CMU 15-671 Models of Software Systems Fall 1997

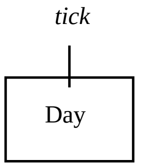
Mod7 Counter Satisfies Days

Garlan Handout 11 8 October 1995

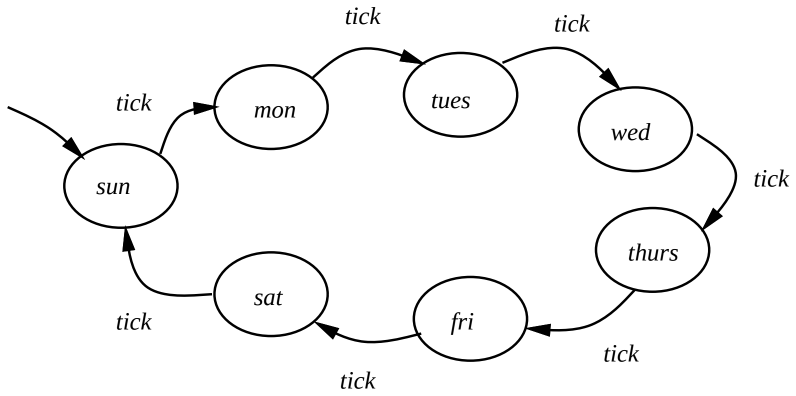
Here's a simple example to show how an "integer mod-7" counter state machine satisfies a "days of the week" machine. Both the abstract and concrete machines have a finite number of states, a finite number of transitions, and infinite traces. My proof of correctness uses an abstraction function that is one-to-one.

1 The Abstract Machine

The Day abstract machine, has one *tick* action.



Day's Interface



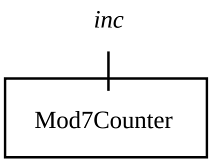
Day's State Transition Diagram

Day is like an enumerated type in Pascal or Ada; its set of states is just the days of the week:

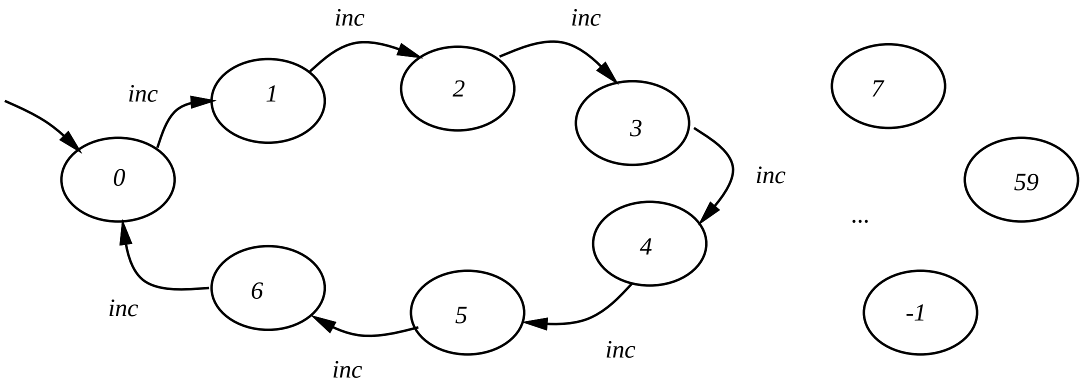
Day = (
$\{sun, mon, tues, wed, thurs, fri, sat\}$,
$\{sun\}$,
$\{tick\}$,
$\delta_A = \{(sun, tick, mon), (mon, tick, tues), (tues, tick, wed), (wed, tick, thurs),$
$(thurs, tick, fri), (fri, tick, sat), (sat, tick, sun)\}$
).

2 The Concrete Machine

Most programming languages do not support something like Day as a built-in type. But fortunately, they usually support integers. So I'm going to represent the Day machine in terms of integers modulo-7.



Mod7Counter's Interface



Mod7Counter's State Transition Diagram

Mod7Counter = (
$\{x : int\}$,
$\{0\}$,
$\{inc\}$,
$\delta_C = \{(0, inc, 1), (1, inc, 2), (2, inc, 3), (3, inc, 4), (4, inc, 5),$
$(5, inc, 6), (6, inc, 0)\}$
).

My Mod7Counter concrete machine has a single action, *inc*, which is defined in the obvious way. I've intentionally defined Mod7Counter's set of states to be the set of integers rather than just the integers between 0 and 6, inclusive. (What are the reachable states of Mod7Counter?)

3 Proof of Correctness

It should be pretty obvious that Mod7Counter machine satisfies the Day machine. But, let's go through the steps.

1. Step 1: Define an abstraction function and representation invariant.

   - First, I define an abstraction function.

     $AF : int \nrightarrow \{sun, mon, tues, wed, thurs, fri, sat\}$
     $AF(0) = sun$
     $AF(1) = mon$
     $\ldots$
     $AF(6) = sat$

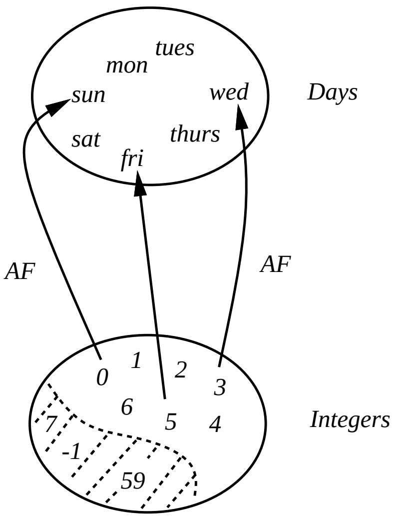
     Notice that, as promised, it's one-to-one.

   - Next, I define the representation invariant. I note that I don't need all integers to represent the days of the week. I only need seven. Moreover, the transition function, δ_C is defined for only those seven.

     $RI : int \rightarrow bool$
     $RI(i) = 0 \leq i \leq 6$

The representation invariant just says that the only integer values I have to worry about are between 0 and 6 inclusive. It characterizes the domain of AF.

The picture relating concrete to abstract states looks like:



2. Step 2: Initial conditions and commuting diagram for each action.

   - Initial conditions: I need to show that each initial state of Mod7Counter is an initial state of Day (under the abstraction function).
     - 0 is the initial state for the Mod7Counter. Thus, $AF(0)$ had better be some initial state of Day. Indeed, $AF(0) = sun$, the initial state of Day.
   - Commuting diagram: I need to show it holds for each action of Mod7Counter.

$$\begin{tikzcd} x \arrow[r, bend left, "tick"] & x' \\ y \arrow[u, "AF"] \arrow[r, bend right, "inc"'] & y' \arrow[u, "AF"'] \end{tikzcd}$$

$$\delta_A(AF(y), tick) = AF (\delta_C(y, inc))$$

In other words, I need to show[1] that $\delta_A(AF(y), tick) = AF(\delta_C(y, inc))$ for all y that satisfy the representation invariant, i.e., for all $y \in \{i : int \mid 0 \leq i \leq 6\}$. I need to show the commuting diagram for only those states that satisfy RI. The simplest proof is to do an exhaustive case analysis. y can take on only seven values so there are seven cases. I'll do the most "interesting" case ($y = 6$).

   - Case: $y = 6$
     $\delta_A(AF(6), tick)$
     $= \delta_A(sat, tick)$ def'n of AF

[1] Note that I use functional notation for the state transition relations δ_A and δ_C; more importantly, because they are both functions, it's sound to use equational reasoning in the proof.

$$
\begin{array}{ll}
= sun & \text{def'n of } \delta_A \\
= AF(0) & \text{def'n of } AF \\
= AF(\delta_C(6, inc)) & \text{def'n of } \delta_C
\end{array}
$$

When I did the last part of the proof above, I didn't really do it as shown, but rather I reduced both sides of the equation at the same time yielding $sun = sun$. I can do that because equality is bi-directional:

$$
\begin{array}{lll}
\delta_A(AF(6), tick) & = AF(\delta_C(6, inc)) & \\
\delta_A(sat, tick) & = AF(0) & \text{def'ns of } AF \text{ and } \delta_C \\
sun & = sun & \text{def'ns of } \delta_A \text{ and } AF
\end{array}
$$

I think this proof is more readable and it's perfectly acceptable. Just remember to give your justification next to each proof step if it's not obvious or clear from context.