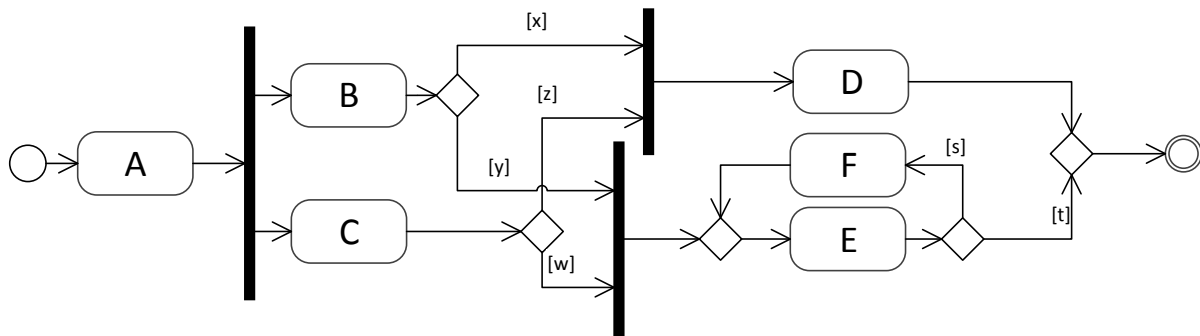


4th Seminar – Analysis and Testing of Models

1 Static analysis of process models

Let's analyse the following process model.



- What conditions will ensure that the process model is fully specified (i.e. consistent, without contradiction)?
- What further conditions will ensure that the process model is also deterministic?
- What further conditions will ensure that the process model is also free of deadlock?
- What further conditions will ensure that the process model will eventually terminate?
- Is the process model well-structured? If not, then how could we make it into an equivalent, but well-structured model? Does this help with the previous problems?

2 Dynamic analysis with testing

We have the following *requirements* for function $f()$:

R1 Function $f()$ must produce at least one output during its execution.

R2 Function $f()$ must terminate eventually regardless of the input sequence.

R3 The last output produced by function $f()$ must be 0.

The following code demonstrates a possible implementation of the function in the C programming language:

```

1 int readInput();
2 void writeOutput(int out);
3
4 void f() {
5     int x = readInput();
6     int y = readInput();
7     int z = x + y;
8     writeOutput(x * y);
9     while (x > 0 && y > 0) {
10        if (1 == readInput() % 2) {
11            y--;
12            z--;
13        } else {
14            x--;
15            y++;
16        }
17        writeOutput(z + x * y * y - x - y);
18    }
19 }

```

Let's verify the correctness of the function in the following steps!

- Model the control flow of $f()$ as a process model!
- Why can we be sure that requirement R1 holds?
- Why can we be sure that requirement R2 holds?
- (Extra task.) Create a state machine that is equivalent with function $f()$. Model the `readInput()` calls as an input channel and the `writeOutput()` calls as an output channel. Model the termination

of the function by producing a special output in the state machine while transitioning into an absorbing state (a state without outgoing transitions).

- e. We are verifying requirement R3 using testing. Our test case is the input sequence $t_1 = \langle 2, 3, 5, 7, 11, 13, \dots \rangle$. Do we detect any errors?
- f. Calculate *instruction coverage* on the process model: during the execution of the test case what portion of the tested code was executed. How does this metric relate to the control flow model?
- g. Our second test case for verifying R3 is the input sequence $t_2 = \langle 1, 2, 4, 1, 2, 4, \dots \rangle$. Do we detect any errors? How much is the combined instruction coverage of the test suite consisting of the two above test cases?
- h. (Extra task.) What kind of test coverage metrics can be calculated on the previous state machine?
- i. Create a *test oracle* automaton that can decide whether R3 holds or not based on the input and output sequence and termination of $f()$. What does the oracle do in case of the previous test inputs?
- j. Give a test case that detects an error in the program! What implicated that our previous test cases weren't perfect and we needed to extend them?
- k. (Homework.) Let's add the input sequences $t_3 = \langle 0, 1, 2, 3, 4, 5, \dots \rangle$ and $t_4 = \langle 1, 2, 3, 4, 5, 6, \dots \rangle$ to our test case collection. Do we detect any errors? How does the instruction coverage change?
- l. Specify the exact conditions under which requirement R3 doesn't hold and recommend some fixes!