High-level languages for fluid approximation of agent-based models

Jane Hillston
LFCS, University of Edinburgh

18th April 2016

Outline

Outline

The Discrete World View

As computer scientists we generally take a discrete view of the world.

The Discrete World View

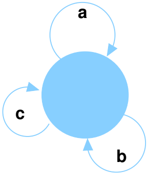
As computer scientists we generally take a discrete view of the world.

This is particularly true when we want to reason about the behaviour of systems, as most formalisms are built upon notions of states and transitions.

The Discrete World View

As computer scientists we generally take a discrete view of the world.

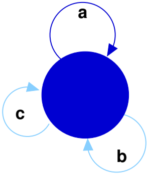
This is particularly true when we want to reason about the behaviour of systems, as most formalisms are built upon notions of states and transitions.



The Discrete World View

As computer scientists we generally take a discrete view of the world.

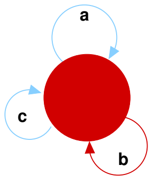
This is particularly true when we want to reason about the behaviour of systems, as most formalisms are built upon notions of states and transitions.



The Discrete World View

As computer scientists we generally take a discrete view of the world.

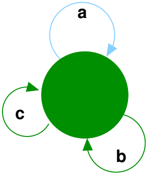
This is particularly true when we want to reason about the behaviour of systems, as most formalisms are built upon notions of states and transitions.



The Discrete World View

As computer scientists we generally take a discrete view of the world.

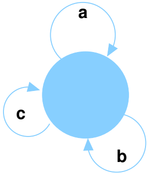
This is particularly true when we want to reason about the behaviour of systems, as most formalisms are built upon notions of states and transitions.



The Discrete World View

As computer scientists we generally take a discrete view of the world.

This is particularly true when we want to reason about the behaviour of systems, as most formalisms are built upon notions of states and transitions.



Various formalisms have been designed for capturing such behaviour.

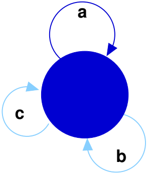
Process Algebra

- Models consist of agents which engage in actions.

$$\alpha.P$$

action type or name

agent/ component



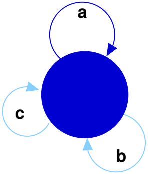
Process Algebra

- Models consist of agents which engage in actions.

$$\alpha . P$$

action type or name

agent/ component



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

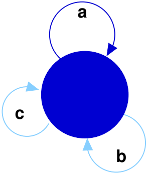
Process Algebra

- Models consist of agents which engage in actions.

$$\alpha.P$$

action type or name

agent/ component



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

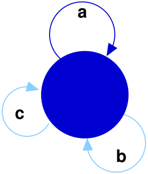
Process algebra model

Process Algebra

- Models consist of agents which engage in actions.

$$\alpha.P$$

action type or name — agent/ component



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

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

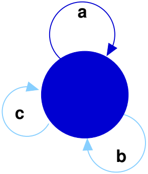
Process Algebra

- Models consist of agents which engage in actions.

$$\alpha.P$$

action type or name

agent/ component



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

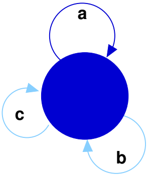
Process algebra model $\xrightarrow{\text{SOS rules}}$ Labelled transition system

Process Algebra

- Models consist of agents which engage in actions.

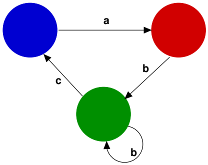
$$\alpha.P$$

action type or name → α; agent/component → P



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

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



Quantitative Modelling

Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the efficient and equitable sharing of resources. Availability and reliability modelling consider the dynamic behaviour of systems with failures and breakdowns.

Quantitative Modelling

Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the efficient and equitable sharing of resources. Availability and reliability modelling consider the dynamic behaviour of systems with failures and breakdowns.

Markovian-based discrete event models have been applied to computer systems since the mid-1960s and communication systems since the early 20th century.

Quantitative Modelling

Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the efficient and equitable sharing of resources. Availability and reliability modelling consider the dynamic behaviour of systems with failures and breakdowns.

Markovian-based discrete event models have been applied to computer systems since the mid-1960s and communication systems since the early 20th century.

Originally queueing networks were primarily used to construct models, and sophisticated analysis techniques were developed.

Quantitative Modelling

Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the efficient and equitable sharing of resources. Availability and reliability modelling consider the dynamic behaviour of systems with failures and breakdowns.

Markovian-based discrete event models have been applied to computer systems since the mid-1960s and communication systems since the early 20th century.

Originally queueing networks were primarily used to construct models, and sophisticated analysis techniques were developed.

These techniques are no longer widely applicable for expressing the dynamic behaviour observed in distributed systems with concurrent behaviour.

Formal Approaches to Quantitative Modelling

The size and complexity of real systems makes the direct construction of discrete state models costly and error-prone.

Formal Approaches to Quantitative Modelling

The size and complexity of real systems makes the direct construction of discrete state models costly and error-prone.

For the last three decades there has been substantial interest in applying formal modelling techniques enhanced with information about timing and probability.

Formal Approaches to Quantitative Modelling

The size and complexity of real systems makes the direct construction of discrete state models costly and error-prone.

For the last three decades there has been substantial interest in applying formal modelling techniques enhanced with information about timing and probability.

From these high-level system descriptions the underlying mathematical model (Continuous Time Markov Chain (CTMC)) can be automatically generated.

Formal Approaches to Quantitative Modelling

The size and complexity of real systems makes the direct construction of discrete state models costly and error-prone.

For the last three decades there has been substantial interest in applying formal modelling techniques enhanced with information about timing and probability.

From these high-level system descriptions the underlying mathematical model (Continuous Time Markov Chain (CTMC)) can be automatically generated.

Primary examples include:

- Stochastic Petri Nets and
- Stochastic/Markovian Process Algebras.

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

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

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

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

SPA MODEL

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

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

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

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

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

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

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

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

SPA MODEL $\xrightarrow{\text{SOS rules}}$ LABELLED MULTI-TRANSITION SYSTEM $\xrightarrow[\text{diagram}]{\text{state transition}}$

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

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

$$\text{SPA MODEL} \xrightarrow{\text{SOS rules}} \text{LABELLED MULTI-TRANSITION SYSTEM} \xrightarrow[\text{diagram}]{\text{state transition}} \text{CTMC } \mathbf{Q}$$

Integrated analysis

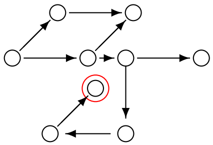
Qualitative verification can now be complemented by quantitative verification.

Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Reachability analysis

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

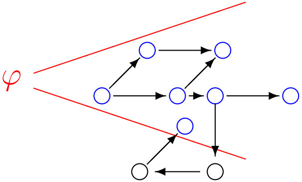


Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Model checking

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

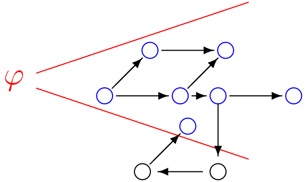


Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Model checking

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



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.

$(\alpha, f).P$	Prefix
$P_1 + P_2$	Choice
$P_1 \underset{L}{\bowtie} P_2$	Co-operation
P/L	Hiding
C	Constant

Performance Evaluation Process Algebra

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

$(\alpha, f).P$	Prefix
$P_1 + P_2$	Choice
$P_1 \underset{L}{\bowtie} P_2$	Co-operation
P/L	Hiding
C	Constant

Performance Evaluation Process Algebra

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

$(\alpha, f).P$	Prefix
$P_1 + P_2$	Choice
$P_1 \underset{L}{\bowtie} P_2$	Co-operation
P/L	Hiding
C	Constant

Performance Evaluation Process Algebra

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

$(\alpha, f).P$	Prefix
$P_1 + P_2$	Choice
$P_1 \underset{L}{\bowtie} P_2$	Co-operation
P/L	Hiding
C	Constant

Performance Evaluation Process Algebra

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

$(\alpha, f).P$	Prefix
$P_1 + P_2$	Choice
$P_1 \underset{L}{\bowtie} P_2$	Co-operation
P/L	Hiding
C	Constant

Performance Evaluation Process Algebra

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

$(\alpha, f).P$	Prefix
$P_1 + P_2$	Choice
$P_1 \underset{L}{\bowtie} P_2$	Co-operation
P/L	Hiding
C	Constant

Performance Evaluation Process Algebra

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

$(\alpha, f).P$	Prefix
$P_1 + P_2$	Choice
$P_1 \underset{L}{\bowtie} P_2$	Co-operation
P/L	Hiding
C	Constant

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

Performance Evaluation Process Algebra

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

$(\alpha, f).P$	Prefix
$P_1 + P_2$	Choice
$P_1 \underset{L}{\bowtie} P_2$	Co-operation
P/L	Hiding
C	Constant

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

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

Performance Evaluation Process Algebra

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

$(\alpha, f).P$	Prefix
$P_1 + P_2$	Choice
$P_1 \underset{L}{\bowtie} P_2$	Co-operation
P/L	Hiding
C	Constant

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

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

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

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

Apparent Rate

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

Structured Operational Semantics: Hiding

Hiding

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

Structured Operational Semantics: Hiding

Hiding

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

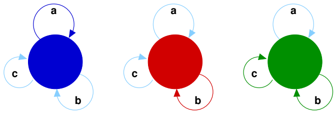
$$\frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\tau,r)} E'/L} \quad (\alpha \in L)$$

Structured Operational Semantics: Constants

Constant

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

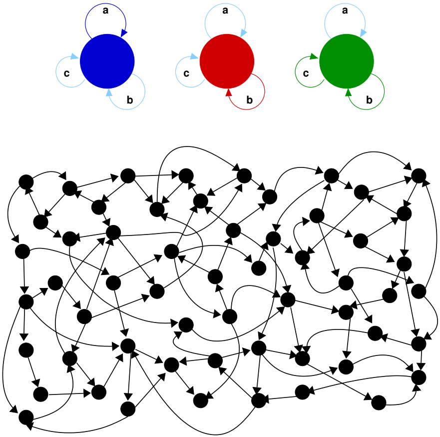
Solving discrete state models



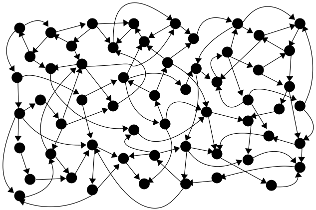
Under the SOS semantics a SPA model is mapped to a CTMC with global states determined by the local states of all the participating components.

Solving discrete state models

Under the SOS semantics a SPA model is mapped to a CTMC with global states determined by the local states of all the participating components.

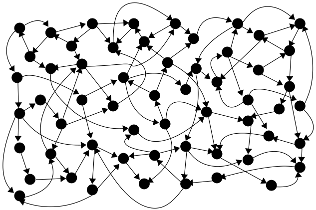


Solving discrete state models



When the size of the state space is not too large they are amenable to numerical solution (linear algebra) to determine a steady state or transient probability distribution.

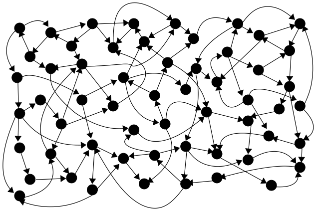
Solving discrete state models



When the size of the state space is not too large they are amenable to numerical solution (linear algebra) to determine a steady state or transient probability distribution.

$$Q = \begin{pmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,N} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,N} \\ \vdots & \vdots & & \vdots \\ q_{N,1} & q_{N,2} & \cdots & q_{N,N} \end{pmatrix}$$

Solving discrete state models



When the size of the state space is not too large they are amenable to numerical solution (linear algebra) to determine a steady state or transient probability distribution.

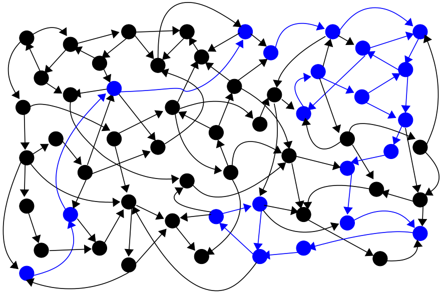
$$Q = \begin{pmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,N} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,N} \\ \vdots & \vdots & & \vdots \\ q_{N,1} & q_{N,2} & \cdots & q_{N,N} \end{pmatrix}$$

$$\pi(t) = (\pi_1(t), \pi_2(t), \ldots, \pi_N(t))$$

$$\pi(\infty)Q = 0$$

Solving discrete state models

Alternatively they may be studied using stochastic simulation. Each run generates a single trajectory through the state space. Many runs are needed in order to obtain average behaviours.



PEPA applied in practice (discrete)

- QoS protocols for mobile devices

  H.Wang, D.I.Laurenson and J.Hillston. A General Performance Evaluation Framework for Network Selection Strategies in 3G-WLAN Interworking Networks. IEEE TMC 2013.

- Disease spread within populations

  S.Benkirane, R.Norman, E.Scott and C.Shankland. Measles Epidemics and PEPA: An Exploration of Historic Disease Dynamics Using Process Algebra. Formal Methods 2012.

- Clinical pathways in hospitals

  X.Yang, R.Han, Y.Guo, J.T.Bradley, B.Cox, R.Dickinson and R.Kitney. Modelling and performance analysis of clinical pathways using the stochastic process algebra PEPA. BMC Bioinformatics 2012.

- Mobile applications

  N.Arijo, R.Heikel, M.Tribastone and S.Gilmore. Modular performance modelling for mobile applications. ICPE 2011.

- Security application: Key distribution centres

  Y.Zhao and N.Thomas. Efficient solutions of a PEPA model of a key distribution centre. Performance Evaluation 2010.

State space explosion

As the size of the state space becomes large it becomes infeasible to carry out numerical solution and extremely time-consuming to conduct stochastic simulation.

The Fluid Approximation Alternative

When there are repeated instances of agents (populations) in the system there is an alternative: fluid approximation.

The Fluid Approximation Alternative

When there are repeated instances of agents (populations) in the system there is an alternative: fluid approximation.

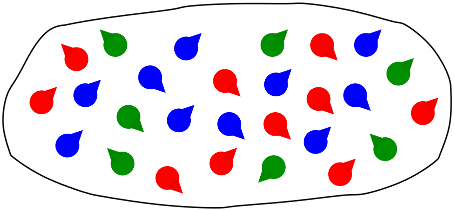
For a large class of models, just as the size of the state space becomes unmanageable, the models become amenable to an efficient, scale-free approximation.

Identity and Individuality

Population systems are constructed from many instances of a set of components.

Identity and Individuality

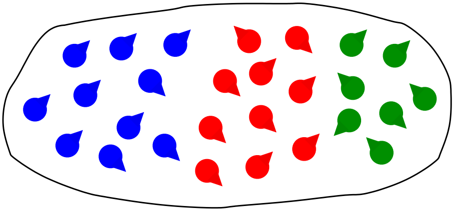
Population systems are constructed from many instances of a set of components.



If we cease to distinguish between instances of components we can form an aggregation or counting abstraction to reduce the state space.

Identity and Individuality

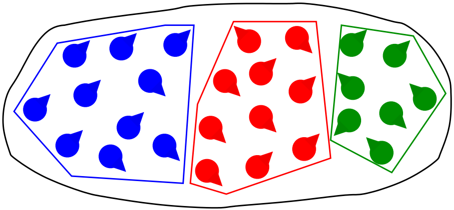
Population systems are constructed from many instances of a set of components.



If we cease to distinguish between instances of components we can form an aggregation or counting abstraction to reduce the state space.

Identity and Individuality

Population systems are constructed from many instances of a set of components.

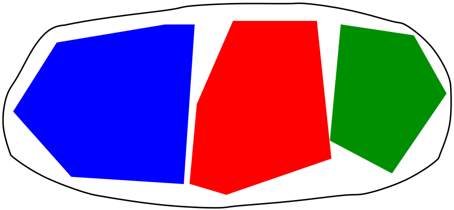


If we cease to distinguish between instances of components we can form an aggregation or counting abstraction to reduce the state space.

We may choose to disregard the identity of components.

Identity and Individuality

Population systems are constructed from many instances of a set of components.

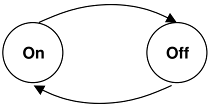


If we cease to distinguish between instances of components we can form an aggregation or counting abstraction to reduce the state space.
We may choose to disregard the identity of components.

Even better reductions can be achieved when we no longer regard the components as individuals.

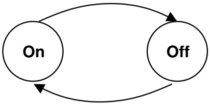
Outline

Population models — intuition



$Y(t)$

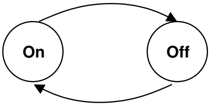
Population models — intuition

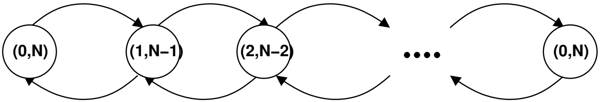


$$Y(t)$$

N copies: $Y_i^{(N)}$

Population models — intuition



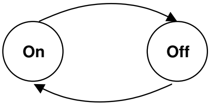


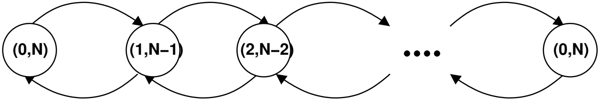
$Y(t)$

N copies: $Y_i^{(N)}$

$\mathbf{X}^{(N)}(t)$

Population models — intuition





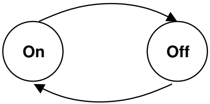
$Y(t)$

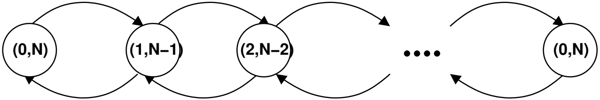
N copies: $Y_i^{(N)}$

$\mathbf{X}^{(N)}(t)$

$$X_j^{(N)} = \sum_{i=1}^{N} \mathbf{1}\{Y_i^{(N)} = j\}$$

Population models — intuition





$Y(t)$

N copies: $Y_i^{(N)}$

$\mathbf{X}^{(N)}(t)$

$$X_j^{(N)} = \sum_{i=1}^{N} \mathbf{1}\{Y_i^{(N)} = j\}$$

- $Y(t)$, $Y_i^{(N)}(t)$ and $\mathbf{X}^{(N)}(t)$ are all CTMCs;
- As N increases we get a sequence of CTMCs, $\mathbf{X}^{(N)}(t)$

Population state space

- The population process $\mathbf{X}^{(N)} = (X_1^{(N)}, \ldots, X_n^{(N)})$ has the dimension of the state space of $Y(t)$.

Population state space

- The population process $\mathbf{X}^{(N)} = (X_1^{(N)}, \ldots, X_n^{(N)})$ has the dimension of the state space of $Y(t)$.
- Importantly, its dimensions are independent of N.

Population state space

- The population process $\mathbf{X}^{(N)} = (X_1^{(N)}, \ldots, X_n^{(N)})$ has the dimension of the state space of $Y(t)$.
- Importantly, its dimensions are independent of N.
- Essentially we are making a counting abstraction and aggregation of the state space.

Population state space

- The population process $\mathbf{X}^{(N)} = (X_1^{(N)}, \ldots, X_n^{(N)})$ has the dimension of the state space of $Y(t)$.
- Importantly, its dimensions are independent of N.
- Essentially we are making a counting abstraction and aggregation of the state space.
- If we make the closed world assumption: $\sum_{j=1}^{n} X_j^{(N)} = N$

Population state space

- The population process $\mathbf{X}^{(N)} = (X_1^{(N)}, \ldots, X_n^{(N)})$ has the dimension of the state space of $Y(t)$.
- Importantly, its dimensions are independent of N.
- Essentially we are making a counting abstraction and aggregation of the state space.
- If we make the closed world assumption: $\sum_{j=1}^{n} X_j^{(N)} = N$
- N.B. PEPA models always satisfy the closed world assumption.

Population transitions

- The dynamics of the population models is expressed in terms of a set of possible transitions, $\mathcal{T}^{(N)}$.

Population transitions

- The dynamics of the population models is expressed in terms of a set of possible transitions, $\mathcal{T}^{(N)}$.
- Transitions are stochastic, and take an exponentially distributed time to happen.

Population transitions

- The dynamics of the population models is expressed in terms of a set of possible transitions, $\mathcal{T}^{(N)}$.
- Transitions are stochastic, and take an exponentially distributed time to happen.
- Their rate may depend on the current global state of the system.

Population transitions

- The dynamics of the population models is expressed in terms of a set of possible transitions, $\mathcal{T}^{(N)}$.
- Transitions are stochastic, and take an exponentially distributed time to happen.
- Their rate may depend on the current global state of the system.
- Each transition is specified by a rate function $r_\tau^{(N)}$, and by an update vector $\mathbf{v}_\tau$, specifying the impact of the event on the population vector.

Population transitions

- The dynamics of the population models is expressed in terms of a set of possible transitions, $\mathcal{T}^{(N)}$.
- Transitions are stochastic, and take an exponentially distributed time to happen.
- Their rate may depend on the current global state of the system.
- Each transition is specified by a rate function $r_\tau^{(N)}$, and by an update vector $\mathbf{v}_\tau$, specifying the impact of the event on the population vector.
- The infinitesimal generator matrix $Q^{(N)}$ of $\mathbf{X}^{(N)}(t)$ is defined as:

$$q_{\mathbf{x},\mathbf{x}'} = \sum\{r_\tau(\mathbf{x}) \mid \tau \in \mathcal{T},\ \mathbf{x}' = \mathbf{x} + \mathbf{v}_\tau\}.$$

Population models — summary of notation

Individuals

We have N individuals $Y_i^{(N)} \in S$, $S = \{1, 2, \ldots, n\}$ in the system (can have multiple classes).

Population models — summary of notation

Individuals

We have N individuals $Y_i^{(N)} \in S$, $S = \{1, 2, \dots, n\}$ in the system (can have multiple classes).

System variables

$X_j^{(N)} = \sum_{i=1}^{N} \mathbf{1}\{Y_i^{(N)} = j\}$, and $\mathbf{X}^{(N)} = (X_1^{(N)}, \dots, X_n^{(N)})$

Population models — summary of notation

Individuals

We have N individuals $Y_i^{(N)} \in S$, $S = \{1, 2, \ldots, n\}$ in the system (can have multiple classes).

System variables

$X_j^{(N)} = \sum_{i=1}^{N} \mathbf{1}\{Y_i^{(N)} = j\}$, and $\mathbf{X}^{(N)} = (X_1^{(N)}, \ldots, X_n^{(N)})$

Dynamics (system level)

$\mathbf{X}^{(N)}$ is a CTMC with transitions $\tau \in \mathcal{T}$:

$$\tau\text{: } \mathbf{X}^{(N)} \text{ to } \mathbf{X}^{(N)} + \mathbf{v}_\tau \text{ at rate } r_\tau^{(N)}(\mathbf{X})$$

Scaling Conditions

Scaling assumptions

- We have a sequence $\mathbf{X}^{(N)}$ of population CTMCs.
- We normalise such models, dividing variables by N:

$$\hat{\mathbf{X}} = \frac{\mathbf{X}}{N}$$

Scaling Conditions

Scaling assumptions

- We have a sequence $\mathbf{X}^{(N)}$ of population CTMCs.
- We normalise such models, dividing variables by N:

$$\hat{\mathbf{X}} = \frac{\mathbf{X}}{N}$$

occupancy measures

Scaling Conditions

Scaling assumptions

- We have a sequence $\mathbf{X}^{(N)}$ of population CTMCs.
- We normalise such models, dividing variables by N:

$$\hat{\mathbf{X}} = \frac{\mathbf{X}}{N} \qquad \text{occupancy measures}$$

- for each $\tau \in \mathcal{T}^{(N)}$
  - the normalised update is $\hat{\mathbf{v}} = \mathbf{v}/N$
  - there is a normalised rate function $\hat{r}_\tau(\hat{\mathbf{X}})$

Scaling Conditions

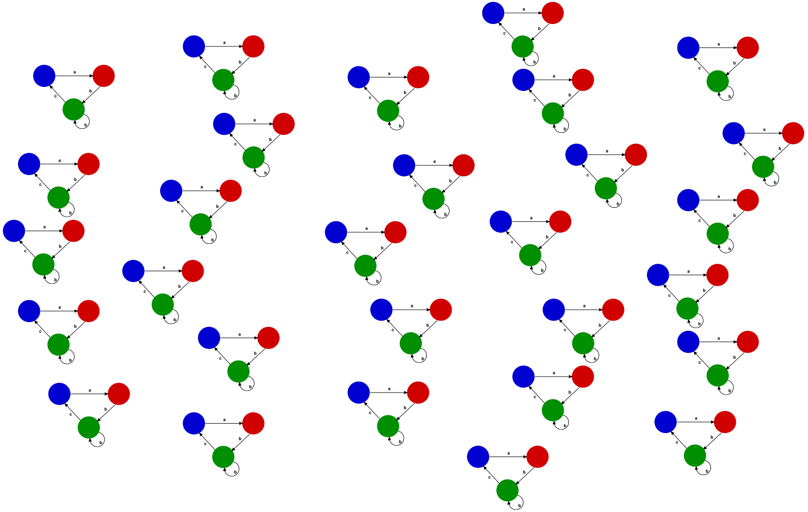
Scaling assumptions

- We have a sequence $\mathbf{X}^{(N)}$ of population CTMCs.
- We normalise such models, dividing variables by N:

$$\hat{\mathbf{X}} = \frac{\mathbf{X}}{N} \quad \text{occupancy measures}$$

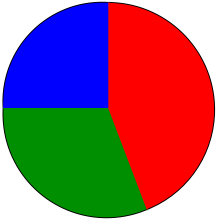
- for each $\tau \in \mathcal{T}^{(N)}$
  - the normalised update is $\hat{\mathbf{v}} = \mathbf{v}/N$
  - there is a normalised rate function $\hat{r}_\tau(\hat{\mathbf{X}})$
- $\forall \tau$ assume there exists a bounded and Lipschitz continuous function $f_\tau(\hat{\mathbf{X}})$, the limit rate function on normalised variables, independent of N, such that $\frac{1}{N}\hat{r}_\tau^{(N)}(\mathbf{x}) \to f_\tau(\mathbf{x})$ uniformly.

Normalised process — intuition



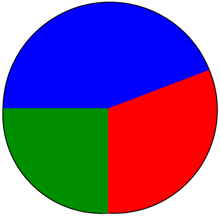
Normalised process — intuition

The whole population is represented as a single process.



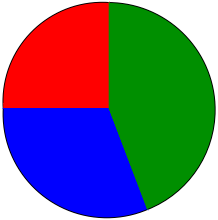
Normalised process — intuition

The whole population is represented as a single process.



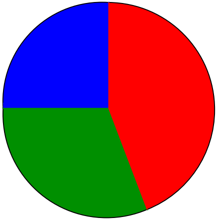
Normalised process — intuition

The whole population is represented as a single process.



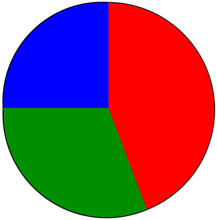
Normalised process — intuition

The whole population is represented as a single process.



Normalised process — intuition

The whole population is represented as a single process.



Even when the number of individuals varies ($N \longrightarrow \infty$) the processes remain comparable.

Drift

Drift

The drift $F^{(N)}(\hat{\mathbf{X}})$ — the mean instantaneous increment of model variables in state $\hat{\mathbf{X}}$ — is defined as

$$F^{(N)}(\hat{\mathbf{X}}) = \sum_{\tau \in \hat{\mathcal{T}}} \frac{1}{N} \mathbf{v}_\tau \, \hat{r}_\tau^{(N)}(\hat{\mathbf{X}})$$

Drift

Drift

The drift $F^{(N)}(\hat{\mathbf{X}})$ — the mean instantaneous increment of model variables in state $\hat{\mathbf{X}}$ — is defined as

$$F^{(N)}(\hat{\mathbf{X}}) = \sum_{\tau \in \hat{\mathcal{T}}} \frac{1}{N} \mathbf{v}_\tau \, \hat{r}_\tau^{(N)}(\hat{\mathbf{X}})$$

Limit Drift

Let f_τ be the limit rate functions.

The limit drift of the model $\hat{\mathcal{X}}^{(N)}$ is

$$F(\hat{\mathbf{X}}) = \sum_{\tau \in \hat{\mathcal{T}}} \mathbf{v}_\tau f_\tau(\hat{\mathbf{X}}),$$

and $F^{(N)}(\mathbf{x}) \to F(\mathbf{x})$ uniformly as $N \longrightarrow \infty$.

Fluid ODE and Fluid approximation theorem

Fluid ODE

The fluid ODE is

$$\frac{d\mathbf{x}}{dt} = F(\mathbf{x}), \quad \textit{with } \mathbf{x}(0) = \mathbf{x_0} \in S.$$

Since F is Lipschitz (all f_τ are), this ODE has a unique solution $\mathbf{x}(t)$ starting from $\mathbf{x_0}$.

Deterministic Approximation Theorem (Kurtz)

Assume that $\exists\, \mathbf{x_0} \in S$ such that $\hat{\mathbf{X}}^{(N)}(0) \to \mathbf{x_0}$ in probability. Then, for any finite time horizon $T < \infty$, it holds that as $N \longrightarrow \infty$:

$$\mathbb{P}\left\{ \sup_{0 \leq t \leq T} ||\hat{\mathbf{X}}^{(N)}(t) - \mathbf{x}(t)|| > \varepsilon \right\} \to 0.$$

T.G.Kurtz. *Solutions of ordinary differential equations as limits of pure jump Markov processes.* Journal of Applied Probability, 1970.

Fluid Approximation ODEs

The fluid approximation ODEs can be interpreted in two different ways:

- as an approximation of the average of the system (usually a first order approximation). This is often termed a mean field approximation.

Fluid Approximation ODEs

The fluid approximation ODEs can be interpreted in two different ways:

- as an approximation of the average of the system (usually a first order approximation). This is often termed a mean field approximation.
- as an approximate description of system trajectories for large populations.

Fluid Approximation ODEs

The fluid approximation ODEs can be interpreted in two different ways:

- as an approximation of the average of the system (usually a first order approximation). This is often termed a mean field approximation.
- as an approximate description of system trajectories for large populations.

We focus on the second interpretation — a functional version of the Law of Large Numbers.

Fluid Approximation ODEs

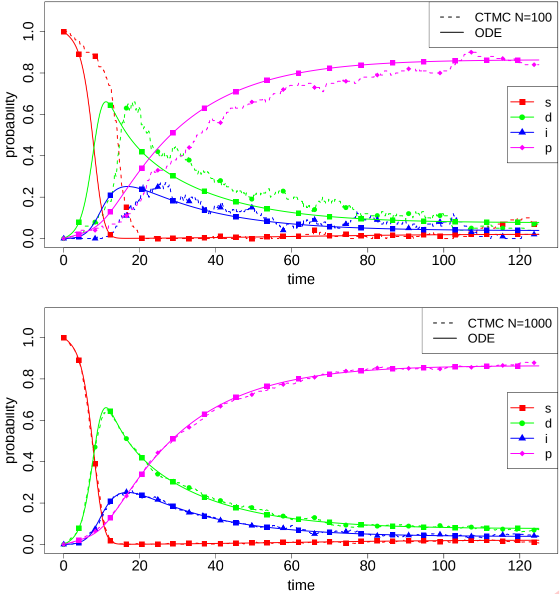
The fluid approximation ODEs can be interpreted in two different ways:

- as an approximation of the average of the system (usually a first order approximation). This is often termed a mean field approximation.
- as an approximate description of system trajectories for large populations.

We focus on the second interpretation — a functional version of the Law of Large Numbers.

Instead of having a sequence of random variables, converging to a deterministic value, here we have a sequence of CTMCs for increasing population size, which converge to a deterministic trajectory, the solution of the fluid ODE.

Illustrative trajectories



Comparison of the limit fluid ODE and a single stochastic trajectory of a network epidemic example, for total populations $N = 100$ and $N = 1000$.

Outline

Process Algebra for Population Systems

Process algebra are well-suited for constructing models of population systems:

- Developed to represent concurrent behaviour compositionally;

Process Algebra for Population Systems

Process algebra are well-suited for constructing models of population systems:

- Developed to represent concurrent behaviour compositionally;
- Represent the interactions between individuals explicitly;

Process Algebra for Population Systems

Process algebra are well-suited for constructing models of population systems:

- Developed to represent concurrent behaviour compositionally;
- Represent the interactions between individuals explicitly;
- Populations are readily and rigorously identified;

Process Algebra for Population Systems

Process algebra are well-suited for constructing models of population systems:

- Developed to represent concurrent behaviour compositionally;
- Represent the interactions between individuals explicitly;
- Populations are readily and rigorously identified;
- Stochastic extensions allow the dynamics of system behaviour to be captured;

Process Algebra for Population Systems

Process algebra are well-suited for constructing models of population systems:

- Developed to represent concurrent behaviour compositionally;
- Represent the interactions between individuals explicitly;
- Populations are readily and rigorously identified;
- Stochastic extensions allow the dynamics of system behaviour to be captured;
- Incorporate formal apparatus for reasoning about the behaviour of systems through model checking.

Process Algebra for Population Systems

Process algebra are well-suited for constructing models of population systems:

- Developed to represent concurrent behaviour compositionally;
- Represent the interactions between individuals explicitly;
- Populations are readily and rigorously identified;
- Stochastic extensions allow the dynamics of system behaviour to be captured;
- Incorporate formal apparatus for reasoning about the behaviour of systems through model checking.

The major impediment is state space explosion and fluid approximation offers a solution to that problem.

Fluid semantics for Stochastic Process Algebras

- Incorporating fluid approximation into a formal high-level language used for CTMC modelling offers quantitative scalable analysis which is immune to state space explosion.

Fluid semantics for Stochastic Process Algebras

- Incorporating fluid approximation into a formal high-level language used for CTMC modelling offers quantitative scalable analysis which is immune to state space explosion.
- Indeed, accuracy increases as the size of the populations in the model grow.

Fluid semantics for Stochastic Process Algebras

- Incorporating fluid approximation into a formal high-level language used for CTMC modelling offers quantitative scalable analysis which is immune to state space explosion.
- Indeed, accuracy increases as the size of the populations in the model grow.
- Embedding the approach in a formal language offers the possibility to establish the conditions for convergence at the language level via the semantics,

Fluid semantics for Stochastic Process Algebras

- Incorporating fluid approximation into a formal high-level language used for CTMC modelling offers quantitative scalable analysis which is immune to state space explosion.
- Indeed, accuracy increases as the size of the populations in the model grow.
- Embedding the approach in a formal language offers the possibility to establish the conditions for convergence at the language level via the semantics,
- This removes the requirement to fulfil the proof obligation on a model-by-model basis.

Fluid semantics for Stochastic Process Algebras

- Incorporating fluid approximation into a formal high-level language used for CTMC modelling offers quantitative scalable analysis which is immune to state space explosion.
- Indeed, accuracy increases as the size of the populations in the model grow.
- Embedding the approach in a formal language offers the possibility to establish the conditions for convergence at the language level via the semantics,
- This removes the requirement to fulfil the proof obligation on a model-by-model basis.
- Moreover the derivation of the ODEs can be automated in the implementation of the language.

Multiple agents

Kurtz's Theorem is based on the notion of a single agent class — many instances of one sequential component.

Multiple agents

Kurtz's Theorem is based on the notion of a single agent class — many instances of one sequential component.

But in a process algebra model we typically work with multiple components composed to evolve concurrently.

Multiple agents

Kurtz's Theorem is based on the notion of a single agent class — many instances of one sequential component.

But in a process algebra model we typically work with multiple components composed to evolve concurrently.

We construct a single agent class in the population CTMC but partition the state space S into subsets, each of which represents the states of a distinct component, and such that there are no transitions between subsets.

Multiple agents

Kurtz's Theorem is based on the notion of a single agent class — many instances of one sequential component.

But in a process algebra model we typically work with multiple components composed to evolve concurrently.

We construct a single agent class in the population CTMC but partition the state space S into subsets, each of which represents the states of a distinct component, and such that there are no transitions between subsets.

The agents whose initial state is in each subset correspond to that component.

Deriving a Fluid Approximation of a PEPA model

The aim is to represent the CTMC implicitly (avoiding state space explosion), and to generate the set of ODEs which are the fluid limit of that CTMC.

Deriving a Fluid Approximation of a PEPA model

The aim is to represent the CTMC implicitly (avoiding state space explosion), and to generate the set of ODEs which are the fluid limit of that CTMC.

The existing SOS semantics is not suitable for this purpose because it constructs the state space of the CTMC explicitly.

Deriving a Fluid Approximation of a PEPA model

The aim is to represent the CTMC implicitly (avoiding state space explosion), and to generate the set of ODEs which are the fluid limit of that CTMC.

The existing SOS semantics is not suitable for this purpose because it constructs the state space of the CTMC explicitly.

$$\text{PEPA MODEL} \xrightarrow{\text{SOS rules}} \text{LABELLED MULTI-TRANSITION SYSTEM} \xrightarrow[\text{diagram}]{\text{state transition}} \text{CTMC } \mathbf{Q}$$

Deriving a Fluid Approximation of a PEPA model

The aim is to represent the CTMC implicitly (avoiding state space explosion), and to generate the set of ODEs which are the fluid limit of that CTMC.

The existing SOS semantics is not suitable for this purpose because it constructs the state space of the CTMC explicitly.

We define a structured operational semantics which defines the possible transitions of an arbitrary abstract state and from this derive the ODEs.

Deriving a Fluid Approximation of a PEPA model

The aim is to represent the CTMC implicitly (avoiding state space explosion), and to generate the set of ODEs which are the fluid limit of that CTMC.

The existing SOS semantics is not suitable for this purpose because it constructs the state space of the CTMC explicitly.

We define a structured operational semantics which defines the possible transitions of an arbitrary abstract state and from this derive the ODEs.

$$\text{PEPA MODEL} \xrightarrow{\text{SOS rules}} \text{SYMBOLIC LABELLED TRANSITION SYSTEM} \xrightarrow[\text{functions}]{\text{generator}} \text{ABSTRACT CTMC } \mathbf{Q} \text{ or ODEs } F_{\mathcal{M}}(x)$$

Fluid Structured Operational Semantics

In order to get to the implicit representation of the CTMC we need to:

Fluid Structured Operational Semantics

In order to get to the implicit representation of the CTMC we need to:

1. Remove excess components to identify the counting abstraction of the process (Context Reduction)

Fluid Structured Operational Semantics

In order to get to the implicit representation of the CTMC we need to:

1. Remove excess components to identify the counting abstraction of the process (Context Reduction)
2. Collect the transitions of the reduced context as symbolic updates on the state representation (Jump Multiset)

Fluid Structured Operational Semantics

In order to get to the implicit representation of the CTMC we need to:

1. Remove excess components to identify the counting abstraction of the process (Context Reduction)
2. Collect the transitions of the reduced context as symbolic updates on the state representation (Jump Multiset)
3. Calculate the rate of the transitions in terms of an arbitrary state of the CTMC.

Fluid Structured Operational Semantics

In order to get to the implicit representation of the CTMC we need to:

1. Remove excess components to identify the counting abstraction of the process (Context Reduction)
2. Collect the transitions of the reduced context as symbolic updates on the state representation (Jump Multiset)
3. Calculate the rate of the transitions in terms of an arbitrary state of the CTMC.

Once this is done we can extract the vector field $F_{\mathcal{M}}(x)$ from the jump multiset, under the assumption that the population size tends to infinity.

M.Tribastone, S.Gilmore and J.Hillston. Scalable Differential Analysis of Process Algebra Models. IEEE TSE 2012.

Consistency results

- The vector field $\mathcal{F}(x)$ is Lipschitz continuous i.e. all the rate functions governing transitions in the process algebra satisfy local continuity conditions.

Consistency results

- The vector field $\mathcal{F}(x)$ is Lipschitz continuous i.e. all the rate functions governing transitions in the process algebra satisfy local continuity conditions.
- Thus the hypotheses of the Deterministic Approximation Theorem are satisfied.

Consistency results

- The vector field $\mathcal{F}(x)$ is Lipschitz continuous i.e. all the rate functions governing transitions in the process algebra satisfy local continuity conditions.
- Thus the hypotheses of the Deterministic Approximation Theorem are satisfied.
- The generated ODEs are the fluid limit of the family of CTMCs and so approximate the discrete behaviour as the size of the system grows.

Consistency results

- The vector field $\mathcal{F}(x)$ is Lipschitz continuous i.e. all the rate functions governing transitions in the process algebra satisfy local continuity conditions.
- Thus the hypotheses of the Deterministic Approximation Theorem are satisfied.
- The generated ODEs are the fluid limit of the family of CTMCs and so approximate the discrete behaviour as the size of the system grows.
- Moreover Lipschitz continuity of the vector field guarantees existence and uniqueness of the solution to the initial value problem.

M.Tribastone, S.Gilmore and J.Hillston. Scalable Differential Analysis of Process Algebra Models. IEEE TSE 2012.

Quantitative properties

The derived vector field $\mathcal{F}(x)$, gives an approximation of the expected count for each population over time.

Quantitative properties

The derived vector field $\mathcal{F}(x)$, gives an approximation of the expected count for each population over time.

This has been extended in a number of ways:

- Fluid rewards which can be safely calculated from the fluid expectation trajectories.

M.Tribastone, J.Ding, S.Gilmore and J.Hillston. Fluid Rewards for a Stochastic Process Algebra. IEEE TSE 2012.

Quantitative properties

The derived vector field $\mathcal{F}(x)$, gives an approximation of the expected count for each population over time.

This has been extended in a number of ways:

- Fluid rewards which can be safely calculated from the fluid expectation trajectories.

M.Tribastone, J.Ding, S.Gilmore and J.Hillston. Fluid Rewards for a Stochastic Process Algebra. IEEE TSE 2012.

- Vector fields have been defined to approximate higher moments.

R.A.Hayden and J.T.Bradley. A fluid analysis framework for a Markovian process algebra. TCS 2010.

Quantitative properties

The derived vector field $\mathcal{F}(x)$, gives an approximation of the expected count for each population over time.

This has been extended in a number of ways:

- Fluid rewards which can be safely calculated from the fluid expectation trajectories.

M.Tribastone, J.Ding, S.Gilmore and J.Hillston. Fluid Rewards for a Stochastic Process Algebra. IEEE TSE 2012.

- Vector fields have been defined to approximate higher moments.

R.A.Hayden and J.T.Bradley. A fluid analysis framework for a Markovian process algebra. TCS 2010.

- Fluid approximation of passage times have been defined.

R.A.Hayden, A.Stefanek and J.T.Bradley. Fluid computation of passage-time distributions in large Markov models. TCS 2012.

PEPA applied in practice (continuous)

- Denial of service attacks

  S.Gilmore, J.Hillston and N.Zon. Abstraction Interpretation of PEPA Models. Semantics, Logics and Calculi 2016.

- Emergency egress

  A.Kumar Singh and S. Kaur. Analysis of Emergency Evacuation of Buildings using PEPA. IDCDIT 2015.

  M.Massink, D.Latella, A.Bracciali and M.Harrion. A scalable fluid flow process algebraic approach to emergency egress. SEFM 2010.

- Security protocols Y.Zhao and N.Thomas. A Simplified Solution of a PEPA model of the Kerberos Protocol. CyberC 2011.

- Energy considerations in computing systems

  A.Stefanek, R.A.Hayden and J.T.Bradley. Fluid computation of the performance/energy trade-off in large scale Markov models. Performance Evaluation Review 2011.

Outline

Conclusions

- Many interesting and important systems can be regarded as examples of collective dynamics and emergent behaviour.

Conclusions

- Many interesting and important systems can be regarded as examples of collective dynamics and emergent behaviour.
- Process algebras, such as PEPA, are well-suited to modelling the behaviour of such systems in terms of the individuals and their interactions.

Conclusions

- Many interesting and important systems can be regarded as examples of collective dynamics and emergent behaviour.
- Process algebras, such as PEPA, are well-suited to modelling the behaviour of such systems in terms of the individuals and their interactions.
- Continuous approximation allows a rigorous mathematical analysis of the average behaviour of such systems.

Conclusions

- Many interesting and important systems can be regarded as examples of collective dynamics and emergent behaviour.
- Process algebras, such as PEPA, are well-suited to modelling the behaviour of such systems in terms of the individuals and their interactions.
- Continuous approximation allows a rigorous mathematical analysis of the average behaviour of such systems.
- This alternative view of systems has opened up many and exciting new research directions.

Thanks

Thanks

Some of this work was carried out under the auspices of the QUANTICOL project funded by the FET-Proactive FoCAS initiative:

quanticol

www.quanticol.eu