Reactive Semantics for User Interface Description Languages

Basile Pesin Celia Picard Cyril Allignol

Fédération ENAC ISAE-SUPAERO ONERA, Université de Toulouse, France

June 20, 2025

Specifying Interactive Applications

The evolution of Safety-Critical HMIs

Concorde (1976)

The evolution of Safety-Critical HMIs

visualize

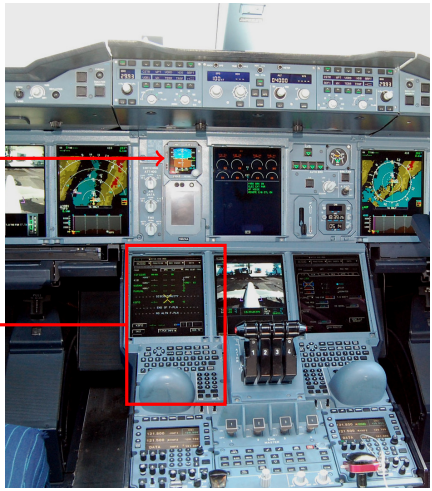
Concorde (1976)

Airbus A380 (2005)

The evolution of Safety-Critical HMIs

visualize

interact

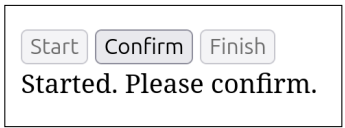


Concorde (1976)

Airbus A380 (2005)

Programming HMIs with procedural languages : the “callback hell”

```
startBtn.addEventListener("click", function () {
  startBtn.disabled = true;
  status.innerText = "Started. Please confirm.";
  confirmBtn.disabled = false;

  confirmBtn.addEventListener("click", function () {
    confirmBtn.disabled = true;
    status.innerText = "Confirmed. Please finish.";
    finishBtn.disabled = false;

    finishBtn.addEventListener("click", function () {
      finishBtn.disabled = true;
      status.innerText = "Process complete.";
    });
  });
});
```



Programming HMIs with procedural languages : the “callback hell”

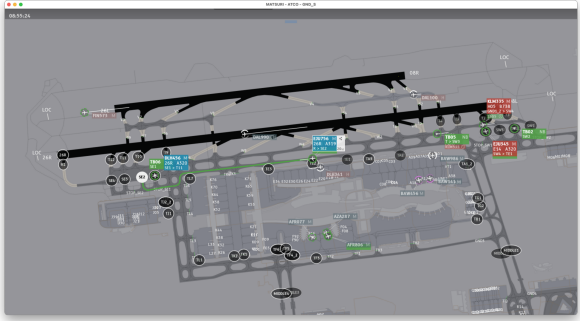
```
startBtn.addEventListener("click", function () {
  startBtn.disabled = true;
  status.innerText = "Started. Please confirm.";
  confirmBtn.disabled = false;

  confirmBtn.addEventListener("click", function () {
    confirmBtn.disabled = true;
    status.innerText = "Confirmed. Please finish.";
    finishBtn.disabled = false;

    finishBtn.addEventListener("click", function () {
      finishBtn.disabled = true;
      status.innerText = "Process complete.";
    });
  });
});
```

[Myers (1991): Separating application code from toolkits: Eliminating the spaghetti of call-backs]

Smala : a UIDL used in safety-critical applications

[Magnaudet, Chatty, Conversy, Leriche, Picard, and Prun (2018): Djnn/Smala: A Conceptual Framework and a Language for Interaction-Oriented Programming]

[Antoine and Conversy (2017): Volta: The First All-Electric Conventional Helicopter]



[Cousy et al. (2022): AEON: Toward a concept of operation and tools for supporting engine-off navigation for ground operations]

Programming with (almost) Smala

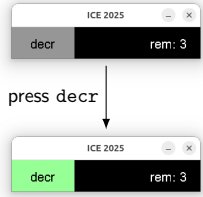

```
Component root {
  Int count 3
  Spike zero
  (count == 0) -> zero
  Frame f ("ICE␣2025", 300, 50)
  Font _ ("arial.ttf", 20)
  FillColor _ (150,150,150)
  Component btn1 {
    Rectangle r (0,0,100,f.height)
    r.press -> { 255 =: green }
    r.release -> { 150 =: green }
    FillColor _ (0,0,0)
    Text _ ("decr", r.x + 30, 13)
  }
  btn1.r.release -> { last count - 1 =: count }
  zero ->! btn1.r
  (count > 0) -> btn1.r

  ... // restart button + counter label
  Exit e (0)
  f.close -> e
}
```

Programming with (almost) Smala


```
Component root {
  Int count 3;
  Spike zero; (count == 0) -> zero;
  Frame f ("ICE␣2025", 300, 50) {
    Font _ ("arial.ttf", 20) {
      FillColor btn1 (150,150,150) {
        Rectangle r (0,0,100,f.height) {
          FillColor _ (0,0,0) {
            Text _ ("decr", r.x + 30, 13) {}
          };
        };
        r.press -> hg; hg: 255 =: green;
        r.release -> dhg; dhg: 150 =: green;
      };
      btn1.r.release -> dec; dec: last count - 1 =: count;
      zero ->! btn1.r; (count > 0) -> btn1.r;

      ... // restart button + counter label
    }
  };
  Exit e (0) { f.close -> trigger };
}
```

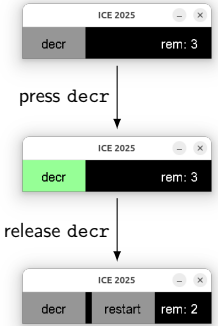
Programming with (almost) Smala

```
Component root {
  Int count 3;
  Spike zero; (count == 0) -> zero;
  Frame f ("ICE␣2025", 300, 50) {
    Font _ ("arial.ttf", 20) {
      FillColor btn1 (150,150,150) {
        Rectangle r (0,0,100,f.height) {
          FillColor _ (0,0,0) {
            Text _ ("decr", r.x + 30, 13) {}
          };
        };
        r.press -> hg; hg: 255 =: green;
        r.release -> dhg; dhg: 150 =: green;
      };
      btn1.r.release -> dec; dec: last count - 1 =: count;
      zero ->! btn1.r; (count > 0) -> btn1.r;

      ... // restart button + counter label
    }
  };
  Exit e (0) { f.close -> trigger };
}
```



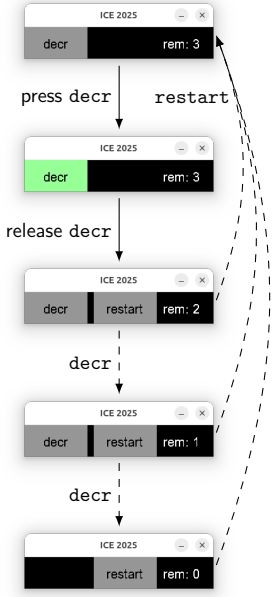
Programming with (almost) Smala

```
Component root {
  Int count 3;
  Spike zero; (count == 0) -> zero;
  Frame f ("ICE␣2025", 300, 50) {
    Font _ ("arial.ttf", 20) {
      FillColor btn1 (150,150,150) {
        Rectangle r (0,0,100,f.height) {
          FillColor _ (0,0,0) {
            Text _ ("decr", r.x + 30, 13) {}
          };
        };
        r.press -> hg; hg: 255 =: green;
        r.release -> dhg; dhg: 150 =: green;
      };
      btn1.r.release -> dec; dec: last count - 1 =: count;
      zero ->! btn1.r; (count > 0) -> btn1.r;

      ... // restart button + counter label
    }
  };
  Exit e (0) { f.close -> trigger };
}
```



Programming with (almost) Smala

```
Component root {
  Int count 3;
  Spike zero; (count == 0) -> zero;
  Frame f ("ICE␣2025", 300, 50) {
    Font _ ("arial.ttf", 20) {
      FillColor btn1 (150,150,150) {
        Rectangle r (0,0,100,f.height) {
          FillColor _ (0,0,0) {
            Text _ ("decr", r.x + 30, 13) {}
          };
        };
        r.press -> hg; hg: 255 =: green;
        r.release -> dhg; dhg: 150 =: green;
      };
      btn1.r.release -> dec; dec: last count - 1 =: count;
      zero ->! btn1.r; (count > 0) -> btn1.r;

      ... // restart button + counter label
    }
  };
  Exit e (0) { f.close -> trigger };
}
```



Formalizing Smala

We would like to:

- specify formally what programs do
- prove properties of source programs
- specify and prove the correctness of compilation to imperative code

We need formal semantics for Smala !

Formalizing Smala

We would like to:

- specify formally what programs do
- prove properties of source programs
- specify and prove the correctness of compilation to imperative code

We need formal semantics for Smala !

Previous work: bigraphical semantics for Smala [Nalpon, Allignol, and Picard (2022): Towards a User Interface Description Language Based on Bigraphs]

- Has been used to specify/prove program properties by model checking
- Rocq mechanization in progress
- Not sure its good for proving compilation algorithms

Formalizing Smala

We would like to:

- specify formally what programs do
- prove properties of source programs
- specify and prove the correctness of compilation to imperative code

We need formal semantics for Smala !

Previous work: bigraphical semantics for Smala [Nalpon, Allignol, and Picard (2022): Towards a User Interface Description Language Based on Bigraphs]

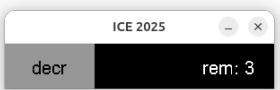
- Has been used to specify/prove program properties by model checking
- Rocq mechanization in progress
- Not sure its good for proving compilation algorithms

This work: a new relational model (mechanized in ROCQ)

Smalite: a Formalized UIDL

Execution Model

$(E_0,\ A_0)$

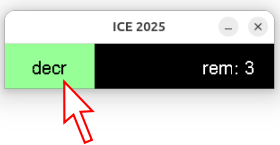


where :

$E_i : Path \rightharpoonup Value$

$A_i \subseteq Path$

Execution Model

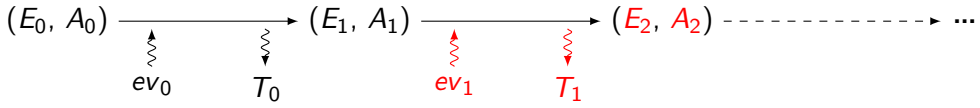


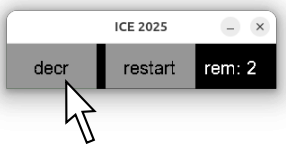
where :

$$
\begin{array}{rcl}
E_i & : & Path \rightharpoonup Value \\
A_i & \subseteq & Path \\
ev_i & \in & Event \\
T_i & \subseteq & Event
\end{array}
$$

$ev ::= \texttt{Trigger}\ path \mid \texttt{Assign}\ v\ path$

Execution Model



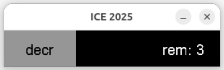


where :

$$
\begin{array}{rcl}
E_i & : & Path \rightharpoonup Value \\
A_i & \subseteq & Path \\
ev_i & \in & Event \\
T_i & \subseteq & Event
\end{array}
$$

$ev ::=$ `Trigger` $path$ | `Assign` v $path$ | `Activate` $path$ | `Deactivate` $path$

System Initialization – Example

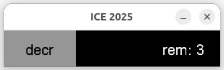



```
Int count 3; ...
Frame f ("ICE␣2025", 300, 50) { ...
  FillColor btn1 (150,150,150) {
    Rectangle r (0,0,100,f.height) { ...
      Text t ("decr", r.x + 30, 13) {} ...
    }; ...
  };
  FillColor btn2 (150,150,150) {
    Rectangle<d> r (0,0,100,f.height) { ...
      Text t ("restart", r.x + 30, 13) {} ...
    }; ...
  }; ...
}; ...
```

$$E = \left\{ \begin{array}{l} \texttt{count} \mapsto 3 \\ \\ \\ \\ \ldots \end{array} \right\}$$

$$A = \left\{ \begin{array}{l} \\ \\ \\ \\ \\ \end{array} \right.$$

System Initialization – Example

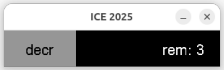



```
Int count 3; ...
Frame f ("ICE␣2025", 300, 50) { ...
  FillColor btn1 (150,150,150) {
    Rectangle r (0,0,100,f.height) { ...
      Text t ("decr", r.x + 30, 13) {} ...
    }; ...
  };
  FillColor btn2 (150,150,150) {
    Rectangle<d> r (0,0,100,f.height) { ...
      Text t ("restart", r.x + 30, 13) {} ...
    }; ...
  }; ...
}; ...
```

$$E = \left\{ \begin{array}{l} \texttt{count} \mapsto 3 \\ \texttt{f.height} \mapsto 50 \\ \\ \\ \ldots \end{array} \right\}$$

$$A = \left\{ \begin{array}{l} \\ \\ \\ \\ \\ \end{array} \right.$$

System Initialization – Example

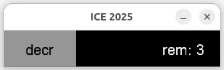



```
Int count 3; ...
Frame f ("ICE 2025", 300, 50) { ...
  FillColor btn1 (150,150,150) {
    Rectangle r (0,0,100,f.height) { ...
      Text t ("decr", r.x + 30, 13) {} ...
    }; ...
  };
  FillColor btn2 (150,150,150) {
    Rectangle<d> r (0,0,100,f.height) { ...
      Text t ("restart", r.x + 30, 13) {} ...
    }; ...
  }; ...
}; ...
```

$$E = \left\{ \begin{array}{l} \texttt{count} \mapsto 3 \\ \texttt{f.height} \mapsto 50 \\ \texttt{f.btn1.r.x} \mapsto 0 \\ \texttt{f.btn1.r.height} \mapsto 50 \\ \texttt{f.btn1.r.t.x} \mapsto 30 \\ \ldots \end{array} \right\}$$

$$A = \left\{ \right.$$

System Initialization – Example




```
Int count 3; ...
Frame f ("ICE 2025", 300, 50) { ...
  FillColor btn1 (150,150,150) {
    Rectangle r (0,0,100,f.height) { ...
      Text t ("decr", r.x + 30, 13) {} ...
    }; ...
  };
  FillColor btn2 (150,150,150) {
    Rectangle<d> r (0,0,100,f.height) { ...
      Text t ("restart", r.x + 30, 13) {} ...
    }; ...
  }; ...
}; ...
```

$$E = \left\{ \begin{array}{l} \texttt{count} \mapsto 3 \\ \texttt{f.height} \mapsto 50 \\ \texttt{f.btn1.r.x} \mapsto 0 \\ \texttt{f.btn1.r.height} \mapsto 50 \\ \texttt{f.btn1.r.t.x} \mapsto 30 \\ \ldots \end{array} \right\}$$

$$A = \left\{ \begin{array}{l} \texttt{f} \end{array} \right.$$

System Initialization – Example


```
Int count 3; ...
Frame f ("ICE 2025", 300, 50) { ...
  FillColor btn1 (150,150,150) {
    Rectangle r (0,0,100,f.height) { ...
      Text t ("decr", r.x + 30, 13) {} ...
    }; ...
  };
  FillColor btn2 (150,150,150) {
    Rectangle<d> r (0,0,100,f.height) { ...
      Text t ("restart", r.x + 30, 13) {} ...
    }; ...
  }; ...
}; ...
```

$$E = \left\{ \begin{array}{l} \texttt{count} \mapsto 3 \\ \texttt{f.height} \mapsto 50 \\ \texttt{f.btn1.r.x} \mapsto 0 \\ \texttt{f.btn1.r.height} \mapsto 50 \\ \texttt{f.btn1.r.t.x} \mapsto 30 \\ \ldots \end{array} \right\}$$

$$A = \left\{ \begin{array}{l} \texttt{f} \\ \texttt{f.btn1} \\ \texttt{f.btn1.r} \\ \texttt{f.btn2} \\ \ldots \end{array} \right\}$$

System Initialization – Example


```
Int count 3; ...
Frame f ("ICE␣2025", 300, 50) { ...
  FillColor btn1 (150,150,150) {
    Rectangle r (0,0,100,f.height) { ...
      Text t ("decr", r.x + 30, 13) {} ...
    }; ...
  };
  FillColor btn2 (150,150,150) {
    Rectangle<d> r (0,0,100,f.height) { ...
      Text t ("restart", r.x + 30, 13) {} ...
    }; ...
  }; ...
}; ...
```

$$E = \left\{ \begin{array}{l} \texttt{count} \mapsto 3 \\ \texttt{f.height} \mapsto 50 \\ \texttt{f.btn1.r.x} \mapsto 0 \\ \texttt{f.btn1.r.height} \mapsto 50 \\ \texttt{f.btn1.r.t.x} \mapsto 30 \\ \ldots \end{array} \right\}$$

$$A = \left\{ \begin{array}{l} \texttt{f} \\ \texttt{f.btn1} \\ \texttt{f.btn1.r} \\ \texttt{f.btn2} \\ \ldots \end{array} \right\}$$

System Initialization – Rules

$$\frac{\emptyset \vdash^{\text{env}}_{\text{init}} proc_{\epsilon} \Downarrow E \qquad \vdash^{\text{activ}}_{\text{init}} proc_{\epsilon} \Downarrow A}{\vdash_{\text{init}} proc \Downarrow E, A}$$

$$\frac{\emptyset, E \vdash_{\text{exp}} e \Downarrow v}{E \vdash^{\text{env}}_{\text{init}} (ty\ x\ e)_{r_x} \Downarrow E[r_x.x \mapsto v]}$$

$$\frac{}{E \vdash^{\text{env}}_{\text{init}} (\texttt{Component<\_>}\ x\ \{\epsilon\})_{r_x} \Downarrow E}$$

$$\frac{E \vdash^{\text{env}}_{\text{init}} p_{(r_x.x)} \Downarrow E' \qquad E' \vdash^{\text{env}}_{\text{init}} (\texttt{Component<\_>}\ x\ \{ps\})_{r_x} \Downarrow E''}{E \vdash^{\text{env}}_{\text{init}} (\texttt{Component<\_>}\ x\ \{p; ps\})_{r_x} \Downarrow E''}$$

$$\frac{}{\vdash^{\text{activ}}_{\text{init}} (\texttt{Component<d>}\ x\ \{ps\})_{r_x} \Downarrow \emptyset}$$

$$\frac{\forall i.\ \vdash^{\text{activ}}_{\text{init}} (ps_i)_{r_x.x} \Downarrow A_i}{\vdash^{\text{activ}}_{\text{init}} (\texttt{Component<a>}\ x\ \{ps\})_{r_x} \Downarrow (\bigcup_i A_i) \cup \{r_x.x\}}$$

System Initialization – Rules

$$\frac{\emptyset \vdash^{\text{env}}_{\text{init}} proc_{\epsilon} \Downarrow E \qquad \vdash^{\text{activ}}_{\text{init}} proc_{\epsilon} \Downarrow A}{\vdash_{\text{init}} proc \Downarrow E, A}$$

$$\frac{\emptyset, E \vdash_{\text{exp}} e \Downarrow v}{E \vdash^{\text{env}}_{\text{init}} (ty\ x\ e)_{r_x} \Downarrow E[r_x.x \mapsto v]}$$

$$\frac{}{E \vdash^{\text{env}}_{\text{init}} (\texttt{Component<\_>}\ x\ \{\epsilon\})_{r_x} \Downarrow E}$$

$$\frac{E \vdash^{\text{env}}_{\text{init}} p_{(r_x.x)} \Downarrow E' \qquad E' \vdash^{\text{env}}_{\text{init}} (\texttt{Component<\_>}\ x\ \{ps\})_{r_x} \Downarrow E''}{E \vdash^{\text{env}}_{\text{init}} (\texttt{Component<\_>}\ x\ \{p; ps\})_{r_x} \Downarrow E''}$$

$$\frac{}{\vdash^{\text{activ}}_{\text{init}} (\texttt{Component<d>}\ x\ \{ps\})_{r_x} \Downarrow \emptyset}$$

$$\frac{\forall i.\ \vdash^{\text{activ}}_{\text{init}} (ps_i)_{r_x.x} \Downarrow A_i}{\vdash^{\text{activ}}_{\text{init}} (\texttt{Component<a>}\ x\ \{ps\})_{r_x} \Downarrow (\bigcup_i A_i) \cup \{r_x.x\}}$$

System Initialization – Rules

$$\frac{\emptyset \vdash^{\text{env}}_{\text{init}} proc_\epsilon \Downarrow E \qquad \vdash^{\text{activ}}_{\text{init}} proc_\epsilon \Downarrow A}{\vdash_{\text{init}} proc \Downarrow E, A}$$

$$\frac{\emptyset, E \vdash_{\text{exp}} e \Downarrow v}{E \vdash^{\text{env}}_{\text{init}} (ty\ x\ e)_{r_x} \Downarrow E[r_x.x \mapsto v]}$$

$$\frac{}{E \vdash^{\text{env}}_{\text{init}} (\texttt{Component<\_>}\ x\ \{\epsilon\})_{r_x} \Downarrow E}$$

$$\frac{E \vdash^{\text{env}}_{\text{init}} p_{(r_x.x)} \Downarrow E' \qquad E' \vdash^{\text{env}}_{\text{init}} (\texttt{Component<\_>}\ x\ \{ps\})_{r_x} \Downarrow E''}{E \vdash^{\text{env}}_{\text{init}} (\texttt{Component<\_>}\ x\ \{p; ps\})_{r_x} \Downarrow E''}$$

$$\frac{}{\vdash^{\text{activ}}_{\text{init}} (\texttt{Component<d>}\ x\ \{ps\})_{r_x} \Downarrow \emptyset}$$

$$\frac{\forall i.\ \vdash^{\text{activ}}_{\text{init}} (ps_i)_{r_x.x} \Downarrow A_i}{\vdash^{\text{activ}}_{\text{init}} (\texttt{Component<a>}\ x\ \{ps\})_{r_x} \Downarrow (\bigcup_i A_i) \cup \{r_x.x\}}$$

Reaction to an Event

$$\frac{\begin{array}{c} T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \\ E, T, E', A' \vdash^{\text{proc}}_{\text{safe}} proc_{\epsilon} \\ E, A, T \vdash_{\text{update}} proc \Downarrow E', A' \end{array}}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

program

initial event

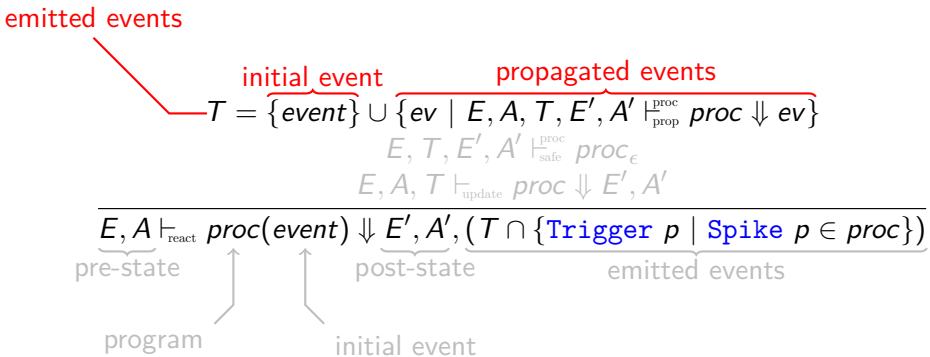
Reaction to an Event

$$\frac{\begin{array}{c} T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \\ E, T, E', A' \vdash^{\text{proc}}_{\text{safe}} proc_{\epsilon} \\ E, A, T \vdash_{\text{update}} proc \Downarrow E', A' \end{array}}{\underbrace{E, A}_{\text{pre-state}} \vdash_{\text{react}} proc(event) \Downarrow \underbrace{E', A'}_{\text{post-state}}, \underbrace{(T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}_{\text{emitted events}}}$$

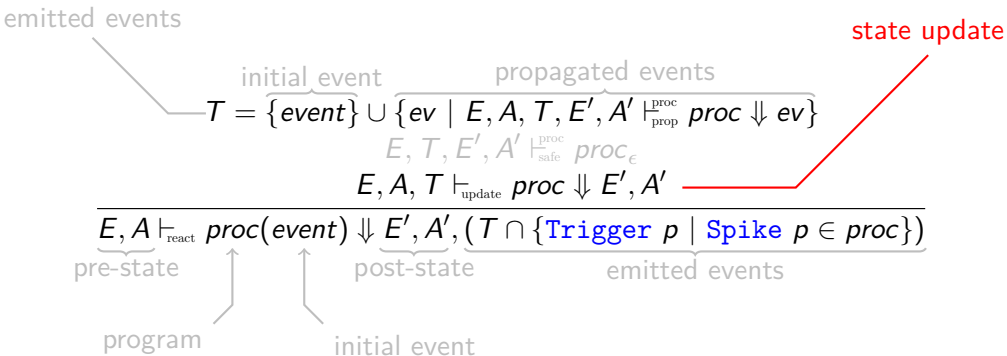
program

initial event

Reaction to an Event



Reaction to an Event



Propagation of Events – Example

$$\frac{\begin{array}{c} T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{proc}_{prop} proc \Downarrow ev\} \\ E, T, E', A' \vdash^{proc}_{sele} proc_\epsilon \\ E, A, T \vdash_{update} proc \Downarrow E', A' \end{array}}{E, A \vdash_{react} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$


```
Int count 3;
Spike zero; (count == 0) -> zero;

...

FillColor btn1 (150,150,150) {
  Rectangle r (0,0,100,f.height) { ... };
  r.press -> hg; hg: 255 =: green;
  r.release -> dhg; dhg: 150 =: green;
};
btn1.r.release -> dec; dec: last count - 1 =: count;
zero ->! btn1.r; (count > 0) -> btn1.r;
```

$$T = \left\{ \begin{array}{l} \texttt{Trigger btn1.r.release} \\ \\ \\ \\ \\ \\ \end{array} \right\}$$

Propagation of Events – Example

$$\frac{\begin{array}{c} T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{proc}_{prop} proc \Downarrow ev\} \\ E, T, E', A' \vdash^{proc}_{sole} proc_\epsilon \\ E, A, T \vdash_{update} proc \Downarrow E', A' \end{array}}{E, A \vdash_{react} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$


```
Int count 3;
Spike zero; (count == 0) -> zero;

...

FillColor btn1 (150,150,150) {
  Rectangle r (0,0,100,f.height) { ... };
  r.press -> hg; hg: 255 =: green;
  r.release -> dhg; dhg: 150 =: green;
};
btn1.r.release -> dec; dec: last count - 1 =: count;
zero ->! btn1.r; (count > 0) -> btn1.r;
```

$$T = \left\{ \begin{array}{l} \texttt{Trigger}\ \texttt{btn1.r.release} \\ \texttt{Trigger}\ \texttt{dhg} \end{array} \right\}$$

Propagation of Events – Example

$$\frac{\begin{array}{c} T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{proc}_{prop} proc \Downarrow ev\} \\ E, T, E', A' \vdash^{proc}_{sole} proc_\epsilon \\ E, A, T \vdash_{update} proc \Downarrow E', A' \end{array}}{E, A \vdash_{react} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$


```
Int count 3;
Spike zero; (count == 0) -> zero;

...

FillColor btn1 (150,150,150) {
  Rectangle r (0,0,100,f.height) { ... };
  r.press -> hg; hg: 255 =: green;
  r.release -> dhg; dhg: 150 =: green;
};
btn1.r.release -> dec; dec: last count - 1 =: count;
zero ->! btn1.r; (count > 0) -> btn1.r;
```

$$T = \left\{ \begin{array}{l} \texttt{Trigger}\ \texttt{btn1.r.release} \\ \texttt{Trigger}\ \texttt{dhg} \\ \texttt{Assign}\ 150\ \texttt{btn1.green} \end{array} \right\}$$

Propagation of Events – Example

$$\frac{T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{proc}_{prop} proc \Downarrow ev\} \quad E, T, E', A' \vdash^{proc}_{sele} proc_\epsilon \quad E, A, T \vdash_{update} proc \Downarrow E', A'}{E, A \vdash_{react} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$


```
Int count 3;
Spike zero; (count == 0) -> zero;

...

FillColor btn1 (150,150,150) {
  Rectangle r (0,0,100,f.height) { ... };
  r.press -> hg; hg: 255 =: green;
  r.release -> dhg; dhg: 150 =: green;
};
btn1.r.release -> dec; dec: last count - 1 =: count;
zero ->! btn1.r; (count > 0) -> btn1.r;
```

$$T = \left\{ \begin{array}{l} \texttt{Trigger btn1.r.release} \\ \texttt{Trigger dhg} \\ \texttt{Assign 150 btn1.green} \\ \texttt{Trigger dec} \\ \texttt{Assign 1 count} \end{array} \right\}$$

Propagation of Events – Example

$$\frac{\begin{array}{c} T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{proc}_{prop} proc \Downarrow ev\} \\ E, T, E', A' \vdash^{proc}_{sele} proc_\epsilon \\ E, A, T \vdash_{update} proc \Downarrow E', A' \end{array}}{E, A \vdash_{react} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$


```
Int count 3;
Spike zero; (count == 0) -> zero;

...

FillColor btn1 (150,150,150) {
  Rectangle r (0,0,100,f.height) { ... };
  r.press -> hg; hg: 255 =: green;
  r.release -> dhg; dhg: 150 =: green;
};
btn1.r.release -> dec; dec: last count - 1 =: count;
zero ->! btn1.r; (count > 0) -> btn1.r;
```

$$T = \left\{ \begin{array}{l} \texttt{Trigger btn1.r.release} \\ \texttt{Trigger dhg} \\ \texttt{Assign 150 btn1.green} \\ \texttt{Trigger dec} \\ \texttt{Assign 1 count} \end{array} \right\}$$

Propagation of Events – Assignments

$$\frac{T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \quad E, T, E', A' \vdash^{\text{proc}}_{\text{sole}} proc_\epsilon \quad E, A, T \vdash_{\text{update}} proc \Downarrow E', A'}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

∃ assignment process rooted at $r_x.x$ in the program

$$\frac{proc(r_x.x) = \lfloor x : e =: r_y.y \rfloor \quad r_x \in A' \quad r_y \in A' \quad \texttt{Trigger}\ r_x.x \in T \quad E, E' \vdash_{\text{exp}} e \Downarrow v}{E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow \texttt{Assign}\ v\ r_y.y}$$

Propagation of Events – Assignments

$$\frac{T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \quad E, T, E', A' \vdash^{\text{proc}}_{\text{sole}} proc_\epsilon \quad E, A, T \vdash_{\text{update}} proc \Downarrow E', A'}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

$\exists$ assignment process rooted at $r_x.x$ in the program

$r_x.x$ is triggered

$$\frac{r_x \in A' \qquad r_y \in A' \qquad proc(r_x.x) = \lfloor x\texttt{:}\ e\ \texttt{=:}\ r_y.y \rfloor \qquad \texttt{Trigger}\ r_x.x \in T \qquad E, E' \vdash_{\text{exp}} e \Downarrow v}{E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow \texttt{Assign}\ \mathsf{v}\ r_y.y}$$

Propagation of Events – Assignments

$$\frac{T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \quad E, T, E', A' \vdash^{\text{proc}}_{\text{sole}} proc_\epsilon \quad E, A, T \vdash_{\text{update}} proc \Downarrow E', A'}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

$\exists$ assignment process rooted at $r_x.x$ in the program

assignment and target parents are active

$r_x.x$ is triggered

$$\frac{r_x \in A' \qquad r_y \in A' \qquad proc(r_x.x) = \lfloor x\texttt{:}\ e\ \texttt{=:}\ r_y.y \rfloor \qquad \texttt{Trigger}\ r_x.x \in T \qquad E, E' \vdash_{\text{exp}} e \Downarrow v}{E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow \texttt{Assign}\ v\ r_y.y}$$

Propagation of Events – Assignments

$$\frac{T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \quad E, T, E', A' \vdash^{\text{proc}}_{\text{sele}} proc_\epsilon \quad E, A, T \vdash_{\text{update}} proc \Downarrow E', A'}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

$\exists$ assignment process rooted at $r_x.x$ in the program

assignment and target parents are active

$r_x.x$ is triggered

expression evaluates to v

$$\frac{r_x \in A' \qquad r_y \in A' \qquad proc(r_x.x) = \lfloor x\texttt{: } e \texttt{=: } r_y.y \rfloor \qquad \texttt{Trigger}\ r_x.x \in T \qquad E, E' \vdash_{\text{exp}} e \Downarrow v}{E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow \texttt{Assign}\ v\ r_y.y}$$

Propagation of Events – Bindings

$$\frac{T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \quad E, T, E', A' \vdash^{\text{proc}}_{\text{sole}} proc_\epsilon \quad E, A, T \vdash_{\text{update}} proc \Downarrow E', A'}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

$$\frac{proc(r_x.x) = \lfloor x\texttt{: }lhs\ \texttt{-><\_>}\ rhs \rfloor \qquad r_x.x \in A' \qquad E, T, E' \vdash^{\text{lhs}}_{\text{prop}} lhs \qquad A, A' \vdash^{\text{rhs}}_{\text{prop}} rhs \Downarrow ev}{E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev}$$

$$\frac{\texttt{Trigger}\ x \in T}{E, T, E' \vdash^{\text{lhs}}_{\text{prop}} \texttt{T?}(x)}$$

ex: `zero ->! btn1.r`

$$\frac{x \in \text{free}(e) \qquad \texttt{Assign v}\ x \in T \qquad E, E' \vdash_{\text{exp}} e \Downarrow \texttt{true}}{E, T, E' \vdash^{\text{lhs}}_{\text{prop}} (e)?}$$

ex: `(count == 0) -> zero`

$$\frac{r_x \in A'}{A, A' \vdash^{\text{rhs}}_{\text{prop}} \texttt{T!}(r_x.x) \Downarrow \texttt{Trigger}\ r_x.x}$$

ex: `(count == 0) -> zero`

$$\frac{r_x \in A' \qquad r_x.x \notin A}{A, A' \vdash^{\text{rhs}}_{\text{prop}} \texttt{A!}(r_x.x) \Downarrow \texttt{Activate}\ r_x.x}$$

ex: `(count > 0) -> btn1.r`

Propagation of Events – Bindings

$$\frac{T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \quad E, T, E', A' \vdash^{\text{proc}}_{\text{sole}} proc_\epsilon \quad E, A, T \vdash_{\text{update}} proc \Downarrow E', A'}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

$$\frac{proc(r_x.x) = \lfloor x\texttt{: } lhs\ \texttt{-><\_>}\ rhs \rfloor \qquad r_x.x \in A' \qquad E, T, E' \vdash^{\text{lhs}}_{\text{prop}} lhs \qquad A, A' \vdash^{\text{rhs}}_{\text{prop}} rhs \Downarrow ev}{E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev}$$

$$\frac{\texttt{Trigger}\ x \in T}{E, T, E' \vdash^{\text{lhs}}_{\text{prop}} \texttt{T?(}x\texttt{)}}$$

ex: `zero ->! btn1.r`

$$\frac{x \in \text{free}(e) \qquad \texttt{Assign v}\ x \in T \qquad E, E' \vdash_{\text{exp}} e \Downarrow \texttt{true}}{E, T, E' \vdash^{\text{lhs}}_{\text{prop}} \texttt{(}e\texttt{)?}}$$

ex: `(count == 0) -> zero`

$$\frac{r_x \in A'}{A, A' \vdash^{\text{rhs}}_{\text{prop}} \texttt{T!(}r_x.x\texttt{)} \Downarrow \texttt{Trigger}\ r_x.x}$$

ex: `(count == 0) -> zero`

$$\frac{r_x \in A' \qquad r_x.x \notin A}{A, A' \vdash^{\text{rhs}}_{\text{prop}} \texttt{A!(}r_x.x\texttt{)} \Downarrow \texttt{Activate}\ r_x.x}$$

ex: `(count > 0) -> btn1.r`

Propagation of Events – Bindings

$$\frac{\begin{array}{c} T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \\ E, T, E', A' \vdash^{\text{proc}}_{\text{sele}} proc_\epsilon \\ E, A, T \vdash_{\text{update}} proc \Downarrow E', A' \end{array}}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

$$\frac{proc(r_x.x) = \lfloor x\texttt{: }lhs\ \texttt{->\!<\_>}\ rhs \rfloor \qquad r_x.x \in A' \qquad E, T, E' \vdash^{\text{lhs}}_{\text{prop}} lhs \qquad A, A' \vdash^{\text{rhs}}_{\text{prop}} rhs \Downarrow ev}{E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev}$$

$$\frac{\texttt{Trigger}\ x \in T}{E, T, E' \vdash^{\text{lhs}}_{\text{prop}} \texttt{T?}(x)} \qquad \frac{x \in \text{free}(e) \qquad \texttt{Assign v}\ x \in T \qquad E, E' \vdash_{\text{exp}} e \Downarrow \texttt{true}}{E, T, E' \vdash^{\text{lhs}}_{\text{prop}} (e)\texttt{?}}$$

ex: `zero ->! btn1.r` ex: `(count == 0) -> zero`

$$\frac{r_x \in A'}{A, A' \vdash^{\text{rhs}}_{\text{prop}} \texttt{T!}(r_x.x) \Downarrow \texttt{Trigger}\ r_x.x}$$

ex: `(count == 0) -> zero`

$$\frac{r_x \in A' \qquad r_x.x \notin A}{A, A' \vdash^{\text{rhs}}_{\text{prop}} \texttt{A!}(r_x.x) \Downarrow \texttt{Activate}\ r_x.x}$$

ex: `(count > 0) -> btn1.r`

Propagation of Events – Children Activation

$$\frac{T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \quad E, T, E', A' \vdash^{\text{proc}}_{\text{sele}} proc_{\epsilon} \quad E, A, T \vdash_{\text{update}} proc \Downarrow E', A'}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

$$\frac{proc(r_x.x) = \lfloor \texttt{Component<a>}\ x\ \{ps\} \rfloor \quad \texttt{Activate}\ r_x.x \in T \quad \vdash^{\text{activ}}_{\text{prop}} (ps_i)_{r_x.x} \Downarrow \texttt{Activate}\ y}{E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow \texttt{Activate}\ y}$$

$$\frac{}{\vdash^{\text{activ}}_{\text{prop}} (x:\ lhs\ \texttt{-><a>}\ rhs)_{r_x} \Downarrow \texttt{Activate}\ r_x}$$

$$\frac{}{\vdash^{\text{activ}}_{\text{prop}} (\texttt{Component<a>}\ x\ \{ps\})_{r_x} \Downarrow \texttt{Activate}\ r_x.x}$$

Propagation of Events – Children Activation

$$\frac{\begin{array}{c} T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \\ E, T, E', A' \vdash^{\text{proc}}_{\text{sele}} proc_{\epsilon} \\ E, A, T \vdash_{\text{update}} proc \Downarrow E', A' \end{array}}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

$$\frac{proc(r_x.x) = \lfloor \texttt{Component<a>}\ x\ \{ps\} \rfloor \qquad \texttt{Activate}\ r_x.x \in T \qquad \vdash^{\text{activ}}_{\text{prop}} (ps_i)_{r_x.x} \Downarrow \texttt{Activate}\ y}{E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow \texttt{Activate}\ y}$$

$$\frac{}{\vdash^{\text{activ}}_{\text{prop}} (x\texttt{:}\ lhs\ \texttt{-><a>}\ rhs)_{r_x} \Downarrow \texttt{Activate}\ r_x}$$

$$\frac{}{\vdash^{\text{activ}}_{\text{prop}} (\texttt{Component<a>}\ x\ \{ps\})_{r_x} \Downarrow \texttt{Activate}\ r_x.x}$$

Propagation of Events – Children Activation

$$\frac{T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \quad E, T, E', A' \vdash^{\text{proc}}_{\text{sele}} proc_\epsilon \quad E, A, T \vdash_{\text{update}} proc \Downarrow E', A'}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

$$\frac{proc(r_x.x) = \lfloor \texttt{Component<a>}\ x\ \{ps\} \rfloor \quad \texttt{Activate}\ r_x.x \in T \quad \vdash^{\text{activ}}_{\text{prop}} (ps_i)_{r_x.x} \Downarrow \texttt{Activate}\ y}{E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow \texttt{Activate}\ y}$$

$$\frac{}{\vdash^{\text{activ}}_{\text{prop}} (x\texttt{:}\ lhs\ \texttt{-><a>}\ rhs)_{r_x} \Downarrow \texttt{Activate}\ r_x}$$

$$\frac{}{\vdash^{\text{activ}}_{\text{prop}} (\texttt{Component<a>}\ x\ \{ps\})_{r_x} \Downarrow \texttt{Activate}\ r_x.x}$$

and similar rules for children deactivation

State Update – Example

$$\frac{\begin{array}{c} T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{proc}_{prop} proc \Downarrow ev\} \\ E, T, E', A' \vdash^{proc}_{sele} proc_\epsilon \\ E, A, T \vdash_{update} proc \Downarrow E', A' \end{array}}{E, A \vdash_{react} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$


```
Int count 3;
Spike zero; (count == 0) -> zero;

...

FillColor btn1 (150,150,150) {
  Rectangle r (0,0,100,f.height) { ... };
  r.press -> hg; hg: 255 =: green;
  r.release -> dhg; dhg: 150 =: green;
};
btn1.r.release -> dec; dec: last count - 1 =: count;
zero ->! btn1.r; (count > 0) -> btn1.r;
```

$$T = \left\{ \begin{array}{l} \texttt{Trigger}\ \texttt{btn1.r.release} \\ \texttt{Trigger}\ \texttt{dhg} \\ \texttt{Assign}\ \texttt{150}\ \texttt{btn1.green} \\ \texttt{Trigger}\ \texttt{dec} \\ \texttt{Assign}\ \texttt{0}\ \texttt{count} \\ \texttt{Trigger}\ \texttt{zero} \\ \texttt{Deactivate}\ \texttt{btn1.r...} \end{array} \right\}$$

State Update – Example

$$\frac{\begin{array}{c} T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{proc}_{prop} proc \Downarrow ev\} \\ E, T, E', A' \vdash^{proc}_{sele} proc_\epsilon \\ E, A, T \vdash_{update} proc \Downarrow E', A' \end{array}}{E, A \vdash_{react} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

$$T = \left\{ \begin{array}{l} \texttt{Trigger btn1.r.release} \\ \texttt{Trigger dhg} \\ \texttt{Assign 150 btn1.green} \\ \texttt{Trigger dec} \\ \texttt{Assign 0 count} \\ \texttt{Trigger zero} \\ \texttt{Deactivate btn1.r...} \end{array} \right\}$$

```
Int count 3;
Spike zero; (count == 0) -> zero;

...

FillColor btn1 (150,150,150) {
  Rectangle r (0,0,100,f.height) { ... };
  r.press -> hg; hg: 255 =: green;
  r.release -> dhg; dhg: 150 =: green;
};
btn1.r.release -> dec; dec: last count - 1 =: count;
zero ->! btn1.r; (count > 0) -> btn1.r;
```

$$E = \{\texttt{count} \mapsto 1; \texttt{green} \mapsto 255; \texttt{red} \mapsto 150; ...\}$$
$$E' = \{\texttt{count} \mapsto 0; \texttt{green} \mapsto 150; \texttt{red} \mapsto 150; ...\}$$

State Update – Example

$$\frac{T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{proc}_{prop} proc \Downarrow ev\} \quad E, T, E', A' \vdash^{proc}_{sele} proc_\epsilon \quad E, A, T \vdash_{update} proc \Downarrow E', A'}{E, A \vdash_{react} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

$$T = \left\{ \begin{array}{l} \texttt{Trigger btn1.r.release} \\ \texttt{Trigger dhg} \\ \texttt{Assign 150 btn1.green} \\ \texttt{Trigger dec} \\ \texttt{Assign 0 count} \\ \texttt{Trigger zero} \\ \texttt{Deactivate btn1.r...} \end{array} \right\}$$

```
Int count 3;
Spike zero; (count == 0) -> zero;

...

FillColor btn1 (150,150,150) {
  Rectangle r (0,0,100,f.height) { ... };
  r.press -> hg; hg: 255 =: green;
  r.release -> dhg; dhg: 150 =: green;
};
btn1.r.release -> dec; dec: last count - 1 =: count;
zero ->! btn1.r; (count > 0) -> btn1.r;
```

$$E = \{\texttt{count} \mapsto 1; \texttt{green} \mapsto 255; \texttt{red} \mapsto 150; ...\}$$
$$E' = \{\texttt{count} \mapsto 0; \texttt{green} \mapsto 150; \texttt{red} \mapsto 150; ...\}$$

$$A = \{\texttt{btn1}; \texttt{btn1.r}; ...\}$$
$$A = \{\texttt{btn1}; ...\}$$

State Update – Rule

$$\frac{T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \quad E, T, E', A' \vdash^{\text{proc}}_{\text{sole}} proc_\epsilon \quad E, A, T \vdash_{\text{update}} proc \Downarrow E', A'}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

$$\frac{\begin{array}{c}\forall p\ v,\ \texttt{Assign}\ v\ p \in T \implies E'(p) = \lfloor v \rfloor \\ \forall p,\ (\forall v,\ \texttt{Assign}\ v\ p \notin T) \implies E'(p) = E(p) \\ \forall p,\ \texttt{Activate}\ p \in T \implies p \in A' \\ \forall p,\ \texttt{Deactivate}\ p \in T \implies p \notin A' \\ \forall p,\ (\texttt{Activate}\ p \notin T \wedge \texttt{Deactivate}\ p \notin T) \implies (p \in A' \iff p \in A)\end{array}}{E, A, T \vdash_{\text{update}} proc \Downarrow E', A'}$$

State Update – Rule

$$\frac{\begin{array}{c} T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{proc}_{prop} proc \Downarrow ev\} \\ E, T, E', A' \vdash^{proc}_{sole} proc_\epsilon \\ E, A, T \vdash_{update} proc \Downarrow E', A' \end{array}}{E, A \vdash_{react} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

$$\frac{\begin{array}{c} \forall p\ v,\ \texttt{Assign v}\ p \in T \implies E'(p) = \lfloor v \rfloor \\ \forall p,\ (\forall v,\ \texttt{Assign v}\ p \notin T) \implies E'(p) = E(p) \\ \forall p,\ \texttt{Activate}\ p \in T \implies p \in A' \\ \forall p,\ \texttt{Deactivate}\ p \in T \implies p \notin A' \\ \forall p,\ (\texttt{Activate}\ p \notin T \wedge \texttt{Deactivate}\ p \notin T) \implies (p \in A' \iff p \in A) \end{array}}{E, A, T \vdash_{update} proc \Downarrow E', A'}$$

State Update – Rule

$$\frac{\begin{array}{c} T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{proc}_{prop} proc \Downarrow ev\} \\ E, T, E', A' \vdash^{proc}_{sole} proc_\epsilon \\ E, A, T \vdash_{update} proc \Downarrow E', A' \end{array}}{E, A \vdash_{react} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

$$\forall p\ v,\ \texttt{Assign}\ v\ p \in T \implies E'(p) = \lfloor v \rfloor$$
$$\forall p,\ (\forall v,\ \texttt{Assign}\ v\ p \notin T) \implies E'(p) = E(p)$$
$$\forall p,\ \texttt{Activate}\ p \in T \implies p \in A'$$
$$\forall p,\ \texttt{Deactivate}\ p \in T \implies p \notin A'$$
$$\forall p,\ (\texttt{Activate}\ p \notin T \wedge \texttt{Deactivate}\ p \notin T) \implies (p \in A' \iff p \in A)$$
$$\overline{E, A, T \vdash_{\text{update}} proc \Downarrow E', A'}$$

State Update – Rule

$$\frac{\begin{array}{c} T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \\ E, T, E', A' \vdash^{\text{proc}}_{\text{calc}} proc_\epsilon \\ E, A, T \vdash_{\text{update}} proc \Downarrow E', A' \end{array}}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

$$\frac{\begin{array}{c} \forall p\ v,\ \texttt{Assign}\ v\ p \in T \implies E'(p) = \lfloor v \rfloor \\ \forall p,\ (\forall v,\ \texttt{Assign}\ v\ p \notin T) \implies E'(p) = E(p) \\ \forall p,\ \texttt{Activate}\ p \in T \implies p \in A' \\ \forall p,\ \texttt{Deactivate}\ p \in T \implies p \notin A' \\ \forall p,\ (\texttt{Activate}\ p \notin T \wedge \texttt{Deactivate}\ p \notin T) \implies (p \in A' \iff p \in A) \end{array}}{E, A, T \vdash_{\text{update}} proc \Downarrow E', A'}$$

Some contradictory programs:

```
Int x 0;
a -> a1; a1: 42 =: x;
a -> a2; a2: 84 =: x;
```

```
Component c { ... }
a -> c;
a ->! c;
```

Rejected by static analysis

Safe Reactions – Issue

```
a: 0 =: y;
(x/y > 10) -> t
```

$$T = \left\{ \begin{array}{c} \texttt{Trigger a} \\ \texttt{Assign y 0} \\ ??? \end{array} \right\}$$

$$\frac{x \in \text{free}(e) \qquad \texttt{Assign v } x \in T \qquad E, E' \vdash_{\text{exp}} e \Downarrow \texttt{true}}{E, T, E' \vdash^{\text{lhs}}_{\text{prop}} (e)?}$$

$$\frac{proc(r_x.x) = \lfloor x\texttt{: } lhs \texttt{ -><_> } rhs \rfloor \qquad r_x.x \in A' \qquad E, T, E' \vdash^{\text{lhs}}_{\text{prop}} lhs \qquad A, A' \vdash^{\text{rhs}}_{\text{prop}} rhs \Downarrow ev}{E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev}$$

$$T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\}$$

$$\frac{E, A, T \vdash_{\text{update}} proc \Downarrow E', A'}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger } p \mid \texttt{Spike } p \in proc\})}$$

Safe Reactions – Issue

```
a: 0 =: y;
(x/y > 10) -> t
```

$$T = \left\{ \begin{array}{c} \texttt{Trigger a} \\ \texttt{Assign y 0} \\ ??? \end{array} \right\}$$

$$\frac{x \in \mathsf{free}(e) \qquad \texttt{Assign v } x \in T \qquad \cancel{E, E' \vdash_{\mathrm{exp}} e \Downarrow \texttt{true}}}{E, T, E' \vdash^{\mathrm{lhs}}_{\mathrm{prop}} (e)?}$$

$$\frac{proc(r_x.x) = \lfloor x\texttt{: } lhs \texttt{ -><_> } rhs \rfloor \qquad r_x.x \in A' \qquad E, T, E' \vdash^{\mathrm{lhs}}_{\mathrm{prop}} lhs \qquad A, A' \vdash^{\mathrm{rhs}}_{\mathrm{prop}} rhs \Downarrow ev}{E, A, T, E', A' \vdash^{\mathrm{proc}}_{\mathrm{prop}} proc \Downarrow ev}$$

$$T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\mathrm{proc}}_{\mathrm{prop}} proc \Downarrow ev\}$$

$$\frac{E, A, T \vdash_{\mathrm{update}} proc \Downarrow E', A'}{E, A \vdash_{\mathrm{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger } p \mid \texttt{Spike } p \in proc\})}$$

Safe Reactions – Issue

```
a: 0 =: y;
(x/y > 10) -> t
```

$$T = \left\{ \begin{array}{c} \texttt{Trigger a} \\ \texttt{Assign y 0} \\ ??? \end{array} \right\}$$

$$\frac{x \in \text{free}(e) \qquad \texttt{Assign v } x \in T \qquad \cancel{E, E' \vdash_{\text{exp}} e \Downarrow \texttt{true}}}{E, T, E' \vdash^{\text{lhs}}_{\text{prop}} (e)?}$$

$$\frac{proc(r_x.x) = \lfloor x\texttt{: } lhs \texttt{ -><_> } rhs \rfloor \qquad r_x.x \in A' \qquad \cancel{E, T, E' \vdash^{\text{lhs}}_{\text{prop}} lhs} \qquad A, A' \vdash^{\text{rhs}}_{\text{prop}} rhs \Downarrow ev}{E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev}$$

$$T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\}$$

$$\frac{E, A, T \vdash_{\text{update}} proc \Downarrow E', A'}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger } p \mid \texttt{Spike } p \in proc\})}$$

Safe Reactions – Issue

```
a: 0 =: y;
(x/y > 10) -> t
```

$$T = \left\{ \begin{array}{c} \texttt{Trigger a} \\ \texttt{Assign y 0} \\ ??? \end{array} \right\}$$

$$\frac{x \in \mathsf{free}(e) \qquad \texttt{Assign v } x \in T \qquad E, E' \vdash_{\mathrm{exp}} e \Downarrow \texttt{true}}{E, T, E' \vdash^{\mathrm{lhs}}_{\mathrm{prop}} (e)?}$$

$$\frac{proc(r_x.x) = \lfloor x\texttt{: } lhs \texttt{ -><_> } rhs \rfloor \qquad r_x.x \in A' \qquad E, T, E' \vdash^{\mathrm{lhs}}_{\mathrm{prop}} lhs \qquad A, A' \vdash^{\mathrm{rhs}}_{\mathrm{prop}} rhs \Downarrow ev}{E, A, T, E', A' \vdash^{\mathrm{proc}}_{\mathrm{prop}} proc \Downarrow ev}$$

$$T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\mathrm{proc}}_{\mathrm{prop}} proc \Downarrow ev\}$$

$$\frac{E, A, T \vdash_{\mathrm{update}} proc \Downarrow E', A'}{E, A \vdash_{\mathrm{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger } p \mid \texttt{Spike } p \in proc\})}$$

Safe Reactions – Issue

```
a: 0 =: y;
(x/y > 10) -> t
```

$$T = \left\{ \begin{array}{l} \texttt{Trigger a} \\ \texttt{Assign y 0} \end{array} \right\}$$

$$\frac{x \in \mathrm{free}(e) \qquad \texttt{Assign v } x \in T \qquad E, E' \vdash_{\mathrm{exp}} e \Downarrow \texttt{true}}{E, T, E' \vdash^{\mathrm{lhs}}_{\mathrm{prop}} (e)?}$$

$$\frac{\mathit{proc}(r_x.x) = \lfloor x\texttt{: } \mathit{lhs} \texttt{ -><_> } \mathit{rhs} \rfloor \qquad r_x.x \in A' \qquad E, T, E' \vdash^{\mathrm{lhs}}_{\mathrm{prop}} \mathit{lhs} \qquad A, A' \vdash^{\mathrm{rhs}}_{\mathrm{prop}} \mathit{rhs} \Downarrow \mathit{ev}}{E, A, T, E', A' \vdash^{\mathrm{proc}}_{\mathrm{prop}} \mathit{proc} \Downarrow \mathit{ev}}$$

$$T = \{\mathit{event}\} \cup \{\mathit{ev} \mid E, A, T, E', A' \vdash^{\mathrm{proc}}_{\mathrm{prop}} \mathit{proc} \Downarrow \mathit{ev}\}$$

only says `t` is not triggered

$$\frac{E, A, T \vdash_{\mathrm{update}} \mathit{proc} \Downarrow E', A'}{E, A \vdash_{\mathrm{react}} \mathit{proc}(\mathit{event}) \Downarrow E', A', (T \cap \{\texttt{Trigger } p \mid \texttt{Spike } p \in \mathit{proc}\})}$$

Safe Reactions – Issue

```
a: 0 =: y;
(x/y > 10) -> t
```

$$T = \left\{ \begin{array}{l} \texttt{Trigger a} \\ \texttt{Assign y 0} \end{array} \right\}$$

$$\frac{x \in \text{free}(e) \qquad \texttt{Assign v } x \in T \qquad E, E' \vdash_{\text{exp}} e \Downarrow \texttt{true}}{E, T, E' \vdash^{\text{lhs}}_{\text{prop}} (e)?}$$

$$\frac{proc(r_x.x) = \lfloor x\texttt{: } lhs \texttt{ ->\<\_\> } rhs \rfloor \qquad r_x.x \in A' \qquad E, T, E' \vdash^{\text{lhs}}_{\text{prop}} lhs \qquad A, A' \vdash^{\text{rhs}}_{\text{prop}} rhs \Downarrow ev}{E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev}$$

$$\frac{\begin{array}{c} T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \\ E, T, E', A' \vdash^{\text{proc}}_{\text{safe}} proc_{\epsilon} \\ E, A, T \vdash_{\text{update}} proc \Downarrow E', A' \end{array}}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger } p \mid \texttt{Spike } p \in proc\})}$$

Safe Reactions – Rules

$$\frac{T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \quad E, T, E', A' \vdash^{\text{proc}}_{\text{safe}} proc_\epsilon \quad E, A, T \vdash_{\text{update}} proc \Downarrow E', A'}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

$$\frac{\forall x \in \text{free}(e),\ \forall v,\ \texttt{Assign}\ \texttt{v}\ x \notin T}{E, T, E' \vdash^{\text{lhs}}_{\text{safe}} (e)?} \qquad \frac{E, E' \vdash_{\text{exp}} e \Downarrow v \qquad v \in \{\texttt{true}, \texttt{false}\}}{E, T, E' \vdash^{\text{lhs}}_{\text{safe}} (e)?}$$

$$\frac{E, T, E' \vdash^{\text{lhs}}_{\text{safe}} lhs}{E, T, E', A' \vdash^{\text{proc}}_{\text{safe}} x\texttt{:}\ lhs\ \texttt{->}\texttt{<\_>}\ rhs}$$

$$\frac{\texttt{Trigger}\ r_x.x \in T \implies E, E' \vdash_{\text{exp}} e \Downarrow v}{E, T, E', A' \vdash^{\text{proc}}_{\text{safe}} (x\texttt{:}\ e\ \texttt{=:}\ y)_{r_x}}$$

and other syntax-directed rules...

Safe Reactions – Rules

$$\frac{T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \quad E, T, E', A' \vdash^{\text{proc}}_{\text{safe}} proc_\epsilon \quad E, A, T \vdash_{\text{update}} proc \Downarrow E', A'}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

```
a: 0 =: y;
(x/y > 10) -> t
```

$$\frac{\forall x \in \text{free}(e),\ \forall v,\ \texttt{Assign}\ v\ x \notin T}{E, T, E' \vdash^{\text{lhs}}_{\text{safe}} (e)?}$$

$$\frac{E, E' \vdash_{\text{exp}} e \Downarrow v \qquad v \in \{\texttt{true}, \texttt{false}\}}{E, T, E' \vdash^{\text{lhs}}_{\text{safe}} (e)?}$$

$$\frac{E, T, E' \vdash^{\text{lhs}}_{\text{safe}} lhs}{E, T, E', A' \vdash^{\text{proc}}_{\text{safe}} x\texttt{:}\ lhs\ \texttt{->}\texttt{<\_>}\ rhs}$$

$$\frac{\texttt{Trigger}\ r_x.x \in T \implies E, E' \vdash_{\text{exp}} e \Downarrow v}{E, T, E', A' \vdash^{\text{proc}}_{\text{safe}} (x\texttt{:}\ e\ \texttt{=:}\ y)_{r_x}}$$

and other syntax-directed rules...

Safe Reactions – Rules

```
a: 0 =: y;
(x/y > 10) -> t
```

$$\frac{T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \quad E, T, E', A' \vdash^{\text{proc}}_{\text{spike}} proc \quad E, A, T \vdash_{\text{spike}} proc \Downarrow E', A'}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

$$\frac{\forall x \in \text{free}(e),\ \forall v,\ \texttt{Assign}\ x\ v \notin T}{E, T, E' \vdash^{\text{lhs}}_{\text{safe}} (e)?}$$

$$\frac{E, E' \vdash_{\text{exp}} e \Downarrow v \qquad v \in \{\texttt{true}, \texttt{false}\}}{E, T, E' \vdash^{\text{lhs}}_{\text{safe}} (e)?}$$

$$\frac{E, T, E' \vdash^{\text{lhs}}_{\text{safe}} lhs}{E, T, E', A' \vdash^{\text{proc}}_{\text{safe}} x\texttt{:}\ lhs\ \texttt{->}\texttt{<\_>}\ rhs}$$

$$\frac{\texttt{Trigger}\ r_x.x \in T \implies E, E' \vdash_{\text{exp}} e \Downarrow v}{E, T, E', A' \vdash^{\text{proc}}_{\text{safe}} (x\texttt{:}\ e\ \texttt{=:}\ y)_{r_x}}$$

and other syntax-directed rules...

Safe Reactions – Rules

$$\frac{T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{\text{proc}}_{\text{prop}} proc \Downarrow ev\} \quad E, T, E', A' \vdash^{\text{proc}}_{\text{safe}} proc_\epsilon \quad E, A, T \vdash_{\text{update}} proc \Downarrow E', A'}{E, A \vdash_{\text{react}} proc(event) \Downarrow E', A', (T \cap \{\texttt{Trigger}\ p \mid \texttt{Spike}\ p \in proc\})}$$

$$\frac{\forall x \in \text{free}(e),\ \forall v,\ \texttt{Assign}\ v\ x \notin T}{E, T, E' \vdash^{\text{lhs}}_{\text{safe}} (e)?} \qquad \frac{E, E' \vdash_{\text{exp}} e \Downarrow v \qquad v \in \{\texttt{true}, \texttt{false}\}}{E, T, E' \vdash^{\text{lhs}}_{\text{safe}} (e)?}$$

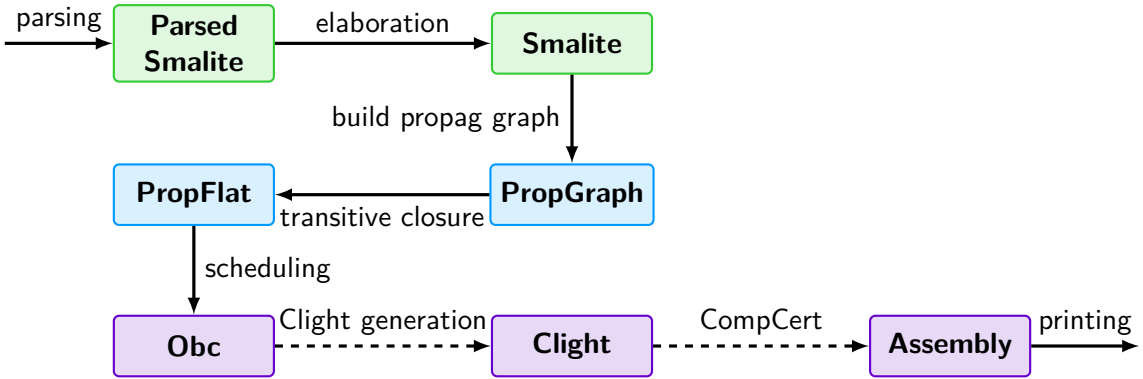
$$\frac{E, T, E' \vdash^{\text{lhs}}_{\text{safe}} lhs}{E, T, E', A' \vdash^{\text{proc}}_{\text{safe}} x\texttt{:}\ lhs\ \texttt{-><\_>}\ rhs}$$

$$\frac{\texttt{Trigger}\ r_x.x \in T \implies E, E' \vdash_{\text{exp}} e \Downarrow v}{E, T, E', A' \vdash^{\text{proc}}_{\text{safe}} (x\texttt{:}\ e\ \texttt{=:}\ y)_{r_x}}$$

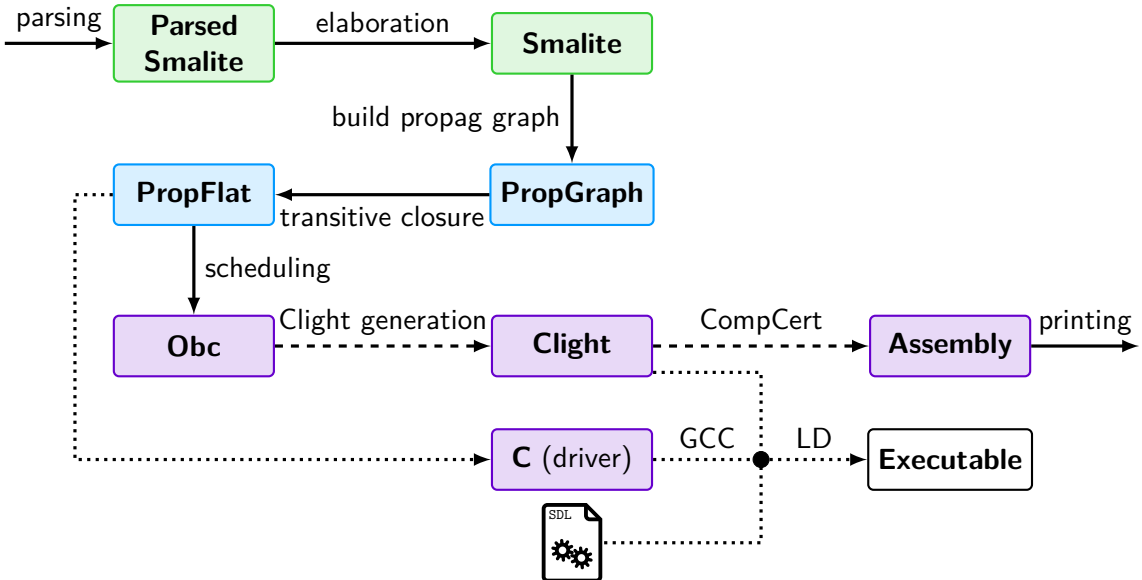
and other syntax-directed rules...

Future Work – A verified compiler for Smala

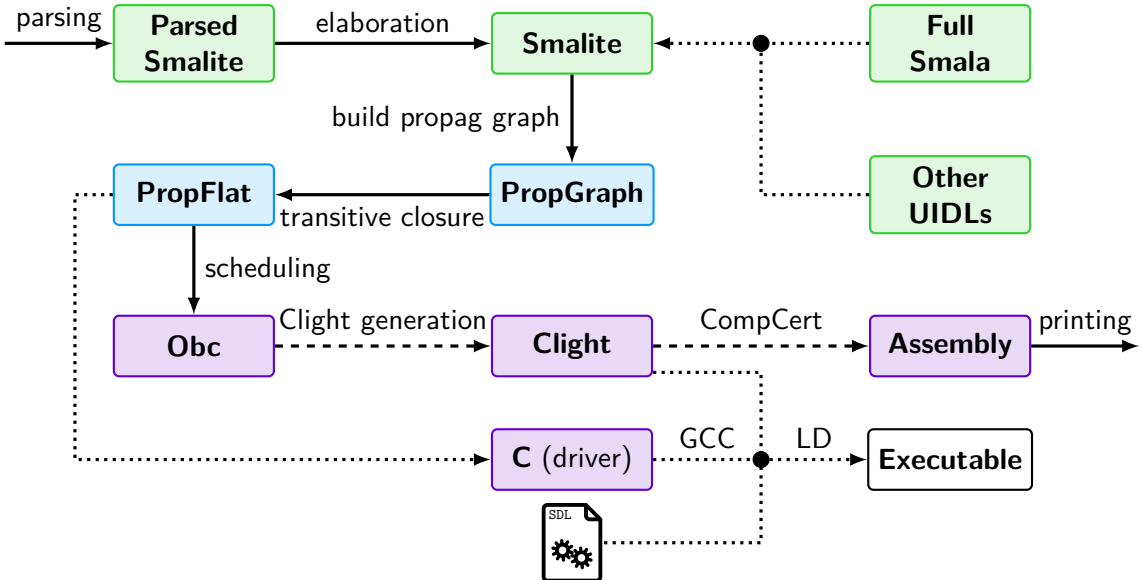
A prototype compiler for Smalite



A prototype compiler for Smalite



A prototype compiler for Smalite



Compiling a Smalite program

```
Component root {
  Int count 3;
  Spike zero; (count == 0) -> zero;
  Frame f ("ICE␣2025", 300, 50) {
    Font _ ("arial.ttf", 20) {
      FillColor btn1 (150,150,150) { ...
        r.release -> dhg;
        dhg: 150 =: green;
      };
      btn1.r.release -> dec;
      dec: last count - 1 =: count;
      ...
    }
  }; ...
}
```

```
typedef struct {
  int root$count; ...
  unsigned char root; ...
} root;

void fun$root$reset(root *self) {
  (*self).root$count = 3; ...
  (*self).root = 1; ...
}

typedef struct {
  unsigned char root$f$elab$2$btn1$r$release;
  unsigned char root$zero;
} btn1$release;

void fun$btn1$release(root *self, btn1$release *out) {
  (*out).root$zero = 0;
  (*out).root$f$elab$2$btn1$r$release = 1;
  if ((*self).root$f$elab$2$btn1$elab$5) {
    (*self).root$f$elab$2$btn1$green = 150;
  }
  if ((*self).root$f$elab$2$elab$6) {
    (*self).root$count = (*self).root$count - 1;
  }
  if ((*self).root$count == 0) {
    if ((*self).root$elab$1) {
      if ((*self).root$f$elab$2$elab$6) {
        (*out).root$zero = 1;
      }
    }
  }
  ... // handle other count and zero events
}
```

Compiling a Smalite program

```
Component root {
  Int count 3;
  Spike zero; (count == 0) -> zero;
  Frame f ("ICE␣2025", 300, 50) {
    Font _ ("arial.ttf", 20) {
      FillColor btn1 (150,150,150) { ...
        r.release -> dhg;
        dhg: 150 =: green;
      };
      btn1.r.release -> dec;
      dec: last count - 1 =: count;
      ...
    }
  }; ...
}
```

```
typedef struct {
  int root$count; ...
  unsigned char root; ...
} root;

void fun$root$reset(root *self) {
  (*self).root$count = 3; ...
  (*self).root = 1; ...
}

typedef struct {
  unsigned char root$f$elab$2$btn1$r$release;
  unsigned char root$zero;
} btn1$release;

void fun$btn1$release(root *self, btn1$release *out) {
  (*out).root$zero = 0;
  (*out).root$f$elab$2$btn1$r$release = 1;
  if ((*self).root$f$elab$2$btn1$elab$5) {
    (*self).root$f$elab$2$btn1$green = 150;
  }
  if ((*self).root$f$elab$2$elab$6) {
    (*self).root$count = (*self).root$count - 1;
  }
  if ((*self).root$count == 0) {
    if ((*self).root$elab$1) {
      if ((*self).root$f$elab$2$elab$6) {
        (*out).root$zero = 1;
      }
    }
  }
  ... // handle other count and zero events
}
```

Compiling a Smalite program

```
Component root {
  Int count 3;
  Spike zero; (count == 0) -> zero;
  Frame f ("ICE␣2025", 300, 50) {
    Font _ ("arial.ttf", 20) {
      FillColor btn1 (150,150,150) { ...
        r.release -> dhg;
        dhg: 150 =: green;
      };
      btn1.r.release -> dec;
      dec: last count - 1 =: count;
      ...
    }
  }; ...
}
```

```
typedef struct {
  int root$count; ...
  unsigned char root; ...
} root;

void fun$root$reset(root *self) {
  (*self).root$count = 3; ...
  (*self).root = 1; ...
}

typedef struct {
  unsigned char root$f$elab$2$btn1$r$release;
  unsigned char root$zero;
} btn1$release;

void fun$btn1$release(root *self, btn1$release *out) {
  (*out).root$zero = 0;
  (*out).root$f$elab$2$btn1$r$release = 1;
  if ((*self).root$f$elab$2$btn1$elab$5) {
    (*self).root$f$elab$2$btn1$green = 150;
  }
  if ((*self).root$f$elab$2$elab$6) {
    (*self).root$count = (*self).root$count - 1;
  }
  if ((*self).root$count == 0) {
    if ((*self).root$elab$1) {
      if ((*self).root$f$elab$2$elab$6) {
        (*out).root$zero = 1;
      }
    }
  }
  ... // handle other count and zero events
}
```

Compiling a Smalite program

```
Component root {
  Int count 3;
  Spike zero; (count == 0) -> zero;
  Frame f ("ICE␣2025", 300, 50) {
    Font _ ("arial.ttf", 20) {
      FillColor btn1 (150,150,150) { ...
        r.release -> dhg;
        dhg: 150 =: green;
      };
      btn1.r.release -> dec;
      dec: last count - 1 =: count;
      ...
    }
  }; ...
}
```

```
typedef struct {
  int root$count; ...
  unsigned char root; ...
} root;

void fun$root$reset(root *self) {
  (*self).root$count = 3; ...
  (*self).root = 1; ...
}

typedef struct {
  unsigned char root$f$elab$2$btn1$r$release;
  unsigned char root$zero;
} btn1$release;

void fun$btn1$release(root *self, btn1$release *out) {
  (*out).root$zero = 0;
  (*out).root$f$elab$2$btn1$r$release = 1;
  if ((*self).root$f$elab$2$btn1$elab$5) {
    (*self).root$f$elab$2$btn1$green = 150;
  }
  if ((*self).root$f$elab$2$elab$6) {
    (*self).root$count = (*self).root$count - 1;
  }
  if ((*self).root$count == 0) {
    if ((*self).root$elab$1) {
      if ((*self).root$f$elab$2$elab$6) {
        (*out).root$zero = 1;
      }
    }
  }
  ... // handle other count and zero events
}
```

Compiling a Smalite program

```
Component root {
  Int count 3;
  Spike zero; (count == 0) -> zero;
  Frame f ("ICE␣2025", 300, 50) {
    Font _ ("arial.ttf", 20) {
      FillColor btn1 (150,150,150) { ...
        r.release -> dhg;
        dhg: 150 =: green;
      };
      btn1.r.release -> dec;
      dec: last count - 1 =: count;
      ...
    }
  }; ...
}
```

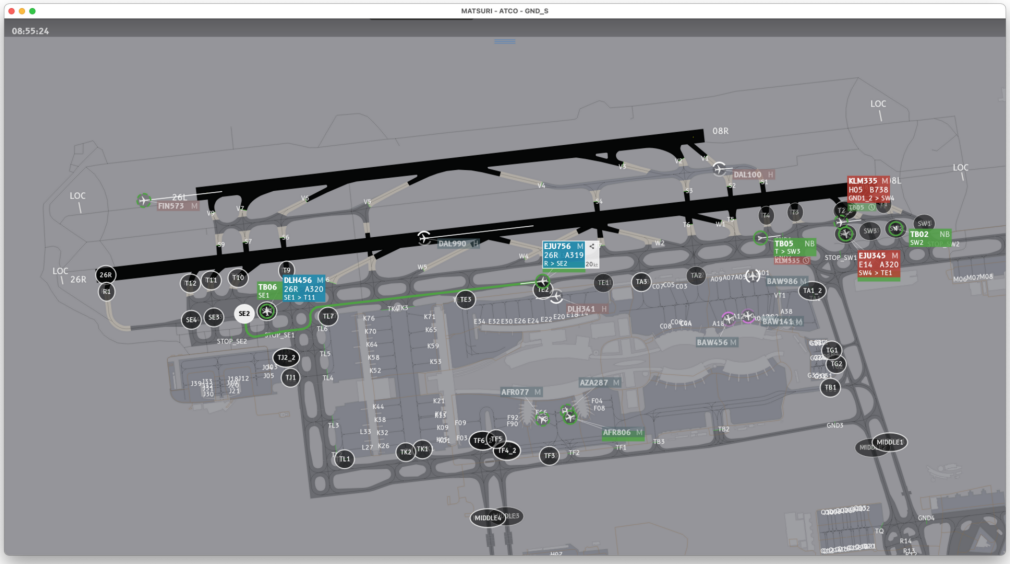
```
typedef struct {
  int root$count; ...
  unsigned char root; ...
} root;

void fun$root$reset(root *self) {
  (*self).root$count = 3; ...
  (*self).root = 1; ...
}

typedef struct {
  unsigned char root$f$elab$2$btn1$r$release;
  unsigned char root$zero;
} btn1$release;

void fun$btn1$release(root *self, btn1$release *out) {
  (*out).root$zero = 0;
  (*out).root$f$elab$2$btn1$r$release = 1;
  if ((*self).root$f$elab$2$btn1$elab$5) {
    (*self).root$f$elab$2$btn1$green = 150;
  }
  if ((*self).root$f$elab$2$elab$6) {
    (*self).root$count = (*self).root$count - 1;
  }
  if ((*self).root$count == 0) {
    if ((*self).root$elab$1) {
      if ((*self).root$f$elab$2$elab$6) {
        (*out).root$zero = 1;
      }
    }
  }
  ... // handle other count and zero events
}
```

Compiling a Smalite program

```
Component root {
  Int count 3;
  Spike zero; (count == 0) -> zero;
  Frame f ("ICE␣2025", 300, 50) {
    Font _ ("arial.ttf", 20) {
      FillColor btn1 (150,150,150) { ...
        r.release -> dhg;
        dhg: 150 =: green;
      };
      btn1.r.release -> dec;
      dec: last count - 1 =: count;
      ...
    }
  }; ...
}
```

```
typedef struct {
  int root$count; ...
  unsigned char root; ...
} root;

void fun$root$reset(root *self) {
  (*self).root$count = 3; ...
  (*self).root = 1; ...
}

typedef struct {
  unsigned char root$f$elab$2$btn1$r$release;
  unsigned char root$zero;
} btn1$release;

void fun$btn1$release(root *self, btn1$release *out) {
  (*out).root$zero = 0;
  (*out).root$f$elab$2$btn1$r$release = 1;
  if ((*self).root$f$elab$2$btn1$elab$5) {
    (*self).root$f$elab$2$btn1$green = 150;
  }
  if ((*self).root$f$elab$2$elab$6) {
    (*self).root$count = (*self).root$count - 1;
  }
  if ((*self).root$count == 0) {
    if ((*self).root$elab$1) {
      if ((*self).root$f$elab$2$elab$6) {
        (*out).root$zero = 1;
      }
    }
  }
  ... // handle other count and zero events
}
```

Dynamicity in Smala



Dynamicity in Smala

```
_action_ del_rect (Process src, Process data) {
  grect = find (&src, "../..")
  delete grect
}

_main_ Component root {
  Frame frame ("dyn␣rect", 600, 600)
  FillColor fg_color (#373700)
  OutlineColor fg_color_outline (#FF0000)
  NativeAction na_del_rect (del_rect, root, 1)
  Layer things { }
  frame.press -> (root) {
    addChildrenTo root.things {
      Component grect {
        Rectangle rect (frame.press.x, frame.press.y, 100, 100)
        rect.press -> na_del_rect
      }
    }
  }
}
```

create process

Dynamicity in Smala

```
_action_ del_rect (Process src, Process data) {
  grect = find (&src, "../..")
  delete grect
}

_main_ Component root {
  Frame frame ("dyn rect", 600, 600)
  FillColor fg_color (#373700)
  OutlineColor fg_color_outline (#FF0000)
  NativeAction na_del_rect (del_rect, root, 1)
  Layer things { }
  frame.press -> (root) {
    addChildrenTo root.things {
      Component grect {
        Rectangle rect (frame.press.x, frame.press.y, 100, 100)
        rect.press -> na_del_rect
      }
    }
  }
}
```

delete process

create process

Conclusion

What we did:

- Relational semantics for a minimalist User Interface Description Language
- A prototype compiler
- Both implemented in 

Future work / Open questions

- Prove compiler correctness
- Support for full Smala (dynamicity, native actions, etc)
- Equivalence with bigraphical semantics, model checking
- Interactive / graphical semantics

Open to questions and collaborations :)