Testing Techniques 2019 – 2020
Tentamen

January 15, 2020 – 8:30–11:30/12:00 h. – HG00.071 / HG00.622

1 Testing with ioco

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 output $!out$.

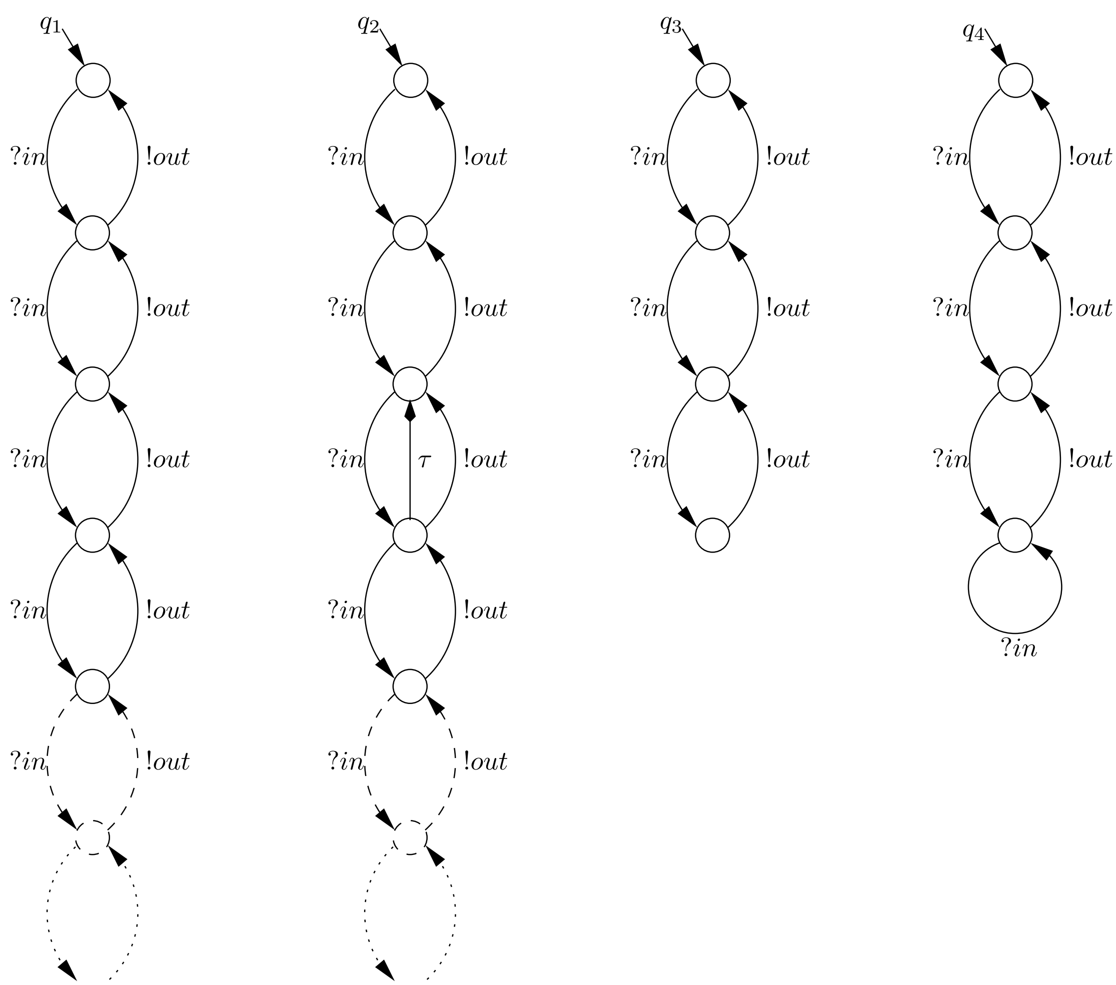

Figure 1: Four models of queues.

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: the third input can get lost. Queues q_3 and q_4 are bounded queues with capacity three, the difference being that q_4 explicitly neglects additional inputs.

a. Which states of q_2 are *quiescent*? Why?

Answer
The initial state of q_2, q_{2_0}, is quiescent: $\forall x \in L_U \cup \{\tau\} : q_{2_0} \xrightarrow{x}\!\!\!\!\!\!/\;\;$. □

b. 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: $\forall q \in Q_i, \forall a \in \{?in\} : q \overset{a}{\Longrightarrow}$.
For q_3 this is not the case: in the lowest state input $?in$ is not enabled. □

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

Answer
Using i **ioco** $s \iff_{\text{def}} \forall \sigma \in \mathit{Straces}(s) : \; out(\, i \,\textbf{after}\, \sigma\,) \subseteq out(\, s \,\textbf{after}\, \sigma\,)$, 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**. □

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

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

e. What can you say about the inverse: can a bounded queue correctly implement an unbounded (lossy) queue specification, i.e., q_4 **ioco** q_1 or q_4 **ioco** q_2? Explain.

Answer
q_4 **io$\not$co** q_1: take $\sigma = ?in\cdot?in\cdot?in\cdot?in\cdot!out\cdot!out\cdot!out \in \mathit{Straces}(q_1)$, then $out(\, q_4 \,\textbf{after}\, \sigma\,) = \{\delta\} \not\subseteq \{!out\} = out(\, q_1 \,\textbf{after}\, \sigma\,)$.

q_4 **ioco** q_2: any ouput of q_4 can be simulated by q_2 by going sufficiently often through the τ-transition; in particular, after n inputs ($n \geq 2$), q_2 can produce any number of outputs between 2 and n, followed by δ.
Consequently, $\forall \sigma \in \mathit{Straces}(q_2) : \; out(\, q_4 \,\textbf{after}\, \sigma\,) \subseteq out(\, q_2 \,\textbf{after}\, \sigma\,)$. □

f. We have that $q_3 \overset{\sigma}{\not\Longrightarrow}$ with $\sigma = ?in\cdot?in\cdot?in\cdot?in$. Moreover, $out(\, q_3 \,\textbf{after}\, \sigma\,) = \emptyset$ for this σ.
Argue that this holds in general for any system $p \in \mathcal{LTS}(L)$ and any $\sigma \in (L \cup \{\delta\})^*$, i.e.,

$$p \overset{\sigma}{\not\Longrightarrow} \quad \text{iff} \quad out(\, p \,\textbf{after}\, \sigma\,) = \emptyset$$

Answer

- *only if*:

	$p \overset{\sigma}{\not\Longrightarrow}$
iff	(* Def. $\not\Rightarrow$ *)
	not $\exists p' : p \overset{\sigma}{\Longrightarrow} p'$
iff	(* Def. **after** *)
	p **after** $\sigma = \emptyset$
implies	(* Def. *out* *)
	$out(\, p \,\mathbf{after}\, \sigma \,) = out(\emptyset) = \bigcup\{out(p) \mid p \in \emptyset\} = \emptyset$

- *if*: By contraposition.

	$p \overset{\sigma}{\Longrightarrow}$
iff	(* Def. $\Rightarrow$ *)
	$\exists p' : p \overset{\sigma}{\Longrightarrow} p'$
implies	(* Def. quiescence *)
	either there is some output $x \in L_U$ such that $p' \overset{x}{\Longrightarrow}$,
	or there is no output at all, in which case p' is quiescent: $\delta(p')$
implies	(* Def. *out* *)
	$(\ \exists x \in L_U : x \in out(p')\ )$ or $(\ \delta \in out(p')\ )$
implies	$out(p') \neq \emptyset$
implies	(* Def. $out(\, p \,\mathbf{after}\, \sigma\,)$; $p' \in p \,\mathbf{after}\, \sigma$ *)
	$out(\, p \,\mathbf{after}\, \sigma \,) \neq \emptyset$

The complete proof is not necessary; yet, the main argument, i.e., that any stable state has either an output, or quiescence, must be given.

Note: For a completely formal proof the *non-divergence* of p is required. (*2 bonus points!*)

□

g. Fig. 2 gives a test case t_1. Give the test runs and determine the verdicts of executing the test case t_1 on q_2.

Answer

$$t_1 \rceil\!\mid q_2 \xRightarrow{?in\cdot !out} \mathbf{pass} \rceil\!\mid q_2^0$$
$$t_1 \rceil\!\mid q_2 \xRightarrow{?in\cdot ?in\cdot !out} \mathbf{pass} \rceil\!\mid q_2^1$$
$$t_1 \rceil\!\mid q_2 \xRightarrow{?in\cdot ?in\cdot ?in\cdot !out\cdot !out\cdot !out} \mathbf{pass} \rceil\!\mid q_2^0$$
$$t_1 \rceil\!\mid q_2 \xRightarrow{?in\cdot ?in\cdot ?in\cdot !out\cdot !out\cdot \theta} \mathbf{fail} \rceil\!\mid q_2^0$$

The first three test runs pass; the last one fails. So, q_2 **fails** t_1.

□

h. For which of the specifications q_1, q_2, q_3, or q_4, is test case t_1 *sound* with respect to implementation relation **ioco**? Explain.

Answer

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

or: $\forall i \in \mathcal{IOTS}(L_I, L_U) :\ i \,\mathbf{fails}\, t$ implies $i \,\mathbf{io\not{c}o}\, s$

Test case t_1 is sound for q_1, q_3, and q_4: t_1 can be generated from these specifications 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 is input-enabled (see b.), so q_2 **ioco** q_2 (lecture notes prop. 2.2 and 1.4), but q_2 **fails** t_1 (ee g.), so t_1 is not sound for q_2.

□

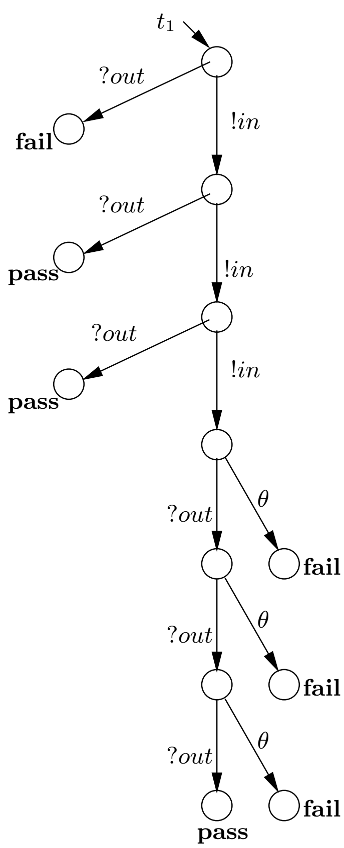

Figure 2: Test case t_1 for queue systems.