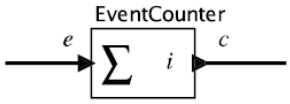
Chapter 3 Discrete Dynamics

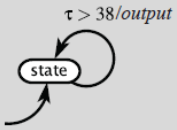
1. Consider an event counter that is a simplified version of the counter in Section 3.1. It has an icon like this:



This actor starts with state i and upon arrival of an event at the input, increments the state and sends the new value to the output. Thus, e is a pure signal, and c has the form $c: \mathbb{R} \to \{absent\} \cup \mathbb{N}$, assuming $i \in \mathbb{N}$. Suppose you are to use such an event counter in a weather station to count the number of times that a temperature rises above some threshold. Your task in this exercise is to generate a reasonable input signal e for the event counter. You will create several versions. For all versions, you will design a state machine whose input is a signal $\tau: \mathbb{R} \to \{absent\} \cup \mathbb{Z}$ that gives the current temperature (in degrees centigrade) once per hour. The output $e: \mathbb{R} \to \{absent, present\}$ will be a pure signal that goes to an event counter.

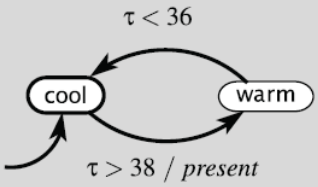
(a) For the first version, your state machine should simply produce a *present* output whenever the input is *present* and greater than 38 degrees. Otherwise, the output should be absent.

Solution: This state machine does not require more than one state:



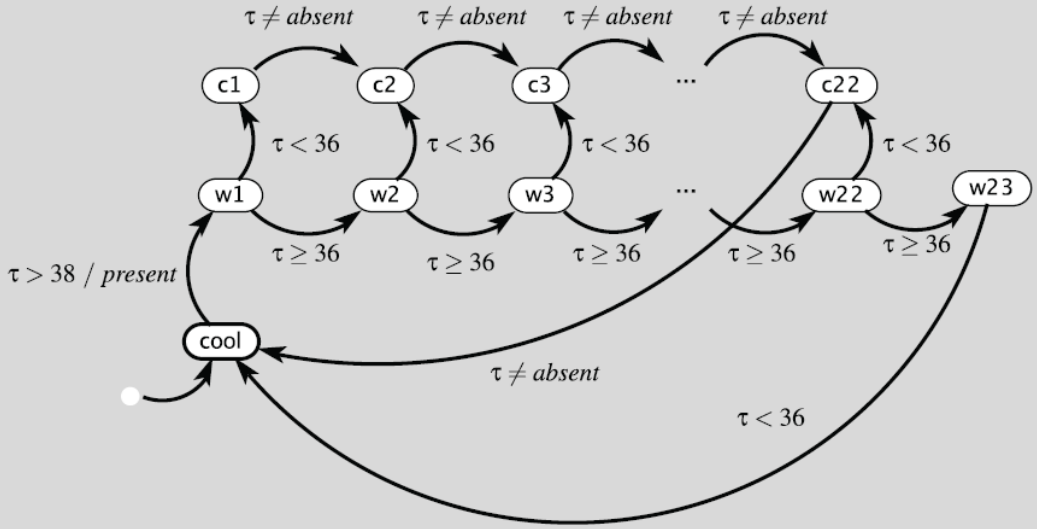
(b) For the second version, your state machine should have hysteresis. Specifically, it should produce a *present* output the first time the input is greater than 38 degrees, and subsequently, it should produce a *present* output anytime the input is greater than 38 degrees but has dropped below 36 degrees since the last time a *present* output was produced.

Solution:



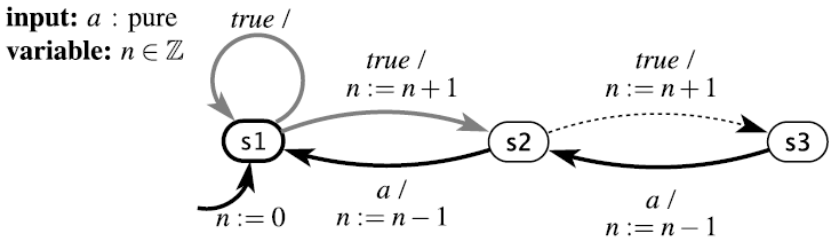
(c) For the third version, your state machine should implement the same hysteresis as in part (b), but also produce a *present* output at most once per day.

Solution: Note that this problem statement is ambiguous. What is meant by "at most once per day?" Is it OK to produce a *present* output at 11 PM and again at 1 AM? Or does it mean that at least 24 hours should elapse between *present* outputs? Either would be correct, given the problem statement. Here is a solution under the second interpretation:



Note that with this solution, if the temperature stays high for several days, there will nonetheless be no *present* output for those several days. Is this likely to be what we intended? The problem appears to ask for this behavior, but it is probably not the behavior we want.

4. How many reachable states does the following state machine have?



Solution: Three. The variable n is redundant, because it always has value 0 in state s1, 1 in s2, and 2 in s3.

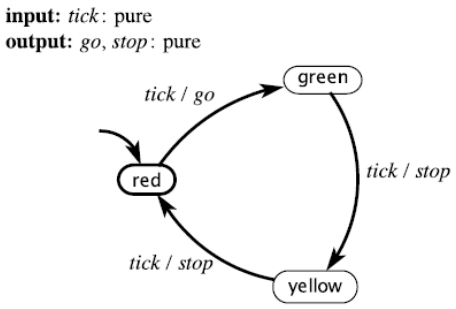


Figure 3.1: Deterministic finite-state machine for Exercise 5

5. Consider the deterministic finite-state machine in Figure 3.1 that models a simple traffic light.

   (a) Formally write down the description of this FSM as a 5-tuple:

$$(States, Inputs, Outputs, update, initialState).$$

Solution: The FSM description is:

$$\begin{aligned} States &= \{\text{red}, \text{yellow}, \text{green}\} \\ Inputs &= (\{tick\} \to \{present, absent\}) \\ Outputs &= (\{go, stop\} \to \{present, absent\}) \\ initialState &= \text{red} \end{aligned}$$

The update function is defined as:

$$update(s,i) = \begin{cases} (\text{green}, go) & \text{if } s = \text{red} \wedge i(tick) = present \\ (\text{yellow}, stop) & \text{if } s = \text{green} \wedge i(tick) = present \\ (\text{red}, stop) & \text{if } s = \text{yellow} \wedge i(tick) = present \\ (s, absent) & \text{otherwise} \end{cases}$$

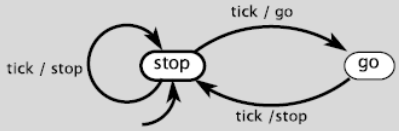
   (b) Give an execution trace of this FSM of length 4 assuming the input *tick* is *present* on each reaction.

Solution:

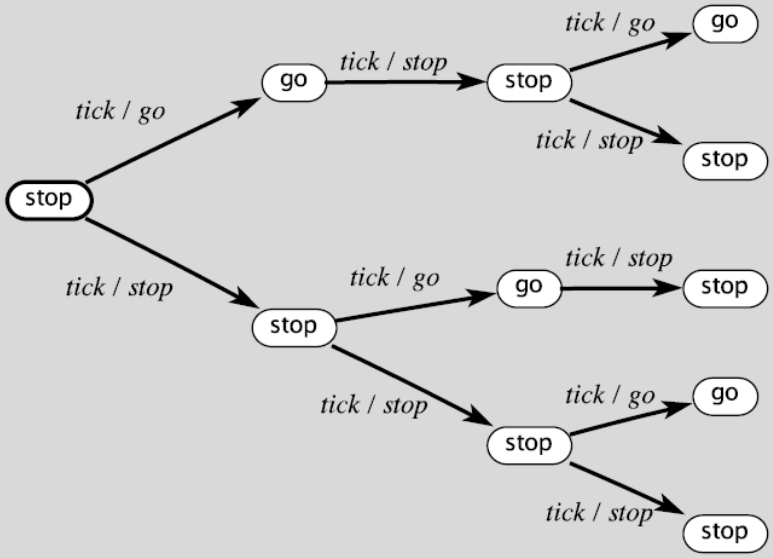
$$\text{red} \xrightarrow{tick/go} \text{green} \xrightarrow{tick/stop} \text{yellow} \xrightarrow{tick/stop} \text{red} \xrightarrow{tick/go} \cdots$$

   (c) Now consider merging the red and yellow states into a single stop state. Transitions that pointed into or out of those states are now directed into or out of the new stop state. Other transitions and the inputs and outputs stay the same. The new stop state is the new initial state. Is the resulting state machine deterministic? Why or why not? If it is deterministic, give a prefix of the trace of length 4. If it is nondeterministic, draw the computation tree up to depth 4.

Solution: The resulting state machine is given below. It is nondeterministic because there are two distinct transitions possible from state stop on input *tick*.

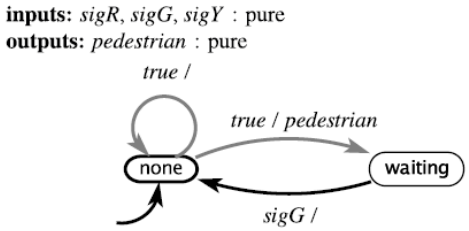


We have renamed the green state go. The computation tree for this FSM, up to depth 4, is given below:



6. This problem considers variants of the FSM in Figure 3.11, which models arrivals of pedestrians at a crosswalk. We assume that the traffic light at the crosswalk is controlled by the FSM in Figure 3.10. In all cases, assume a time triggered model, where both the pedestrian model and the traffic light model react once per second. Assume further that in each reaction, each machine sees as inputs the output produced by the other machine *in the same reaction* (this form of composition, which is called synchronous composition, is studied further in Chapter 6).

   (a) Suppose that instead of Figure 3.11, we use the following FSM to model the arrival of pedestrians:

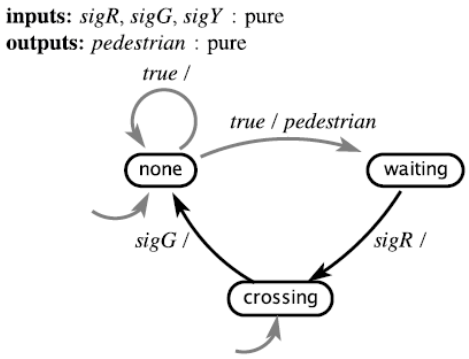


   Find a trace whereby a pedestrian arrives (the above machine transitions to waiting) but the pedestrian is never allowed to cross. That is, at no time after the pedestrian arrives is the traffic light in state red.

Solution: The traffic light begins in state red while the pedestrian model begins in state none. Suppose that the pedestrian model transitions to waiting in exactly the same reaction where the traffic light transitions to state green. The system will now perpetually remain in the same state, where the pedestrian model is in waiting and the traffic light is in state green.

Put another way, in the same reaction, we get red $\rightarrow$ green, which emits *sigG*, and none $\rightarrow$ waiting, which emits *pedestrian*. Once the composition is in state (green, none), all remaining reactions are stuttering transitions.

   (b) Suppose that instead of Figure 3.11, we use the following FSM to model the arrival of pedestrians:



   Here, the initial state is nondeterministically chosen to be one of none or crossing. Find a trace whereby a pedestrian arrives (the above machine transitions from none to waiting) but the pedestrian is never allowed to cross. That is, at no time after the pedestrian arrives is the traffic light in state red.

Solution: Suppose the initial state is chosen to be (red, none) and sometime in the first 60 reactions transitions to (red, waiting). Then eventually the composite machine will transition to (green, waiting), after which all reactions will stutter.

8. (NOTE: This exercise is rather advanced.) This exercise studies properties of discrete signals as formally defined in the sidebar on page 44. Specifically, we will show that discreteness is not a compositional property. That is, when combining two discrete behaviors in a single system, the resulting combination is not necessarily discrete.

   (a) Consider a pure signal x: $\mathbb{R} \to \{present, absent\}$ given by

   $$x(t) = \begin{cases} present & \text{if } t \text{ is a non-negative integer} \\ absent & \text{otherwise} \end{cases}$$

   for all $t \in \mathbb{R}$. Show that this signal is discrete.

   Solution: We need only to give an order-preserving one-to-one function of the form $f: T \to \mathbb{N}$, where T is defined as in the sidebar. In this case, the set of times when the signal is present is $T = \mathbb{N}$, so we can choose the identity function for f, which is trivial order preserving and one-to-one.

   (b) Consider a pure signal y: $\mathbb{R} \to \{present, absent\}$ given by

   $$y(t) = \begin{cases} present & \text{if } t = 1 - 1/n \text{ for any positive integer } n \\ absent & \text{otherwise} \end{cases}$$

   for all $t \in \mathbb{R}$. Show that this signal is discrete.

   Solution: We need only to give an order-preserving one-to-one function of the form $f: T \to \mathbb{N}$. In this case, the set of times when the signal is present is

   $$T = \{1 - 1/1, 1 - 1/2, 1 - 1/3, \cdots, 1 - 1/n, \cdots\}$$

   or

   $$T = \{0, 1/2, 2/3, \cdots\}$$

   Thus, we can define f as

   $$\forall\, t \in T, \quad f(t) = n \text{ where } t = 1 - 1/n\ .$$

   This is clearly one-to-one and order preserving. Hence, y is discrete.

   (c) Consider a signal w that is the merge of x and y in the previous two parts. That is, $w(t) = present$ if either $x(t) = present$ or $y(t) = present$, and is *absent* otherwise. Show that w is not discrete.

   Solution: Assume to the contrary that w is discrete. Then there exists an order-preserving one-to-one function $f: T \to \mathbb{N}$. Note that $1 \in T$ and that $1 - 1/n \in T$ for all $n \in \mathbb{N}$, $n > 0$. Moreover, $1 > 1 - 1/n$ for all $n \in \mathbb{N}$, $n > 0$. Since f is one-to-one and order preserving, it must be true that

   $$f(1) > f(1 - 1/n)\ .$$

   However, $1 - 1/n$ has no upper bound for $n \in \mathbb{N}$, $n > 0$, so $f(1 - 1/n)$ has no upper bound in $\mathbb{N}$, a contradiction. Hence, w must not be discrete.

   (d) Consider the example shown in Figure 3.1. Assume that each of the two signals *arrival* and *departure* is discrete. Show that this does not imply that the output *count* is a discrete signal.

Solution: Let $arrival = x$ from part (a) and $departure = y$ from part (b). Then the set T of times when *count* is present is the same as the set T in part (c). Therefore, *count* is not discrete.