State Machines

This lecture describes how state machines are designed and specified.
After this lecture you should be able to design a state machine from a description of its behaviour. You should be able to describe the state machine using state transition diagrams and state transition tables.

State Machines

A state machine is an abstract way of describing the behaviour of a controller. The state machine description reduces the description of any controller to four things: inputs, outputs, states, and state transition rules.

The state machine description contains no details about the implementation of the controller (no flowcharts, programs, schematics, mechanical diagrams, etc.). Instead, a state machine only describes the different states that the controller can be in, the outputs in each state, and what input conditions will cause transitions between states.

A state machine description can be used to describe and analyze existing systems and as an aid in the design of new systems. The simplified description represented by a state machine allows us to verify proper operation and document the logical operation of a controller before proceeding with detailed design (e.g. writing a program or designing hardware).

Design of State Machines

There are four steps to designing a state machine:

- define the outputs – these are the signals that control those things that the controller needs to control
- define the inputs – the signals that provide the controller with information about things that it needs to monitor in order to operate properly
- define all of the possible states that the controller will need to be in and what values the outputs will take for each of these states
- define the exact input conditions that will cause the state machine to switch from one state to another

Inputs and Outputs

A controller's inputs are typically signals from sensors such as switches, thermometers, pressure gauges, and so on. The outputs are signals that drive actuators such as motors or valves, displays, etc.

We will restrict ourselves to inputs and outputs that are binary values (represented as 0/1, false/true, or low/high).

To simplify the state machine description, each input and output is given an abbreviation, usually mnemonic. For example an input that indicates that a tank is full might be given the label `full`, an output that turns a drill motor on might be given the label `drill`.

States

A state machines (also called a *Finite State Machine* or FSM) is a controller that can be in a finite number of different conditions. Each of these conditions is called a state.

For example, a simple controller for a motor may have two states: motor turned on and motor turned off. A more complicated controller may have more states. For example, a controller for an overhead projector cooling fan motor (which must continue to run until it the bulb has cooled down, even if the projector is turned off) may have three states: motor running, motor cooling down, and motor off.

The number of states required to describe the state machine and the definitions of each of these states will be determined by the application.

Each state is given a mnemonic symbol (`ON`, `COOLING`, `OFF`, etc).

In this course we will only consider state machines whose output is a function of the current state[1]. Because of this *there must be at least one state for each*

[1] It is also possible to have state machines whose output is a function of the inputs as well as the state

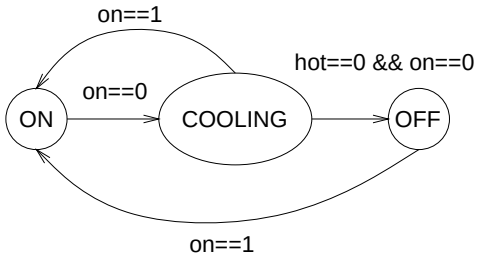
desired **combination** *of output values.* In some cases there may actually be more states than there are output conditions. For example, the second motor controller described above might have only one binary output (to control the motor) but might have three states (`ON`, `OFF`, and `COOLING`).

State Transition Diagram

The rules describing the transitions between states are commonly given as a diagram called a state transition diagram (or simply "state diagram").

In the state diagram each state is represented by a circle. Each circle is labeled with its state name. Lines are drawn to show the possible transitions between the states and each such line is labelled with a logical expression describing the input conditions required for that transition to happen. If none of the transition conditions is met, the controller remains in its current state.

For example, the 3-state motor controller described above might have a switch input (`on`) and an input from a temperature sensor (`hot`). The state transition diagram might look as follows:



There are many different conventions for drawing state diagrams. For example, it's common to also show the values of the outputs on the state diagram. In the examples in this course we will describe the output values for each state separately.

Tabular Representation

The operation of a state machine can also be described by using two tables: one to list the output values for each state and one to give the input conditions which cause the state machine to switch from one state to another. The first table has one row for each state. Each column gives the value of the outputs for that state. For example, the motor controller has only one output, say `run`, and the table might look like:

state	`run`
ON	1
COOLING	1
OFF	0

The second table has one row for each possible combination of current state and input conditions. It has three columns: the current state, the input conditions and the state to switch to if that particular combination of state and input conditions happens.

For example, for the motor controller described above, the tabular form of the state transition diagram shown above is:

current state	input conditions		next state
	`on`	`hot`	
ON	0	0	COOLING
ON	0	1	COOLING
ON	1	0	ON
ON	1	1	ON
COOLING	0	0	OFF
COOLING	0	1	COOLING
COOLING	1	0	ON
COOLING	1	1	ON
OFF	0	0	OFF
OFF	0	1	OFF
OFF	1	0	ON
OFF	1	1	ON

Exercise:

A simple burglar alarm has two inputs: an on/off switch (`arm`) and an open-window detector (`open`). It has one output that sounds a siren (`siren`). If the alarm is armed and the window is opened then the siren is turned on. The siren remains on until the alarm is shut off even if the window is closed again.

List the inputs and outputs. How many unique combinations of outputs are there? How many states will be required? Give a table showing the outputs for each state. Draw the state transition diagram. Give the state transition table in the format described above.

Deadlock and Unreachable States

The state machine diagram allows us to easily detect certain errors. One of these is a state that can never be reached. This indicates an incomplete specification. Another problem is a state that can be reached but can't be exited. This is called deadlock and is usually undesired.