



# Compiling Esterel

**Stephen A. Edwards**

Department of Computer Science,  
Columbia University

[www.cs.columbia.edu/~sedwards](http://www.cs.columbia.edu/~sedwards)

[sedwards@cs.columbia.edu](mailto:sedwards@cs.columbia.edu)

# Outline

Introduction to Esterel and Existing Compilers

My Software Compiler [DAC 2000, TransCAD 2002]

My Hardware Compiler [SLAP 2002, IWLS 2002]

# The Esterel Language

Developed by Gérard Berry starting 1983

Originally for robotics applications

Imperative, textual language

Synchronous model of time like that in digital circuits

Concurrent

# An Example

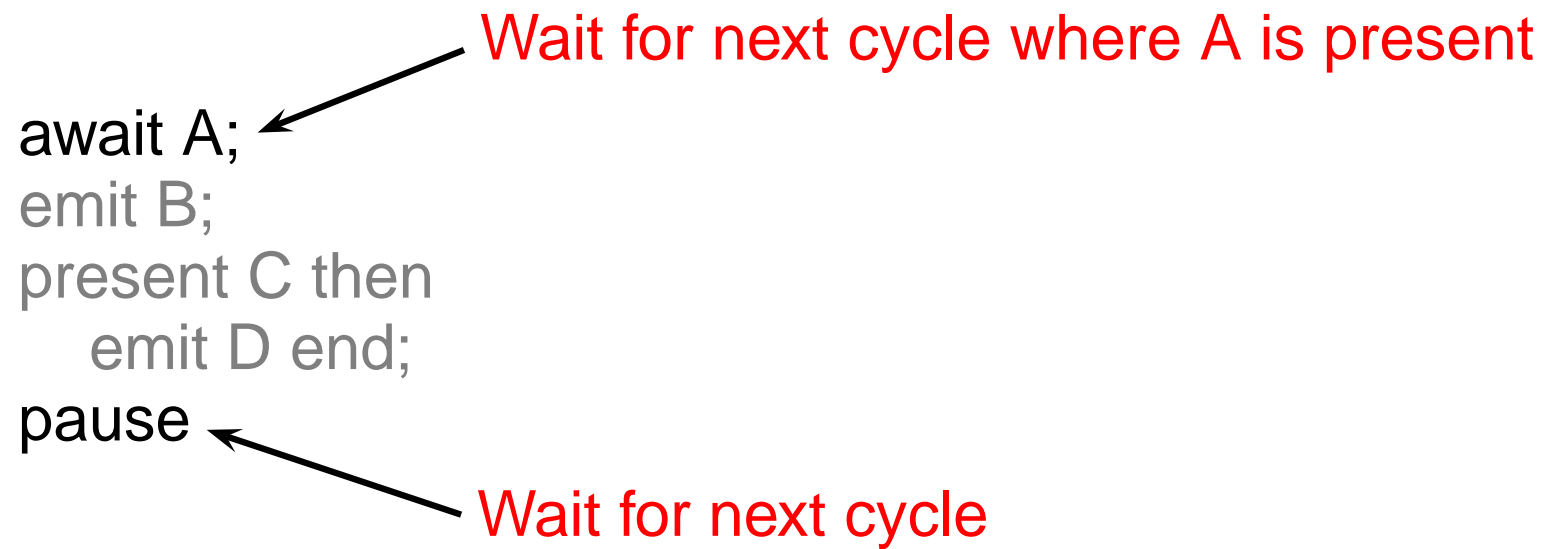
emit B; ← Force signal present in this cycle  
present C then ← Make D present if C is  
emit D end;

# An Example

```
await A;
emit B;
present C then
  emit D end;
pause
```

Wait for next cycle where A is present

Wait for next cycle



# An Example

```
loop ←———— Infinite Loop
  await A;
  emit B;
  present C then
    emit D end;
  pause
end
```

# An Example

```
loop
  await A;
  emit B;
  present C then
    emit D end;
  pause
end
```

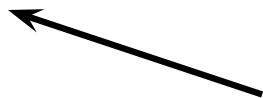
|| ←———— Run Concurrently

```
loop
  present B then
    emit C end;
  pause
end
```

# An Example

```
every R do
  loop
    await A;
    emit B;
    present C then
      emit D end;
    pause
  end
||
  loop
    present B then
      emit C end;
    pause
  end
end
```

Restart on R



# An Example

```
every R do
  loop
    await A;
    emit B;
    present C then
      emit D end
    pause
  end
||
  loop
    present B then
      emit C end;
    pause
  end
end
```

Same-cycle bidirectional communication



# An Example

```
every R do
  loop
    await A;
    emit B;
    present C then
      emit D end;
    pause
  end
||
  loop
    present B then
      emit C end;
    pause
  end
end
```

Good for hierarchical FSMs

Bad at manipulating data

Hardware Esterel variant  
proposed to address this

# Automata Compilers

Esterel is a finite-state language, so build an automata:

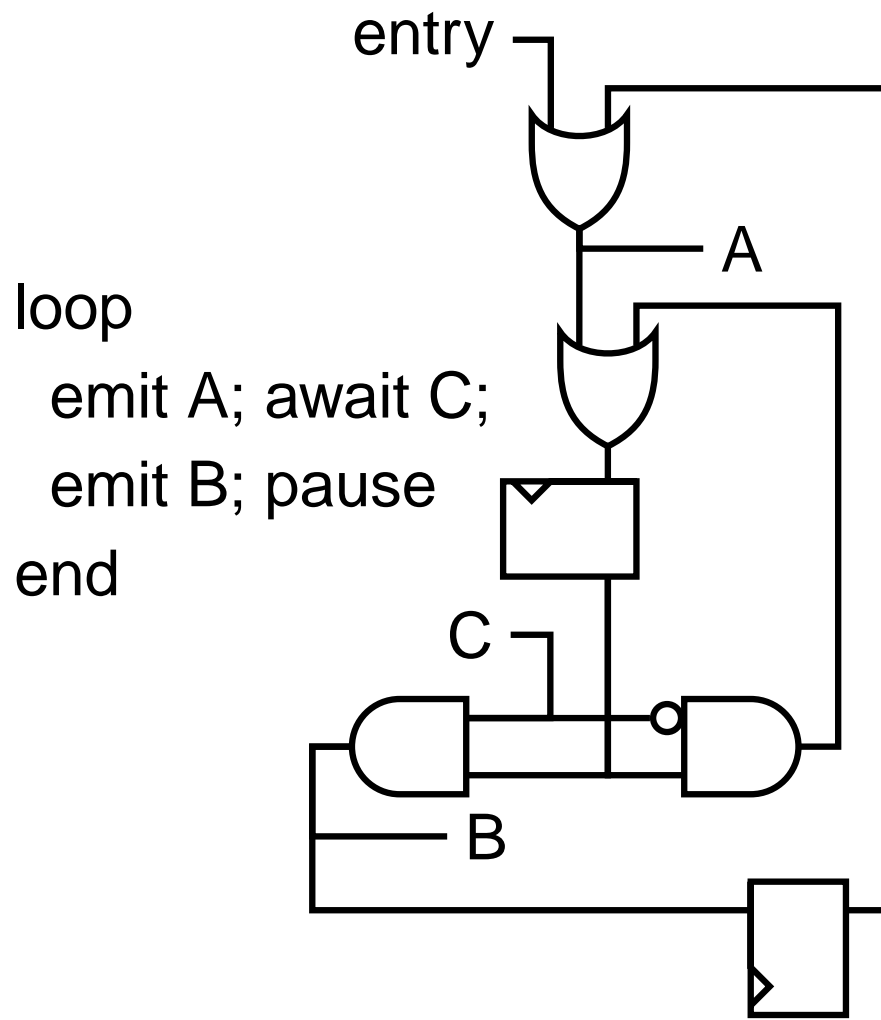
```
loop                switch (s) {
  emit A; await C;  case 0: A = 1; s = 1; break;
  emit B; pause     case 1: if (C) { B = 1; s = 0; } break;
end                 }
```

V1, V2, V3 (INRIA/CMA) [Berry, Gonthier 1992]

Fastest known code; great for programs with few states.

Does not scale; concurrency causes state explosion.

# 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 Esterel program into event functions dispatched by a fixed scheduler.

```

    unsigned curr = 0x1;
    unsigned next = 0;

    static void f1() {
        A = 1;
        curr &= ~0x1; next |= 0x2;
    }
loop
    emit A; await C;
    emit B; pause
end

    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;
}
```

# **My Esterel Compiler for Software**

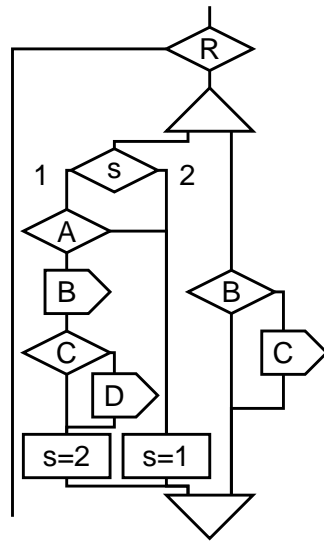
Presented at DAC 2000 (also TransCAD 2002)

Used inside Synopsys' CoCentric System Studio to  
generate control code

# Overview

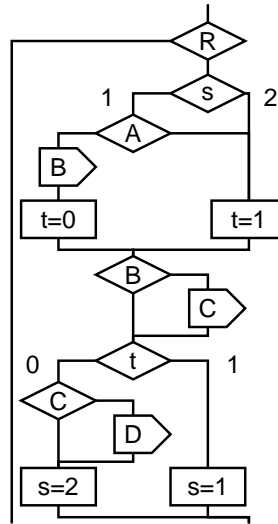
```

every R do
  loop
    await A;
    emit B;
    present C then
      emit D end;
    pause
  end
||
loop
  present B then
    emit C end;
  pause
end
end
  
```



Concurrent

CFG



CFG

```

if ((s0 & 3) == 1) {
  if (s) {
    s3 = 1; s2 = 1; s1 = 1;
  } else
    if (s1 >> 1)
      s1 = 3;
    else {
      if ((s3 & 3) == 1) {
        s3 = 2; t3 = L1;
      } else {
        t3 = L2;
      }
    }
}
  
```

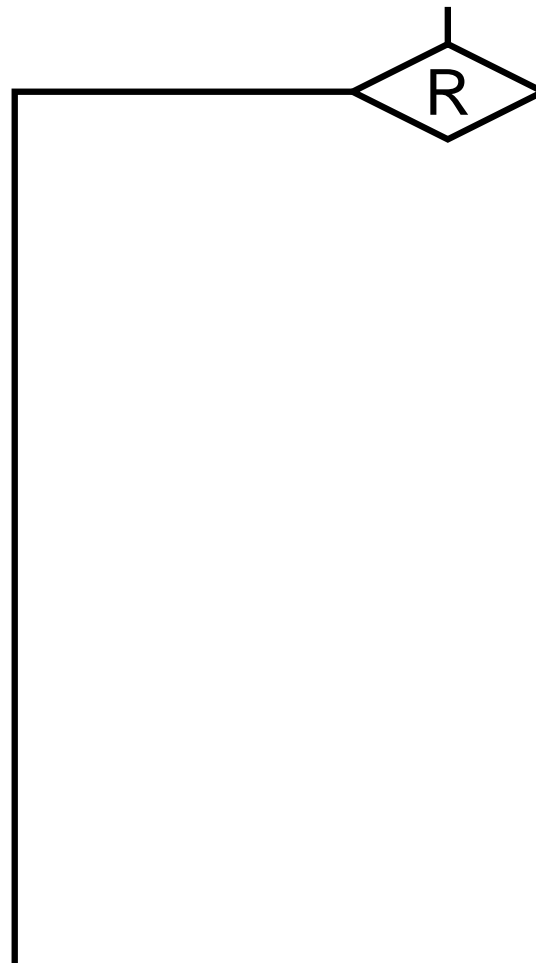
C code

Esterel

Source

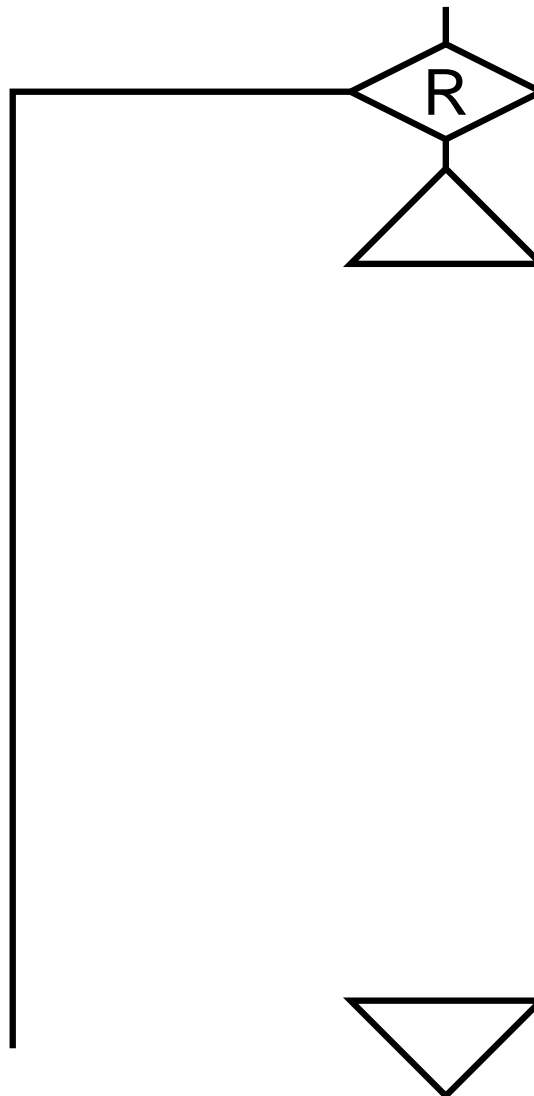
# Translate every

```
every R do
  loop
    await A;
    emit B;
    present C then
      emit D end;
    pause
  end
||
  loop
    present B then
      emit C end;
    pause
  end
end
```



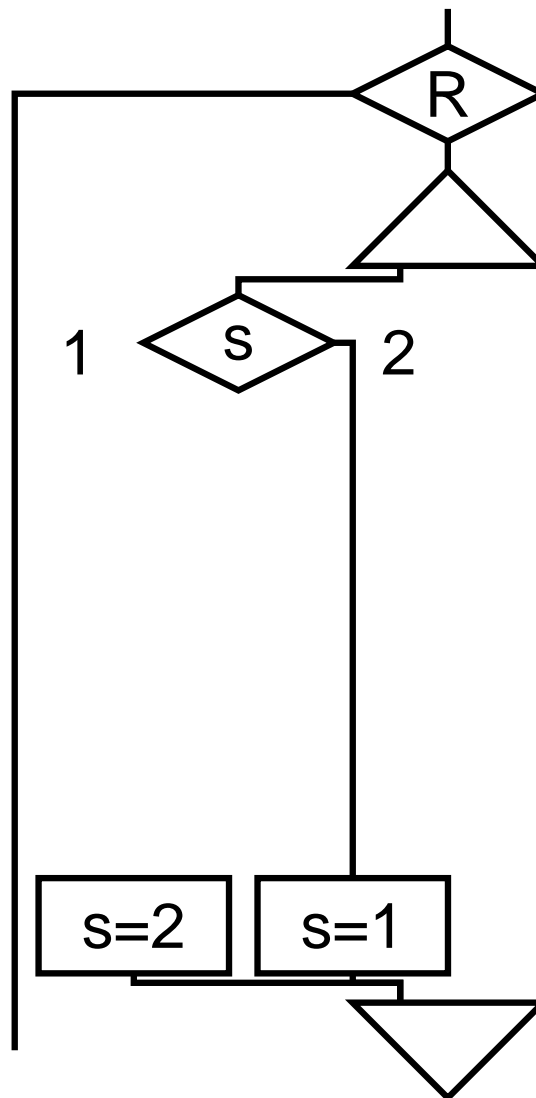
# Add Threads

```
every R do
  loop
    await A;
    emit B;
    present C then
      emit D end;
    pause
  end
||
  loop
    present B then
      emit C end;
    pause
  end
end
```



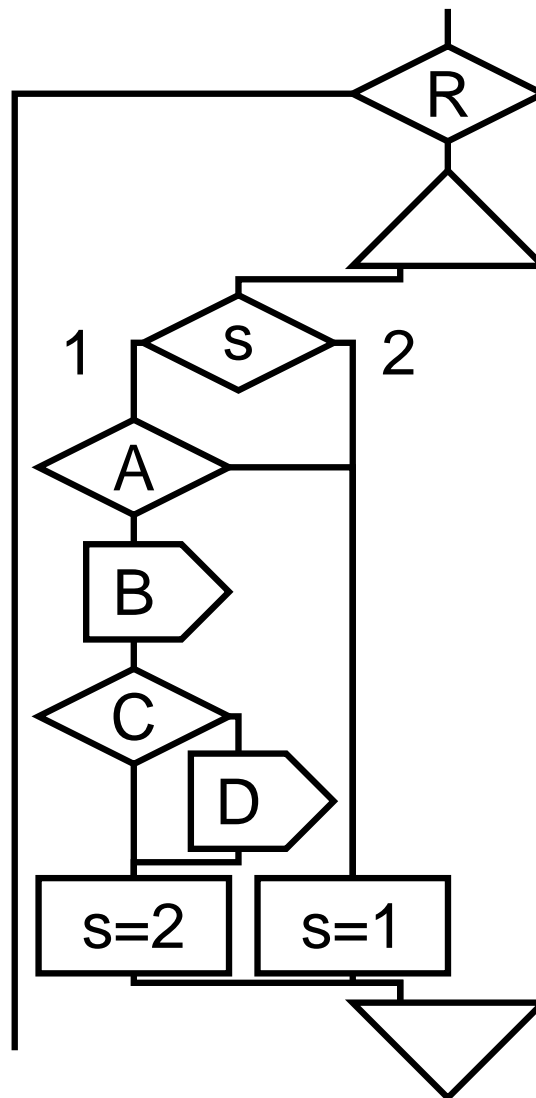
# Split at Pauses

```
every R do
  loop
    await A;
    emit B;
    present C then
      emit D end;
    pause
  end
||
  loop
    present B then
      emit C end;
    pause
  end
end
```



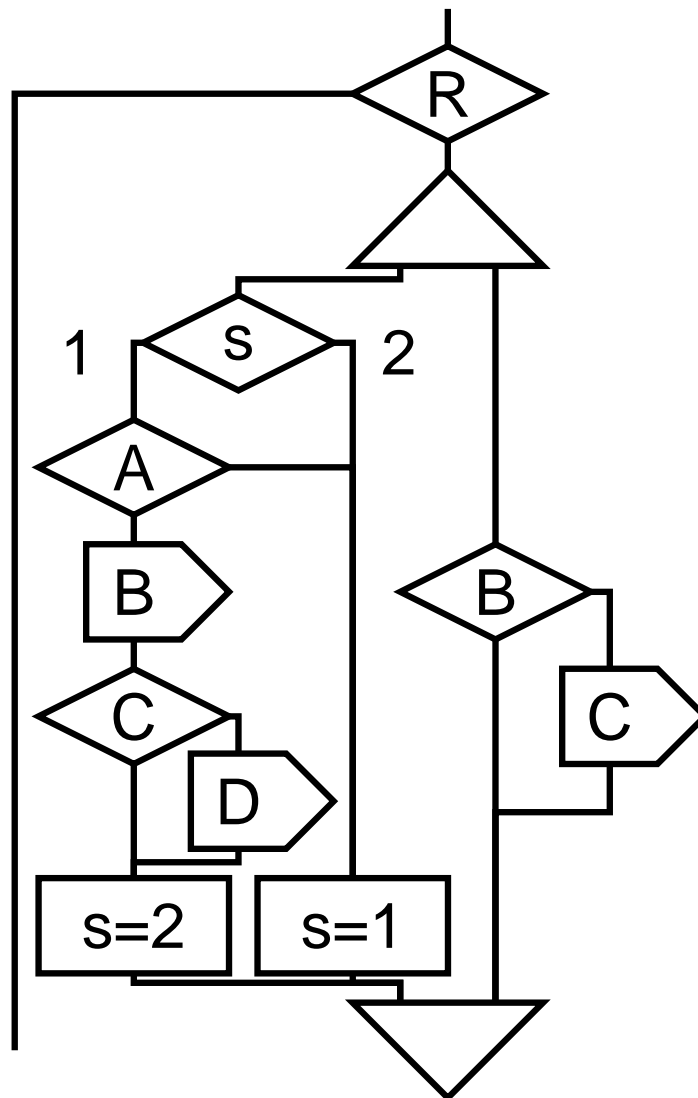
# Add Code Between Pauses

```
every R do
  loop
    await A;
    emit B;
    present C then
      emit D end;
    pause
  end
||
  loop
    present B then
      emit C end;
    pause
  end
end
```



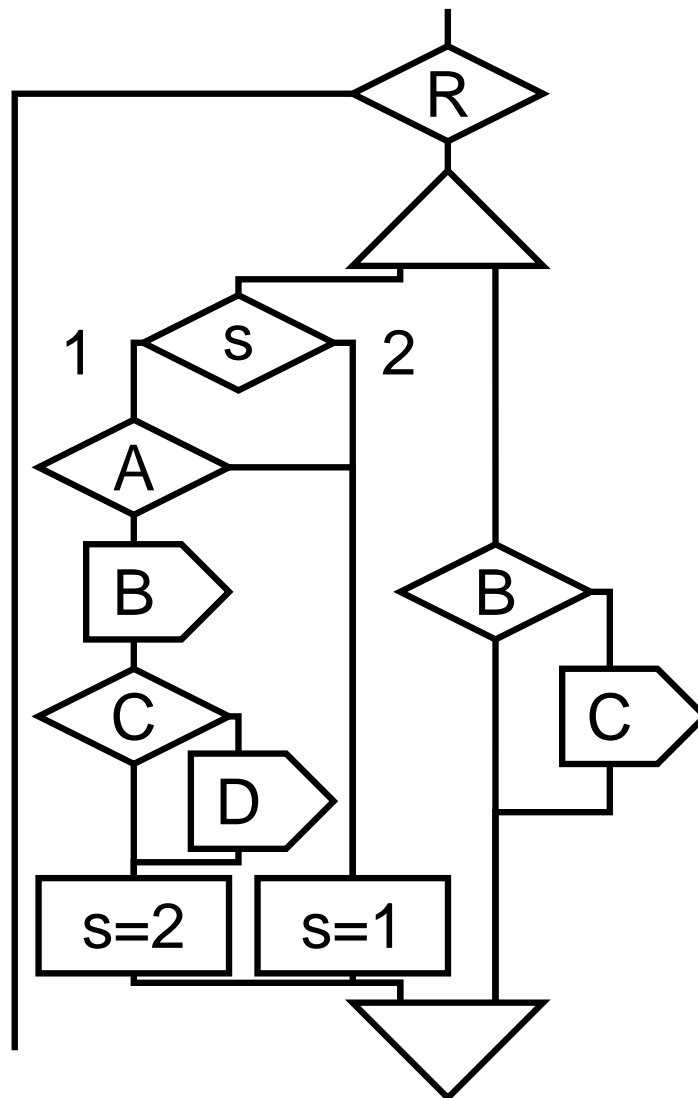
# Translate Second Thread

```
every R do
  loop
    await A;
    emit B;
    present C then
      emit D end;
    pause
  end
||
loop
  present B then
    emit C end;
  pause
end
end
```



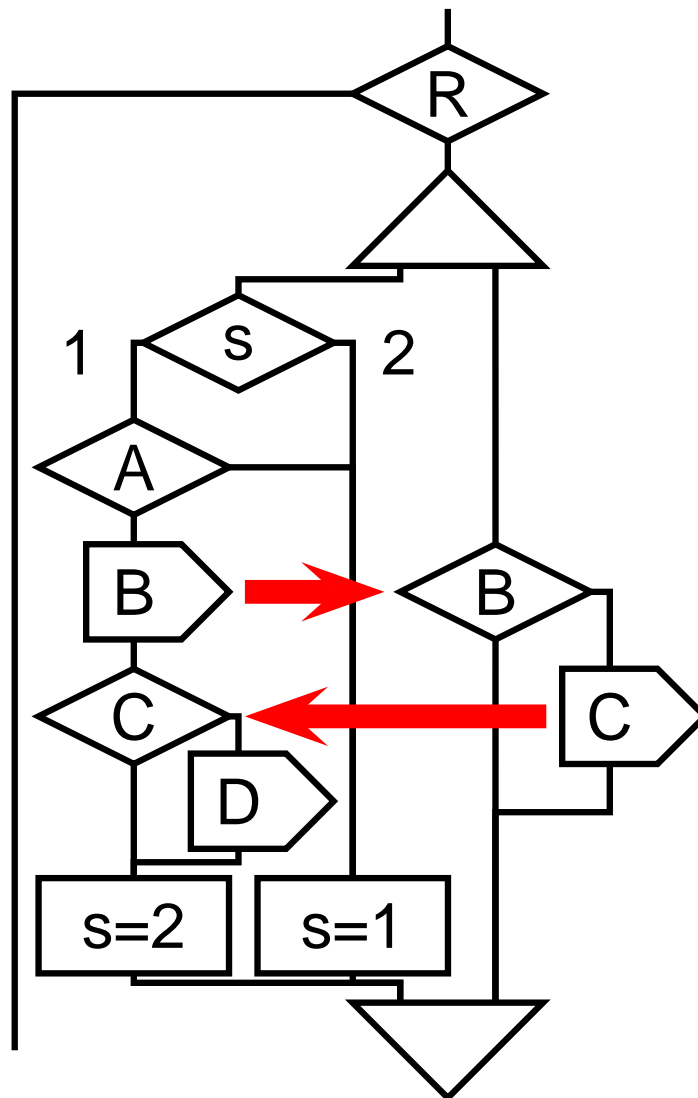
# Finished Translating

```
every R do
  loop
    await A;
    emit B;
    present C then
      emit D end;
    pause
  end
||
  loop
    present B then
      emit C end;
    pause
  end
end
```

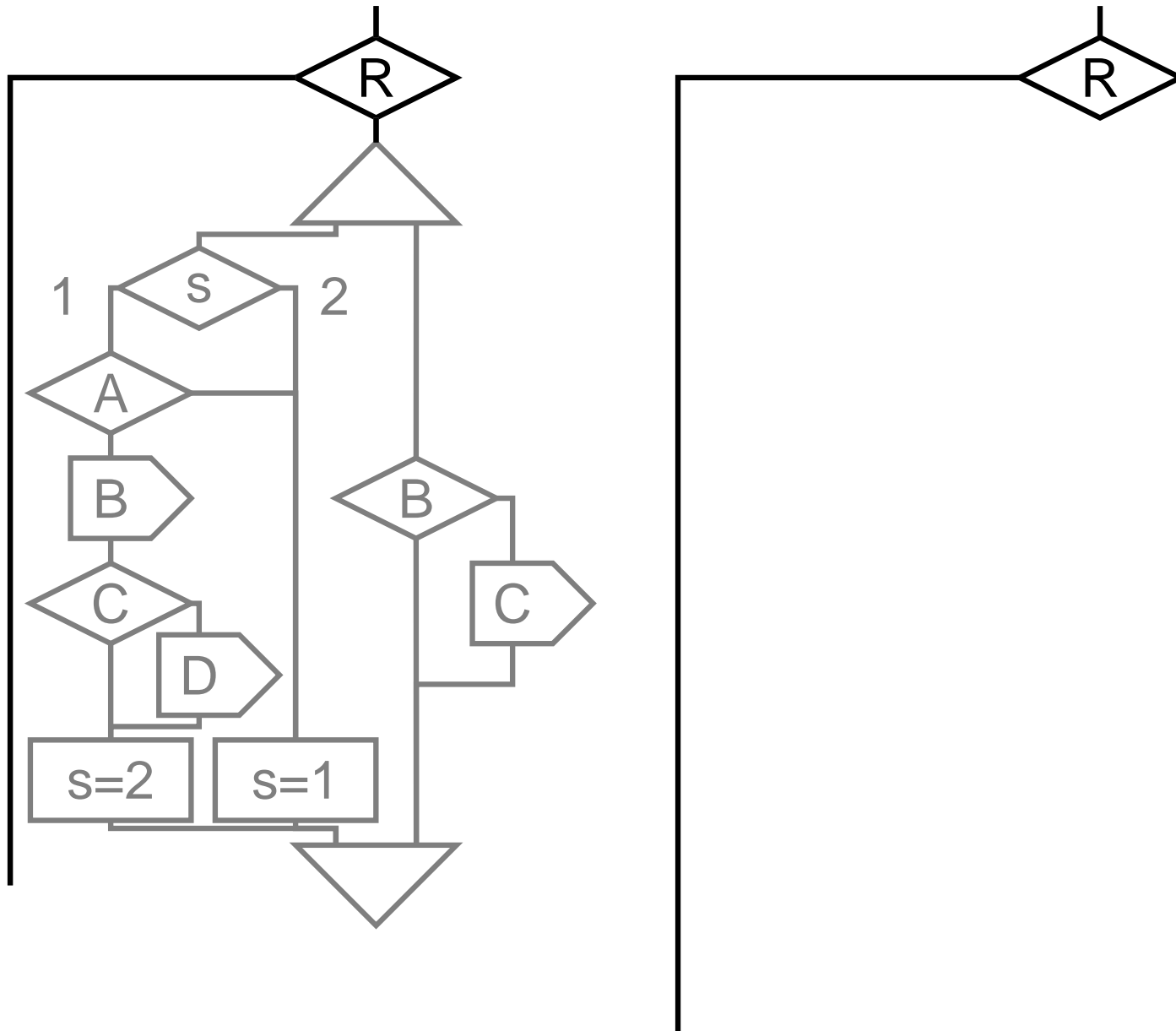


# Add Dependencies and Schedule

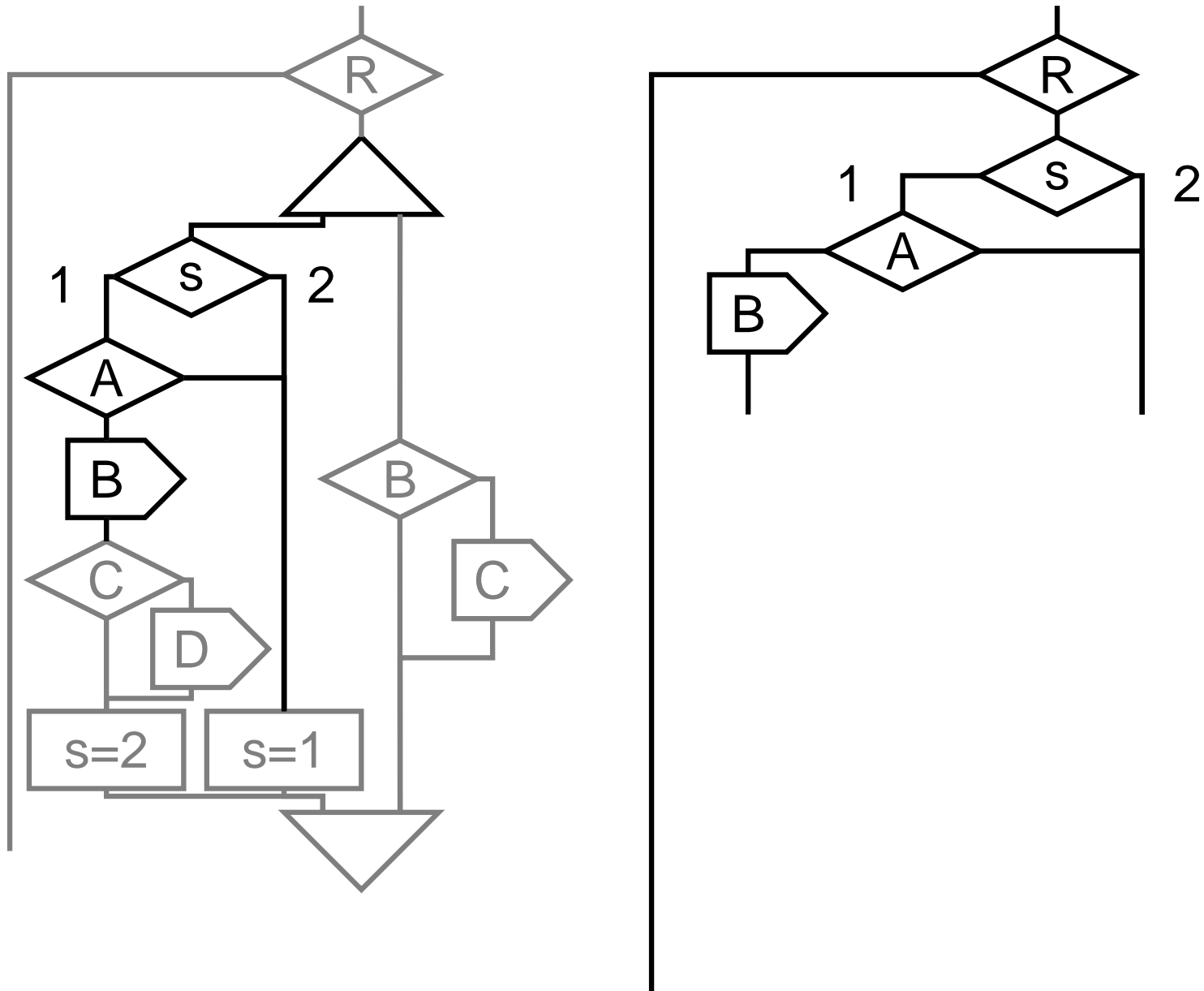
```
every R do
  loop
    await A;
    emit B;
    present C then
      emit D end;
    pause
  end
||
loop
  present B then
    emit C end;
  pause
end
end
```



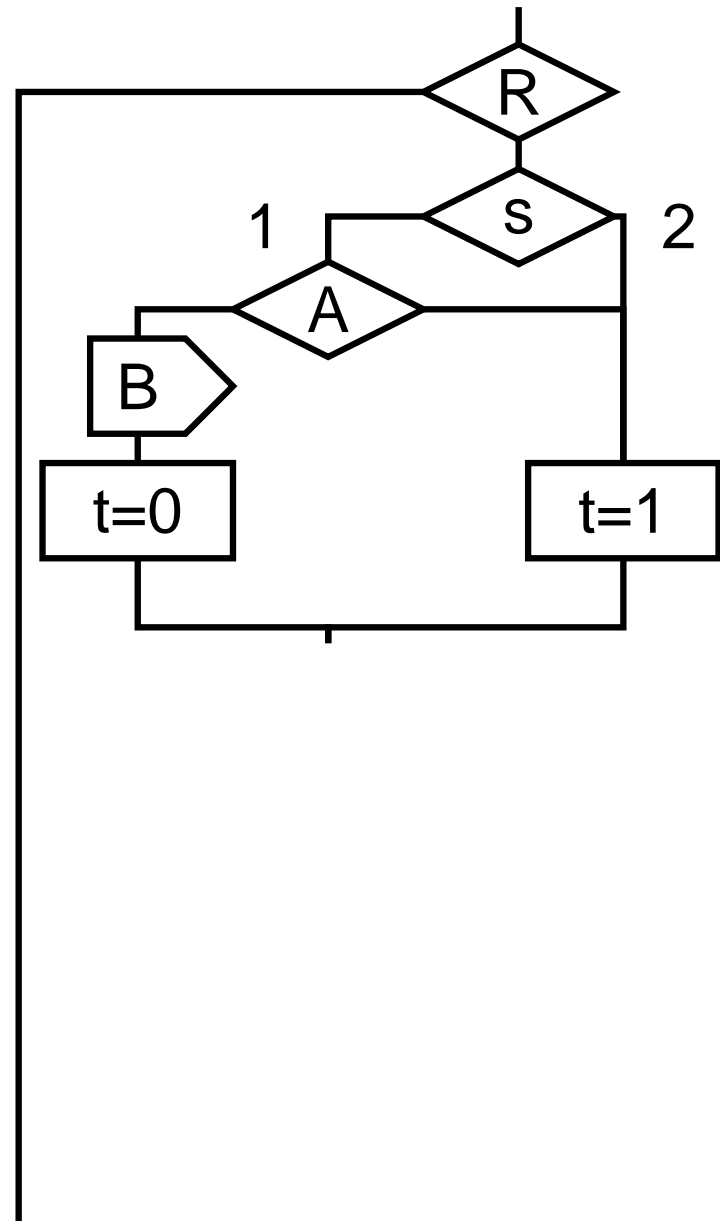
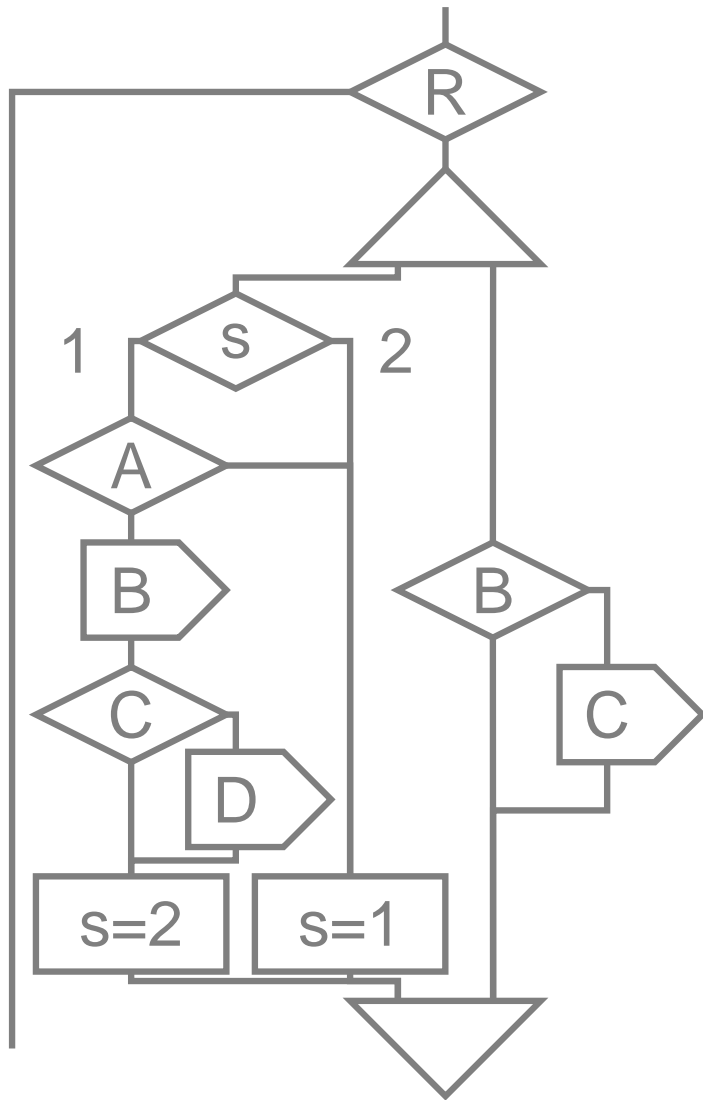
# Run First Node



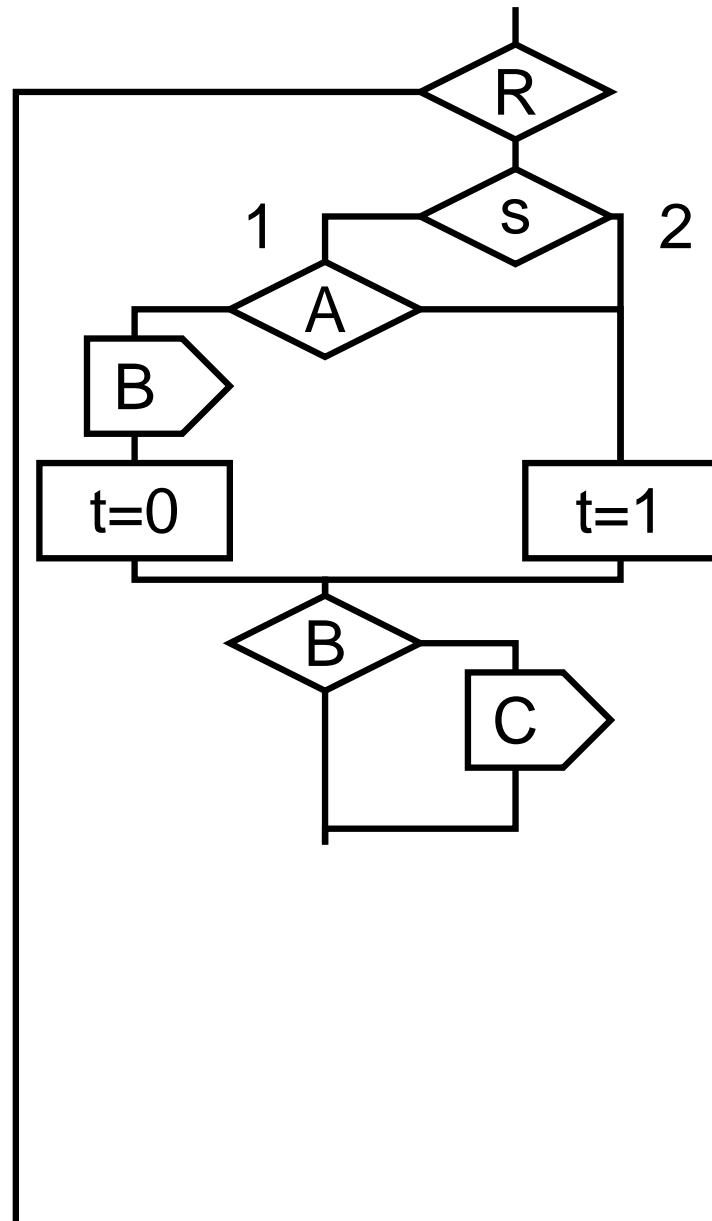
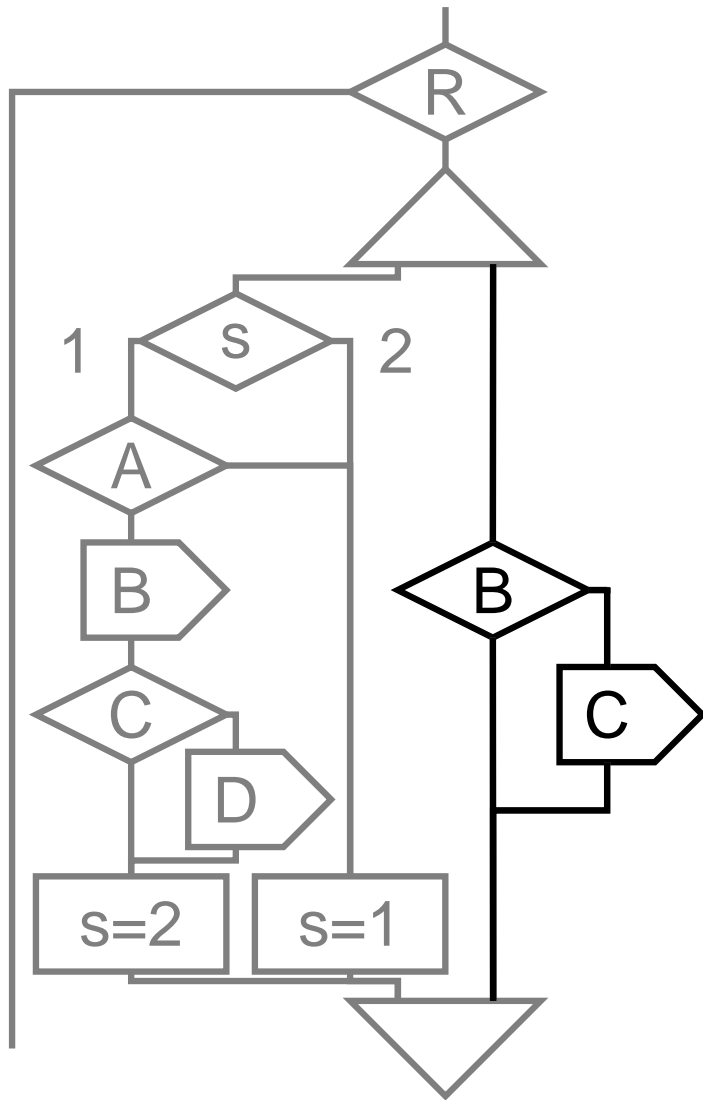
# Run First Part of Left Thread



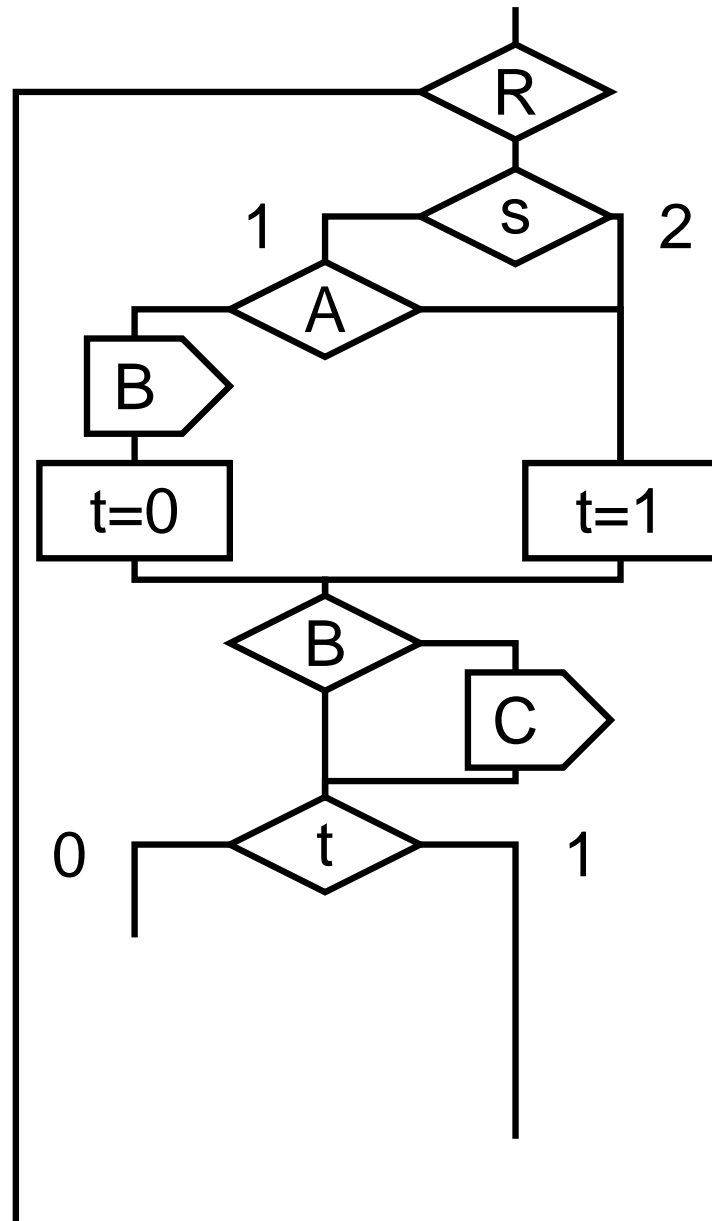
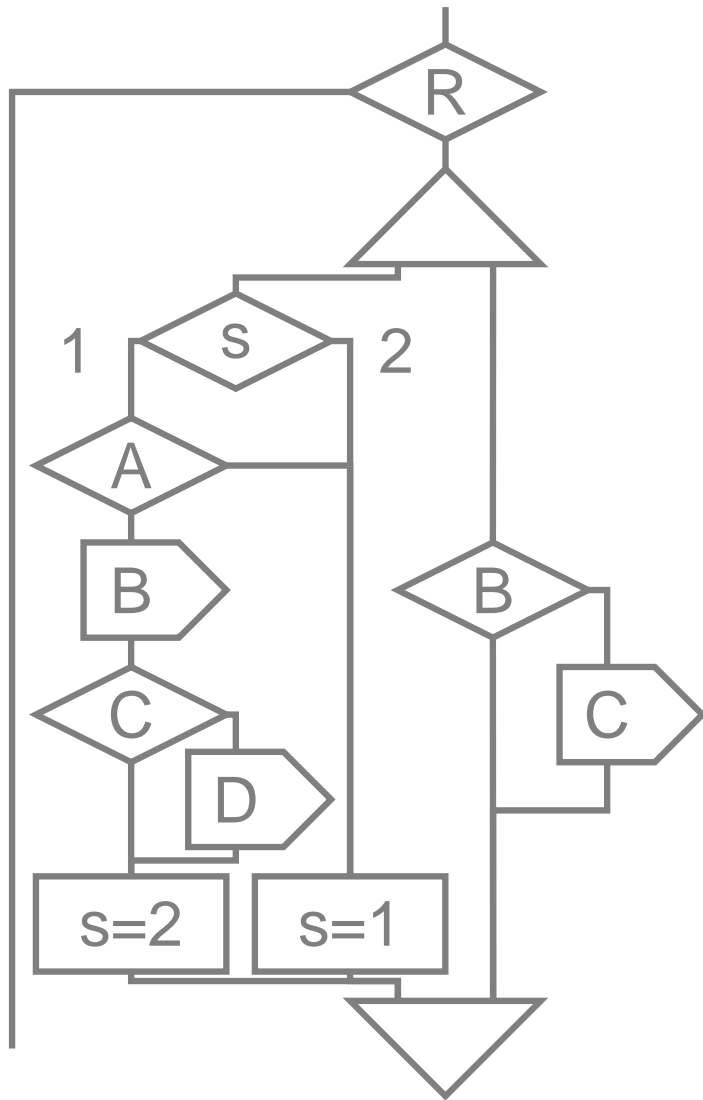
# Context Switch



# Run Right Thread

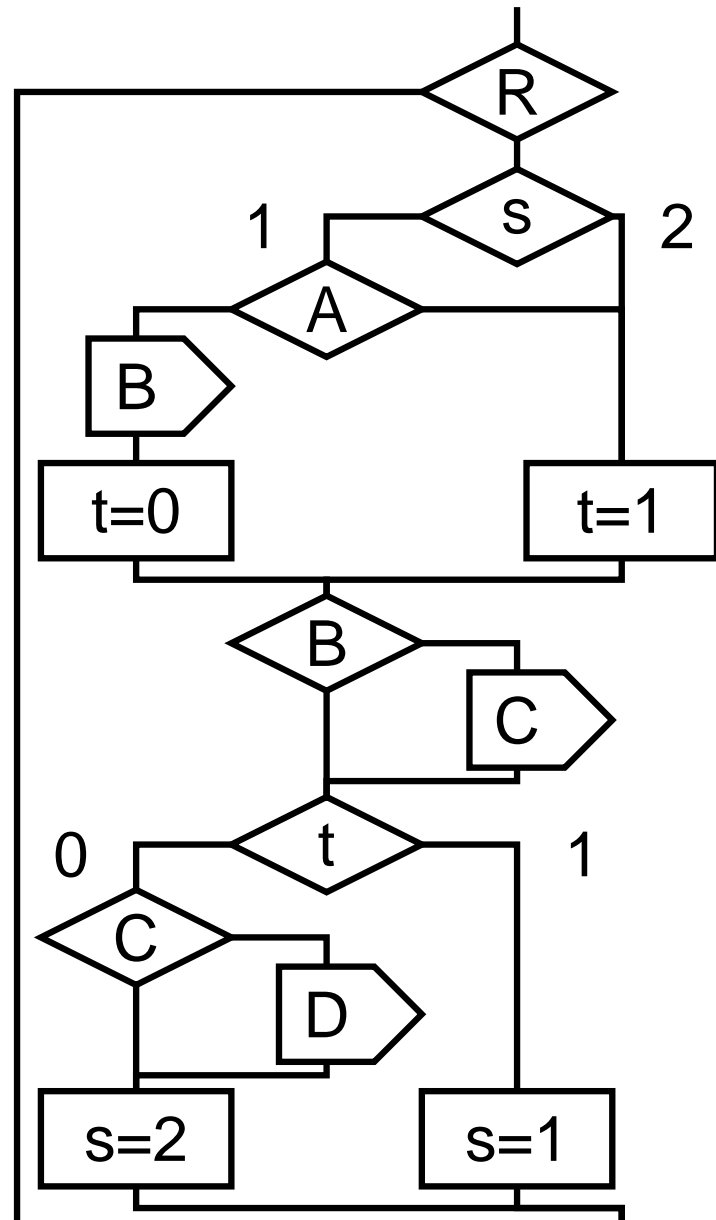
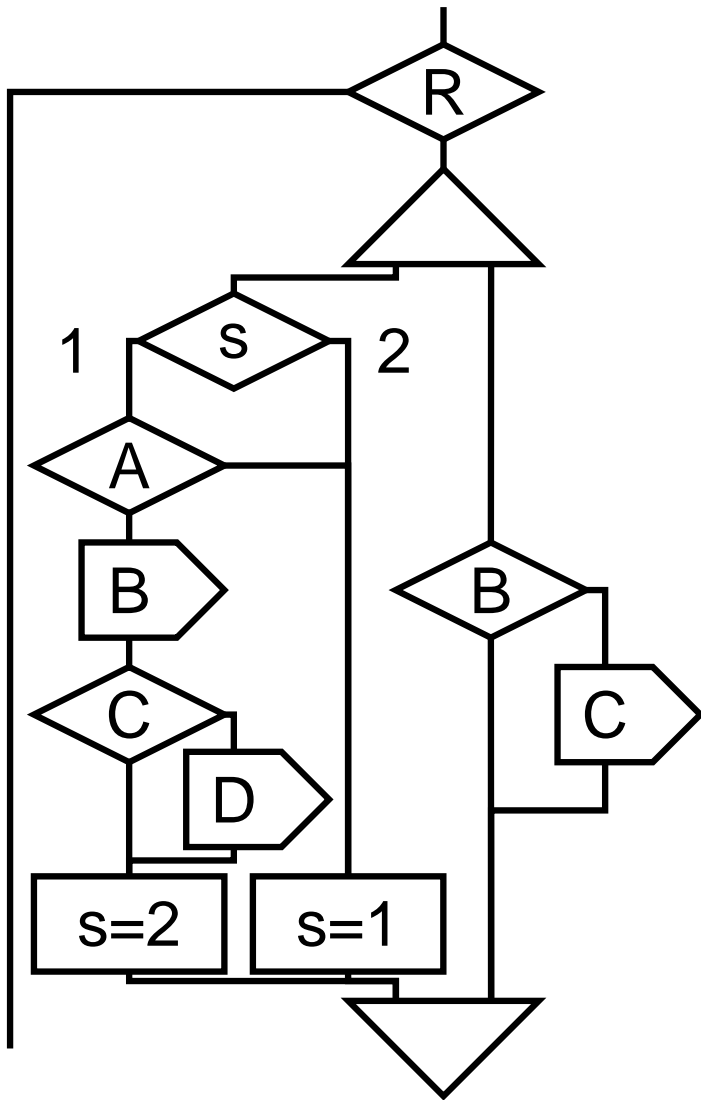


# Context Switch

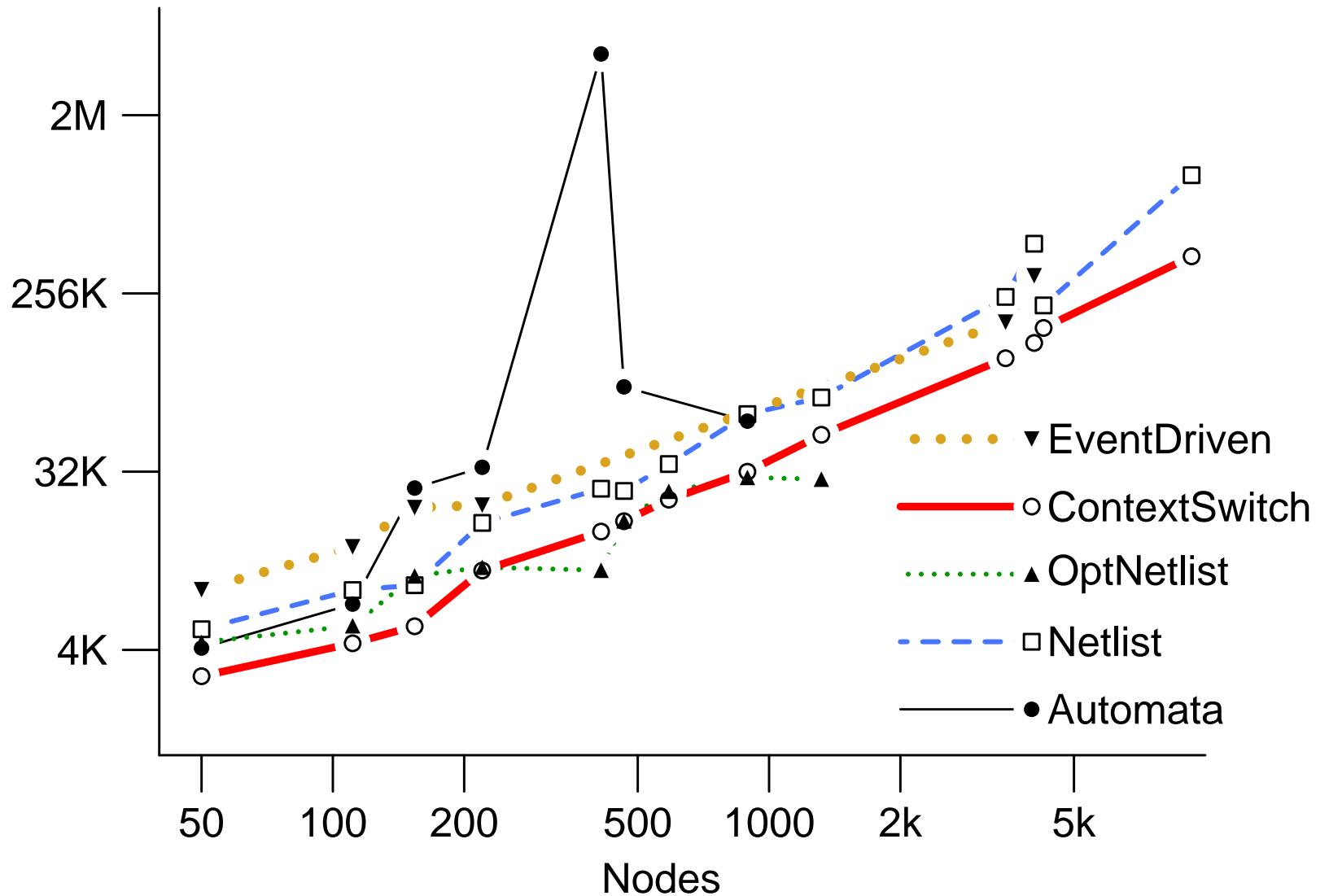




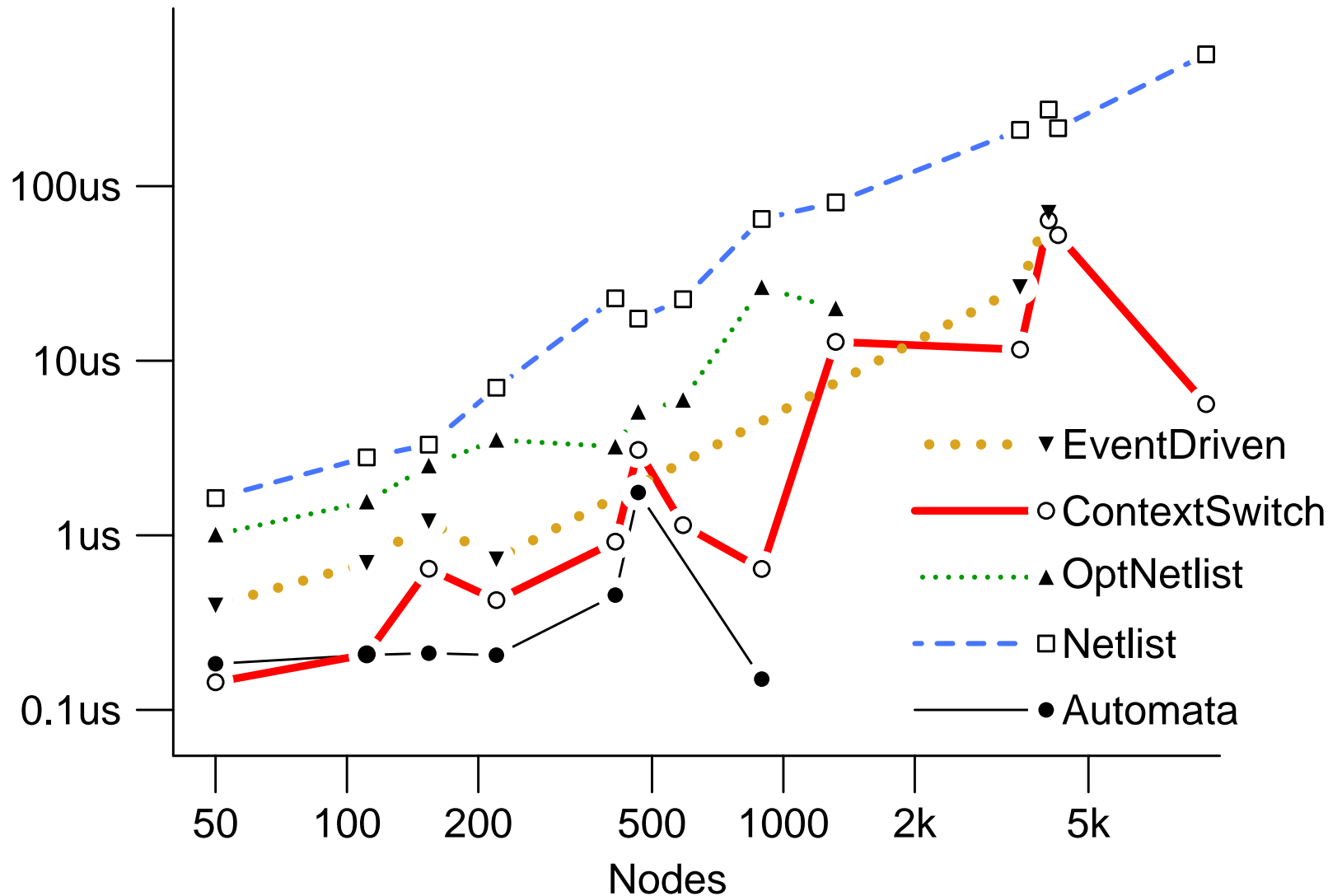
# Completed Example



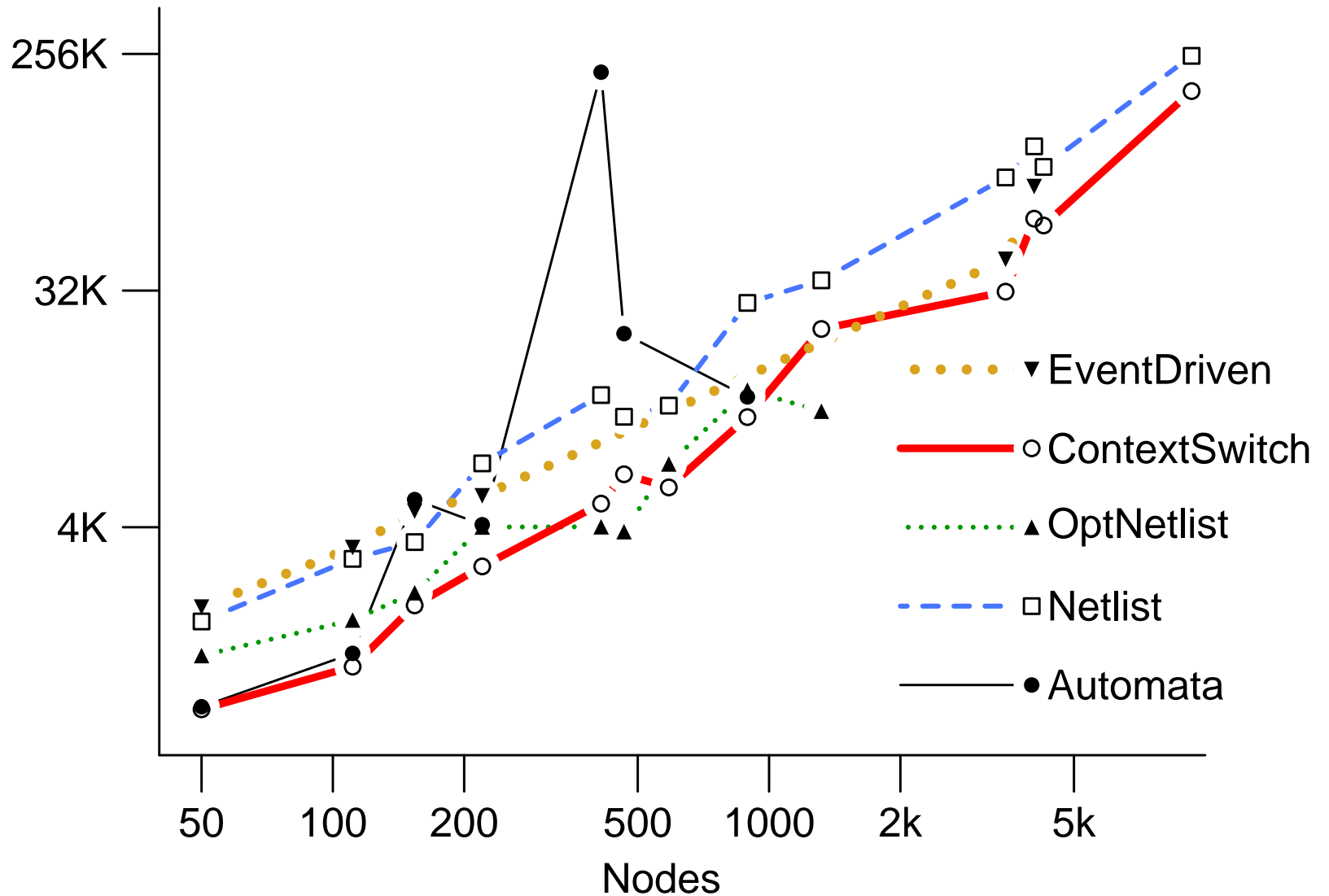
# Size of Generated Code on an UltraSparc-II



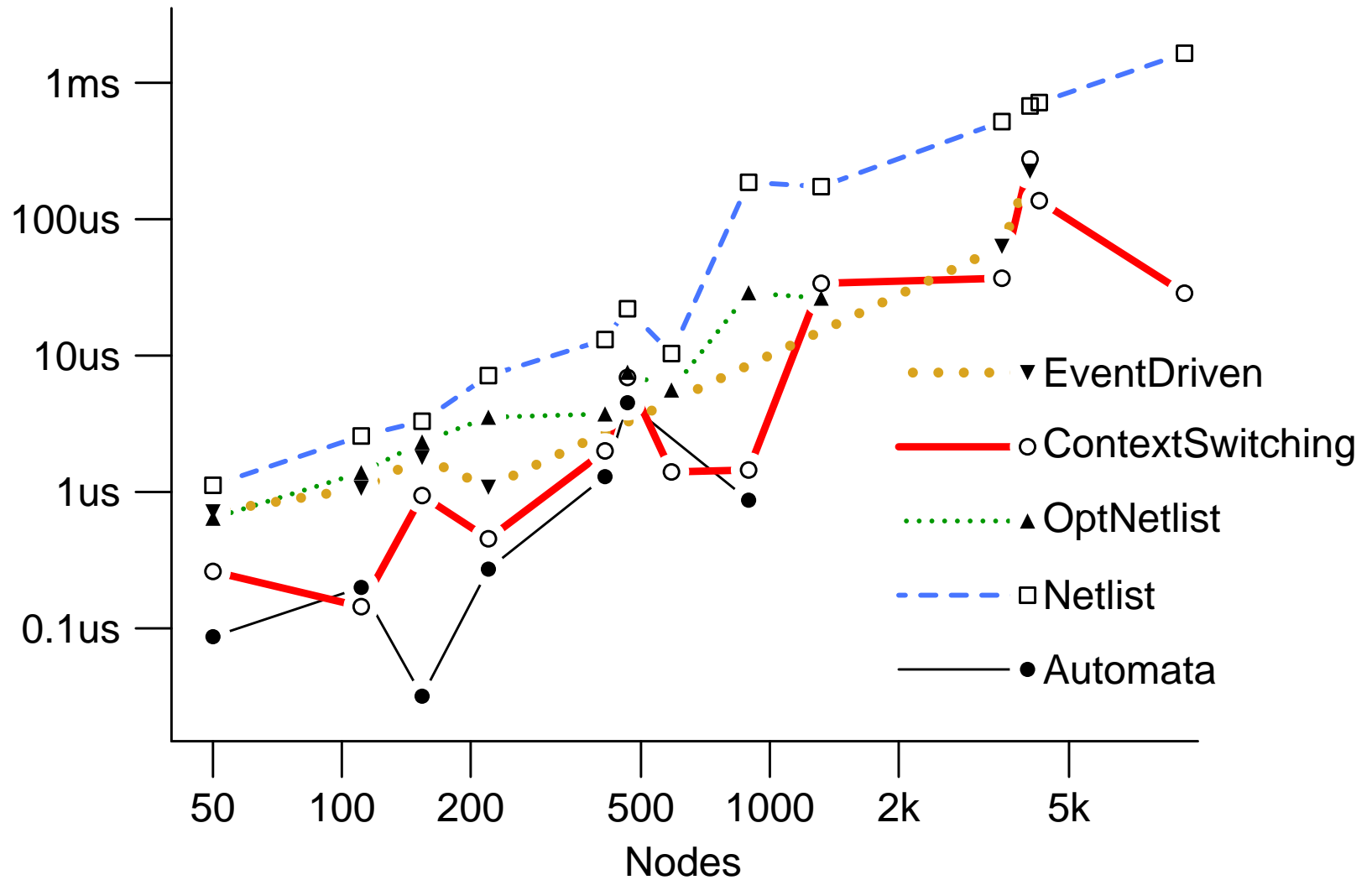
# Average Cycle Times on an UltraSparc-II



# Size of Generated Code on a Pentium



# Average Cycle Times on a Pentium



# **My Esterel Compiler for Hardware**

The ESUIF Open Source Esterel Compiler

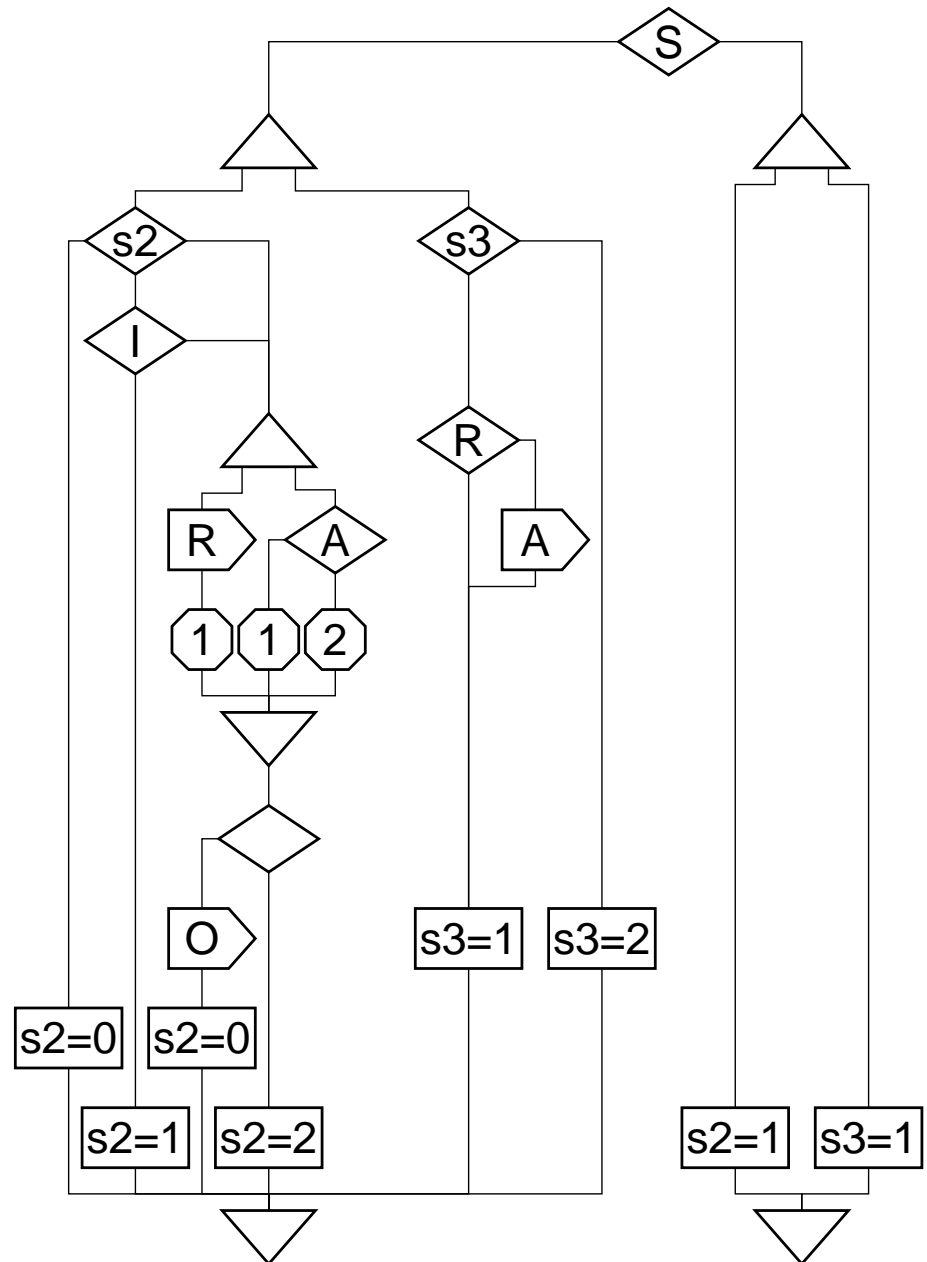
(Work in Progress)

Presented at SLAP 2002, IWLS 2002

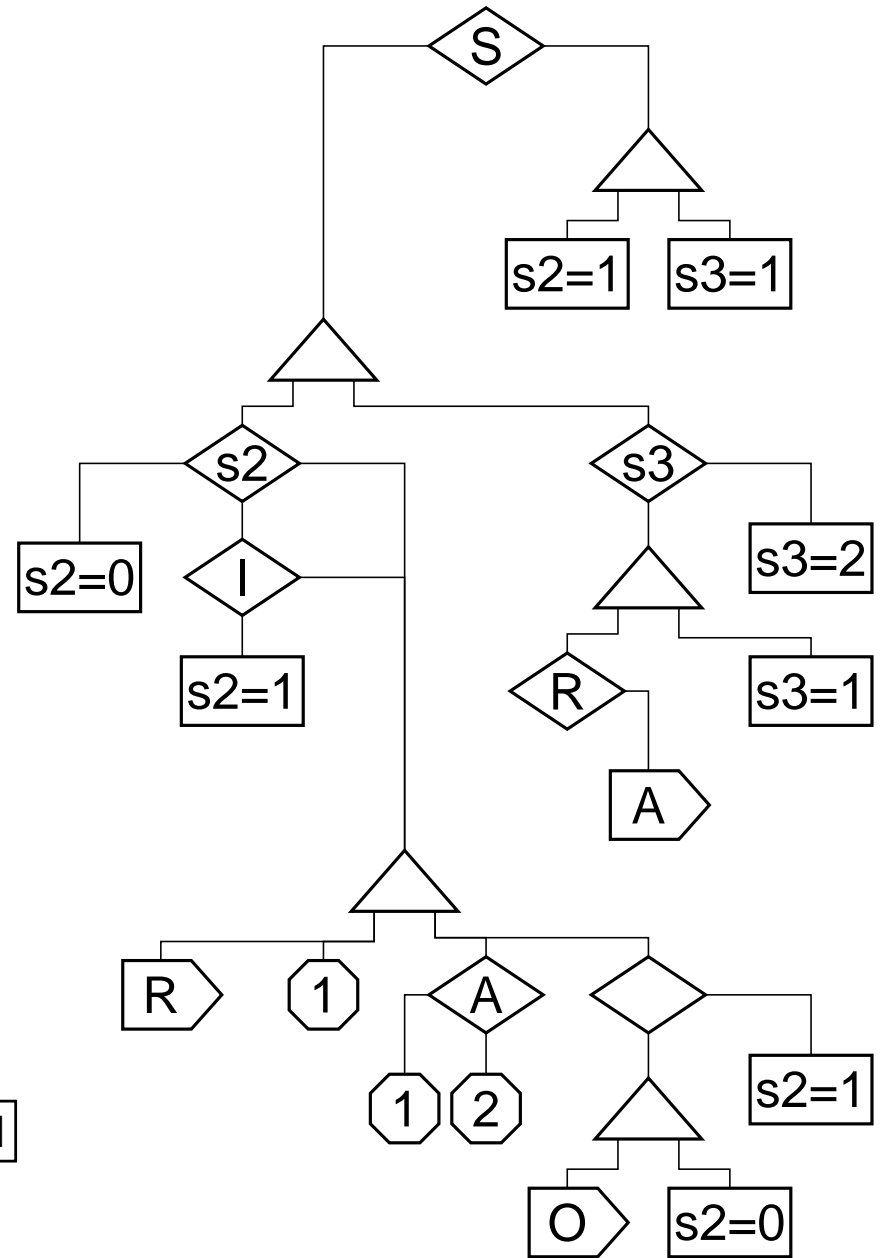
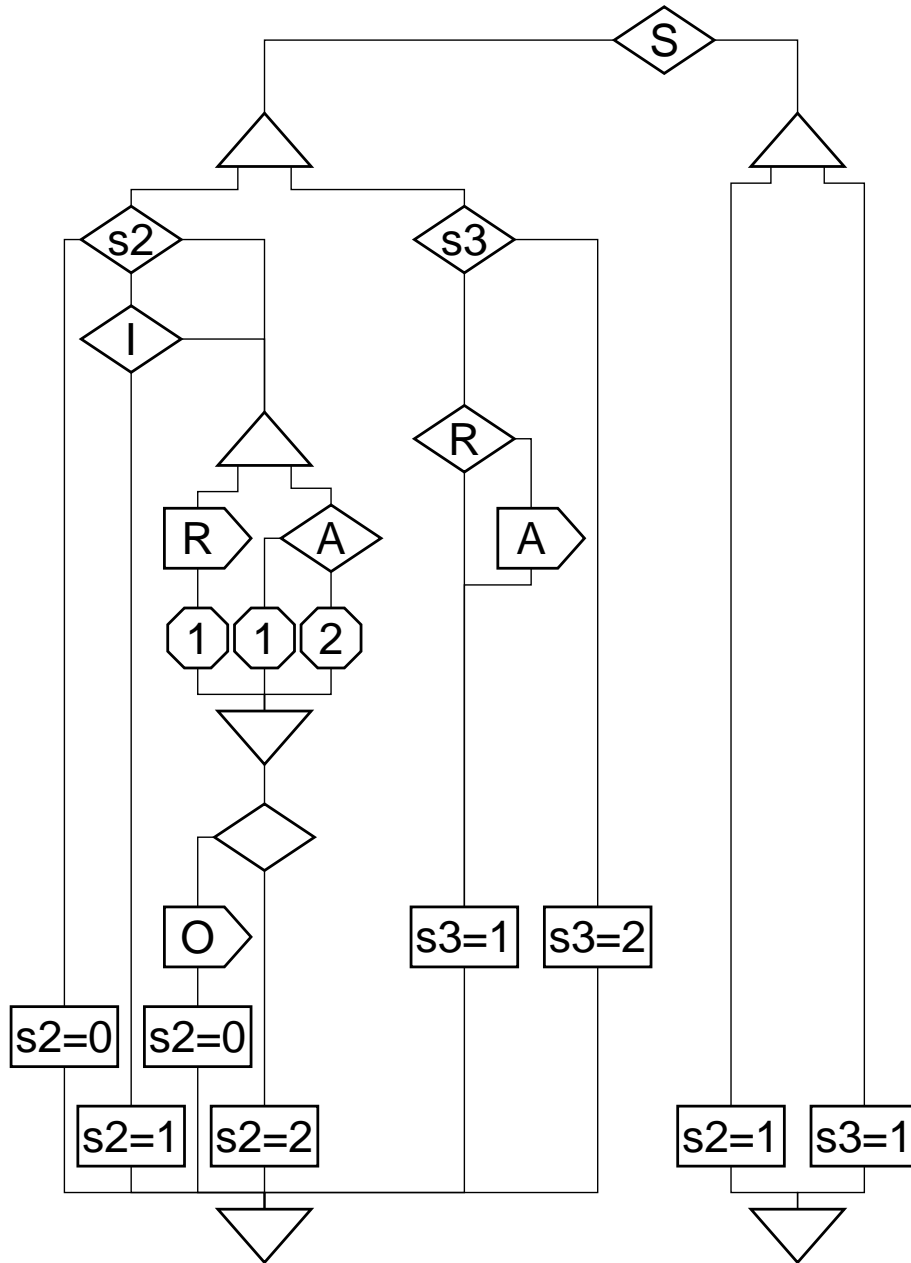
# Translation to CCFG

```

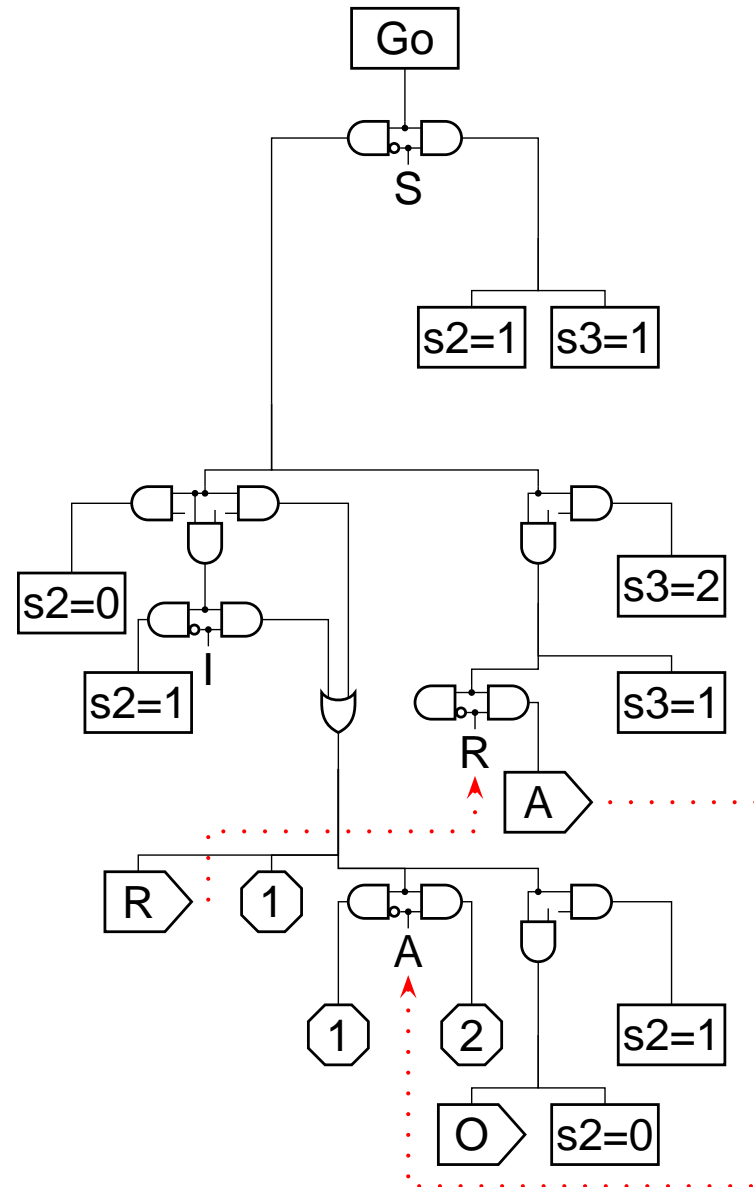
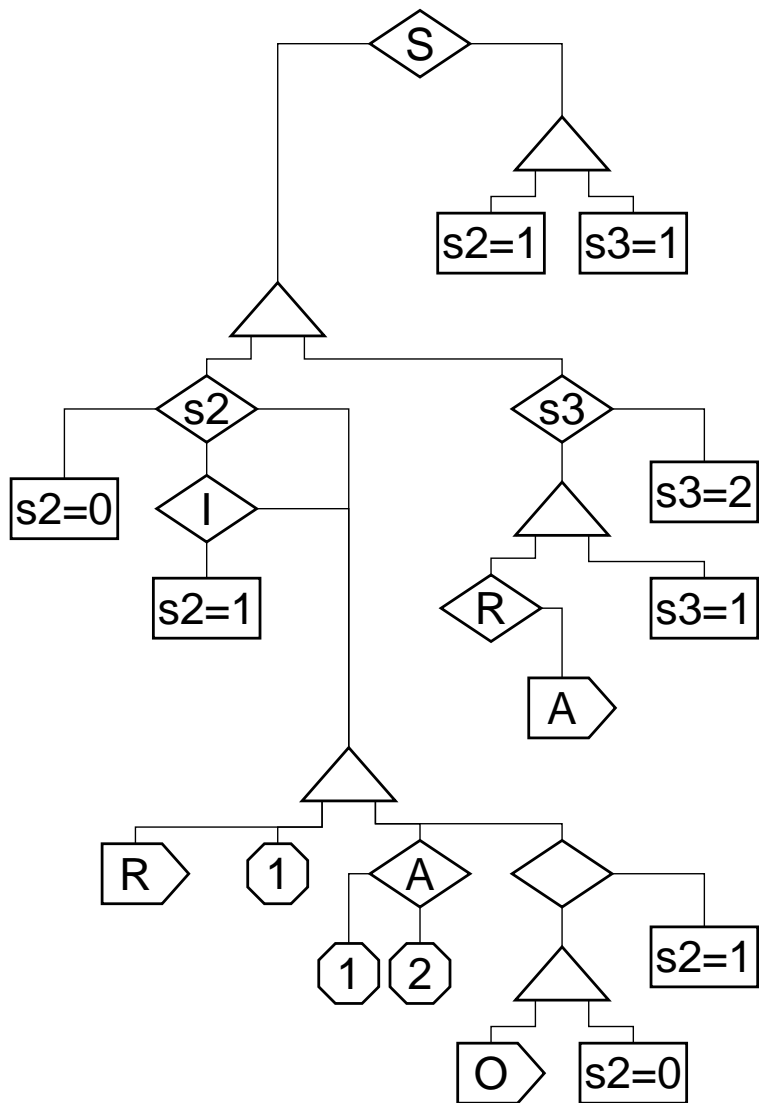
every S do
  await I;
  weak abort
    sustain R
  when immediate A;
  emit O
||
loop
  pause; pause;
  present R then
    emit A
  end
end
end
end
  
```



# Translation to PDG



# Translation to Circuitry



# Summary

Introduction to Esterel and Existing Compilers

Synchronous, Concurrent, Textual Language

Automata, Netlist, and Control-based compilers

My Software Compiler [DAC 2000, TransCAD 2002]

Translate to Concurrent CFG, schedule, then  
synthesize Sequential CFG

My Hardware Compiler: ESUIF [SLAP 2002, IWLS 2002]

Translate CCFG to Program Dependence Graph

Trivially translate PDG to circuitry

Open-source, under development

# Thanks For Your Attention

**Stephen A. Edwards**

Department of Computer Science

Columbia University

[www.cs.columbia.edu/~sedwards](http://www.cs.columbia.edu/~sedwards)

[sedwards@cs.columbia.edu](mailto:sedwards@cs.columbia.edu)