FINITE STATE MACHINES: A DESIGN PATTERN FOR FPGAS AND REACT

Tessa Bradbury

LINUX.CONF.AU
21-25 January 2019
Christchurch, NZ

WHY THIS TOPIC?

REACT?

REACT?

```
Greeting = ({ name }) => (
  <p className="awesomest-font">
    Hello {name}! Welcome to LCA!
  </p>
)
```

REACT?

```
Greeting = ({ name }) => (
  <p className="awesomest-font">
    Hello {name}! Welcome to LCA!
  </p>
)
```

```
<Greeting name={currentName}/>
```

REACT STATE

```
class Greeting extends React.Component {
  state = { name: '' }

  updateName = event => {
    this.setState({ name: event.target.value })
  }

  render() (
    <div>
      <label for="name">Name</label>
      <input id="name" onChange={this.updateName}/>
      <Greeting name={this.state.name}/>
    </div>
  )
}
```

FPGA?

FPGA?

Field Programmable Gate Array

FPGA?

Field Programmable Gate Array

```
always @(posedge sysclk) begin
  red_led <= ~red_led
end
```

FPGA?

Field Programmable Gate Array

```
always @(posedge sysclk) begin
  red_led <= ~red_led
end
```

```
# sysclk=sysclk
NET "sysclk" LOC = "C9";
# led_5=red_led
NET "red_led" LOC = "D11" | IOSTANDARD = LVTTL | SLEW = SLOW |
```

FINITE STATE MACHINE?

FINITE STATE MACHINE?

1. A finite set of states

FINITE STATE MACHINE?

1. A finite set of states
2. An initial state

FINITE STATE MACHINE?

1. A finite set of states
2. An initial state
3. A transition function
   - (state, event) => state

FINITE STATE MACHINE?

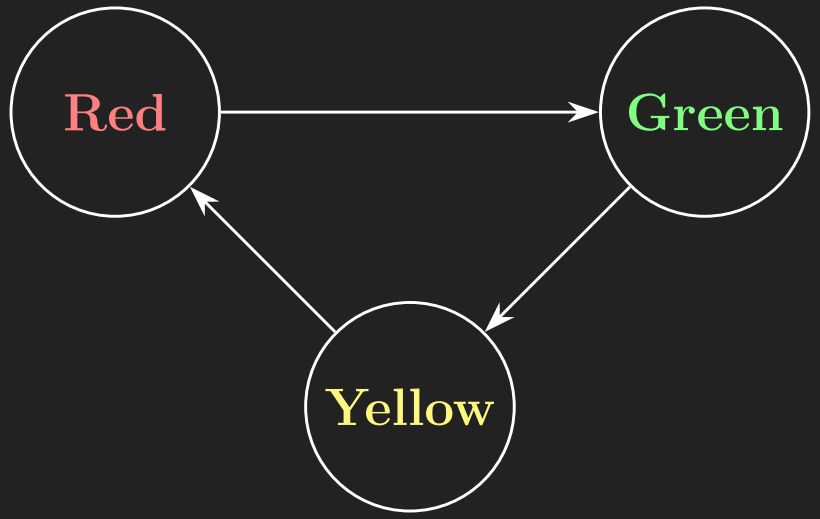
1. A finite set of states
2. An initial state
3. A transition function
   - (state, event) => state
4. A mapping from state to outputs*

FINITE STATE MACHINE?

1. A finite set of states
2. An initial state
3. A transition function
   - (state, event) => state
4. A mapping from state to outputs*

   * Not part of official definition

TRAFFIC LIGHTS


Red
Green
Yellow

WHAT ELSE?

WHAT ELSE?

- TCP

WHAT ELSE?

- TCP
- Bug tracker

WHAT ELSE?

- TCP
- Bug tracker
- Registration wizard

WHAT ELSE?

- TCP
- Bug tracker
- Registration wizard
- Games

WHAT ELSE?

- TCP
- Bug tracker
- Registration wizard
- Games
- …

WHAT DO THESE HAVE IN COMMON?

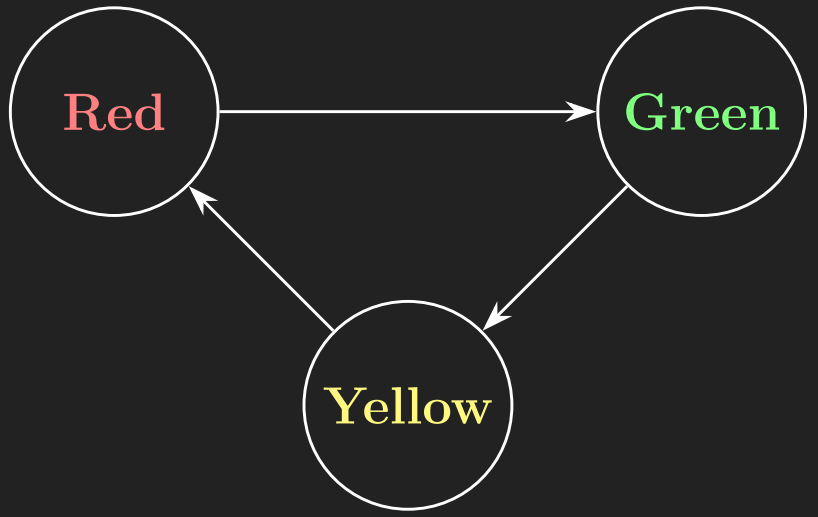
- States make sense
- Different behaviour in different states
- Clear transitions between states

If FSMs are everywhere, why are they so central to React and FPGAs?

If FSMs are everywhere, why are they so central to React and FPGAs?

Event orientated systems

TRAFFIC LIGHTS — DEEP DIVE



ASIDE: WHAT'S THE EVENT?

ASIDE: WHAT'S THE EVENT?

- Some sort of timer

ASIDE: WHAT'S THE EVENT?

- Some sort of timer
- Platform dependent

ASIDE: WHAT'S THE EVENT?

- Some sort of timer
- Platform dependent
- Want to abstract that away

ASIDE: WHAT'S THE EVENT?

- Some sort of timer
- Platform dependent
- Want to abstract that away

Focus on the transition function

REACT

S

REACT

```
updateLights = event => {
  this.setState(state => {
    if (state === RED) {
      return GREEN
    } else if (state === GREEN) {
      return YELLOW
    } else if (state === YELLOW) {
      return RED
    } else {
      return state
    }
  }
}
```

S

VERILOG

```
always @(*) begin
  if (update_lights)
    case(state)
      `RED:
        next_state = `GREEN;
      `GREEN:
        next_state = `YELLOW;
      `YELLOW:
        next_state = `RED;
    endcase
  else
    next_state = state;
end
```

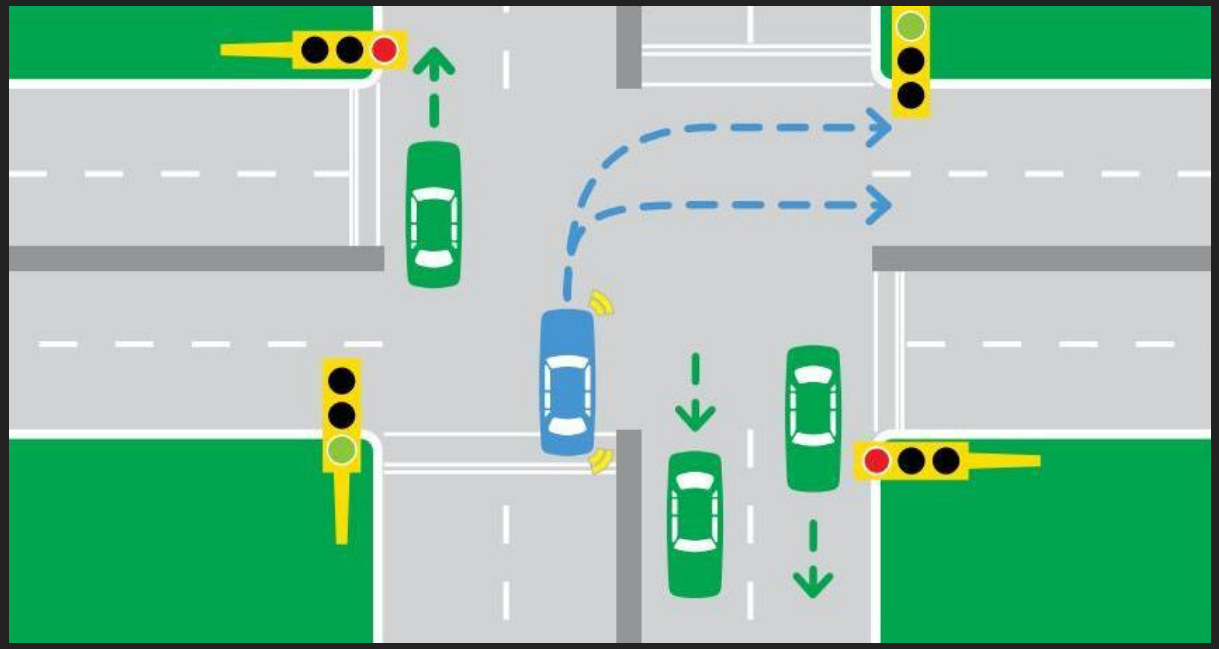
```
always @(posedge sysclk) begin
  state <= next_state
end
```

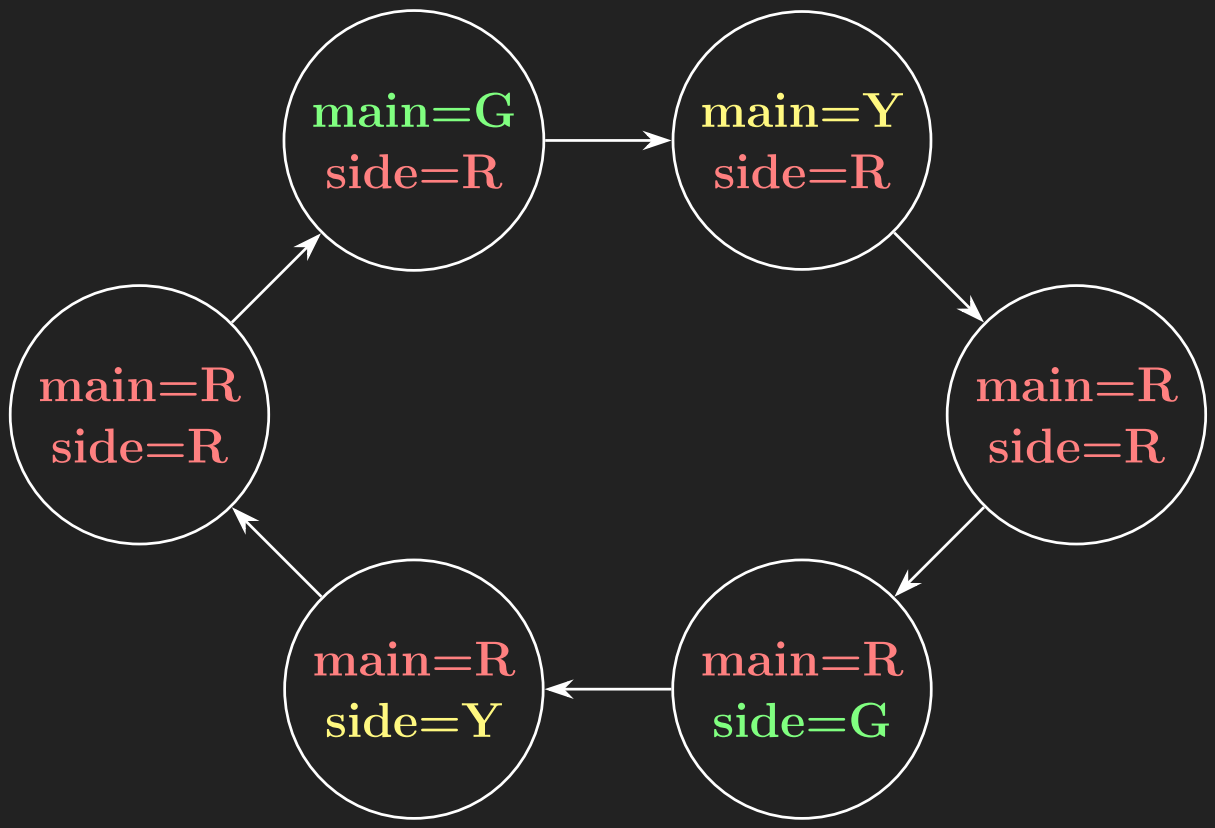
Outputs

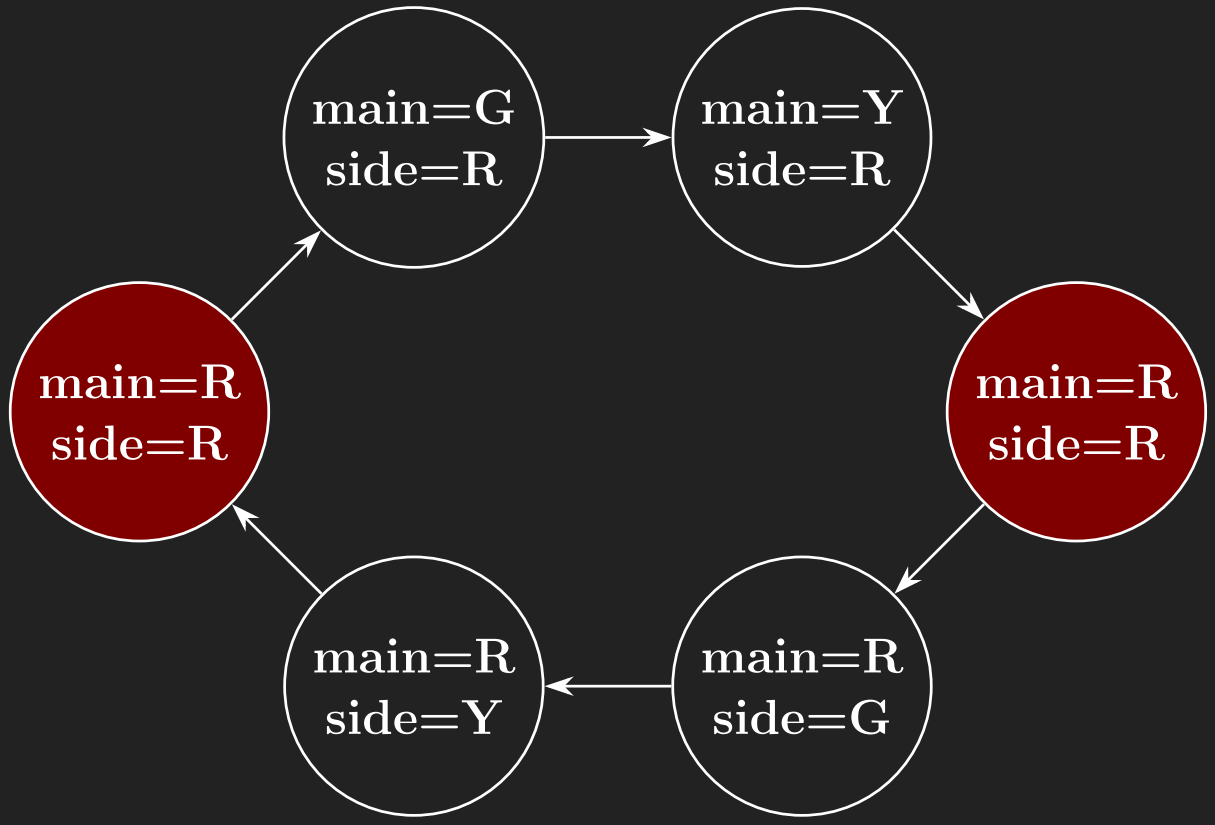
```
TrafficLight = function(state) {
  return (
    <div>
      <Light colour="red"    lit={state == RED}/>
      <Light colour="yellow" lit={state == YELLOW}/>
      <Light colour="green"  lit={state == GREEN}/>
    </div>
  )
}
```

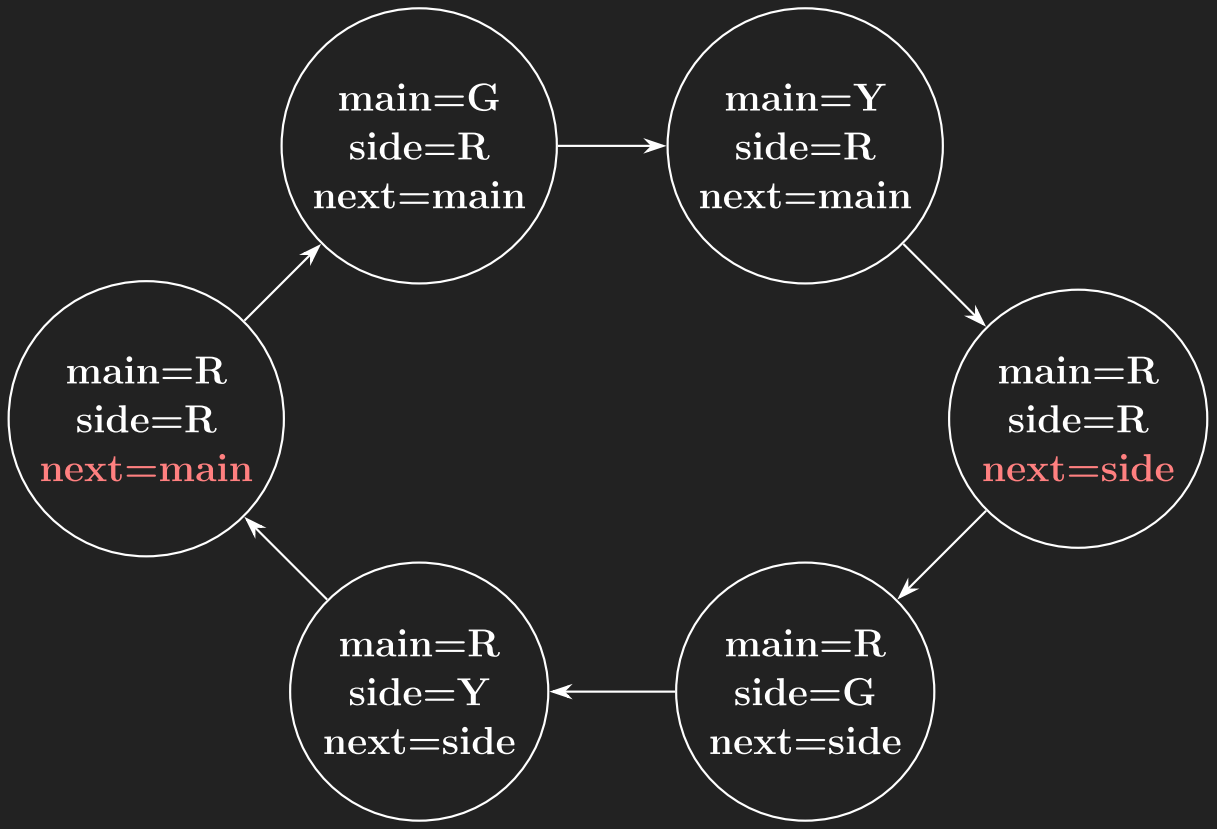
```
assign main_red_light    = state == `RED;
assign main_yellow_light = state == `YELLOW;
assign main_green_light  = state == `GREEN;
```

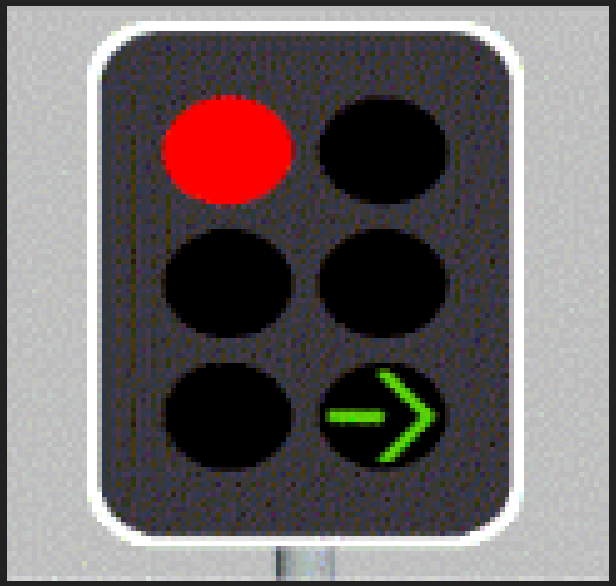
INTERSECTIONS

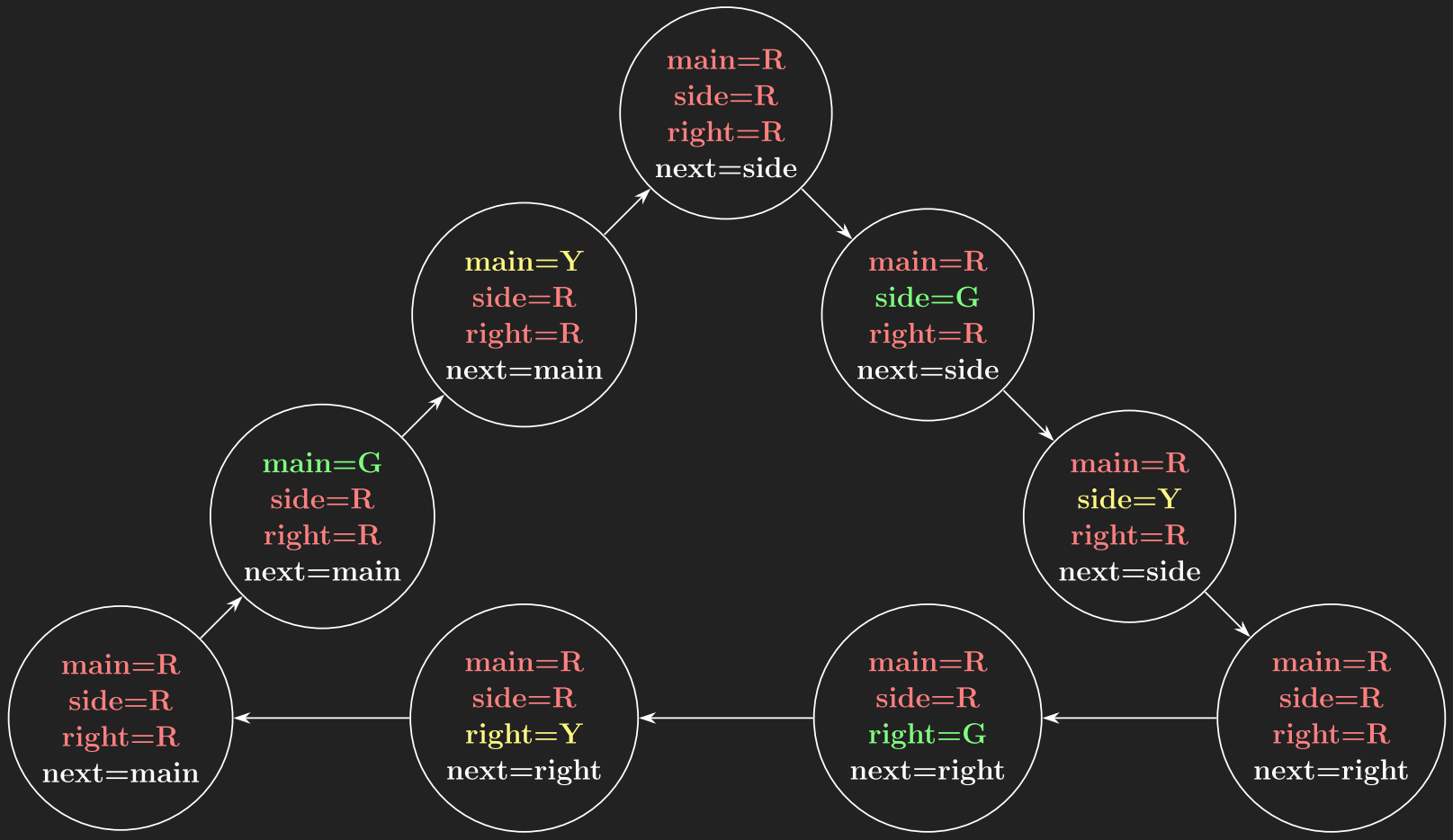



main=G
side=R
main=Y
side=R
main=R
side=R
main=R
side=G
main=R
side=Y
main=R
side=R


main=G
side=R
main=Y
side=R
main=R
side=R
main=R
side=R
main=R
side=Y
main=R
side=G


main=G
side=R
next=main
main=Y
side=R
next=main
main=R
side=R
next=side
main=R
side=G
next=side
main=R
side=Y
next=side
main=R
side=R
next=main



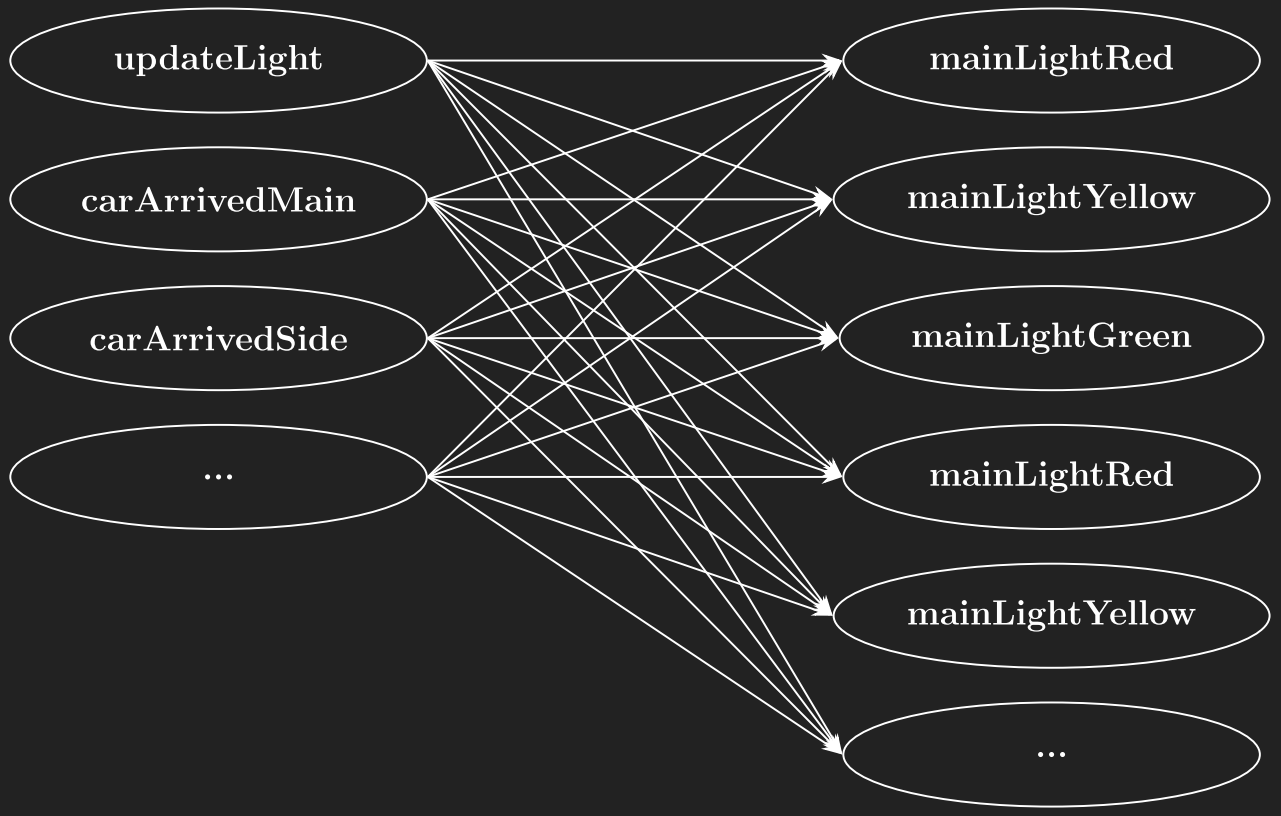

main=R
side=R
right=R
next=side
main=R
side=G
right=R
next=side
main=R
side=Y
right=R
next=side
main=R
side=R
right=R
next=right
main=R
side=R
right=G
next=right
main=R
side=R
right=Y
next=right
main=R
side=R
right=R
next=main
main=G
side=R
right=R
next=main
main=Y
side=R
right=R
next=main

SO WHAT DOES THIS GIVE US?

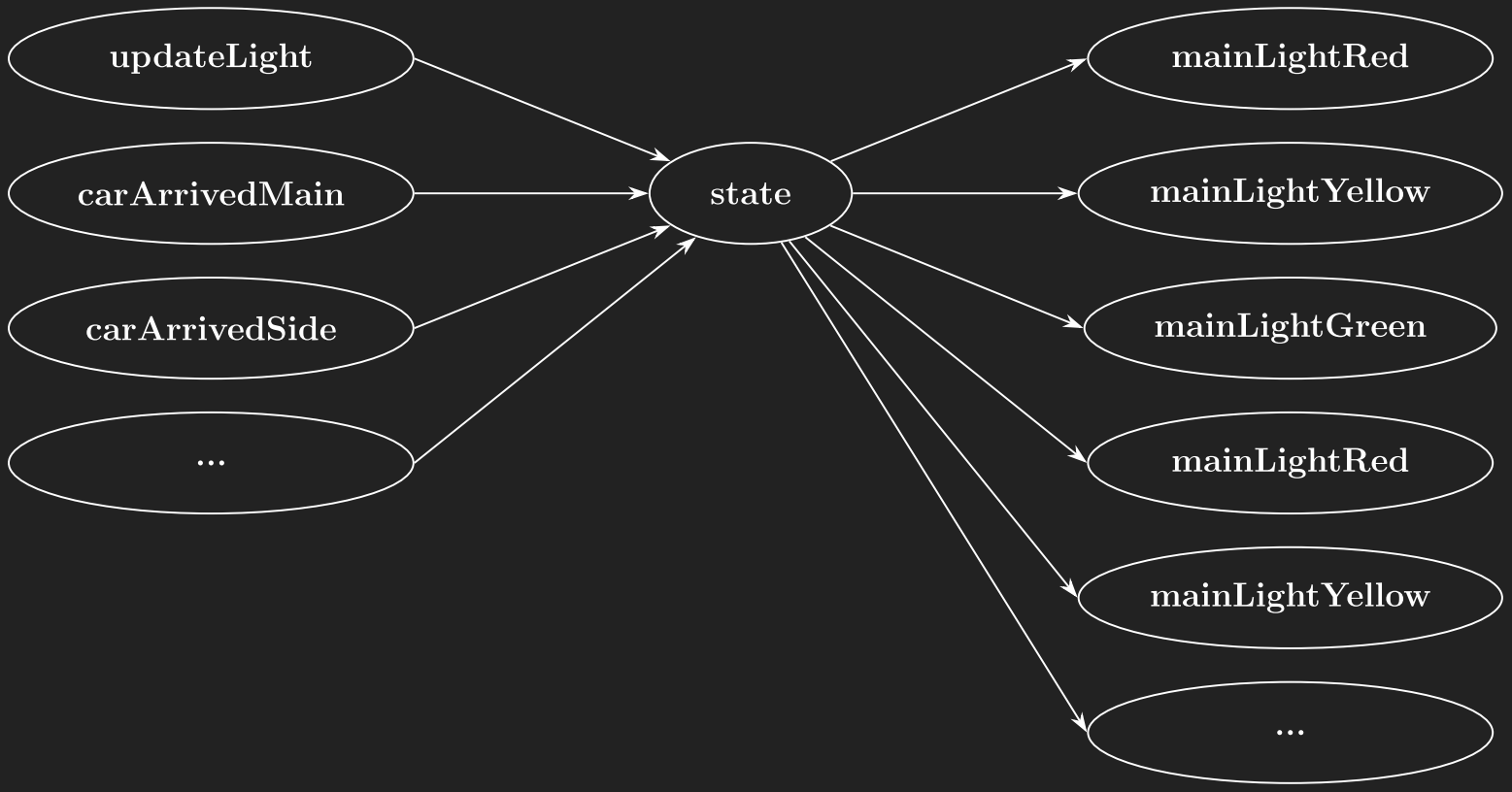
MOAR FEATURES!

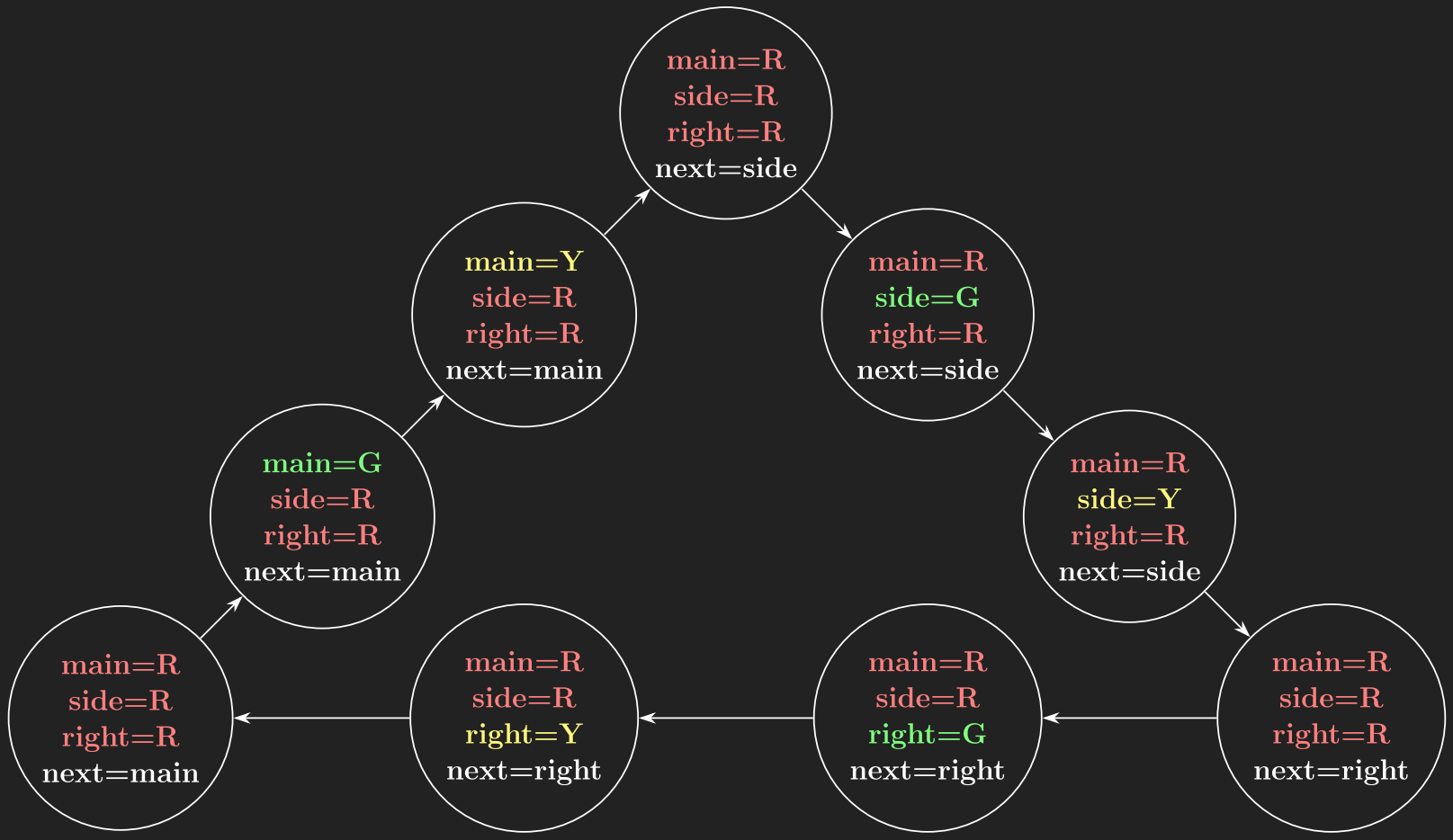
- Pedestrian lights
- Car arrival detection
- Different directions combinations
- …

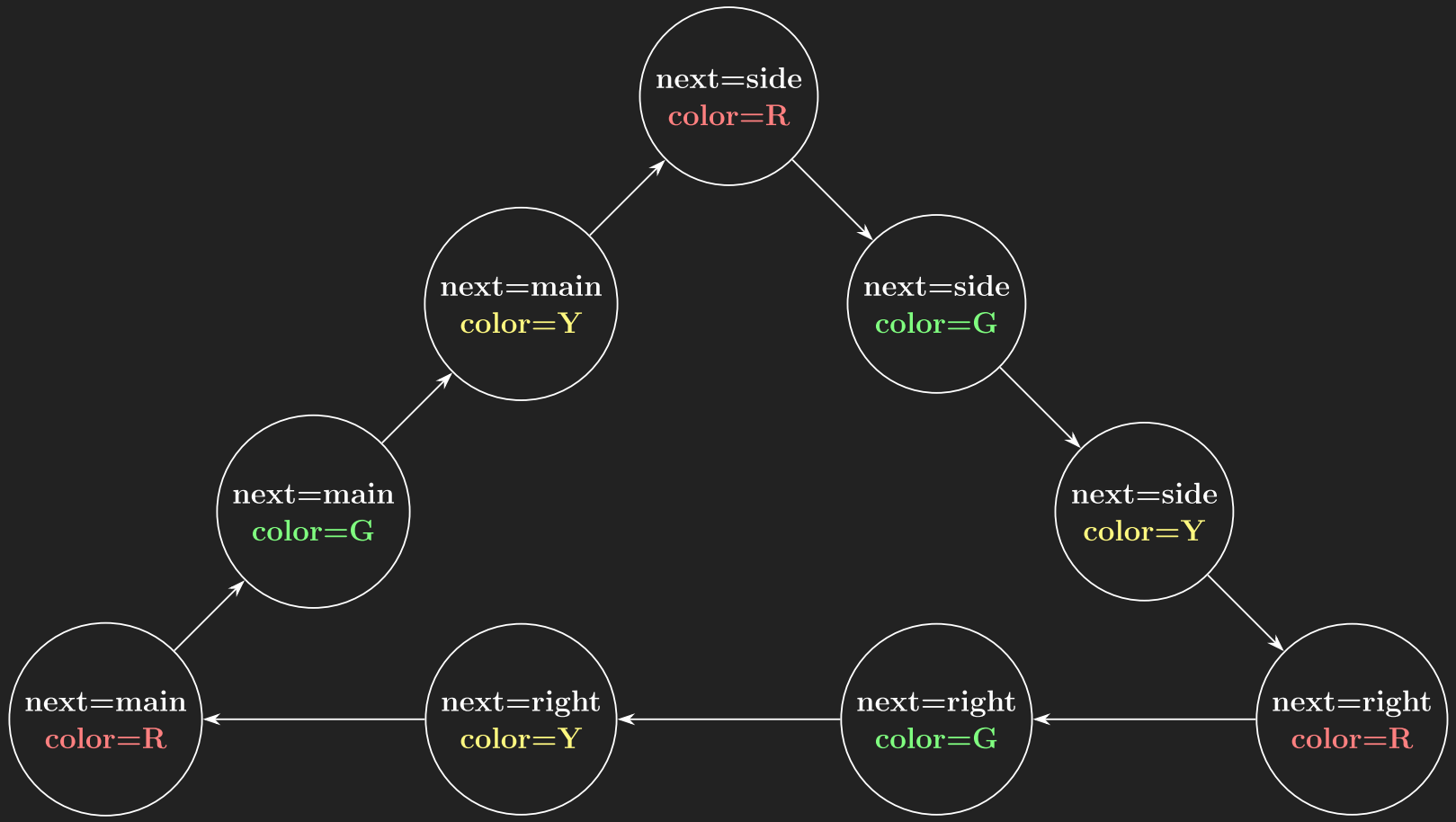
WITHOUT STATE



WITH STATE


updateLight
carArrivedMain
carArrivedSide
...
state
mainLightRed
mainLightYellow
mainLightGreen
mainLightRed
mainLightYellow
...


main=R
side=R
right=R
next=side
main=R
side=G
right=R
next=side
main=R
side=Y
right=R
next=side
main=R
side=R
right=R
next=right
main=R
side=R
right=G
next=right
main=R
side=R
right=Y
next=right
main=R
side=R
right=R
next=main
main=G
side=R
right=R
next=main
main=Y
side=R
right=R
next=main


next=side
color=R
next=side
color=G
next=side
color=Y
next=right
color=R
next=right
color=G
next=right
color=Y
next=main
color=R
next=main
color=G
next=main
color=Y

REMEMBER

- Be intentional about your
  - States
  - Events
  - Transitions
- Draw a diagram!

Finite state machines provide a useful abstraction to help us reason about the complex behaviour of our systems

Finite state machines provide a useful abstraction to help us reason about the complex behaviour of our systems

---

There are a lot more similarities between disciplines than you might think

THANK YOU!

github.com/tessereth/talks