Exam on testing – Spring 2005

Stavros Tripakis

March 17, 2005

1 Testing Mealy Machines

1.1 Question M1

Look at the machine M_1 of Figure 1:

1. Is M_1 minimal? If not, build an equivalent minimal machine.

   ANSWER: No. Merge states 1 and 2 to obtain the minimal machine.

2. Does M_1 have a homing sequence? If not, why? If yes, which one? Why is it a homing sequence?

   ANSWER: a is a homing sequence because it leads the machine to state 2, independently of the initial state.

3. Does M_1 have a synchronizing sequence? If not, why? If yes, which one? Why is it a synchronizing sequence?

   ANSWER: a is a synchronizing sequence because it leads the machine to state 2, independently of the initial state.

4. Does M_1 have a distinguishing sequence? If not, why? If yes, which one? Why is it a distinguishing sequence?

   ANSWER: The machine does not have a distinguishing sequence because it is not minimal. A non-minimal machine cannot have a distinguishing sequence, because equivalent states cannot be distinguished.

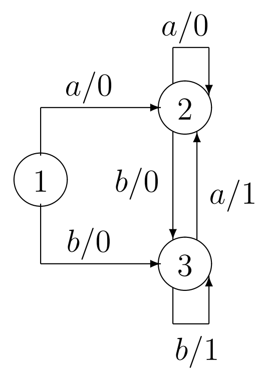

Figure 1: Mealy machine M_1

1.2 Question M2

Look at the machine M_2 of Figure 2:

1. M_2 is minimal. Why?

   ANSWER: States 1 and 3, or 2 and 4, can be distinguished by sequence a. States 1 and 4, or 2 and 3, or 3 and 4, can be distinguished by sequence b. States 1 and 2 can be distinguished by sequence ab.

2. Does M_2 have a preset (non-adaptive) distinguishing sequence? If not, why? If yes, which one?

   ANSWER: No. Such a hypothetic sequence could not start with b, because it "merges" states 1 and 2, which both produce output 0 and lead to state 1. So states 1 and 2 can no longer be distinguished if we start with b. If we start with a, then the initial uncertainty $\{1, 2, 3, 4\}$ is "split" into two groups $\{1, 2\}$ and $\{3, 4\}$, depending on whether we observe 1 or 0, respectively. No matter how many a's we issue after the first a, these two groups of uncertainty remain unchanged. If we issue b at some point, we have the same problem as with the initial b, that is, we "merge" states 1 and 2 of the group $\{1, 2\}$. Thus, a preset distinguishing sequence does not exist.

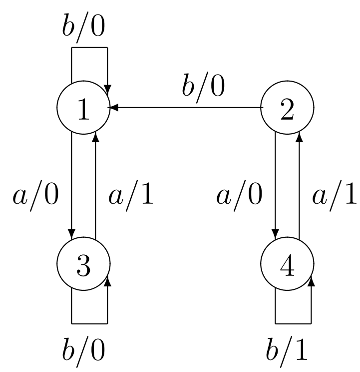

Figure 2: Mealy machine M_2

3. Does M_2 have an adaptive distinguishing test? If not, why? If yes, which one?

ANSWER: Yes:

- issue a,
- if output 0 is observed, you know you started from state 1 or 2 and now you are at state 3 or 4;
  - issue b;
  - if output 0 is observed, you know you started from state 1 and now you are at state 3;
  - if output 1 is observed, you know you started from state 2 and now you are at state 4;
  - end of test.
- if output 1 is observed, you know you started from state 3 or 4 and now you are at state 1 or 2;
  - issue ab;
  - if output 00 is observed, you know you started from state 3 and now you are at state 3;
  - if output 01 is observed, you know you started from state 4 and now you are at state 4;
  - end of test.

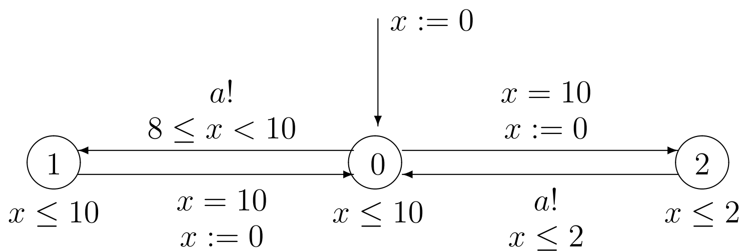

Figure 3: Timed automaton A_1

2 Testing Timed Automata

Look at the timed automaton A_1 of Figure 3. This automaton has one clock x and a single output a. This automaton has no inputs.

We consider this automaton as the specification of a system. What kind of behaviors does this automaton specify? How would you state this specification in English or French?

> ANSWER: "The n-th a, for $n = 1, 2, ...$, arrives somewhere in the interval $[10n - 2, 10n + 2]$".

Hint: draw a time axis and consider the behaviors below[1]:

$$\rho_1 = (a, 10)\ (a, 20)\ (a, 30)\ (a, 40) \cdots \quad (1)$$
$$\rho_2 = (a, 9)\ (a, 18)\ (a, 27)\ (a, 36) \cdots \quad (2)$$
$$\rho_3 = (a, 11)\ (a, 19)\ (a, 31)\ (a, 39) \cdots \quad (3)$$

Is the behavior ρ_1 acceptable according to the specification? If not, why? Is the behavior ρ_2 acceptable according to the specification? If not, why? Is the behavior ρ_3 acceptable according to the specification? If not, why?

> ANSWER: ρ_1 and ρ_3 are conforming. ρ_2 is not conforming: the third a arrives too early.

Suppose we want to test that a given black-box conforms to this specification (with respect to "tioco"). For this, we want to use a digital-clock

[1] Recall that the behavior $(e_1, t_1)\,(e_2, t_2) \cdots$ means that event e_1 occurred at time t_1, then event e_2 occurred at time t_2, and so on.

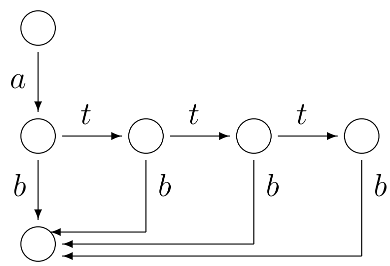

Figure 4: Labeled transition system L_1

tester[2]. We assume that the digital clock of the tester is perfectly periodic with period T. For each of the three values $T = 5$, $T = 2$ and $T = 1$, give an example of a non-conforming behavior that can be "caught" and an example of a non-conforming behavior that cannot be "caught" by the tester. The behaviors must only use event a and time delays.

> ANSWER: For $T = 5$, $(a, 4)$ is a non-conforming execution that the tester can catch: the tester will perceive it as $(a, [0, 5[)$. $(a, 7)$ is a non-conforming execution that the tester cannot catch: the tester will perceive it as $(a, [5, 10[)$, thus it cannot conclude (it could be the behavior $(a, 9)$ which is conforming).

3 "Hybrid" question

For verification, timed automata is not the only way to model behavior with timing constraints. Labeled transition systems (LTS) can be used also, to model discrete time. For instance, the LTS L_1 shown in Figure 4 models time using a special event t which represents one time unit. L_1 expresses the property "b occurs at most three time units after a".

Can you use the same idea for testing? That is, can you express input/output specifications with discrete time constraints as LTS with inputs, outputs and internal actions? How would you express the specification "output b must occur at most four time units after input a is received"? What conformance relation would you use?

[2]Here the tester is in fact a *monitor* since it only observes the black box, without providing any inputs.

ANSWER: Yes, input/output specifications with discrete time constraints as LTS with inputs, outputs and internal actions can be expressed, **by considering t as an output**. The above specification can be expressed by an LTS like the one of Figure 4, where a is replaced by input $a?$, b by output $b!$, and one more state is added for the 4th tick t. I would use “ioco” without quiescence, where t is treated as an output.

Note: for all events that you use in your models, including t, you must specify whether they are inputs or outputs.