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!

Grading: Total

assignment	1	2	3	4	grade
points	max 20	max 20	max 30	max 40	total/11

1 Equivalence

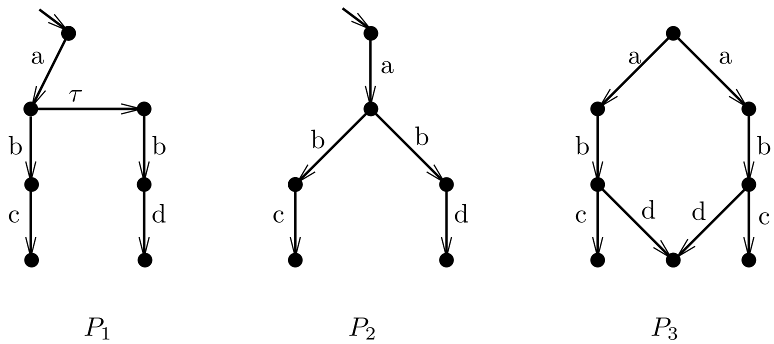


Figure 1:

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?

Answer

1. $P_1 \approx_{te} P_2$: $\forall \sigma \in L^*,\ \forall A \subseteq L$: P_1 **after** σ **refuses** A iff P_2 **after** σ **refuses** A
2. $P_1 \not\approx_{te} P_3$: P_1 **after** $a{\cdot}b$ **refuses** $\{d\}$, not P_3 **after** $a{\cdot}b$ **refuses** $\{d\}$.
3. $P_2 \not\approx_{te} P_3$: P_2 **after** $a{\cdot}b$ **refuses** $\{d\}$, not P_3 **after** $a{\cdot}b$ **refuses** $\{d\}$.

□

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?

Answer
P_1 and P_3, and P_2 and P_3, are not testing equivalent, so they can already be distinguished by doing the above refusal-experiment: do $a{\cdot}b$ and then try d, then in P_3 this is always successful, but in P_1 and P_2, d might be refused.

P_1 and P_2 can be distinguished with the following experiment:

> do a and take a snapshot and make infinitely many copies;
> with each copy, try b and then try c,
> repeat this whole experiment infinitely often.

Then you can make the following observations with P_1:
if after a, the snapshot is in the left-hand state of P_1, then some copies will refuse c after b;
if after a, the snapshot is in the right-hand state of P_1, then all copies will refuse c after b.
So, by repeatung this experiment, sometimes you will see that all copies refuse c after b.

With P_2 there is only one state after a, in which you will observe that some copies will refuse c after b. Even by repeating the expriment infinitely often, you will never see that all copies will refuse c after b.

In this way, you can distinguish P_1 and P_2 with this observation.

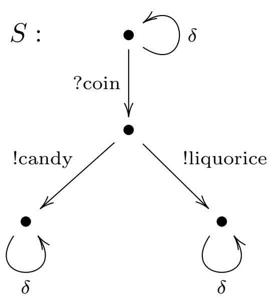
□

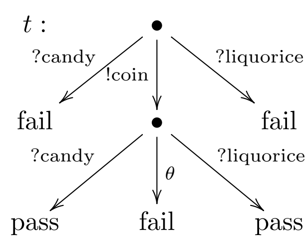
Grading: Assignment 1

a	b	points
12	8	max 20

2 Conformance

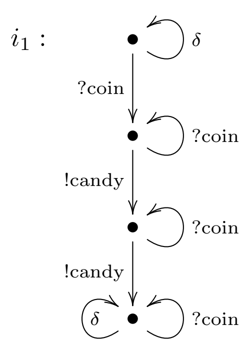
A company that produces sweets provides the following specification S, with $L_I = \{?\text{coin}\}$ and $L_U = \{!\text{candy}, !\text{liquorice}\}$. From S, they obtained a test case t, using the **uioco**-test derivation algorithm.

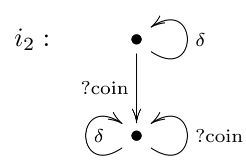




a. Is there an implementation that is *not* **uioco**-conforming to S, but that passes t? If yes, then give such an implementation.

Answer
Yes, there is, e.g., the implemetation i_1 that is not quiescent after ?coin·!candy:





i_1 $\mathbf{uio\not{c}o}$ s, because
$out(\, i_1\ \mathbf{after}\ ?\text{coin}\cdot!\text{candy}\,) = \{!\text{candy}\} \not\subseteq \{\delta\} = out(\, s\ \mathbf{after}\ ?\text{coin}\cdot!\text{candy}\,)$,
and, moreover, i_1 **passes** t, because $t\rceil\!|\, i_1 \xRightarrow{!\text{coin}\cdot?\text{candy}} \mathbf{pass}\rceil\!|\, i_1''$ is the only test run, and it passes.

□

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

Answer
Yes, there is, e.g., the implemetation i_2 that is quiescent after ?coin.

i_1 $\mathbf{uio\not{c}o}$ s, because
$out(\, i_2\ \mathbf{after}\ ?\text{coin}\,) = \{\delta\} \not\subseteq \{!\text{candy}, !\text{liquorice}\} = out(\, s\ \mathbf{after}\ ?\text{coin}\cdot!\text{candy}\,)$,
and, moreover, i_2 **fails** t, because $t\rceil\!|\, i_2 \xRightarrow{!\text{coin}\cdot\theta} \mathbf{fail}\rceil\!|\, i_2'$.

□

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

Answer
Yes, it sound, because t can be obtained using the **uioco**-test generation algorithm.

Or, in other words, any **fail** in t corresponds to non-conformance, i.e., any trace $\sigma \cdot x \in (L \cup \{\delta\})^* \cdot (L_U \cup \{\delta\})$ leading to **fail**, has that $x \notin out(\, s \;\mathbf{after}\; \sigma \,)$.

□

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

Answer
No, it is not exhaustive, since for i_1 above we have that $i_1 \mathrel{\not\mathbf{uioco}} s$, yet i_1 **passes** t.

Or, in other words, the not **uioco**-conforming implementation i_1 is not detected by test suite $\{ t \}$, so $\{ t \}$ is not exhaustive.

□

Grading: Assignment 2

a	b	c	d	points
5	5	5	5	max 20

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.

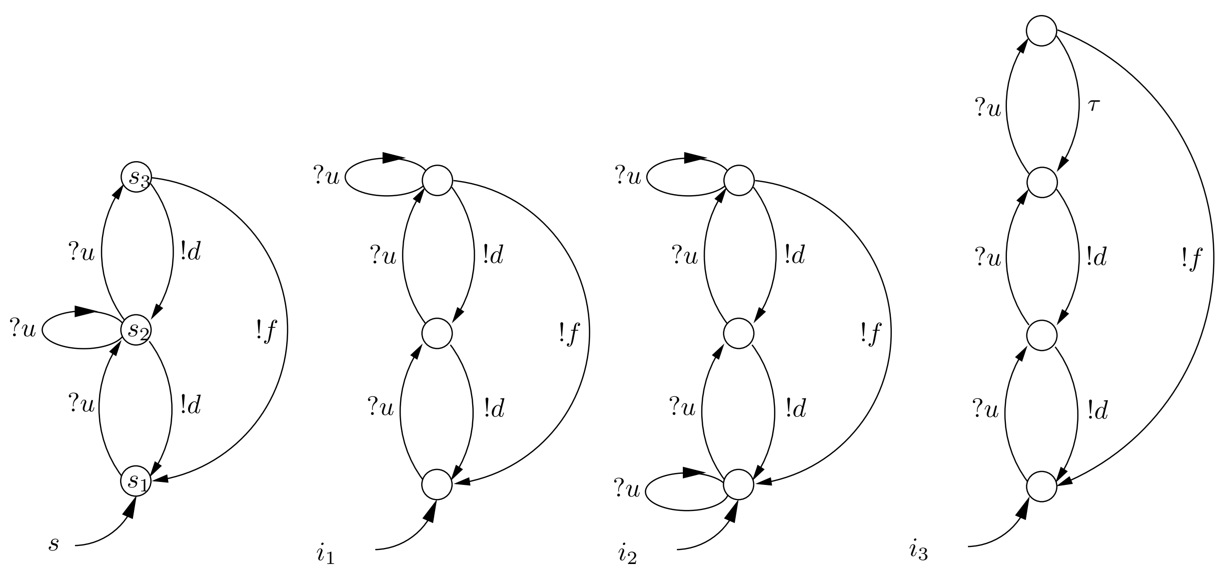


Figure 2: Models of *What goes up, must come down.*

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

Answer
The initial state of i_3 is quiescent: $\forall x \in L_U \cup \{\tau\} : {i_3}_0 \overset{x}{\not\longrightarrow}$.

□

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

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

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?

Answer
$s \xRightarrow{?u\cdot ?u}$ and $s \xRightarrow{?u\cdot ?u\cdot ?u}$, so, $?u\cdot ?u \in \mathit{Straces}(s)$ and $?u\cdot ?u\cdot ?u \in \mathit{Straces}(s)$.

We have that not s **after** ε **refuses** $\{?u\}$
not s **after** $?u$ **refuses** $\{?u\}$
s **after** $?u\cdot ?u$ **refuses** $\{?u\}$
implying that $?u\cdot ?u \in \mathit{Utraces}(s)$, but $?u\cdot ?u\cdot ?u \notin \mathit{Utraces}(s)$.

□

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

Answer
i_1 **uioco** s holds: after doing a number $?u$-actions, you can reach a state of i_1 which can also be reached in s, with the same outputs. So, after any trace of s, the outputs of i_1 can also be produced by s, so, i_1 **uioco** s.

□

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

Answer
$i_2\ \mathbf{uio\!\!\!/co}\ s$: $\mathit{out}(\, i_2\ \mathbf{after}\ ?u\,) = \{\, \delta, !d \,\} \not\subseteq \{\, !d \,\} = \mathit{out}(\, s\ \mathbf{after}\ ?u\,)$.

□

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

Answer
i_1 **uioco** s holds: the top two states of i_3 behave as the top state of s, producing either $!d$ or $!f$ as output, which is allowed according to s.

□

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.

Answer

$$
\begin{array}{lll}
t_1 \rceil\!| \; i_3 & \xRightarrow{!u\cdot ?d} & \mathbf{pass} \rceil\!| \; i_{3_0} \\
t_1 \rceil\!| \; i_3 & \xRightarrow{!u\cdot !u\cdot ?d\cdot ?d} & \mathbf{pass} \rceil\!| \; i_{3_0} \\
t_1 \rceil\!| \; i_3 & \xRightarrow{!u\cdot !u\cdot ?d\cdot !u\cdot ?d} & \mathbf{pass} \rceil\!| \; i_{3_1}
\end{array}
$$

All test runs pass, so i_3 **passes** t_1.

□

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?)

Answer
No, t_1 cannot be generated from s with the **uioco**-test generation algorithm. If generated, the verdict in the lowest $?f$-branch, i.e., the verdict after $!u\cdot !u\cdot ?d\cdot !u\cdot ?f$, should be **pass**.

Alternatively, using the result of *h.*: if t_1 were generated with the **uioco**-test generation algorithm, then it would be sound, as all test cases generated with the algorithm are sound, according to the theorem. Item *h.* shows that t_1 is not sound, so it is not generated.

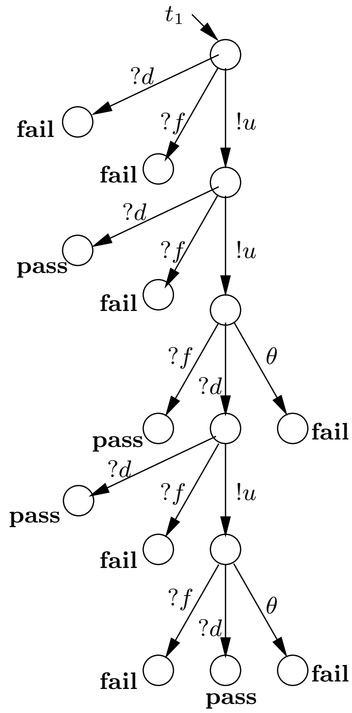


Figure 3: Test case t_1.

□

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**?)

Answer
Soundness: $\forall i \in \mathcal{IOTS}(L_I, L_U):\ i\ \mathbf{uioco}\ s$ implies $i\ \mathbf{passes}\ t_1$,
or: $\forall i \in \mathcal{IOTS}(L_I, L_U):\ i\ \mathbf{fails}\ t_1$ implies $i\ \not\mathbf{uioco}\ s$.

The test case t_1 is not sound, because $out(\,s\ \mathbf{after}\ ?u\cdot?u\cdot?d\cdot?u\,) = \{\ d,\ \ f\,\}$, so output $!f$ is allowed, but test cases t_1 gives the verdict **fail** when output $!f$ is observed after trace $?u\cdot?u\cdot?d\cdot?u$.

Alternatively, $t_1 \rceil\!|\, i_1 \xrightarrow{!u\cdot!u\cdot?d\cdot!u\cdot?f} \mathbf{fail} \rceil\!|\, i_{1_0}$, so i_1 **fails** t_1, whereas i_1 **uioco** s holds, see above, so t_1 is not sound.

□

i. We have that $s \overset{?u\cdot?u}{\Longrightarrow}$ and $out(\,s\ \mathbf{after}\ ?u\cdot?u\,) = \{!d, !f\} \neq \emptyset$.

Argue that this holds in general (a formal proof is not necessary), i.e.,

$$s \overset{\sigma}{\Longrightarrow} \quad \text{implies} \quad out(\,s\ \mathbf{after}\ \sigma\,) \neq \emptyset$$

Answer
If $s \overset{\sigma}{\Longrightarrow}$ then $\exists s' : s \overset{\sigma}{\Longrightarrow} s'$. For this state s', either it holds that s' can do some output $x \in L_U$, in which case $x \in out(\,s\ \mathbf{after}\ \sigma\,)$, so $out(\,s\ \mathbf{after}\ \sigma\,) \neq \emptyset$; or it holds that s' cannot do an output, in which case s' is quiescent, i.e., $\delta(s')$ holds, and, consequently, $\delta \in out(\,s\ \mathbf{after}\ \sigma\,)$, so also in this case $out(\,s\ \mathbf{after}\ \sigma\,) \neq \emptyset$.

□

Grading: Assignment 3

a	b	c	d	e	f	g	h	i	points
3	4	3	3	3	3	3	4	4	max 30

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.

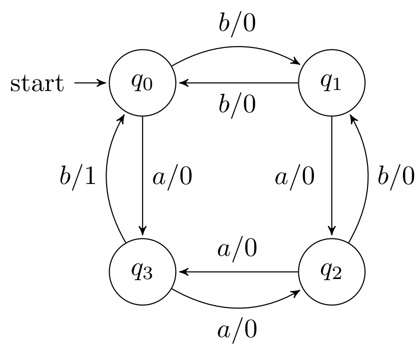


Figure 4: FSM $\mathcal{M}$ that describes the behavior of the SUT.

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

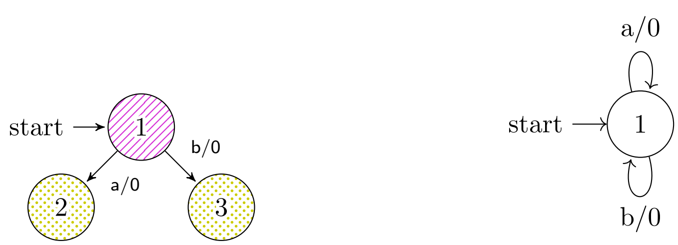


Figure 5: Observation tree after queries a and b (left) and first hypothesis $\mathcal{H}_1$ (right)

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 $\mathcal{H}_1$

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.

Solutions and Correction Guidelines.

a. **(3pts)** An *access sequence* for a state q is a sequence of inputs σ such that $\delta^*(q^0, \sigma) = q$. A *set of access sequences* for an FSM is a set containing an access sequence for each state of the FSM; we require that ϵ is in this set. Since ϵ is an access sequence for the only state 1 of $\mathcal{H}_1$, a set of access sequences for $\mathcal{H}_1$ is $A = \{\epsilon\}$.
(When students just give the correct set without any explanation they get maximal points. However, when they give an access sequences set for $\mathcal{M}$ like $A = \{\epsilon, b, ba, a\}$ they get no points since they clearly misread the question.)

b. **(3pts)** A *characterization set* for an FSM $M = (Q, q_0, I, O, \delta, \lambda)$ is a set $C \subseteq I^*$ that contains a separating sequence for every pair of states $q, q' \in Q$ (with $q \neq q'$). Here a separating sequence for state q and q' is a sequence $\sigma \in I^*$ such that $\lambda^*(q, \sigma) \neq \lambda^*(q', \sigma)$. Since FSM $\mathcal{H}_1$ only has a single state the empty set is a characterization set: for every pair of distinct states (there is no such pair!) the empty set contains a separating sequence.

c. **(4pts)** A test suite T is *n-complete* for a specification S if, for any implementation I with at most n more states than S, I passes T iff I is equivalent to S. In particular, a test suite T is 0-complete for FSM $\mathcal{H}_1$ if, for any implementation I with a single state, I passes T iff I is equivalent to $\mathcal{H}_1$. The empty test suite T is not 0-complete for $\mathcal{H}_1$, because the following FSM I, which has only one state, passes T even though it is not equivalent to $\mathcal{H}_1$:

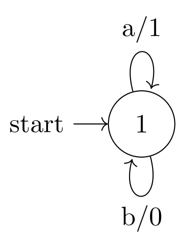


Figure 6: Incorrect implementation

d. **(6pts)** As explained during the lectures, a 0-complete test suite for FSM $\mathcal{H}_1$ is

$$T \quad = \quad A \cdot C + A \cdot I + A \cdot I \cdot C$$

where $A = \{\epsilon\}$ is the set of access sequences given in item a), $C = \emptyset$ is the characterisation set from item b), and $I = \{a, b\}$ is the set of inputs. We can simplify this to $T = I = \{a, b\}$. (Note that the observation from the lecture that $A \cdot I$ only contains prefixes from $A \cdot I \cdot C$ is only valid when C is nonempty.) This test suite is minimal because when we leave out test a we can no longer rule out the incorrect implementation of Figure 6, and if we leave out test b we can not rule out the incorrect implementation of Figure 7.

(Students may earn 3pts for the correct test suite, 2pts for the explanation why it is minimal, and 1pt for explanation why it does not find counterexample for $\mathcal{H}_1$.)

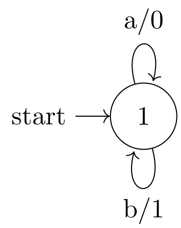


Figure 7: Another incorrect implementation

e. **(4pts)** A minimal 1-complete test suite for $\mathcal{H}_1$ is $T = \{aa, ab, ba, bb\}$. Just like in (d), we can show that whenever we leave out a test from T there exists a faulty implementation (with two states) that is not eliminated by the resulting test suite. The test ab will fail when applied to the SUT and demonstrate that it does not conform to $\mathcal{H}_1$.

(Students earn 2pts for the correct test suite, 1pt for the explanation why it is minimal, and 1pt for counterexample for $\mathcal{H}_1$.)

f. **(6pts)** After we have add counterexample ab to the observation tree, states 1 and 2 are apart through witness b. So we add state 2 to the basis and extend the frontier with state 5. Next we identify the three frontier states using the witness b. Figure 8(left) shows the observation tree from which hypothesis $\mathcal{H}_2$ from Figure 8(right) is constructed.

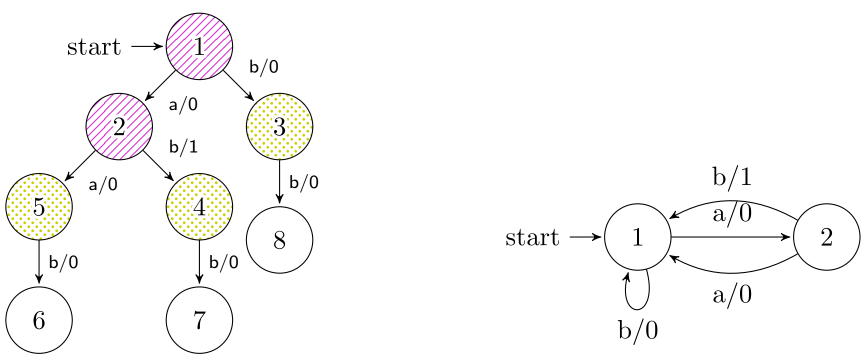


Figure 8: Extended observation tree (left) and hypothesis $\mathcal{H}_2$ (right)

(Students earn 2pts for a good explanation, 2pts for the correct observation tree, and 2pts for the correct hypothesis. Deduct 1pt if frontier states are not identified.)

g. **(4pts)** A set of access sequences for $\mathcal{H}_2$ is $A = \{\epsilon, a\}$, and a characterization set is $C = \{b\}$. As explained during the lecture (and since now C is nonempty),

$$T_n \quad = \quad A \cdot I^{\leq n+1} \cdot C$$

is an n-complete test suite for $\mathcal{H}_2$, for any $n \geq 0$. Thus 0-complete test suite T_0 equals

$$T_0 \quad = \quad \{b, ab, bb, aab, abb\}$$

Test suite T_0 does not reveal any problem with hypothesis $\mathcal{H}_2$: all tests pass. Test suite T_1, however, will contain a test bab (take access sequence ϵ, infix ba, and separating sequence b) that constitutes a counterexample for $\mathcal{H}_2$. Note that, since T_0 is 0-complete, also the test suite $T_0 \cup \{bab\}$ is 0-complete. In fact, also T_n is 0-complete, for any $n > 0$.

(2pts for the test suite, 1pt for explanation that n is minimal (both 0 and 1 are correct when explained), 1pt for test that reveals that $\mathcal{H}_2$ is incorrect.)

h. **(6pts)** After adding counterexample bab to the observation tree, we note that now states 1 and 3 are apart (witness ab). Thus state 3 can be added to the basis. Next we run een output query ab from each of the four frontier states in order to identify them, and construct hypothesis $\mathcal{H}_3$. Figure 9 shows the final observation tree and corresponding hypothesis $\mathcal{H}_3$.

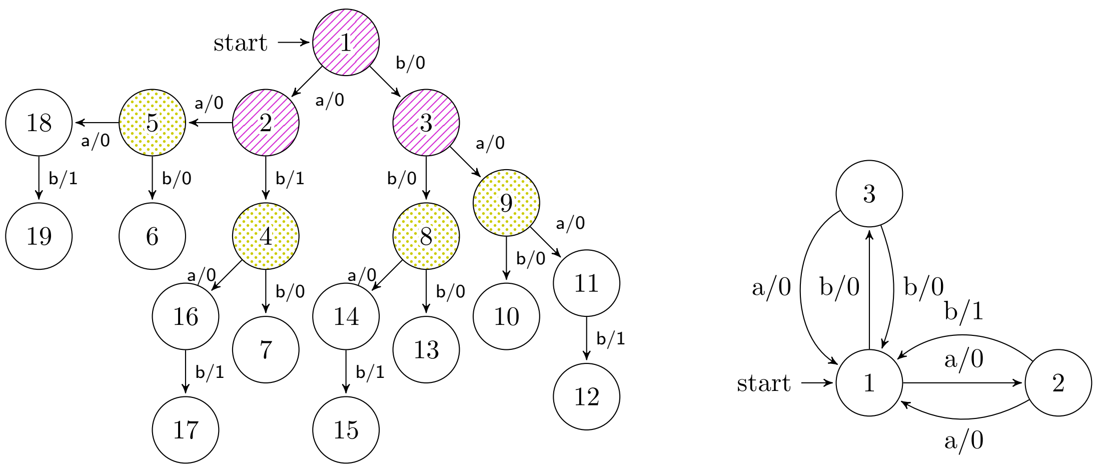


Figure 9: Extended observation tree (left) and hypothesis $\mathcal{H}_3$ (right)

(Students earn 2pts for a good explanation, 2pts for the correct observation tree, and 2pts for the correct hypothesis.)

i. **(4pts)** Hypothesis $\mathcal{H}_3$ and $\mathcal{M}$ are equivalent! FSM $\mathcal{M}$ is not minimal, as states q_0 and q_2 are equivalent. This is easy to see since (1) both q_0 and q_2 have an outgoing a-transition with output 0 leading to state q_3, and (2) both q_0 and q_2 have an outgoing b-transition with output 0 leading to state q_1. Another way to show that $\mathcal{H}_3$ and $\mathcal{M}$ are equivalent is to establish a bisimulation between them (as defined on slide 39 of the slides on FSM testing). It is routine to check that the following relation constitutes a bisimulation between FSMS $\mathcal{H}_3$ and $\mathcal{M}$:

$$R \quad = \quad \{(1, q_0), (1, q_2), (2, q_3), (3, q_1)\}.$$

(Students may earn maximal points if their $\mathcal{H}_3$ is incorrect but relation with $\mathcal{M}$ is correctly explained.)

The End