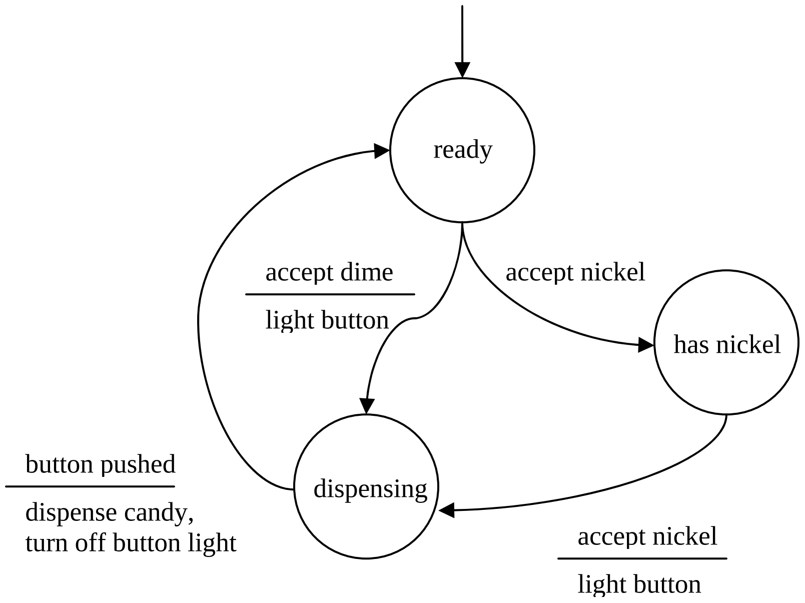
Finite State Machines – a short explanation

A simple way to model the behavior of certain kinds of artifacts or systems is by using a finite state machine. The finite state machine is intended to capture the notion that at any point in time the system is in a particular condition, or state, where it is capable of responding to a given (sub)set of stimuli or inputs. Depending on the stimulus, it may react in a particular way and also adapt itself by changing its state to one with a different set of stimuli to which it will next respond. The system is inert to a stimulus that is made in a state which does not respond to that stimulus. The model answers such questions as "What are all the distinct states of the system?", "To what stimuli will the system respond in a given state?", "What state will the system be in if it is presented with a given stimulus in a given state?", and "What sequence of inputs are needed so that the system will be in a particular state?"

Informally, a finite state machine consists of a finite set of states and a finite set of directed edges. Each state is typically given a name that indicates its meaning in terms of the entity or system which the finite state machine is intended to model. One of the states, identified as the initial state, represents the modeled system at its inception. Each edge connects a single source state to a single destination state. The source and destination states may be the same. The edges represent the transitions from one system state to another. Associated with each edge is a label that consists of two parts: a condition and an action. The condition part of the label determines under what circumstances the edge may cause a transition from the source to the destination states. The action part of the label defines any reaction or response of the system when the transition occurs. While the concept of a finite state machine can be precisely defined in mathematical terms, an informal approach is used here for simplicity.

A graphical representation of a finite state machine is shown in the figure below. This finite state machine models a simple system for dispensing candy that costs 10 cents by inserting either one dime or two nickels and then pressing a button when the button is lit up. A circle is used to represent a state. The (optional) name of the state is written inside the circle. The initial state is denoted by a unique arc that has no source state and the initial state as a destination. The state named "ready" in the finite state machine diagram below is the initial state. The directed edges (transitions) are shown as arrows that begin in a source state and end in a destination state. The labels are shown as an annotation near the arc to which it applies. The condition is written above a line that separates it from the actions which are written below the line. If there is no action then the line is omitted.

The finite state machine for the candy dispenser behaves as follows. The dispenser begins in a *ready* state. When the dispenser is ready it can accept either a dime or a nickel. If it accepts a dime the dispenser turns on the button's light and enters the *dispensing* state. Alternatively, if a nickel is accepted when the dispenser is ready the dispenser enters the *has nickel* state. The *has nickel* state indicates that another nickel must be provided. When a nickel is accepted in the *has nickel* state the dispenser turns on the button's light and enters the *dispensing* state. When the dispenser's button is pushed in the *dispensing* state the candy is provided and, after turning off the button's light, the dispenser returns to its *ready* state.



Candy dispenser finite state machine

The candy dispenser finite state machine expresses a number of salient features about the possible actions of the dispenser. For example:

- Pushing the button when the button's light is not on has no effect.
- Candy will not be dispensed until 10 cents has been accepted.
- Pennies cannot be provided as payment.
- Exactly two nickel or exactly one dime will be accepted for payment.
- Once the correct amount has been entered the dispenser must dispense the candy.
- The dispenser never runs out of candy.

This last property is particularly appealing for a candy dispenser.

Terminology

The above description has used two terms, state and behavior, that are important. These notions occur in many different contexts in computer science because they capture very basic ideas. The two ideas are also closely connected to each other.

State captures two important aspects of a system's behavior:

- Reactivity: in a given condition a system is only capable of performing certain actions. These actions are appropriate and possible given the condition of the system. This occurs in many real-world systems and artifacts. A door cannot be opened when it is locked. An automatic transmission car cannot be started unless the shifter is in park. When an elevator is travelling between floors the doors cannot be opened. An ATM machine will not dispense money without a proper authorization. In the candy dispenser example, the pushing of the button has no effect in the *ready* or *has nickel* states.
- History: a system's current behavior is conditioned on what has happened in the past. In this sense, a system "remembers" past actions. For example, the *has nickel* state in the candy dispenser reflects that fact that one nickel has already been accepted by the dispenser. This ability is important so that a system can sustain an interaction over multiple steps (such as entering two nickels, one after the other).

To correctly model a system it is important to know all of the possible states of a system.

A system's behavior is defined by all of the possible sequences of actions that the system can take over time. Suppose that *dime, nickel, light, press*, and *dispense* are used to denote the actions of accepting a dime, accepting a nickel, turning on the button's light, pressing the button, and dispensing the candy, respectively. Some possible behaviors of the candy dispenser are:

dime, light, press, dispense
nickel, nickel, light, press dispense
dime, light, press, dispense, dime, light, press, dispense
dime, light, press, dispense, nickel, nickel, light, press, dispense
…

There are, in fact, an infinite number of behaviors of this simple system. Fortunately, the behavior has strong patterns so it is not too difficult to understand. But the fact that such a simple machine can have an infinite behavior should raise cautions about more realistic systems that have much more complex behaviors. Perhaps this occasions some sense of why building a very complex software system is challenging.

Exercises

1. Modify the candy dispenser so that it will give back a nickel if a nickel and a dime are provided.
2. Modify the candy dispenser so that there is a choice between two different candies both of which cost 10 cents.
3. Modify the candy dispenser so that there is a choice between two different candies, one of which costs 5 cents and the other costs 10 cents. Be sure to provide change if a dime is given and a 5 cent candy is chosen.

4. Modify the candy dispenser so that there is a refund button which gives back the payment if the candy has not already been dispensed. The refund button also has a light that shows when it pushing it has some effect.