# Formal models in industry standard tools: An Argos block within Simulink

IEHSC, May 2005, Singapore.

Timothy Bourke Arcot Sowmya

School of Computer Science and Engineering

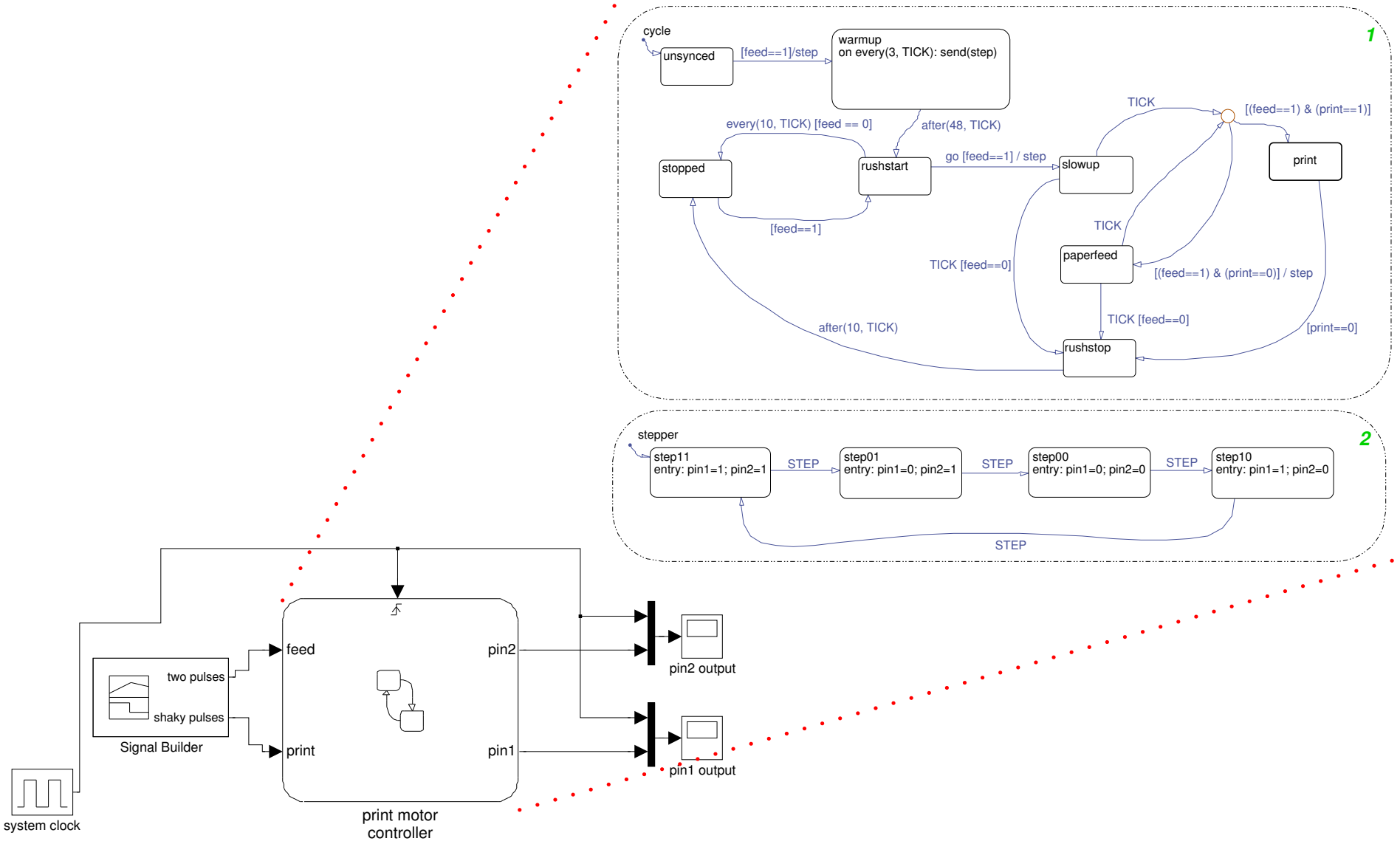
University of New South Wales, Sydney

and National ICT Australia

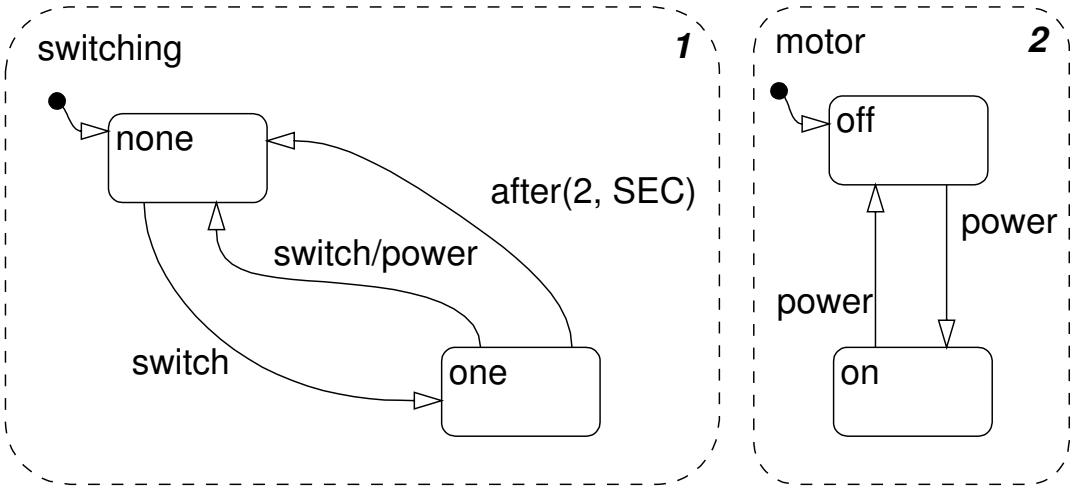

THE UNIVERSITY OF NEW SOUTH WALES
SYDNEY · AUSTRALIA


Date: 2005/05/22 07:16:17 Revision: 1.5

# HYBRID = CONTINUOUS + DISCRETE



# Stateflow




Many Statecharts features:

- hierarchy
- parallelism
- history junctions

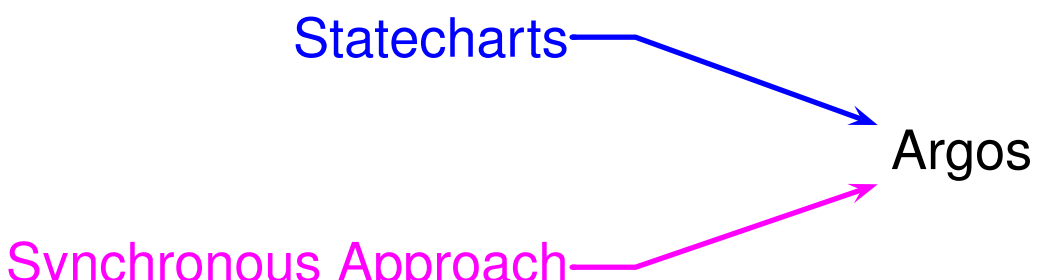
Flowchart-like transitions:

- sequencing
- branching
- loops

**Thinking/communicating about designs is involved:**

1. intricate ordering rules
2. queued event processing
3. stacking of communications
4. implicit assumption of synchrony

# Argos: a synchronous language



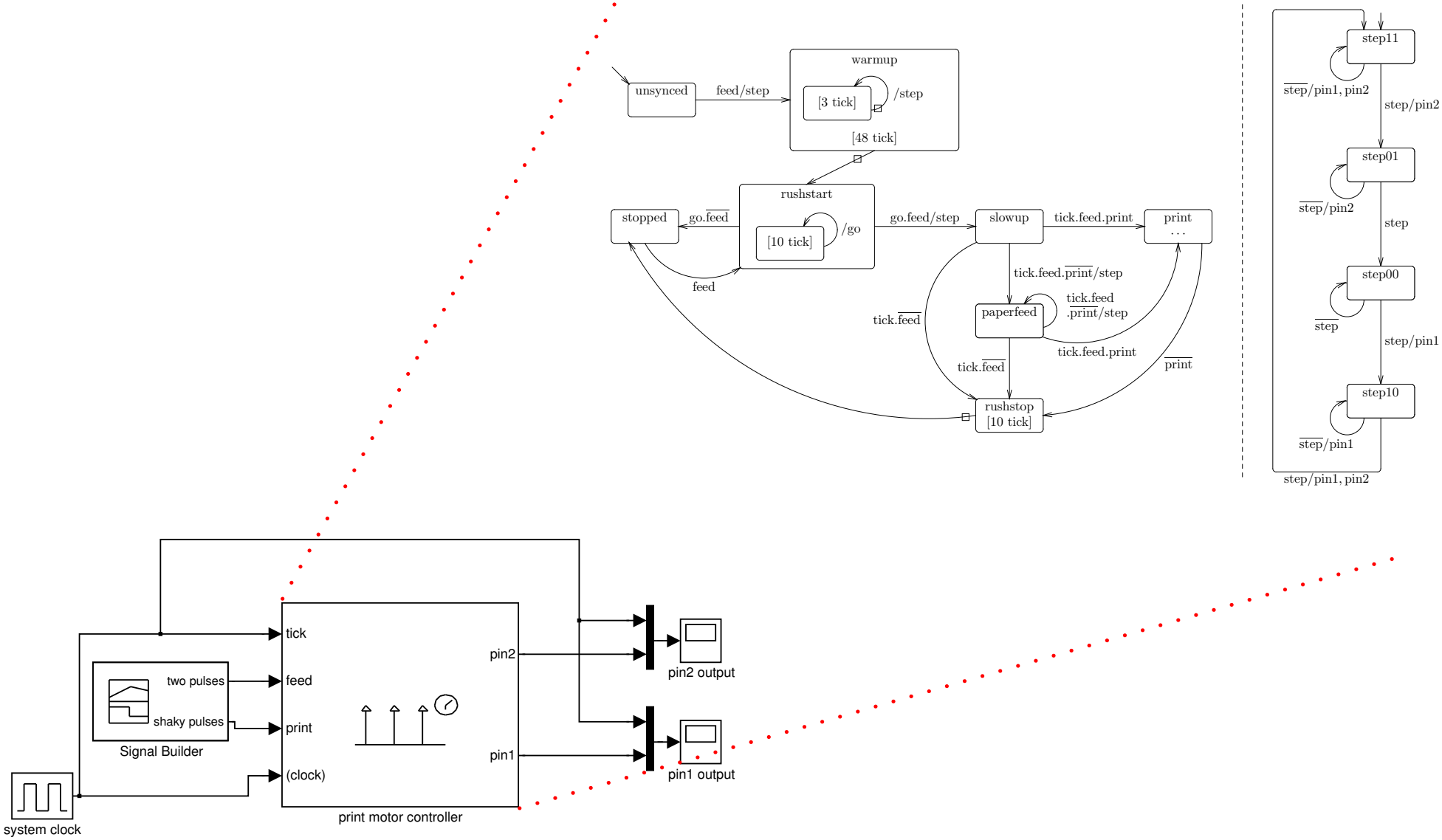

- Mealy machines
- Hierarchy
- Parallelism
- Discrete reactions
- Well-defined internal behaviour

- Esterel, Lustre, Signal
- CMA, INRIA, Verimag, IRISA

**Argos is a synchronous version of Statecharts.**

- Developed by Maraninchi and Rémond [Mar91, MR01].
- Well suited to some reactive programming tasks.

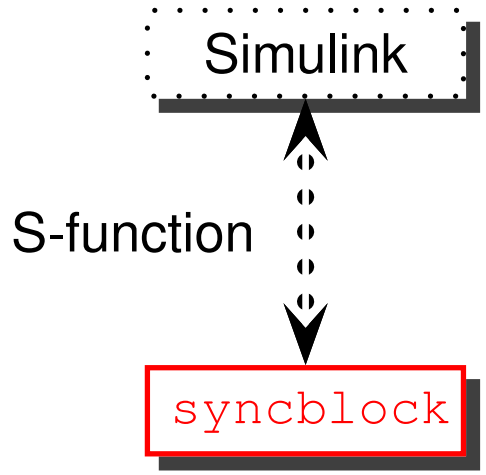
# An Argos block: Syncblock




[CCM+03, SSC+04]

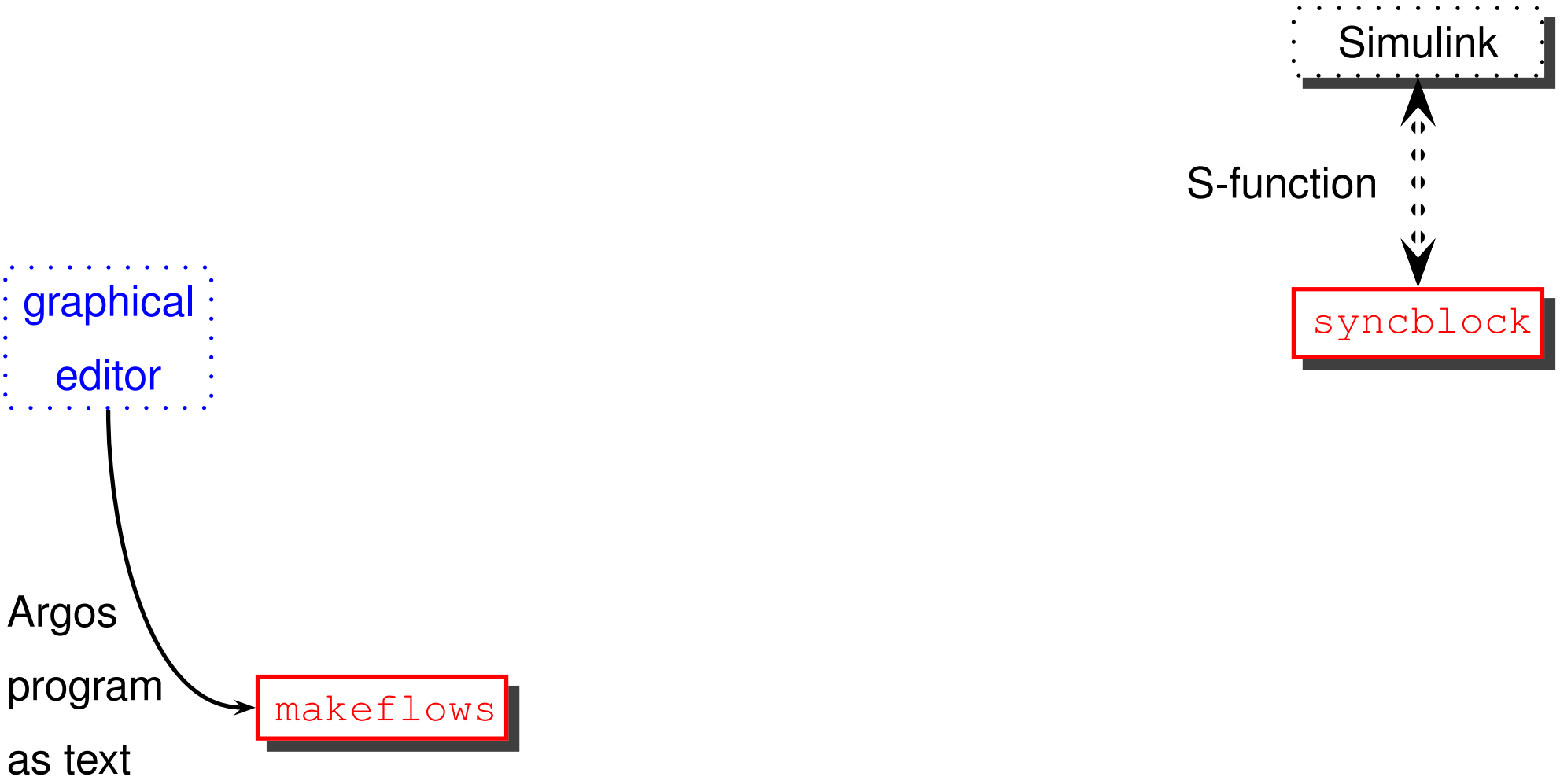
# Syncblock Implementation

# Syncblock Implementation

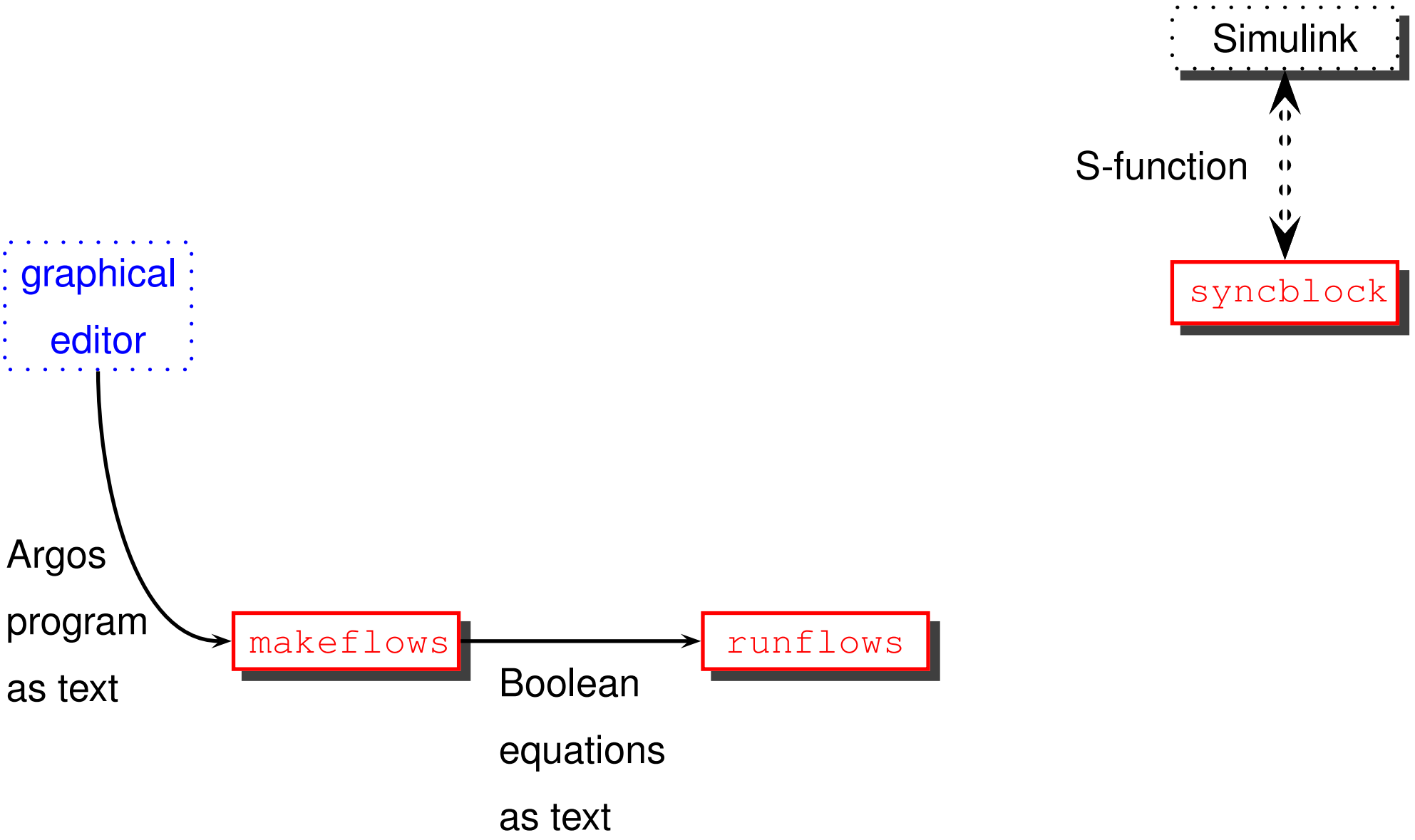


# Syncblock Implementation

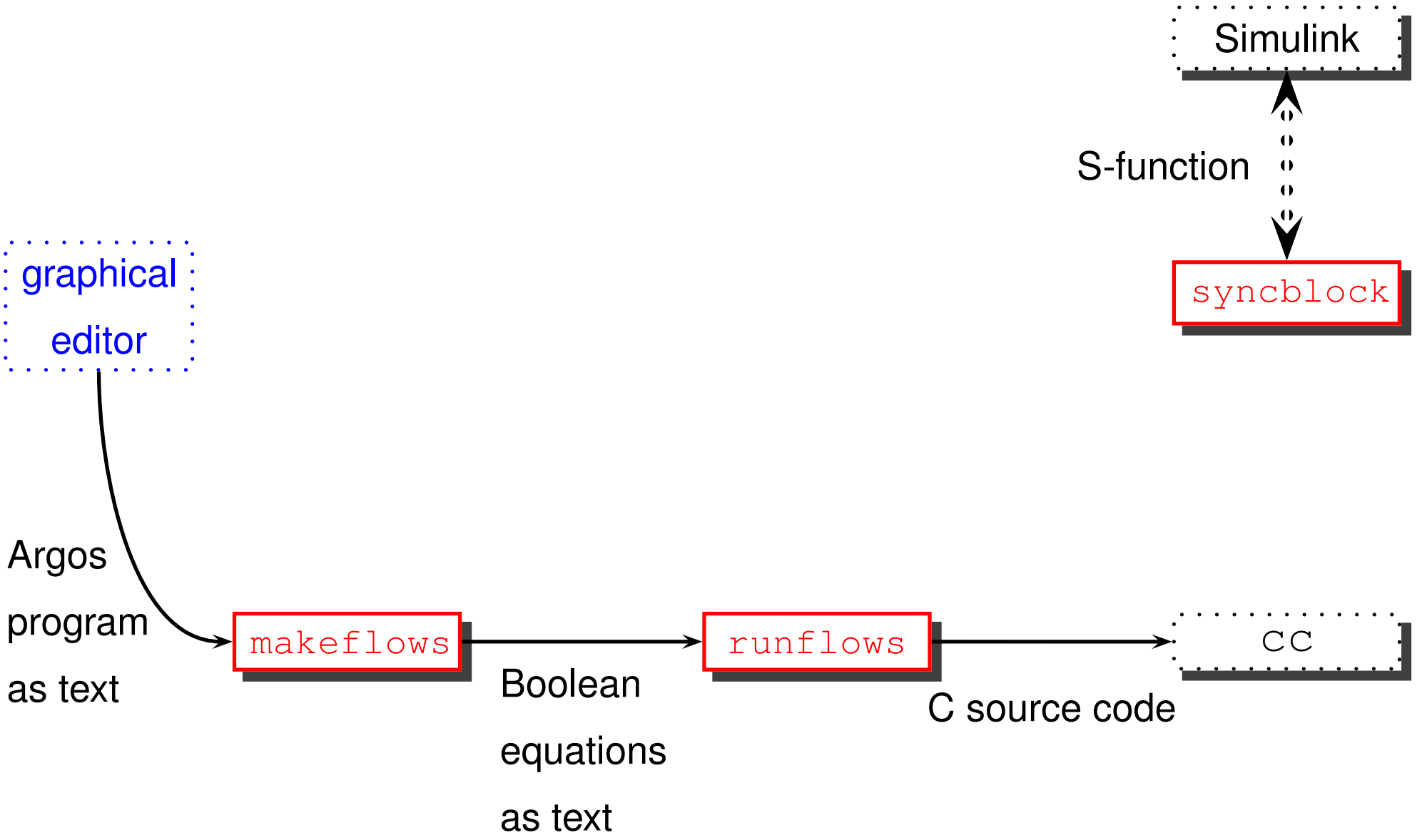
# Syncblock Implementation



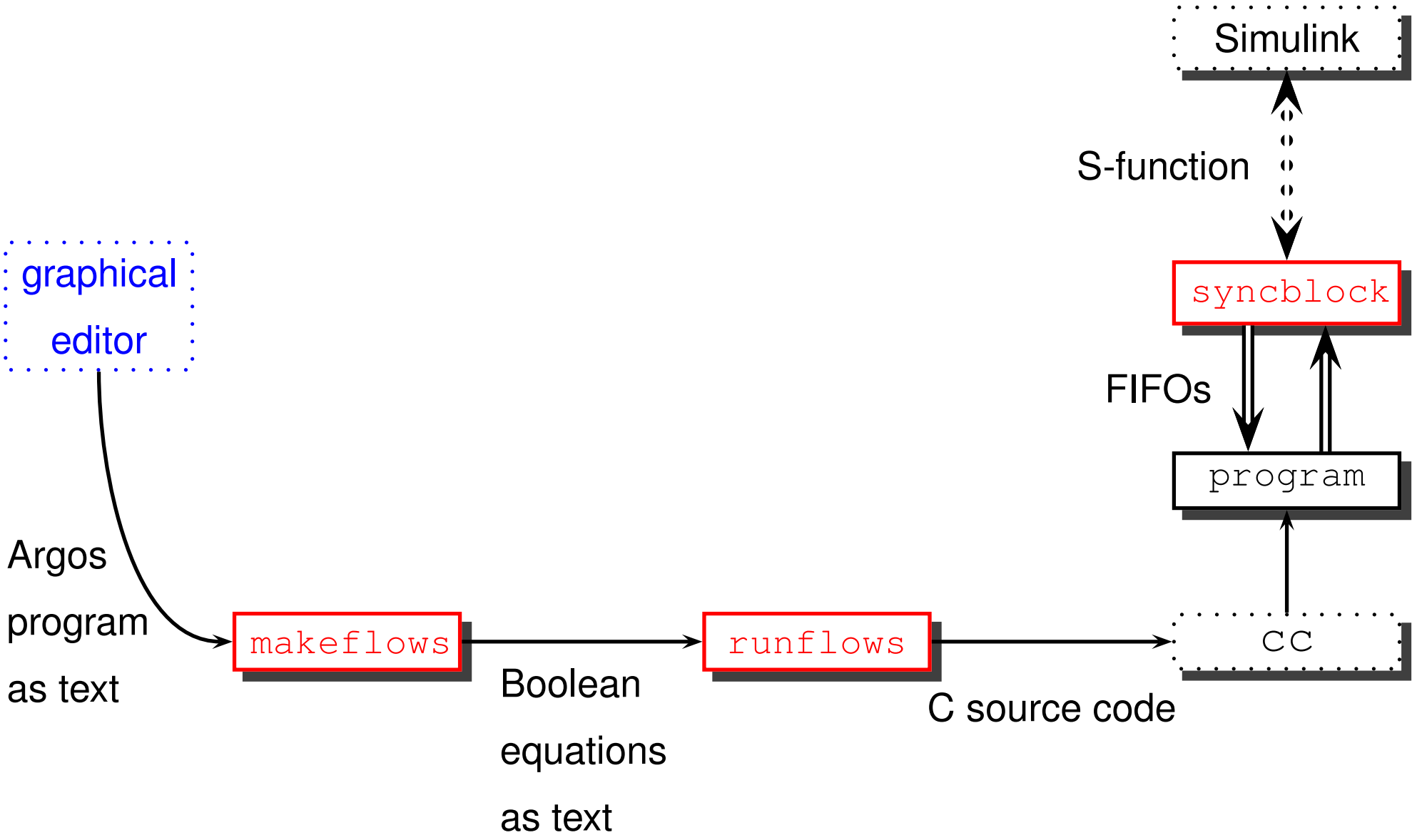
# Syncblock Implementation



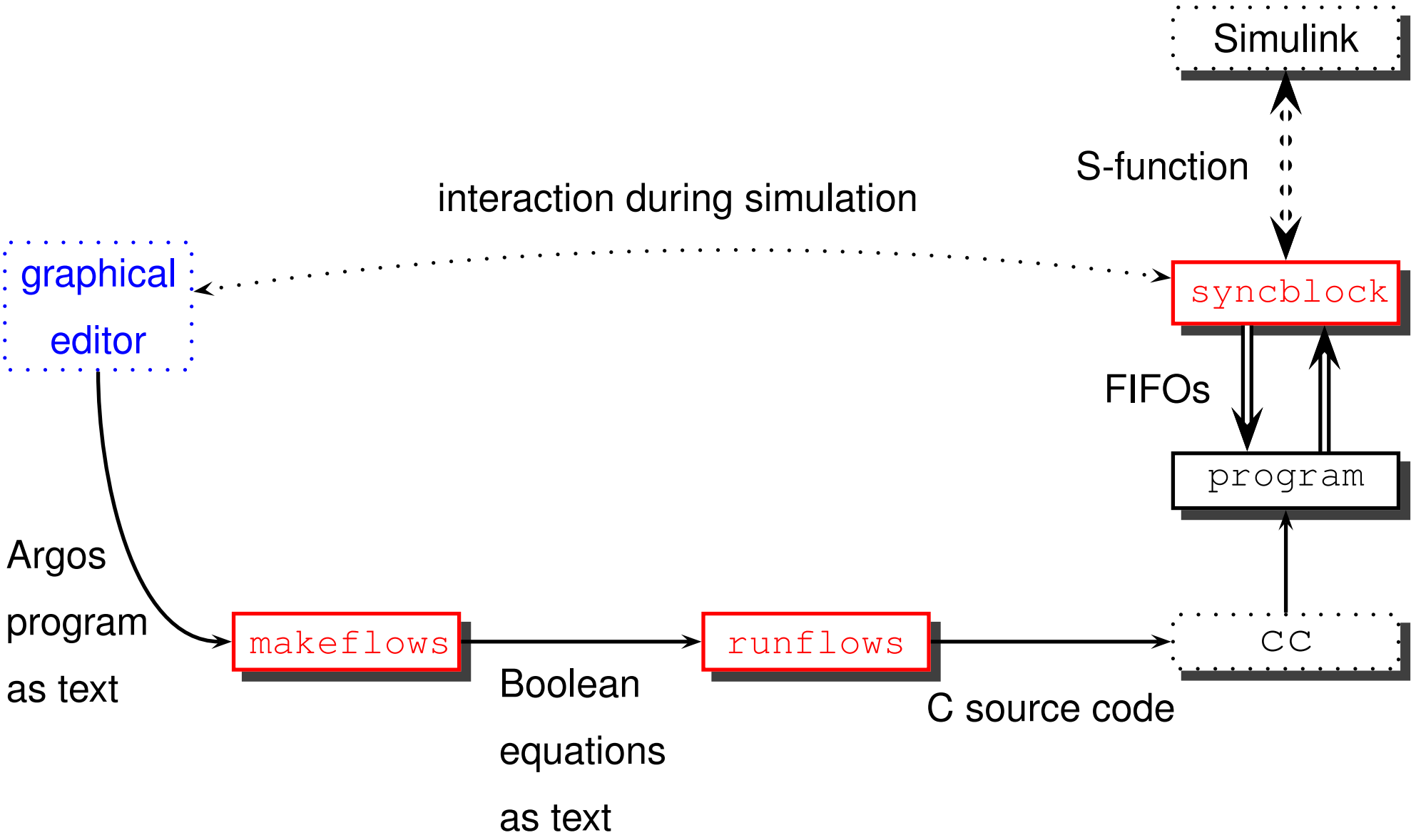
# Syncblock Implementation



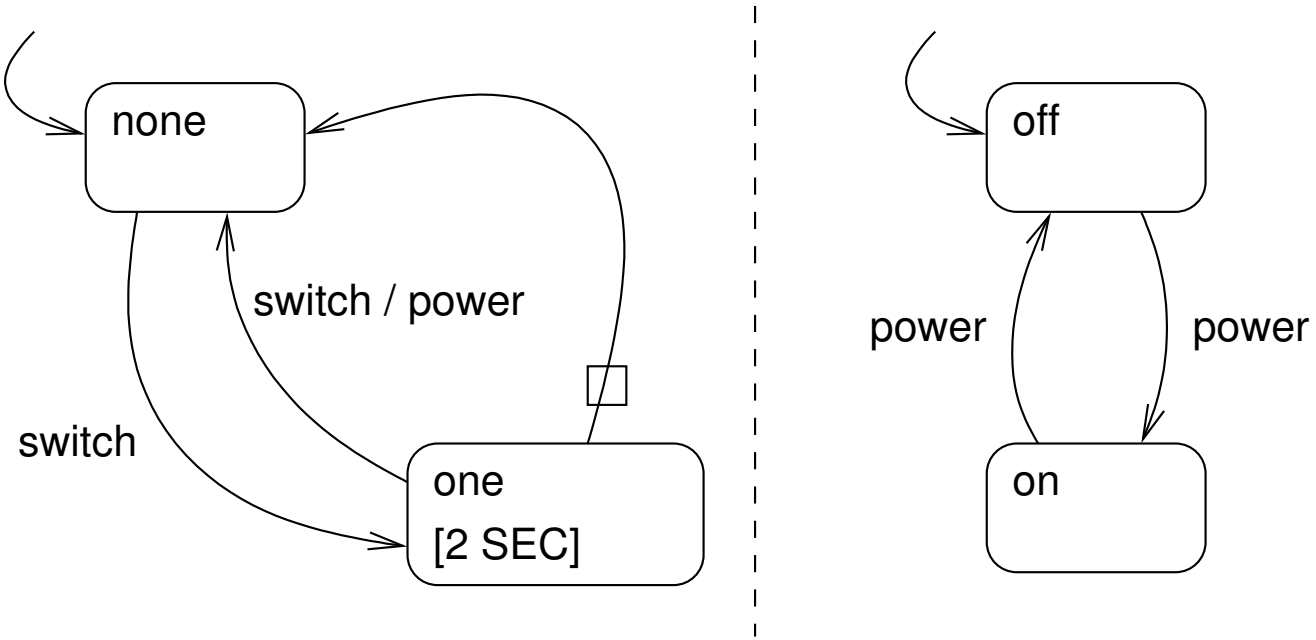
# Syncblock Implementation



# Syncblock Implementation

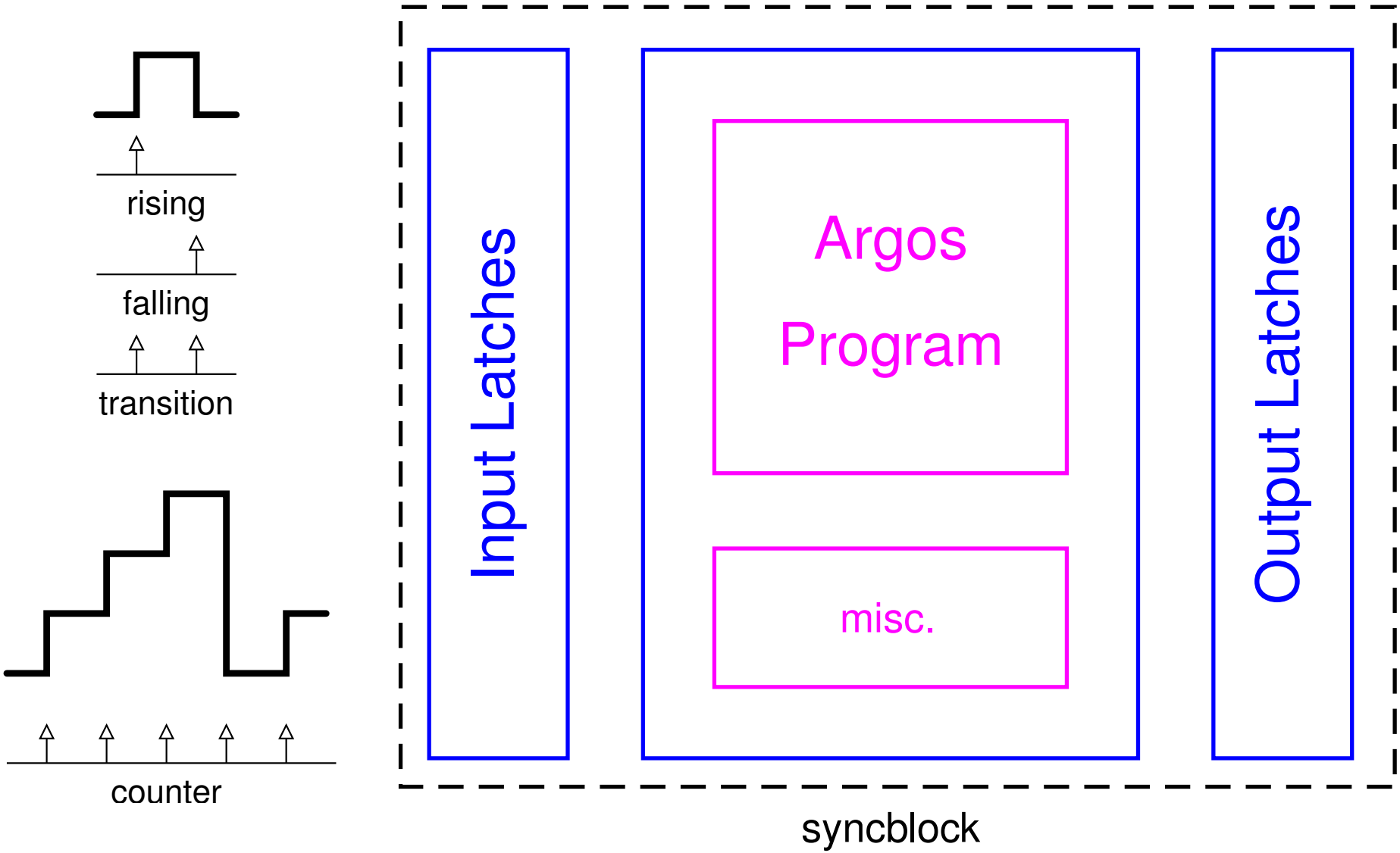


# Compilation [MH96]

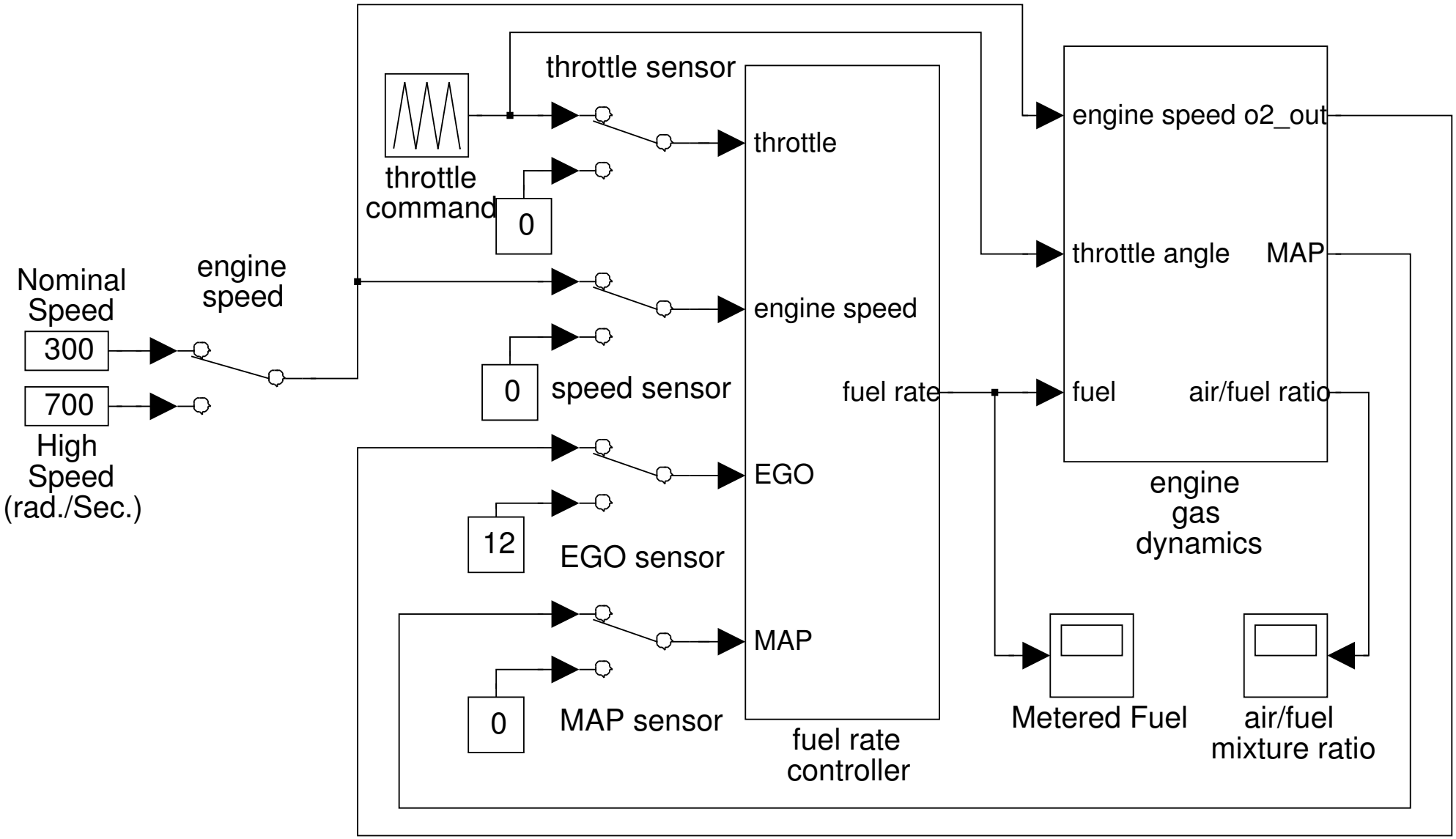


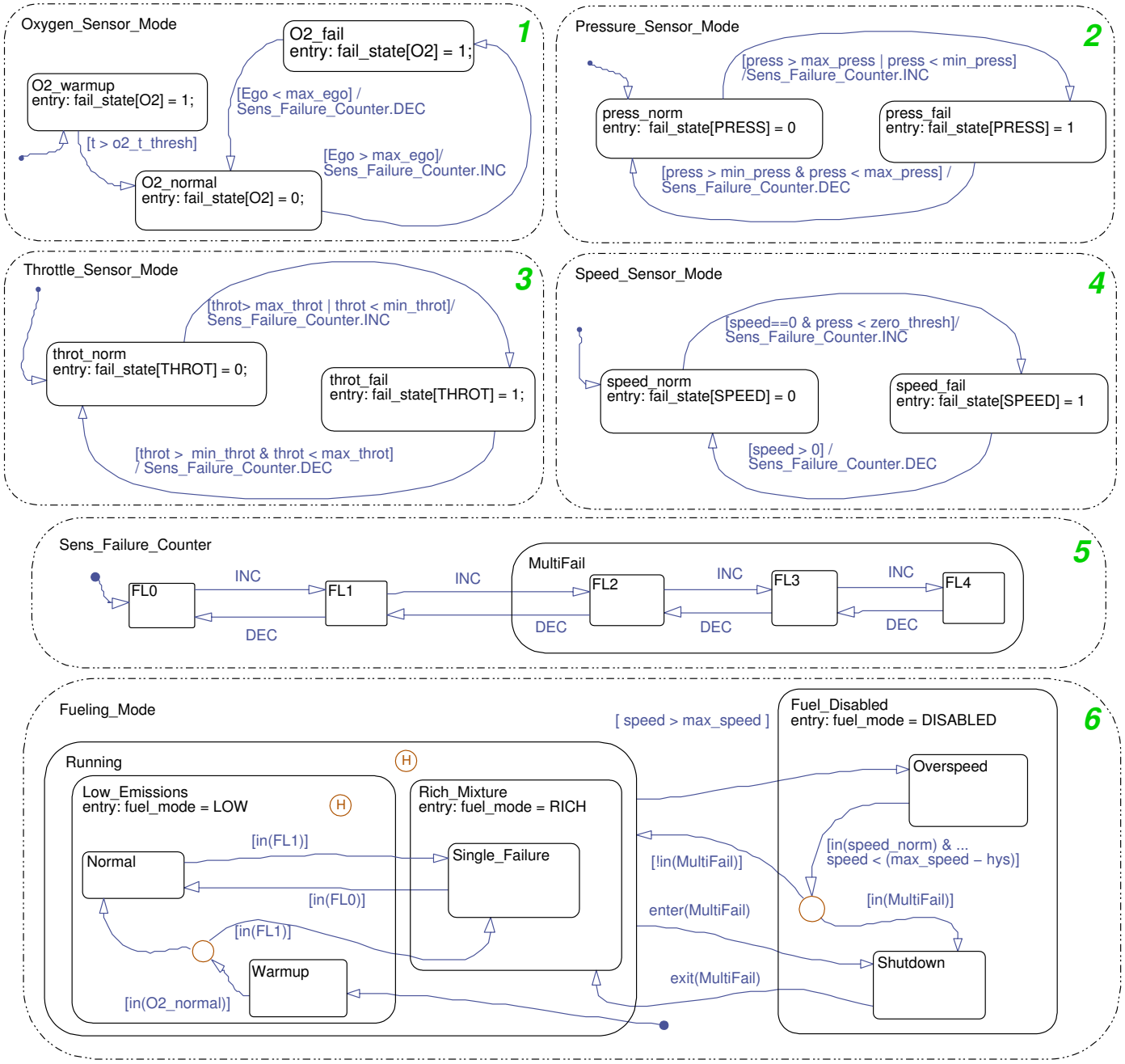

```
power   = (one && switch)
none'   = (one && timeout && !switch) || (one && switch)
          || (none && !switch)
one'    = (none && switch) || (one && !timeout && !switch)
off'    = (off && !power) || (on && power)
on'     = (on && !power) || (off && power)
```

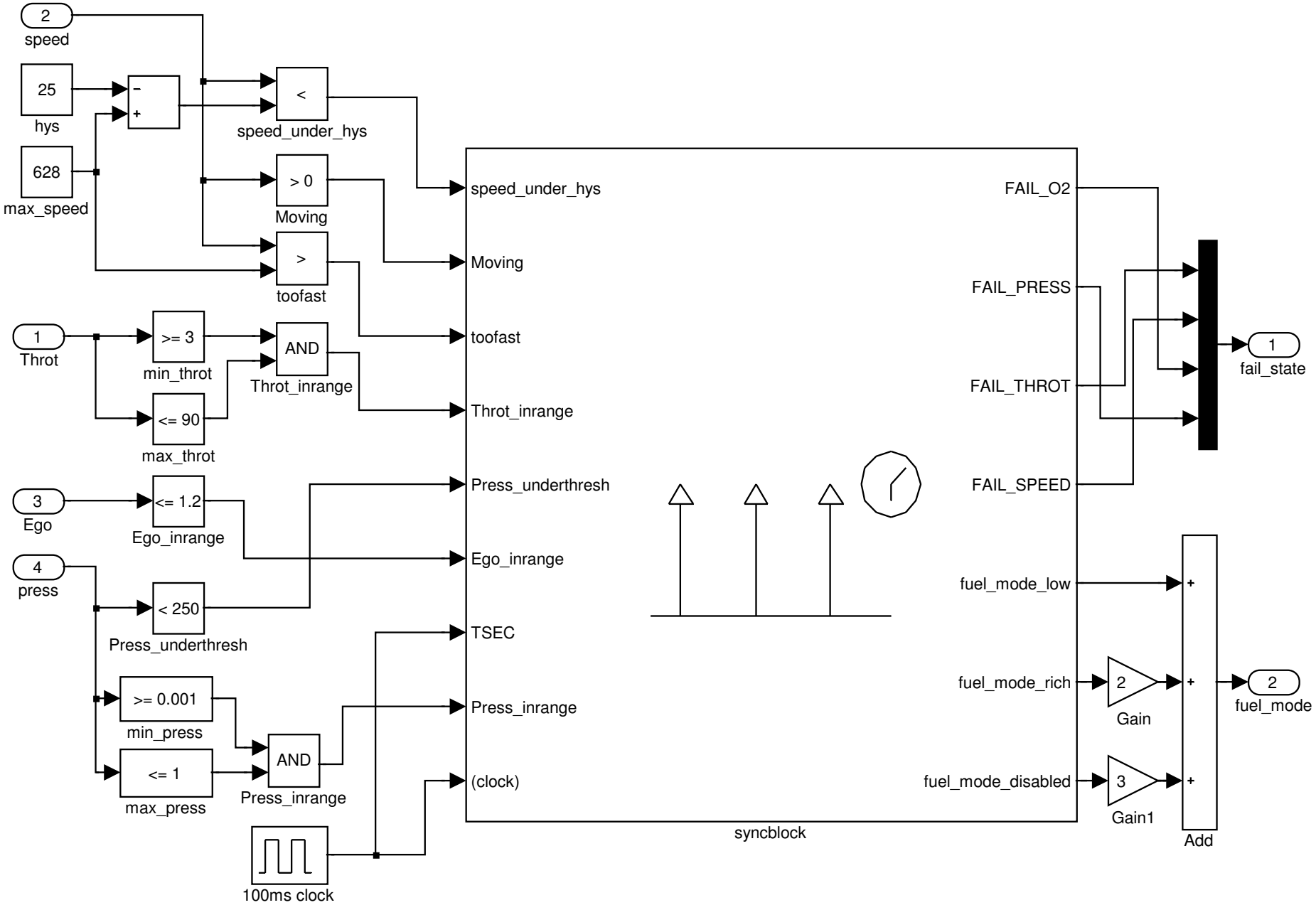
# Block interfacing

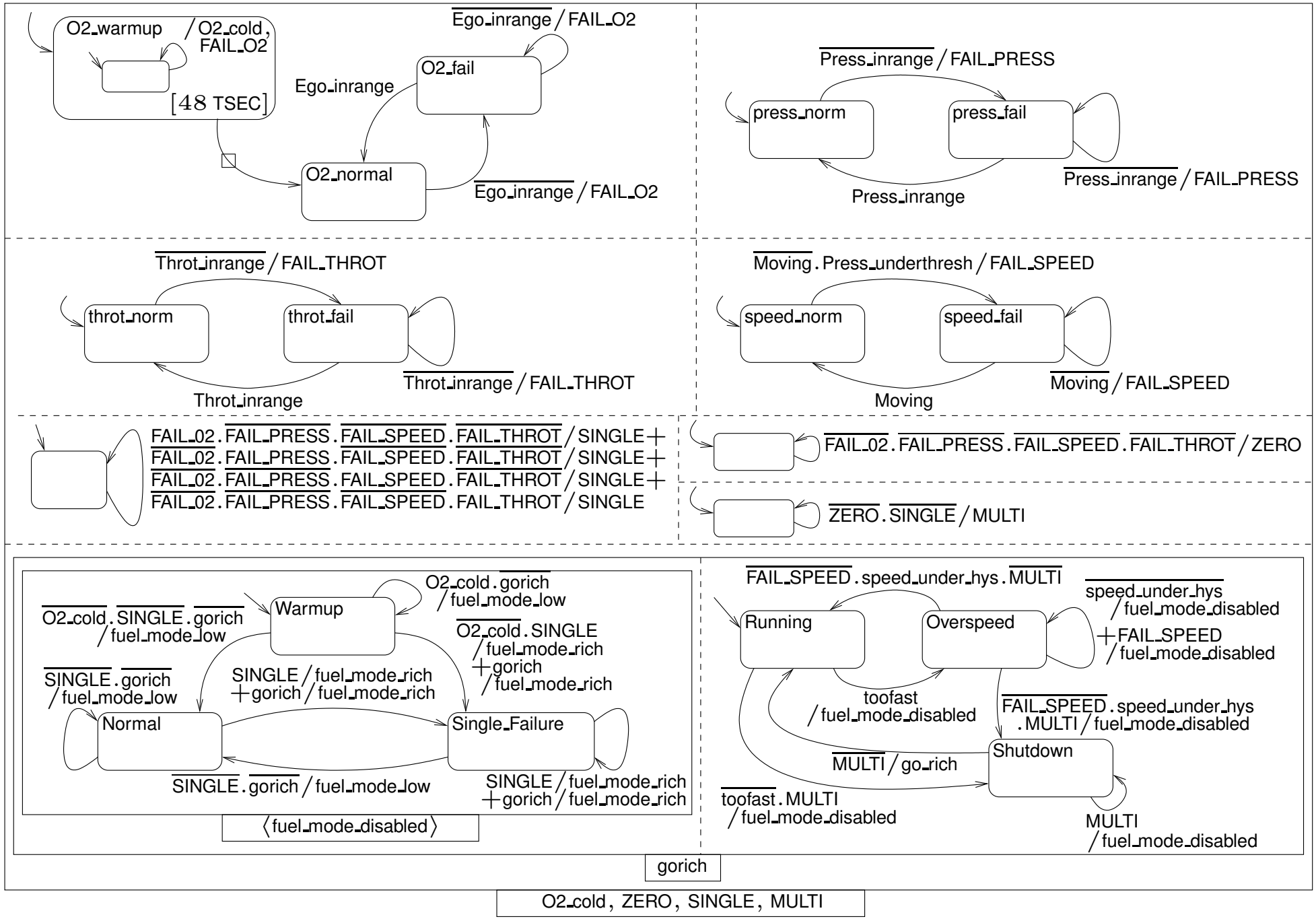


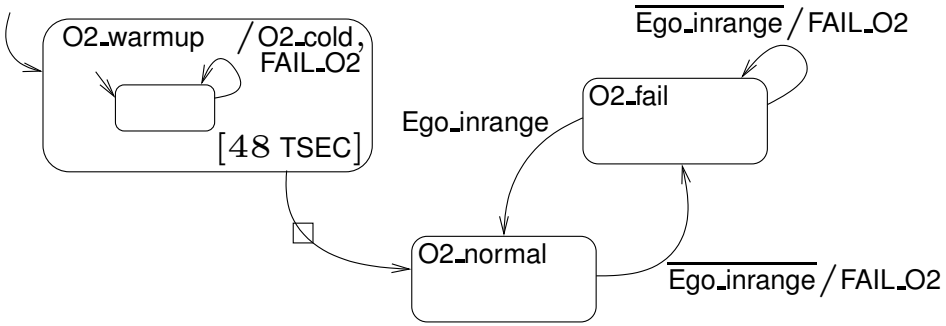
# Example: Fault-Tolerant Fuel Control System [Mat]

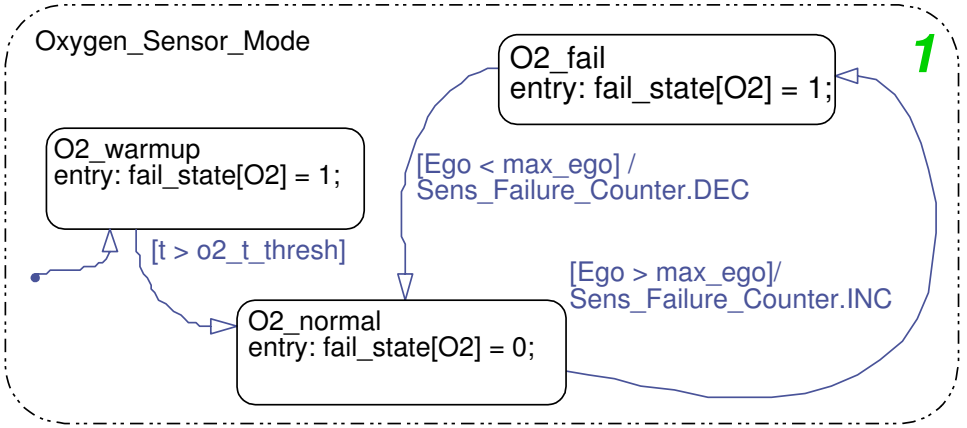


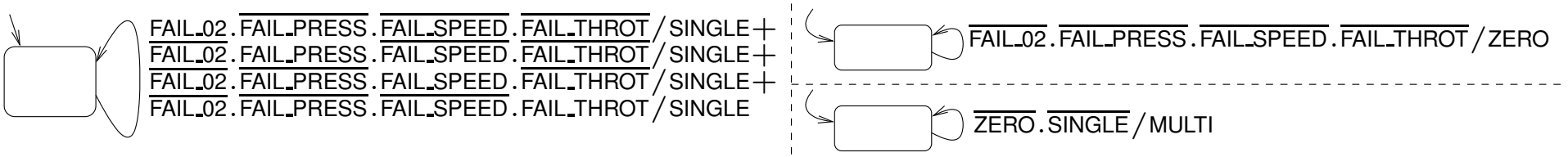

Oxygen_Sensor_Mode
1
O2_fail
entry: fail_state[O2] = 1;
O2_warmup
entry: fail_state[O2] = 1;
[Ego < max_ego] /
Sens_Failure_Counter.DEC
[t > o2_t_thresh]
[Ego > max_ego]/
Sens_Failure_Counter.INC
O2_normal
entry: fail_state[O2] = 0;
Pressure_Sensor_Mode
2
[press > max_press | press < min_press]
/Sens_Failure_Counter.INC
press_norm
entry: fail_state[PRESS] = 0
press_fail
entry: fail_state[PRESS] = 1
[press > min_press & press < max_press] /
Sens_Failure_Counter.DEC
Throttle_Sensor_Mode
3
[throt> max_throt | throt < min_throt]/
Sens_Failure_Counter.INC
throt_norm
entry: fail_state[THROT] = 0;
throt_fail
entry: fail_state[THROT] = 1;
[throt > min_throt & throt < max_throt]
/ Sens_Failure_Counter.DEC
Speed_Sensor_Mode
4
[speed==0 & press < zero_thresh]/
Sens_Failure_Counter.INC
speed_norm
entry: fail_state[SPEED] = 0
speed_fail
entry: fail_state[SPEED] = 1
[speed > 0] /
Sens_Failure_Counter.DEC
Sens_Failure_Counter
5
FL0
INC
FL1
INC
MultiFail
FL2
INC
FL3
INC
FL4
DEC
DEC
DEC
DEC
Fueling_Mode
6
[ speed > max_speed ]
Fuel_Disabled
entry: fuel_mode = DISABLED
Overspeed
[in(speed_norm) & ...
speed < (max_speed – hys)]
[in(MultiFail)]
Shutdown
Running
H
Low_Emissions
entry: fuel_mode = LOW
H
Rich_Mixture
entry: fuel_mode = RICH
[in(FL1)]
Normal
Single_Failure
[in(FL0)]
[in(FL1)]
Warmup
[in(O2_normal)]
[!in(MultiFail)]
enter(MultiFail)
exit(MultiFail)

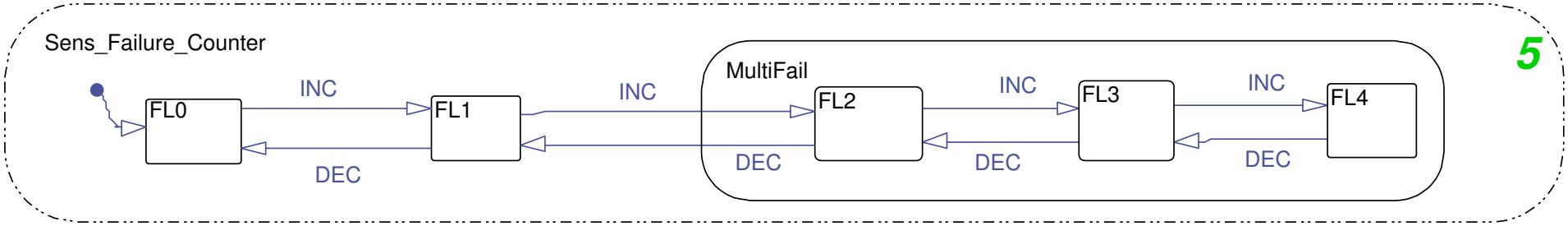

2
speed
25
hys
628
max_speed
speed_under_hys
Moving
toofast
1
Throt
min_throt
Throt_inrange
max_throt
3
Ego
Ego_inrange
4
press
Press_underthresh
min_press
max_press
Press_inrange
100ms clock
speed_under_hys
Moving
toofast
Throt_inrange
Press_underthresh
Ego_inrange
TSEC
Press_inrange
(clock)
syncblock
FAIL_O2
FAIL_PRESS
FAIL_THROT
FAIL_SPEED
fuel_mode_low
fuel_mode_rich
fuel_mode_disabled
Gain
Gain1
Add
1
fail_state
2
fuel_mode

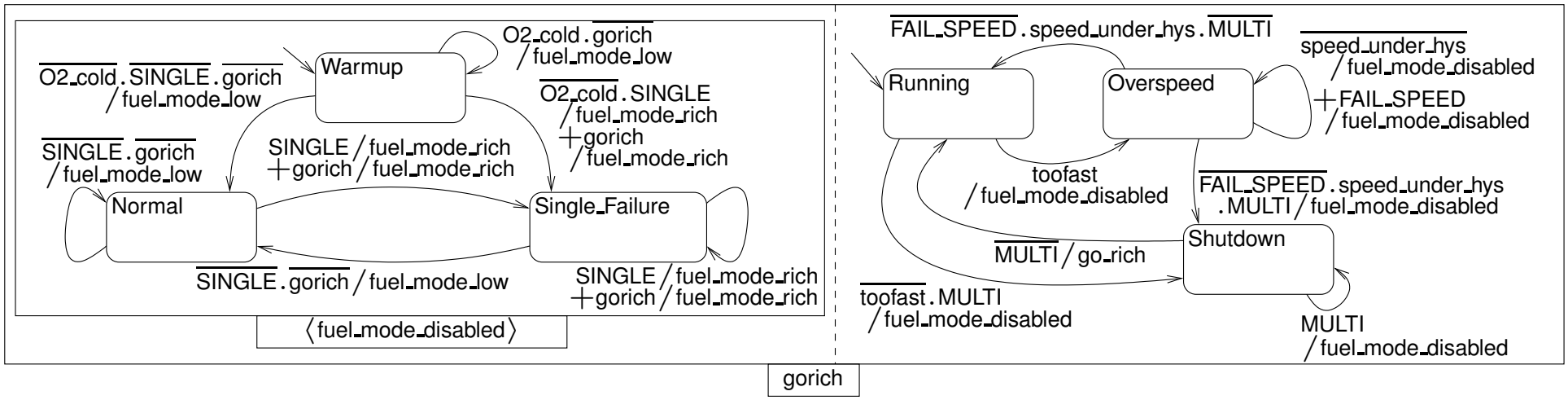

O2_warmup
/ O2_cold, FAIL_O2
[48 TSEC]
Ego_inrange / FAIL_O2
O2_fail
Ego_inrange
O2_normal
Ego_inrange / FAIL_O2
Press_inrange / FAIL_PRESS
press_norm
press_fail
Press_inrange
Press_inrange / FAIL_PRESS
Throt_inrange / FAIL_THROT
throt_norm
throt_fail
Throt_inrange
Throt_inrange / FAIL_THROT
Moving . Press_underthresh / FAIL_SPEED
speed_norm
speed_fail
Moving
Moving / FAIL_SPEED
FAIL_02 . FAIL_PRESS . FAIL_SPEED . FAIL_THROT / SINGLE +
FAIL_02 . FAIL_PRESS . FAIL_SPEED . FAIL_THROT / SINGLE +
FAIL_02 . FAIL_PRESS . FAIL_SPEED . FAIL_THROT / SINGLE +
FAIL_02 . FAIL_PRESS . FAIL_SPEED . FAIL_THROT / SINGLE
FAIL_02 . FAIL_PRESS . FAIL_SPEED . FAIL_THROT / ZERO
ZERO . SINGLE / MULTI
O2_cold . gorich / fuel_mode_low
Warmup
O2_cold . SINGLE . gorich / fuel_mode_low
O2_cold . SINGLE / fuel_mode_rich + gorich / fuel_mode_rich
SINGLE . gorich / fuel_mode_low
SINGLE / fuel_mode_rich + gorich / fuel_mode_rich
Normal
Single_Failure
SINGLE . gorich / fuel_mode_low
SINGLE / fuel_mode_rich + gorich / fuel_mode_rich
⟨fuel_mode_disabled⟩
FAIL_SPEED . speed_under_hys . MULTI
speed_under_hys / fuel_mode_disabled
Running
Overspeed
+ FAIL_SPEED / fuel_mode_disabled
toofast / fuel_mode_disabled
FAIL_SPEED . speed_under_hys . MULTI / fuel_mode_disabled
MULTI / go_rich
Shutdown
toofast . MULTI / fuel_mode_disabled
MULTI / fuel_mode_disabled
gorich
O2_cold, ZERO, SINGLE, MULTI

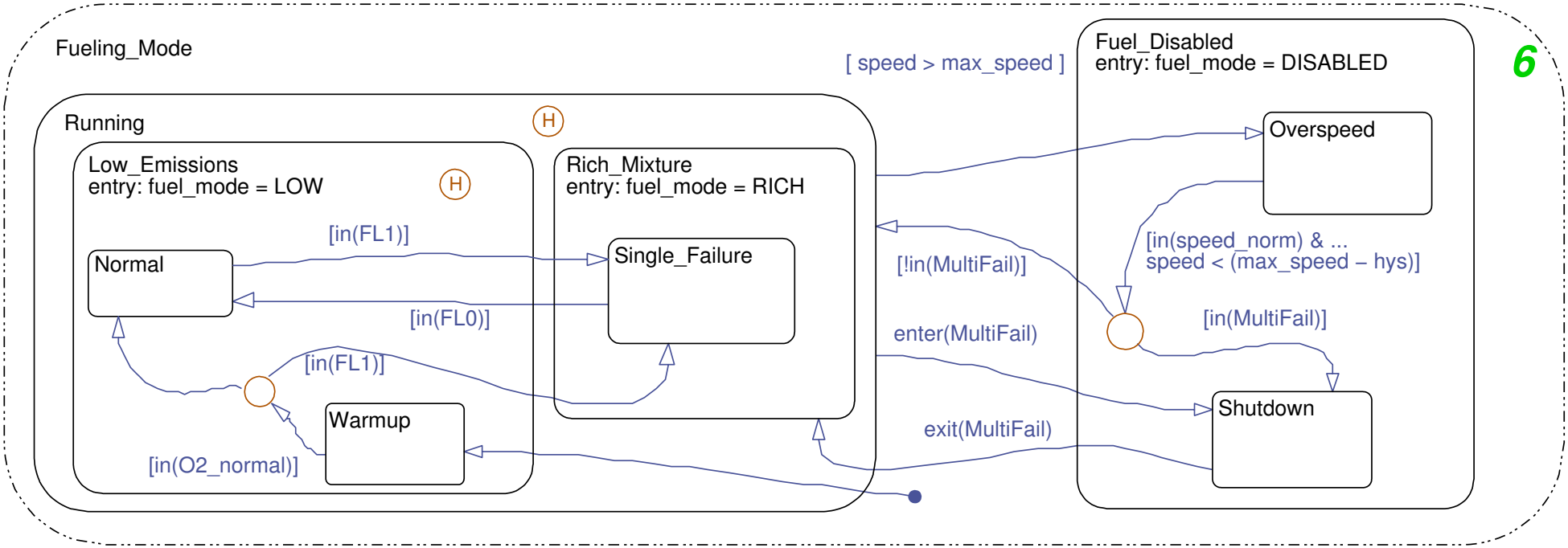

O2_warmup
/ O2_cold,
FAIL_O2
[48 TSEC]
Ego_inrange / FAIL_O2
O2_fail
Ego_inrange
O2_normal
Ego_inrange / FAIL_O2


Oxygen_Sensor_Mode
1
O2_fail
entry: fail_state[O2] = 1;
O2_warmup
entry: fail_state[O2] = 1;
[Ego < max_ego] /
Sens_Failure_Counter.DEC
[t > o2_t_thresh]
[Ego > max_ego]/
Sens_Failure_Counter.INC
O2_normal
entry: fail_state[O2] = 0;


FAIL_02 . FAIL_PRESS . FAIL_SPEED . FAIL_THROT / SINGLE +
FAIL_02 . FAIL_PRESS . FAIL_SPEED . FAIL_THROT / SINGLE +
FAIL_02 . FAIL_PRESS . FAIL_SPEED . FAIL_THROT / SINGLE +
FAIL_02 . FAIL_PRESS . FAIL_SPEED . FAIL_THROT / SINGLE
FAIL_02 . FAIL_PRESS . FAIL_SPEED . FAIL_THROT / ZERO
ZERO . SINGLE / MULTI


Sens_Failure_Counter
5
FL0
FL1
FL2
FL3
FL4
MultiFail
INC
INC
INC
INC
DEC
DEC
DEC
DEC


Warmup
O2_cold . gorich / fuel_mode_low
O2_cold . SINGLE . gorich / fuel_mode_low
O2_cold . SINGLE / fuel_mode_rich + gorich / fuel_mode_rich
SINGLE . gorich / fuel_mode_low
SINGLE / fuel_mode_rich + gorich / fuel_mode_rich
Normal
Single_Failure
SINGLE . gorich / fuel_mode_low
SINGLE / fuel_mode_rich + gorich / fuel_mode_rich
⟨fuel_mode_disabled⟩
FAIL_SPEED . speed_under_hys . MULTI
Running
Overspeed
speed_under_hys / fuel_mode_disabled
+ FAIL_SPEED / fuel_mode_disabled
toofast / fuel_mode_disabled
FAIL_SPEED . speed_under_hys . MULTI / fuel_mode_disabled
MULTI / go_rich
Shutdown
toofast . MULTI / fuel_mode_disabled
MULTI / fuel_mode_disabled
gorich


Fueling_Mode
[ speed > max_speed ]
Fuel_Disabled
entry: fuel_mode = DISABLED
6
Running
H
Low_Emissions
entry: fuel_mode = LOW
H
Rich_Mixture
entry: fuel_mode = RICH
Overspeed
[in(FL1)]
Normal
Single_Failure
[in(FL0)]
[in(FL1)]
Warmup
[in(O2_normal)]
[!in(MultiFail)]
enter(MultiFail)
exit(MultiFail)
[in(speed_norm) & ...
speed < (max_speed – hys)]
[in(MultiFail)]
Shutdown

## Summary

- Stateflow is powerful but has shortcomings
- Existing research might help
- Argos block developed:
  - contrast with Stateflow
  - simple examples possible
  - paucity of features has pros and cons

**Need the right tool for the task at hand.**

# References


[CCM+03] Paul Caspi, Adrian Curic, Aude Maignan, Christos Sofronis, Stavros Tripakis, and Peter Niebert. From Simulink to SCADE/Lustre to TTA: a layered approach for distributed embedded applications. In *Proc. 2003 ACM SIGPLAN conference on Languages, Compilers, and Tools for Embedded Systems (LCTES '03)*, pages 153–162. ACM Press, 2003.

[Mar91] F. Maraninchi. The Argos language: Graphical representation of automata and description of reactive systems. In *Proc. IEEE Workshop on Visual Languages*, pages 254–259, October 1991.

[Mat] Mathworks. Fault-tolerant fuel control system. Matlab/Simulink/Stateflow example model.

[MH96] F. Maraninchi and N. Halbwachs. Compiling Argos into boolean equations. In Bengt Jonsson and Joachim Parrow, editors, *Proc. 4th International Symposium on Formal Techniques for Real-Time and Fault-Tolerance (FTRTFT '96)*, volume 1135 of *Lecture Notes in Computer Science*, pages 72–89, Uppsala, Sweden, September 1996. Springer-Verlag.

[MR01] Florence Maraninchi and Yann Rémond. Argos: an automaton-based synchronous

language. *Computer Languages*, 27(1–3):61–92, 2001.

[SSC+04] N. Scaife, C. Sofronis, P. Caspi, S. Tripakis, and F. Maraninchi. Defining and translating a "safe" subset of Simulink/Stateflow into Lustre. In G. Buttazzo and S. Edwards, editors, *Proc. 4th ACM International Conference on Embedded Software (EMSOFT'04)*, pages 259 – 268, Pisa, Italy, September 2004. ACM, ACM Press.