FORMAL METHODS FOR SYSTEM VERIFICATION

Examples: how the language may be used to describe systems

Sabina Rossi

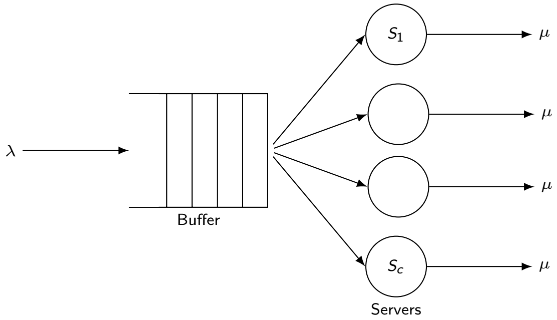
DAIS
Università Ca' Foscari
Venezia

Example 1: Multiple server queue as a single component

Description

We consider a queue with

- c servers
- a buffer with capacity N, where $N > c$
- customers arrive at rate λ
- the service rate of each server is μ



Example 1: Multiple server queue as a single component

Description

- When the queue is not full, then the queue will engage in an *accept* activity at rate λ, representing the acceptance of a customer into the queue.
- When the queue is full, since the arrival process will not be suspended, the queue will be involved in a *loss* activity, losing a customer at rate λ.
- When there are i costumers in the queue, then the queue will engage in a *serve* activity at rate $i\mu$, if $i < c$, and at rate $c\mu$ when $c \leq i \leq N$.

Example 1: Multiple server queue as a single component

PEPA model

- Let Q_i denote the component representing the behaviour of the queue when there are i costumers present (including those in service).

$$\begin{array}{lcll}
Q_0 & \stackrel{\text{def}}{=} & (accept, \lambda).Q_1 & \\
\vdots & & & \\
Q_i & \stackrel{\text{def}}{=} & (accept, \lambda).Q_{i+1} + (serve, i\mu).Q_{i-1} & 1 \leq i < c \\
\vdots & & & \\
Q_j & \stackrel{\text{def}}{=} & (accept, \lambda).Q_{j+1} + (serve, c\mu).Q_{j-1} & c \leq j < N \\
\vdots & & & \\
Q_N & \stackrel{\text{def}}{=} & (loss, \lambda).Q_N + (serve, c\mu).Q_{N-1} &
\end{array}$$

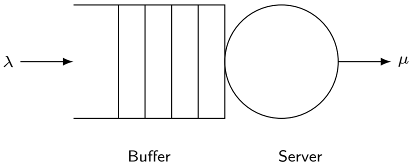
Example 2: Single server queue as two cooperating components

Description

We consider a single server queue with

- buffer capacity N
- customer population N
- customers arrive at rate λ
- the service rate is μ

The arrival process will be suspended when the queue is full as all the costomers will already be present in the queue.



Example 2: Single server queue as two cooperating components

PEPA model of the server

- We represent the queue as two interacting components: *Server* and *Buffer*.
- The behaviour of the *Server* is very simple: whenever it is able, the server will engage in a *serve* activity at rate μ.

$$\mathit{Server} \stackrel{\text{def}}{=} (\mathit{serve}, \mu).\mathit{Server}$$

Example 2: Single server queue as two cooperating components

PEPA model of the buffer

- When the buffer is not full customers will arrive at rate λ, so the *Buffer* will engage in an *accept* activity at rate λ.
- When the buffer is non-empty a customer will be available for service at a rate determined by the server, so the *Buffer* will engage in a *serve* activity at an unspecified rate.
- $Buffer_i$ will denote the behaviour of the buffer when there are i customers in the buffer.

Example 2: Single server queue as two cooperating components

PEPA model of the buffer

$$
\begin{aligned}
Buffer_0 &\stackrel{\text{def}}{=} (accept, \lambda).Buffer_1 \\
&\vdots \\
Buffer_i &\stackrel{\text{def}}{=} (accept, \lambda).Buffer_{i+1} + (serve, \top).Buffer_{i-1} \quad 1 \leq i < N \\
&\vdots \\
Buffer_N &\stackrel{\text{def}}{=} (serve, \top).Buffer_{N-1}
\end{aligned}
$$

Example 2: Single server queue as two cooperating components

PEPA model of the queue

- The *Queue* is formed by the cooperation of the *Buffer* and the *Server* for the *serve* activity.

$$\mathit{Queue}_0 \overset{\text{def}}{=} \mathit{Buffer}_0 \underset{\{\mathit{serve}\}}{\bowtie} \mathit{Server}$$

Example 3: Simple resource usage as two cooperating components

PEPA model

- We consider a simple system in which a process repeatedly carries out a *task*.
- In order to complete its task the process needs access to a resource for part, but not all, of the time.
- Thus the task can be regarded as being in two stages:
    - the first requiring access to the resource
    - the second involving only the process
- The resource is continuously available except for a short period after it has been used during which it is reset and therefore unavailable.

Example 3: Simple resource usage as two cooperating components

PEPA model

- The system has two components: *Process* and *Resource*.
- The *Process* will undertake two activities consecutively: *use* with some rate r_1 in cooperation with the resource, and *task* at some rate r_2.
- The *Resource* will engage in two activities consecutively: *use* at some rate r_3 and *update* at a rate r_4.

$$\begin{aligned} Process &\overset{\text{def}}{=} (use, r_1).(task, r_2).Process \\ Resource &\overset{\text{def}}{=} (use, r_3).(update, r_4).Resource \\ System &\overset{\text{def}}{=} Process \underset{\{use\}}{\bowtie} Resource \end{aligned}$$

Example 4: Simple resource usage as one component

PEPA model

- We could model the system as a single component.
- However, this does not reflect what is happening in the system as clearly as the previous representation.

$$\begin{aligned} System' \;\overset{\text{def}}{=}\; & (use, r_{13}).((task, r_2).(update, r_4).System' \\ & \qquad +(update, r_4).(task, r_2).System') \end{aligned}$$

where $r_{13} = min(r_1, r_3)$.

Example 5: Simple resource usage as several cooperating components

PEPA model

- Representing the components of the system as separate components in the model means that we can easly extend the model to represent a system in which there are two processes, or more, independent of each other but competing for the use of the resource.

$$\mathit{System}'' \stackrel{\text{def}}{=} (\mathit{Process} \| \mathit{Process}) \underset{\{\mathit{use}\}}{\bowtie} \mathit{Resource}$$

Example 6: A PC LAN

The system

- Suppose we wish to determine the mean waiting time for data packets at a PC connected to a local area network, operating as a token ring.
- The transmission medium supports no more than one transmission at any given time. To resolve conflicts, a token is passed round the network from one node to another in round robin order.

Example 6: A PC LAN

Token ring communication

- A node has control of the medium, i.e., it can transmit, only whilst it holds the token.
- In a PC LAN every PC corresponds to a node on the network.
- Other nodes on the network might be peripheral devices such as printers or faxes but for the purposes of this study we make no distinction and assume that all nodes are PCs.

Example 6: A PC LAN

Token ring communication: assumption

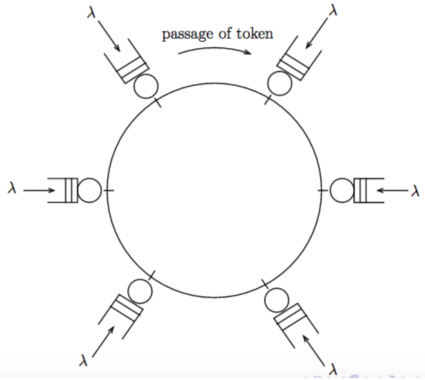
- We assume there are currently four PCs (or similar devices) connected to the LAN in a small office.

Example 6: A PC LAN

Modelling assumptions

- Each PC can only store one data packet waiting for transmission at a time, so at each visit of the token there is either one packet waiting or no packet waiting.
- The average rate at which each PC generates data packets for transmission is known to be λ.
- We also know the mean duration, $d = 1/\mu$, of a data packet transmission, and the mean time, $m = 1/\omega$, taken for the token to pass from one PC to the next.
- It is assumed that if another data packet is generated, whilst the PC is transmitting, this second data packet must wait for the next visit of the token before it can be transmitted. In other words, each PC can transmit at most one data packet per visit of the token.

Example 6: A PC LAN



Example 6: A PC LAN

Modelling the system: choosing the components

- The first stage in developing a model of the system in PEPA is to determine the components of the system and the actions which they can undertake.
- It seems clear that one type of component should be used to represent the PCs. The components representing the four PCs will have essentially the same behaviour. But since token visits the nodes in order we will need to distinguish the components.
- We will need another component to represent the medium. As remarked previously, the medium can be represented solely by the token.

Example 6: A PC LAN

Modelling the PCs

- The description of the PCs is very simple. Each PC only has two activities which it can undertake:
  - generate a data packet;
  - transmit a data packet.

  Moreover we are told that it can only hold one data packet at a time and so these activities must be undertaken sequentially.
- This suggests the following PEPA component for the *i*th PC.

$$\begin{aligned} PC_{i0} &\stackrel{\text{def}}{=} (arrive, \lambda).PC_{i1} \\ PC_{i1} &\stackrel{\text{def}}{=} (transmit, \mu).PC_{i0} \end{aligned}$$

This will need some refinement when we consider interaction with the token.

Example 6: A PC LAN

Modelling the token

- For the token we can think of its current state being characterised by its current position. Thus, if there are N PCs in the network the states of the token correspond to the values $\{1, 2, \ldots, N\}$.
- When it is at the ith PC then the token may
  - transmit a data packet if there is one to transmit and then walk on to the next PC
  - or walk on without any transmission if there is no data packet waiting.

$$\begin{aligned} Token_i \quad &\stackrel{\text{def}}{=} \quad (walkon_{i+1}, \omega).Token_{i+1} + \\ & \qquad (transmit_i, \mu).(walk_{i+1}, \omega).Token_{i+1} \end{aligned}$$

Example 6: A PC LAN

Refining the components

- In order to ensure that the token's choice is made dependent on the state of PC being visited, we add a walkon and a walk action to the PC and impose a cooperation between the PC and the Token for walk, walkon and transmit.

$$
\begin{aligned}
PC_{i0} &\stackrel{\text{def}}{=} (arrive, \lambda).PC_{i1} + (walkon_{i+1}, \omega).PC_{i0} + (walk_{i+1}, \omega).PC_{i0} \\
PC_{i1} &\stackrel{\text{def}}{=} (transmit_i, \mu).PC_{i0} + (walk_{i+1}, \omega).PC_{i1}
\end{aligned}
$$

Example 6: A PC LAN

Complete model: four PC case

$$
\begin{aligned}
PC_{10} &\overset{\text{def}}{=} (arrive, \lambda).PC_{11} + (walkon_2, \omega).PC_{10} + (walk_2, \omega).PC_{10} \\
PC_{11} &\overset{\text{def}}{=} (transmit_1, \mu).PC_{10} + (walk_2, \omega).PC_{11}
\end{aligned}
$$

$$
\begin{aligned}
PC_{20} &\overset{\text{def}}{=} (arrive, \lambda).PC_{21} + (walkon_3, \omega).PC_{20} + (walk_3, \omega).PC_{20} \\
PC_{21} &\overset{\text{def}}{=} (transmit_2, \mu).PC_{20} + (walk_3, \omega).PC_{21}
\end{aligned}
$$

$$
\begin{aligned}
PC_{30} &\overset{\text{def}}{=} (arrive, \lambda).PC_{31} + (walkon_4, \omega).PC_{30} + (walk_4, \omega).PC_{30} \\
PC_{31} &\overset{\text{def}}{=} (transmit_3, \mu).PC_{30} + (walk_4, \omega).PC_{31}
\end{aligned}
$$

$$
\begin{aligned}
PC_{40} &\overset{\text{def}}{=} (arrive, \lambda).PC_{41} + (walkon_1, \omega).PC_{40} + (walk_1, \omega).PC_{40} \\
PC_{41} &\overset{\text{def}}{=} (transmit_4, \mu).PC_{40} + (walk_1, \omega).PC_{41}
\end{aligned}
$$

Example 6: A PC LAN

Complete model: four PC case

$$\begin{aligned}
Token_1 &\stackrel{\text{def}}{=} (walkon_2, \omega).Token_2 + (transmit_1, \mu).(walk_2, \omega).Token_2\\
Token_2 &\stackrel{\text{def}}{=} (walkon_3, \omega).Token_3 + (transmit_2, \mu).(walk_3, \omega).Token_3\\
Token_3 &\stackrel{\text{def}}{=} (walkon_4, \omega).Token_4 + (transmit_3, \mu).(walk_4, \omega).Token_4\\
Token_4 &\stackrel{\text{def}}{=} (walkon_1, \omega).Token_1 + (transmit_4, \mu).(walk_1, \omega).Token_1
\end{aligned}$$

$$LAN \stackrel{\text{def}}{=} (PC_{10} \| PC_{20} \| PC_{30} \| PC_{40} \|) \underset{L}{\bowtie} Token_1$$

where
$L = \{walk_1, walk_2, walk_3, walk_4, walkon_1, walkon_2, walkon_3, walkon_4, transmit_1, transmit_2, transmit_3, transmit_4\}$.

Here we have arbitrarily chosen a starting state in which all the PCs are empty and the Token is at PC1.

Example 6: A PC LAN

Transition diagram for PC_i

$$\begin{tikzcd}
PC_{i0} \arrow[loop above, "(walk_2{,}\omega)"] \arrow[loop below, "(walkon_2{,}\omega)"'] \arrow[rr, bend left, "(arrive{,}\lambda)"] & & PC_{i1} \arrow[ll, bend left, "(transmit{,}\mu)"] \arrow[loop right, "(walk_2{,}\omega)"]
\end{tikzcd}$$

Example 6: A PC LAN

Transition diagram for the Token in a LAN with three PCs

$$\begin{tikzcd}
 & & Token'_1 \arrow[dr, bend left, "{(walk_2\,,\omega)}"] & & \\
 & Token_1 \arrow[ur, bend left, "{(transmit_1\,,\mu)}"] \arrow[rr, bend left, "{(walkon_2\,,\omega)}"] & & Token_2 \arrow[dr, bend left, "{(transmit_2\,,\mu)}"] \arrow[ddl, bend left, "{(walkon_3\,,\omega)}"'] & \\
Token'_3 \arrow[ur, bend left, "{(walk_1\,,\omega)}"] & & & & Token'_2 \arrow[dll, bend left, "{(walk_3\,,\omega)}"] \\
 & & Token_3 \arrow[ull, bend left, "{(transmit_3\,,\mu)}"] \arrow[uul, bend left, "{(walkon_1\,,\omega)}"'] & &
\end{tikzcd}$$