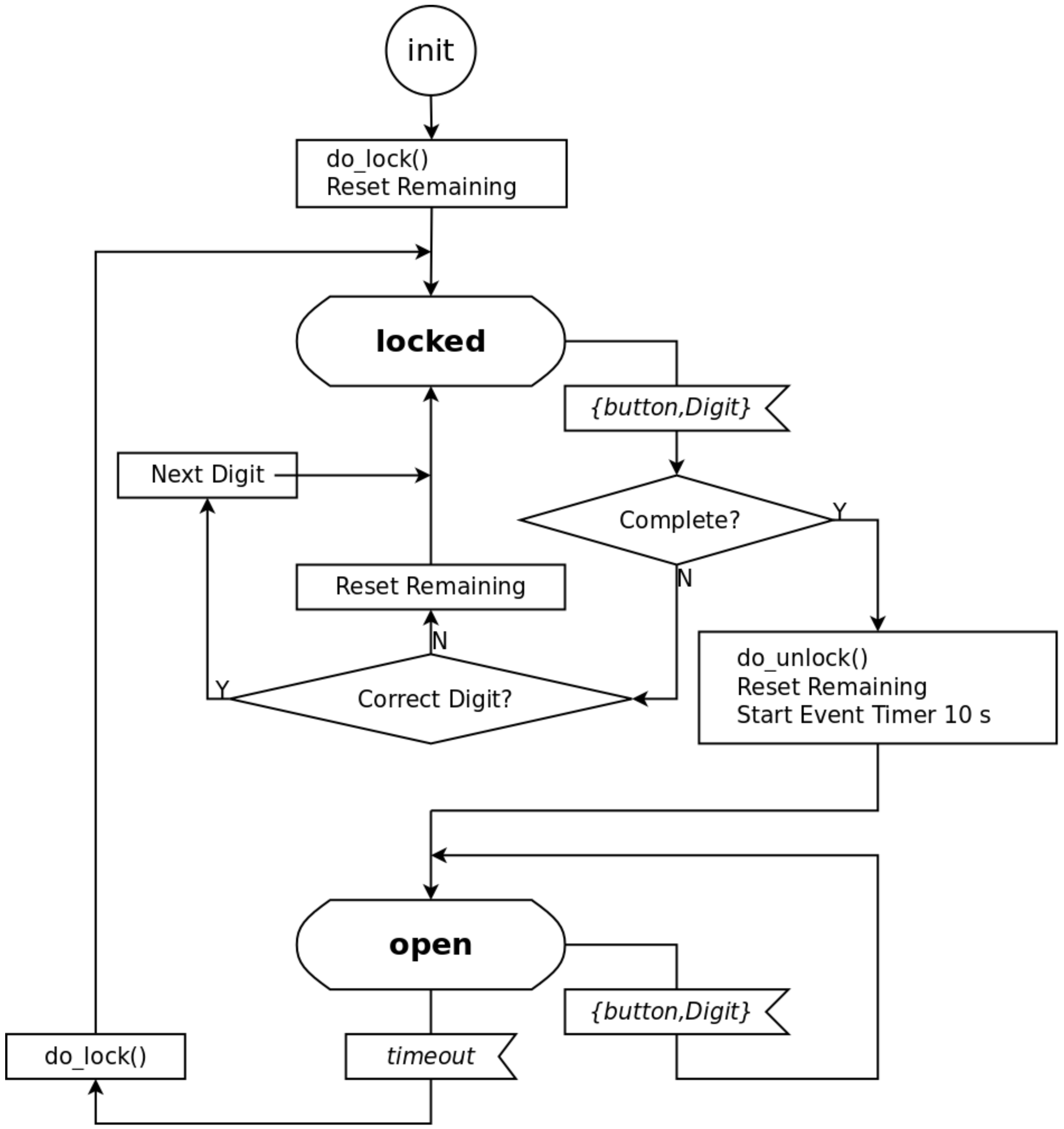
gen_state_machine

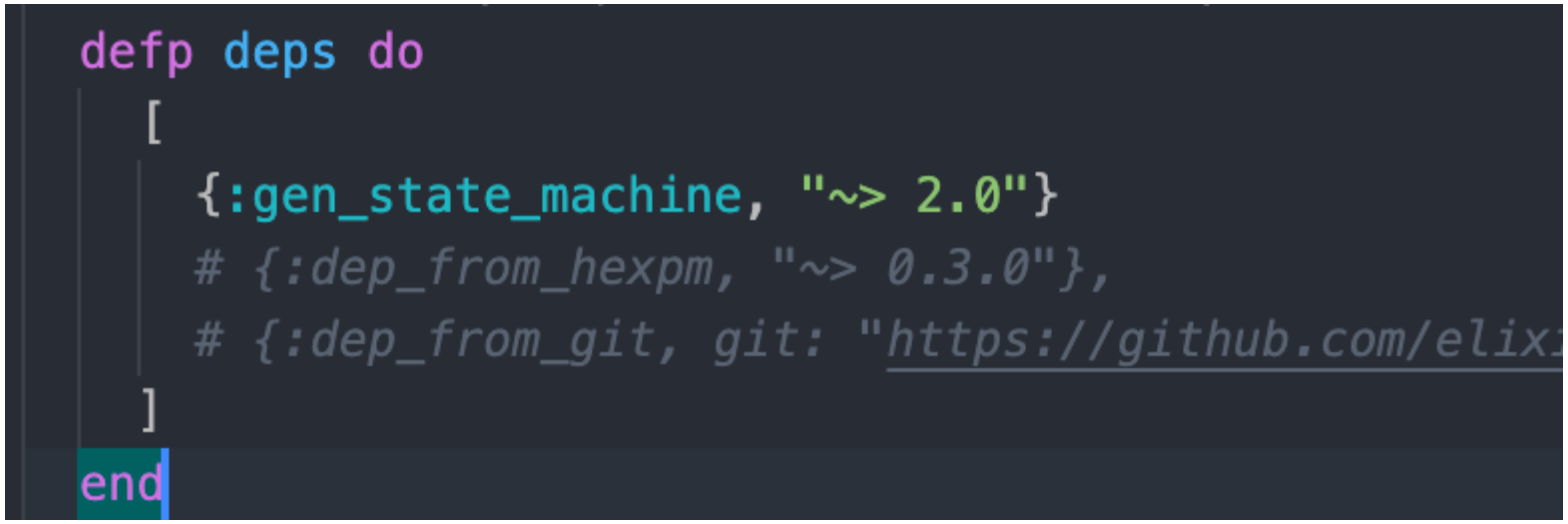
The elixir wrapper around gen_statem

Inspired from https://www.smoothterminal.com/articles/genstatemachine

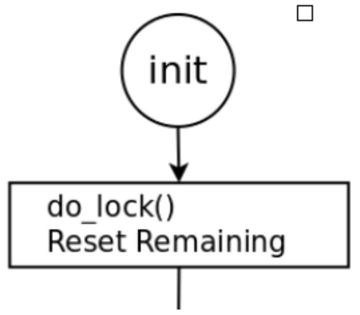
What is a state machine?

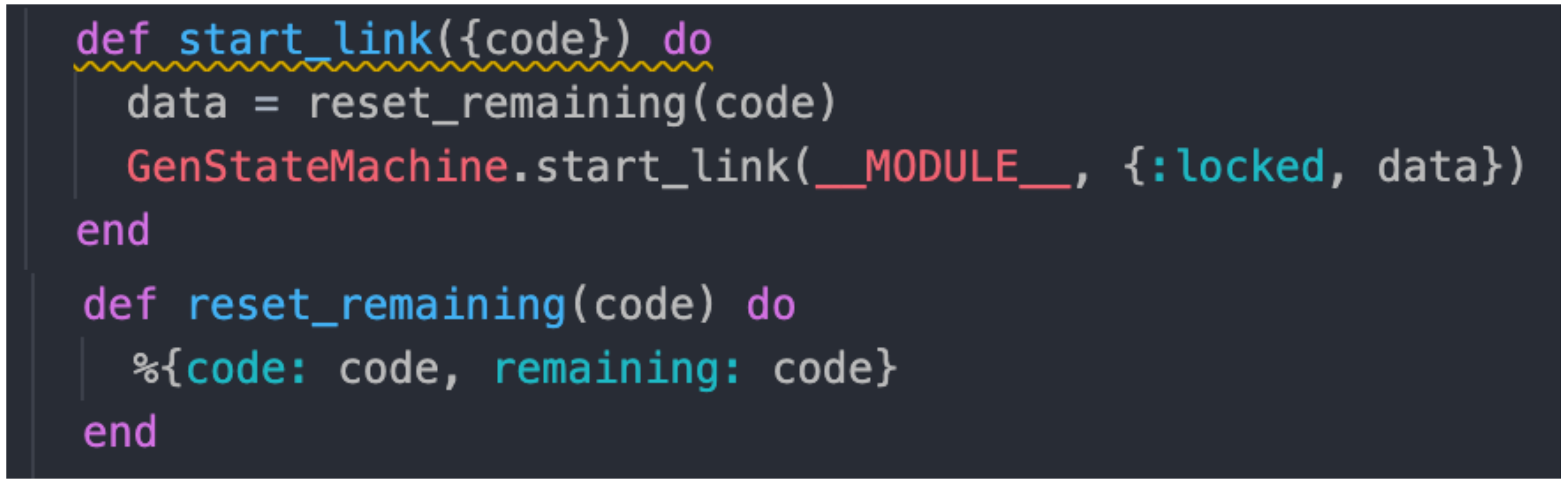


How to build a simple door

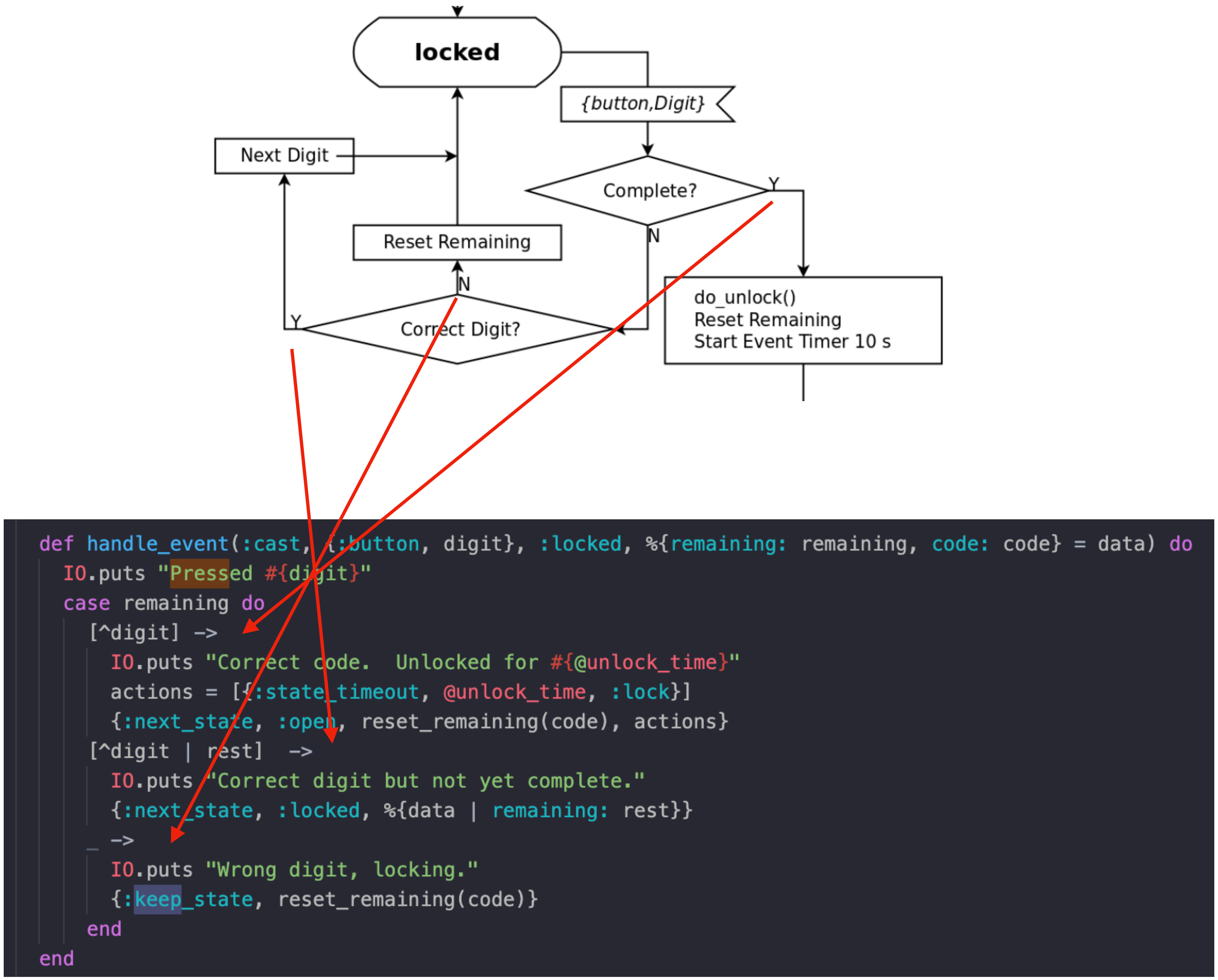


```
defp deps do
  [
    {:gen_state_machine, "~> 2.0"}
    # {:dep_from_hexpm, "~> 0.3.0"},
    # {:dep_from_git, git: "https://github.com/elix
  ]
end
```





```
def start_link({code}) do
  data = reset_remaining(code)
  GenStateMachine.start_link(__MODULE__, {:locked, data})
end
 def reset_remaining(code) do
   %{code: code, remaining: code}
 end
```

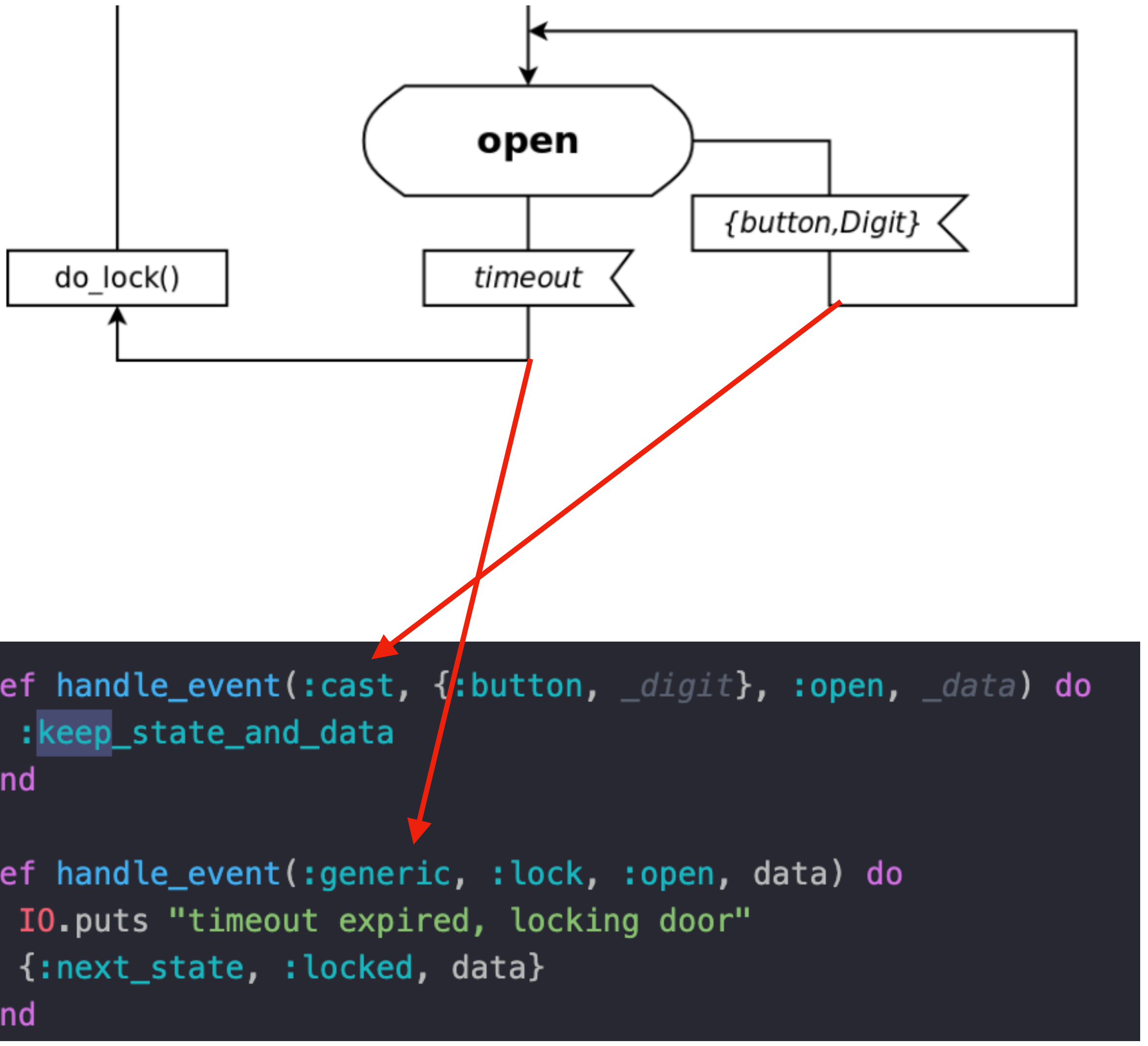


```
def handle_event(:cast, {:button, digit}, :locked, %{remaining: remaining, code: code} = data) do
  IO.puts "Pressed #{digit}"
  case remaining do
    [^digit] ->
      IO.puts "Correct code.  Unlocked for #{@unlock_time}"
      actions = [{:state_timeout, @unlock_time, :lock}]
      {:next_state, :open, reset_remaining(code), actions}
    [^digit | rest]  ->
      IO.puts "Correct digit but not yet complete."
      {:next_state, :locked, %{data | remaining: rest}}
    _ ->
      IO.puts "Wrong digit, locking."
      {:keep_state, reset_remaining(code)}
  end
end
```

A bit about timeouts

```
actions = [{:generic, @unlock_time, :lock}]
{:next_state, :open, reset_remaining(code), actions}
```

Time-Out	Cancellation	Cancelled When...
Event	Automatic	Any event handled
State	Automatic/Manual	Reset to `:infinity` or state changes
Generic	Manual	Reset to `:infinity`

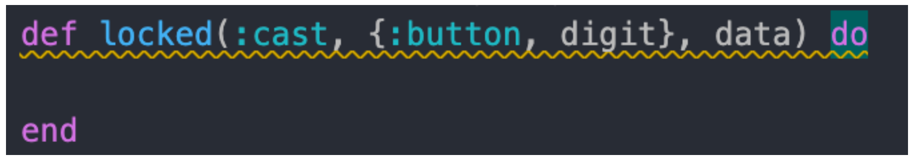


```
def handle_event(:cast, {:button, _digit}, :open, _data) do
  :keep_state_and_data
end

def handle_event(:generic, :lock, :open, data) do
  IO.puts "timeout expired, locking door"
  {:next_state, :locked, data}
end
```

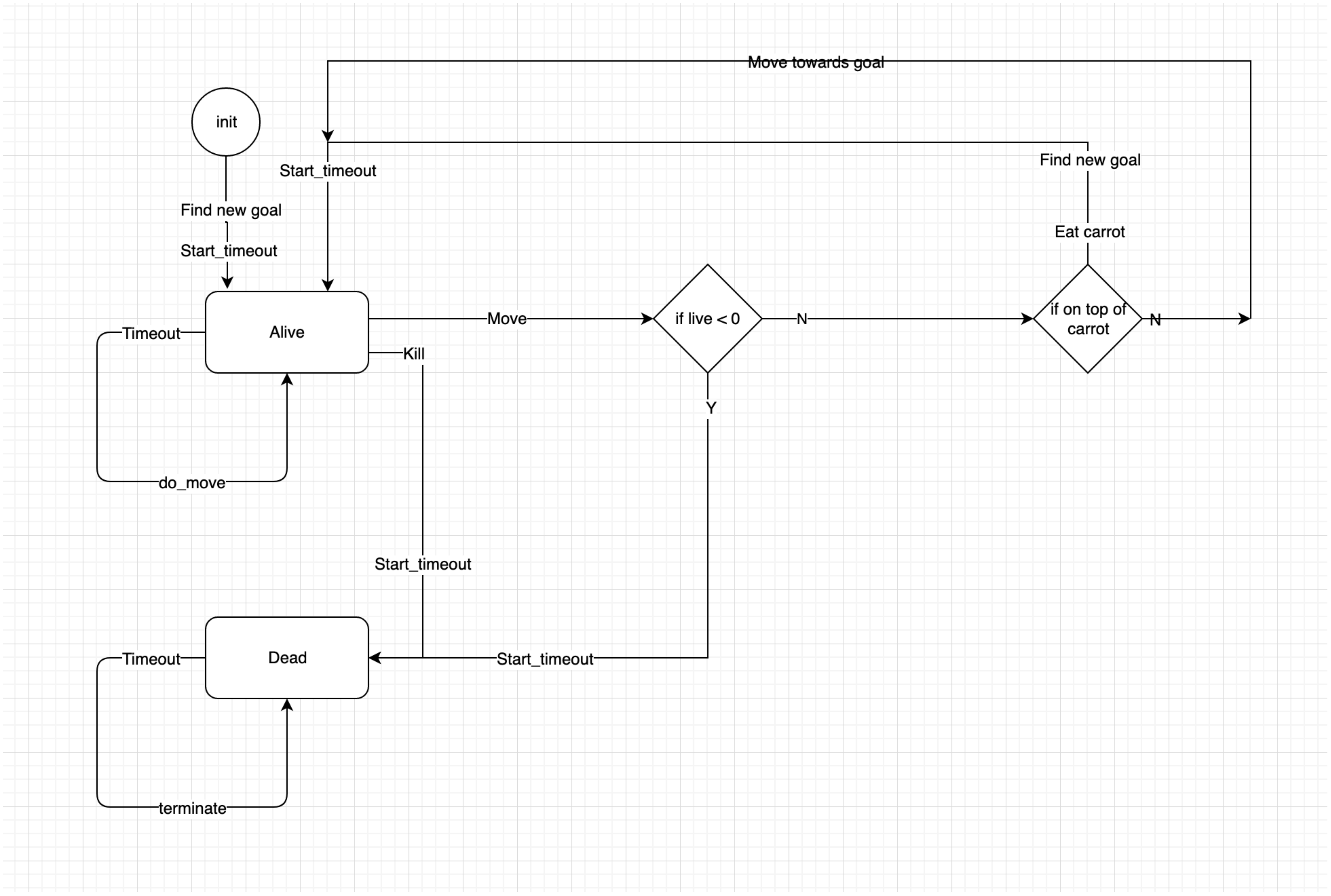
Lets try it out

Advances features

- callback_mode
  - :state_functions
    - 
      ```
      def locked(:cast, {:button, digit}, data) do

      end
      ```
    - For simple state can make it more simple
  - :handle_event_function
    - Default implementation
    - Can handle complex state like {:open, :keep_open}
- State enter functions
  - Makes it possible to perform actions when entering a state

Something a bit more interesting



init
Find new goal
Start_timeout
Start_timeout
Move towards goal
Find new goal
Eat carrot
Alive
Timeout
do_move
Move
Kill
if live < 0
N
Y
if on top of carrot
N
Start_timeout
Dead
Start_timeout
Timeout
terminate