1 Introduction

An automaton is a construct of states and inputs where each state and input together lead to a possibly new state. Applying this to a machine its working is described press the form of states, and reading of inputs can change the state.

In the simplest case the machine controller does nothing except when inputs are happening.

As an example we observe a lamp, that can be switched with a tactile pushbutton, and the 'controller machine' transforms the function into that of a latching button.

The input events we can deduct are:

press pressing the button against its spring pressure to have a contact

release at releasing the button its rest state state will be non-contact

The stable states we can see are:

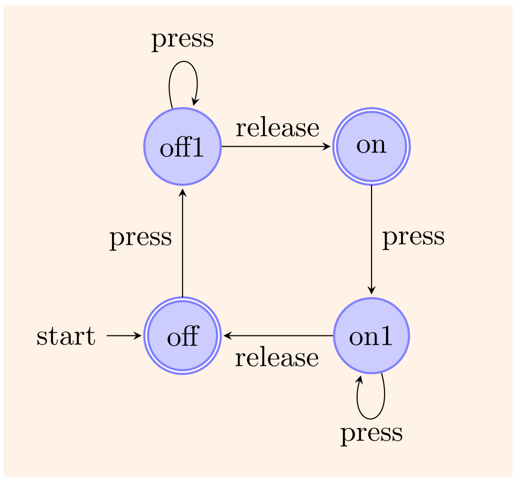
off the lamp is off, choosen here as the starting state

on the lamp is on

We would like the lamp to have a 'natural' response, meaning switching on at the press and switching off at the release of the button. Furthermore we want to suppress some milliseconds of contact bounce of the mechanical part. Therefore the addition of 2 passing states can help:

off1 the lamp was off and the button pressed

on1 the lamp was on and the button pressed



The example can be programmed into a 4×2 decision matrix with rows representing the states and columns representing the input events. The cells hold the new state, possibly prefixed with a comma and before that press italics the action to be done for the lamp ouput, or a minus sign if nothing changes with the combination of state and input:

	press	release
off	*switch on*, off1	-
off1	-	on
on	on1	-
on1	-	*switch off*, off

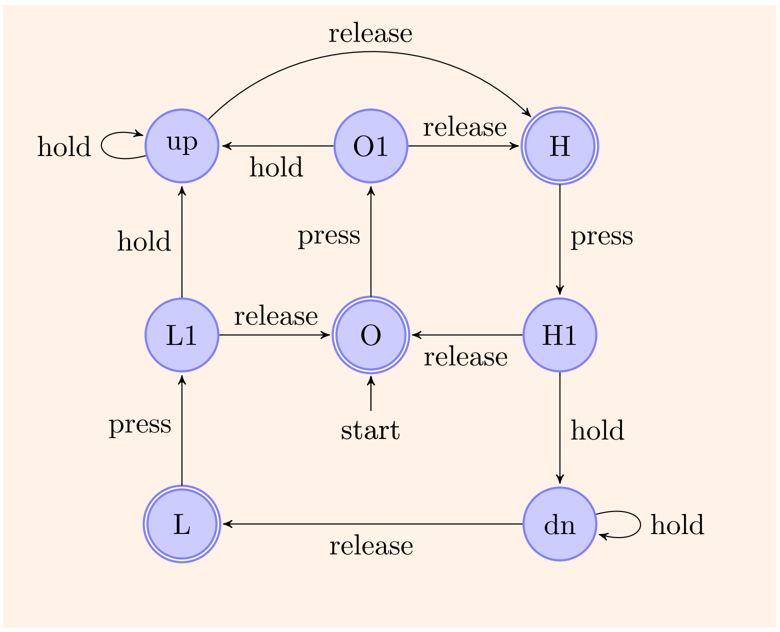
2 dimming automaton

Just as with the introduction we would like a lamp to immediately switch on at button press and only switch off at the button release.

The state diagram from the introduction is expanded with holding the tactile push button, leading to dimming up or down press lamp intensity. In order to acoomodate these extras, a new input (column) **hold** is added press the table, repeating at regular intervals. Two additional states, **up** and **dn**, are added where the lamp intensity is adjusted.

A memory function is added to hold the intensity before switching off and to restore this value at switching on, so for example a night stand lamp will switch on at the chosen dimmed intensity.

We discover that 2 stable on-positions are needed for that. Renaming on and on1 to H an H1 for high intensity we can add L and L1 for low intensity and press after the stable state, so we can also switch the lamp off with a short press from L. Renaming off and off1 to O and O1 leads to an elegant-looking diagram with diagonal positions for stable states and dimming, and transition states in between:



The corresponding matrix with possible implementation functions can be:

	release	press	hold
O	-	f1()	-
O1	h()	-	f2()
up	h()	-	f2()
H	-	h1()	-
H1	f4()	-	f3()
dn	l()	-	f3()
L	-	l1()	-
L1	f4()	-	f2()

```
f1() {return lamp=saved, O1;}
f2() {if(++lamp>max) lamp=max; return up;}
f3() {if(--lamp<min) lamp=min; return dn;}
f4() {return saved=lamp,lamp=0, O;}
 h() {return H;}
h1() {return H1;}
 l() {return L;}
l1() {return L1;}
```

In a real-world project I could optimize the above as well as add more functionality.