Synthesizing Safe State Machines from Esterel

Steffen Prochnow, Claus Traulsen, Reinhard v. Hanxleden

Christian-Albrechts Universität zu Kiel
Department of Computer Science and Applied Mathematics
Real-Time Systems and Embedded Systems Group
rvh@informatik.uni-kiel.de

LCTES’06, Ottawa, June 2006
Introduction

- **Esterel**: Synchronous language for programming reactive systems [Berry 1984]
- **Statecharts**: Visual formalism to model reactive systems [Harel 1987]
- **Safe State Machines (SSMs)**: Statechart dialect with fully synchronous model of computation [André 1996/2003]
Introduction

- **Esterel**: Synchronous language for programming reactive systems [Berry 1984]
- **Statecharts**: Visual formalism to model reactive systems [Harel 1987]
- **Safe State Machines (SSMs)**: Statechart dialect with fully synchronous model of computation [André 1996/2003]
- Visual formalism facilitates understanding complex behaviors
Introduction

- **Esterel**: Synchronous language for programming reactive systems [Berry 1984]
- **Statecharts**: Visual formalism to model reactive systems [Harel 1987]
- **Safe State Machines (SSMs)**: Statechart dialect with fully synchronous model of computation [André 1996/2003]
- Visual formalism facilitates understanding complex behaviors—*but there are also drawbacks wrt textual formalisms!*
Introduction

- **Esterel**: Synchronous language for programming reactive systems [Berry 1984]
- **Statecharts**: Visual formalism to model reactive systems [Harel 1987]
- **Safe State Machines (SSMs)**: Statechart dialect with fully synchronous model of computation [André 1996/2003]
- Visual formalism facilitates understanding complex behaviors—*but there are also drawbacks wrt textual formalisms!*

*The work presented here aims to make the best of both worlds accessible to the system developer*
Overview

Introduction
Example: ABRO/ABCRO
Textual vs. Graphical—Editing Speed
Textual vs. Graphical—Tracability

From Esterel to Safe State Machines
Step 1: Transform Esterel to SSM
Step 2: Reduce to Fully Graphical SSM
Step 3: Optimizations

Evaluation
Experimental Results
Summary
Example: ABRO/ABCRO

module ABRO:
  input A, B, R;
  output O;
  loop
    [ await A || await B ];
    emit O;
  each R
end module
module ABRO:
input A, B, R;
output O;
loop
  [ await A ||
    await B ];
  emit O;
each R
end module
Example: ABRO/ABCRO

module ABCRO:
input A, B, C, R;
output O;
loop
  [ 
    await A \\
    || \\
    await B \\
    || \\
    await C 
  ];
emit O;
each R
end module
Which is faster?

Editing Esterel:

1. move cursor to position
2. type “, C”
3. move cursor to position
4. type “|| await C”

Editing Safe State Machine:

1. make room: shift neighbor states, enlarge parent state
2. click on “add state”
3. move mouse to location and place new state
4. click on “add state”
5. move mouse to location and place new state
6. double click on new state, toggle terminal field
7. click on “initial state”
8. move mouse to location and place new initial state
9. click on “transition”
10. move mouse to location of initial state
11. press left mouse button and keep pressed until reaching state
12. click on “transition”
13. move mouse to location of state
14. press left mouse button and keep pressed until reaching terminal state
15. double click on transition
16. write “C” in trigger field
17. press “OK”
Which is faster?

Editing Esterel:

1. move cursor to position
2. type “, C”
3. move cursor to position
4. type “|| await C”

Editing Safe State Machine:

1. make room: shift neighbor states, enlarge parent state
2. click on “add state”
3. move mouse to location and place new state
4. click on “add state”
5. move mouse to location and place new state
6. double click on new state, toggle terminal field
7. click on “initial state”
8. move mouse to location and place new initial state
9. click on “transition”
10. move mouse to location of initial state
11. press left mouse button and keep pressed until reaching state
12. click on “transition”
13. move mouse to location of state
14. press left mouse button and keep pressed until reaching terminal state
15. double click on transition
16. write “C” in trigger field
17. press “OK”
18. click on “delimiter line”
19. move mouse to location and place delimiter line
Which is more tracable?

**Textual:**

```plaintext
1,2c1,2
< module ABRO:
< input A, B, R;
---
> module ABCRO:
> input A, B, C, R;
5c5
< [ await A || await B ];
---
> [ await A || await B || await C ];
```
Which is more tracable?

Textual:

\[
1,2c1,2
\]
\[
< \text{module ABRO:} \\
< \text{input A, B, R;} \\
--- \\
> \text{module ABCRO:} \\
> \text{input A, B, C, R;} \\
5c5 \\
< \begin{array}{c}
[ \text{await A} \text{| wait B }]
\end{array} \\
--- \\
> \begin{array}{c}
[ \text{await A} \text{| wait B }| \text{ wait C }]
\end{array}
\]

Graphical:
Overview

Introduction

Example: ABRO/ABCRO
Textual vs. Graphical—Editing Speed
Textual vs. Graphical—Tracability

From Esterel to Safe State Machines

Step 1: Transform Esterel to SSM
Step 2: Reduce to Fully Graphical SSM
Step 3: Optimizations

Evaluation

Experimental Results
Summary
Step 1: Transform Esterel program into SSM with textual macro states
From Esterel to Safe State Machines

**Step 1:** Transform Esterel program into SSM with textual macro states

**Step 2:** Iteratively apply reduction rules to transform Esterel constructs into graphical components
From Esterel to Safe State Machines

Step 1: Transform Esterel program into SSM with textual macro states

Step 2: Iteratively apply reduction rules to transform Esterel constructs into graphical components

Step 3: Optimize SSM
Step 1: Transform Esterel to SSM

module mod_name:
    input $l_1, \ldots, l_n$;
    output $O_1, \ldots, O_m$;
    $p$
end module
Step 1: Transform Esterel to SSM

```plaintext
module mod_name:
  input l_1, ..., l_n;
  output O_1, ..., O_m;
  p
end module
```

Synthesizing Safe State Machines from Esterel
Step 2: Reduce to Fully Graphical SSM

Signal emission

\[ \text{emit } \text{exp} \implies \text{emit } \text{I } \text{/exp} \]
Step 2: Reduce to Fully Graphical SSM

Signal emission

$$emit\ s(exp) \Rightarrow I/s(exp)$$

Sequence

$$p_1; \ldots ;p_n \Rightarrow sequence\ I/p_1\ \ldots \\ p_n$$
Step 2: Reduce to Fully Graphical SSM

Signal emission

\[
\text{emit } s(\text{exp}) \quad \Rightarrow 
\]

Sequence

\[
p1; \ldots ; pn \quad \Rightarrow 
\]

Signal awaiting

\[
\text{await } de 
\]
Step 2: Reduce to Fully Graphical SSM

Weak Abortion

`weak abort
  p
  when
    case de_1 do q1
    ..
    case de_n do qn
  end abort`

\[\Rightarrow\]

\[\text{weakabort}\]

\[\begin{array}{ll}
\text{weak abort} & \\
p & \\
\text{when} & \\
\text{case de}_1 & \text{do q1} \\
\quad & .. \\
\text{case de}_n & \text{do qn} \\
\text{end abort} \\
\end{array}\]

+ 19 additional rules
Translation of traps not trivial—see paper
Example: ABRO
Applying Rule (module)

```plaintext
module ABRO:
  input A, B, R;
  output O;
  loop
    [ 
      await A
    ||
      await B
    ];
  emit O;
  each R
end module
```
Example: ABRO

Applying Rule (module)

```
module ABRO:
    input A, B, R;
    output O;
    loop
        [ await A || await B ];
    emit O;
    each R
end module
```
Example: ABRO

Applying Rule (loopeach)
Example: ABRO

Applying Rule (loopeach)
Example: ABRO

Applying Rule (sequence)
Example: ABRO

Applying Rules (parallel) + (emit)
Example: ABRO

Applying Rule (simple await)
Step 3: Optimizations

Motivation

- Automatic synthesis produces “verbose” modules
- However, also human modelers (esp. novices) may produce sub-optimal models
Step 3: Optimizations

Motivation

- Automatic synthesis produces “verbose” modules
- However, also human modelers (esp. novices) may produce sub-optimal models

Notes:

- It may be a matter of style/opinion what “optimal” means
- However, consistency in style is desirable in any case—and standardized optimization rules help to achieve this
Step 3: Optimizations

Removing Simple States

Precondition: State must be transient
Step 3: Optimizations

Flattening Hierarchy

Preconditions:

- no abort originate from S
- S has no local signals

+ further rules to remove conditionals, combine terminal states, remove normal terminations
Overview

Introduction
Example: ABRO/ABCRO
Textual vs. Graphical—Editing Speed
Textual vs. Graphical—Tracability

From Esterel to Safe State Machines
Step 1: Transform Esterel to SSM
Step 2: Reduce to Fully Graphical SSM
Step 3: Optimizations

Evaluation
Experimental Results
Summary
<table>
<thead>
<tr>
<th>Model</th>
<th>Lines of Code (LoC)</th>
<th>States/Pseudo-states</th>
<th>Transitions</th>
<th>Total Graphical Elements (GEs)</th>
<th>GE5/LoC</th>
<th>States/Pseudo-states</th>
<th>Transitions</th>
<th>Total Graphical Elements (GEs)</th>
<th>GE5/LoC</th>
<th>SSM Reduction Factor</th>
<th>Transformation [ms]</th>
<th>Optimization [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABRO</td>
<td>7</td>
<td>12/8</td>
<td>12</td>
<td>32</td>
<td>4.57</td>
<td>8/4</td>
<td>8</td>
<td>20</td>
<td>2.86</td>
<td>0.63</td>
<td>861</td>
<td>56</td>
</tr>
<tr>
<td>SCHIZOPHRENIA</td>
<td>10</td>
<td>13/9</td>
<td>14</td>
<td>36</td>
<td>3.60</td>
<td>4/3</td>
<td>6</td>
<td>13</td>
<td>1.30</td>
<td>0.36</td>
<td>838</td>
<td>81</td>
</tr>
<tr>
<td>REINCARNATION</td>
<td>25</td>
<td>27/17</td>
<td>28</td>
<td>72</td>
<td>2.88</td>
<td>5/2</td>
<td>6</td>
<td>13</td>
<td>0.52</td>
<td>0.18</td>
<td>642</td>
<td>83</td>
</tr>
<tr>
<td>JACKY1</td>
<td>27</td>
<td>31/19</td>
<td>33</td>
<td>83</td>
<td>3.07</td>
<td>11/5</td>
<td>12</td>
<td>28</td>
<td>1.04</td>
<td>0.34</td>
<td>913</td>
<td>93</td>
</tr>
<tr>
<td>RUNNER</td>
<td>55</td>
<td>39/24</td>
<td>42</td>
<td>105</td>
<td>1.91</td>
<td>21/11</td>
<td>25</td>
<td>57</td>
<td>1.04</td>
<td>0.54</td>
<td>725</td>
<td>160</td>
</tr>
<tr>
<td>TOKENRING3</td>
<td>79</td>
<td>77/61</td>
<td>91</td>
<td>229</td>
<td>2.90</td>
<td>15/20</td>
<td>38</td>
<td>73</td>
<td>0.92</td>
<td>0.32</td>
<td>733</td>
<td>278</td>
</tr>
<tr>
<td>GREYCOUNTER</td>
<td>82</td>
<td>211/148</td>
<td>254</td>
<td>613</td>
<td>7.48</td>
<td>42/50</td>
<td>106</td>
<td>198</td>
<td>2.41</td>
<td>0.32</td>
<td>789</td>
<td>593</td>
</tr>
<tr>
<td>ABCD</td>
<td>101</td>
<td>231/130</td>
<td>250</td>
<td>611</td>
<td>6.05</td>
<td>78/41</td>
<td>97</td>
<td>216</td>
<td>2.14</td>
<td>0.35</td>
<td>943</td>
<td>731</td>
</tr>
<tr>
<td>TOKENRING10</td>
<td>247</td>
<td>245/194</td>
<td>294</td>
<td>733</td>
<td>2.97</td>
<td>43/62</td>
<td>122</td>
<td>227</td>
<td>0.92</td>
<td>0.31</td>
<td>1267</td>
<td>736</td>
</tr>
<tr>
<td>MEJIA</td>
<td>555</td>
<td>374/246</td>
<td>414</td>
<td>1034</td>
<td>1.86</td>
<td>127/76</td>
<td>181</td>
<td>384</td>
<td>0.69</td>
<td>0.37</td>
<td>3085</td>
<td>1266</td>
</tr>
<tr>
<td>TCINT</td>
<td>687</td>
<td>475/285</td>
<td>543</td>
<td>1303</td>
<td>1.90</td>
<td>163/81</td>
<td>221</td>
<td>465</td>
<td>0.68</td>
<td>0.36</td>
<td>3382</td>
<td>1310</td>
</tr>
<tr>
<td>ATDS-100</td>
<td>948</td>
<td>961/558</td>
<td>1092</td>
<td>2611</td>
<td>2.75</td>
<td>352/184</td>
<td>504</td>
<td>1040</td>
<td>1.10</td>
<td>0.40</td>
<td>7046</td>
<td>2760</td>
</tr>
<tr>
<td>WW</td>
<td>1088</td>
<td>342/228</td>
<td>386</td>
<td>965</td>
<td>0.89</td>
<td>102/85</td>
<td>177</td>
<td>364</td>
<td>0.33</td>
<td>0.38</td>
<td>4470</td>
<td>1053</td>
</tr>
<tr>
<td>TOKENRING50</td>
<td>1207</td>
<td>1205/954</td>
<td>1454</td>
<td>3613</td>
<td>2.99</td>
<td>203/302</td>
<td>602</td>
<td>1107</td>
<td>0.92</td>
<td>0.31</td>
<td>6608</td>
<td>8148</td>
</tr>
<tr>
<td>TOKENRING100</td>
<td>2407</td>
<td>2405/1904</td>
<td>2904</td>
<td>7213</td>
<td>3.00</td>
<td>403/602</td>
<td>1202</td>
<td>2207</td>
<td>0.92</td>
<td>0.31</td>
<td>20910</td>
<td>25528</td>
</tr>
<tr>
<td>MCA200</td>
<td>7269</td>
<td>5159/3931</td>
<td>5947</td>
<td>15037</td>
<td>2.07</td>
<td>179/925</td>
<td>1794</td>
<td>2898</td>
<td>0.40</td>
<td>0.19</td>
<td>61510</td>
<td>100594</td>
</tr>
</tbody>
</table>
Summary

Graphical vs. textual modeling

- Graphical models nice to browse/simulate
- Textual models much faster to edit, better to maintain
Summary

Graphical vs. textual modeling

- Graphical models nice to browse/simulate
- Textual models much faster to edit, better to maintain

Enablers to take best of both worlds

1. Matching of Esterel semantics and SSM semantics
2. Ability to do automatic layout

Thanks!
Summary

Graphical vs. textual modeling
  - Graphical models nice to browse/simulate
  - Textual models much faster to edit, better to maintain
Enablers to take best of both worlds
  1. Matching of Esterel semantics and SSM semantics
  2. Ability to do automatic layout
Have implemented this in KIEL framework
  - Edit Esterel code, simulate SSMs
  - Facilitates tracability
  - Allows designer to focus on content, rather than layout
Summary

Graphical vs. textual modeling
- Graphical models nice to browse/simulate
- Textual models much faster to edit, better to maintain

Enablers to take best of both worlds
1. Matching of Esterel semantics and SSM semantics
2. Ability to do automatic layout

Have implemented this in KIEL framework
- Edit Esterel code, simulate SSMs
- Facilitates tracability
- Allows designer to focus on content, rather than layout

Thanks! Questions/Comments?
Translating Traps

The Trap Reduction Rule

```
trap t1,...,tm in
  p
handle ehe_1 do p1
...
handle ehe_n do pn
end trap
```

\[ \Rightarrow \]

Translating Traps

The Trap Reduction Rule

Diagram showing the trap reduction rule with states and transitions.
Translating Traps
Detecting Instantaneous Loops

module LOOP_WITH_TRAP:
    loop
        trap T in
            pause;
            exit T
        end trap
    end loop
end module
Translating Traps

Detecting Instantaneous Loops

Problem: Compiler (falsely) assumes that loop body may be instantaneous
Translating Traps
Detecting Instantaneous Loops

Solution: Add parallel pause statement
Translating Traps
Detecting Instantaneous Loops

module LOOP_WEAK_ABORT:
    loop
        signal T in
            weak abort
            pause;
        emit T
        when immediate T
    end signal
||
        pause
    end loop
end module