

# Discrete-Event Esterel Code Generation

Stephen A. Edwards

Columbia University

Octopi Workshop  
Chalmers University of Technology  
Gothenburg, Sweden  
December 2018

# Automata Compilers

Esterel is finite-state; build an automata:

```
loop                                switch (s) {
  emit A; await C;                  case 0: A = 1; s = 1; break;
  emit B; pause                      case 1: if (C) { B = 1; s = 0; }
end                                  break;
                                    }
```

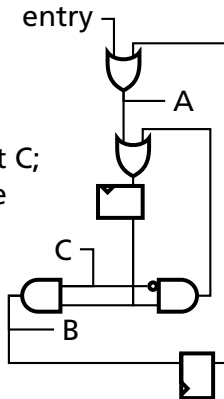
V1, V2, V3 (INRIA/CMA)

Very fast code for programs with few states.

Exponential number of states.

# Netlist-based Compilers

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



```
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;

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

                                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

Uses a variant of Potop-Butucaru's GRC format.

Both hardware and software back ends.

Supports most of Esterel V5.

Open-source, BSD license.

<http://www.cs.columbia.edu/~sedwards>

# Esterel program and GRC (selection tree + CFG)

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

||

loop

present R then

pause; emit Q

else

pause

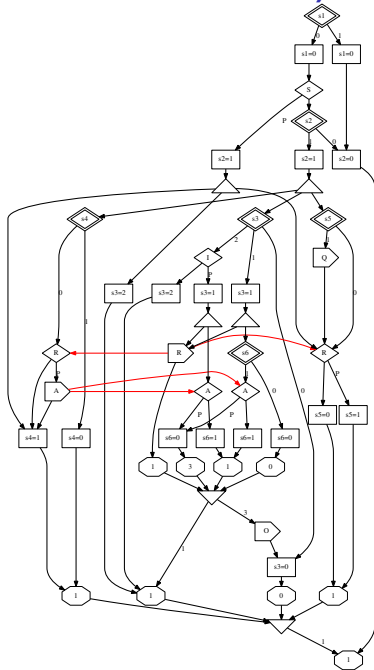
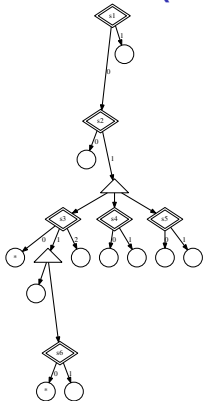
end present

end loop

end every

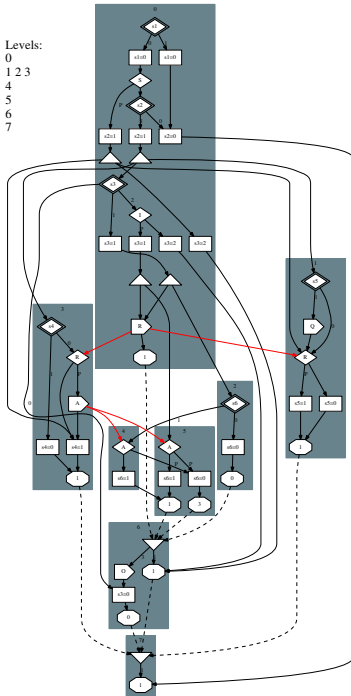
end signal

end module



# After Clustering

Levels:  
0  
1 2 3  
4  
5  
6  
7



## Generated code (1)

```
#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
```



## Generated code (2)

```
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 {
```

## Generated code (3)

```
    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)

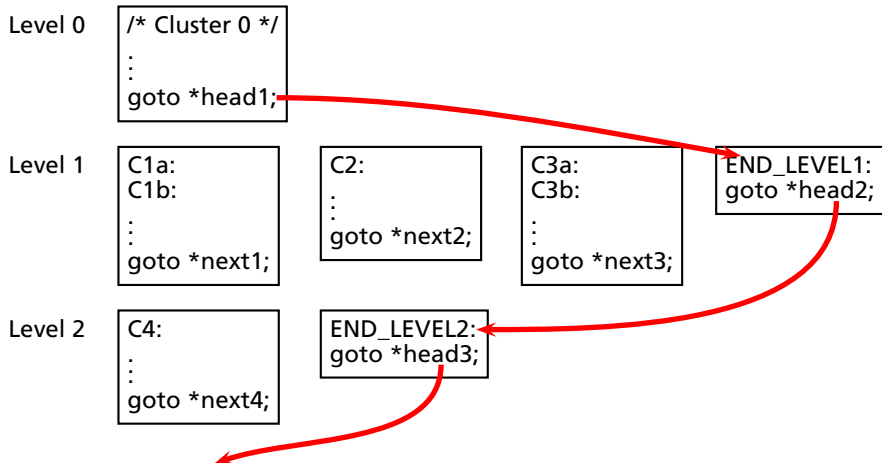
```
C1a:  if (s5) Q = 1;
C1b:  if (R) s5 = 1;
      else s5 = 0;
      code0 &= -(1 << 1);
      goto *next1;

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

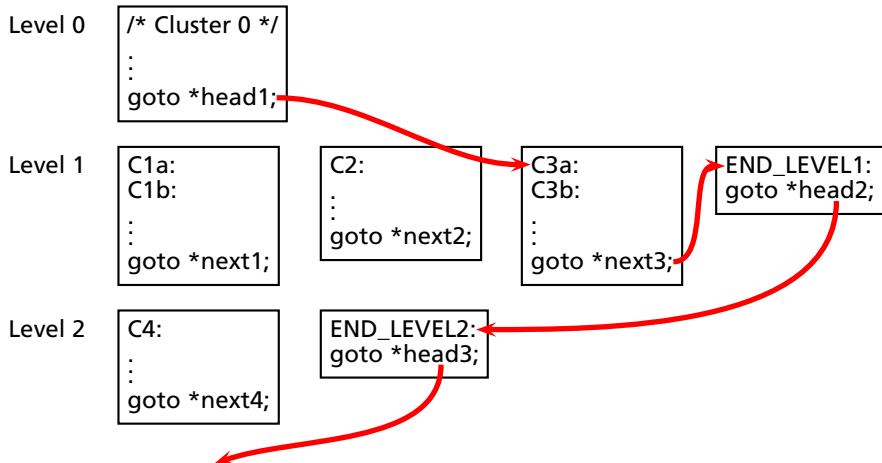
C3a:  if (s4) s4 = 0;
      else {
          if (R) A = 1;
C3b:   s4 = 1;
      }
      code0 &= -(1 << 1);
      goto *next3;

END_LEVEL1:  goto *head2;
```

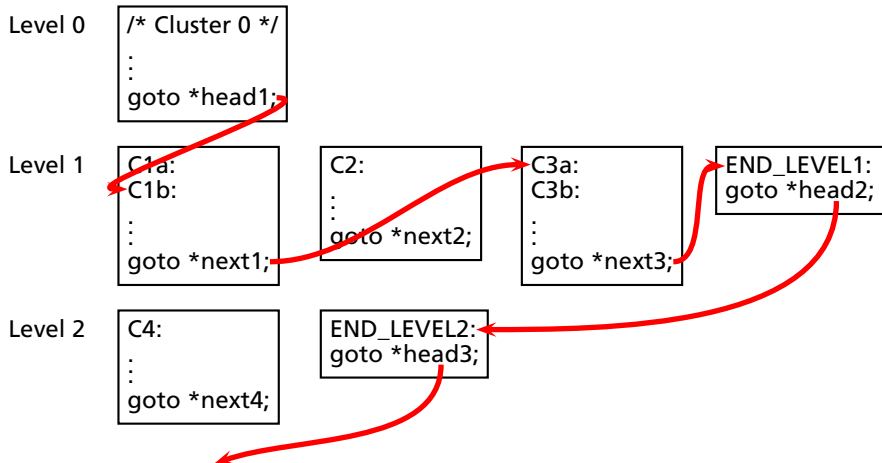
# Linked Lists — initial state



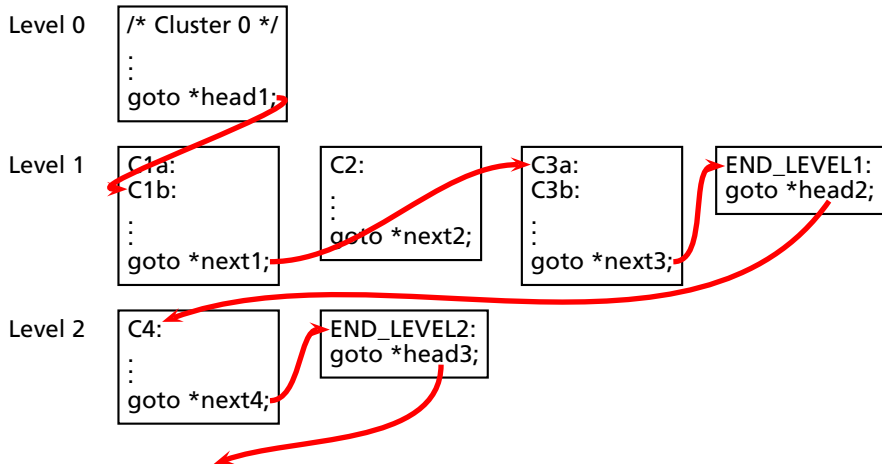
# Linked Lists — schedule C3a



# Linked Lists — schedule C1b



# Linked Lists — schedule C4



# Statistics

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