Synchronisation in PEPA models

Jeremy Bradley  Stephen Gilmore  Nigel Thomas

Email: jb@doc.ic.ac.uk  nigel.thomas@ncl.ac.uk
stephen.gilmore@ed.ac.uk

Department of Computing,
Imperial College London

LFCS,
University of Edinburgh

Department of
Computer Science,
University of Newcastle

Produced with prosper and \LaTeX
In stochastic models...

Synchronisation can significantly affect performance results
Presentation

- PEPA and analysis
- The PEPA process algebra
- Synchronisation in practice
- Results
Types of Analysis

Steady-state and transient analysis in PEPA:

\[ A1 \text{ def } = (\text{start, } r_1).A2 + (\text{pause, } r_2).A3 \]
\[ A2 \text{ def } = (\text{run, } r_3).A1 + (\text{fail, } r_4).A3 \]
\[ A3 \text{ def } = (\text{recover, } r_1).A1 \]
\[ AA \text{ def } = (\text{run, } \top).(\text{alert, } r_5).AA \]
\[ \text{Sys def } = AA \{ \text{run} \} A1 \]
Types of Analysis

Steady-state and transient analysis in PEPA:

\[
\begin{align*}
A1 & \overset{\text{def}}{=} (\text{start}, r_1).A2 + (\text{pause}, r_2).A3 \\
A2 & \overset{\text{def}}{=} (\text{run}, r_3).A1 + (\text{fail}, r_4).A3 \\
A3 & \overset{\text{def}}{=} (\text{recover}, r_1).A1 \\
AA & \overset{\text{def}}{=} (\text{run}, T).(\text{alert}, r_5).AA \\
\text{Sys} & \overset{\text{def}}{=} \text{AA} \{ \text{run} \} A1
\end{align*}
\]
Types of Analysis

Steady-state and transient analysis in PEPA:

\[
\begin{align*}
A1 & \text{ def } = (\text{start, } r_1).A2 + (\text{pause, } r_2).A3 \\
A2 & \text{ def } = (\text{run, } r_3).A1 + (\text{fail, } r_4).A3 \\
A3 & \text{ def } = (\text{recover, } r_1).A1 \\
AA & \text{ def } = (\text{run, } \top).(\text{alert, } r_5).AA \\
\text{Sys} & \text{ def } = AA \{\text{run}\} A1
\end{align*}
\]
Types of Analysis

Steady-state and transient analysis in PEPA:

A1 $\overset{\text{def}}{=} (\text{start}, r_1).A2 + (\text{pause}, r_2).A3$

A2 $\overset{\text{def}}{=} (\text{run}, r_3).A1 + (\text{fail}, r_4).A3$

A3 $\overset{\text{def}}{=} (\text{recover}, r_1).A1$

AA $\overset{\text{def}}{=} (\text{run}, \top).(\text{alert}, r_5).AA$

Sys $\overset{\text{def}}{=} \begin{array}{c} \text{AA} \{ \text{run} \} \end{array} A1$

$\Rightarrow$

![Graph showing PEPA model output]
Types of Analysis

Steady-state and transient analysis in PEPA:

\[ A_1 \overset{\text{def}}{=} \text{(start, } r_1\text{)}.A_2 + \text{(pause, } r_2\text{)}.A_3 \]
\[ A_2 \overset{\text{def}}{=} \text{(run, } r_3\text{)}.A_1 + \text{(fail, } r_4\text{)}.A_3 \]
\[ A_3 \overset{\text{def}}{=} \text{(recover, } r_1\text{)}.A_1 \]
\[ AA \overset{\text{def}}{=} \text{(run, } \top\text{).(alert, } r_5\text{)}.AA \]
\[ \text{Sys} \overset{\text{def}}{=} AA \{\text{run}\} A_1 \]
Extract a passage-time density from a PEPA model:

\[
\begin{align*}
A1 & \overset{\text{def}}{=} (\text{start}, r_1) . A2 + (\text{pause}, r_2) . A3 \\
A2 & \overset{\text{def}}{=} (\text{run}, r_3) . A1 + (\text{fail}, r_4) . A3 \\
A3 & \overset{\text{def}}{=} (\text{recover}, r_1) . A1 \\
AA & \overset{\text{def}}{=} (\text{run}, \top) . (\text{alert}, r_5) . AA \\
\text{Sys} & \overset{\text{def}}{=} AA \{ \text{run} \} A1
\end{align*}
\]
Stochastic Process Algebra

PEPA syntax:

\[
P ::= (a, \lambda).P \mid P + P \mid P \otimes L P \mid P/L \mid A
\]
Stochastic Process Algebra

PEPA syntax:

\[
P ::= (a, \lambda).P \mid P + P \mid P \parallel P \mid P/L \mid A
\]

❖ Action prefix: \((a, \lambda).P\)
Stochastic Process Algebra

PEPA syntax:

\[
P ::= (a, \lambda).P \mid P + P \mid P \boxplus P \mid P/L \mid A
\]

- Action prefix: \((a, \lambda).P\)
- Competitive choice: \(P_1 + P_2\)
PEPA syntax:

\[ P ::= (a, \lambda).P \mid P + P \mid P \otimes_{L} P \mid P/L \mid A \]

- **Action prefix:** \((a, \lambda).P\)
- **Competitive choice:** \(P_1 + P_2\)
- **Cooperation:** \(P_1 \otimes_{L} P_2\)
PEPA syntax:

\[ P ::= (a, \lambda).P \mid P + P \mid P \triangleright L P \mid P/L \mid A \]

- Action prefix: \((a, \lambda).P\)
- Competitive choice: \(P_1 + P_2\)
- Cooperation: \(P_1 \triangleright L P_2\)
- Action hiding: \(P/L\)
Stochastic Process Algebra

PEPA syntax:

\[
P ::= (a, \lambda).P \mid P + P \mid P \circ_L P \mid P/L \mid A
\]

- Action prefix: \((a, \lambda).P\)
- Competitive choice: \(P_1 + P_2\)
- Cooperation: \(P_1 \circ_L P_2\)
- Action hiding: \(P/L\)
- Constant label: \(A\)
PEPA model: passage time/transient analysis - $O(10^8)$ states

Semi-Markov PEPA: passage time/transient analysis - $O(10^7)$ states
PEPA: A Transmitter-Receiver

System \( \overset{\text{def}}{=} \) (Transmitter \( \square \) Receiver) \( \square \) Network

Transmitter \( \overset{\text{def}}{=} \) (transmit, \( \lambda_1 \)).(t – recover, \( \lambda_2 \)).Transmitter

Receiver \( \overset{\text{def}}{=} \) (receive, \( T \)).(r – recover, \( \mu \)).Receiver

Network \( \overset{\text{def}}{=} \) (transmit, \( T \)).(delay, \( \nu_1 \)).(receive, \( \nu_2 \)).Network

A simple transmitter-receiver over a network
Apparent Rate

- Apparent rate of a component $P$ is given by $r_a(P)$.
Apparent Rate

- Apparent rate of a component $P$ is given by $r_a(P)$
- Apparent rate describes the overall observed rate that $P$ performs an $a$-action
Apparent Rate

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

$$r_a(P) = \sum_{P \rightarrow (a, \lambda_i)} \lambda_i$$
Apparent Rate Examples

\[ r_a(P \xrightarrow{(a,\lambda)} ) = \lambda \]
Apparent Rate Examples

$r_a(P \rightarrow (a, \lambda)) = \lambda$

$r_a(P \rightarrow (a, T)) = T$
Apparent Rate Examples

\[ r_a(P \xrightarrow{(a, \lambda)} ) = \lambda \]

\[ r_a(P \xrightarrow{(a, \top)} ) = \top \]

\[ r_a \left( \left( \begin{array}{c} (a, \lambda_1) \\ (a, \lambda_2) \end{array} \right) \right) = \lambda_1 + \lambda_2 \]
Apparent Rate Examples

- \( r_a(P \xrightarrow{(a, \lambda)} ) = \lambda \)
- \( r_a(P \xrightarrow{(a, \top)} ) = \top \)
- \( r_a \left( P \xrightarrow{(a, \lambda_1)} \xrightarrow{(a, \lambda_2)} \right) = \lambda_1 + \lambda_2 \)
- \( r_a \left( P \xrightarrow{(a, \top)} \xrightarrow{(a, \top)} \right) = 2\top \)
In PEPA, when synchronising two model components, \( P \) and \( Q \) where both \( P \) and \( Q \) enable many \( a \)-actions:

\[
\begin{align*}
P & \quad \quad , \quad \quad , \\
\text{and} & \quad \quad , \quad \quad , \\
Q & \quad \quad , \quad \quad ,
\end{align*}
\]

The synchronised rate for \( P \) and \( Q \) is:

\[
R = \min (r_a(P), r_a(Q))
\]
In PEPA, when synchronising two model components, $P$ and $Q$ where both $P$ and $Q$ enable many $\alpha$-actions:

The synchronised rate for $P \circlearrowright_{\alpha} Q \rightarrow P' \circlearrowright_{\alpha} Q'$ is:

$$R = \frac{\lambda}{r_{\alpha}(P)} \frac{\mu}{r_{\alpha}(Q)} \min(r_{\alpha}(P), r_{\alpha}(Q))$$
Some tools such as: Möbius, PRISM, PWB use an approximate synchronisation model.
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:

\[
\begin{align*}
\text{P} & \xrightarrow{(a, \lambda)} \text{P}' & \text{Q} & \xrightarrow{(a, \mu)} \text{Q}' \\
\text{P} & \xrightarrow{(a, \cdot)} & \text{Q} & \xrightarrow{(a, \cdot)}
\end{align*}
\]
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:

  $P \xrightarrow{(a, \lambda)} P'$ and $Q \xrightarrow{(a, \mu)} Q'$

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

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

As an example:

- \( \text{Client} \stackrel{\text{def}}{=} (\text{data}, \lambda).\text{Client}' \)
- \( \text{Network} \stackrel{\text{def}}{=} (\text{data}, \top).\text{NetworkGo} + (\text{data}, \top).\text{NetworkStall} \)
Example

As an example:

- $\text{Client} \overset{\text{def}}{=} (\text{data}, \lambda).\text{Client}'$

- $\text{Network} \overset{\text{def}}{=} (\text{data}, \top).\text{NetworkGo} + (\text{data}, \top).\text{NetworkStall}$

The combination $\text{Client} \otimes_{\text{data}} \text{Network}$ should evolve with an overall $\text{data}$ rate parameter of $\lambda$. 
As an example:

\[ \text{Client} \overset{\text{def}}{=} (\text{data}, \lambda) \cdot \text{Client}' \]

\[ \text{Network} \overset{\text{def}}{=} (\text{data}, \top) \cdot \text{NetworkGo} + (\text{data}, \top) \cdot \text{NetworkStall} \]

The combination \( \text{Client} \otimes \text{Network} \) should evolve with an overall \( \text{data} \) rate parameter of \( \lambda \)

Under the tool approximation the overall synchronised rate becomes \( 2\lambda \)
\[\begin{align*}
A & \overset{\text{def}}{=} (\text{run, } \lambda_1).(\text{stop, } \lambda_2).A \\
B & \overset{\text{def}}{=} (\text{run, } \top).(\text{pause, } \lambda_3).B \\
\text{Sys}_A & \overset{\text{def}}{=} A \begin{array}{c} \text{run} \end{array} (B \parallel B)
\end{align*}\]

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 real-rate actions (in \((B \parallel B)\)) are synchronised against a single real-rate action (in \(A\))
Results: Multiple Active

- Percentage difference in CDF functions over passage time between consecutive stop actions (for decreasing $\mu$)
Isn’t this really unusual?

Q: How common is this kind of modelling problem? Isn’t this bizarre non-determinism to see in a component?

A: Having an explicit individual component with either:

\[ P \overset{\text{def}}{=} (a, \lambda).P' + (a, \mu).P'' \]  
(multiple active)

\[ Q \overset{\text{def}}{=} (a, \top).Q' + (a, \top).Q'' \]  
(multiple passive)

...might be unusual, but simple multi-agent synchronisation of \[ S \boxdot \{R \parallel R \parallel \cdots \parallel R\} \] for some \( S \)

where \( R \overset{\text{def}}{=} (a, \top).(b, \mu).R' \) causes just this problem

This is a very common client–server architecture
Conclusion

- Synchronisation style makes a big difference to performance results!

- To summarise, using the tool approximation:
  - with multiple passive actions – sees an overestimation of passage-time results
  - with multiple active actions – sees an underestimation of passage-time results – why?