Modeling Concurrent and Probabilistic Systems

Exercises, Series 7

Joost-Pieter Katoen   Thomas Noll

Software Modeling and Verification Group
RWTH Aachen University
noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/mcps07/

Winter Semester 2007/08

Exercise 1a)

Modify the ABP and replace the failure situations $ack_\perp$ and $trans_\perp$ by a timeout handling to model lossy channels. The modified version of the ABP should behave as follows:

- If *Sender* sends a message, it starts a timer. If a *timeout* occurs before the acknowledgement is received, the message is retransmitted. Messages can get lost.
- If *Receiver* sends an acknowledgement, it starts a timer. If a *timeout* occurs before the next message is received, it retransmits the acknowledgement. Acknowledgements can get lost.

Give the modified process definition for the alternating bit protocol. Use the following timer process:

$$Timer = start.(\overline{timeout}.Timer + stop.Timer).$$

Compose your new *Sender* and *Receiver* each with a *Timer* process!

Solution 1a) – Sender

$$
\begin{aligned}
Sender &= \mathsf{new}\ \{timeout, start, stop\}\big(Sender_0 \parallel Timer\big) \\
Sender_b &= \sum_{d \in D} accept_d.Send_{db} \\
Send_{db} &= \overline{send}_{db}.\overline{start}.Wait_{db} \\
Wait_{db} &= ack_b.(\overline{stop}.Sender_{1-b} + timeout.Sender_{1-b}) \\
&\quad + ack_{1-b}.(\overline{stop}.Send_{db} + timeout.Send_{db}) \\
&\quad + timeout.Send_{db}
\end{aligned}
$$

$$
Timer = start.(\overline{timeout}.Timer + stop.Timer)
$$

Solution 1a) – Receiver

$$Receiver = \textsf{new}\ \{timeout, start, stop\}(Receiver'_0 \parallel Timer)$$

$$\begin{aligned} Receiver'_0 &= \sum_{d \in D} trans_{d0}.Reply_{d0} \\ &+ \sum_{d \in D} trans_{d1}.\overline{reply}_1.\overline{start}.Receiver_0 \\ Receiver_b &= \sum_{d \in D} trans_{db}.(timeout.Reply_{db} + \overline{stop}.Reply_{db}) \\ &+ \sum_{d \in D} trans_{d(1-b)}.(timeout.\overline{reply}_{1-b}.\overline{start}.Receiver_b + \\ &\qquad \overline{stop}.\overline{reply}_{1-b}.\overline{start}.Receiver_b) \\ &+ timeout.\overline{reply}_{1-b}.\overline{start}.Receiver_b \\ Reply_{db} &= \overline{deliver}_d.\overline{reply}_b.\overline{start}.Receiver_{1-b} \end{aligned}$$

$$Timer = start.(\overline{timeout}.Timer + stop.Timer)$$

Solution 1a) – System

$$Trans = \sum_{\substack{d \in D \\ b \in \{0,1\}}} send_{db}.\big(\underbrace{\overline{trans}_{db}.Trans}_{\text{msg success}} + \underbrace{Trans}_{\text{msg loss}}\big)$$

$$Ack = \sum_{b \in \{0,1\}} reply_b.\big(\underbrace{\overline{ack}_b.Ack}_{\text{ack success}} + \underbrace{Ack}_{\text{ack loss}}\big)$$

$$ABP(accept, deliver) = \mathsf{new}\ L\ \big(Sender \parallel Trans \parallel Ack \parallel Receiver\big)$$

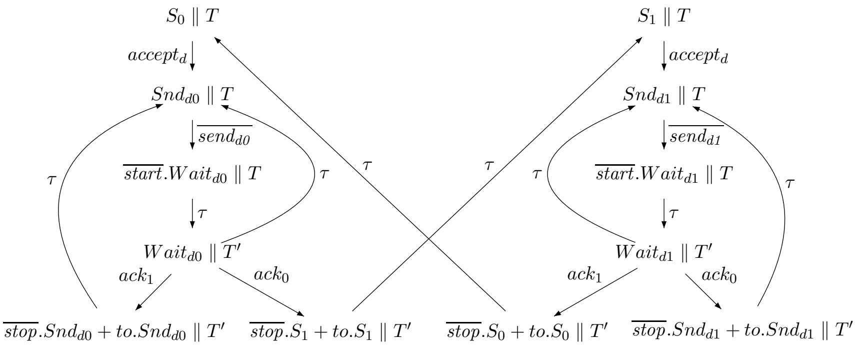
where $L = \{send_{db}, trans_{db}, reply_b, ack_b \mid d \in D, b \in \{0,1\}\}$.

Exercise 1b)

Minimize the LTS of the *Sender* and its *Timer* by computing its quotient under weak bisimulation! Use the partitioning algorithm from the lecture!

Solution 1b)

LTS of *Sender* $\parallel$ *Timer*:



Solution 1b)

Partitioning algorithm, first iteration:

P	$S_0 \parallel T$	$Snd_{d0} \parallel T$	$\overline{start}.W_{d0} \parallel T$	$W_{d0} \parallel T'$	$(\overline{stop}.S_1$ $+to.S_1) \parallel T'$	$(\overline{stop}.Snd_{d0}$ $+to.Snd_{d0}) \parallel T'$
$\overline{accept}^*$	$\{S\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{S\}$	$\emptyset$
$\overline{send_{d0}}^*$	$\emptyset$	$\{S\}$	$\{S\}$	$\{S\}$	$\emptyset$	$\{S\}$
$\overline{send_{d1}}^*$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
ack_0^*	$\emptyset$	$\emptyset$	$\{S\}$	$\{S\}$	$\emptyset$	$\emptyset$
ack_1^*	$\emptyset$	$\emptyset$	$\{S\}$	$\{S\}$	$\emptyset$	$\emptyset$
τ^*	$\{S\}$	$\{S\}$	$\{S\}$	$\{S\}$	$\{S\}$	$\{S\}$
Block	B_1	B_2	B_3	B_3	B_1	B_2

P	$S_1 \parallel T$	$Snd_{d1} \parallel T$	$start.W_{d1} \parallel T$	$W_{d1} \parallel T'$	$(\overline{stop}.S_0$ $+to.S_0) \parallel T'$	$(\overline{stop}.Snd_{d1}$ $+to.Snd_{d1}) \parallel T'$
$\overline{accept}^*$	$\{S\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{S\}$	$\emptyset$
$\overline{send_{d0}}^*$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\overline{send_{d1}}^*$	$\emptyset$	$\{S\}$	$\{S\}$	$\{S\}$	$\emptyset$	$\{S\}$
ack_0^*	$\emptyset$	$\emptyset$	$\{S\}$	$\{S\}$	$\emptyset$	$\emptyset$
ack_1^*	$\emptyset$	$\emptyset$	$\{S\}$	$\{S\}$	$\emptyset$	$\emptyset$
τ^*	$\{S\}$	$\{S\}$	$\{S\}$	$\{S\}$	$\{S\}$	$\{S\}$
Block	B_1	B_4	B_5	B_5	B_1	B_4

Second iteration:

P	$S_0 \parallel T$	$\overline{stop}.S_0 + to.S_0 \parallel$. $(\overline{to}.T + stop.T)$	.	$S_1 \parallel T$	$\overline{stop}.S_1 + to.S_1 \parallel$ $(\overline{to}.T + stop.T)$
$accept^*$	$\{B_2\}$	$\{B_2\}$	.	$\{B_4\}$	$\{B_4\}$
$\overline{send}^*_{d0}$	$\emptyset$	$\emptyset$	.	$\emptyset$	$\emptyset$
$\overline{send}^*_{d1}$	$\emptyset$	$\emptyset$	.	$\emptyset$	$\emptyset$
ack^*_0	$\emptyset$	$\emptyset$	.	$\emptyset$	$\emptyset$
ack^*_1	$\emptyset$	$\emptyset$	.	$\emptyset$	$\emptyset$
τ^*	$\{B_1\} \rightsquigarrow \{B_{1,1}\}$	$\{B_1\} \rightsquigarrow \{B_{1,1}\}$	.	$\{B_1\} \rightsquigarrow \{B_{1,2}\}$	$\{B_1\} \rightsquigarrow \{B_{1,2}\}$

P	$Snd_{d0} \parallel T$ $\parallel \overline{to}.T + stop.T$	$\overline{stop}.Snd_{d0} + to.Snd_{d0}$	$\overline{start}.wait_{d0} \parallel T$ $(\overline{to}.T + stop.T)$	$W_{d0} \parallel$
$accept^*$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\overline{send}^*_{d0}$	$\{B_3, B_2\}$	$\{B_3, B_2\}$	$\{B_2, B_3\}$	$\{B_2, B_3\}$
$\overline{send}^*_{d1}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
ack^*_0	$\emptyset$	$\emptyset$	$\{B_1\} \rightsquigarrow \{B_{1,2}\}$	$\{B_1\} \rightsquigarrow \{B_{1,2}\}$
ack^*_1	$\emptyset$	$\emptyset$	$\{B_2\}$	$\{B_2\}$
τ^*	$\{B_2\}$	$\{B_2\}$	$\{B_3, B_2\}$	$\{B_3, B_2\}$

Solution 1b)

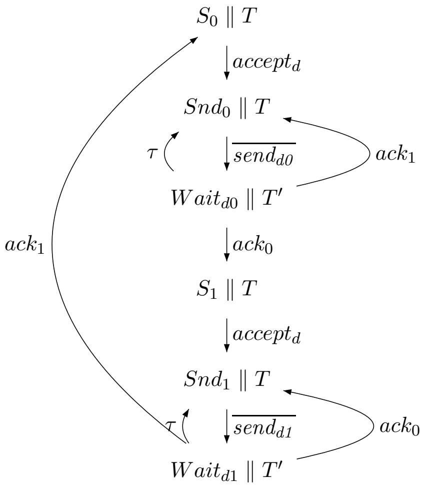
Second iteration (continued):

P	$Snd_{d1} \parallel T$	$\overline{stop}.Snd_{d1} + to.Snd_{d1} \parallel (\overline{to}.T + stop.T)$	$\overline{start}.wait_{d1} \parallel T$	$W_{d1} \parallel (\overline{to}.T + stop.T)$
$accept^*$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\overline{send}^*_{d0}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\overline{send}^*_{d1}$	$\{B_5, B_4\}$	$\{B_5, B_4\}$	$\{B_5, B_4\}$	$\{B_5, B_4\}$
ack^*_0	$\emptyset$	$\emptyset$	$\{B_4\}$	$\{B_4\}$
ack^*_1	$\emptyset$	$\emptyset$	$\{B_1\} \rightsquigarrow \{B_{1,1}\}$	$\{B_1\} \rightsquigarrow \{B_{1,1}\}$
τ^*	$\{B_4\}$	$\{B_4\}$	$\{B_5, B_4\}$	$\{B_5, B_4\}$

$\Rightarrow$ split B_1 into $B_{1,1}$ and $B_{1,2}$, all other blocks remain unchanged.
Result: Weak bisimulation quotient with six states.

Solution 1b)

Reduced LTS (w.r.t. weak bisimilarity):

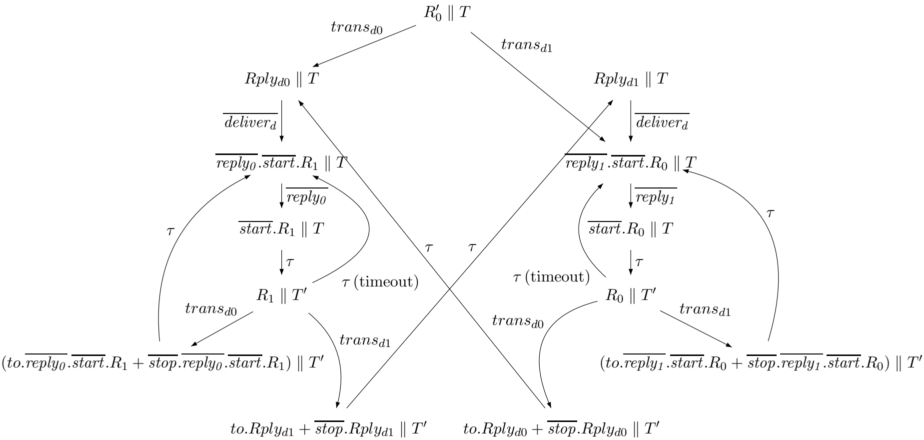


Exercise 1c)

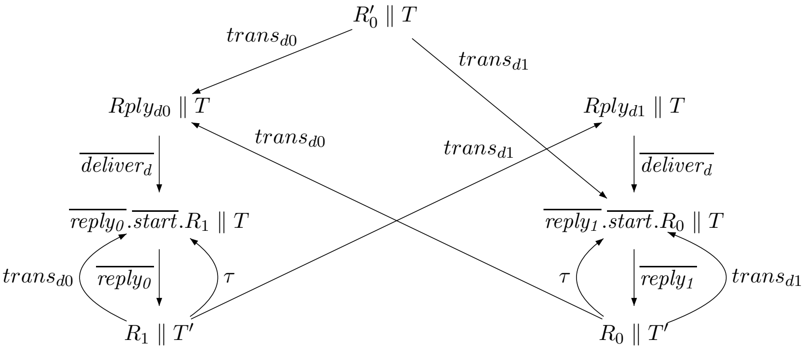
Do the same for the LTS of the *Receiver* and its *Timer*! You may do this directly, i.e. without applying the partitioning algorithm.

Solution 1c)

LTS of *Receiver* ∥ *Timer*:



Reduced LTS (w.r.t. weak bisimilarity):



Q: To prove the new protocol correct, one could replace the *Sender* $\|$ *Timer* and *Receiver* $\|$ *Timer* components by their quotients under weak bisimulation to obtain a smaller LTS. Why is this approach incorrect in general? Why can it still be applied in our setting?

A: Weak bisimilarity is not a congruence. In general, if $P_1 \approx P_2$ and $Q_1 \approx Q_2$, $P_1 + Q_1 \not\approx P_2 + Q_2$. However, weak bisimilarity is preserved under parallel composition and restriction. As these are the only operators we use to compose the *ABP* process, this is a valid approach.

Exercise 2

Show that the following simple communication protocol works correctly.
To this aim, prove that $\mathit{Protocol}(a, f)$ is observationally congruent to a one-place buffer:

$$
\begin{aligned}
\mathit{Protocol}(a, f) &= \mathsf{new}\ b, c, d, e\ \big(\mathit{Sender}(a, b, d, e) \parallel \mathit{Medium}(b, c, d) \parallel \mathit{Receiver}(c, e, f)\big) \\
\mathit{Sender}(a, b, d, e) &= a.\mathit{Sender}'(a, b, d, e) \\
\mathit{Sender}'(a, b, d, e) &= \overline{b}.\big(d.\mathit{Sender}'(a, b, d, e) + e.\mathit{Sender}(a, b, d, e)\big) \\
\mathit{Medium}(b, c, d) &= b.\big(\overline{c}.\mathit{Medium}(b, c, d) + \overline{d}.\mathit{Medium}(b, c, d)\big) \\
\mathit{Receiver}(c, e, f) &= c.\overline{f}.\overline{e}.\mathit{Receiver}(c, e, f)
\end{aligned}
$$

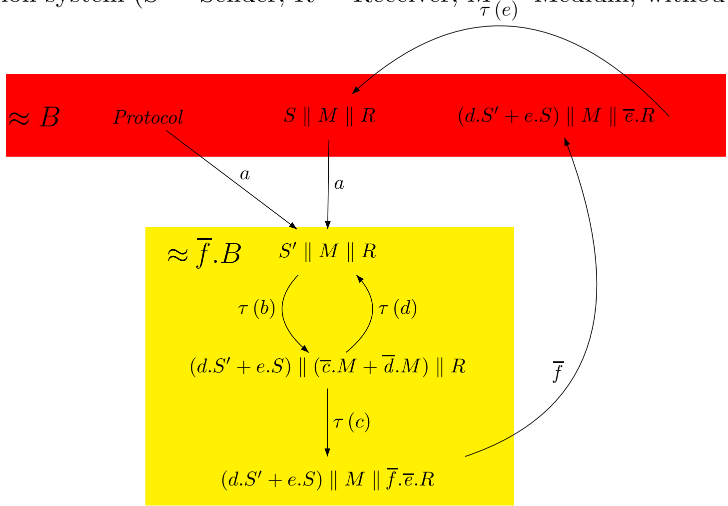
Here the single actions can be interpreted as follows:

- a *Sender* is requested to transmit data
- b *Sender* sends data along *Medium*
- c *Medium* transmits data correctly
- d *Medium* transmits data incorrectly
- e *Receiver* acknowledges transmission
- f *Receiver* delivers data

Reminder: the one-place buffer is defined by $B(a, f) = a.\overline{f}.B(a, f)$.

Solution 2

Unrealistic: direct, error-free connection from Receiver to Sender (action e).
Transition system (S = Sender, R = Receiver, M = Medium; without `new`):



One-place buffer: $B \underset{\overline{f}}{\overset{a}{\rightleftarrows}} \overline{f}.B \Rightarrow Protocol \approx B$

Solution 2

Moreover *Protocol* and B can only execute a

$\Rightarrow$ *Protocol* $\simeq B$ (because neither *Protocol* nor B can execute a τ-action)