Reactive Semantics for User Interface Description Languages

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

June 20, 2025

Basile Pesin Celia Picard Cyril Allignol

Specifying Interactive Applications

The evolution of Safety-Critical HMIs



Concorde (1976)

The evolution of Safety-Critical HMIs



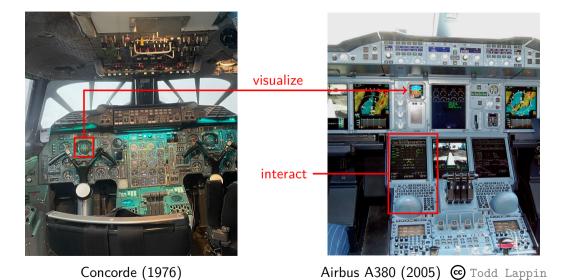
visualize



Concorde (1976)

Airbus A380 (2005) © Todd Lappin

The evolution of Safety-Critical HMIs



Programming HMIs with procedural languages: the "callback hell"

```
startBtn.addEventListener("click", function () {
  startBtn.disabled = true:
  status.innerText = "Started.uPleaseuconfirm.";
  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.";
    });
 });
}):
```

Start Confirm Finish
Started. Please confirm.

Programming HMIs with procedural languages: the "callback hell"

```
startBtn.addEventListener("click", function () {
  startBtn.disabled = true:
  status.innerText = "Started.uPleaseuconfirm.";
  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:
```

Confirm Finish Start Started, Please confirm.

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



```
Component root {
  Int count 3
  Spike zero
  (count == 0) \rightarrow zero
 Frame f ("ICE_12025", 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) \rightarrow btn1.r
  ... // restart button + counter label
 Exit e(0)
 f.close -> e
```

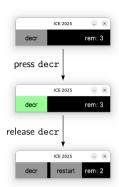


```
Component root {
  Int count 3:
  Spike zero; (count == 0) -> zero;
 Frame f ("ICE_2025", 300, 50) {
    Font _ ("arial.ttf", 20) {
      FillColor btn1 (150,150,150) {
        Rectangle r (0,0,100,f.height) {
          FillColor (0,0,0) {
            Text ("decr", r.x + 30, 13) {}
          };
        };
        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 };
```

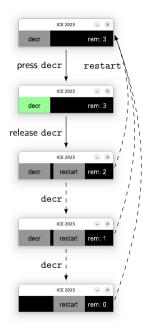
```
Component root {
  Int count 3:
  Spike zero; (count == 0) -> zero;
 Frame f ("ICE_2025", 300, 50) {
    Font _ ("arial.ttf", 20) {
      FillColor btn1 (150,150,150) {
        Rectangle r (0,0,100,f.height) {
          FillColor (0,0,0) {
            Text ("decr", r.x + 30, 13) {}
          };
        };
        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 };
```



```
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 };
```



```
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)



Execution Model

$$(E_0, A_0)$$



where : E_i : $Path \rightarrow Value$ $A_i \subseteq Path$

Execution Model

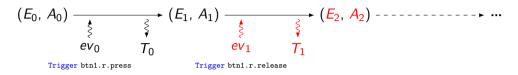




where : $E_i : Path \rightarrow Value \\ A_i \subseteq Path \\ ev_i \in Event \\ T_i \subseteq Event$

ev ::= Trigger path | Assign v path

Execution Model





where : $E_i : Path \rightarrow Value \\ A_i \subseteq Path \\ ev_i \in Event \\ T_i \subseteq Event$

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

```
ICE 2025
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) {} ...
   }; ...
 }; ...
```

```
ICE 2025
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) {} ...
   }; ...
 }; ...
```

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

```
ICE 2025
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} \mathtt{count} \mapsto 3 \\ \mathtt{f.height} \mapsto 50 \\ \mathtt{f.btn1.r.x} \mapsto 0 \\ \mathtt{f.btn1.r.height} \mapsto 50 \\ \mathtt{f.btn1.r.t.x} \mapsto 30 \\ \ldots \end{array} \right\}
```

```
ICE 2025
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} \mathtt{count} \mapsto 3 \\ \mathtt{f.height} \mapsto 50 \\ \mathtt{f.btn1.r.x} \mapsto 0 \\ \mathtt{f.btn1.r.height} \mapsto 50 \\ \mathtt{f.btn1.r.t.x} \mapsto 30 \\ \ldots \end{array} \right\}
```

$${\cal A}= \left\{
ight.$$



```
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) {} ...
  }; ...
 }; ...
```

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

$$A = \left\{ egin{array}{l} \mathtt{f}.\mathtt{btn1} \\ \mathtt{f}.\mathtt{btn1}.\mathtt{r} \\ \mathtt{f}.\mathtt{btn2} \\ ... \end{array}
ight\}$$



```
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) {} ...
  }; ...
 }; ...
```

$$E = \left\{ egin{array}{l} \mathtt{count} \mapsto 3 \ \mathtt{f.height} \mapsto 50 \ \mathtt{f.btn1.r.x} \mapsto 0 \ \mathtt{f.btn1.r.height} \mapsto 50 \ \mathtt{f.btn1.r.t.x} \mapsto 30 \ \mathrm{...} \end{array}
ight.$$

$$A = \left\{ egin{array}{l} \mathtt{f}.\mathtt{btn1} \\ \mathtt{f}.\mathtt{btn1}.\mathtt{r} \\ \mathtt{f}.\mathtt{btn2} \\ ... \end{array}
ight\}$$

System Initialization – Rules

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

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

$$\overline{E \vdash_{\text{init}}^{\text{env}} (Component < > \times \{\epsilon\})_{r_{x}} \Downarrow E}$$

$$E \vdash_{\text{init}}^{\text{env}} (Component < > \times \{ps\})_{r_{x}} \Downarrow E'$$

$$E \vdash_{\text{init}}^{\text{env}} (Component < > \times \{ps\})_{r_{x}} \Downarrow E''$$

$$E \vdash_{\text{init}}^{\text{env}} (Component < > \times \{ps\})_{r_{x}} \Downarrow E''$$

$$\begin{array}{c} \frac{1 \text{ activ}}{1 \text{ init}} \left(\text{Component} < \text{d} > x \left\{ ps \right\} \right)_{r_x} \Downarrow \emptyset \\ \\ \forall i. \ \ \frac{1 \text{ activ}}{1 \text{ init}} \left(ps_i \right)_{r_x, x} \Downarrow A_i \\ \\ \frac{1 \text{ init}}{1 \text{ init}} \left(\text{Component} < \text{a} > x \left\{ ps \right\} \right)_{r_x} \Downarrow \left(\bigcup_i A_i \right) \cup \left\{ r_x, x \right\} \end{array}$$

System Initialization – Rules

$$\frac{\emptyset \stackrel{\mathsf{L}^{env}}{\underset{\mathsf{init}}{\mathsf{ent}}} \operatorname{\textit{proc}}_{\epsilon} \Downarrow E \qquad \stackrel{\mathsf{L}^{\mathsf{activ}}}{\underset{\mathsf{init}}{\mathsf{ent}}} \operatorname{\textit{proc}}_{\epsilon} \Downarrow A}{\vdash_{\mathsf{init}} \operatorname{\textit{proc}} \Downarrow E, A}$$

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

$$\overline{E \vdash_{\text{init}}^{\text{env}} (\text{Component} < > x \{\epsilon\})_{r_{x}} \Downarrow E}$$

$$E \vdash_{\text{init}}^{\text{env}} (p_{r_{x}.x}) \Downarrow E'$$

$$E' \vdash_{\text{init}}^{\text{env}} (p_{r_{x}.x}) \Downarrow E'$$

$$E' \vdash_{\text{init}}^{\text{env}} (p_{r_{x}.x}) \Downarrow E'$$

$$E \vdash_{\text{init}}^{\text{env}} (p_{r_{x}.x}) \Downarrow E'$$

System Initialization – Rules

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

$$\begin{array}{c} \emptyset, E \vdash_{\exp} e \Downarrow v \\ \hline E \vdash_{\inf}^{env} (ty \times e)_{r_{x}} \Downarrow E[r_{x}.x \mapsto v] \\ \hline \\ E \vdash_{\inf}^{env} (Component < > \times \{\epsilon\})_{r_{x}} \Downarrow E \\ \hline \\ E' \vdash_{\inf}^{env} (Component < > \times \{ps\})_{r_{x}} \Downarrow E' \\ \hline \\ E \vdash_{\inf}^{env} (Component < > \times \{ps\})_{r_{x}} \Downarrow E'' \\ \hline \\ E \vdash_{\inf}^{env} (Component < > \times \{ps\})_{r_{x}} \Downarrow E'' \\ \hline \end{array}$$

$$\begin{array}{c} \overline{|}_{\text{init}}^{\text{activ}} \left(\text{Component} < \text{d} > x \; \{ps\} \right)_{r_x} \downarrow \emptyset \\ \\ \forall i. \;\; |_{\text{init}}^{\text{activ}} \left(ps_i \right)_{r_x \cdot x} \downarrow A_i \\ \\ \overline{|}_{\text{init}}^{\text{activ}} \left(\text{Component} < \text{a} > x \; \{ps\} \right)_{r_x} \downarrow \left(\bigcup_i A_i \right) \cup \{r_x \cdot x\} \end{array}$$

$$T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash_{prop} proc \Downarrow ev\}$$

$$E, T, E', A' \vdash_{safe} proc_{\epsilon}$$

$$E, A, T \vdash_{update} proc \Downarrow E', A'$$

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

$$program \qquad initial event$$

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

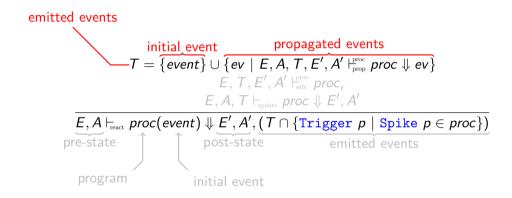
$$E, T, E', A' \vdash_{enfe}^{proc} proc_{\epsilon}$$

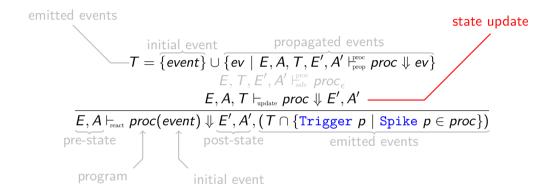
$$E, A, T \vdash_{update} proc \Downarrow E', A'$$

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

$$pre-state \qquad post-state \qquad emitted events$$

$$program \qquad initial event$$





Propagation of Events – Example



```
T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash_{\leftarrow}^{\leftarrow} proc \Downarrow ev\}
E, T, E', A' \vdash_{\leftarrow}^{\leftarrow} proc_{\leftarrow}
E, A, T \vdash_{\leftarrow} proc \Downarrow E', A'
E, A \vdash_{\leftarrow} proc(event) \Downarrow E', A', (T \cap \{Trigger \ p \mid 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;
```

Propagation of Events – Example



```
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\{egin{array}{l} 	ext{Trigger btn1.r.release} \ 	ext{Trigger dhg} \end{array}
ight.
```

Propagation of Events – Example



```
T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash_{i=v}^{uv} proc \Downarrow ev\}
E, T, E', A' \vdash_{i=v}^{uv} proc_{\epsilon}
E, A, T \vdash_{i=vu} proc \Downarrow E', A'
E, A \vdash_{uv} proc(event) \Downarrow E', A', (T \cap \{Trigger p \mid 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\{ egin{array}{ll} {	t Trigger btn1.r.release} \ {	t Trigger dhg} \ {	t Assign 150 btn1.green} \end{array} 
ight.
```

Propagation of Events – Example



```
T = \{event\} \cup \{ev \mid E, A, T, E', A' \mid_{t=0}^{top} proc \Downarrow ev\}
E, T, E', A' \mid_{t=0}^{top} proc_{\epsilon}
E, A, T \mid_{toplan} proc \Downarrow E', A'
E, A \mid_{toplan} proc(event) \Downarrow E', A', (T \cap \{Trigger p \mid 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\{ egin{array}{ll} {	t Trigger btn1.r.release} \ {	t Trigger dhg} \ {	t Assign 150 btn1.green} \ {	t Trigger dec} \ {	t Assign 1 count} \end{array} 
ight\}
```

Propagation of Events – Example



```
T = \{ event \} \cup \{ ev \mid \textit{E}, \textit{A}, \textit{T}, \textit{E}', \textit{A}' \not\vdash_{i=0}^{no} proc \ \downarrow ev \}
E, \textit{T}, \textit{E}', \textit{A}' \not\vdash_{i=0}^{no} proc_{e}
E, \textit{A}, \textit{T} \vdash_{i=\text{sub}} proc \ \downarrow \textit{E}', \textit{A}'
E, \textit{A} \vdash_{i=\text{sub}} proc(event) \ \downarrow \textit{E}', \textit{A}', (\textit{T} \cap \{\text{Trigger } p \mid \text{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\{ egin{array}{ll} {	t Trigger btn1.r.release} \ {	t Trigger dhg} \ {	t Assign 150 btn1.green} \ {	t Trigger dec} \ {	t Assign 1 count} \end{array} 
ight\}
```

```
T = \{ event \} \cup \{ ev \mid E, A, T, E', A' \mid_{\frac{prov}{even}}^{proc} proc \Downarrow ev \}
E, T, E', A' \mid_{\frac{prov}{even}}^{proc} proc \downarrow_{even}
E, A, T \mid_{\frac{prov}{even}} proc \Downarrow E', A'
E, A \mid_{\frac{prov}{even}} proc(event) \Downarrow E', A', (T \cap \{ Trigger \ p \mid Spike \ p \in proc \})
```

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

```
T = \{ event \} \cup \{ ev \mid \textbf{\textit{E}}, \textbf{\textit{A}}, \textbf{\textit{T}}, \textbf{\textit{E}'}, \textbf{\textit{A}'} \vdash_{pole}^{moc} proc \Downarrow ev \} E, T, E', A' \vdash_{pole}^{moc} proc _{e} E, \textbf{\textit{A}}, T \vdash_{pole} proc \Downarrow E', \textbf{\textit{A}'} E, A \vdash_{pole} proc(event) \Downarrow E', A', (T \cap \{ \text{Trigger } p \mid \text{Spike } p \in proc \})
```

 \exists assignment process rooted at $r_x.x$ in the program $r_x.x \text{ is triggered}$ $proc(r_x.x) = \lfloor x \colon e = : r_y.y \rfloor$ $r_x \in A' \qquad r_y \in A' \qquad \text{Trigger } r_x.x \in T \qquad E, E' \vdash_{exp} e \Downarrow v$ $E, A, T, E', A' \vdash_{prop}^{proc} proc \Downarrow \text{Assign v } r_y.y$

```
T = \{ event \} \cup \{ ev \mid \underline{E}, A, T, \underline{E}', A' \vdash_{i = 0}^{\text{time}} proc \Downarrow ev \}
E, T, E', A' \vdash_{i = 0}^{\text{time}} proc \Downarrow E', A'
E, A, T \vdash_{i = 0}^{\text{time}} proc \Downarrow E', A'
\overline{E}, A \vdash_{i = 0}^{\text{time}} proc(event) \Downarrow E', A', (T \cap \{\text{Trigger } p \mid \text{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
$$\underbrace{ r_x.x}_{proc}(r_x.x) = \lfloor x: \ e =: \ r_y.y \rfloor$$

$$\underbrace{ r_x \in A' \quad r_y \in A' \quad \text{Trigger } r_x.x \in T \quad E, E' \vdash_{exp} e \Downarrow v }_{E, A, T, E', A' \vdash_{prop} proc} \Downarrow \text{Assign } \lor r_y.y$$

```
T = \{ event \} \cup \{ ev \mid \textbf{\textit{E}}, \textbf{\textit{A}}, \textbf{\textit{T}}, \textbf{\textit{E}'}, \textbf{\textit{A}'} \vdash_{pole}^{moc} proc \Downarrow ev \} E, T, E', A' \vdash_{pole}^{moc} proc _{e} E, \textbf{\textit{A}}, T \vdash_{pole} proc \Downarrow E', \textbf{\textit{A}'} E, A \vdash_{pole} proc(event) \Downarrow E', A', (T \cap \{ \text{Trigger } p \mid \text{Spike } p \in proc \})
```

assignment process rooted at $r_x.x$ in the program $r_x.x \text{ is triggered}$ assignment and target parents are active $r_x.x \text{ is triggered}$ $r_x.x \text{ is triggered}$ $r_x.x \text{ e =: } r_y.y \text{ expression evaluates to } v$ $r_x \in A' \quad r_y \in A' \quad \text{Trigger } r_x.x \in T \quad E, E' \vdash_{\text{exp}} e \Downarrow v$ $E, A, T, E', A' \vdash_{\text{prop}}^{\text{proc}} proc \Downarrow \text{Assign v } r_y.y$

Propagation of Events - Bindings

```
T = \{ event \} \cup \{ ev \mid E, A, T, E', A' \mid_{p=0}^{mer} proc \Downarrow ev \}
E, T, E', A' \mid_{p=0}^{mer} proc 
E, A, T \mid_{p=0}^{mer} proc \Downarrow E', A'
\overline{E, A \mid_{mer} proc(event) \Downarrow E', A', (T \cap \{ Trigger \ p \mid Spike \ p \in proc \})}
```

$$\frac{proc(r_{x}.x) = \lfloor x : lhs \rightarrow <_> rhs \rfloor}{E, T, E' \mid_{prop}^{lhs} lhs} \quad A, A' \mid_{prop}^{rhs} rhs \Downarrow ev}$$

$$\frac{E, A, T, E', A' \mid_{prop}^{proc} proc \Downarrow ev}$$

$$\frac{\text{Trigger } x \in T}{E, T, E' \mid_{\text{prop}}^{\text{ths}} T?(x)} \qquad \frac{x \in \text{free}(e) \quad \text{Assign } v \times \in T \quad E, E' \mid_{\text{exp}} e \Downarrow \text{true}}{E, T, E' \mid_{\text{prop}}^{\text{ths}} (e)?}$$

$$x: \text{zero } \rightarrow ! \text{ btn1.r} \qquad ex: (count == 0) \rightarrow \text{zero}$$

$$r_{X} \in A'$$

$$A, A' \vdash_{\text{prop}}^{\text{ths}} T! (r_{X}.x) \Downarrow \text{Trigger } r_{X}.x$$

$$ex: (count == 0) \rightarrow \text{zero}$$

$$r_{X} \in A'$$

$$A, A' \vdash_{\text{prop}}^{\text{ths}} A! (r_{X}.x)$$

$$ex: (count == 0) \rightarrow \text{zero}$$

Propagation of Events - Bindings

$$T = \{event\} \cup \{ev \mid E, A, T, E', A' \mid_{\leftarrow}^{\leftarrow} proc \downarrow ev\}$$

$$E, T, E', A' \mid_{\leftarrow}^{\leftarrow} proc \downarrow$$

$$E, A, T \mid_{\leftarrow}^{\leftarrow} proc \downarrow E', A'$$

$$E, A \mid_{\leftarrow}^{\leftarrow} proc(event) \downarrow E', A', (T \cap \{Trigger \ p \mid Spike \ p \in proc\})$$

$$\frac{proc(r_{x}.x) = \lfloor x \colon \mathit{lhs} \ -><_> \ \mathit{rhs} \rfloor \quad r_{x}.x \in A'}{E, T, E' \mid_{prop}^{\underline{lhs}} \ \mathit{lhs} \quad A, A' \mid_{prop}^{\underline{rhs}} \ \mathit{rhs} \Downarrow ev}$$

$$E, A, T, E', A' \mid_{prop}^{\underline{proc}} \ \mathit{proc} \Downarrow ev}$$

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

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

ex: zero ->! btn1.r

$$\frac{r_X \in A'}{A, A' \vdash_{\text{prop}}^{\text{rhs}} T! (r_X.x) \Downarrow \text{Trigger } r_X.x}$$
ex: (count == 0) -> zero

$$r_X \in A' \qquad r_X.x \notin A$$

$$A, A' \vdash_{\text{prop}}^{\text{ths}} A! (r_X.x) \Downarrow \text{Activate } r_X.x$$

$$ex: (count > 0) \rightarrow \text{btn1.r}$$

Propagation of Events - Bindings

$$T = \{event\} \cup \{ev \mid E, A, T, E', A' \mid_{p^{me}}^{pme} proc \Downarrow ev\}$$

$$E, T, E', A' \mid_{p^{me}}^{pme} proc \downarrow E', A'$$

$$E, A \mid_{n_{met}} proc(event) \Downarrow E', A', (T \cap \{Trigger \ p \mid Spike \ p \in proc\})$$

$$\frac{proc(r_{x}.x) = \lfloor x \colon lhs \to <_> rhs \rfloor}{E, T, E' \vdash_{prop}^{lhs} lhs} \frac{r_{x}.x \in A'}{A, A' \vdash_{prop}^{rhs} rhs \Downarrow ev}$$

$$\frac{E, A, T, E', A' \vdash_{prop}^{proc} proc \Downarrow ev}{E, A, T, E', A' \vdash_{prop}^{proc} proc \Downarrow ev}$$

Trigger
$$x \in T$$

$$E, T, E' \vdash_{\text{prop}}^{\text{llis}} T?(x)$$

$$x \in \text{free}(e) \quad \text{Assign } v \times \in T \quad E, E' \vdash_{\text{exp}} e \Downarrow \text{true}$$

$$E, T, E' \vdash_{\text{prop}}^{\text{llis}} (e)?$$

$$E, T, E' \vdash_{\text{prop}}^{\text{llis}} (e)?$$

$$Ex: \text{geno } ->! \text{ btn1.r}$$

$$ex: (\text{count.} == 0) \rightarrow \text{zero}$$

$$r_X \in A'$$

$$A, A' \vdash_{\text{prop}}^{\text{rhs}} T! (r_X.x) \Downarrow \text{Trigger } r_X.x$$

$$ex: (count == 0) \rightarrow \text{zero}$$

$$r_X \in A' r_X.x \notin A$$

$$A, A' \vdash_{\text{prop}}^{\text{rhs}} A! (r_X.x) \Downarrow \text{Activate } r_X.x$$

$$ex: (count > 0) \rightarrow \text{btn1.r}$$

Propagation of Events – Children Activation

```
T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash_{post}^{post} proc \Downarrow ev\}
E, T, E', A' \vdash_{post}^{post} proc _{\epsilon}
E, A, T \vdash_{post} proc \Downarrow E', A'
E, A \vdash_{most} proc(event) \Downarrow E', A', (T \cap \{Trigger p \mid Spike p \in proc\})
```

$$arphi_{ ext{prop}}^{ ext{activ}} (x \colon ext{\it lhs} ext{\it -><} ext{a>} ext{\it rhs})_{r_X} \Downarrow ext{Activate} \; r_X$$

$$\Gamma_{\text{prop}}^{\text{activ}}$$
 (Component\$x \{ps\}\$ \) \$_{r_x} \Downarrow \text{Activate } r_x.x_x \in r_x\$

Propagation of Events – Children Activation

```
T = \{event\} \cup \{ev \mid E, A, T, E', A' \mid_{loop}^{loop} proc \downarrow ev\}
E, T, E', A' \mid_{loop}^{loop} proc \downarrow
E, A, T \mid_{loop}^{loop} proc \downarrow E', A'
E, A \mid_{loop}^{loop} proc (event) \downarrow E', A', (T \cap \{Trigger \ p \mid Spike \ p \in proc\})
```

$$\frac{proc(r_{x}.x) = \lfloor \text{Component} < a > x \{ps\} \rfloor}{\text{Activate } r_{x}.x \in T} \frac{\prod_{\text{prop}}^{\text{activ}} (ps_{i})_{r_{x}.x} \Downarrow \text{Activate } y}{E, A, T, E', A' \vdash_{\text{prop}}^{\text{proc}} proc \Downarrow \text{Activate } y}$$

$$\vdash_{\text{prop}}^{\text{activ}} (x: \textit{lhs} \rightarrow < \texttt{a} > \textit{rhs})_{r_x} \Downarrow \texttt{Activate} \ r_x$$

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

Propagation of Events - Children Activation

```
\begin{split} T &= \{ event \} \cup \{ ev \mid E, A, T, E', A' \vdash_{mo}^{\text{top}} proc \Downarrow ev \} \\ &= E, T, E', A' \vdash_{mo}^{\text{top}} proc_{e} \\ &= E, A, T \vdash_{mon} proc \uplus E', A' \\ \hline E, A \vdash_{mon} proc(event) \Downarrow E', A', (T \cap \{ \text{Trigger } p \mid \text{Spike } p \in proc \} ) \end{split}
```

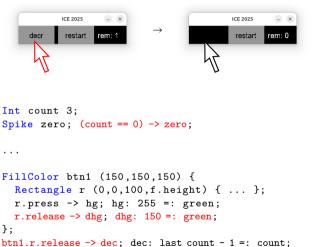
$$\frac{proc(r_{x}.x) = \lfloor \texttt{Component} < \texttt{a} > x \{ps\} \rfloor}{\texttt{Activate } r_{x}.x \in T} \frac{\mid_{\text{prop}}^{\text{activ}} (ps_{i})_{r_{x}.x} \Downarrow \texttt{Activate } y}{E, A, T, E', A' \vdash_{\text{prop}}^{\text{proc}} proc \Downarrow \texttt{Activate } y}$$

$$\vdash_{\text{prop}}^{\text{activ}} (x: \textit{lhs} \rightarrow < \texttt{a} > \textit{rhs})_{r_X} \Downarrow \texttt{Activate} \ r_X$$

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

and similar rules for children deactivation

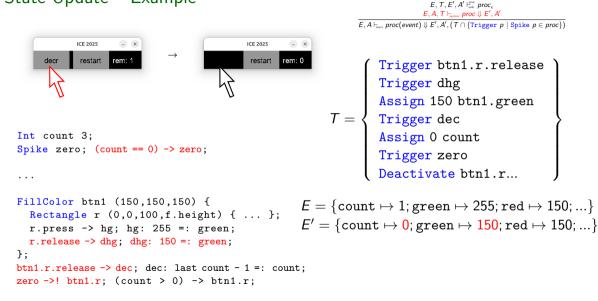
State Update - Example



zero ->! btn1.r; (count > 0) -> btn1.r;

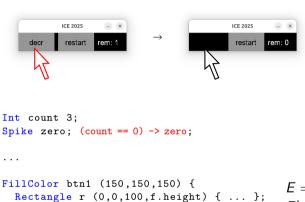
```
T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash_{mon}^{proc} proc \downarrow ev\}
                                                 E. T. E', A' Harin proc.
                                              F, A, T \vdash \dots proc \bot F', A'
               E.A \vdash proc(event) \Downarrow E', A', (T \cap \{Trigger p \mid Spike p \in proc\})
Trigger btn1.r.release
Trigger dhg
Assign 150 btn1.green
Trigger dec
Assign 0 count
Trigger zero
Deactivate btn1.r...
```

State Update - Example



 $T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash_{mon}^{proc} proc \downarrow ev\}$

State Update - Example



```
F. T. F'. A' \vdash^{\text{proc}} proc.
                                                   E.A.T \vdash_{mint} proc \Downarrow E', A'
                        E.A \vdash proc(event) \Downarrow E', A', (T \cap \{Trigger p \mid Spike p \in proc\})
          Trigger btn1.r.release
Trigger dhg
Assign 150 btn1.green
Trigger dec
Assign 0 count
Trigger zero
Deactivate btn1.r...
E = \{ \text{count} \mapsto 1; \text{green} \mapsto 255; \text{red} \mapsto 150; ... \}
```

 $T = \{event\} \cup \{ev \mid E, A, T, E', A' \vdash_{mon}^{proc} proc \downarrow ev\}$

```
r.press -> hg; hg: 255 =: green; E' = \{ \text{count} \mapsto 0; \text{green} \mapsto 150; \text{red} \mapsto 150; \dots \}
r.release -> dhg; dhg: 150 =: green; E' = \{ \text{count} \mapsto 0; \text{green} \mapsto 150; \text{red} \mapsto 150; \dots \}
\{ \text{btn1.r.release} \rightarrow \text{dec}; \text{dec: last count} - 1 =: \text{count}; A = \{ \text{btn1; btn1.r; } \dots \}
zero ->! btn1.r; (count > 0) -> btn1.r; A = \{ \text{btn1; } \dots \}
```

```
\begin{split} T &= \{event\} \cup \{ev \mid E, A, T, E', A' \vdash_{lim}^{mo} proc \Downarrow ev\} \\ &= E, T, E', A' \vdash_{lim}^{mo} proc \\ &= E, A, T \vdash_{lim} proc \Downarrow E', A' \\ &= E, A \vdash_{lim} proc (event) \Downarrow E', A', (T \cap \{\text{Trigger } p \mid \text{Spike } p \in proc\}) \end{split}
```

$$\forall p \ v, \ \operatorname{Assign} \ v \ p \in T \implies E'(p) = \lfloor v \rfloor$$

$$\forall p, \ (\forall v, \ \operatorname{Assign} \ v \ p \notin T) \implies E'(p) = E(p)$$

$$\forall p, \ \operatorname{Activate} \ p \in T \implies p \in A'$$

$$\forall p, \ \operatorname{Deactivate} \ p \in T \implies p \notin A'$$

$$\underbrace{\forall p, \ (\operatorname{Activate} \ p \notin T \land \operatorname{Deactivate} \ p \notin T) \implies (p \in A' \iff p \in A)}_{E, A, T \ \text{Extens } proc \ \downarrow E', A'}$$

```
\begin{split} T &= \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{me}_{\mathbb{R}^m} proc \Downarrow ev\} \\ &= E, T, E', A' \vdash^{me}_{\mathbb{L}^m} proc \\ &= E, A, T \vdash_{\mathbb{L}^m} proc \Downarrow E', A' \\ &= E, A \vdash^{me}_{\mathbb{L}^m} proc (event) \Downarrow E', A', (T \cap \{\text{Trigger } p \mid \text{Spike } p \in proc\}) \end{split}
```

$$\forall p \ v, \ \mathsf{Assign} \ v \ p \in T \implies E'(p) = \lfloor v \rfloor$$

$$\forall p, \ (\forall v, \ \mathsf{Assign} \ v \ p \notin T) \implies E'(p) = E(p)$$

$$\forall p, \ \mathsf{Activate} \ p \in T \implies p \in A'$$

$$\forall p, \ \mathsf{Deactivate} \ p \in T \implies p \notin A'$$

$$\forall p, \ (\mathsf{Activate} \ p \notin T \land \mathsf{Deactivate} \ p \notin T) \implies (p \in A' \iff p \in A)$$

$$E. A. T \vdash_{\mathsf{Torted}} \mathsf{proc} \Downarrow E', A'$$

```
\begin{split} T &= \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{me}_{proc} proc \ \downarrow ev\} \\ &= E, T, E', A' \vdash^{me}_{proc} proc_{e} \\ &= E, A, T \vdash^{me}_{proc} proc_{e} \ \not \in A \vdash^
```

$$\forall p \ v, \ \operatorname{Assign} \lor p \in T \implies E'(p) = \lfloor v \rfloor$$

$$\forall p, \ (\forall v, \ \operatorname{Assign} \lor p \notin T) \implies E'(p) = E(p)$$

$$\forall p, \ \operatorname{Activate} p \in T \implies p \in A'$$

$$\forall p, \ \operatorname{Deactivate} p \in T \implies p \notin A'$$

$$\forall p, \ (\operatorname{Activate} p \notin T \land \operatorname{Deactivate} p \notin T) \implies (p \in A' \iff p \in A)$$

$$E. A. T \vdash_{\text{trade}} proc \Downarrow E', A'$$

```
\begin{split} T &= \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{me}_{proc} proc \ \downarrow ev\} \\ &= E, T, E', A' \vdash^{me}_{proc} proc \\ &= E, A, T \vdash_{quar} proc \ \downarrow E', A' \\ \hline E, A \vdash_{mu} proc(event) \ \downarrow E', A', (T \cap \{\text{Trigger } p \mid \text{Spike } p \in proc\}) \end{split}
```

```
\forall p \ v, \ \operatorname{Assign} \ v \ p \in T \implies E'(p) = \lfloor v \rfloor
\forall p, \ (\forall v, \ \operatorname{Assign} \ v \ p \notin T) \implies E'(p) = E(p)
\forall p, \ \operatorname{Activate} \ p \in T \implies p \in A'
\forall p, \ \operatorname{Deactivate} \ p \in T \implies p \notin A'
\forall p, \ (\operatorname{Activate} \ p \notin T \land \operatorname{Deactivate} \ p \notin T) \implies (p \in A' \iff p \in A)
E, A, T \vdash_{\operatorname{undate}} \operatorname{proc} \Downarrow E', A'
```

Some contradictory programs:

Rejected by static analysis

a: 0 =: y;
$$(x/y > 10) \rightarrow t$$

$$T = \left\{ egin{array}{l} ext{Trigger a} \ ext{Assign y 0} \ ext{???} \end{array}
ight\}$$

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

$$\frac{proc(r_{x}.x) = \lfloor x \colon lhs \to <_> rhs \rfloor}{E, T, E' \vdash_{prop}^{lhs} lhs} \quad A, A' \vdash_{prop}^{rhs} rhs \Downarrow ev}{E, A, T, E', A' \vdash_{prop}^{proc} proc \Downarrow ev}$$

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

$$E, A, T \vdash_{\text{update}} proc \Downarrow E', A'$$

a: 0 =: y;

$$(x/y > 10)$$
 -> t

$$x \in free(e) \quad \underset{E, T, E' \mid_{prop}}{Assign \lor x \in T} \quad \underset{E, E' \mid_{exp} e \Downarrow true}{E, E' \mid_{exp} e \Downarrow true}$$

$$\frac{proc(r_{x}.x) = \lfloor x \colon \mathit{lhs} \to <_{-} > \mathit{rhs} \rfloor \quad r_{x}.x \in A'}{E, T, E' \mid_{\text{prop}}^{\text{lhs}} \quad \mathit{lhs} \quad A, A' \mid_{\text{prop}}^{\text{prob}} \quad \mathit{rhs} \Downarrow ev}$$

$$E, A, T, E', A' \mid_{\text{prop}}^{\text{proc}} \quad \mathit{proc} \Downarrow ev}$$

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

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

$$E, A, T \vdash_{\text{update}} proc \Downarrow E', A'$$

a: 0 =: y;

$$(x/y > 10) -> t$$

$$x \in free(e) \qquad \underset{E, T, E' \mid_{prop}}{Assign \ v \ x \in T} \qquad \underset{E, T, E' \mid_{prop}}{E, F' \mid_{prop}} (e)?$$

$$\frac{proc(r_{x}.x) = \lfloor x \colon \mathit{lhs} \to <_> \mathit{rhs} \rfloor \quad r_{x}.x \in A'}{E, T, E' \mid_{prop}^{lhs} \quad Hs} \quad A, A' \mid_{prop}^{rhs} rhs \Downarrow ev}{E, A, T, E', A' \mid_{prop}^{proc} proc \Downarrow ev}$$

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

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

$$E, A, T \vdash_{\text{update}} proc \Downarrow E', A'$$

a: 0 =: y;

$$(x/y > 10)$$
 -> t

$$x \in free(e) \quad \text{Assign } v \times f \quad E, E' \mid_{exp} c \text{ true}$$

$$E, T, E' \mid_{prop}^{lhs} (e)$$
?

$$\frac{proc(r_{x}.x) = \lfloor x : \ lhs \ -><_ \ rhs \rfloor}{E, T, E' \mid_{\text{prop}}^{\text{lhs}} \ lhs} \underbrace{A, A' \mid_{\text{prop}}^{\text{pro}} \ rhs \Downarrow ev}_{E, A, T, E', A' \mid_{\text{prop}}^{\text{proc}} \ proc \Downarrow ev}$$

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

$$T = \{event\} \cup \{ev \mid \overline{E, A, T, E', A'} \mid_{prop}^{proc} proc \Downarrow ev}\}$$

$$E, A, T \vdash_{\text{update}} proc \Downarrow E', A'$$

a: 0 =: y;

$$(x/y > 10) -> t$$

$$x \in free(e) \quad Assign \lor x \in T \quad E, E' \mid_{exp} e \Downarrow true$$

$$E, T, E' \mid_{num}^{lin} (e)?$$

$$\frac{proc(r_{x}.x) = \lfloor x : \ lhs \ -><_> \ rhs \rfloor}{E, T, E' \mid_{\text{prop}}^{\text{liss}} \ lhs} \quad A, A' \mid_{\text{prop}}^{\text{prop}} \ rhs \ \psi \text{ ev}}{E, A, T, E', A' \mid_{\text{prop}}^{\text{proc}} \ proc \ \psi \text{ ev}}$$

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

$$T = \{event\} \cup \{ev \mid \frac{E, A, T, E', A' \mid_{\text{prop}}^{\text{proc}} proc \Downarrow ev}{E, A, T \mid_{\text{update}} proc \Downarrow E', A'}$$
 only says t is not triggered

a: 0 =: y;

$$(x/y > 10) -> t$$

 $x \in free(e)$ Assign $\forall x \in T$ $E, E' \vdash_{exp} e \Downarrow tr$

$$egin{align*} proc(r_{\chi}.x) &= \lfloor x \colon \mathit{lhs} \ -><_> \mathit{rhs} \rfloor & r_{\chi}.x \in A' \\ E, T, E' & \vdash_{\mathrm{prop}}^{\mathrm{lhs}} \mathit{lhs} & A, A' & \vdash_{\mathrm{prop}}^{\mathrm{rhs}} \mathit{rhs} & v \end{aligned}$$
 $E, A, T, E', A' & \vdash_{\mathrm{prop}}^{\mathrm{proc}} \mathit{proc} & v \end{aligned}$

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

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

$$E, T, E', A' \vdash_{safe} proc_{\epsilon}$$

$$E, A, T \vdash_{update} proc \Downarrow E', A'$$

$$E, A \vdash_{reset} proc(event) \Downarrow E', A', (T \cap \{Trigger \ p \mid Spike \ p \in proc\})$$

$$\begin{split} T &= \{event\} \cup \{ev \mid E, A, T, E', A' \vdash_{loo}^{\text{time}} proc \ \downarrow ev\} \\ &= \underbrace{F, T, E', A' \vdash_{loo}^{\text{time}} proc}_{E, A, T \vdash_{loo}^{\text{time}}} proc \ \downarrow E', A' \\ \hline E, A \vdash_{loo}^{\text{time}} proc (event) \ \downarrow E', A', (T \cap \{\text{Trigger } p \mid \text{Spike } p \in proc\}) \end{split}$$

$$\frac{\forall x \in \mathsf{free}(e), \ \forall v, \ \mathsf{Assign} \ v \ x \notin T}{E, T, E' \vdash_{\mathsf{safe}} (e)?} \qquad \frac{E, E' \vdash_{\mathsf{exp}} e \Downarrow v \qquad v \in \{\mathsf{true}, \mathsf{false}\}}{E, T, E' \vdash_{\mathsf{safe}} (e)?}$$

$$\frac{E, T, E' \vdash_{\mathsf{safe}} \mathsf{lhs}}{E, T, E', A' \vdash_{\mathsf{safe}}^{\mathsf{proc}} x : \mathsf{lhs} \rightarrow < > \mathsf{rhs}}$$

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

```
T = \{event\} \cup \{ev \mid E, A, T, E', A' \mid_{prop}^{proc} proc \downarrow ev\}
E, T, E', A' \mid_{proc}^{proc} proc \downarrow
E, A, T \mid_{quino} proc \downarrow E', A'
E, A \mid_{max} proc(event) \downarrow E', A', (T \cap \{Trigger \ p \mid Spike \ p \in proc\})
```

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

$$\frac{\forall x \in \mathsf{free}(e), \ \forall v, \ \underset{\mathsf{Assign}}{\mathsf{Assign}} \ \mathsf{v} \ \mathsf{v} \notin \mathsf{T}}{\mathsf{E}, \mathsf{T}, \mathsf{E}' \vdash_{\mathsf{safe}}^{\mathsf{lhs}} (e)?}$$

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

$$\frac{E, T, E' \vdash_{\text{safe}}^{\text{lhs}} lhs}{E, T, E', A' \vdash_{\text{safe}}^{\text{proc}} x: lhs \rightarrow < > rhs}$$

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

```
T = \{event\} \cup \{ev \mid E, A, T, E', A' \mid_{p=p}^{p=p} proc \Downarrow ev\}
E, T, E', A' \mid_{p=p}^{p=p} proc \Downarrow E', A'
E \land A \vdash T \mid_{p=p}^{p=p} proc \Downarrow E', A'
E \land A \vdash_{p=p}^{p=p} proc \downarrow E', A'
E \land A \vdash_{p=p}^{p=p} proc \downarrow E', A' \land (T \cap \{Trigger \ p \mid Spike \ p \in proc\})
```

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

$$\frac{\forall x \in \mathsf{free}(e), \ \forall v, \ \underset{\mathsf{safe}}{\mathsf{Assign}} \ \mathsf{v} \ \mathsf{x} \notin T}{E, T, E' \vdash_{\mathsf{safe}}^{\mathsf{lhs}} (e)?}$$

$$\frac{E, E'|_{\text{exp}} e \Downarrow v \quad v \in \{\text{true}, \text{false}\}}{E, T, E'|_{\text{safe}}^{\text{libs}} (e)?}$$

$$\frac{E, T, E', A' \vdash_{\text{safe}}^{\text{Proc}} \textit{lhs}}{E, T, E', A' \vdash_{\text{safe}}^{\text{Proc}} x: \textit{lhs} -><_> \textit{rhs}}$$

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

$$\begin{split} T &= \{event\} \cup \{ev \mid E, A, T, E', A' \vdash^{me}_{proc} proc \ \downarrow ev\} \\ &= \underbrace{E, T, E', A' \vdash^{e}_{proc} proc}_{E, A, T \vdash^{e}_{proc} proc \ \downarrow E', A'}_{E, A \vdash_{men} proc (event) \ \downarrow E', A', (T \cap \{\text{Trigger } p \mid \text{Spike } p \in proc\}) \end{split}$$

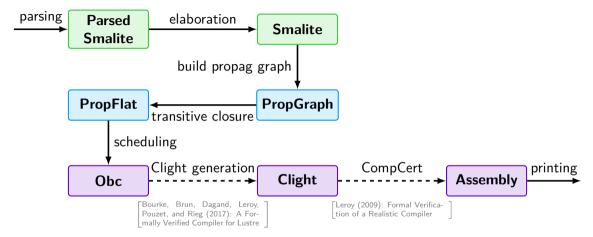
$$\frac{\forall x \in \mathsf{free}(e), \ \forall v, \ \mathsf{Assign} \ v \ x \notin T}{E, T, E' \vdash_{\mathsf{safe}} (e)?} \qquad \frac{E, E' \vdash_{\mathsf{exp}} e \Downarrow v \qquad v \in \{\mathsf{true}, \mathsf{false}\}}{E, T, E' \vdash_{\mathsf{safe}} (e)?}$$

$$\frac{E, T, E' \vdash_{\mathsf{safe}} \mathsf{lhs}}{E, T, E', A' \vdash_{\mathsf{safe}} \mathsf{x} : \mathsf{lhs} \rightarrow <_> \mathsf{rhs}}$$

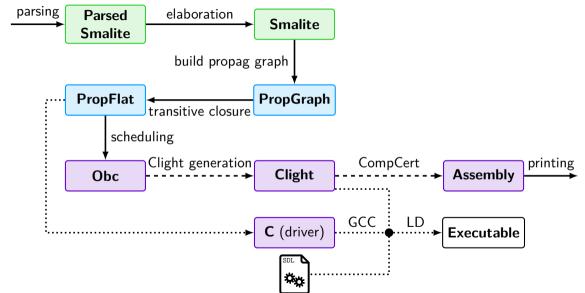
$$\frac{\text{Trigger } r_{x}.x \in T \implies E, E' \vdash_{\exp} e \Downarrow v}{E, T, E', A' \vdash_{\text{safe}}^{\text{proc}} (x: e =: y)_{r_{x}}}$$

Future Work – A verified compiler for Smala

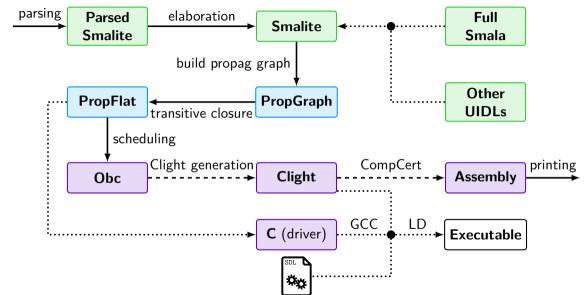
A prototype compiler for Smalite



A prototype compiler for Smalite



A prototype compiler for Smalite



```
Component root {
  Int count 3:
  Spike zero; (count == 0) -> zero;
  Frame f ("ICE<sub>11</sub>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
```

```
Component root {
  Int count 3:
  Spike zero; (count == 0) -> zero;
  Frame f ("ICE<sub>11</sub>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
```

```
Component root {
  Int count 3:
  Spike zero; (count == 0) -> zero;
  Frame f ("ICE<sub>11</sub>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
```

```
Component root {
  Int count 3:
  Spike zero; (count == 0) -> zero;
  Frame f ("ICE<sub>11</sub>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
```

```
Component root {
  Int count 3:
  Spike zero; (count == 0) -> zero;
  Frame f ("ICE<sub>11</sub>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
```

```
Component root {
  Int count 3:
  Spike zero; (count == 0) -> zero;
  Frame f ("ICE<sub>11</sub>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 ("dvn_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
```

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
```

Conclusion

What we did:

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

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 :)