Testing Techniques 2021 – 2022
Tentamen

January 24, 2022

1 Model-Based Testing

Consider the labelled transition systems q_1, q_2, q_3, and q_4 in Fig. 1. These systems model *queues* with input $?in$ and outputs $!out$ and $!full$.

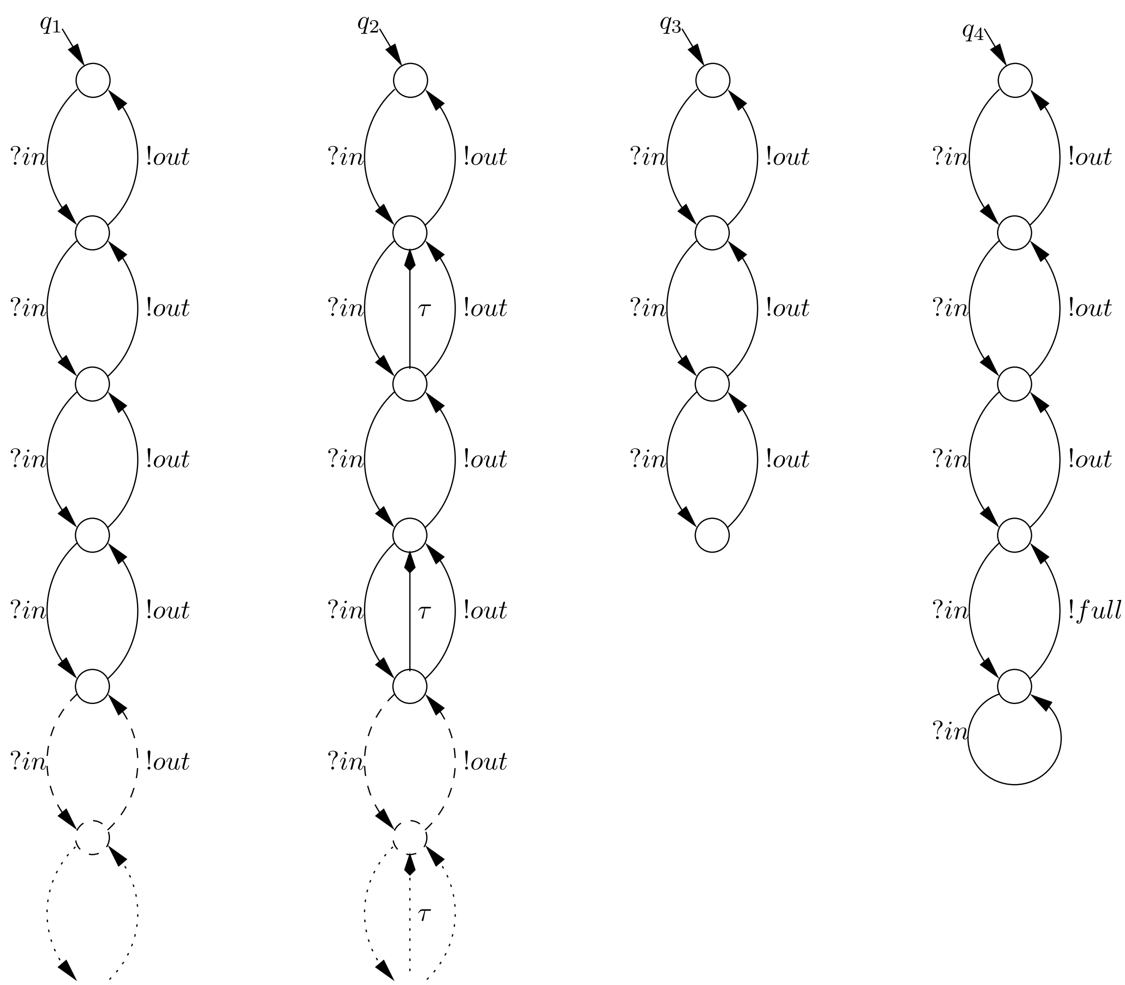

Figure 1: Four models of queues.

System q_1 represents an unbounded queue; the dotted lines at the bottom of q_1 are meant to indicate that there are infinitely many states, and that there is no bound on the number of $?in$ actions that can be performed after each other. System q_2 is also an unbounded queue, but it is a lossy queue: every second input may get lost. Queues q_3 and q_4 are bounded queues with capacity three, the difference being that q_4 gives an explicit message when the queue is full.

a. Which of the systems q_1, q_2, q_3, q_4 are *input-enabled*? Why?

Answer
For q_1, q_2, and q_4 all inputs, i.e., $?in$, are enabled in all states:
for $i = 1, 2, 4$, $\forall q \in Q_i, \forall a \in \{?in\} : q \xrightarrow{a}$.
For q_3 this is not the case: in the lowest state input $?in$ is not enabled.
So, q_1, q_2, and q_4 are input-enabled; q_3 is not. □

b. Consider **ioco** as implementation relation:

$$i \;\mathbf{ioco}\; s \quad \Longleftrightarrow_{\text{def}} \quad \forall \sigma \in \mathit{Straces}(s) : \; \mathit{out}(\, i \;\mathbf{after}\; \sigma\,) \subseteq \mathit{out}(\, s \;\mathbf{after}\; \sigma\,)$$

Take q_3 as specification and q_4 as implementation. Is q_4 an **ioco**-correct implementation of q_3, i.e., does q_4 **ioco** q_3 hold? Explain.

Answer
We have that q_4 **ioco** q_3 holds. The only difference between q_4 and q_3 is that input $?in$ is not specified in the lowest state of q_3 whereas it is enabled in the lowest state of q_4. This is allowed according to **ioco**, since $?in{\cdot}?in{\cdot}?in{\cdot}?in \notin \mathit{Straces}(q_3)$, so any behaviour after this trace is allowed. □

c. Can an unbounded queue correctly implement a bounded queue specification, i.e., does q_1 **ioco** q_3 hold? And if the queue is lossy: does q_2 **ioco** q_3 hold? Explain.

Answer
q_1 **ioco** q_3 holds, because, like above, q_1 allows input $?in$ where it is under-specified in q_3, which is **ioco**-conforming.

q_2 $\mathbf{io\not{c}o}$ q_3: take $\sigma = ?in{\cdot}?in{\cdot}!out \in \mathit{Straces}(q_3)$,
then $\mathit{out}(\, q_2 \;\mathbf{after}\; \sigma\,) = \{!out, \delta\} \not\subseteq \{!out\} = \mathit{out}(\, q_3 \;\mathbf{after}\; \sigma\,)$. □

d. Compare the two unbounded queues: does q_1 **ioco** q_2 or q_2 **ioco** q_1 hold?

Answer
q_1 **ioco** q_2 holds, because, if $\mathit{out}(\, q_1 \;\mathbf{after}\; \sigma\,) \neq \emptyset$, then $q_2 \xRightarrow{\sigma}$ and whatever output q_1 can do after σ, q_2 can do, too.

q_2 $\mathbf{io\not{c}o}$ q_1: take $\sigma = ?in{\cdot}?in{\cdot}!out \in \mathit{Straces}(q_1)$,
then $\mathit{out}(\, q_2 \;\mathbf{after}\; \sigma\,) = \{!out, \delta\} \not\subseteq \{!out\} = \mathit{out}(\, q_1 \;\mathbf{after}\; \sigma\,)$. □

e. Now compare the two unbounded queues for implementation relation **uioco**:

$$\begin{aligned} \mathit{Utraces}(s) \quad &=_{\text{def}} \quad \{\, \sigma \in \mathit{Straces}(s) \mid \forall \sigma_1, \sigma_2 \in L_\delta^*,\; a \in L_I : \\ &\qquad\qquad \sigma = \sigma_1{\cdot}a{\cdot}\sigma_2 \text{ implies not } s \;\mathbf{after}\; \sigma_1 \;\mathbf{refuses}\; \{a\} \;\} \\ i \;\mathbf{uioco}\; s \quad &\Longleftrightarrow_{\text{def}} \quad \forall \sigma \in \mathit{Utraces}(s) : \; \mathit{out}(\, i \;\mathbf{after}\; \sigma\,) \;\subseteq\; \mathit{out}(\, s \;\mathbf{after}\; \sigma\,) \end{aligned}$$

What differs with respect to the previous question? Does q_1 **uioco** q_2 or q_2 **uioco** q_1 hold?

Answer
q_1 **ioco** q_2 and **ioco** $\subseteq$ **uioco**, so q_1 **uioco** q_2.

Since q_1 is deterministic, we have that $\mathit{Straces}(q_1) = \mathit{Utraces}(q_1)$:
let $\sigma \in \mathit{Straces}(q_1)$, such that $\sigma = \sigma_1{\cdot}a{\cdot}\sigma_2$, then $\exists q' : q_1 \xRightarrow{\sigma_1} q' \xRightarrow{a}$. Since q_1 is deterministic, we have that $q_1 \xRightarrow{\sigma_1} q'$ and $q_1 \xRightarrow{\sigma_1} q''$ implies $q' = q''$. So, not $\exists q'' : q_1 \xRightarrow{\sigma_1} q''$ and $\forall \mu \in \{a, \tau\} : q'' \xrightarrow{\mu}\!\!\!\!\!/$, *so* not s **after** σ_1 **refuses** $\{a\}$. It follows that $\sigma \in \mathit{Straces}(q_1)$ implies $\sigma \in \mathit{Utraces}(q_1)$, and **ioco** and **uioco** are the same for q_1 as specification, and consequently, q_2 $\mathbf{uio\not{c}o}$ q_1 .

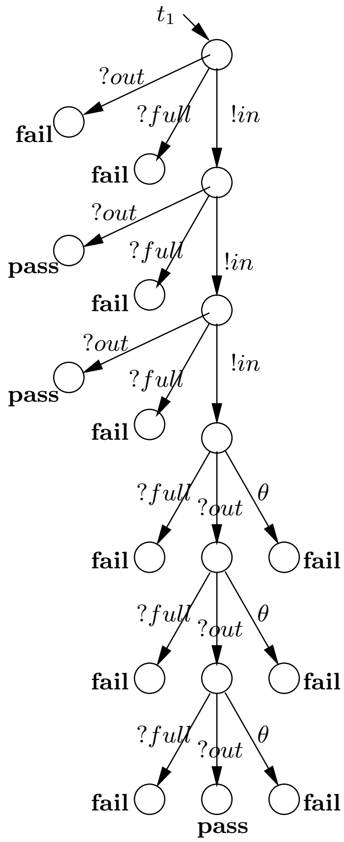

Figure 2: Test case t_1 for queue systems.

□

f. Fig. 2 gives a test case t_1. From which of the models q_1 and q_2 can this test case be generated using the **ioco**-test generation algorithm?

Answer
Test case t_1 can be generated fom q_1, but not from q_2: after three times an $?in$-action, q_2 can be in the second, third, or fourth state (from the top, i.e., from the initial state). This means that only one $!out$ action is guaranteed, and that after the first $!out$, *quiescence* may occur. In test case t_1 this means that the second and third θ-transition should lead to a **pass**-state, i.e., t_1 should be such that t_1 **after** $?in{\cdot}?in{\cdot}?in{\cdot}!out = \{\textbf{pass}\}$ and t_1 **after** $?in{\cdot}?in{\cdot}?in{\cdot}!out{\cdot}!out = \{\textbf{pass}\}$, in order to be generated by the **ioco**-test generation algorithm. □

g. Give the test runs and determine the verdict of executing the test case t_1 on q_2.

Answer

$$t_1 \,]|\, q_2 \xRightarrow{?in\cdot!out} \textbf{pass} \,]|\, q_2^0$$
$$t_1 \,]|\, q_2 \xRightarrow{?in\cdot?in\cdot!out} \textbf{pass} \,]|\, q_2^1$$
$$(\; t_1 \,]|\, q_2 \xRightarrow{?in\cdot?in\cdot!out} \textbf{pass} \,]|\, q_2^0 \;)$$
$$t_1 \,]|\, q_2 \xRightarrow{?in\cdot?in\cdot?in\cdot!out\cdot\theta} \textbf{fail} \,]|\, q_2^0$$
$$t_1 \,]|\, q_2 \xRightarrow{?in\cdot?in\cdot?in\cdot!out\cdot!out\cdot\theta} \textbf{fail} \,]|\, q_2^0$$
$$t_1 \,]|\, q_2 \xRightarrow{?in\cdot?in\cdot?in\cdot!out\cdot!out\cdot!out} \textbf{pass} \,]|\, q_2^0$$

There are passing and failing test runs, so q_2 **fails** t_1.
(which is consistent with the previous question: if t_1 could have been generated from q_2 then the input-enabled process q_2 should have passed its own test case).

$\Box$

h. Is test case t_1 *sound* for specification q_1 with respect to implementation relation **ioco**? And is it *sound* for q_2 with respect to **ioco**? Explain.

Answer
Soundness: $\forall i \in \mathcal{IOTS}(L_I, L_U):\ i \textbf{ ioco } s \text{ implies } i \textbf{ passes } t$

Test case t_1 is sound for specification q_1: t_1 can be generated from q_1 using the **ioco**-test generation algorithm, which generates only sound test cases (Theorem 2.1 of the *MBT with LTS* paper).

Test case t_1 is not sound for q_2:
q_2 **ioco** q_2, since $q_2 \in \mathcal{IOTS}$ (see *a.*) and **ioco** is reflexive on $\mathcal{IOTS}$ (Proposition 2.4 of the *MBT with LTS* paper). Yet, according to *g.*, q_2 **fails** t_1, so t_1 is not sound for q_2.

$\Box$

2 Labelled Transition Systems

Consider the processes P_1, P_2, and P_3, which are represented as labelled transition systems in Figure 3 with labelset $L = \{a, b, c, d\}$.

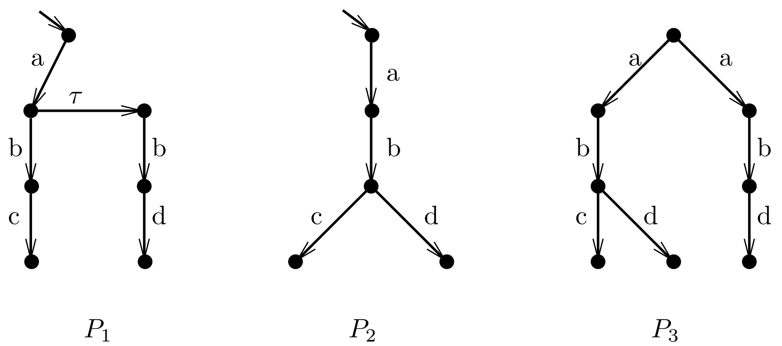

Figure 3:

a. Argue informally whether you consider the following pairs of processes to be equivalent or not, i.e., describe test experiments or observations that can distinguish the processes, or argue why they cannot be distinguished:

1. P_1 and P_2

   Answer
   Try to do $a{\cdot}b{\cdot}d$: P_1 may refuse this trace, whereas P_2 always allows it, so they are not equivalent. □

2. P_2 and P_3

   Answer
   Try to do $a{\cdot}b{\cdot}c$: P_2 always allows this trace, whereas P_3 may refuse it, so they are not equivalent. □

3. P_1 and P_3

   Answer
   Try to do $a{\cdot}b{\cdot}d$: P_1 may refuse this trace, whereas P_3 always allows it, so they are not equivalent. □

b. Consider testing equivalence, that is, perform a trace of actions σ and see which actions can then be refused. Formally:

$$p \approx_{te} q \quad \Longleftrightarrow_{\text{def}} \quad \forall \sigma \in L^*,\ \forall A \subseteq L : \ \ p\,\mathbf{after}\,\sigma\,\mathbf{refuses}\,A \ \text{ iff } \ q\,\mathbf{after}\,\sigma\,\mathbf{refuses}\,A$$

with $p\,\mathbf{after}\,\sigma\,\mathbf{refuses}\,A \quad \Longleftrightarrow_{\text{def}} \quad \exists p' : \ p \overset{\sigma}{\Longrightarrow} p' \ \text{ and } \ \forall a \in A \cup \{\tau\} : \ p' \overset{a}{\not\longrightarrow}$.

Which of the following pairs of processes are $\approx_{te}$? Why?

1. P_1 and P_2

   Answer
   Not testing equivalent: $P_1\,\mathbf{after}\,a{\cdot}b\,\mathbf{refuses}\,\{d\}$, not $P_2\,\mathbf{after}\,a{\cdot}b\,\mathbf{refuses}\,\{d\}$. □

2. P_2 and P_3

   Answer
   Not testing equivalent: not $P_2\,\mathbf{after}\,a{\cdot}b\,\mathbf{refuses}\,\{c\}$, $P_3\,\mathbf{after}\,a{\cdot}b\,\mathbf{refuses}\,\{c\}$.

□

3. P_1 and P_3

   Answer
   Not testing equivalent: P_1 **after** $a{\cdot}b$ **refuses** $\{d\}$, not P_3 **after** $a{\cdot}b$ **refuses** $\{d\}$. □

c. Now consider the processes as input-output transition systems with $L_I = \{a\}$ and $L_U = \{b, c, d\}$ and completed with self-loops in order to make them input-complete. Which of the following pairs of processes are **ioco**-related?

1. P_1 **ioco** P_2

   Answer
   P_1 **ioco** P_2 holds:
   $out(\, P_1 \textbf{ after } \epsilon\,) = out(\, P_2 \textbf{ after } \epsilon\,) = out(\, P_3 \textbf{ after } \epsilon\,) = \{\delta\}$,
   $out(\, P_1 \textbf{ after } a\,) = out(\, P_2 \textbf{ after } a\,) = out(\, P_3 \textbf{ after } a\,) = \{b\}$,
   $out(\, P_1 \textbf{ after } a \cdot b\,) = out(\, P_2 \textbf{ after } a \cdot b\,) = out(\, P_3 \textbf{ after } a \cdot b\,) = \{c, d\}$,
   $out(\, P_1 \textbf{ after } a \cdot b \cdot c\,) = out(\, P_2 \textbf{ after } a \cdot b \cdot c\,) = out(\, P_3 \textbf{ after } a \cdot b \cdot c\,) = \{\delta\}$,
   $out(\, P_1 \textbf{ after } a \cdot b \cdot d\,) = out(\, P_2 \textbf{ after } a \cdot b \cdot d\,) = out(\, P_3 \textbf{ after } a \cdot b \cdot d\,) = \{\delta\}$.
   □

2. P_2 **ioco** P_3

   Answer
   P_2 **ioco** P_3 holds: see above. □

3. P_3 **ioco** P_1

   Answer
   P_3 **ioco** P_1 holds: see above. □