Exam Testing Techniques (192170015)
— 11 April 2014 —

To make this exam:

- You are allowed to have 1 A4 sheet with your notes and nothing else.
- Make each exercise on a separate page.
- Write your name on each separate page that you hand in.
- Hand in the exam as well.

Division of points:

exercise 1:	3
exercise 2:	31
exercise 3:	19
exercise 4:	20
exercise 5:	27

We wish you a lot of success!

1 What is testing?

Exercise 1 (Test automation)

1. What is the purpose of testing?
2. In his guest lecture, Machiel van der Bijl presented how Axini uses model-based testing in practice. Describe two alterations he made to the ioco-based testing framework to facilitate the use of model-based testing in practice.

2 Classical Testing.

1. The function

   ```
   search_step: ARRAY[REAL] x REAL --> INT
   ```

   takes an array A of reals and a real y. The array A represents a function by giving the values $(1, A[1]), (2, A[2]), \ldots, (n, A[n])$. We assume that A is indexed from 1 to $length(A)$. We also assume that A is strictly monotonically increasing, i.e. $\forall 1 \leq i < j \leq length(A)\ .\ A[i] \leq A[j]$.

   Now, given A and y, `search_step(A,y)` finds the largest integer i, $1 \leq i \leq length(A)$ such that $A[i] \leq y$, or 0 if no such i exists. If the array is not strictly monotonically increasing, then an appropriate error value is returned.

   We assume that this function has been type checked, so you do not have to check for wrong input types, or the wrong number of arguments.

   (a) Use the equivalence partitioning technique to divide the input into suitable equivalence classes and give a test suite that covers all equivalence classes.

   (b) Extend the test suite using the principle of boundary value analysis.

   (c) Extend the test suite using the principle of error guessing.

   (d) For one test case, give a narrative as presented in the lecture by Christoph Bockisch.

   Present each test suite in a table.

2. We consider the following transformation on programs. Given a program A, we obtain a program A' by replacing all statements of the form

   ```
   if C || D
   then S
   endif
   ```

   by

   ```
   if C
   then
       S
   else if D
       then S
       endif
   endif
   ```

   (a) Do these transformations preserve decision coverage?

       (I.e. if a test suite T has 100% decision coverage on program A, does T have 100% decision coverage for the transformed program A' as well?) If your answer is "no", present a program A, its transformation A', and a test suite T such that T has 100% coverage on A, but not on A'.

       If yes, explain why, otherwise give a counter example.

   (b) Same question, but for condition coverage.

   (c) Is the following true: if an arbitrary program transformation preserves decision coverage, it also ~~statement decision coverage~~? If so, explain why. If not, provide a counter example (I.e. a program transformation with the desired properties).

$$\begin{tikzcd}
s_1 \arrow[r, "a?/b!"] & s_2 \arrow[dl, "a?/c!"] \\
s_3 \arrow[u, "a?/b!"] &
\end{tikzcd}$$

Figure 1: The FSM A; s_1 is initial.

3 FSM testing

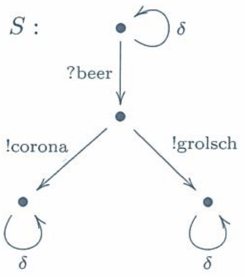
Consider the FSM A in Figure 1. Recall that the definition of completeness assumes that the number of states in the implementation FSM is less or equal to the number of state in the specification. Below, we assume that ~~both~~ FSMs start from the initial states.

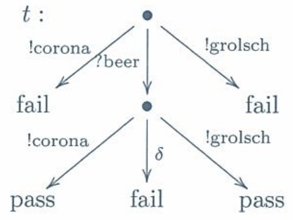
1. Give a test suite T_1 for A consisting of 3 tests that is sound, and a test suite T_2 for A consisting of 3 tests that is not sound.
2. Is the test suite $\{a?/b!\ a?/c!\ a?/b!\}$ complete? Explain your answer.
3. Is $\{a?/b!\ a?/c!\ a?/b!\ a?/b!\ a?/c!\ a?/b!\}$ complete? Explain your answer.
4. For the FSM above, give the following sets and sequences, if they exist
   - a transition tour
   - a W-set
   - a transfer sequence
   - a synchronizing sequence

   No explanations are required!
5. Suppose you have tested a system implementation using FSM transition testing, and it passed all tests. Give four reasons why the implementation could not work as desired, despite the fact that FSM transition testing is complete.

4 Test derivation for ioco

A company that produces beer tenders provides the following specification S, with $L_I = \{?\text{beer}\}$ and $L_O = \{!\text{grolsch}, !\text{corona}\}$. From S, they obtained a test case t, using the ioco-test derivation algorithm.





1. If possible, give implementation DQTSs i_1, i_2, i_3, i_4 over L_I and L_O with the following properties. If an implementation does not exist, explain why.

   (a) i_1 ioco-conforming to S, and passes t.

   (b) i_2 ioco-conforming to S, and fails t.

   (c) i_3 not ioco-conforming to S, and passes t.

   (d) i_4 not ioco-conforming to S, and fails t.

2. Provide an annotated test suite T that is sound and complete w.r.t. S. Explain why T is complete.

3. Provide two annotated test cases that are inconsistent w.r.t. S.

5 Proofs

We consider several transformations on DQTSs. Are the statements below true? If so, give a detailed proof otherwise give a counter example. We assume the following.

- Let I, I', R, R' be DQTSs with label set $L = \langle L_I, L_O \rangle$.
- We assume that I, I' and R are input-enabled; R' does not need to be so.
- We assume that I, I', R, R' do not contain any τ-transitions.

1. If I **ioco** R and R **ioco** R' then I **ioco** R'.
2. If $I \approx_{tr} I'$ and I **ioco** R, then I' **ioco** R.
3. If I **ioco** R, then $I \approx_{tr} R$.
4. Same questions as above, but now R is not input-enabled. Only give a true/false answer; no proofs or counter example is needed.
   (a) If I **ioco** R and R **ioco** R' then I **ioco** R'.
   (b) If $I \approx_{tr} I'$ and I **ioco** R, then I' **ioco** R.
   (c) If I **ioco** R, then $I \approx_{tr} R$.