Mastering views with Finite State Machines

Rubén Sospedra

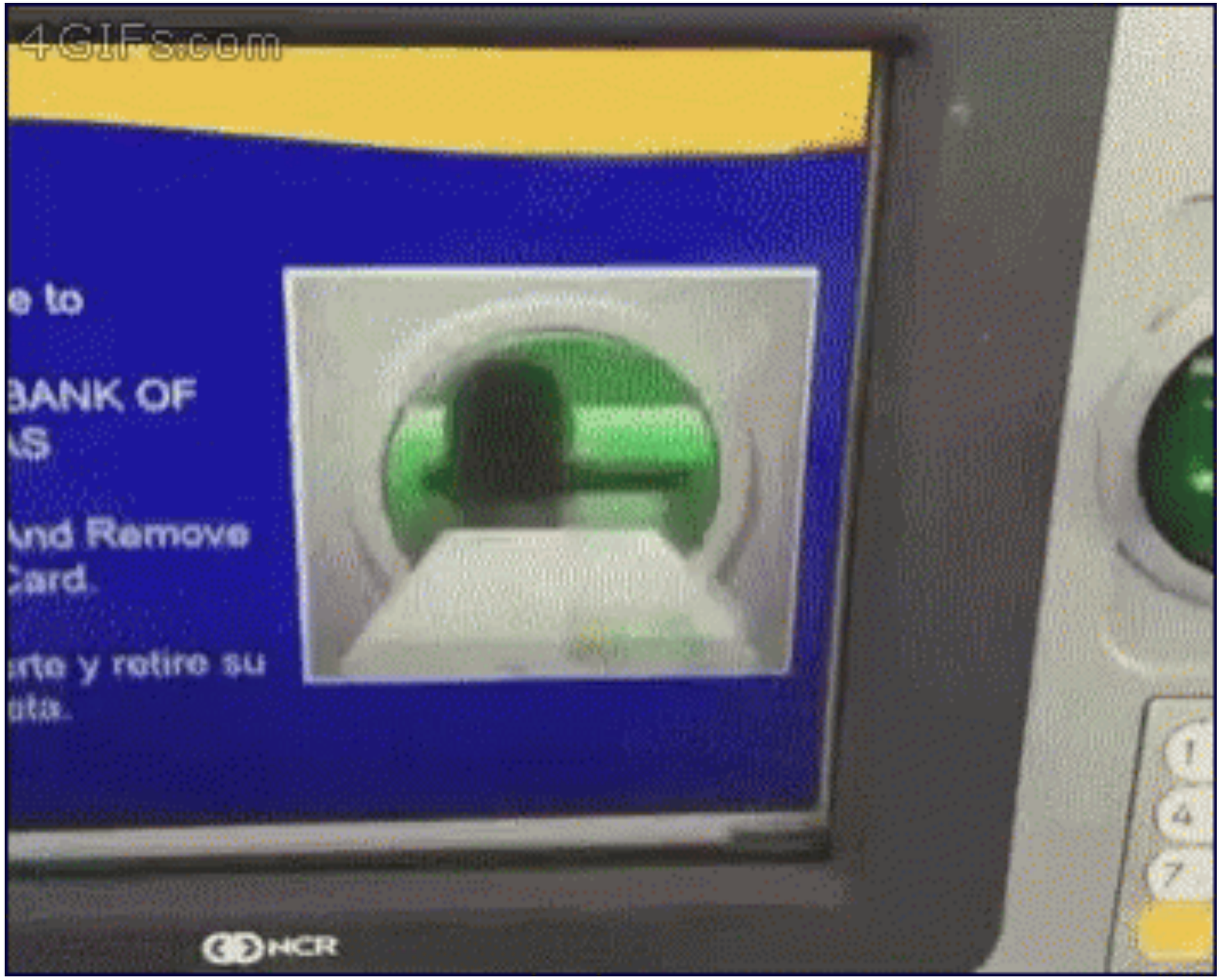
@sospedra_r

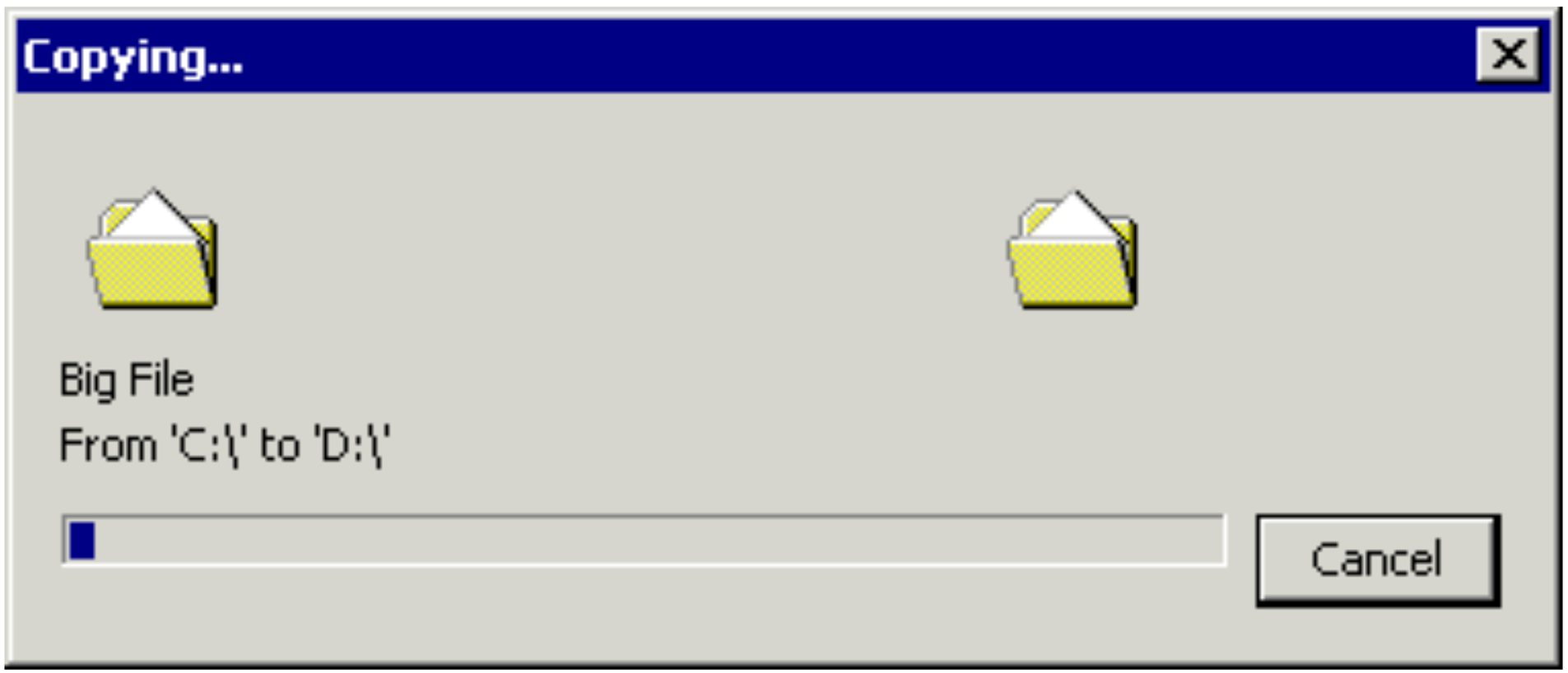
Javascript hacker

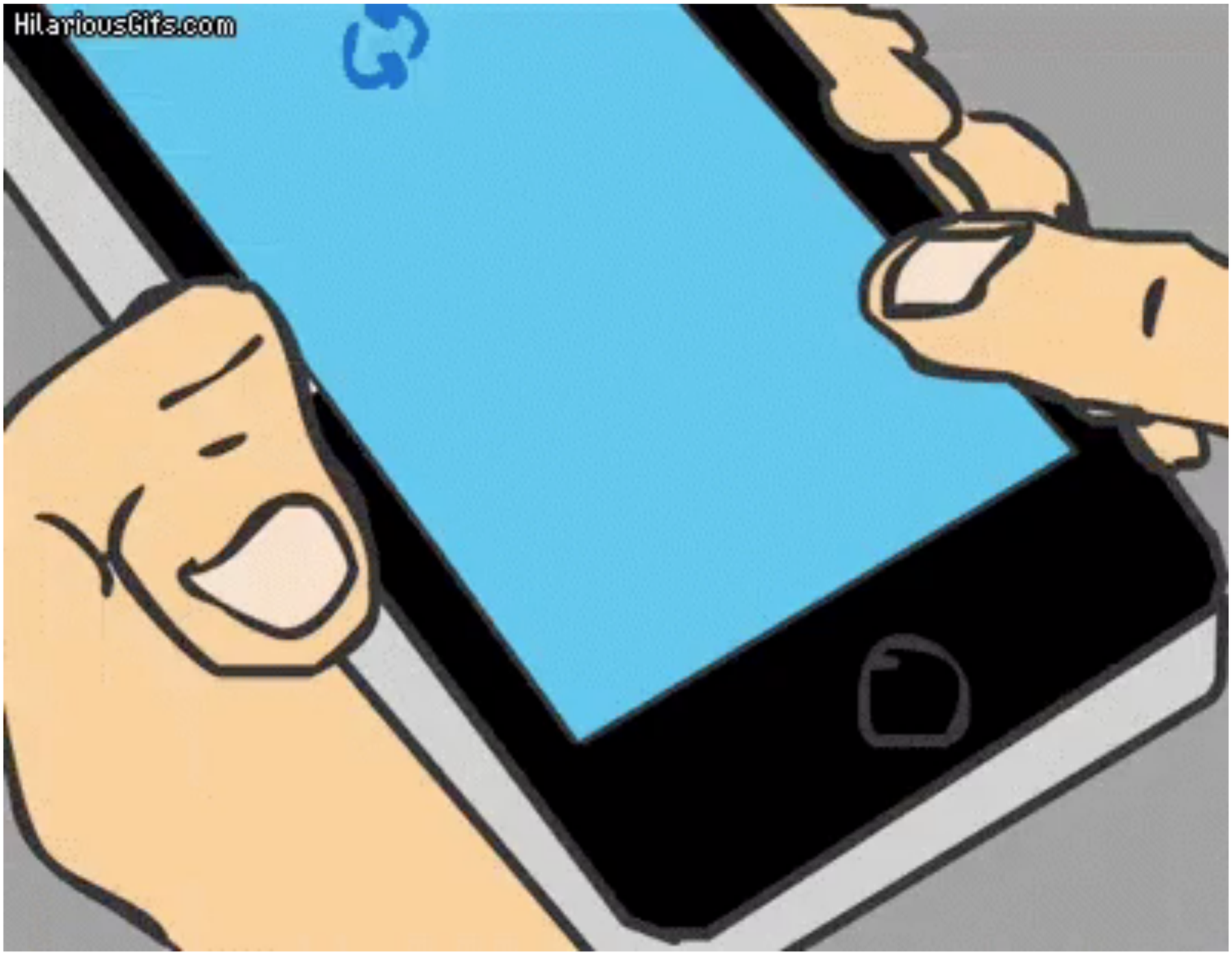
Senior software engineer at mytaxi

“develop user interfaces is not easy”

David Khourshid, 2017


4GIFs.com
e to
BANK OF
AS
And Remove
Card.
rte y retire su
ta.
NCR
1
4
7


Copying...
Big File
From 'C:\' to 'D:\'
Cancel


HilariousGifs.com

```
  fetchUser () {
    fetch('https://api.domain.com/user').then((response) => {
      response.json().then((payload) => {
        this.setState({ user: payload.user })
      })
    })
  }

  render () {
    <Main>
      <Text>{this.state.user.name}</Text>
    </Main>
  }
```

```
  fetchUser () {
    this.setState({ isLoading: true })
    fetch('https://api.domain.com/user').then((response) => {
      response.json().then((payload) => {
        this.setState({
          user: payload.user,
          isLoading: false
        })
      })
    })
  }

  render () {
    <Main>
      {this.state.isLoading
        ? <Loading />
        : <Text>{this.state.user.name}</Text>}
    </Main>
  }
```

```
  fetchUser () {
    this.setState({ isLoading: true })
    fetch('https://api.domain.com/user').then((response) => {
      response.json().then((payload) => {
        this.setState({
          user: payload.user,
          isLoading: false
        })
      }).catch(() => this.setState({ isError: true, isLoading: false }))
    }).catch(() => this.setState({ isError: true, isLoading: false }))
  }

  render () {
    <Main>
      {this.state.isError && <Error />}
      {!this.state.isError && this.state.isLoading
        ? <Loading />
        : <Text>{this.state.user.name}</Text>}
    </Main>
  }
```

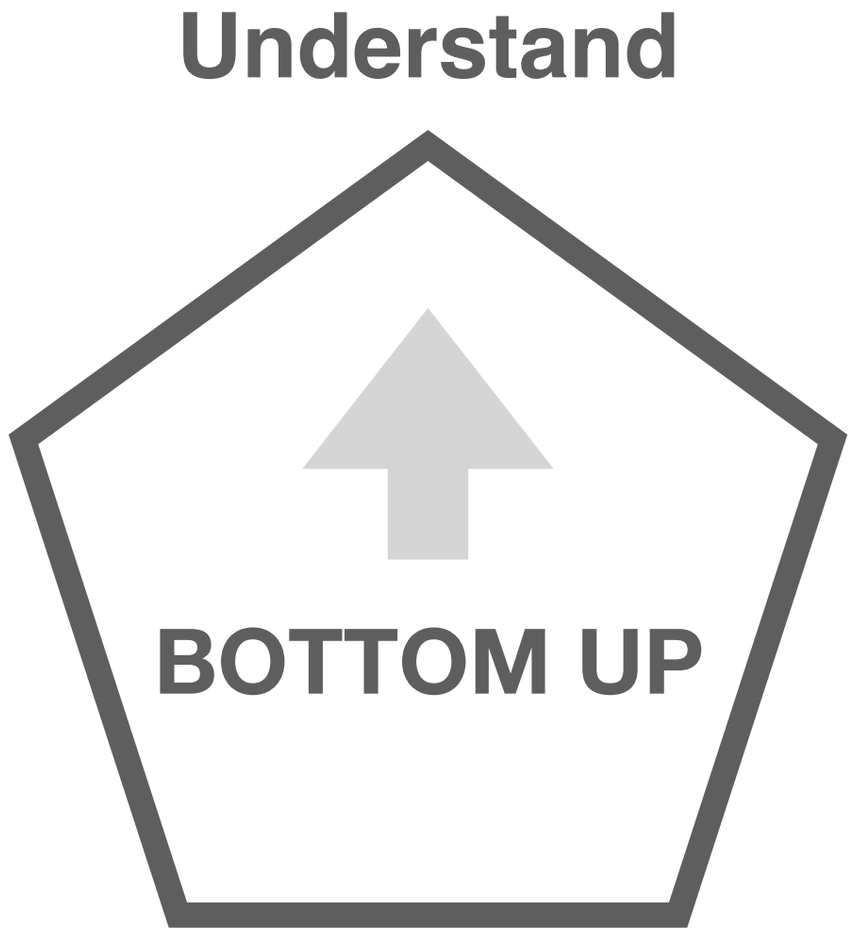
```
fetchUser () {
  this.setState({ isLoading: true })
  fetch('https://api.domain.com/user').then((response) => {
    if (this.state.hasCancel) return
    response.json().then((payload) => {
      this.setState({
        user: payload.user,
        isLoading: false
      })
    }).catch(() => {
      if (this.state.hasCancel) return
      this.setState({ isError: true, isLoading: false })
    })
  }).catch(() => {
    if (this.state.hasCancel) return
    this.setState({ isError: true, isLoading: false })
  })
}

render () {
  <Main>
    {this.state.isError && <Error />}
    {!this.state.isError && this.state.isLoading
      ? <Loading />
      : <Text>{this.state.user.name}</Text>}
    <button onClick={() => this.setState({ hasCancel: true })}>
      Cancel
    </button>
  </Main>
}
```

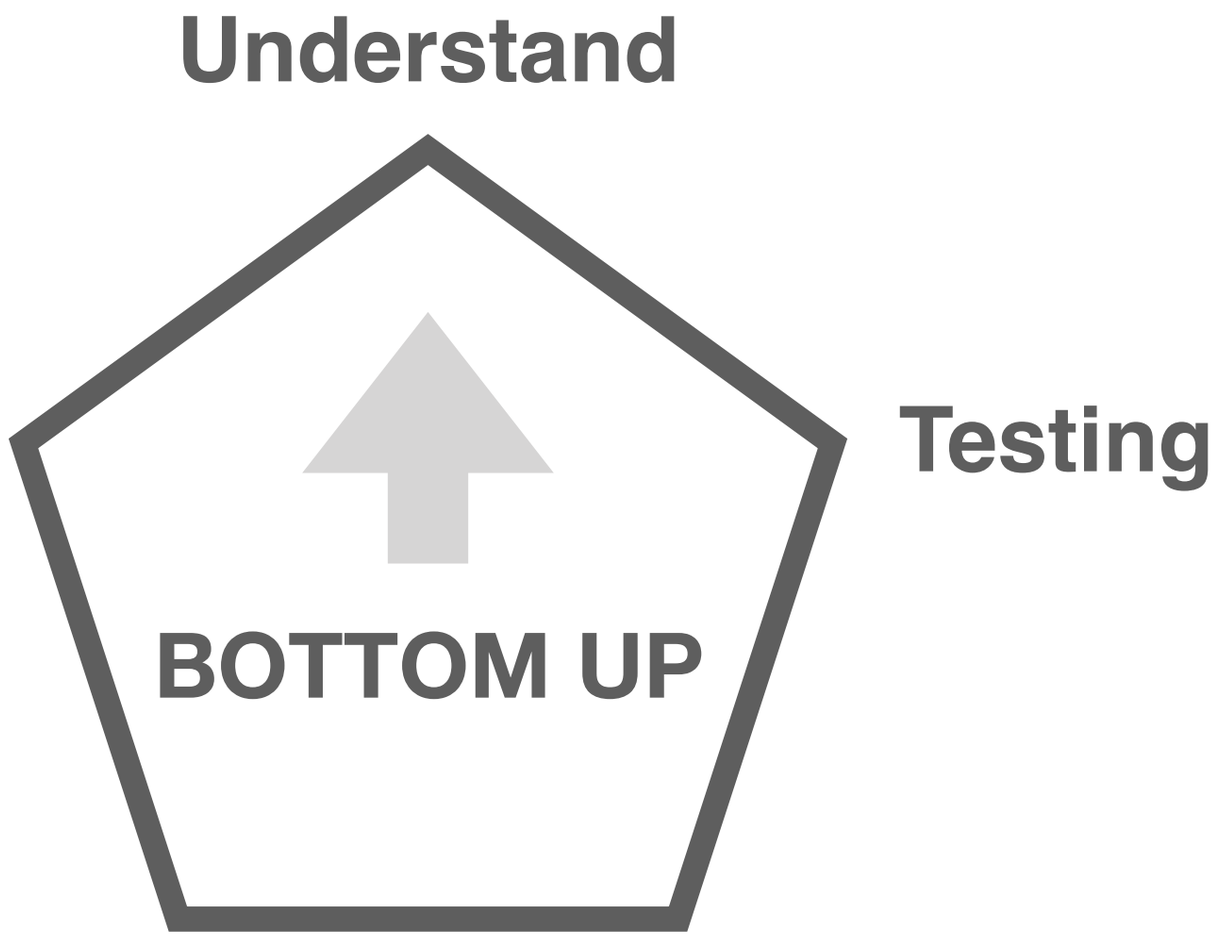
```
    this.setState({ isLoading: true })
    fetch('https://api.domain.com/user').then((response) => {
      if (this.state.hasCancel) return
      response.json().then((payload) => {
        this.setState({
          user: payload.user,
          isLoading: false
        })
      }).catch(() => {
        if (this.state.hasCancel) return
        this.setState({ isError: true, isLoading: false })
      })
    }).catch(() => {
      if (this.state.hasCancel) return
      this.setState({ isError: true, isLoading: false })
    })
  }

  render () {
    <Main>
      {this.state.isError && <Error />}
```

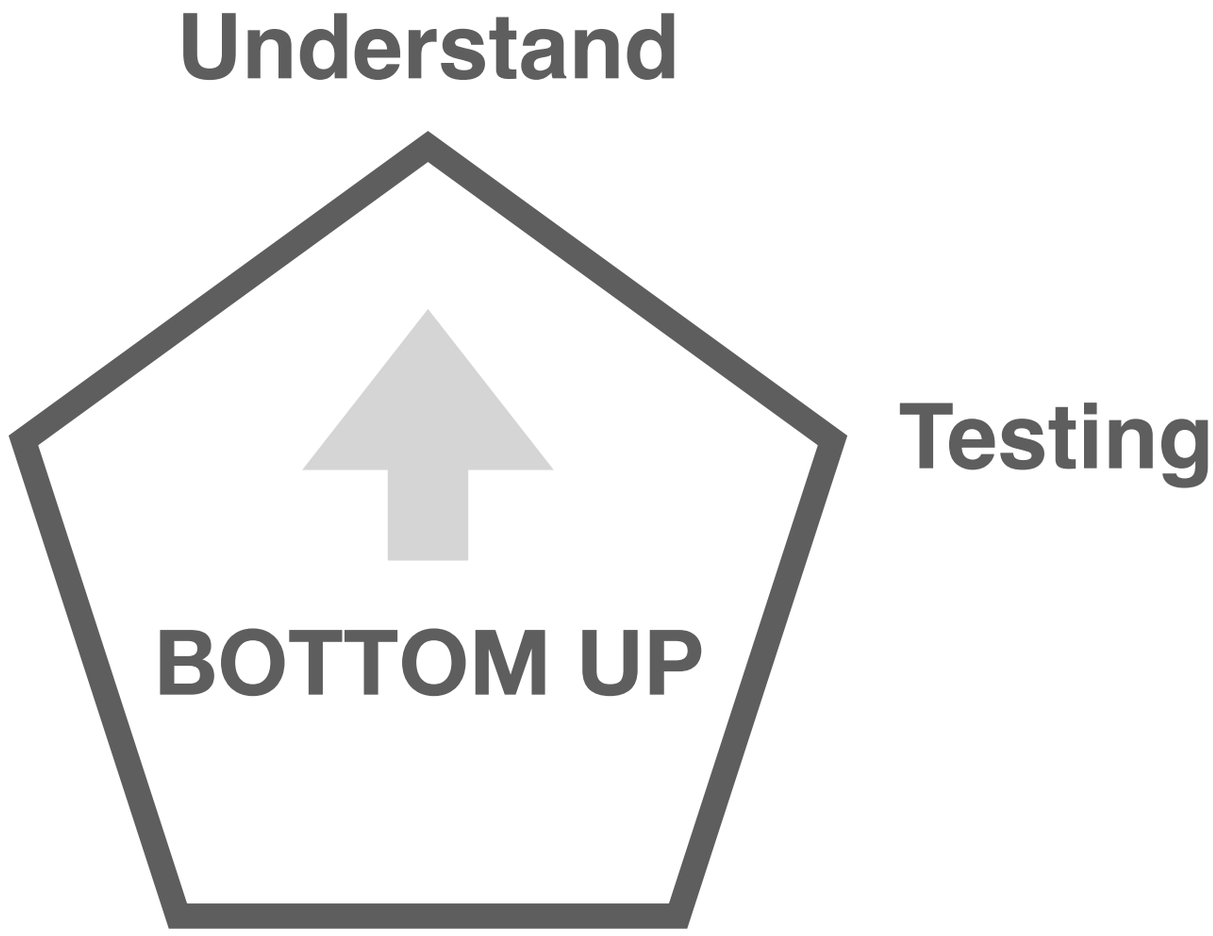
```
this.setState({ isLoading: tr
fetch('https://api.domain.com
  if (this.state.hasCancel) r
  response.json().then((paylo
    this.setState({¬
      user: payload.user,¬
      isLoading: false¬
    })¬
  }).catch(() => {¬
```

```
setState({ isLoad
('https://api.dom
(this.state.hasCa
ponse.json().then
his.setState({
```

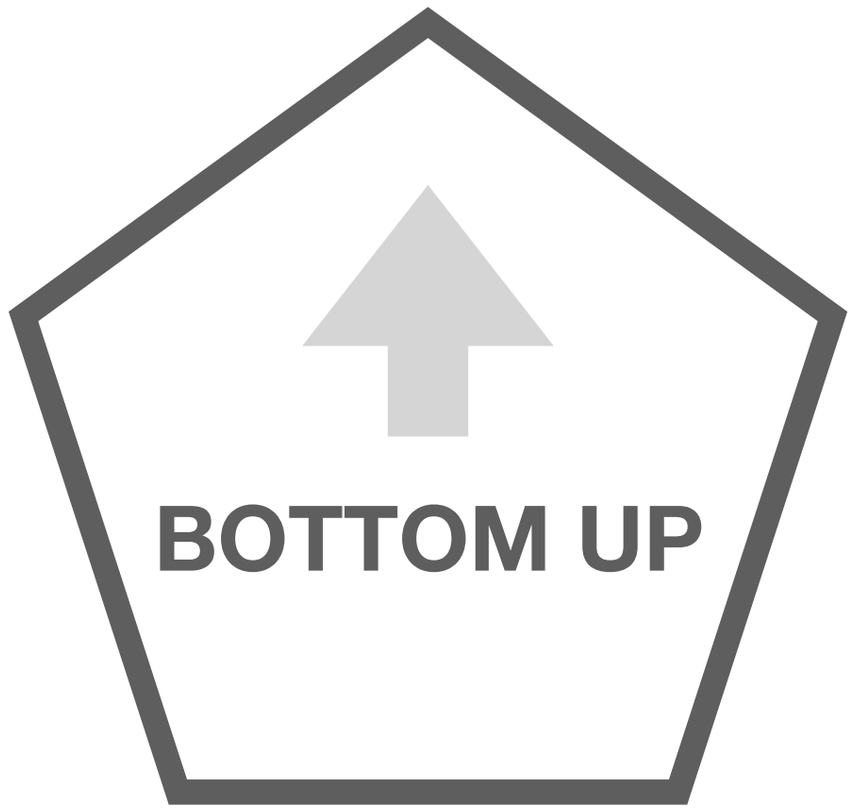
```
setState({ isLoad
('https://api.dom
(this.st e.hasCa
ponse.json().then
his.setState({
```


Understand
BOTTOM UP


Understand
Testing
BOTTOM UP


Understand
Testing
BOTTOM UP
Enhance

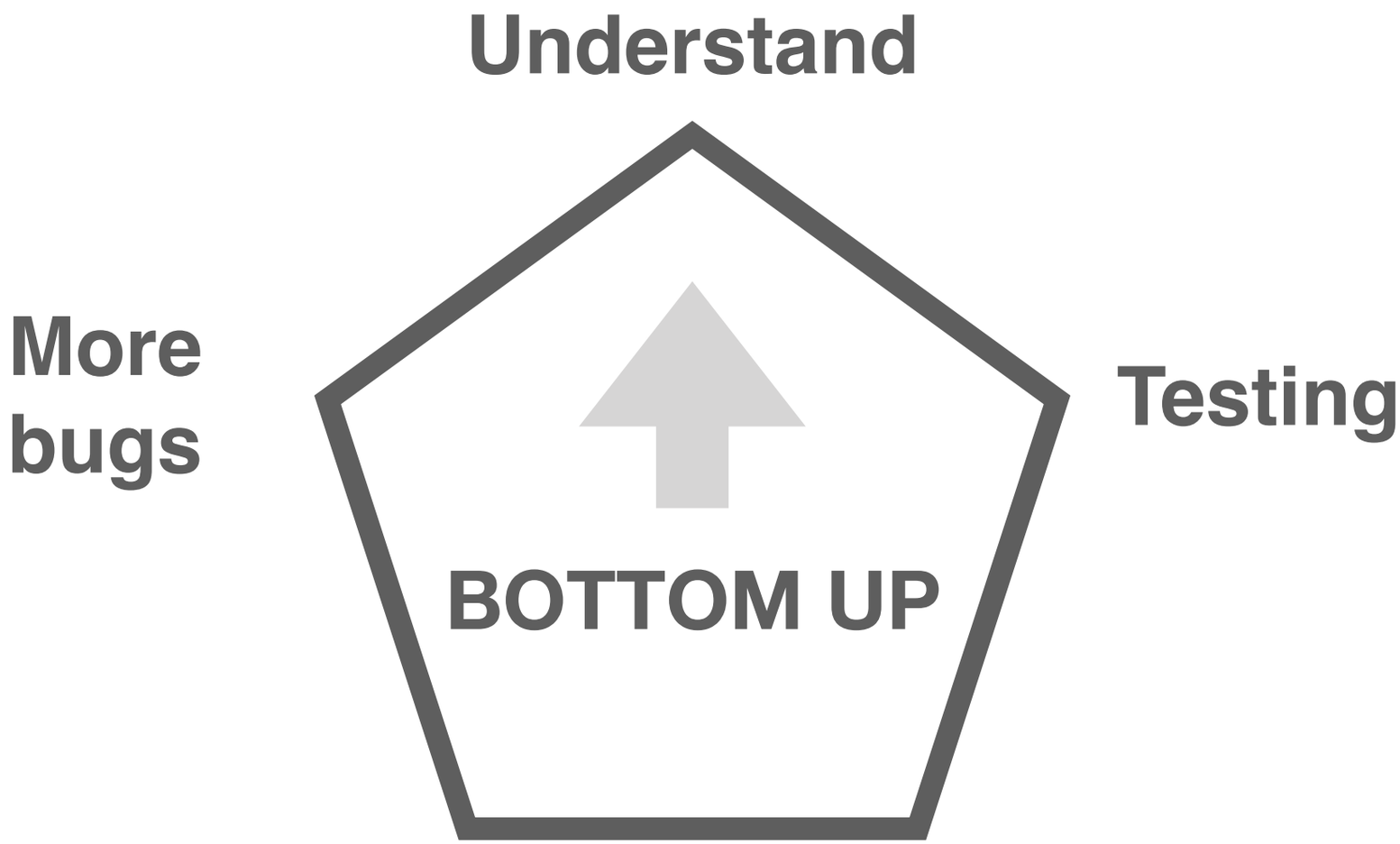
Understand

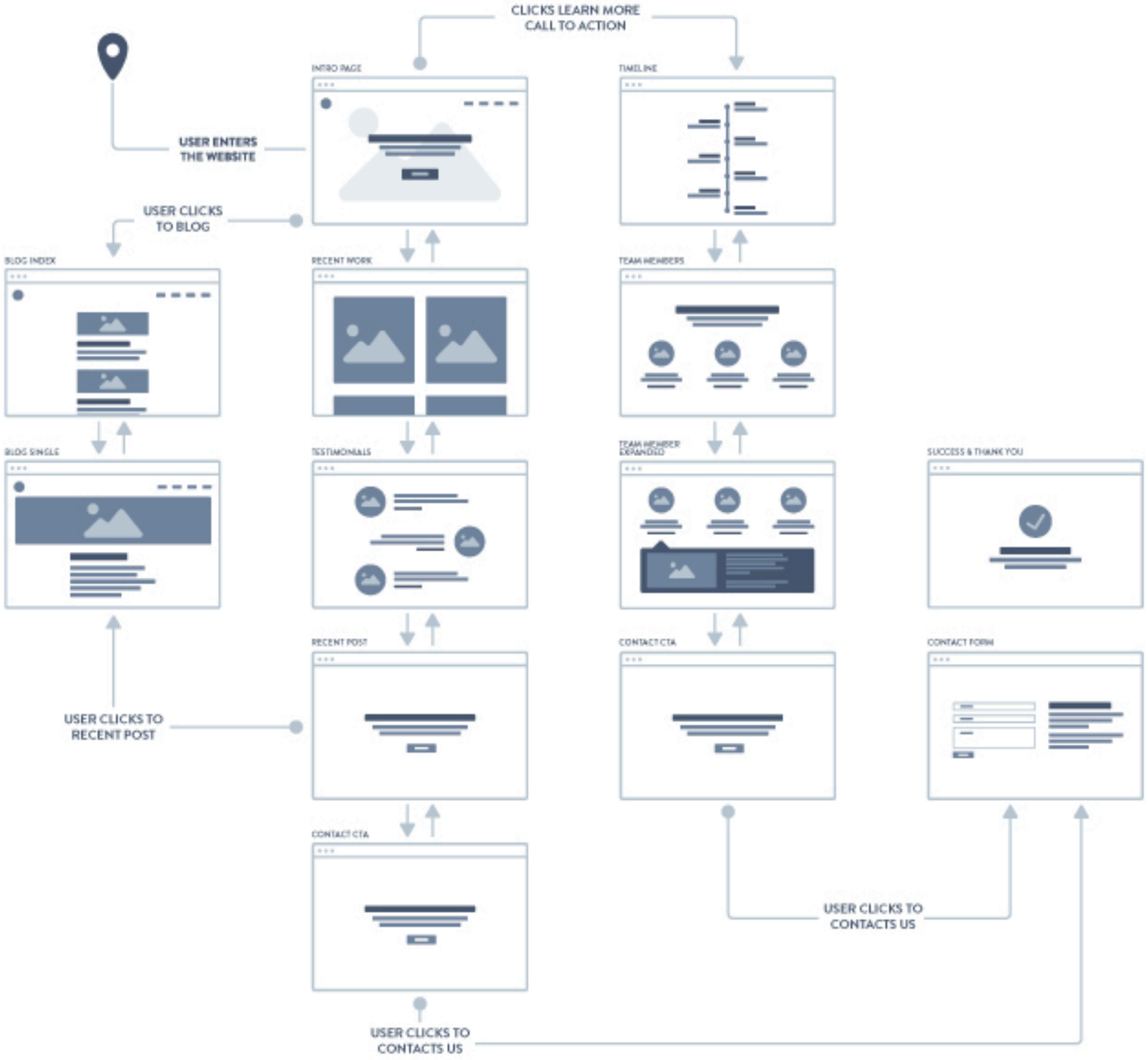


Testing

Bugs

Enhance


Understand
More bugs
Testing
BOTTOM UP
Bugs
Enhance


CLICKS LEARN MORE
CALL TO ACTION
INTRO PAGE
TIMELINE
USER ENTERS
THE WEBSITE
USER CLICKS
TO BLOG
BLOG INDEX
RECENT WORK
TEAM MEMBERS
BLOG SINGLE
TESTIMONIALS
TEAM MEMBER
EXPANDED
SUCCESS & THANK YOU
RECENT POST
CONTACT CTA
CONTACT FORM
USER CLICKS TO
RECENT POST
CONTACT CTA
USER CLICKS TO
CONTACTS US
USER CLICKS TO
CONTACTS US

finite state machines

In accordance with the general classification, the following formal definitions are found:

- A *deterministic finite state machine* or *acceptor deterministic finite state machine* is a quintuple $(\Sigma, S, s_0, \delta, F)$, where:
  - Σ is the input alphabet (a finite, non-empty set of symbols).
  - S is a finite, non-empty set of states.
  - s_0 is an initial state, an element of S.
  - δ is the state-transition function: $\delta : S \times \Sigma \rightarrow S\delta : S \times \Sigma \rightarrow \mathcal{P}(S)\delta$
  - F is the set of final states, a (possibly empty) subset of S.

Σ is the input alphabet (finite non-empty set)

Σ is the input alphabet (finite non-empty set)

```
const alphabet = {
  LOCK: 'LOCK',
  UNLOCK: 'UNLOCK'
}
```

S is a finite, non-empty set of states

S is a finite, non-empty set of states

```
const states = {
  LOCK: { ... },
  UNLOCK: { ... }
}
```

s_0 is an initial state, an element of S

s_0 is an initial state, an element of S

```
const initial = states[FIRST]
```

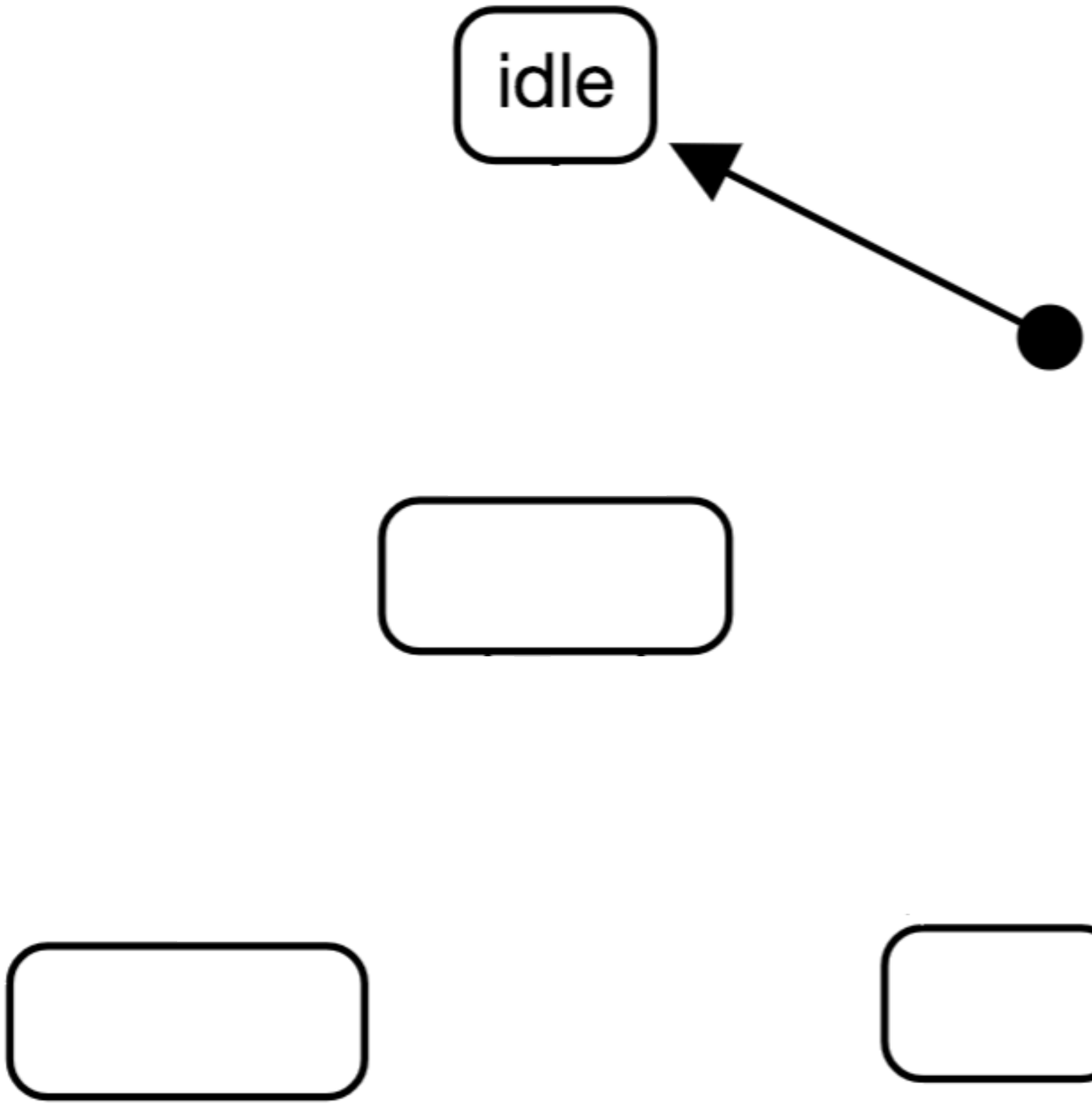
δ is the state-transition function: $\delta : S \text{ x } \Sigma \rightarrow S$

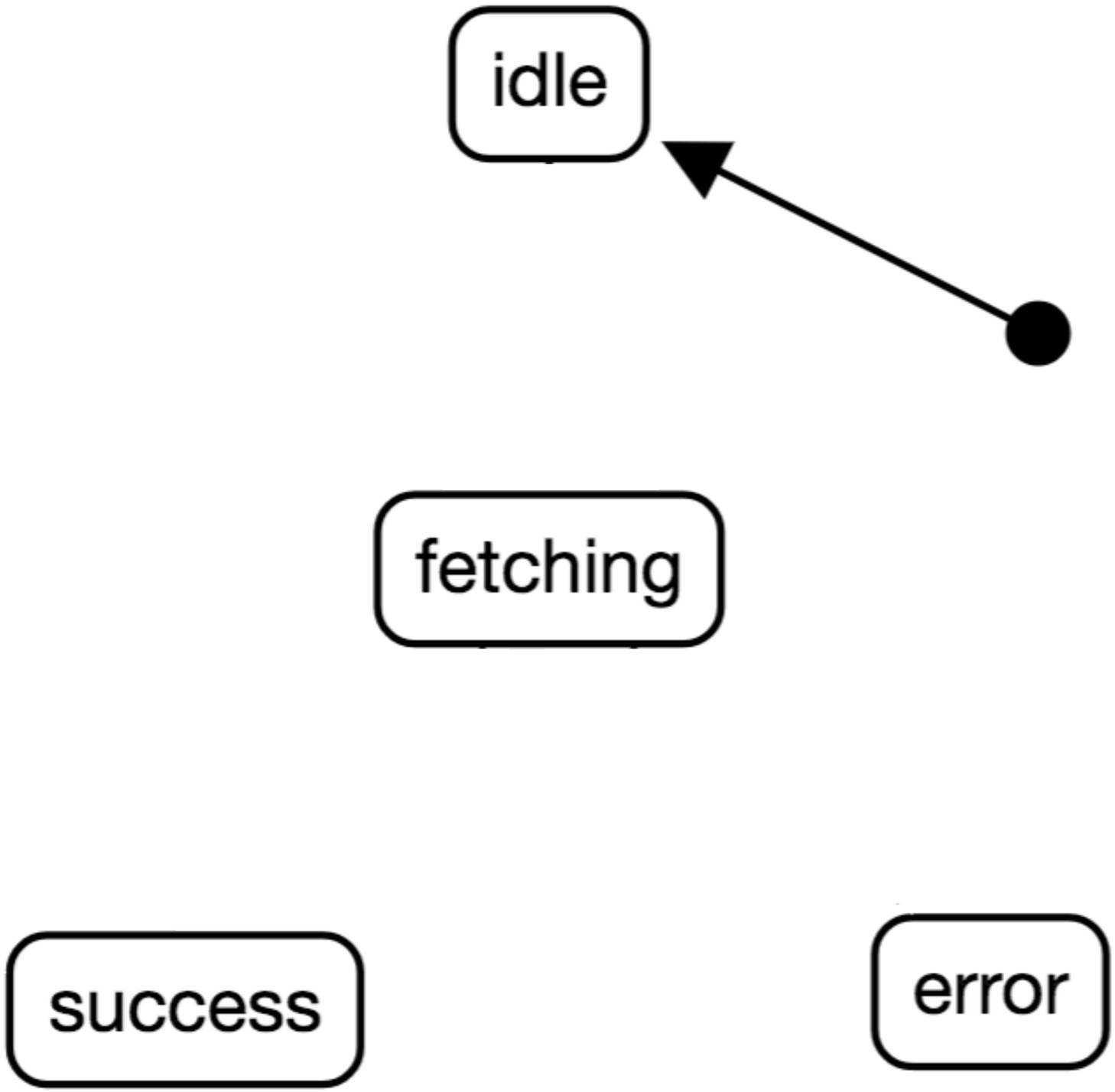
δ is the state-transition function: $\delta : S \text{ x } \Sigma \rightarrow S$

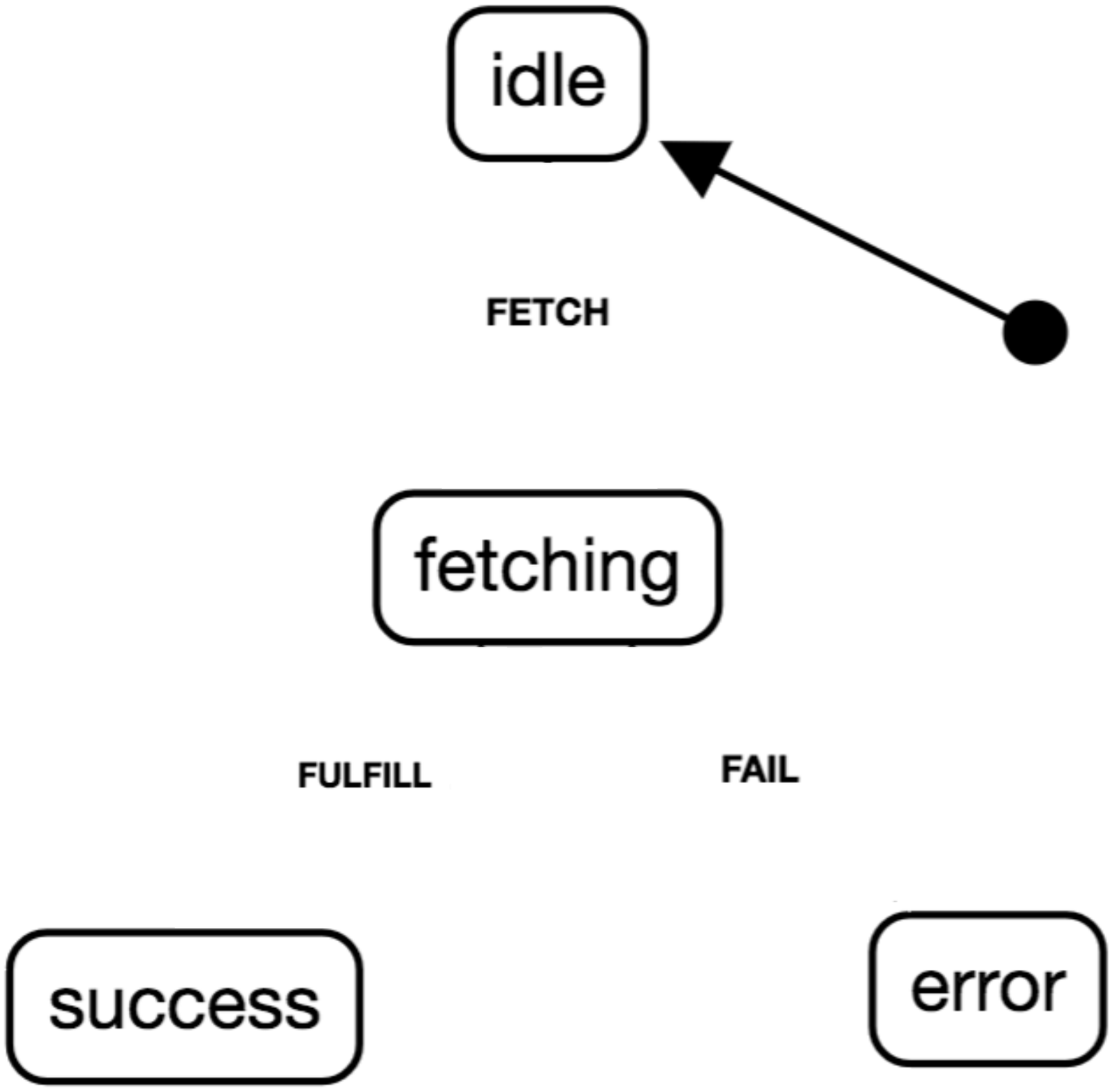
```
const transition = (states, alpha) => {
  return states[alpha]
}
```

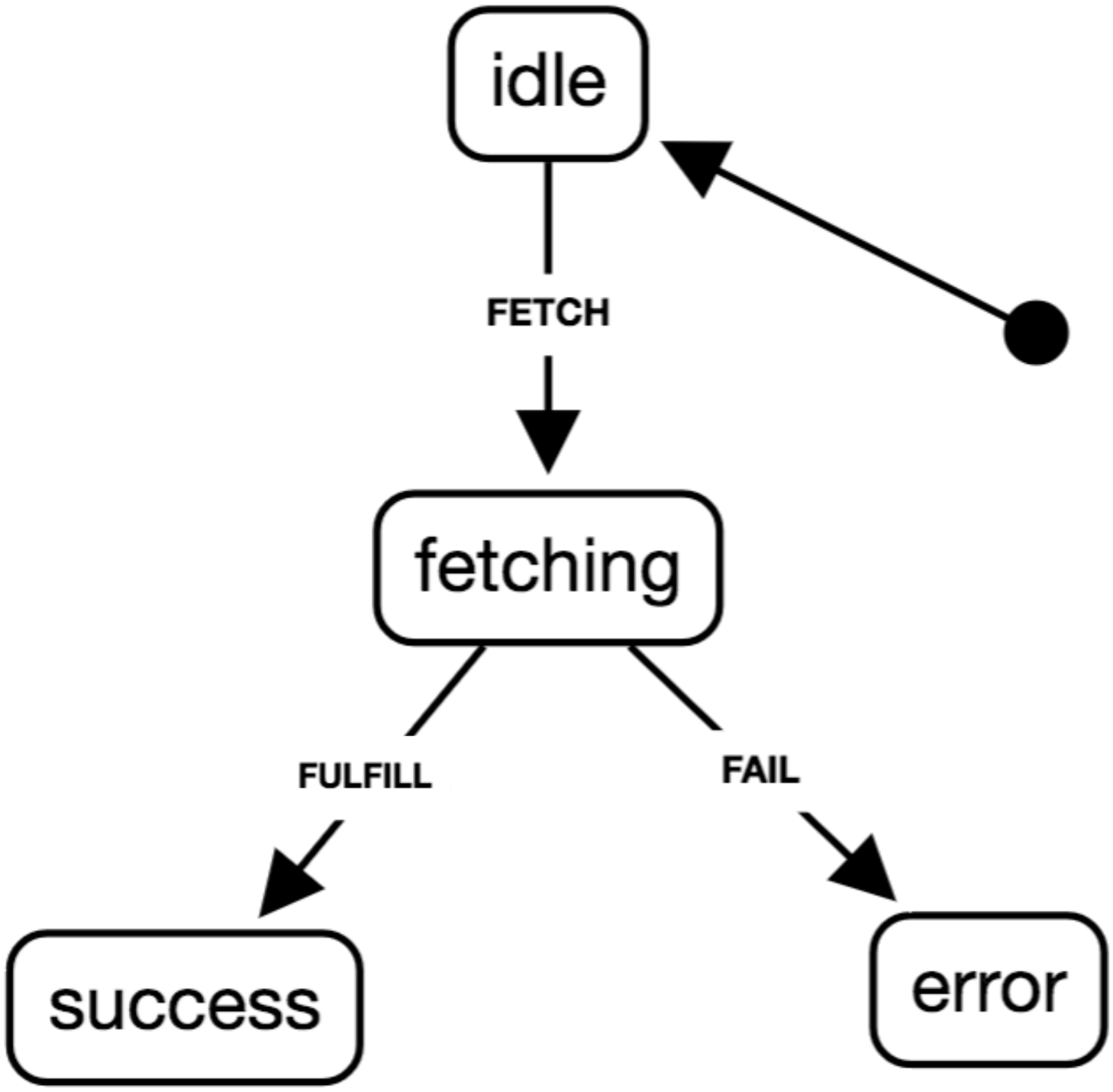
```
const machine = {
  LOCK: {
    next: UNLOCK
  },
  UNLOCK: {
    next: LOCK
  }
}

const initial = LOCK

const transition = (current) => {
  return machine[current].next
}
```

anatomy


idle


idle
fetching
success
error

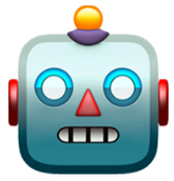

idle
FETCH
fetching
FULFILL
FAIL
success
error


idle
FETCH
fetching
FULFILL
FAIL
success
error

deterministic

END

finite

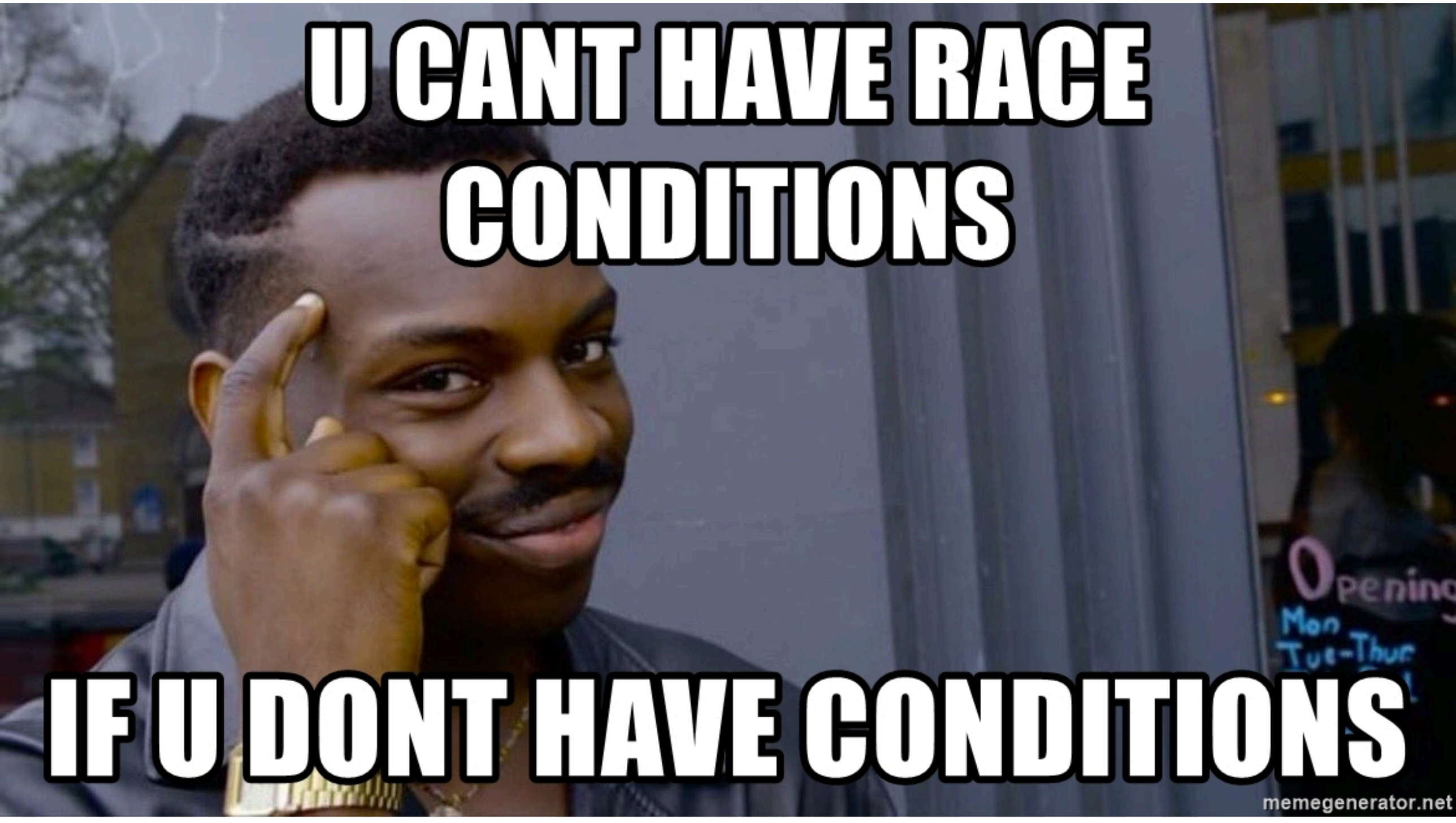


automata

demo

commands


WOAH


U CANT HAVE RACE CONDITIONS
IF U DONT HAVE CONDITIONS
memegenerator.net

decoupled

testing


[adult swim]

statecharts

“a visual formalism for complex systems”

David Harel, 1987

clustering

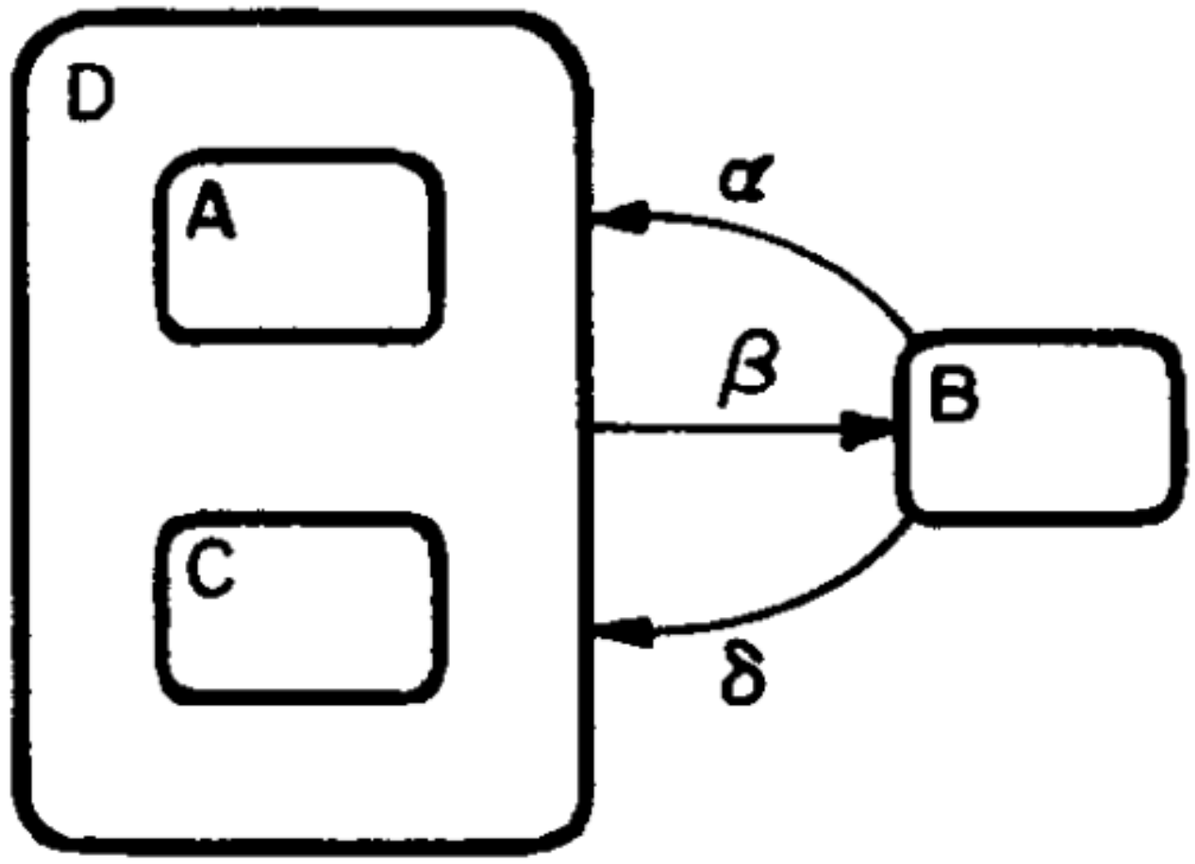


Fig. 4.

orthogonality

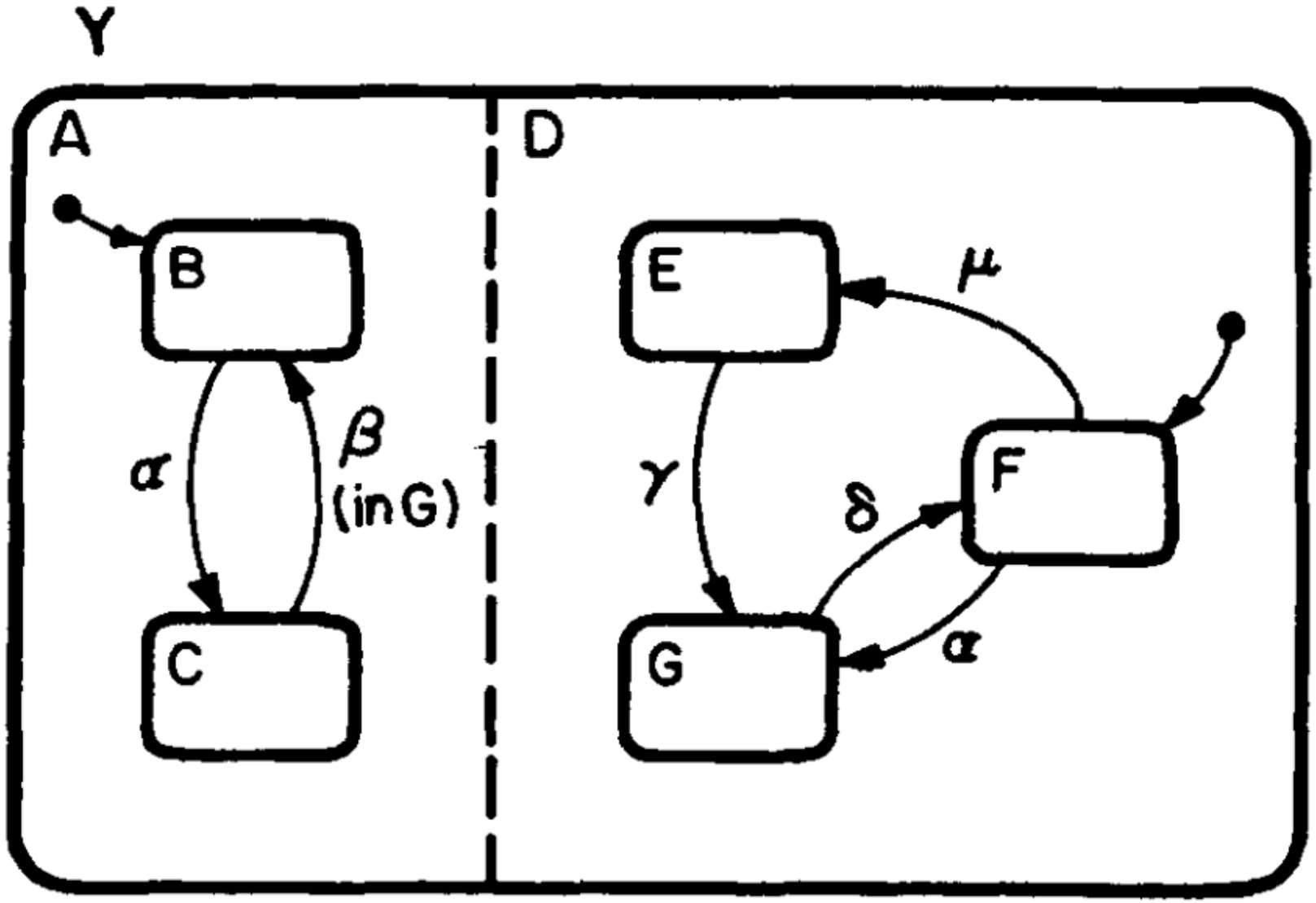


Fig. 19.

guards

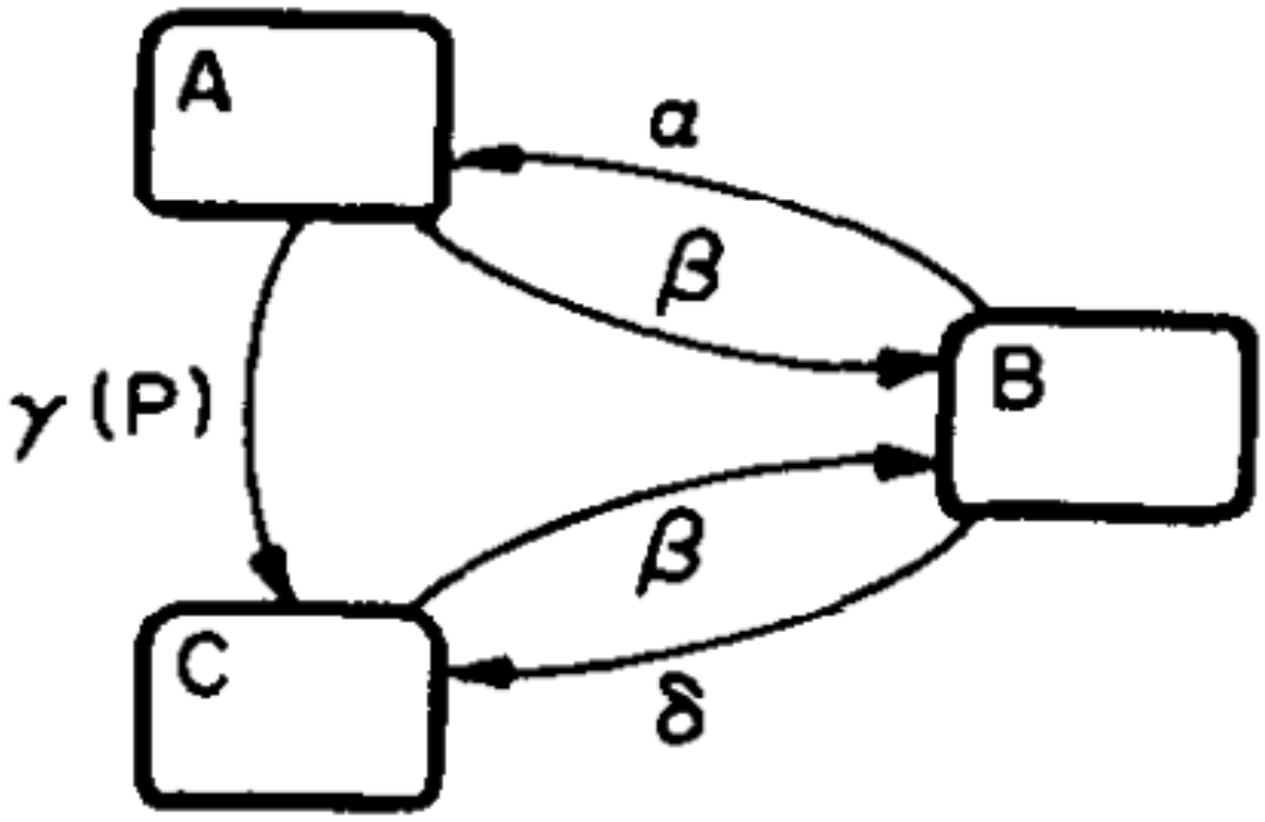
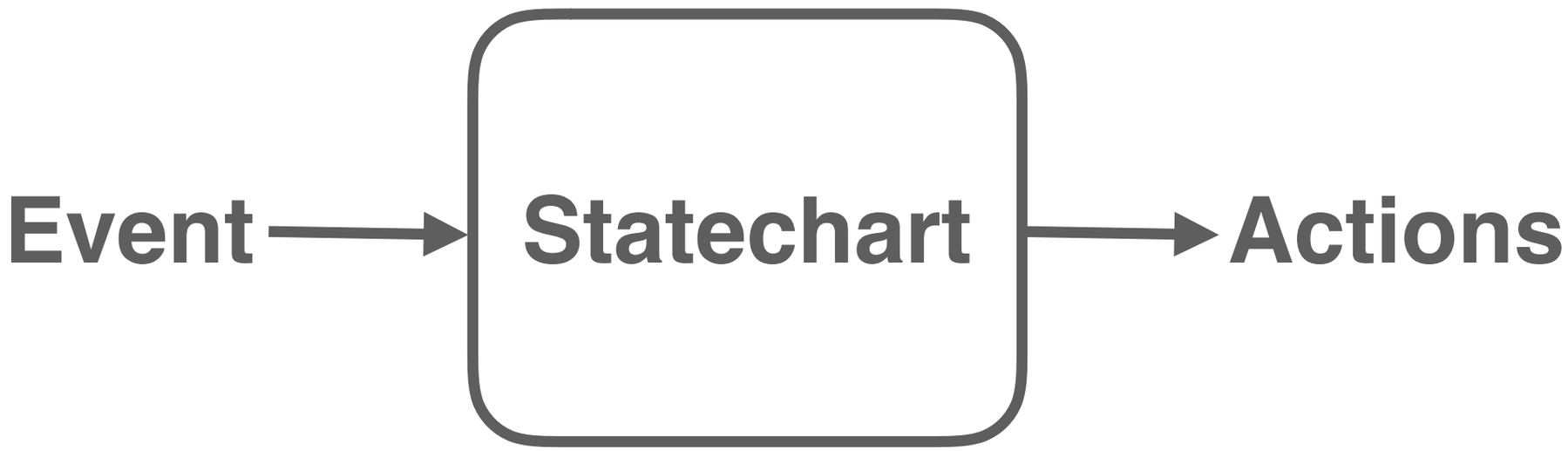


Fig. 1.

actions



resources

- Libraries
  - https://github.com/davidkpiano/xstate
  - https://github.com/MicheleBertoli/react-automata
  - https://github.com/cytoscape/cytoscape
- Reads
  - “Constructing the UI with Statecharts” by Ian Horrocks
  - https://rauchg.com/2015/pure-ui
  - http://www.inf.ed.ac.uk/teaching/courses/seoc/2005_200
    resources/statecharts.pdf
  - https://statecharts.github.io/
- Real world examples
  - VSCode telemetry
  - Curiosity rover

thank you

https://github.com/sospedra/talks

https://github.com/sospedra/finite-state-machine-demo