The Columbia Esterel Compiler

Stephen A. Edwards, Cristian Soviani, and Jia Zeng

Columbia University
Yet Another Esterel Compiler?

Not quite.
First complete open-source Esterel compiler.
Designed for research; modular like the CMA compilers.
What happened to ESUIF?

Collapsed under its own weight.
Less and less of SUIF became useful as development progressed.
SUIF does not compile under gcc 3.0 (standard library problems).
Requiring people to install SUIF to use ESUIF prohibitive.
Automata Compilers

Esterel is finite-state; build an automata:

```
loop
  emit A; await C;
  emit B; pause
end

switch (s) {
  case 0: A = 1; s = 1; break;
  case 1: if (C) {
           B = 1; s = 0;
        } break;
}
```

V1, V2, V3 (INRIA/CMA)

Very fast code for programs with few states.
Exponential number of states.
Netlist-based Compilers

loop
emit A; await C;
emit B; pause
end

entry

A = entry || s2q;
cf = !C && s1q;
s1d = cf || A;
B = s2d = C && s1q;

Clean semantics, scales well, but inefficient.
Can be 100 times slower than automata code.
Discrete-Event Based Compilers

SAXO-RT [Weil et al. 2000] Divides program into event functions dispatched by fixed scheduler

unsigned curr = 0x1;
unsigned next = 0;

static void f1() {
    A = 1;
    curr &= ~0x1; next |= 0x2;
}

static void f2() {
    if (!C) return;
    B = 1;
    curr &= ~0x2; next |= 0x1;
}

void tick() {
    if (curr & 0x1) f1();
    if (curr & 0x2) f2();
    curr |= next;
    next = 0;
}
The Columbia Esterel Compiler

Back Ends

Hardware (BLIF/Verilog). Cristian Soviani.


Software (PDG-based Static C). Jia Zeng.
module Example:
input I, S;
output O, Q;
signal R, A in
every S do
  await I;
  weak abort
  sustain R
  when immediate A;
  emit O
  loop
    pause; pause;
    present R then
    emit A
    end present
  end loop
end every
end signal
end module
GRC Selection Tree
GRC Control-flow graph
After Clustering
Generated code (1)

```c
#define sched1a next1 = head1, head1 = &&C1a
#define sched1b next1 = head1, head1 = &&C1b
#define sched2 next2 = head1, head1 = &&C2
#define sched3a next3 = head1, head1 = &&C3a
#define sched3b next3 = head1, head1 = &&C3b
#define sched4 next4 = head2, head2 = &&C4
#define sched5a next5 = head3, head3 = &&C5a
#define sched5b next5 = head3, head3 = &&C5b
#define sched5c next5 = head3, head3 = &&C5c
#define sched6a next6 = head4, head4 = &&C6a
#define sched6b next6 = head4, head4 = &&C6b
#define sched6c next6 = head4, head4 = &&C6c
#define sched7a next7 = head5, head5 = &&C7a
#define sched7b next7 = head5, head5 = &&C7b
```
int cycle() {
    void *next1;
    void *next2;
    void *next3;
    /* other next pointers */

    void *head1 = &&END_LEVEL_1;
    void *head2 = &&END_LEVEL_2;
    /* other level pointers */

    if (s1) {s1 = 0; goto N26; } else {
        s1 = 0;
        if (S) {
            s2 = 1; code0 = -1;
            sched7a; sched1b; sched3b;
            s3 = 2; sched6b;
        } else {
            The Columbia Esterel Compiler – p. 18/24
if (s2) {
    s2 = 1;
    code0 = -1;
    sched7a; sched1a; sched3a;
    switch (s3) {
    case 0: sched6c; break;
    case 1:
        s3 = 1; code1 = -1;
        sched6a; sched2; goto N38;
    case 2:
        if (I) {
            s3 = 1; code1 = -1;
            sched6a; sched5a;
        }
    N38: R = 1; code1 &= -(1 << 1);
    }else {s3 = 2; sched6b; }
    break;
}else {
    N26: s2 = 0; sched7b;
}}
goto *head1;
Generated code (4)

C1a: if (s5) Q = 1;
C1b: if (R) s5 = 1;
else s5 = 0;
code0 &= -(1 << 1);
goto *next1;

C2: if (s6) sched4;
else s6 = 0;
goto *next2;

C3a: if (s4) s4 = 0;
else {
    if (R) A = 1;
C3b: s4 = 1;
}
code0 &= -(1 << 1);
goto *next3;

END_LEVEL1: goto *head2;
# Linked Lists — initial state

<table>
<thead>
<tr>
<th>Level 0</th>
<th>Level 1</th>
<th>Level 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>/* Cluster 0 */</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C1a: C1b:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>: goto *next1;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>: goto *next1;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>: goto *next1;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C4:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>: goto *next4;</td>
<td></td>
</tr>
</tbody>
</table>
### Linked Lists — schedule C3a

<table>
<thead>
<tr>
<th>Level 0</th>
<th>/* Cluster 0 */</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>goto *head1;</td>
</tr>
<tr>
<td>Level 1</td>
<td></td>
</tr>
<tr>
<td>C1a:</td>
<td>goto *next1;</td>
</tr>
<tr>
<td>C1b:</td>
<td>goto *next1;</td>
</tr>
<tr>
<td></td>
<td>goto *next1;</td>
</tr>
<tr>
<td>Level 2</td>
<td></td>
</tr>
<tr>
<td>C4:</td>
<td>goto *next4;</td>
</tr>
<tr>
<td>END_LEVEL1:</td>
<td>goto *head2;</td>
</tr>
<tr>
<td>C2:</td>
<td>goto *next2;</td>
</tr>
<tr>
<td>C3a:</td>
<td>goto *next3;</td>
</tr>
<tr>
<td>C3b:</td>
<td>goto *next3;</td>
</tr>
<tr>
<td>END_LEVEL2:</td>
<td>goto *head3;</td>
</tr>
</tbody>
</table>
### Linked Lists — schedule C1b

<table>
<thead>
<tr>
<th>Level</th>
<th>Code</th>
</tr>
</thead>
</table>
| 0     | /* Cluster 0 */  
    |     | goto *head1;  |
| 1     | C1a:  
    | C1b:  
    |     | goto *next1;  |
| 2     | C4:  
    |     | goto *next4;  |
|       | C2:  
    |     | goto *next2;  |
|       | C3a:  
    | C3b:  
    |     | goto *next3;  |
|       | END_LEVEL1:  
    |     | goto *head2;  |
|       | END_LEVEL2:  
    |     | goto *head3;  |
Linked Lists — schedule C4

Level 0
/* Cluster 0 */
::
goto *head1;

Level 1
C1a: 
::
goto *next1;
C1b: 
::
goto *next1;

Level 2
C4: 
::
goto *next4;

C2: 
::
goto *next2;

C3a: 
::
goto *next3;

C3b: 
::
goto *next3;

END_LEVEL1: goto *head2;

END_LEVEL2: goto *head3;
Results (seconds/1 000 000 cycles)

- atds
- Chorus
- mca200
- tcint
- Wristwatch
- CEC
- grc2c
- SAXO
- EC
- V3
## Statistics

<table>
<thead>
<tr>
<th>Example</th>
<th>Size</th>
<th>Clusters</th>
<th>Levels</th>
<th>C/L</th>
<th>Threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>atds</td>
<td>622</td>
<td>156</td>
<td>16</td>
<td>9.8</td>
<td>138</td>
</tr>
<tr>
<td>Chorus</td>
<td>3893</td>
<td>662</td>
<td>22</td>
<td>30.1</td>
<td>563</td>
</tr>
<tr>
<td>mca200</td>
<td>5354</td>
<td>148</td>
<td>15</td>
<td>9.9</td>
<td>135</td>
</tr>
<tr>
<td>tcint</td>
<td>357</td>
<td>101</td>
<td>19</td>
<td>5.3</td>
<td>85</td>
</tr>
<tr>
<td>Wristwatch</td>
<td>360</td>
<td>87</td>
<td>13</td>
<td>6.7</td>
<td>87</td>
</tr>
</tbody>
</table>
Conclusions

CEC: First open-source Esterel compiler
Hardware and software back ends
Event-driven back end like SAXO-RT
Contribution: clustering, levelizing, and linked lists
http://www.cs.columbia.edu/~sedwards