Testing Techniques 2022 – 2023
Tentamen

January 16, 2023

- This examination consists of 4 assignments, with weights 2, 2, 3, and 4, respectively.
- The exam has 5 pages, numbered from 1 to 5.
- 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* (62/154 slides);
  - the slide set: *Vaandrager: Model Learning* (121 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

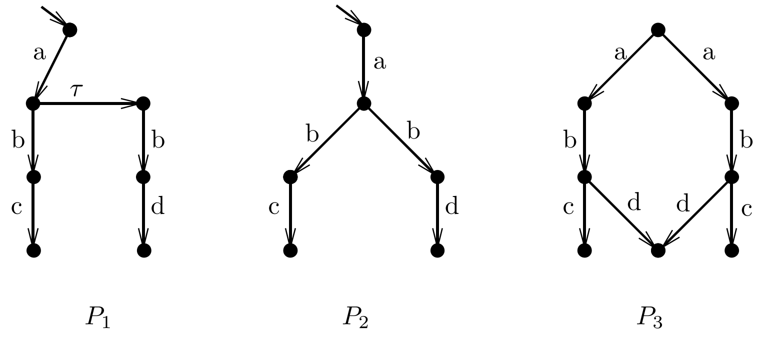


Figure 1:

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

Compare the processes P_1, P_2, and P_3 according to testing equivalence:

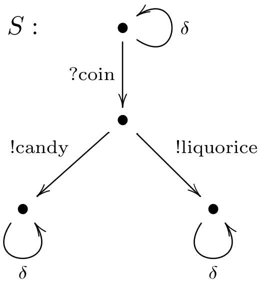
$$p \approx_{te} q \quad \Longleftrightarrow_{\text{def}} \quad \forall \sigma \in L^*,\ \forall A \subseteq L : \ \ p \textbf{ after } \sigma \textbf{ refuses } A \ \text{ iff } \ q \textbf{ after } \sigma \textbf{ refuses } A$$
$$p \textbf{ after } \sigma \textbf{ refuses } A \quad \Longleftrightarrow_{\text{def}} \quad \exists p' : p \overset{\sigma}{\Longrightarrow} p' \text{ and } \forall a \in A \cup \{\tau\} : p' \overset{a}{\not\longrightarrow}$$

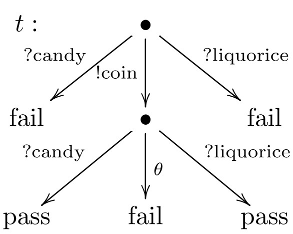
Which pairs of processes are testing equivalent?

b. Consider again the processes P_1, P_2, and P_3 in Figure 1. When more powerful experiments can be made than those that are possible with testing equivalence $\approx_{te}$, e.g., doing an *undo*, or taking snapshots of states, then more processes can be distinguished than only the ones which are not testing equivalent. Which processes can be distinguished with more powerful experiments and how?

2 Conformance

A company that produces sweets provides the following specification S, with $L_I = \{?\text{coin}\}$ and $L_U = \{!\text{candy}, !\text{liquorice}\}$. From S, they obtained a test case t, using the **uioco**-test derivation algorithm.





a. Is there an implementation that is *not* **uioco**-conforming to S, but that passes t? If yes, then give such an implementation.

b. Is there an implementation that is *not* **uioco**-conforming to S, and that fails t? If yes, then give such an implementation.

c. Is the test suite $\{ t \}$ sound, and why?

d. Is the test suite $\{ t \}$ exhaustive, and why?

3 Model-Based Testing

What goes up, must come down: Consider the labelled transition systems s, i_1, i_2, and i_3 in Fig. 2, where you can go up by giving input $?u$, after which the system can put you down through $!d$. When you are too high up, you can also fall down with output $!f$. The specification also allows that sometimes when you try to go up, you will not manage and you stay at the same hight.

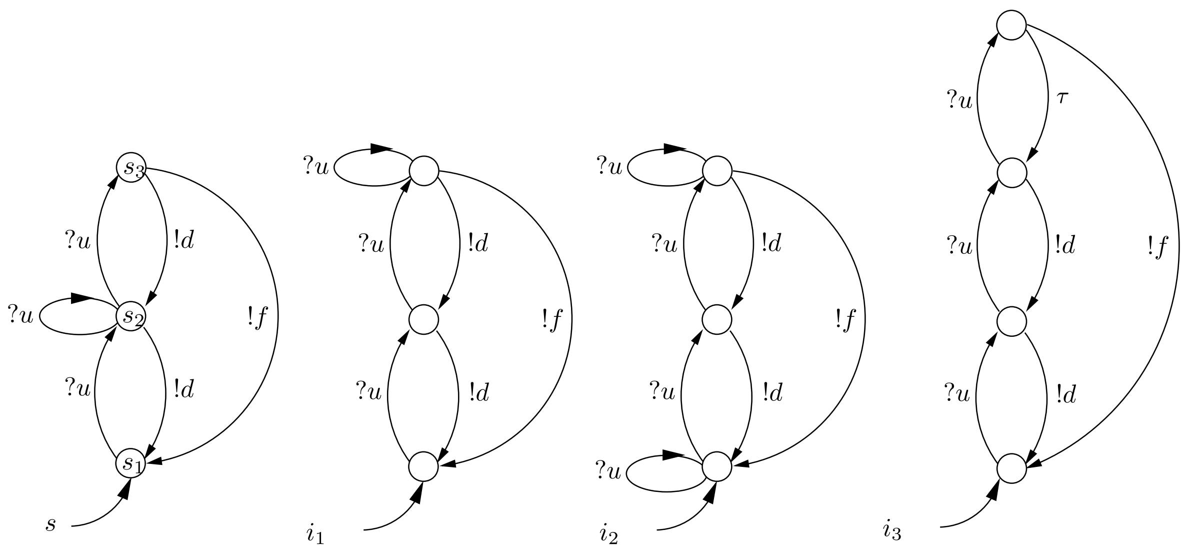


Figure 2: Models of *What goes up, must come down.*

a. Which states of i_3 are *quiescent*, and why?

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

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

Consider the traces $?u{\cdot}?u$ and $?u{\cdot}?u{\cdot}?u$. Are they an element of $\mathit{Straces}(s)$ and/or of $\mathit{Utraces}(s)$, i.e., $?u{\cdot}?u \in \mathit{Straces}(s)$, $?u{\cdot}?u \in \mathit{Utraces}(s)$, $?u{\cdot}?u{\cdot}?u \in \mathit{Straces}(s)$, $?u{\cdot}?u{\cdot}?u \in \mathit{Utraces}(s)$, and why?

c. Is the implementation i_1 an **uioco**-correct implementation of s? Why?

d. Is the implementation i_2 an **uioco**-correct implementation of s? Why?

e. Is the implementation i_3 an **uioco**-correct implementation of s? Why?

f. Figure 3 shows the test case t_1 for the *up-down*-system. Give the test run(s) and verdict of applying t_1 to implementation i_3.

g. Can test case t_1 be generated from s with the **uioco**-test generation algorithm? (or, if you prefer, from the **ioco**-test generation algorithm?)

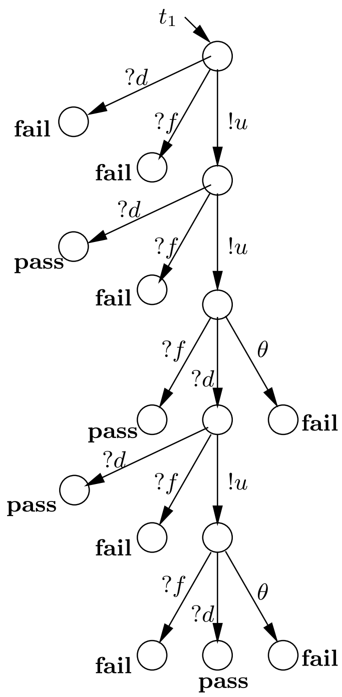


Figure 3: Test case t_1.

h. Is test case t_1 *sound* with respect to s and **uioco**, and why (not)?
(or, if you prefer, with respect to s and **ioco**?)

i. We have that $s \xRightarrow{?u\cdot?u}$ and $out(\, s\, \textbf{after}\ ?u\cdot?u\,) = \{!d, !f\} \neq \emptyset$.
Argue that this holds in general (a formal proof is not necessary), i.e.,

$$s \xRightarrow{\sigma} \quad \text{implies} \quad out(\, s\, \textbf{after}\ \sigma\,) \neq \emptyset$$

4 Model Learning

Consider the FSM $\mathcal{M}$ of Figure 4. This machine always outputs 0 in response to an input, except in one specific situation. Output 1 is produced in response to input b if the previous input was a and the total number of preceding inputs is odd. Consider a scenario where a System Under Test (SUT) behaves like $\mathcal{M}$, and a learner uses the $L^{\#}$ algorithm to infer a model of this SUT.

After posing output queries a and b, the learner constructs the initial hypothesis $\mathcal{H}_1$ shown in Figure 5(right).

The learner now wants to construct a test suite in order to either find a counterexample for hypothesis $\mathcal{H}_1$, or to obtain some confidence in its correctness.

a. Give an access sequence set for $\mathcal{H}_1$.

b. Explain why the empty set is a characterization set for $\mathcal{H}_1$.

c. Explain why the empty set is not a 0-complete test suite for $\mathcal{H}_1$

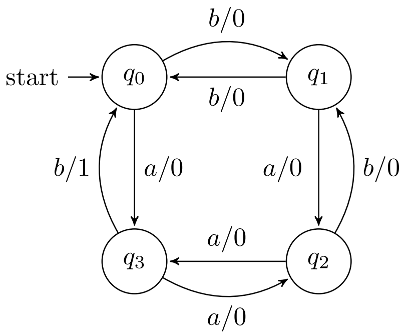


Figure 4: FSM $\mathcal{M}$ that describes the behavior of the SUT.

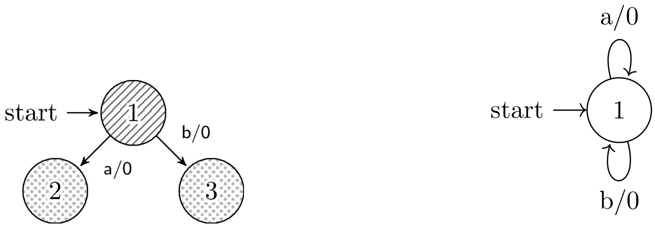


Figure 5: Observation tree after queries a and b (left) and first hypothesis $\mathcal{H}_1$ (right)

d. Give a minimal 0-complete test suite for $\mathcal{H}_1$. Explain why your test suite is minimal, and why it will not find a counterexample for $\mathcal{H}_1$.

e. Give a minimal 1-complete test suite for $\mathcal{H}_1$. Which test from this suite will demonstrate that the SUT does not conform to $\mathcal{H}_1$ (and thus provide a counterexample for $\mathcal{H}_1$?).

f. Describe how $L^{\#}$ uses the counterexample from (e) to construct a second hypothesis $\mathcal{H}_2$. Draw the observation tree that is constructed as well as the corresponding hypothesis $\mathcal{H}_2$.

g. Describe an n-complete test suite, for minimal n, that demonstrates that the SUT does not conform to $\mathcal{H}_2$.

h. Describe how $L^{\#}$ uses a counterexample found by the test suite from (g) to construct a third hypothesis $\mathcal{H}_3$. Draw the observation tree that is constructed as well as the corresponding hypothesis $\mathcal{H}_3$.

i. Explain why $\mathcal{H}_3$ and $\mathcal{M}$ are equivalent, or provide a counterexample.

The End