_2 & \text{if } P \models F_1 \text{ and } P \models F_2 \\
P & \models \nabla \alpha & \text{if } P \stackrel{\alpha}{\not\rightarrow}
\end{array}
$$

Interpreting PML over PEPA processes

Let P be a model of a PEPA process.

$$
\begin{array}{lll}
P & \models \mathrm{tt} & \\
P & \models \neg F & \text{if } P \not\models F \\
P & \models F_1 \wedge F_2 & \text{if } P \models F_1 \text{ and } P \models F_2 \\
P & \models \nabla\alpha & \text{if } P \stackrel{\alpha}{\not\longrightarrow} \\
P & \models \langle\alpha\rangle_\mu F & \text{if } P \stackrel{(\alpha,\nu)}{\longrightarrow} S \text{ for some } \nu \geq \mu, \\
 & & \quad \text{and for all } P' \in S, P' \models F
\end{array}
$$

Modal characterisation of strong equivalence

Let P be a model of a PEPA process. Then

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

Modal characterisation of strong equivalence

Let P be a model of a PEPA process. Then

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

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

Using PML

PML is interesting because of its characterisation of strong equivalence in PEPA, but it is not a very useful logic from a practical point of view.

Using PML

PML is interesting because of its characterisation of strong equivalence in PEPA, but it is not a very useful logic from a practical point of view.

In particular it is not supported by any model checking tools.

Using PML

PML is interesting because of its characterisation of strong equivalence in PEPA, but it is not a very useful logic from a practical point of view.

In particular it is not supported by any model checking tools.

For real use in practice we use the richer logic Continuous Stochastic Logic (CSL).

PRISM model and model checking

- Probabilistic model checking in PRISM is based on a CTMC and the logic CSL.

PRISM model and model checking

- Probabilistic model checking in PRISM is based on a CTMC and the logic CSL.
- Formally the mapping from PEPA is based on the structured operational semantics, generating the underlying CTMC in the usual way.

PRISM model and model checking

- Probabilistic model checking in PRISM is based on a CTMC and the logic CSL.
- Formally the mapping from PEPA is based on the structured operational semantics, generating the underlying CTMC in the usual way.
- In practice PEPA is an input language for PRISM with a direct mapping between PEPA components and the interacting, reactive modules of the PRISM input language.

PRISM model and model checking

- Probabilistic model checking in PRISM is based on a CTMC and the logic CSL.
- Formally the mapping from PEPA is based on the structured operational semantics, generating the underlying CTMC in the usual way.
- In practice PEPA is an input language for PRISM with a direct mapping between PEPA components and the interacting, reactive modules of the PRISM input language.
- Note, however, that this places a restriction to have synchronisations in which only one participant is active as PRISM cannot handle the apparent rate based calculations of cooperation in PEPA.

The CSL logic

The syntax of CSL is as follows:

$$\begin{array}{rcl}\phi & ::= & \texttt{true} \mid a \mid \neg\phi \mid \phi \wedge \phi \mid \\ & & \mathbf{P}_{\sim p}[\phi\ \mathbf{U}^{I}\ \phi] \mid \mathbf{S}_{\sim p}[\phi] \mid \\ & & \mathbf{R}_{\sim r}[I^{=t}] \mid \mathbf{R}_{\sim r}[C^{\leq t}] \mid \mathbf{R}_{\sim r}[\mathbf{F}\ \phi] \mid \mathbf{R}_{\sim r}[\mathbf{S}]\end{array}$$

where a is an atomic proposition, $\sim \in \{<, \leq, \geq, >\}$, $p \in [0, 1]$, I is an interval of $\mathbb{R}^{\geq 0}$ and $r, t \in \mathbb{R}^{\geq 0}$.

P and **S** are probabilistic operators which include a probabilistic bound $\sim p$.

R is a reward operator with a reward bound $\sim r$.

Probabilistic operators

A formula $\mathbf{S}_{\sim p}[\phi]$ is true in state s if the probability that the formula ϕ being satisfied in a steady state reached from state s meets the bound $\sim p$.

Probabilistic operators

A formula $\mathbf{S}_{\sim p}[\phi]$ is true in state s if the probability that the formula ϕ being satisfied in a steady state reached from state s meets the bound $\sim p$.

A formula $\mathbf{P}_{\sim p}[\phi\ \mathbf{U}^{I}\ \phi]$ is true in a state s if the probability of the formula $(\phi\ \mathbf{U}^{I}\ \phi)$ being satisfied from state s meets the bound $\sim p$.

Probabilistic operators

A formula $\mathbf{S}_{\sim p}[\phi]$ is true in state s if the probability that the formula ϕ being satisfied in a steady state reached from state s meets the bound $\sim p$.

A formula $\mathbf{P}_{\sim p}[\phi \; \mathbf{U}^{I} \; \phi]$ is true in a state s if the probability of the formula $(\phi \; \mathbf{U}^{I} \; \phi)$ being satisfied from state s meets the bound $\sim p$.

A formula of type $\phi_1 \; \mathbf{U}^{I} \; \phi_2$ is an until formula.

It is true of a path σ through the state space if, for some time instant $t \in I$, at time t in the path σ the CSL subformula ϕ_2 is true and the subformula ϕ_1 is true at all preceding time instants.

The CSL Reward operator

The CSL reward operator **R** is used to express properties concerning the expected value of rewards.

The CSL Reward operator

The CSL reward operator **R** is used to express properties concerning the expected value of rewards.

$\mathbf{R}_{\sim r}[I^{=t}]$ asserts that the expected value of the state reward at time instant t meets the bound $\sim r$.

The CSL Reward operator

The CSL reward operator **R** is used to express properties concerning the expected value of rewards.

$\mathbf{R}_{\sim r}[I^{=t}]$ asserts that the expected value of the state reward at time instant t meets the bound $\sim r$.

$\mathbf{R}_{\sim r}[C^{\leq t}]$ refers to the expected reward accumulated up until t.

The CSL Reward operator

The CSL reward operator **R** is used to express properties concerning the expected value of rewards.

$\mathbf{R}_{\sim r}[I^{=t}]$ asserts that the expected value of the state reward at time instant t meets the bound $\sim r$.

$\mathbf{R}_{\sim r}[C^{\leq t}]$ refers to the expected reward accumulated up until t.

$\mathbf{R}_{\sim r}[\mathbf{F}\ \phi]$ asserts that the expected reward accumulated before a state satisfying ϕ is reached meets the bound $\sim r$.

The CSL Reward operator

The CSL reward operator **R** is used to express properties concerning the expected value of rewards.

$\mathbf{R}_{\sim r}[I^{=t}]$ asserts that the expected value of the state reward at time instant t meets the bound $\sim r$.

$\mathbf{R}_{\sim r}[C^{\leq t}]$ refers to the expected reward accumulated up until t.

$\mathbf{R}_{\sim r}[\mathbf{F}\ \phi]$ asserts that the expected reward accumulated before a state satisfying ϕ is reached meets the bound $\sim r$.

$\mathbf{R}_{\sim r}[\mathbf{S}]$ asserts that the long-run/steady state expected reward meets the bound $\sim r$.

Example CSL formulae

- $\mathbf{P}_{>0.9}[\texttt{true}\ \mathbf{U}^{[0,4.5]}\ \mathit{served}]$ — the probability that a request is served within the first 4.5 seconds is greater than 0.9;

Example CSL formulae

- $\mathbf{P}_{>0.9}[\texttt{true}\ \mathbf{U}^{[0,4.5]}\ \mathit{served}]$ — the probability that a request is served within the first 4.5 seconds is greater than 0.9;
- $\mathbf{P}_{\leq 0.1}[\texttt{true}\ \mathbf{U}^{[10,\infty)}\ \mathit{error}]$ — the probability that an error occurs after 10 seconds of operation is at most 0.1;

Example CSL formulae

- $\mathbf{P}_{>0.9}[\texttt{true}\ \mathbf{U}^{[0,4.5]}\ \mathit{served}]$ — the probability that a request is served within the first 4.5 seconds is greater than 0.9;
- $\mathbf{P}_{\leq 0.1}[\texttt{true}\ \mathbf{U}^{[10,\infty)}\ \mathit{error}]$ — the probability that an error occurs after 10 seconds of operation is at most 0.1;
- $\mathit{down} \longrightarrow \mathbf{P}_{>0.75}[\neg \mathit{fail}\ \mathbf{U}^{[1,2]}\ \mathit{up}]$ — when a shutdown occurs, the probability of system recovery being completed in between 1 and 2 hours without further failures occurring is greater than 0.75;

Example CSL formulae

- $\mathbf{P}_{>0.9}[\mathtt{true}\ \mathbf{U}^{[0,4.5]}\ \mathit{served}]$ — the probability that a request is served within the first 4.5 seconds is greater than 0.9;
- $\mathbf{P}_{\leq 0.1}[\mathtt{true}\ \mathbf{U}^{[10,\infty)}\ \mathit{error}]$ — the probability that an error occurs after 10 seconds of operation is at most 0.1;
- $\mathit{down} \longrightarrow \mathbf{P}_{>0.75}[\neg \mathit{fail}\ \mathbf{U}^{[1,2]}\ \mathit{up}]$ — when a shutdown occurs, the probability of system recovery being completed in between 1 and 2 hours without further failures occurring is greater than 0.75;
- $\mathbf{S}_{<0.01}[\mathit{insufficient\ routers}]$ — in the long-run, the probability that an inadequate number of routers are operational is less than 0.01.

Computation in PRISM

The underlying computation in PRISM for explicit state model checking involves a combination of:

graph-theoretical algorithms, for conventional temporal logic model checking and qualitative probabilistic model checking;

numerical computation, for quantitative probabilistic model checking, i.e. calculation of probabilities and reward values.

Computation in PRISM

Graph algorithms are used to find the satisfiability set for each formula ϕ: $Sat(\phi) = \{s \in S \mid s \models \phi\}$.

- $Sat(\texttt{true}) = S$
- $Sat(a) = \{s \mid a \in L(s)\}$
- $Sat(\neg\phi) = S \setminus Sat(\phi)$
- $Sat(\phi \wedge \psi) = Sat(\phi) \cap Sat(\psi)$
- $Sat(\mathbf{P}_{\sim p}[\phi]) = \{s \in S \mid Prob^{C}(s, \phi) \sim p\}$
- $Sat(\mathbf{S}_{\sim p}[\psi]) = \{s \in S \mid \sum_{s' \models \psi} \pi_s^C(s') \sim p\}$.

Statistical model checking

Symbolic model checking works very well provided it is possible to explicitly build the entire state space.

Statistical model checking

Symbolic model checking works very well provided it is possible to explicitly build the entire state space.

Unfortunately, state space explosion means that this is not always possible. In these cases the most commonly used alternative is statistical model checking.

Statistical model checking

Symbolic model checking works very well provided it is possible to explicitly build the entire state space.

Unfortunately, state space explosion means that this is not always possible. In these cases the most commonly used alternative is statistical model checking.

The basic idea of statistical model checking is to simulate the system for finitely many runs, and use statistics to infer whether the samples provide evidence for the satisfaction or violation of the property of interest.

Advantages of statistical model checking

- Much larger models can be handled since the state space does not need to be constructed and stored all at once.

Advantages of statistical model checking

- Much larger models can be handled since the state space does not need to be constructed and stored all at once.
- Since the approach is based on observations and samples it can be applied to any system which is executable — the underlying stochastic process does not need to be a CTMC.

Advantages of statistical model checking

- Much larger models can be handled since the state space does not need to be constructed and stored all at once.
- Since the approach is based on observations and samples it can be applied to any system which is executable — the underlying stochastic process does not need to be a CTMC.
- Since many independent samples are required it is susceptible to coarse-grained parallelization.

Advantages of statistical model checking

- Much larger models can be handled since the state space does not need to be constructed and stored all at once.
- Since the approach is based on observations and samples it can be applied to any system which is executable — the underlying stochastic process does not need to be a CTMC.
- Since many independent samples are required it is susceptible to coarse-grained parallelization.

These advantages are off-set by the disadvantage that is it an approximation compared with the exact, explicit state approach.

General framework

Consider a CTMC $\mathcal{X}$ and a property ϕ.

General framework

Consider a CTMC $\mathcal{X}$ and a property ϕ.

An execution or run of $\mathcal{X}$ is a, possibly infinite, sequence of states in $\mathcal{X}$.

General framework

Consider a CTMC $\mathcal{X}$ and a property ϕ.

An execution or run of $\mathcal{X}$ is a, possibly infinite, sequence of states in $\mathcal{X}$.

We wish to decide $\mathbf{P}_{\sim p}[\phi]$, i.e. whether $\mathcal{X}$ satisfies ϕ with probability satisfying the bound $\sim p$.

General framework

Consider a CTMC $\mathcal{X}$ and a property ϕ.

An execution or run of $\mathcal{X}$ is a, possibly infinite, sequence of states in $\mathcal{X}$.

We wish to decide $\mathbf{P}_{\sim p}[\phi]$, i.e. whether $\mathcal{X}$ satisfies ϕ with probability satisfying the bound $\sim p$.

The result of each execution is taken to be the result of a Bernoulli trial, 0 or 1, according to whether ϕ is satisfied or not.

General framework

Consider a CTMC $\mathcal{X}$ and a property ϕ.

An execution or run of $\mathcal{X}$ is a, possibly infinite, sequence of states in $\mathcal{X}$.

We wish to decide $\mathbf{P}_{\sim p}[\phi]$, i.e. whether $\mathcal{X}$ satisfies ϕ with probability satisfying the bound $\sim p$.

The result of each execution is taken to be the result of a Bernoulli trial, 0 or 1, according to whether ϕ is satisfied or not.

Let q be the probability that ϕ is satisfied, then we seek to establish if $q \sim p$.

Schematic for statistical model checking