Testing Techniques 2023 – 2024
Tentamen

January 22, 2024

- This examination consists of 4 assignments, with weights 2, 2, 3, and 4, respectively.
- The exam has 6 pages, numbered from 1 to 6.
- You are not allowed to use any material during the examination, except for pen and paper, and
  - the paper: *Tretmans: Model Based Testing with Labelled Transition Systems* (38 pages);
  - the slide set: *Vaandrager: Black Box Testing of Finite State Machines* (152 slides);
  - the slide set: *Kruger, Vaandrager: Model Learning* (115 slides).
- Use one or more separate pieces of paper per assignment.
- Write clearly and legibly.
- Give explanations for your answers to open questions, but keep them concise.
- We wish you a lot of success!

1 Equivalence

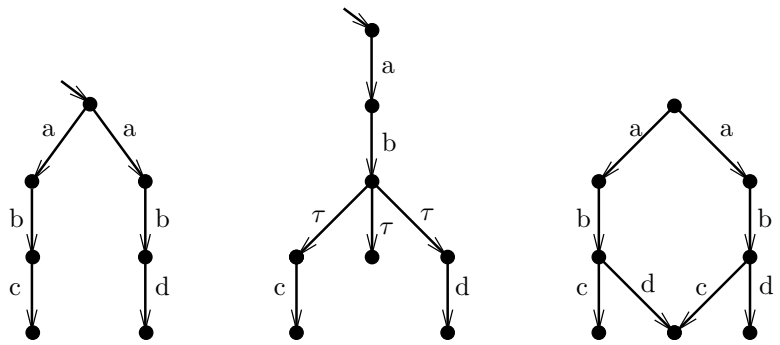

Figure 1:

Consider the processes P_1, P_2, and P_3, which are represented as labelled transition systems in Figure 1, with labelset $L = \{a, b, c, d\}$.

Consider the following definitions:

$$
\begin{array}{rcl}
p \leq_{tr} q & \Longleftrightarrow_{\text{def}} & \mathit{traces}(p) \subseteq \mathit{traces}(q) \\
p \approx_{tr} q & \Longleftrightarrow_{\text{def}} & p \leq_{tr} q \text{ and } q \leq_{tr} p \\
p \approx_{ct} q & \Longleftrightarrow_{\text{def}} & \mathit{traces}(p) = \mathit{traces}(q) \text{ and } \mathit{Ctraces}(p) = \mathit{Ctraces}(q) \\
p \approx_{te} q & \Longleftrightarrow_{\text{def}} & \forall \sigma \in L^*,\ \forall A \subseteq L : \\
 & & \quad p \textbf{ after } \sigma \textbf{ refuses } A \text{ iff } q \textbf{ after } \sigma \textbf{ refuses } A \\
p \textbf{ after } \sigma \textbf{ refuses } A & \Longleftrightarrow_{\text{def}} & \exists p' : p \stackrel{\sigma}{\Longrightarrow} p' \text{ and } \forall a \in A \cup \{\tau\} : p' \stackrel{a}{\not\longrightarrow} \\
\mathit{Ctraces}(p) & =_{\text{def}} & \{\, \sigma \in L^* \mid p \textbf{ after } \sigma \textbf{ refuses } L \,\}
\end{array}
$$

Compare the processes P_1, P_2, and P_3 according to

a. trace equivalence $\approx_{tr}$;

b. completed trace equivalence $\approx_{ct}$;

c. testing equivalence $\approx_{te}$.

d. Prove that $\approx_{ct}$ is strictly stronger than $\approx_{tr}$, i.e., $\approx_{ct} \subset \approx_{tr}$ and $\approx_{ct} \neq \approx_{tr}$.

2 Conformance

Consider the labelled transition systems in Fig. 2 with $L_I = \{?a.?b\}$ and $L_U = \{!x, !y\}$.

a. Which of the implementations i_1, i_2, i_3 are **uioco**-conforming to specification s, and why?

b. For each of the implementations i_j, $j = 1, 2, 3$, give a test case t_j, if such a test case exists, such that

- t_j is generated from s using the **uioco**-test generation algorithm; and
- t_j fails with i_j.

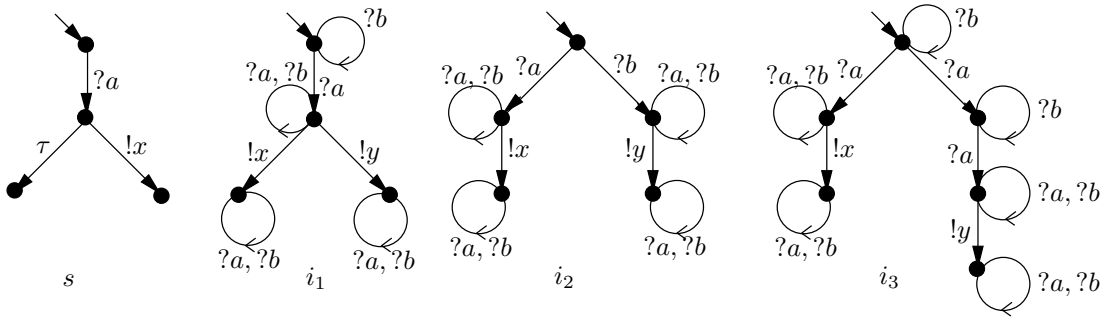

Figure 2:

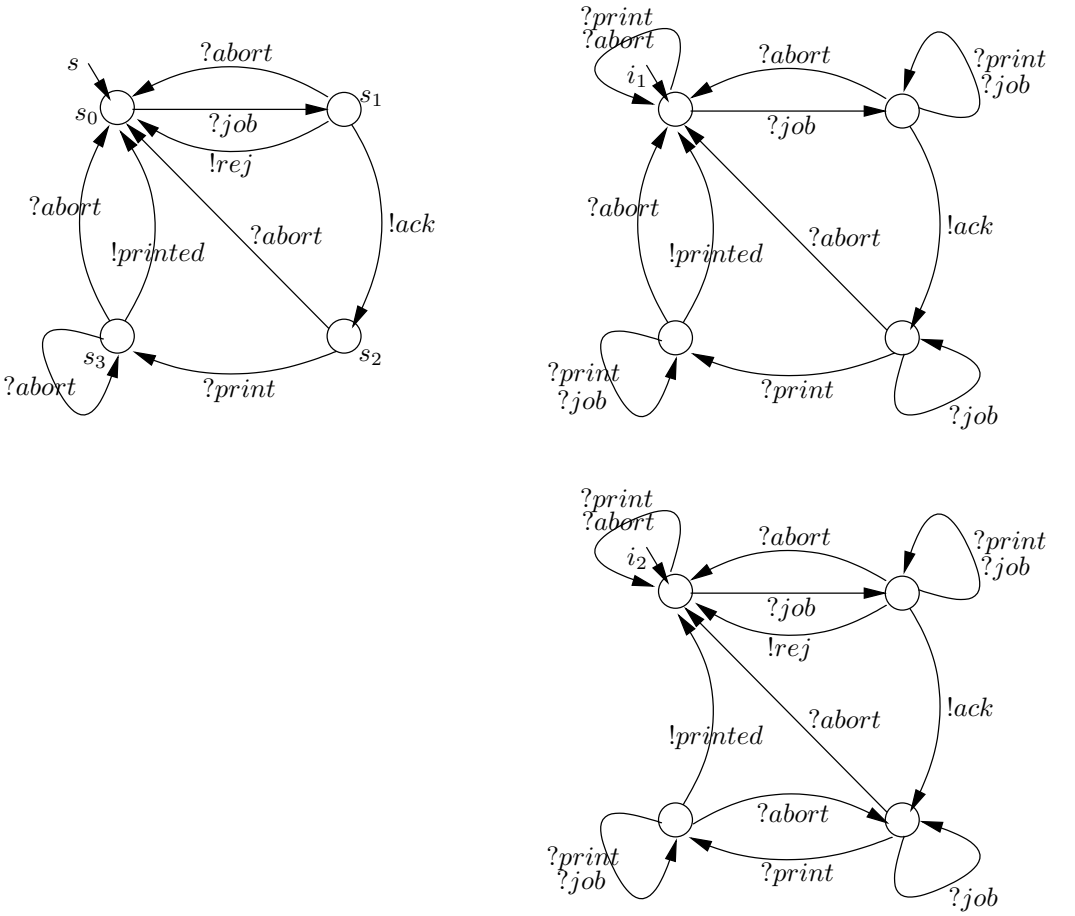

Figure 3: Models of printers, with $L_I = \{?job, ?print, ?abort\}$ and $L_U = \{!ack, !rej, !printed\}$.

3 Model-Based Testing

Consider the labelled transition systems s, i_1, and i_2 in Fig. 3, that represent printers. A user can submit a printer job with ?*job*, after which the printer indicates whether the submitted job is well-formed or not, via !*ack* and !*rej*, respectively. A well-formed job can be printed using the ?*print*-command, which produces the !*printed* output. During the process, a user can ?*abort* the printing, after which the printer will go to the initial state again, except if after the ?*print*-command the user is too slow with her ?*abort*, and the printing has already started and cannot be aborted anymore.

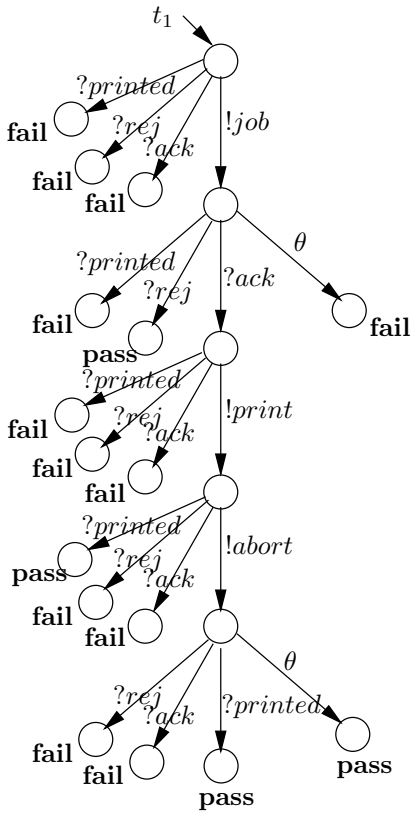

Figure 4: Test case t_1.

a. Which states of s are *quiescent*, and why?

b. Is s *deterministic*, and why?

c. Consider **uioco** as implementation relation:

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

Give a trace of s that is element of $Straces(s)$ but not of $Utraces(s)$.

d. Consider the trace ?*job*·!*ack*·?*print*·?*abort*. How can you observe that the user was 'too slow' to abort the printing? How can you observe that the user was 'fast enough' to abort the printing?

e. The implementation i_1 has a feature that auto-repairs not well-formed jobs. Therefore, the !*rej* output is not used. Moreover, i_1 has an AI-based module which takes care that the ?*abort* is always 'fast enough' to abort the printing.

Is the implementation i_1 a **uioco**-correct implementation of s, and why?

f. Implementation i_2 also has the AI-based module so that ?*abort* is always 'fast enough'. In i_2, however, ?*abort* after ?*print* does not throw away the job, by going back to the initial state, but goes back to the state before ?*print*, so that the user does not have to re-submit the job, when she wants to print it later.

Is the implementation i_2 a **uioco**-correct implementation of s, and why?

g. Figure 4 shows the test case t_1 for the *printer*. Test case t_1 aims at testing the output of ?*abort* after ?*print*. Give the test run(s) and verdict of applying t_1 to implementation i_2.

h. Can test case t_1 be generated from s with the **uioco**-test generation algorithm, and why?

i. Show that test case t_1 is not exhaustive for specification s.

4 Model Learning

In 2012, Arjan Blom, then a student at Radboud University, performed a security analysis of the E.dentifier2 system of the ABN AMRO bank, in which customers use a USB-connected device — a smartcard reader with a display and numeric keyboard — to authorise transactions with their bank card and PIN code. Arjen found a security vulnerability in the E.dentifier2 that was so serious that he even made it to the evening news on Dutch national TV. He did not use systematic testing techniques to find this vulnerability, but in 2014 Erik Poll and colleagues showed that black-box testing of FSMs and model learning could easily reveal the problem with the E.dentifier2. This assignment is based on the FSM models described by Erik Poll et al.

Figure 5 shows the FSM S that specifies the behavior of the E.dentifier2. There are three states $\{q_0, q_1, q_2\}$, five inputs $\{C, D, G, R, S\}$, and four outputs $\{C, L, T, OK\}$. Note that we use commas to indicate multiple transitions. For instance, in S there is both a transition $q_1 \xrightarrow{C/OK} q_1$ and a transition $q_1 \xrightarrow{R/T} q_1$.

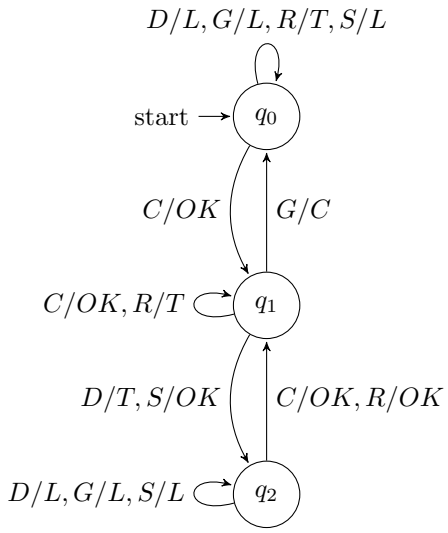

Figure 5: FSM S that specifies the required behavior of the E.dentifier2.

a. Give an access sequence set for S.

b. Give a distinguishing sequence for S or show that this does not exist.

c. Give a UIO for each of the three states of S or show that these do not exist.

d. Give a characterization set for S.

e. Give a 0-complete test suite T for S that is minimal in the sense that when any test in T is omitted it is no longer 0-complete. Explain why your test suite is minimal.

Figure 6 shows the FSM model M for the faulty implementation of the E.dentiefier2.

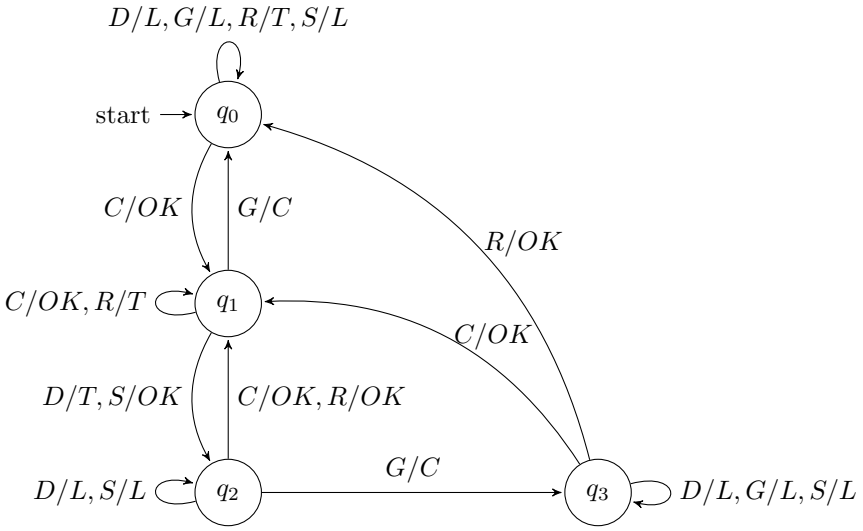

Figure 6: FSM M that describes the faulty implementation of the E.dentifier2.

f. Give a test from your test suite T (if any) that demonstrates that implementation M does not satisfy specification S.

g. Describe a test suite that would reveal for any implementation FSM with at most four states whether or not it satisfies specification S.

h. Describe the details of a run of either L^* or $L^\#$ (the choice is yours!) when used to learn specification S. You may also select the counterexamples provided by the teacher.

In the case of L^*, describe the initial rows and columns of the observation table, the specific reason for adding additional rows or columns, all intermediate hypotheses and corresponding counterexamples, and the final table.

For $L^\#$, describe the specific sequence of rule applications to extend the observation tree, all intermediate hypotheses and corresponding counterexamples, and the final observation tree.

The End