Performance Analysis

Peter Harrison and Jeremy Bradley

Room 372. Email: jb@doc.ic.ac.uk

Department of Computing, Imperial College London

Produced with prosper and LaTeX

The story so far...

- In the "beginning" there were birth–death processes
- ...and Markov chains
- Everything was Markovian...
- ...most analysis applied to small Markovian systems or infinite queues
- We *now* have tools that can analyse Markov chains with 100 million states and semi-Markov Processes with $\sim$20 million states

An exponential distribution

- If $X \sim \exp(\lambda)$ then:
  - Probability density function (PDF)

$$f_X(t) = \lambda e^{-\lambda t}$$

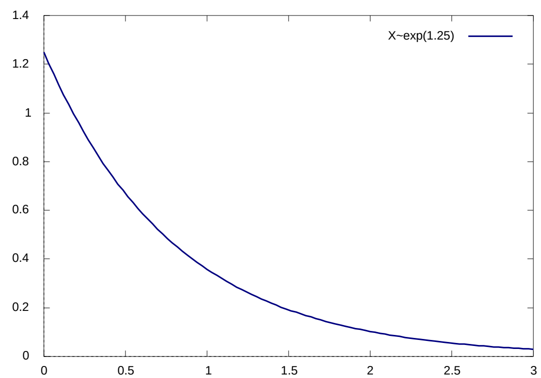
  - Cumulative density function (CDF)

$$F_X(t) = \mathbb{P}(X \leq t) = \int_0^t f_X(u)\,\mathrm{d}u = 1 - e^{-\lambda t}$$

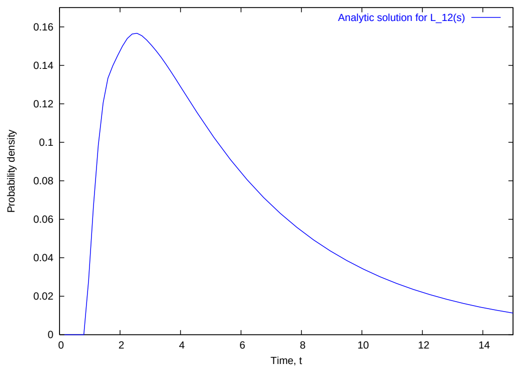
  - Laplace transform of PDF

$$L_X(s) = \frac{\lambda}{\lambda + s}$$

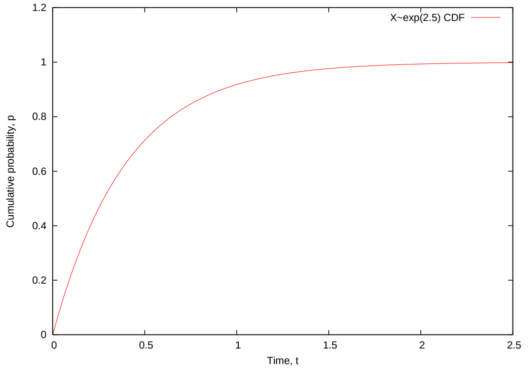
An exponential distribution



A non-exponential distribution



An exponential CDF

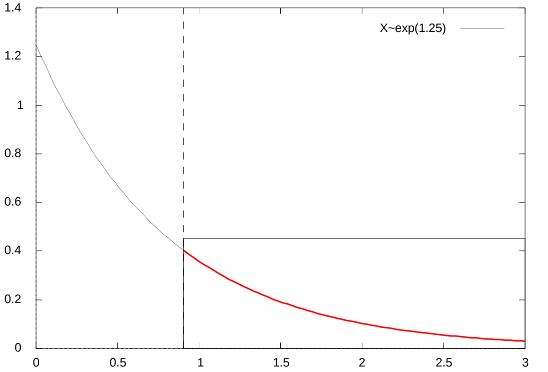


Memoryless property

- The exponential distribution is unique by being *memoryless*
  - i.e. if you interrupt an exponential event, the remaining time is also exponential
  - Let $X \sim \exp(\lambda)$ and at time, t', where $X > t'$, let $Y = X - t'$ is the distribution of the *remaining time*:

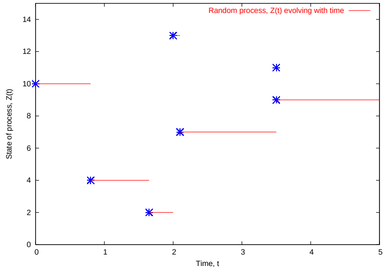
$$f_{(Y|X>t')}(t) = f_X(t)$$

Memoryless property



So what is a stochastic process...

- A stochastic process is a set of random variables
  - Discrete: $\{Z_n \ : \ n \in \mathbb{N}\}$, e.g. DTMC
  - Continuous: $\{Z(t) \ : \ t \geq 0\}$. e.g. CTMC, SMP



PEPA

- PEPA is a language for describing systems which are composed of individual continuous time Markov chains
- PEPA is useful because:
  - it is a formal, algebraic description of a system
  - it is compositional
  - it is parsimonious (succinct)
  - it is easy to learn!
  - it is used in research and in industry

Tool Support

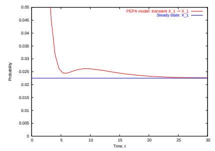
- PEPA has several methods of execution and analysis, through comprehensive tool support:
  - PEPA Workbench: Edinburgh
  - Möbius: Urbana-Champaign, Illinois
  - PRISM: Birmingham
  - ipc: Imperial College London

Types of Analysis

Steady-state and transient analysis in PEPA:

$$\begin{aligned}
\mathrm{A1} &\stackrel{\mathrm{def}}{=} (\mathrm{start}, r_1).\mathrm{A2} + (\mathrm{pause}, r_2).\mathrm{A3} \\
\mathrm{A2} &\stackrel{\mathrm{def}}{=} (\mathrm{run}, r_3).\mathrm{A1} + (\mathrm{fail}, r_4).\mathrm{A3} \\
\mathrm{A3} &\stackrel{\mathrm{def}}{=} (\mathrm{recover}, r_1).\mathrm{A1} \\
\mathrm{AA} &\stackrel{\mathrm{def}}{=} (\mathrm{run}, \top).(\mathrm{alert}, r_5).\mathrm{AA} \\
\mathrm{Sys} &\stackrel{\mathrm{def}}{=} \mathrm{AA} \underset{\{run\}}{\bowtie} \mathrm{A1}
\end{aligned}$$

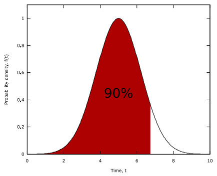
$\Rightarrow$



Passage-time Quantiles

Extract a passage-time density from a PEPA model:

$$\begin{aligned}
\mathrm{A1} &\stackrel{\mathrm{def}}{=} (\mathrm{start}, r_1).\mathrm{A2} + (\mathrm{pause}, r_2).\mathrm{A3} \\
\mathrm{A2} &\stackrel{\mathrm{def}}{=} (\mathrm{run}, r_3).\mathrm{A1} + (\mathrm{fail}, r_4).\mathrm{A3} \\
\mathrm{A3} &\stackrel{\mathrm{def}}{=} (\mathrm{recover}, r_1).\mathrm{A1} \\
\mathrm{AA} &\stackrel{\mathrm{def}}{=} (\mathrm{run}, \top).(\mathrm{alert}, r_5).\mathrm{AA} \\
\mathrm{Sys} &\stackrel{\mathrm{def}}{=} \mathrm{AA} \underset{\{run\}}{\bowtie} \mathrm{A1}
\end{aligned} \quad \Rightarrow$$



PEPA Syntax

Syntax:

$$P \ ::= \ (a, \lambda).P \ \big| \ P + P \ \big| \ P \underset{L}{\bowtie} P \ \big| \ P/L \ \big| \ A$$

- Action prefix: $(a, \lambda).P$
- Competitive choice: $P_1 + P_2$
- Cooperation: $P_1 \underset{L}{\bowtie} P_2$
- Action hiding: P/L
- Constant label: A

Prefix: $(a, \lambda).A$

- Prefix is used to describe a process that evolves from one state to another by *emitting* or *performing* an action
- Example:

$$P \stackrel{\mathrm{def}}{=} (a, \lambda).A$$

  ...means that the process P evolves with rate λ to become process A, by emitting an a-action
- λ is an exponential rate parameter
- This is also be written:

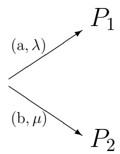
$$P \xrightarrow{(a,\lambda)} A$$

Choice: $P_1 + P_2$

- PEPA uses a type of choice known as *competitive choice*
- Example:

$$P \stackrel{\mathrm{def}}{=} (a, \lambda).P_1 + (b, \mu).P_2$$

  ...means that P can evolve *either* to produce an a-action with rate λ *or* to produce a b-action with rate μ
- In state-transition terms, P



Choice: $P_1 + P_2$

- $P \stackrel{\text{def}}{=} (a, \lambda).P_1 + (b, \mu).P_2$
- This is competitive choice since:
  - P_1 and P_2 are in a *race condition* – the first one to perform an a or a b will dictate the direction of choice for $P_1 + P_2$
- What is the probability that we see an a-action?

Cooperation: $P_1 \bowtie_L P_2$

- $\bowtie_L$ defines concurrency and communication within PEPA
- The L in $P_1 \bowtie_L P_2$ defines the set of actions over which two components are to cooperate
- Any other actions that P_1 and P_2 can do, not mentioned in L, can happen independently
- If $a \in L$ and P_1 enables an a, then P_1 has to wait for P_2 to enable an a before the cooperation can proceed
- Easy source of deadlock!

Cooperation: $P_1 \bowtie_L P_2$

- If $P_1 \xrightarrow{(a,\lambda)} P_1'$ and $P_2 \xrightarrow{(a,\top)} P_2'$ then:

$$P_1 \bowtie_{\{a\}} P_2 \xrightarrow{(a,\lambda)} P_1' \bowtie_{\{a\}} P_2'$$

- $\top$ represents a passive rate which, in the cooperation, inherits the λ-rate of from P_1
- If both rates are specified and the only a-evolutions allowed from P_1 and P_2 are, $P_1 \xrightarrow{(a,\lambda)} P_1'$ and $P_2 \xrightarrow{(a,\mu)} P_2'$ then:

$$P_1 \bowtie_{\{a\}} P_2 \xrightarrow{(a,\min(\lambda,\mu))} P_1' \bowtie_{\{a\}} P_2'$$

Cooperation: $P_1 \bowtie_L P_2$

- The general cooperation case is where:
  - P_1 enables m a-actions
  - P_2 enables n a-actions

  at the moment of cooperation
- ...in which case there are mn possible transitions for $P_1 \bowtie_{\{a\}} P_2$
- $P_1 \bowtie_{\{a\}} P_2 \xrightarrow{(a,R)}$ where $R = \frac{\lambda}{r_a(P_1)} \frac{\mu}{r_a(P_2)} \min(r_a(P_1), r_a(P_2))$
- More on this later...

Hiding: P/L

- Used to turn observable actions in P into hidden or silent actions in P/L
- L defines the set of actions to hide
- If $P \xrightarrow{(a,\lambda)} P'$:

$$P/\{a\} \xrightarrow{(\tau,\lambda)} P'/\{a\}$$

- τ is the *silent* action
- Used to hide complexity and create a component interface
- Cooperation on τ not allowed

Constant: A

- Used to define components labels, as in:
  - $P \stackrel{\text{def}}{=} (a,\lambda).P'$
  - $Q \stackrel{\text{def}}{=} (q,\mu).W$
- P,P', Q and W are all constants

Steady-state reward vectors

- Reward vectors are a way of relating the analysis of the CTMC back to the PEPA model
- A reward vector is a vector, $\vec{r}$, which expresses a looked-for property in the system:
  - e.g. utilisation, loss, delay, mean buffer length
- To find the reward value of this property at steady state – need to calculate:

$$\text{reward} = \vec{\pi} \cdot \vec{r}$$

Constructing reward vectors

- Typically reward vectors match the states where particular actions are enabled in the PEPA model

$$\begin{aligned} Client &= (use, \top).(think, \mu).Client \\ Server &= (use, \lambda).(swap, \gamma).Server \\ Sys &= Client \underset{use}{\bowtie} Server \end{aligned}$$

- There are 4 states – enumerated as $1 : (C, S)$, $2 : (C', S')$, $3 : (C, S')$ and $4 : (C', S)$

Constructing reward vectors

- If we want to measure *server usage* in the system, we would reward states in the global state space where the action *use* is enabled or active
- Only the state $1 : (C, S)$ enables *use*
- So we set $r_1 = 1$ and $r_i = 0$ for $2 \leq i \leq 4$, giving:

$$\vec{r} = (1, 0, 0, 0)$$

- These are typical *action-enabled* rewards, where the result of $\vec{r} \cdot \vec{\pi}$ is a probability

Mean Occupation as a Reward

- Quantities such as mean buffer size can also be expressed as rewards

$$\begin{aligned}
B_0 &= (arrive, \lambda).B_1 \\
B_1 &= (arrive, \lambda).B_2 + (service, \mu).B_0 \\
B_2 &= (arrive, \lambda).B_3 + (service, \mu).B_1 \\
B_3 &= (service, \mu).B_2
\end{aligned}$$

- For this M/M/1/3 queue, number of states is 4

Mean Occupation as a Reward

- Having a reward vector which reflects the number of elements in the queue will give the mean buffer occupation for M/M/1/3
- i.e. set $\vec{r} = (0, 1, 2, 3)$ such that:

$$\text{mean buffer size} = \vec{\pi} \cdot \vec{r} = \sum_{i=0}^{3} \pi_i r_i$$

Transient rewards

- For the same reward vector, $\vec{r}$
  - If we have a transient function $\vec{\pi}(t)$, such that:

$$\pi_i(t) = \mathbb{P}(\text{in state } i \text{ at time } t)$$

  - Can construct a time-based reward, $r(t)$, in similar fashion:

$$r(t) = \vec{r} \cdot \vec{\pi}(t)$$

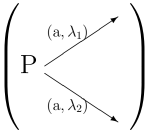
Apparent Rate

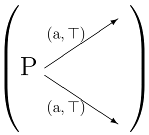
- Apparent rate of a component P is given by $r_a(\mathrm{P})$
- Apparent rate describes the overall observed rate that P performs an a-action
- Apparent rate is given by:

$$r_a(\mathrm{P}) = \sum_{\mathrm{P} \xrightarrow{(a,\lambda_i)}} \lambda_i$$

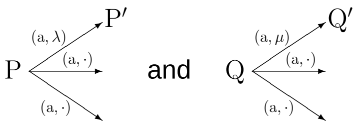
- Note: $\lambda + \top$ is forbidden by the apparent rate calculation

Apparent Rate Examples

- $r_a(\mathrm{P} \xrightarrow{(a,\lambda)}) = \lambda$
- $r_a(\mathrm{P} \xrightarrow{(a,\top)}) = \top$
- r_a  $= \lambda_1 + \lambda_2$

- r_a  $= 2\top$

Synchronisation Rate

- In PEPA, when synchronising two model components, P and Q where both P and Q enable many a-actions:
  - 

- The synchronised rate for $\mathrm{P} \underset{\{a\}}{\bowtie} \mathrm{Q} \xrightarrow{(a,R)} \mathrm{P}' \underset{\{a\}}{\bowtie} \mathrm{Q}'$ is:

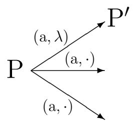
$$R = \frac{\lambda}{r_a(\mathrm{P})} \frac{\mu}{r_a(\mathrm{Q})} \min(r_a(\mathrm{P}), \mathrm{r_a}(\mathrm{Q}))$$

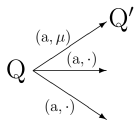
Apparent Rate Rules

- In PEPA, rate λ is drawn from the set: $\lambda \in \mathbb{R}^+ \cup \{n\top \;:\; n \in \mathbb{Q}, n > 0\}$
- $n\top$ is shorthand for $n \times \top$
- $n\top$ for $n \neq 1$ is never used as rate in a model but will occur as result of $r_a(P)$ function
- Other $\top$-rules required:

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

Approximate Synchronisation

- Some tools such as: Möbius, PRISM, PWB use an approximate synchronisation model
- With two model components, P and Q where both P and Q enable many a-actions:
  - 

and


- The *approximated* rate for $\mathrm{P} \underset{\{a\}}{\bowtie} \mathrm{Q} \xrightarrow{(a,R)} \mathrm{P}' \underset{\{a\}}{\bowtie} \mathrm{Q}'$ is:

$$R = \min(\lambda, \mu)$$

Example

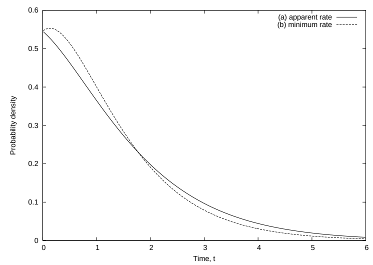
- As an example:
  - $\mathrm{Client} \stackrel{\mathrm{def}}{=} (\mathrm{data}, \lambda).\mathrm{Client}'$
  - $\mathrm{Network} \stackrel{\mathrm{def}}{=} (\mathrm{data}, \top).\mathrm{NetworkGo} + (\mathrm{data}, \top).\mathrm{NetworkStall}$
- The combination $\mathrm{Client} \underset{\{data\}}{\bowtie} \mathrm{Network}$ should evolve with an overall data rate parameter of λ
- Under the tool approximation the overall synchronised rate becomes 2λ

Results: Multiple Passive

$$\begin{aligned} \mathrm{A} &\stackrel{\mathrm{def}}{=} (\mathrm{run}, \lambda_1).(\mathrm{stop}, \lambda_2).\mathrm{A} \\ \mathrm{B} &\stackrel{\mathrm{def}}{=} (\mathrm{run}, \top).(\mathrm{pause}, \lambda_3).\mathrm{B} \\ \mathrm{Sys_A} &\stackrel{\mathrm{def}}{=} \mathrm{A} \underset{\{run\}}{\bowtie} (\mathrm{B} \mid\mid \mathrm{B}) \end{aligned}$$

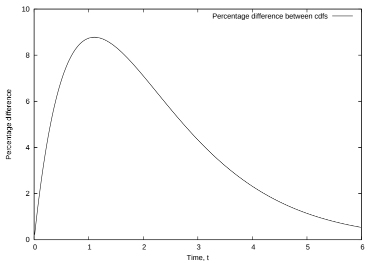
- Multiple passive ($\top$-rate) actions are enabled against a single real rate

Results: Multiple Passive



- Passage time density between consecutive stop actions

Results: Multiple Passive



- Percentage difference in CDF functions over passage time between consecutive stop actions

Multiple Active

$$\begin{aligned} \mathrm{A} &\stackrel{\mathrm{def}}{=} (\mathrm{run}, \lambda_1).(\mathrm{stop}, \lambda_2).\mathrm{A} \\ \mathrm{B} &\stackrel{\mathrm{def}}{=} (\mathrm{run}, \mu_1).(\mathrm{pause}, \lambda_3).\mathrm{B} \\ \mathrm{Sys}_{\mathrm{C}} &\stackrel{\mathrm{def}}{=} \mathrm{A} \underset{\{run\}}{\bowtie} (\mathrm{B} \mid\mid \mathrm{B}) \end{aligned}$$

- Multiple real-rate actions (in $(\mathrm{B} \mid\mid \mathrm{B})$) are synchronised against a single real-rate action (in A)

How usual is this?

- Have an explicit individual component with either:
  - $\mathrm{P} \stackrel{\mathrm{def}}{=} (\mathrm{a}, \lambda).\mathrm{P}' + (\mathrm{a}, \mu).\mathrm{P}''$ (multiple active)
  - $\mathrm{Q} \stackrel{\mathrm{def}}{=} (\mathrm{a}, \top).\mathrm{Q}' + (\mathrm{a}, \top).\mathrm{Q}''$ (multiple passive)
- ...simple multi-agent synchronisation of $\mathrm{S} \underset{\{a\}}{\bowtie} (\mathrm{R} \mid\mid \mathrm{R} \mid\mid \cdots \mid\mid \mathrm{R})$ for some S where $\mathrm{R} \stackrel{\mathrm{def}}{=} (\mathrm{a}, \top).(\mathrm{b}, \mu).\mathrm{R}'$ requires use of the full $r_a(\cdot)$ formula
- This is a very common client–server architecture

Apparent rate example

- From initial model:

$$\begin{aligned} A &\stackrel{\mathrm{def}}{=} (a, s).(b, r).A \\ B &\stackrel{\mathrm{def}}{=} (a, \top).(b, s).B + (a, \top).B \end{aligned}$$

- Rewrite as equivalent model:

$$\begin{aligned} A &\stackrel{\mathrm{def}}{=} (a, s).A' \\ A' &\stackrel{\mathrm{def}}{=} (b, r).A \\ B &\stackrel{\mathrm{def}}{=} (a, \top).B' + (a, \top).B \\ B' &\stackrel{\mathrm{def}}{=} (b, s).B \end{aligned}$$

State space searching

- Abbreviate $X \underset{L}{\bowtie} Y$ as (X, Y):
  - $(P, Q) \xrightarrow{(a,R_1)} (P', Q')$
  - $(P, Q) \xrightarrow{(a,R_2)} (P', Q)$
  - $(P', Q) \xrightarrow{(b,r)} (P, Q)$
  - $(P', Q') \xrightarrow{(b,s)} (P', Q)$
  - $(P', Q') \xrightarrow{(b,r)} (P, Q')$
  - $(P, Q') \xrightarrow{(b,s)} (P, Q)$
- In this case $R_1 = R_2$ (not always case):

$$\begin{aligned} R_1 = R_2 &= \frac{s}{r_a(P)} \frac{\top}{r_a(Q)} \min(r_a(P), r_a(Q)) \\ &= \frac{s}{s} \frac{\top}{2\top} \min(s, 2\top) = \frac{s}{2} \end{aligned}$$

Constructing the generator matrix

- 4 distinct states, $(P, Q), (P', Q), (P', Q'), (P, Q')$ gives generator matrix A:

$$A = \begin{pmatrix} -s & s/2 & s/2 & 0 \\ r & -r & 0 & 0 \\ 0 & s & -(s+r) & r \\ s & 0 & 0 & -s \end{pmatrix}$$

- Solve $\vec{\pi} A = 0$ subject to $\sum_i \pi_i = 1$
- $\vec{\pi} = \frac{1}{3r^2+4rs+2s^2}(2r(r+s), s(r+2s), rs, r^2)$

Equivalences relations

- Equivalence relations relate the semantics of PEPA processes
- We equate processes that behave in the same way
- Equivalence relation help compute performance measures in smaller processes
  - reducing the state space (aggregation)
  - preserving the Markov property in the smaller process
  - relating performance measures back to the original stochastic process

Lumpability

Let S be the state space of a CTMC, such that $S = \bigcup \{S_1, \ldots S_N\}$ is a partition of the CTMC.

A CTMC is *ordinarily lumpable* with respect to S if and only if for any partition S_I with states $s_i, s_j \in S_I$:

$$\mathbf{R}(s_i, S_K) = \mathbf{R}(s_j, S_K) \quad \text{for all } 0 < K \leq N$$

where:

$$\mathbf{R}(s_i, S_K) = \sum_{s_k \in S_K} \mathbf{R}(s_i, s_k)$$

Lumpability in words

- For any two states the cumulative rate of moving to any other partition is the same
- The performance measures of the CTMC and the lumped counterpart are strongly related
- The (macro)-probability of being lumped CTMC being in state S_I equals $\sum_{s_i \in S_I} \pi(s_i)$ where $\pi(s_i)$ is the probability of being in the state s_i
- We know how to express this property in a CTMCs, but how to express it in PEPA?

Relating CTMCs

Two CTMCs are *lumpable equivalent* if they have lumpable partition generating the same number of equivalence classes with the same aggregate transition rate

S and T are two state spaces of CTMCs. $S = \bigcup\{S_1, \ldots S_N\}$ and $T = \bigcup\{T_1, \ldots T_N\}$ be the respective partitions.

Two CTMCs are *lumpable equivalent* if:

$$\mathbf{R}(s_i, S_k) = \mathbf{R}(t_j, T_k) \text{ for all } 0 < K \leq N$$

for all $i \leq |S|$ such that there exists a $j \leq |T|$

Strong equivalence

Let $\mathcal{S}$ be an equivalence relation over the set of PEPA processes.

$\mathcal{S}$ is a *strong equivalence* if for any pair of processes P, Q such that $P\mathcal{S}Q$ implies that for all equivalence classes C (over the set of processes)

$$\mathbf{R}(P, C, a) = \mathbf{R}(Q, C, a)$$

where $\mathbf{R}(P, T, a) = \sum_{P \xrightarrow{(a,\cdot)} P'}^{P' \in T} \mathbf{R}(P, P')$

$P \cong Q$, if $P\mathcal{S}Q$ for some strong equivalence $\mathcal{S}$

Strong equivalence (2)

- If two processes are strongly equivalent then their CTMCs are lumpable equivalent
- For any PEPA process P:

  $$ds(P)/\cong$$

  induces a lumpable partition on the state space of the CTMC corresponding to P

Properties of Strong equivalence

If $P \cong Q$ then

1. $(a, \lambda).P \cong (a, \lambda).Q$
2. $P + R \cong Q + R$
3. $P \underset{L}{\bowtie} R \cong R \underset{L}{\bowtie} P$
4. $P/L \cong Q/L$

Very useful for modular reasoning

More properties of SE

- Choice
  - $P + Q \cong Q + P$
  - $(P + Q) + R \cong P + (Q + R)$
- Cooperation
  - $P \underset{L}{\bowtie} Q \cong Q \underset{L}{\bowtie} P$
  - $(P \underset{L}{\bowtie} Q) \underset{L}{\bowtie} R \cong P \underset{L}{\bowtie} (Q \underset{L}{\bowtie} R)$
- Hiding
  - $(P + Q)/L \cong P/L + Q/L$
  - $P/L/K \cong P/(L \cup K)$
  - $P/\emptyset \cong P$

Useful facts about queues

- Little's Law: $L = \gamma W$
  - L – mean buffer length; γ – arrival rate; W – mean waiting time/passage time
  - only applies to system in steady-state; no creating/destroying of jobs
- For M/M/1 queue:
  - λ – arrival rate, μ – service rate
  - Stability condition, $\rho = \lambda/\mu < 1$ for steady state to exist
  - Mean queue length $= \frac{\rho}{1-\rho}$
  - $\mathbb{P}(n \text{ jobs in queue at s-s}) = \rho^n(1 - \rho)$

Small bit of queueing theory

- Going to show for M/M/1 queue, that:

1. steady-state probability for buffer having k customers is:

$$\pi_k = (1 - \rho)\rho^k$$

2. mean queue length, N, at steady-state is:

$$\frac{\rho}{1 - \rho}$$

Small bit of queueing theory

- As $N = \sum_{k=0}^{\infty} k\pi_k$, we need to find π_k:
  - Derive steady-state equations from time-varying equations
  - Solve steady-state equations to get π_k
  - Calculate M/M/1 mean queue length, N
- (In what follows, remember $\rho = \lambda/\mu$)

Small bit of queueing theory

- Write down time-varying equations for M/M/1 queue:
  - At time t, in state $k = 0$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\pi_0(t) = -\lambda\pi_0(t) + \mu\pi_1(t)$$

  - At time, t, in state $k \geq 1$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\pi_k(t) = -(\lambda+\mu)\pi_k(t)+\lambda\pi_{k-1}(t)+\mu\pi_{k+1}(t)$$

Steady-state for M/M/1

- At steady-state, $\pi_k(t)$ are constant (i.e. π_k) and $\frac{\mathrm{d}}{\mathrm{d}t}\pi_k(t) = 0$ for all k

$\Rightarrow$ Balance equations:
  - $-\lambda\pi_0 + \mu\pi_1 = 0$
  - $-(\lambda + \mu)\pi_k + \lambda\pi_{k-1} + \mu\pi_{k+1} = 0 \quad : k \geq 1$
- Rearrange balance equations to give:
  - $\pi_1 = \frac{\lambda}{\mu}\pi_0 = \rho\pi_0$
  - $\pi_{k+1} = \frac{\lambda+\mu}{\mu}\pi_k - \frac{\lambda}{\mu}\pi_{k-1} \quad : k \geq 1$
- Solution: $\pi_k = \rho^k\pi_0$ (proof by induction)

Normalising to find π_0

- As these π_k are probabilities which sum to 1:

$$\sum_{k=0}^{\infty} \pi_k = 1$$

- i.e. $\sum_{k=0}^{\infty} \pi_k = \sum_{k=0}^{\infty} \rho^k \pi_0 = \frac{\pi_0}{1-\rho} = 1$

$\Rightarrow$ $\pi_0 = 1 - \rho$ as long as $\rho < 1$
- So overall steady-state formula for M/M/1 queue is:

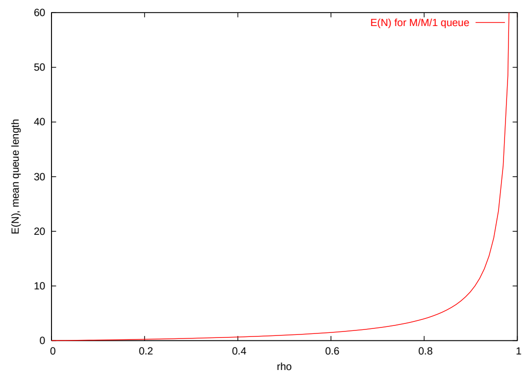
$$\pi_k = (1-\rho)\rho^k$$

M/M/1 Mean Queue Length

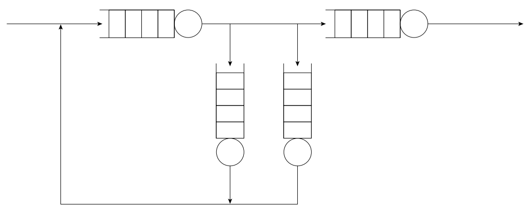
- N is queue length random variable
- N could be 0 or 1 or 2 or 3 ...
- Mean queue length is written N:

$$\begin{aligned} N &= 0.\mathbb{P}(\text{in state } 0) + 1.\mathbb{P}(\text{in state } 1) + 2.\mathbb{P}(\text{in state } 2) + \cdots \\ &= \sum_{k=0}^{\infty} k\pi_k \\ &= \pi_0 \sum_{k=0}^{\infty} k\rho^k = \pi_0\rho \sum_{k=0}^{\infty} k\rho^{k-1} = \pi_0\rho \sum_{k=0}^{\infty} \frac{\mathrm{d}}{\mathrm{d}\rho}\rho^k \\ &= \pi_0\rho \frac{\mathrm{d}}{\mathrm{d}\rho} \sum_{k=0}^{\infty} \rho^k = \pi_0\rho \frac{\mathrm{d}}{\mathrm{d}\rho}\left(\frac{1}{1-\rho}\right) \\ &= \frac{\pi_0\rho}{(1-\rho)^2} = \frac{\rho}{1-\rho} \quad \square \end{aligned}$$

M/M/1 Mean Queue Length



Queueing Networks



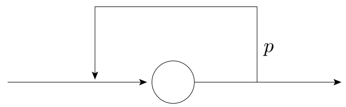
- Individual queue nodes represent contention for single resources
- A system consists of many inter-dependent resources – hence we need to reason about a *network* of queues to represent a system

Open Queueing Networks

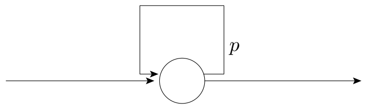
- A network of queueing nodes with inputs/outputs connected to each other
- Called an *open* queueing network (or OQN) because, traffic may enter (or leave) one or more of the nodes in the system from an external source (to an external sink)
- An open network is defined by:
  - γ_i, the exponential arrival rate from an external source
  - q_{ij}, the probability that traffic leaving node i will be routed to node j
  - μ_i exponential service rate at node i

OQN: Notation

- A node whose output can be probabilistically redirected into its input is represented as:



- or...



- probability p of being rerouted back into buffer

OQN: Network assumptions

In the following analysis, we assume:

- Exponential arrivals to network
- Exponential service at queueing nodes
- FIFO service at queueing nodes
- A network may be stable (be capable of reaching steady-state) or it may be unstable (have unbounded buffer growth)
- If a network reaches steady-state (becomes stationary), a single rate, λ_i, may be used to represent the throughput (both arrivals and departure rate) at node i

OQN: Traffic Equations

- The traffic equations for a queueing network are a linear system in λ_i
- λ_i represents the aggregate arrival rate at node i (taking into account any traffic feedback from other nodes)
- For a given node i, in an open network:

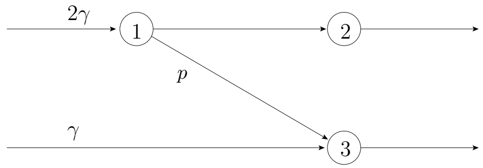
$$\lambda_i = \gamma_i + \sum_{j=1}^{n} \lambda_j q_{ji} \quad : i = 1, 2, \ldots, n$$

OQN: Traffic Equations

- Define:
  - the vector of aggregate arrival rates $\vec{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_n)$
  - the vector of external arrival rates $\vec{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_n)$
  - the matrix of routeing probabilities $Q = (q_{ij})$
- In matrix form, traffic equations become:

$$\begin{aligned} \vec{\lambda} &= \vec{\gamma} + \vec{\lambda} Q \\ &= \vec{\gamma}(I - Q)^{-1} \end{aligned}$$

OQN: Traffic Equations: example 1

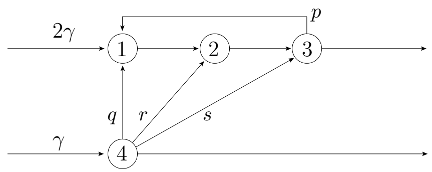


- Set up and solve traffic equations to find λ_i:

$$\vec{\lambda} = (2\gamma, 0, \gamma) + \vec{\lambda}\begin{pmatrix} 0 & 1-p & p \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

- i.e. $\lambda_1 = 2\gamma$, $\lambda_2 = (1-p)\lambda_1$, $\lambda_3 = \gamma + p\lambda_1$

OQN: Traffic Equations: example 2



- Set up and solve traffic equations to find λ_i:

$$\vec{\lambda} = (2\gamma, 0, 0, \gamma) + \vec{\lambda}\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & 0 & 0 & 0 \\ q & r & s & 0 \end{pmatrix}$$

OQN: Network stability

- Stability of network (whether it achieves steady-state) is determined by utilisation, $\rho_i < 1$ at every node i
- After solving traffic equations for λ_i, need to check that:

$$\rho_i = \frac{\lambda_i}{\mu_i} < 1 \quad : \forall i$$

Recall facts about M/M/1

- If λ is arrival rate, μ service rate then $\rho = \lambda/\mu$ is utilisation
- If $\rho < 1$, then steady state solution exists
- Average buffer length:

$$\mathbb{E}(N) = \frac{\rho}{1-\rho}$$

- Distribution of jobs in queue is:

$$\mathbb{P}(k \text{ jobs is queue at steady-state}) = (1-\rho)\rho^k$$

OQN: Jackson's Theorem

- Where node i has a service rate of μ_i, define $\rho_i = \lambda_i / \mu_i$
- If the arrival rates from the traffic equations are such that $\rho_i < 1$ for all $i = 1, 2, \ldots, n$, then the steady-state exists and:

$$\pi(r_1, r_2, \ldots, r_n) = \prod_{i=1}^{n} (1 - \rho_i)\rho_i^{r_i}$$

- This is a *product form* result!

OQN: Jackson's Theorem Results

- The marginal distribution of no. of jobs at node i is same as for isolated M/M/1 queue: $(1 - \rho)\rho^k$
- Number of jobs at any node is independent of jobs at any other node – hence *product form* solution
- Powerful since queues can be reasoned about separately for queue length – summing to give overall network queue occupancy

OQN: Mean Jobs in System

- If only need mean results, we can use Little's law to derive mean performance measures
- Product form result implies that each node can be reasoned about as separate M/M/1 queue in isolation, hence:

$$\text{Av. no. of jobs at node } i = L_i = \frac{\rho_i}{1 - \rho_i}$$

- Thus total av. number of jobs in system is:

$$L = \sum_{i=1}^{n} \frac{\rho_i}{1 - \rho_i}$$

OQN: Mean Total Waiting Time

- Applying Little's law to whole network gives:

$$L = \gamma W$$

where γ is total external arrival rate, W is mean response time.

- So mean response time from entering to leaving system:

$$W = \frac{1}{\gamma} \sum_{i=1}^{n} \frac{\rho_i}{1 - \rho_i}$$

OQN: Intermediate Waiting Times

- r_i represents the the average waiting time from arriving at node i to leaving the system
- w_i represents average response time at node i, then:

$$r_i = w_i + \sum_{j=1}^{n} q_{ij} r_j$$

- which as before gives a vector equation:

$$\begin{aligned} \vec{r} &= \vec{w} + Q\vec{r} \\ &= (I - Q)^{-1}\vec{w} \end{aligned}$$

Closed Queueing Networks

- A network of queueing nodes with inputs/outputs connected to each other
- Called a *closed* queueing network (CQN) because, traffic must stay within the system i.e. total number of customers in network buffers remains constant at all times
- Independent Delay Nodes (IDNs) used to represent an arbitrary delay in transit *between* queueing nodes
- Now routeing probabilities reflect closure of network, $\sum_{j=0}^{N} q_{ij} = 1$, for all i

CQN: State enumeration

- For K jobs in the network, the state of the CQN is represented by a tuple $(n_1, n_2, \ldots, n_N)$ where $\sum_{i=1}^{N} n_i = K$ and n_i is no. of jobs at node i
- For N queues, K customers, we have:

$$\binom{K + N - 1}{N - 1} \text{ states}$$

...obtained by looking at all possible combinations of K jobs in N queues

CQN: Traffic Equations

- As with OQN, linear traffic equations constructed for steady-state network:

$$\lambda_i = \sum_{j=1}^{N} \lambda_j q_{ji}$$

- ...in CQN case, no input traffic, thus:

$$\vec{\lambda}(I - Q) = \vec{0}$$

- Clearly $|I - Q| = 0$ and if $rnk(I - Q) = N - 1$, we will be able to state all λ_i in terms of λ_1 for instance

CQN: Gordon–Newell Theorem

- Steady-state distribution for CQN:
  - For ρ_i, the utilisation at node i:

$$\pi(r_1, r_2, \ldots, r_N) = \frac{1}{G} \prod_{i=1}^{N} \beta_i(r_i) \rho_i^{r_i}$$

where:

$$\beta_i(r_i) = \begin{cases} 1 & : \text{if node } i \text{ is single server} \\ \frac{1}{r_i!} & : \text{if node } i \text{ is IDN} \end{cases}$$

$$G = \sum_{\{r_i\}\,:\, r_1 + r_2 + \cdots + r_N = K} \prod_{i=1}^{N} \beta_i(r_i) \rho_i^{r_i}$$

CQN: Simplified Gordon–Newell

- For closed queueing networks with no independent delay nodes, we can simplify the full Gordon–Newell result considerably
- Steady-state result:

$$\pi(r_1, r_2, \ldots, r_N) = \frac{1}{G} \prod_{i=1}^{N} \rho_i^{r_i}$$

where:

$$G = \sum_{\{r_i\}\,:\, r_1 + r_2 + \cdots + r_N = K} \prod_{i=1}^{N} \rho_i^{r_i}$$

CQN: Normalisation Constant

- Hard issue behind Gordon–Newell is finding the normalisation constant G
- To find G you have to enumerate the state space – as with other concurrent systems, there is a state space explosion as number of queues/customers grows
- Recall that for N queues, K customers, we have:

$$\binom{K + N - 1}{N - 1} \text{ states}$$

Recall Jackson's theorem

- For a steady-state probability $\pi(r_1, \ldots, r_N)$ of there being r_1 jobs in node 1, r_2 nodes at node 2, etc.:

$$\begin{aligned} \pi(r_1, r_2, \ldots, r_N) &= \prod_{i=1}^{N} (1 - \rho_i) \rho_i^{r_i} \\ &= \prod_{i=1}^{N} \pi_i(r_i) \end{aligned}$$

where $\pi_i(r_i)$ is the steady-state probability there being n_i jobs at node i independently

PEPA and Product Form

- A product form result links the overall steady-state of a system to the product of the steady state for the components of that system
  - e.g. Jackson's theorem
- In PEPA, a simple product form can be got from:

$$P_1 \underset{\emptyset}{\bowtie} P_2 \underset{\emptyset}{\bowtie} \cdots \underset{\emptyset}{\bowtie} P_n$$

- $\pi(P_1^{r_1}, P_2^{r_2}, \ldots, P_n^{r_n}) = \frac{1}{G} \prod_{i=1}^{n} \pi(P_1^{r_1}) \cdots \pi(P_n^{r_n})$
- where $\pi(P_i^{r_i})$ is steady state prob. that component P_i is in state r_i

PEPA and RCAT

- RCAT: *Reversed Compound Agent Theorem*
- RCAT can take the more general cooperation:

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

- ...and find a product form, given structural conditions, in terms of the individual components P and Q

What does RCAT do?

- RCAT expresses the reversed component $\overline{P \underset{L}{\bowtie} Q}$ in terms of $\overline{P}$ and $\overline{Q}$ (almost)
- This is powerful since it avoids the need to expand the state space of $P \underset{L}{\bowtie} Q$
- This is useful since from the forward and reversed processes, $P \underset{L}{\bowtie} Q$ and $\overline{P \underset{L}{\bowtie} Q}$, we can find the steady state distribution $\pi(P_i, Q_i)$
- $\pi(P_i, Q_i)$ is the steady state distribution of both the forward and reversed processes (by definition)

Recall: Reversed processes

The *reversed process* of a stochastic process is a dual process:

- with the same state space
- in which the direction of time is reversed (like seeing a film backwards)
- if the reversed process is stochastically identical to the original process, that process is called *reversible*

Recall: Reversed processes

- The reversed process of a stationary Markov process $\{X_t : t \geq 0\}$ with state space S, generator matrix Q and stationary probabilities $\vec{\pi}$ is a stationary Markov process with generator matrix Q' defined by:

$$q'_{ij} = \frac{\pi_j q_{ji}}{\pi_i} \qquad : i, j \in S$$

  and with the same stationary probabilities $\vec{\pi}$.

Reversible processes

- If $\{X(t_1), \dots X(t_n)\}$ has the same distribution as $\{X(\tau - t_1), \dots X(\tau - t_n)\}$ for all τ, $t_1, \dots\ t_n$ then the process is called *reversible*
- Reversible processes are stationary i.e. stationary means that the joint distribution is independent of shifts of time
- Reversible processes satisfy the *detailed balance equations*

$$\pi_i q_{ij} = \pi_j q_{ji}$$

where π is the steady state probability and q_{ij} are the transition from i to j

Kolmogorov's Generalised Criteria

A stationary Markov process with state space S and generator matrix Q has reversed process with generator matrix Q' if and only if:

1. $q'_i = q_i$ for every state $i \in S$
2. For every finite sequence of states $i_1, i_2, ..., i_n \in S$,

$$q_{i_1 i_2} q_{i_2 i_3} \dots q_{i_{n-1} i_n} q_{i_n i_1} = q'_{i_1 i_n} q'_{i_n i_{n-1}} \dots q'_{i_3 i_2} q'_{i_2 i_1}$$

where $q_i = -q_{ii} = \sum_{j\,:\,j \neq i} q_{ij}$

Finding π from the reversed process

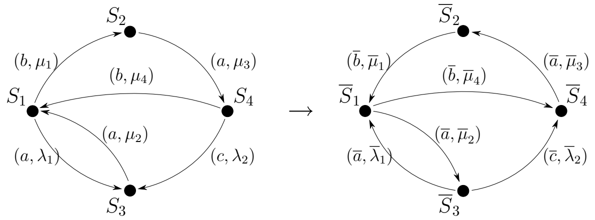
- Once reversed process rates Q' have been found, can be used to extract $\vec{\pi}$
- In an irreducible Markov process, choose a reference state 0 arbitrarily
- Find a sequence of connected states, in either the forward or reversed process, $0, \dots, j$ (i.e. with either $q_{i,i+1} > 0$ or $q'_{i,i+1} > 0$ for $0 \leq i \leq j-1$) for any state j and calculate:

$$\pi_j = \pi_0 \prod_{i=0}^{j-1} \frac{q_{i,i+1}}{q'_{i+1,i}} = \pi_0 \prod_{i=0}^{j-1} \frac{q'_{i,i+1}}{q_{i+1,i}}$$

Reversing a sequential component

- Reversing a sequential component, S, is straightforward:

$$\overline{S} \stackrel{\text{def}}{=} \sum_{i : R_i \xrightarrow{(a_i,\lambda_i)} S} (\overline{a}_i, \overline{\lambda}_i).\overline{R}_i$$



Activity substitution

- We need to be able to substitute a PEPA activity $\alpha = (a, r)$ for another $\alpha' = (a', r')$:

$$(\beta.P)\{\alpha \leftarrow \alpha'\} = \begin{cases} \alpha'.(P\{\alpha \leftarrow \alpha'\}) & \text{: if } \alpha = \beta \\ \beta.(P\{\alpha \leftarrow \alpha'\}) & \text{: otherwise} \end{cases}$$

$$(P + Q)\{\alpha \leftarrow \alpha'\} = P\{\alpha \leftarrow \alpha'\} + Q\{\alpha \leftarrow \alpha'\}$$

$$(P \underset{L}{\bowtie} Q)\{\alpha \leftarrow \alpha'\} = P\{\alpha \leftarrow \alpha'\} \underset{L\{\alpha \leftarrow \alpha'\}}{\bowtie} Q\{\alpha \leftarrow \alpha'\}$$

where $L\{(a, \lambda) \leftarrow (a', \lambda')\} = (L \setminus \{a\}) \cup \{a'\}$ if $a \in L$ and L otherwise

- A set of substitutions can be applied with:

$$P\{\alpha \leftarrow \alpha', \beta \leftarrow \beta'\}$$

RCAT Conditions (Informal)

For a cooperation $P \underset{L}{\bowtie} Q$, the reversed process $\overline{P \underset{L}{\bowtie} Q}$ can be created if:

1. Every passive action in P or Q that is involved in the cooperation $\underset{L}{\bowtie}$ must always be enabled in P or Q respectively.
2. Every reversed action $\overline{a}$ in $\overline{P}$ or $\overline{Q}$, where a is active in the original cooperation $\underset{L}{\bowtie}$, must:
   (a) always be enabled in $\overline{P}$ or $\overline{Q}$ respectively
   (b) have the same rate throughout $\overline{P}$ or $\overline{Q}$ respectively

RCAT Notation

In the cooperation, $P \underset{L}{\bowtie} Q$:

- $\mathcal{A}_P(L)$ is the set of actions in L that are also active in the component P
- $\mathcal{A}_Q(L)$ is the set of actions in L that are also active in the component Q
- $\mathcal{P}_P(L)$ is the set of actions in L that are also passive in the component P
- $\mathcal{P}_Q(L)$ is the set of actions in L that are also passive in the component Q
- $\overline{L}$ is the reversed set of actions in L, that is $\overline{L} = \{\overline{a} \mid a \in L\}$

RCAT Conditions (Formal)

For a cooperation $P \underset{L}{\bowtie} Q$, the reversed process $\overline{P \underset{L}{\bowtie} Q}$ can be created if:

1. Every passive action type in $\mathcal{P}_P(L)$ or $\mathcal{P}_Q(L)$ is always enabled in P or Q respectively (i.e. enabled in all states of the transition graph)
2. Every reversed action of an active action type in $\mathcal{A}_P(L)$ or $\mathcal{A}_Q(L)$ is always enabled in $\overline{P}$ or $\overline{Q}$ respectively
3. Every occurrence of a reversed action of an active action type in $\mathcal{A}_P(L)$ or $\mathcal{A}_Q(L)$ has the same rate in $\overline{P}$ or $\overline{Q}$ respectively

RCAT (I)

For $P \underset{L}{\bowtie} Q$, the reversed process is:

$$\overline{P \underset{L}{\bowtie} Q} = R^* \underset{\overline{L}}{\bowtie} S^*$$

where:

$$\begin{aligned} R^* &= \overline{R}\{(\overline{a}, \overline{p}_a) \leftarrow (\overline{a}, \top) \mid a \in \mathcal{A}_P(L)\} \\ S^* &= \overline{S}\{(\overline{a}, \overline{q}_a) \leftarrow (\overline{a}, \top) \mid a \in \mathcal{A}_Q(L)\} \\ R &= P\{(a, \top) \leftarrow (a, x_a) \mid a \in \mathcal{P}_P(L)\} \\ S &= Q\{(a, \top) \leftarrow (a, x_a) \mid a \in \mathcal{P}_Q(L)\} \end{aligned}$$

where the reversed rates, $\overline{p}_a$ and $\overline{q}_a$, of reversed actions are solutions of Kolmogorov equations.

RCAT (II)

x_a are solutions to the linear equations:

$$x_a = \begin{cases} \overline{q}_a & : \text{if } a \in \mathcal{P}_P(L) \\ \overline{p}_a & : \text{if } a \in \mathcal{P}_Q(L) \end{cases}$$

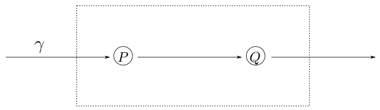
and $\overline{p}_a$, $\overline{q}_a$ are the symbolic rates of action types $\overline{a}$ in $\overline{P}$ and $\overline{Q}$ respectively.

RCAT in words

To obtain $\overline{P \underset{L}{\bowtie} Q} = R^* \underset{\overline{L}}{\bowtie} S^*$:

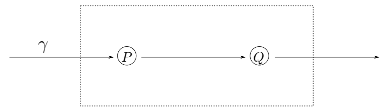
1. substitute all the cooperating passive rates in P, Q with symbolic rates, x_{action}, to get R, S
2. reverse R and S, to get $\overline{R}$ and $\overline{S}$
3. solve non-linear equations to get reversed rates, $\{\overline{r}\}$ in terms of forward rates $\{r\}$
4. solve non-linear equations to get symbolic rates $\{x_{action}\}$ in terms of forward rates
5. substitute all the cooperating active rates in $\overline{R}$, $\overline{S}$ with $\top$ to get R^*, S^*

Example: Tandem queues (I)



- Jobs arrive to node P with activity (e, γ)
- Jobs are serviced at node P with rate μ_1
- Jobs move between node P and Q with action a
- Jobs are serviced at node Q with rate μ_2
- Jobs depart Q with action d

Example: Tandem queues (II)



- PEPA description, $P_0 \underset{\{a\}}{\bowtie} Q_0$, where:

$$
\begin{aligned}
P_0 &\stackrel{\text{def}}{=} (e,\gamma).P_1 \\
P_n &\stackrel{\text{def}}{=} (e,\gamma).P_{n+1} + (a,\mu_1).P_{n-1} && : n > 0 \\
Q_0 &\stackrel{\text{def}}{=} (a,\top).Q_1 \\
Q_n &\stackrel{\text{def}}{=} (a,\top).Q_{n+1} + (d,\mu_2).Q_{n-1} && : n > 0
\end{aligned}
$$

Example: Tandem queues (III)

- Replace passive rates in cooperation with variables:

$$
\begin{aligned}
R &= P\{(a,\top) \leftarrow (a,x_a) \mid a \in \mathcal{P}_P(L)\} \\
S &= Q\{(a,\top) \leftarrow (a,x_a) \mid a \in \mathcal{P}_Q(L)\}
\end{aligned}
$$

- Transformed PEPA model:

$$
\begin{aligned}
R_0 &\stackrel{\text{def}}{=} (e,\gamma).R_1 \\
R_n &\stackrel{\text{def}}{=} (e,\gamma).R_{n+1} + (a,\mu_1).R_{n-1} && : n > 0 \\
S_0 &\stackrel{\text{def}}{=} (a,x_a).S_1 \\
S_n &\stackrel{\text{def}}{=} (a,x_a).S_{n+1} + (d,\mu_2).S_{n-1} && : n > 0
\end{aligned}
$$

Example: Tandem queues (IV)

- Reverse components R and S to get:

$$
\begin{aligned}
\overline{R}_0 &\stackrel{\text{def}}{=} (\overline{a},\overline{\mu}_1).\overline{R}_1 \\
\overline{R}_n &\stackrel{\text{def}}{=} (\overline{a},\overline{\mu}_1).\overline{R}_{n+1} + (\overline{e},\overline{\gamma}).\overline{R}_{n-1} && : n > 0 \\
\overline{S}_0 &\stackrel{\text{def}}{=} (\overline{d},\overline{\mu}_2).\overline{S}_1 \\
\overline{S}_n &\stackrel{\text{def}}{=} (\overline{d},\overline{\mu}_2).\overline{S}_{n+1} + (\overline{a},\overline{x}_a).\overline{S}_{n-1} && : n > 0
\end{aligned}
$$

- Now need to find in this order:
  1. reverse rates in terms of forward rates
  2. variable x_a in terms of forward rates

Example: Tandem queues (V.1)

- To find reverse rates – easiest route is to use *reversibility* of $M/M/1$ queue. In an $M/M/1$ queue:
  - forward arrival rate = reverse service rate
  - forward service rate = reverse arrival rate
  - Thus: $\overline{\mu}_1 = \gamma$, $\overline{\mu}_2 = x_a$,$\overline{\gamma} = \mu_1$ and $\overline{x}_a = \mu_2$
- Sometimes Kolmogorov Criteria will be needed to generate extra equations (see over for alternative method involving exit rate and Kolmogorov)

Example: Tandem queues (V.2)

- Finding reverse rates using Kolmogorov
  - Compare forward/reverse leaving rate from states R_0, S_0:

$$exit\_rate(R_0) = exit\_rate(\overline{R}_0) : \quad \overline{\mu}_1 = \gamma$$
$$exit\_rate(S_0) = exit\_rate(\overline{S}_0) : \quad \overline{\mu}_2 = x_a$$

  - Compare rate cycles in R, $\overline{R}$ and S, $\overline{S}$:

$$R_0 \to R_1 \to R_0 : \quad \gamma\mu_1 = \overline{\mu}_1\overline{\gamma}$$
$$S_0 \to S_1 \to S_0 : \quad x_a\mu_2 = \overline{\mu}_2\overline{x}_a$$

  - Giving: $\overline{\gamma} = \mu_1$ and $\overline{x}_a = \mu_2$

Example: Tandem queues (VI)

- Finding symbolic rates – recall:

$$x_a = \begin{cases} \overline{q}_a & : \text{if } a \in \mathcal{P}_P(L) \\ \overline{p}_a & : \text{if } a \in \mathcal{P}_Q(L) \end{cases}$$

- In this case, $a \in \mathcal{P}_Q(L)$, so $x_a = \overline{p}_a =$ reversed rate of a-action in $\overline{R}$
- Thus $x_a = \overline{\mu}_1 = \gamma$
- This agrees with rate of customers leaving forward network – why?

Example: Tandem queues (VII)

- Constructing $\overline{P \underset{L}{\bowtie} Q}$
  - $\overline{P_0 \underset{\{a\}}{\bowtie} Q_0} = R_0^* \underset{\{\overline{a}\}}{\bowtie} S_0^*$ where:

$$R_0^* \stackrel{\text{def}}{=} (\overline{a}, \top).R_1^*$$
$$R_n^* \stackrel{\text{def}}{=} (\overline{a}, \top).R_{n+1}^* + (\overline{e}, \mu_1).R_{n-1}^* \quad : n > 0$$
$$S_0^* \stackrel{\text{def}}{=} (\overline{d}, \gamma).S_1^*$$
$$S_n^* \stackrel{\text{def}}{=} (\overline{d}, \gamma).S_{n+1}^* + (\overline{a}, \mu_2).S_{n-1}^* \quad : n > 0$$

Example: Tandem queues (VIII)

- Finding the steady state distribution:
  - Need to use the following formula:

$$\pi_j = \pi_0 \prod_{i=0}^{j-1} \frac{q_{i,i+1}}{q'_{i+1,i}}$$

  ...to find the steady state distribution

  - First need to construct a sequence of events to a generic state (n, m) in network
    - where (n, m) represents n jobs in node P and m in node Q

Example: Tandem queues (IX)

- Generic state can be reached by:
  1. $n + m$ arrivals or e-actions to node P (forward rate $= \gamma$, reverse rate $= \mu_1$)
  2. followed by m departures or a-actions from node P and arrivals to node Q (forward rate $= \mu_1$, reverse rate $= \mu_2$)

Thus:
$$\begin{aligned} \pi(n, m) &= \pi_0 \prod_{i=0}^{n+m-1} \frac{\gamma}{\mu_1} \times \prod_{i=0}^{m-1} \frac{\mu_1}{\mu_2} \\ &= \pi_0 \left(\frac{\gamma}{\mu_1}\right)^n \left(\frac{\gamma}{\mu_2}\right)^m \end{aligned}$$

References

- RCAT
  - *Turning back time in Markovian Process Algebra.* Peter Harrison. TCS 290(3), pp. 1947–1986. January 2003.
- Generalised RCAT: less strict structural conditions
  - *Reversed processes, product forms and a non-product form.* Peter Harrison. LAA 386, pp. 359–381. July 2004.
- MARCAT: N-way cooperation extension:
  - *Separable equilibrium state probabilities via time-reversal in Markovian process algebra.* Peter Harrison and Ting-Ting Lee. TCS, pp. 161–182. November 2005.