New Zealand

The Synchronous Approach

Partha Roop
Electrical & computer engineering

27 July 2022

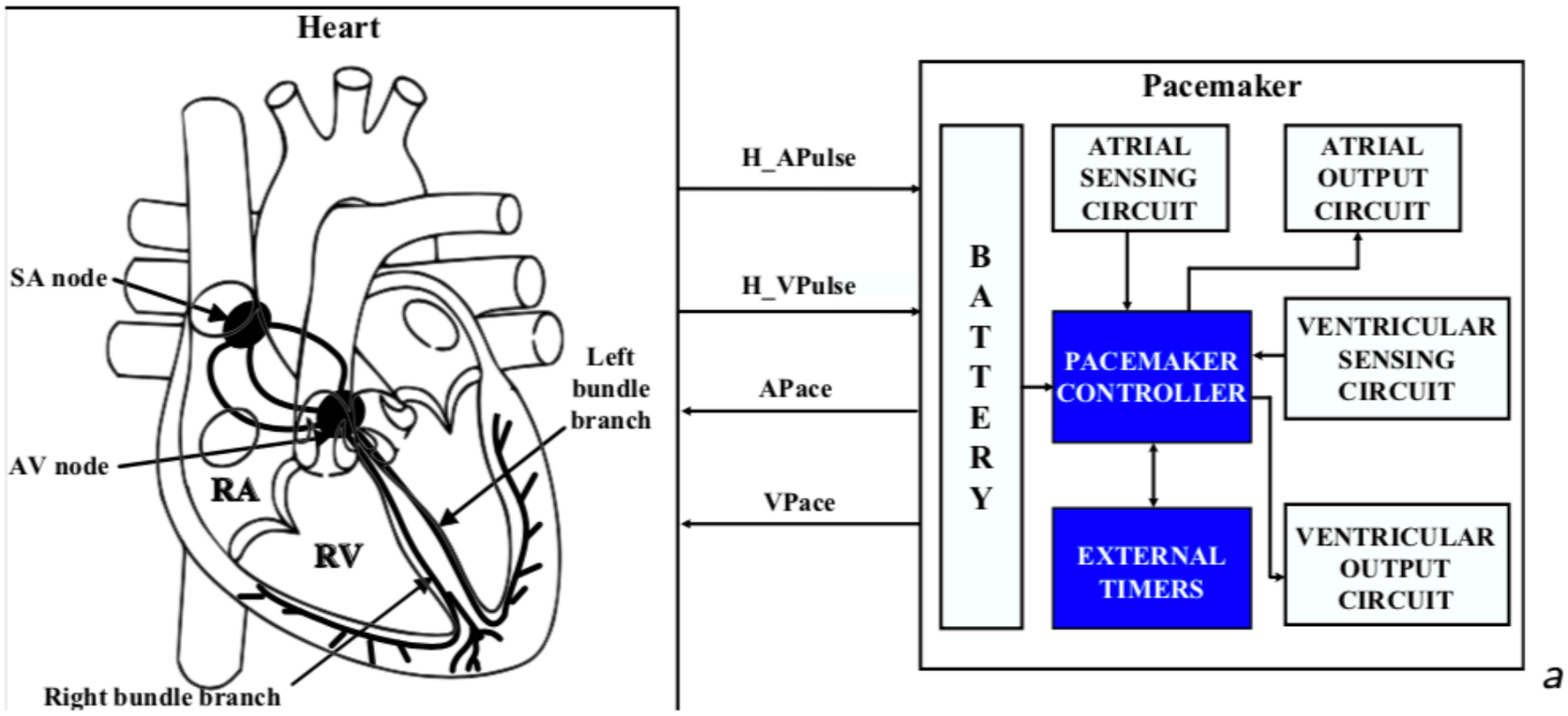
THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

The University of Auckland

Acknowledgement: these slides are mainly based on **Prof. Reinhard von Hanxleden**'s Lecture on SCCharts at **Kiel University, Germany**

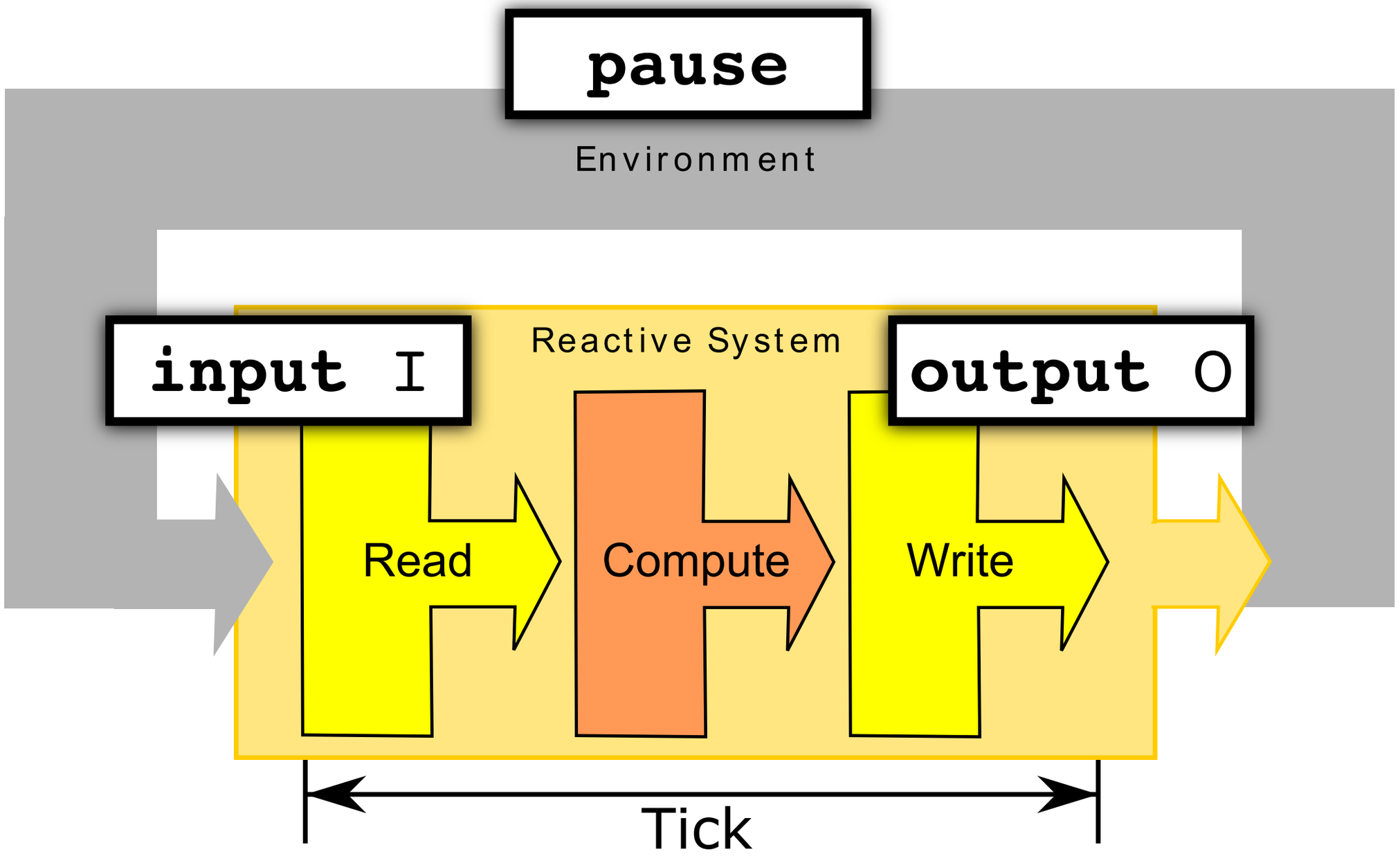
CPS Example

A cyber-physical system (CPS)

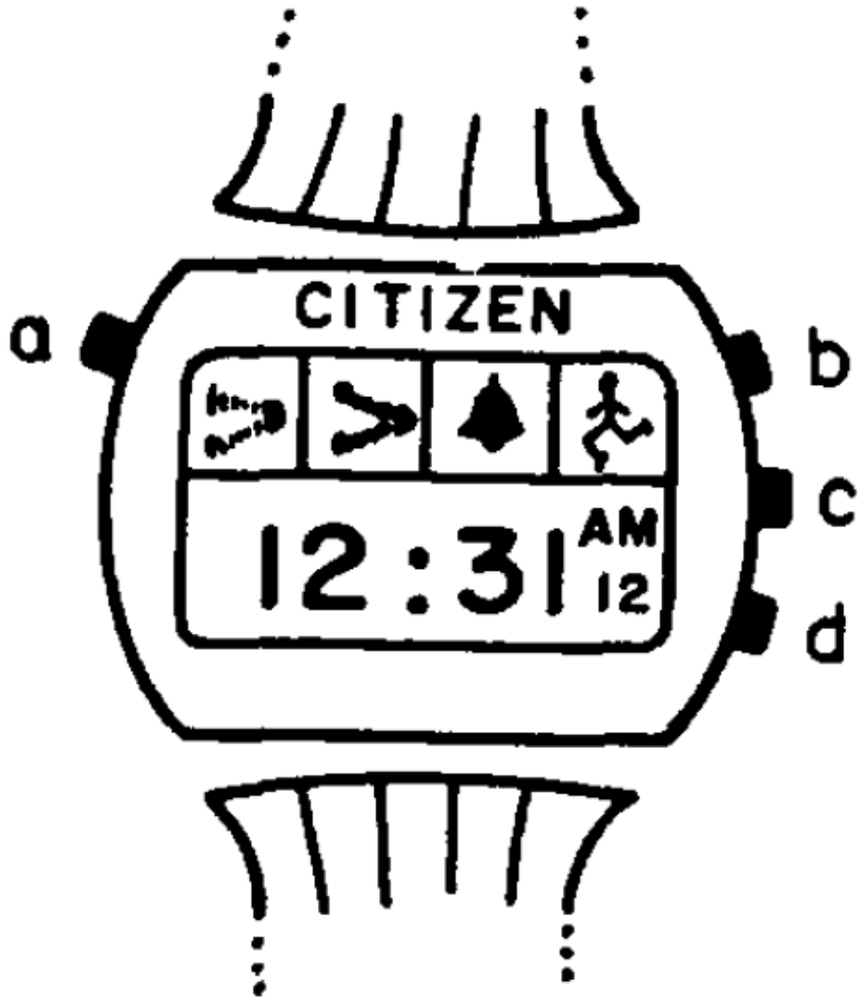


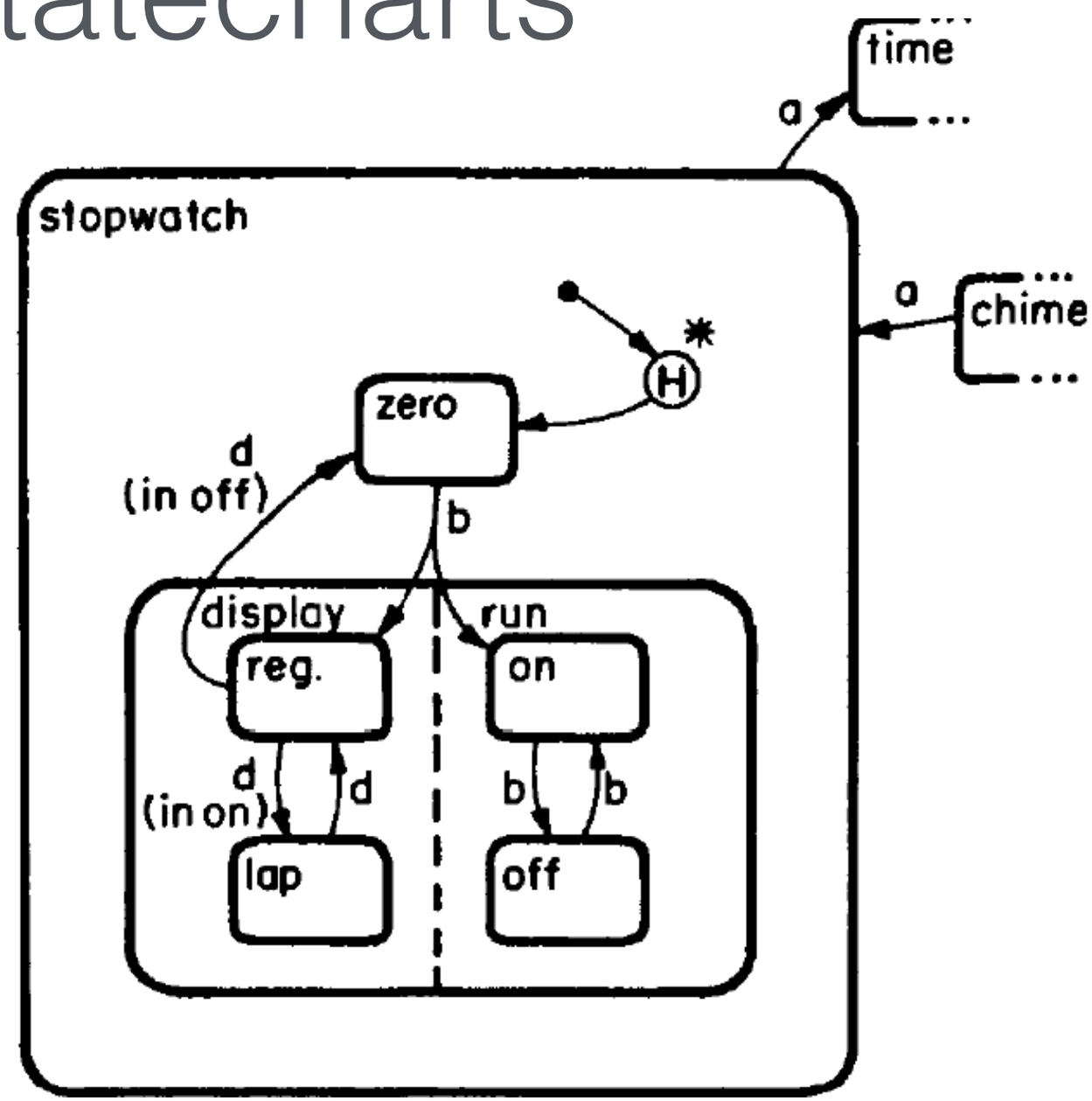
[a] Zhao and Roop, "Model Driven Design of Cardiac Pacemakers using IEC61499, CRC Press, 2015".

Reactive Systems



1980s: Statecharts





Harel
Statecharts: A Visual Formalism for Complex Systems
Science of Computer Programming, 1987

1990s: Many Statecharts

Statecharts Feature	Variant 1	2	3	4	5	6	7	8	9	10
graphical / textual	g	g	g	g	g	g	g	g/t	g/t	g
negated trigger event	+	+	+	+	+	+	+	+	+	-
timeout event	+	-	-	-	-	-	-	+	+	+[9]
timed transition	-	-	-	-	-	-	-	-	-	-
disjunction of trigger events	-	+	+	+	+	-[13]	-[13]	+	+	-

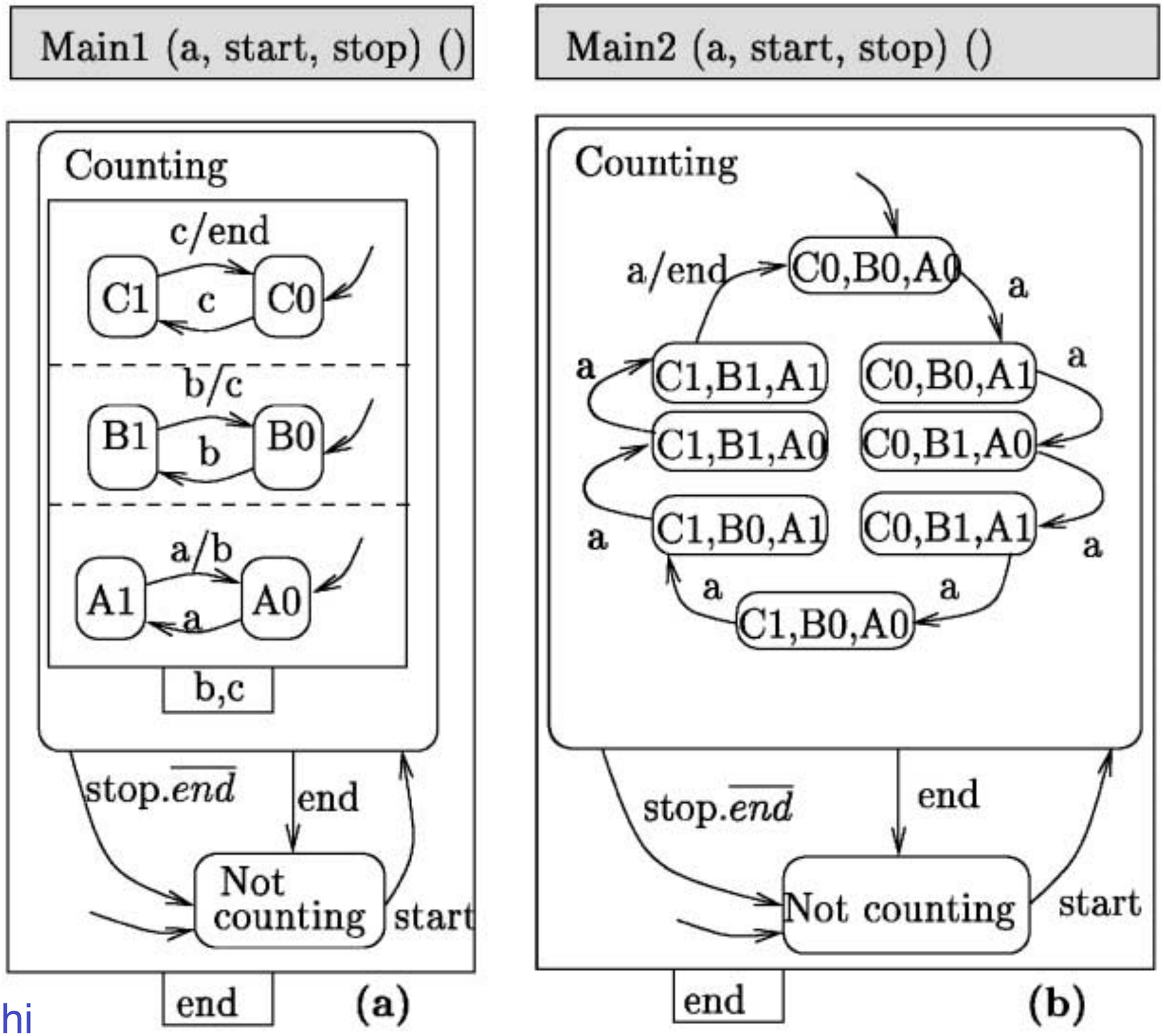
Statecharts Feature	Variant Number 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
graphical / textual	g	g	g	g	g	g	g	g/t	g/t	g	g	g/t	g	g	g	g	g	g	g	g	g
negated trigger event	+	+	+	+	+	+	+	+	+	-	+	-	+	+	-	-	-	-	+	+	+
timeout event	+	-	-	-	-	-	-	+	+[9]	-	-	-	+	+[4]	+[4]	+[4]	+[4]	+[4]	+[4]	+[4]	+[4]
timed transition	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	+	+	+[2]	+[2]	+
disjunction of trigger events	-	+	+	+	+	-[13]	-[13]	+	+	-	-	-	+	+	-	-	-	-	-	-	+
trigger condition	+	+	-	-	-	-[13]	-[13]	+[3]	-	+	-	-	-	+	-	-	-	-	+	+	-
state reference	+	+	-	-	-	-[13]	-[13]	+[3]	-	+	-	-	-	+	-	-	-	-	-	-	-
assignment to variable	+	+	-	-	-	-[13]	-[13]	-	-	+	-	-	-	+	-	-	-	-	+	+	-
inter-level transition	+	+	-[14]	-[14]	-[14]	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	-
history mechanism	+	+	-	-	-	-	-	-	-	-	-	-	+	-	-	-	-	-	-	-	-
operational/denotatio.	-	o	o	o	o	o	[11]	d	d	o	o	d	o	o	d	d	d	d	o	o	?
compositional	-	-	-	-	-	-	+	+	-	+	+		-	+	+	+	+	+	+	+	+
synchrony hypothesis	+	+	+	+	+	+	+	+	+[7]	+	+	+	-	-	-	-	-	-[8]	-[8]	-	
deterministic	-	-	-	-	-	-	-	-	-	+	-	-	-	-	-	-	-	-	-	-	-
interleav./true concurr.	i	i	i	i	i	i	i	i	i	i	i	i	i	t	t	t	t	[15]	[15]	?	
discrete/contin. time	d	d	d	d	d	d	d	d	d	d	d	d	d	c	c	c	c	c	c	c	
globally consistent	-	-	-	+	+	+	+[13]	+	+[5]	-	+[5]	-	-	+[5]	+[5]	+[5]	+[5]	-	-	-	
causal	+	+	+	+	-	+	+	+	+	+[1]	-[12]	+	+	+	+	+	+	+	+	+	+
instantaneous state	?	-	-	-	-	-	-	-	+	-	+	-	-	-	-	+	+	+	+	+	
finite transition no.	?	+	+	+	+	+	+	+	-	+	-	+	+	+	+	+	+	-	-	+	
priorities	-	-	-[14]	-[14]	-[14]	-[10]	-[10]	-	-	+	-	-	+	-	+	-	+	-	-	+	
non-preempt. interrupt	?	-	-[14]	-[14]	-[14]	-	-	-	-	?	+	-	-	-	-	+	+	?	?	+	
preemptive interrupt	?	+	-[14]	-[14]	-[14]	+	+	+	+	+	-	+	+	+	+	-	-	+	+	+	
distinc. int./ext. event	+	+	+	+	+	+	+	+	+[6]	+	+	+	-	+	+	+	+	+	+	+	
local event	-	-	-	-	-	-	-	+	+	+	+	+	-	-	-	-	-	-	-	-	+
discrete/contin. event	d	d	d	d	d	d	d	d	d	d	d	d	d	c	c	c	c	d	d	d	

Michael von der Beeck

A Comparison of Statecharts Variants

Formal Techniques in Real-Time and Fault-Tolerant Systems, LNCS 863, 1994

1991: Argos

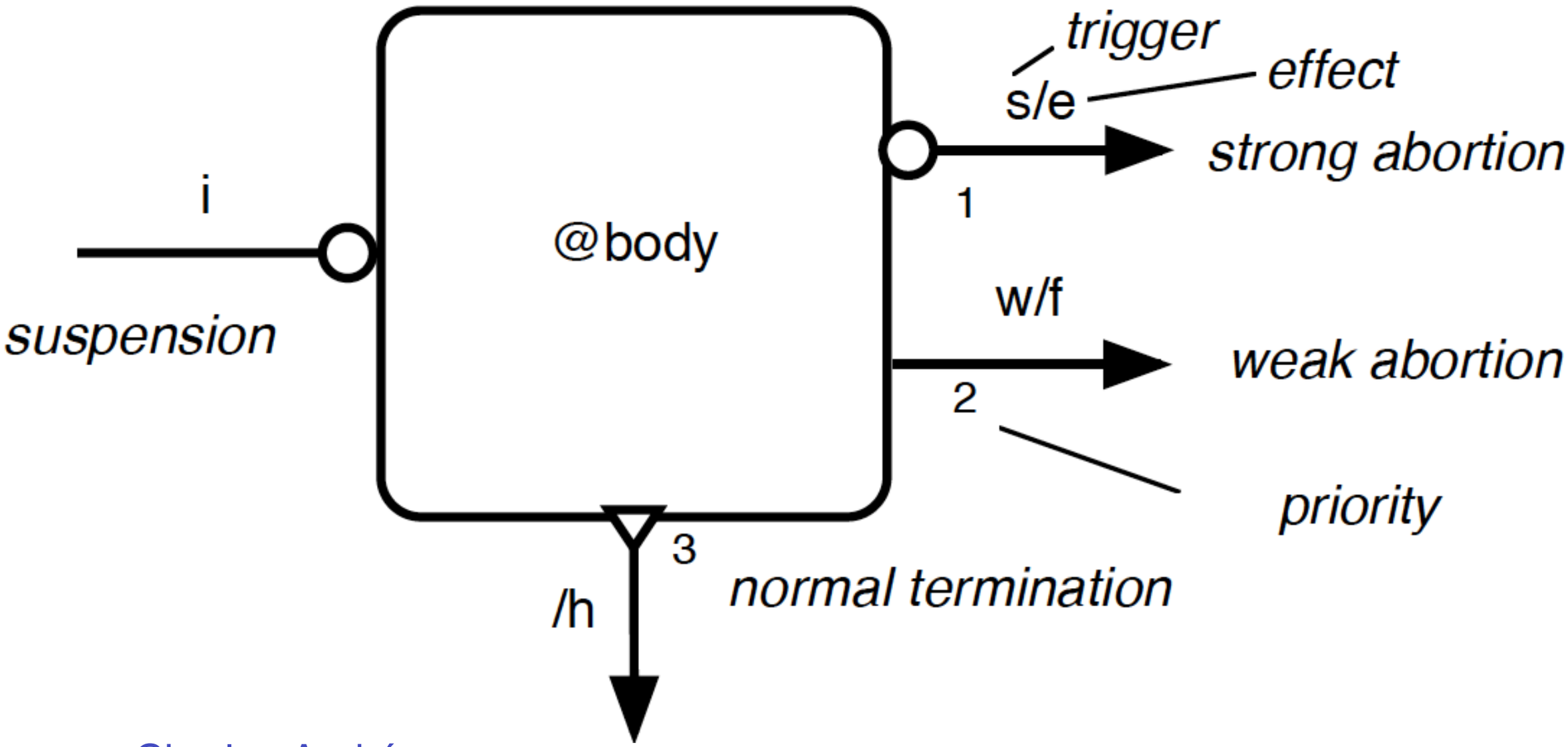


Florence Maraninchi

The Argos language: Graphical Representation of Automata and Description of Reactive Systems

IEEE Workshop on Visual Languages, Kobe, Japan, 1991

1995: SyncCharts, a.k.a. Safe State Machines



Charles André
SyncCharts: A Visual Representation of Reactive Behaviors
Research Report 95-52, I3S, Sophia Antipolis, 1995

SCCharts – Motivation

Preserve nice properties of synchronous programming

- Determinacy, sound semantic basis
- Static causality (i.e., determinacy) checking, no run-time surprises
- Efficient synthesis

Reduce the pain

- Make it easy to adopt for mainstream programmer
- Reject only models where determinacy is compromised
- Approach: harness scheduling information provided by sequential/imperative language constructs

Interface declaration

Initialization

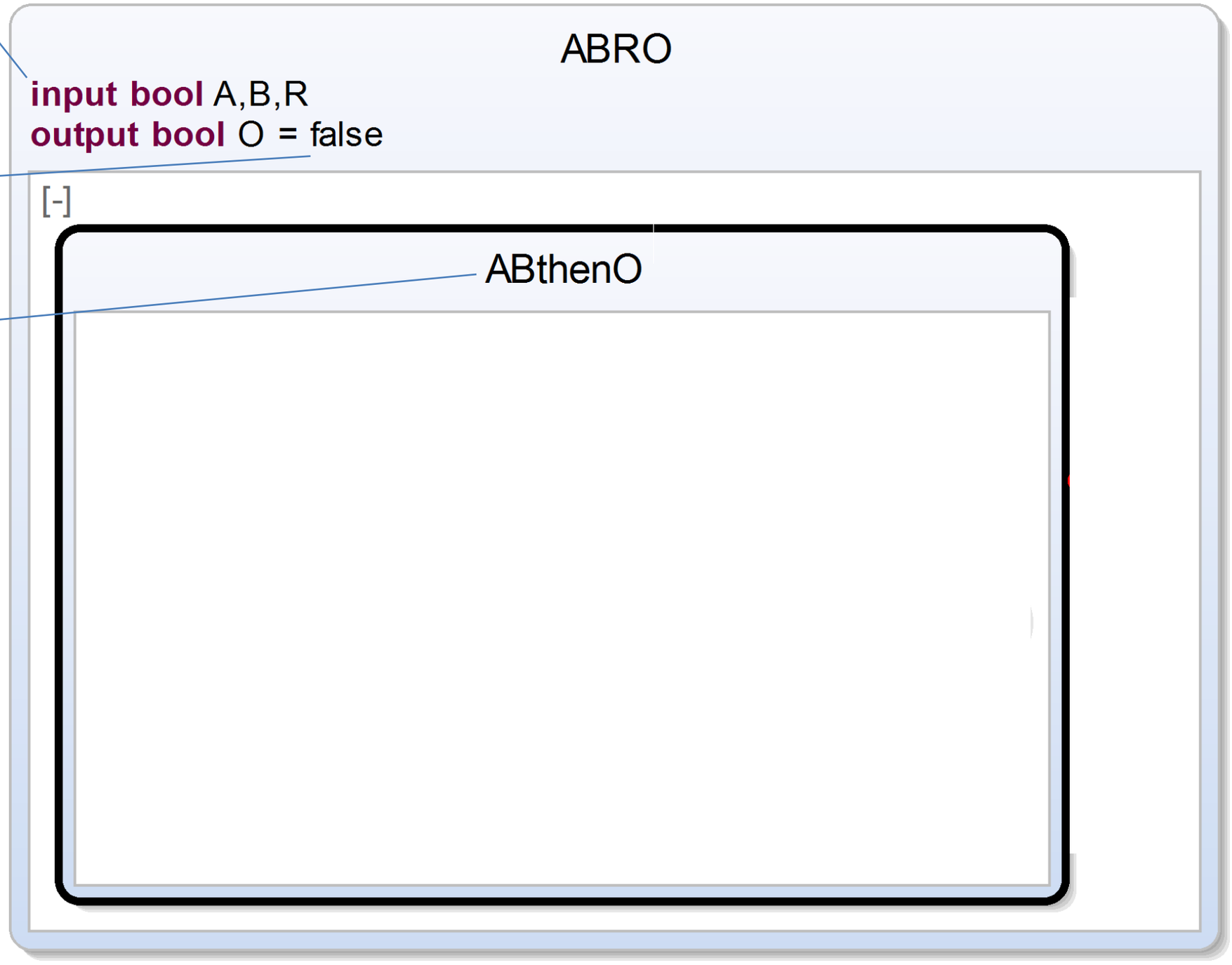
ABRO

```
input bool A,B,R
output bool O = false
```

Interface declaration

Initialization

Superstate

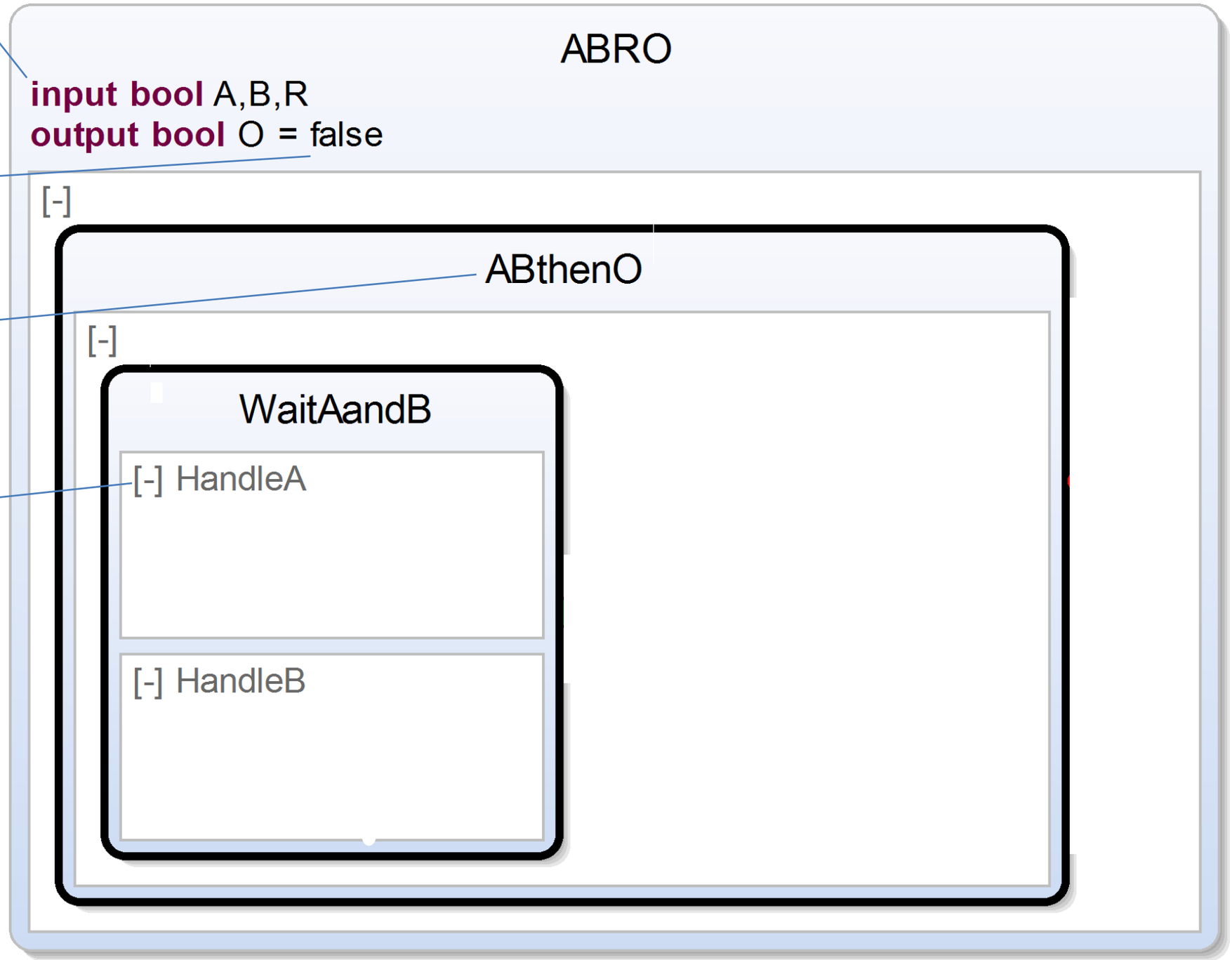


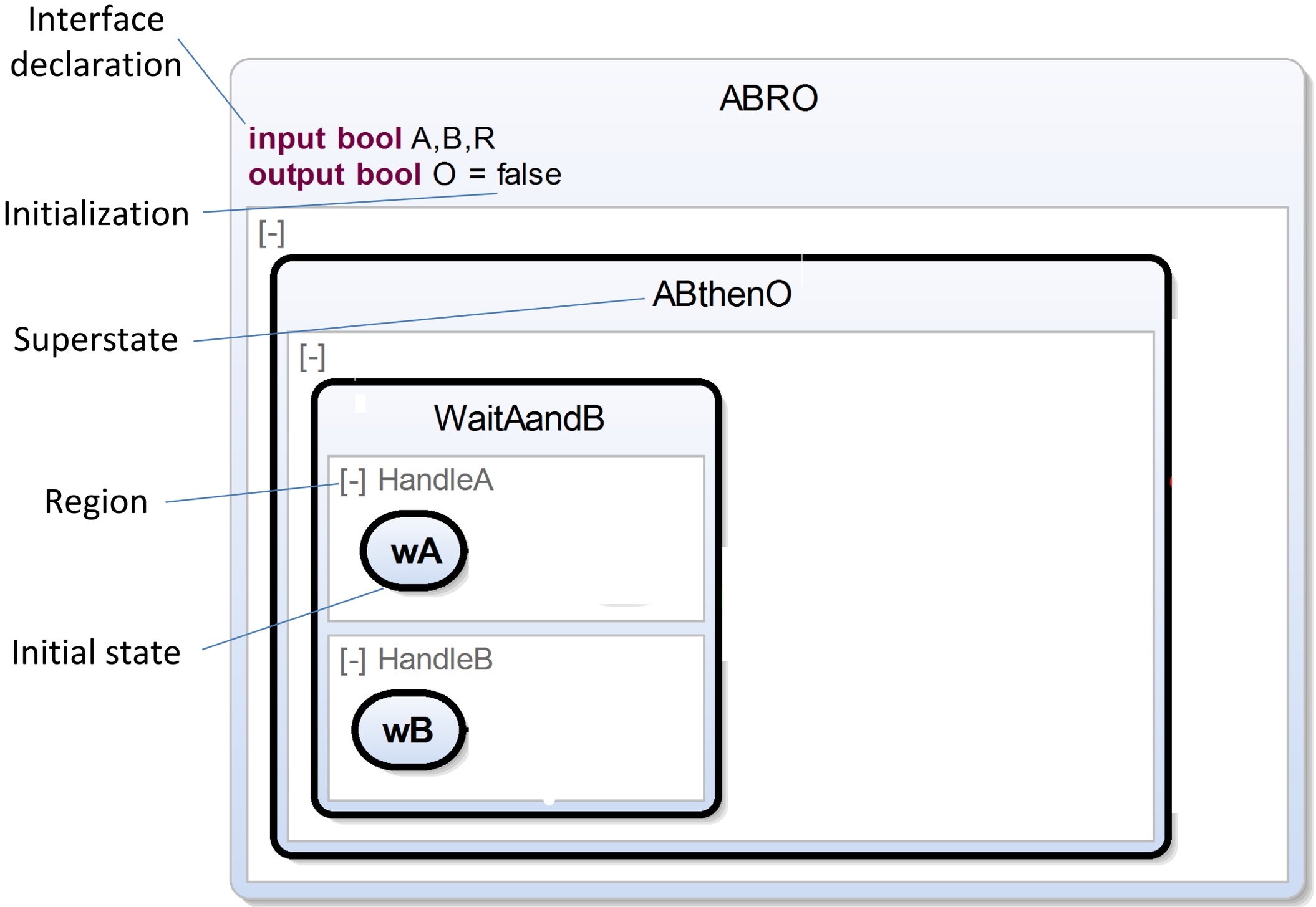
Interface declaration

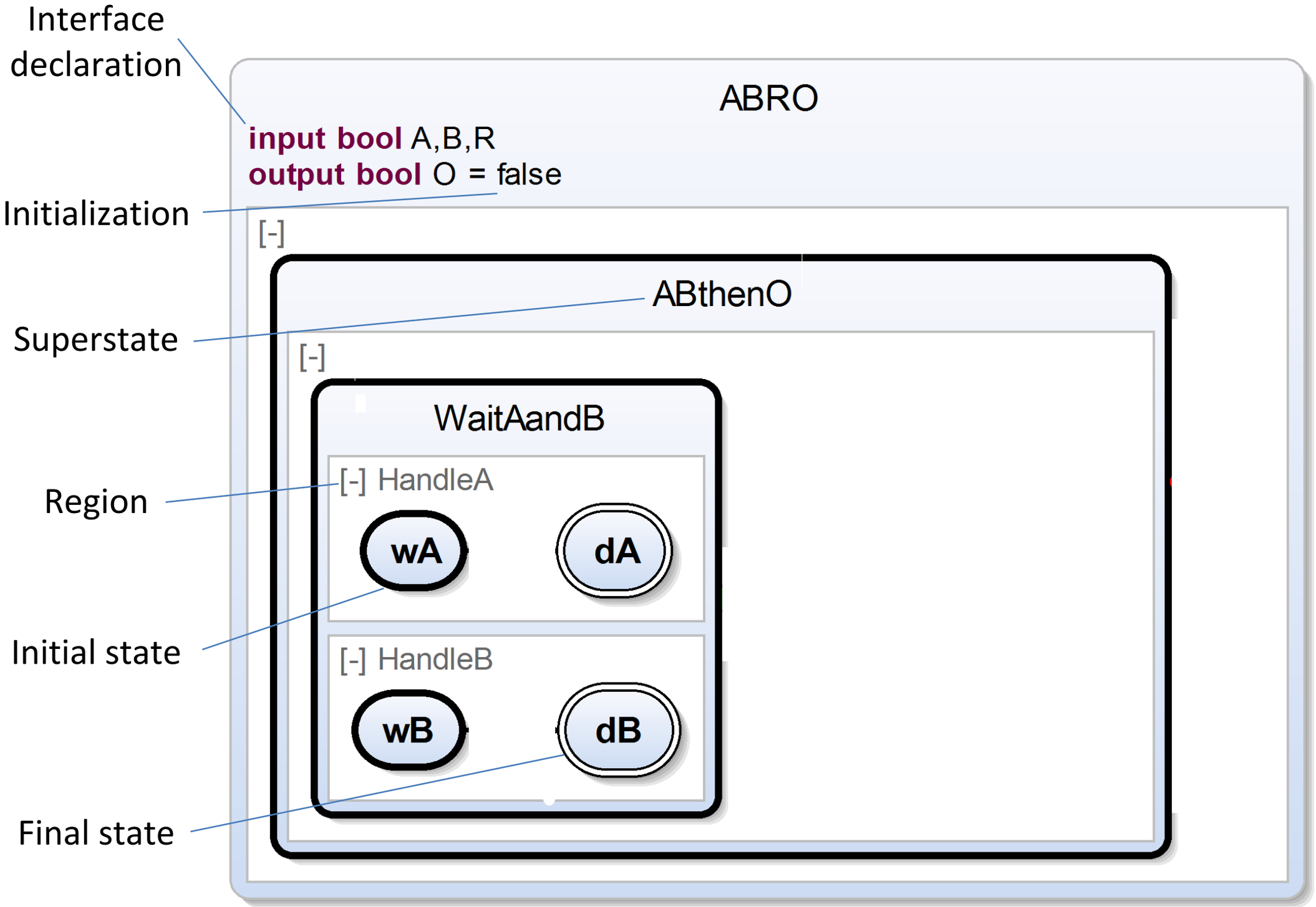
Initialization

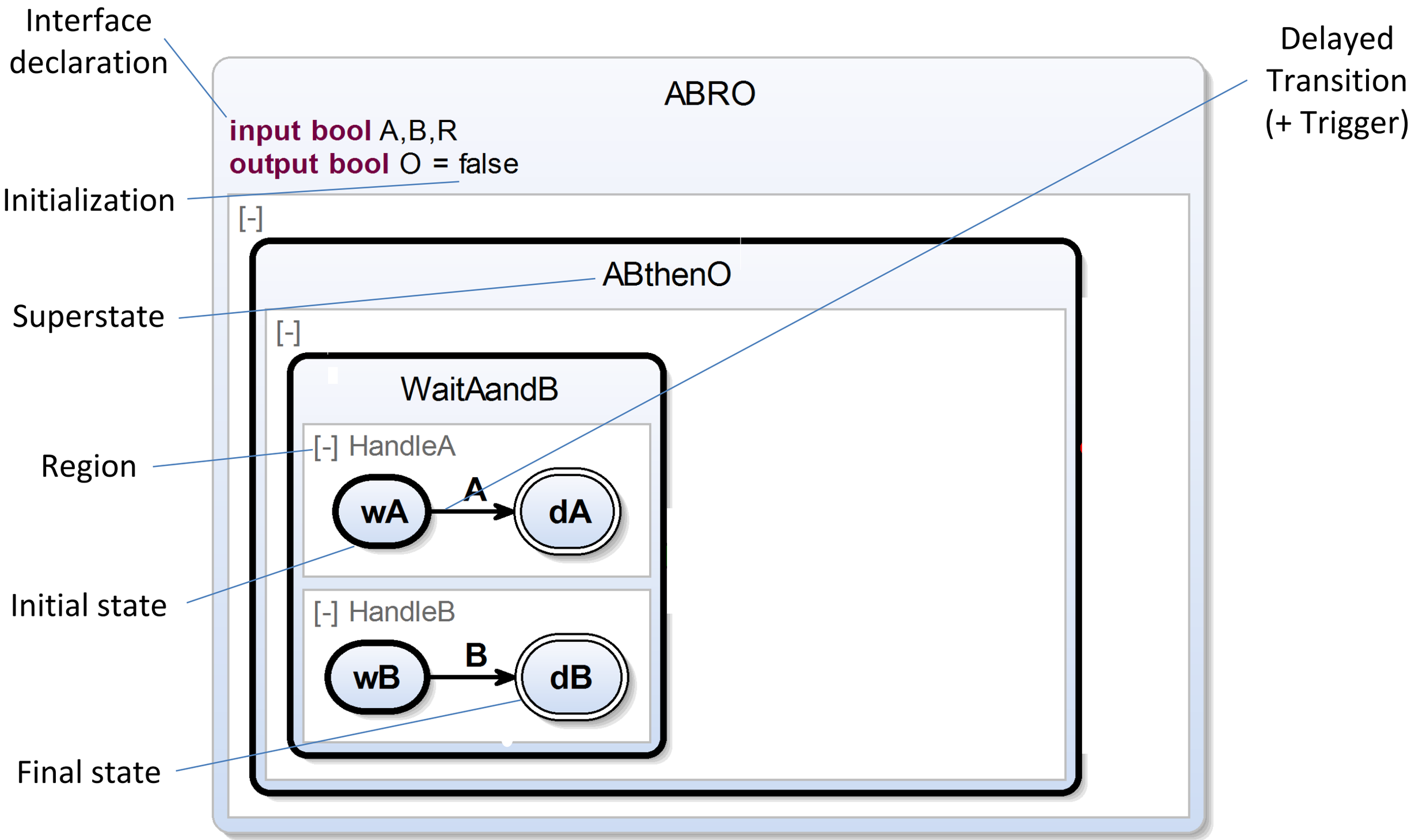
Superstate

Region




Interface declaration
Initialization
Superstate
Region
Initial state
ABRO
input bool A,B,R
output bool O = false
[-]
ABthenO
[-]
WaitAandB
[-] HandleA
wA
[-] HandleB
wB


Interface declaration
Initialization
Superstate
Region
Initial state
Final state
ABRO
input bool A,B,R
output bool O = false
[-]
ABthenO
[-]
WaitAandB
[-] HandleA
wA
dA
[-] HandleB
wB
dB


Interface declaration
Initialization
Superstate
Region
Initial state
Final state
Delayed Transition (+ Trigger)
ABRO
input bool A,B,R
output bool O = false
[-]
ABthenO
[-]
WaitAandB
[-] HandleA
wA
A
dA
[-] HandleB
wB
B
dB

Interface declaration

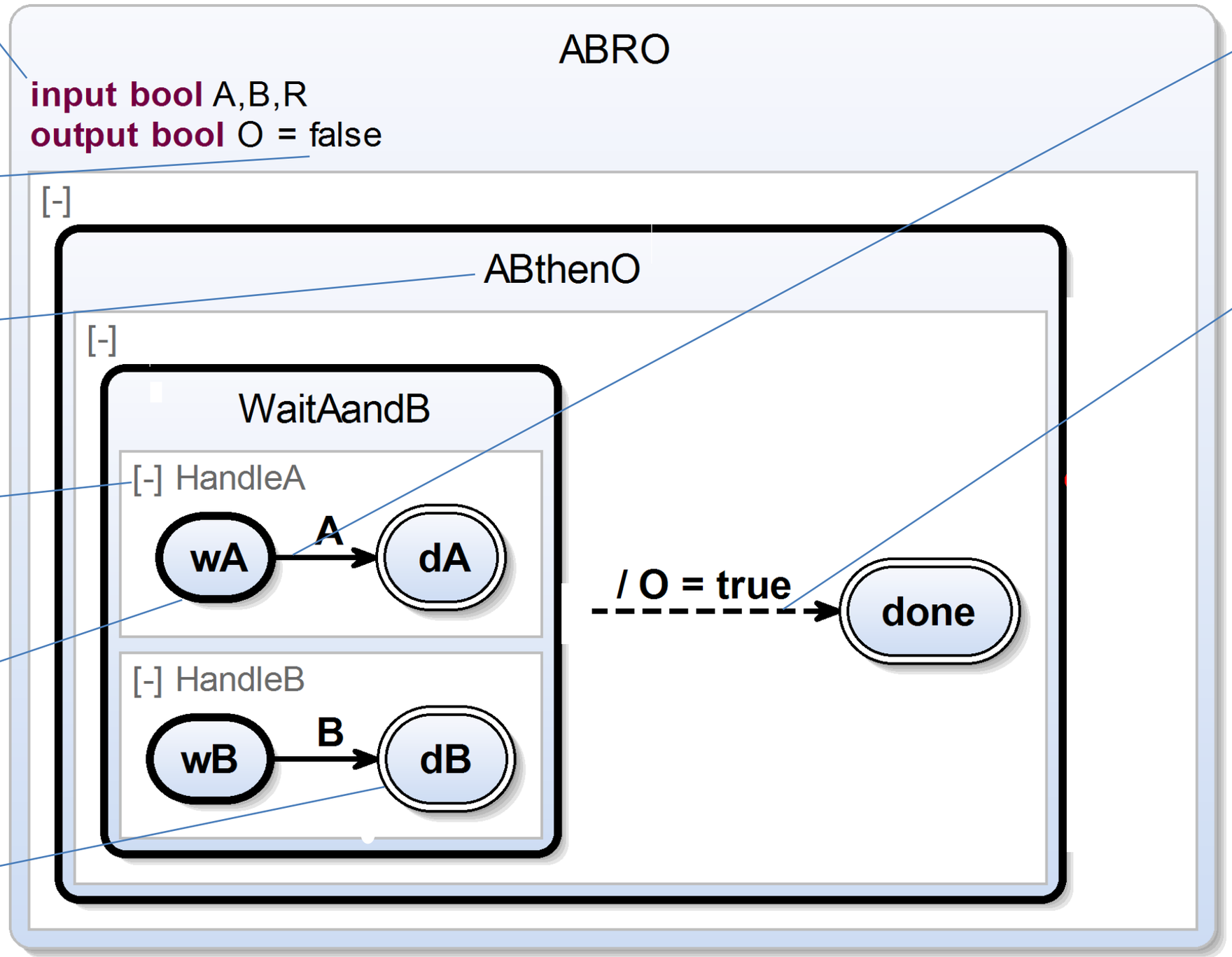
Initialization

Superstate

Region

Initial state

Final state



Delayed Transition (+ Trigger)

Immediate transition (+ Effect)

Interface declaration

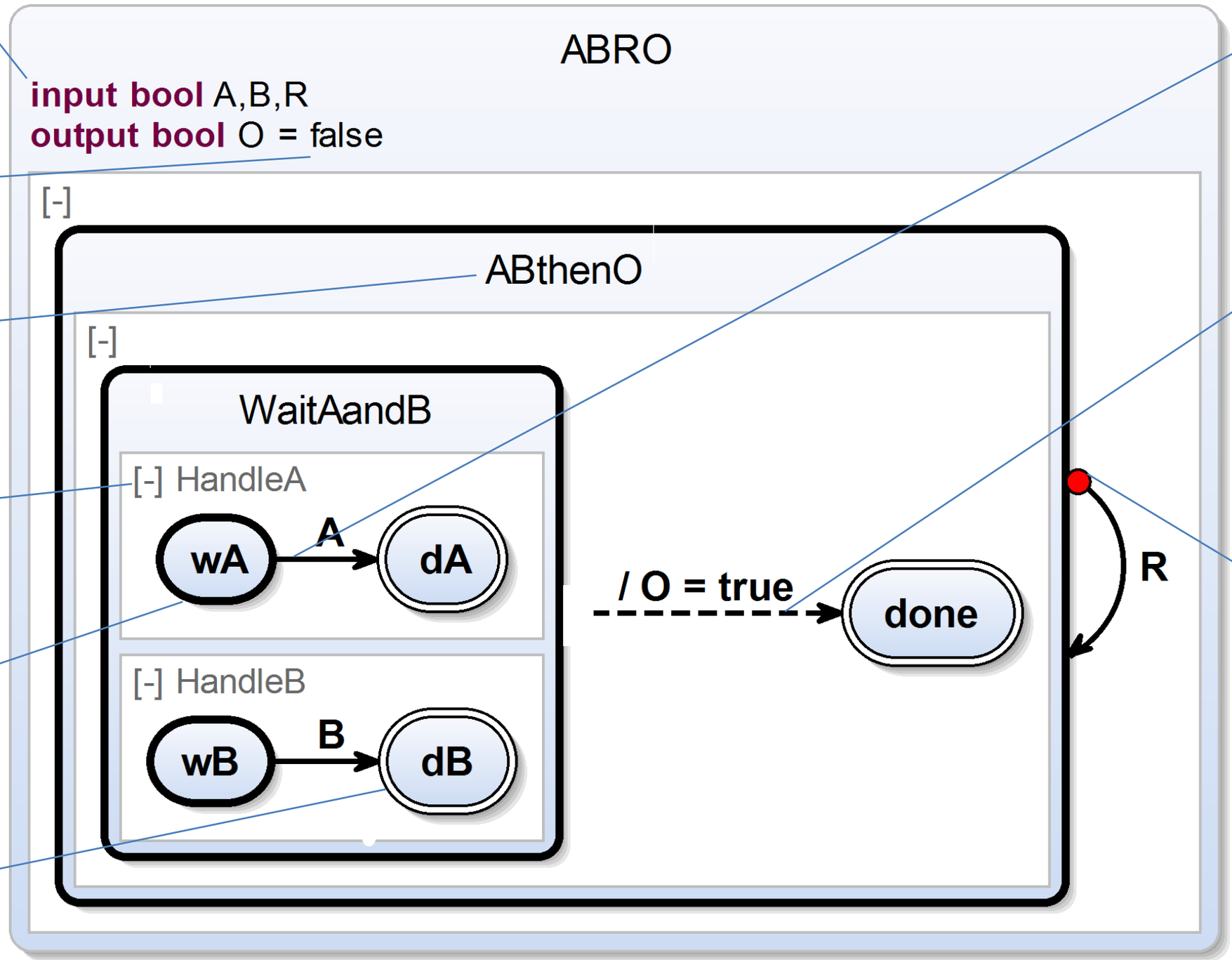
Initialization

Superstate

Region

Initial state

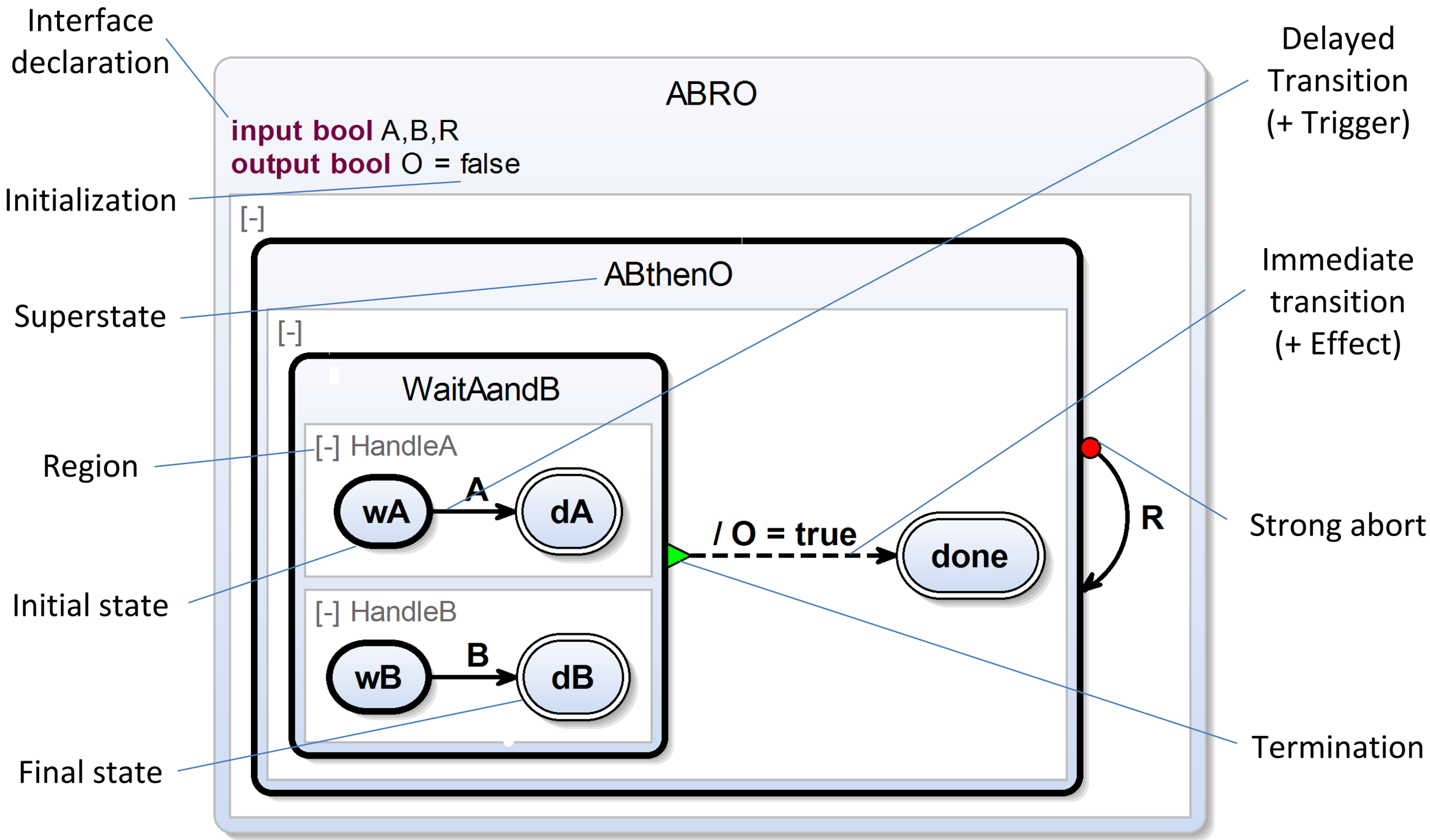
Final state



Delayed Transition (+ Trigger)

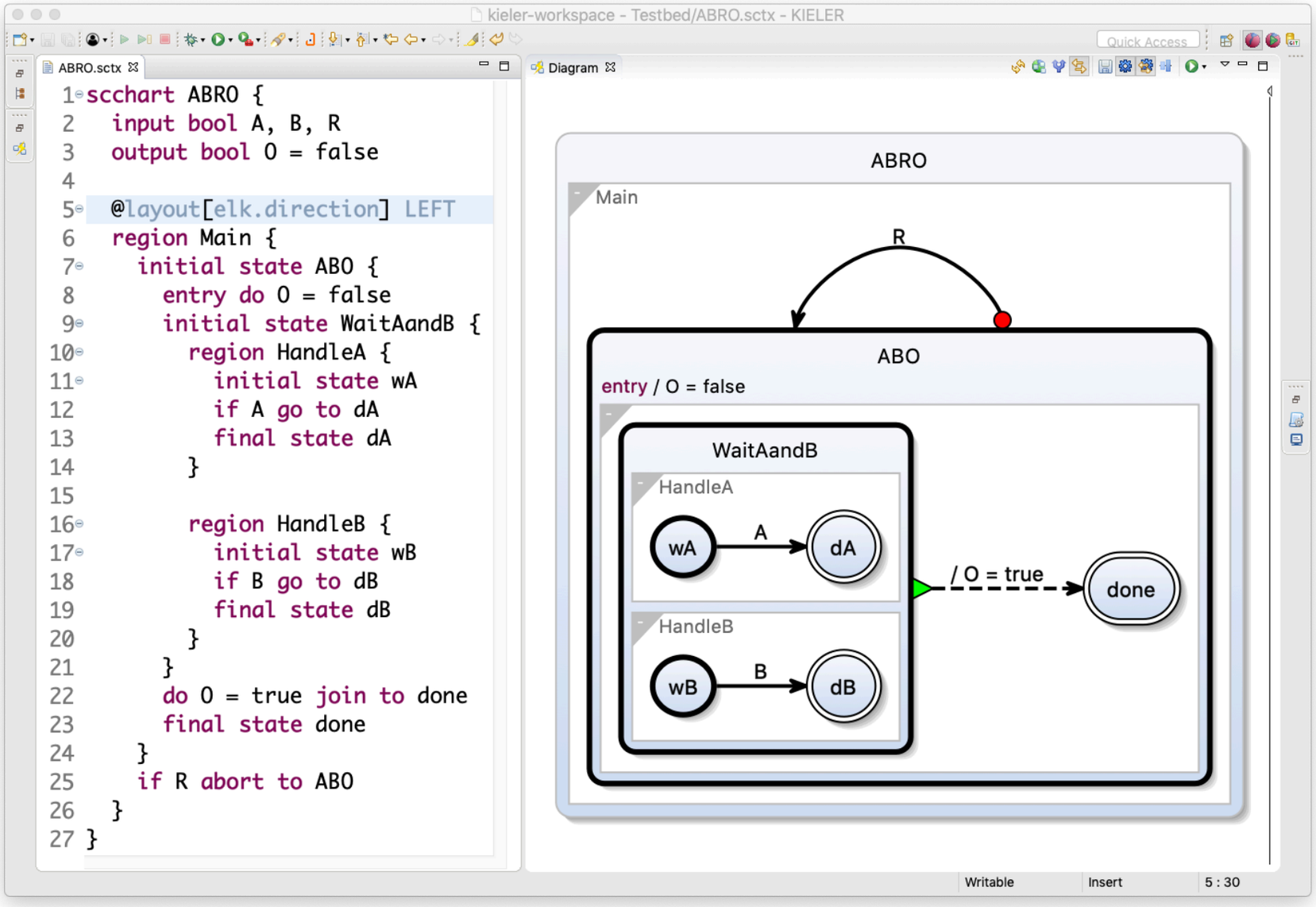
Immediate transition (+ Effect)

Strong abort

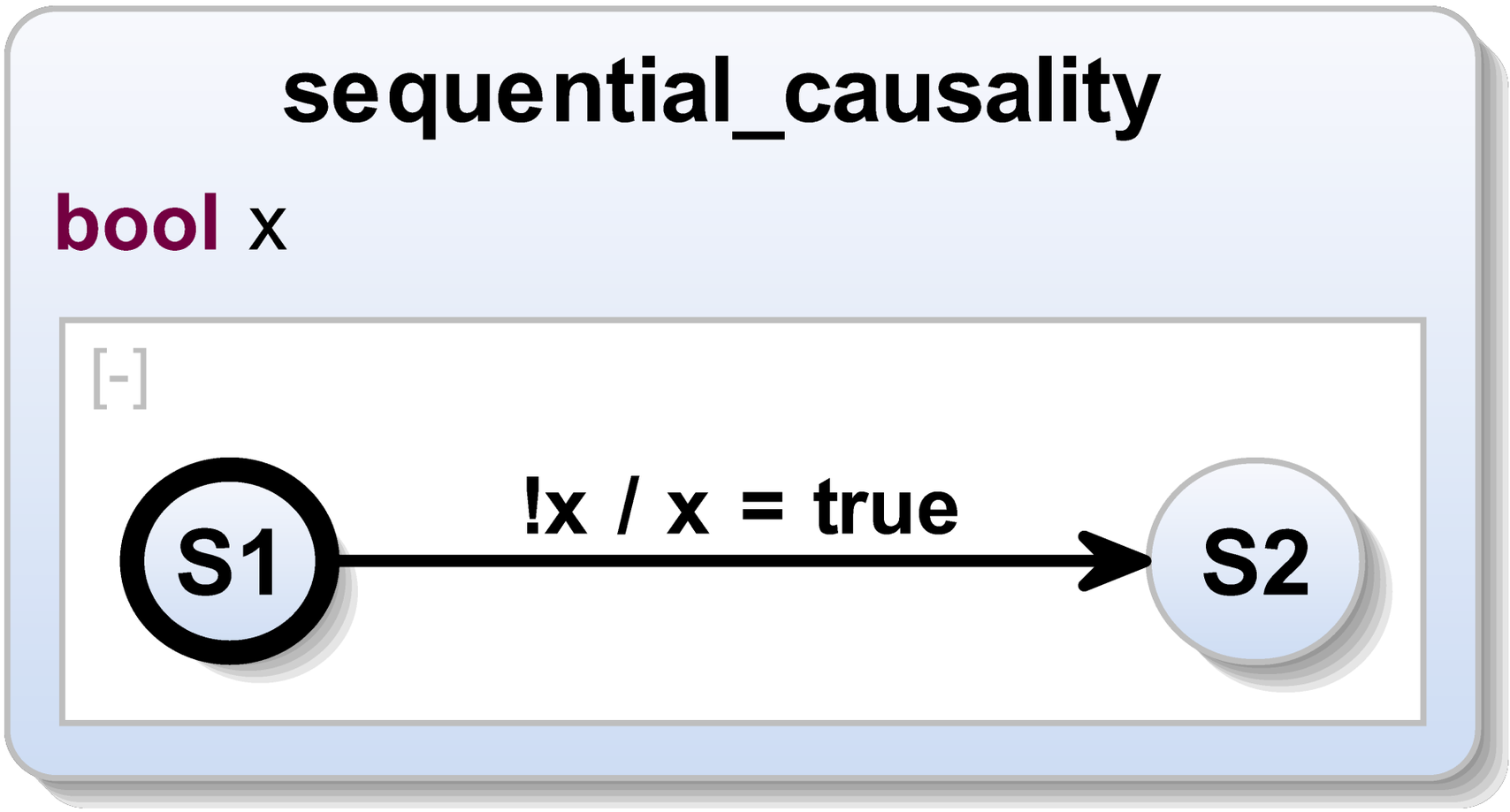

Interface declaration
Initialization
Superstate
Region
Initial state
Final state
Delayed Transition (+ Trigger)
Immediate transition (+ Effect)
Strong abort
Termination
ABRO
input bool A,B,R
output bool O = false
[-]
ABthenO
[-]
WaitAandB
[-] HandleA
wA
A
dA
[-] HandleB
wB
B
dB
/ O = true
done
R

SCCharts Syntax

- SCCharts have textual and graphical syntax
- In KIELER tool, modeler writes textually, tool provides graphical views
- Uses auto-layout from Eclipse Layout Kernel (ELK)



Limitations of Strict Synchrony




```
if (!x) {
   ...
   x = true;
}
```

Not allowed in SyncCharts, Esterel, etc.!

Sequential Constructiveness

Idea: Sequential control flow overrides „write before read“

Writes visible **only** to reads that are

1. sequential successors or
2. concurrent

v. Hanxleden, Mendler, et al.
Sequentially Constructive Concurrency—A Conservative Extension of the Synchronous Model of Computation.
ACM TECS ’14

IUR-Protocol

IUR protocol schedules variable accesses within reaction

- Sequential accesses to `x`: unconstrained
- Concurrent accesses to `x`: must follow IUR protocol

init ⇒ **updates** ⇒ **reads**

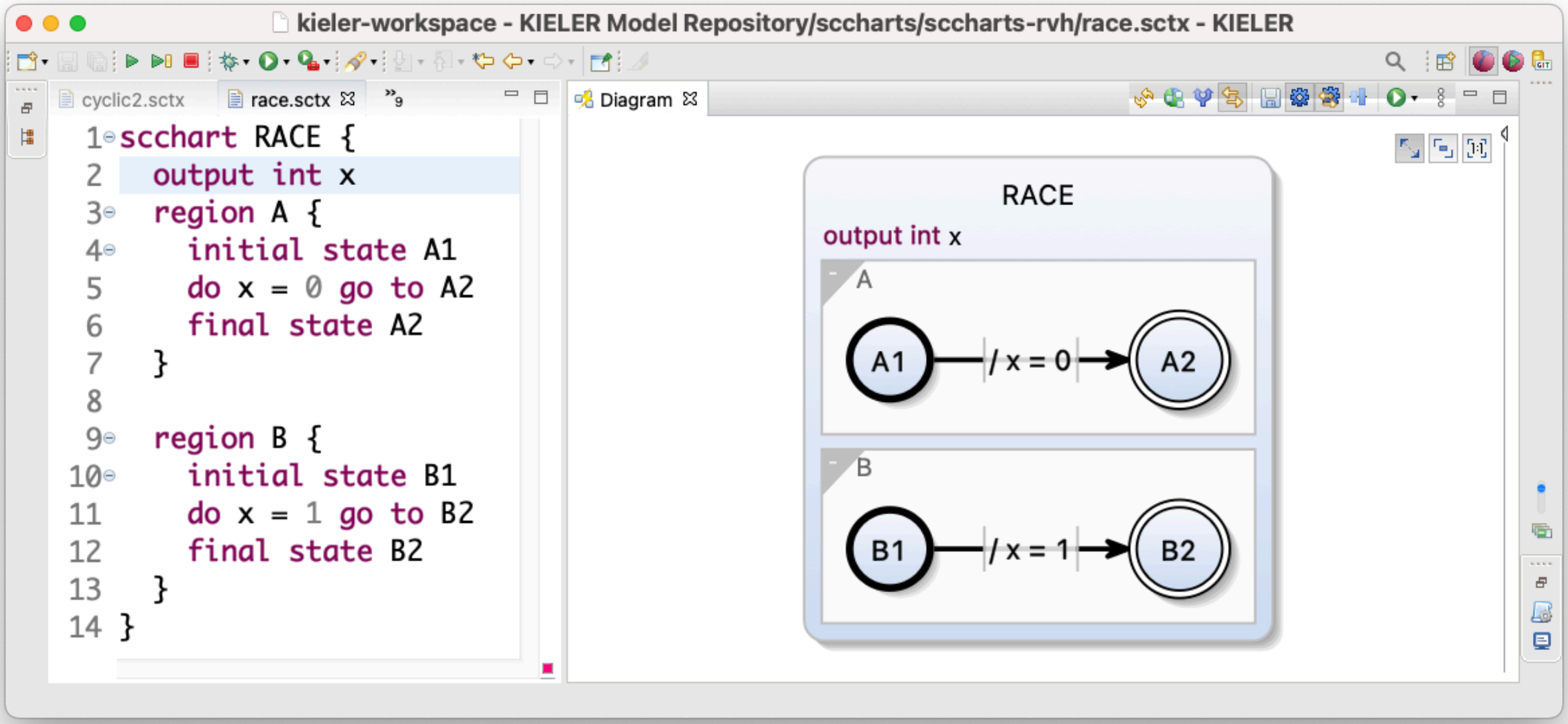
```
x = 1 …  x += 2 … x += 5 …  y = x … z = x
```

I.e., in each tick, for each variable, for each set of concurrent accesses to x, allow the following schedule:

1. An arbitrary number of init's that are confluent (i.e., write the same value)
2. An arbitrary numer of updates that are confluent (i.e., use same combination function)
3. An arbitrary number of reads

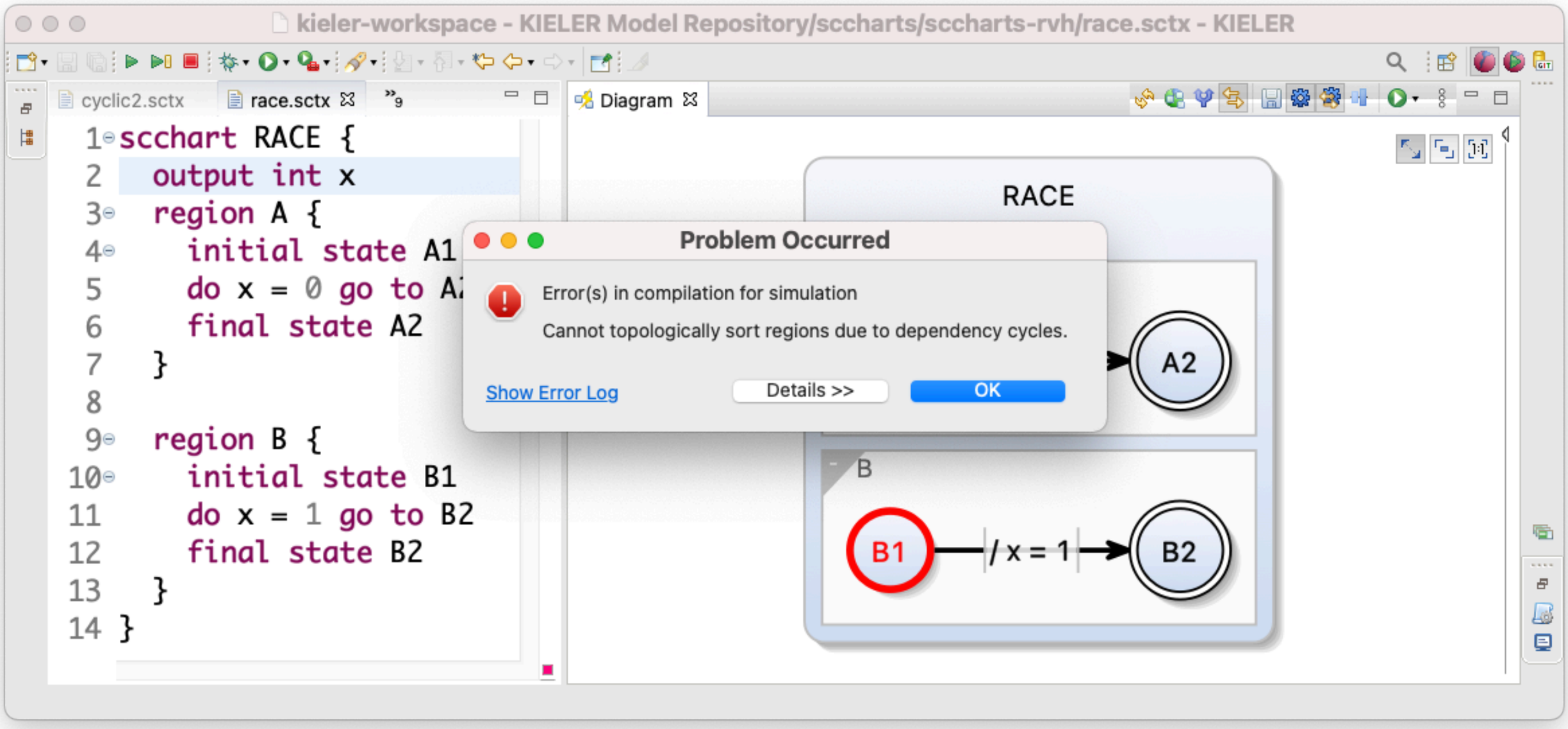
Must Still Reject Some Models ...

Concurrent accesses may lead to causality cycles



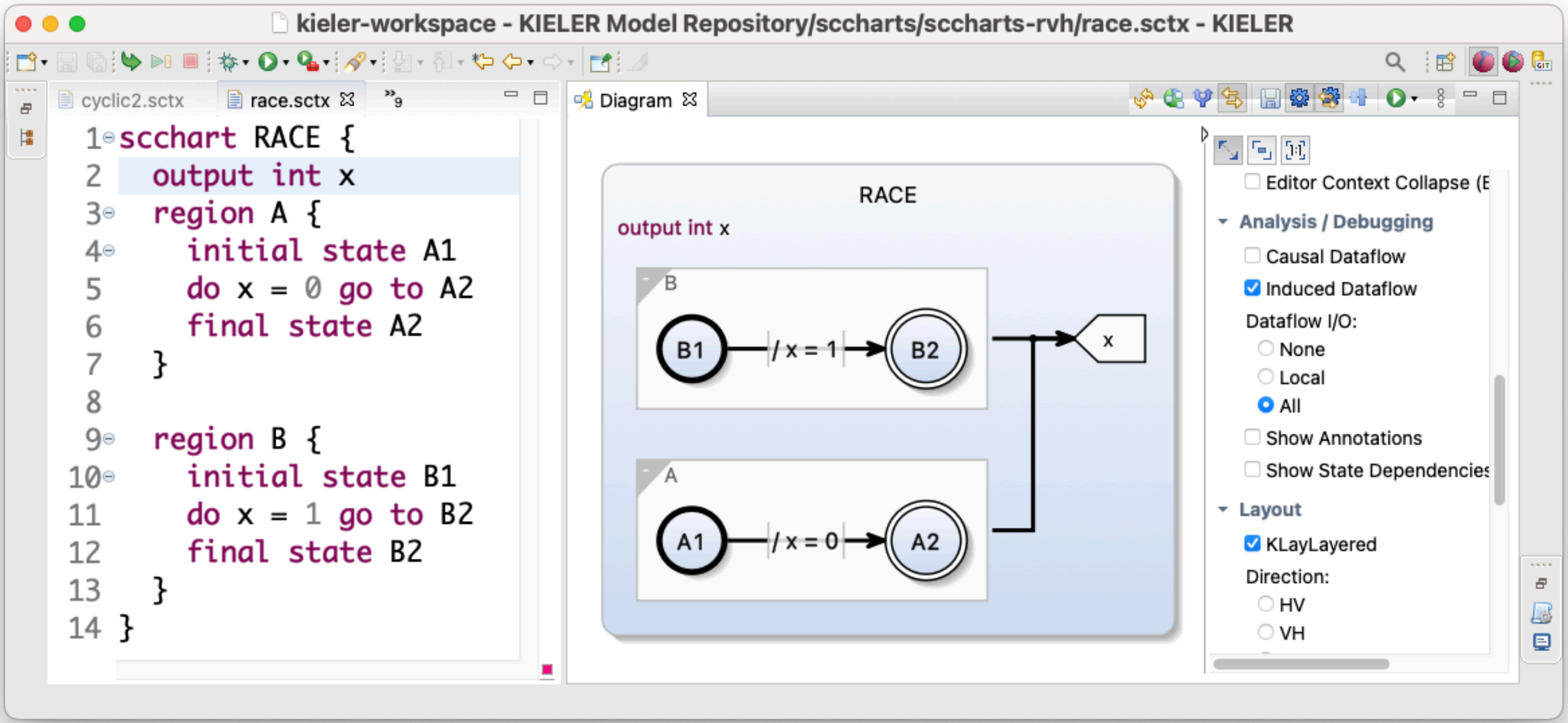
Must Still Reject Some Models ...

Concurrent accesses may lead to causality cycles

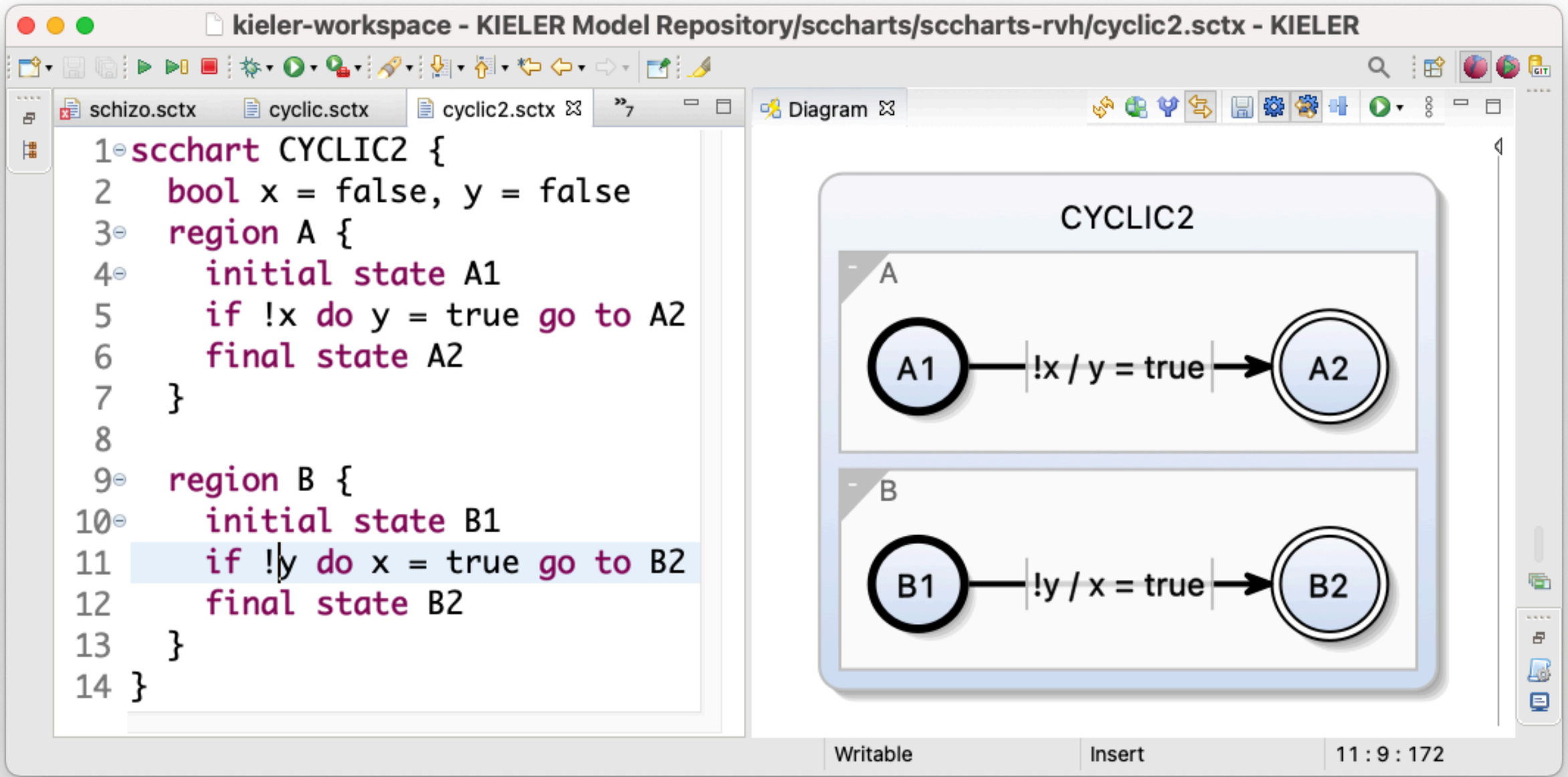


Must Still Reject Some Models …

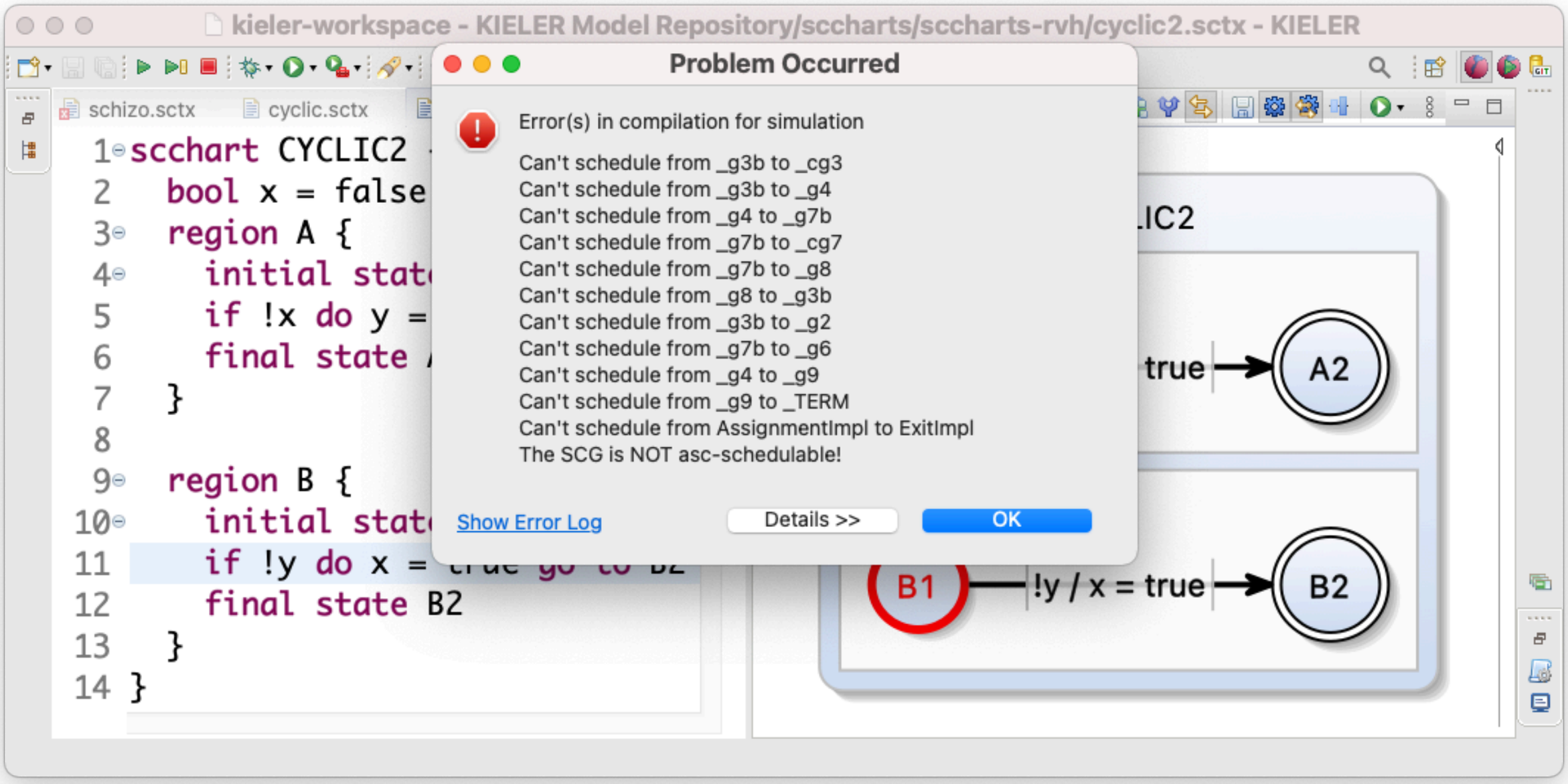
Concurrent accesses may lead to causality cycles



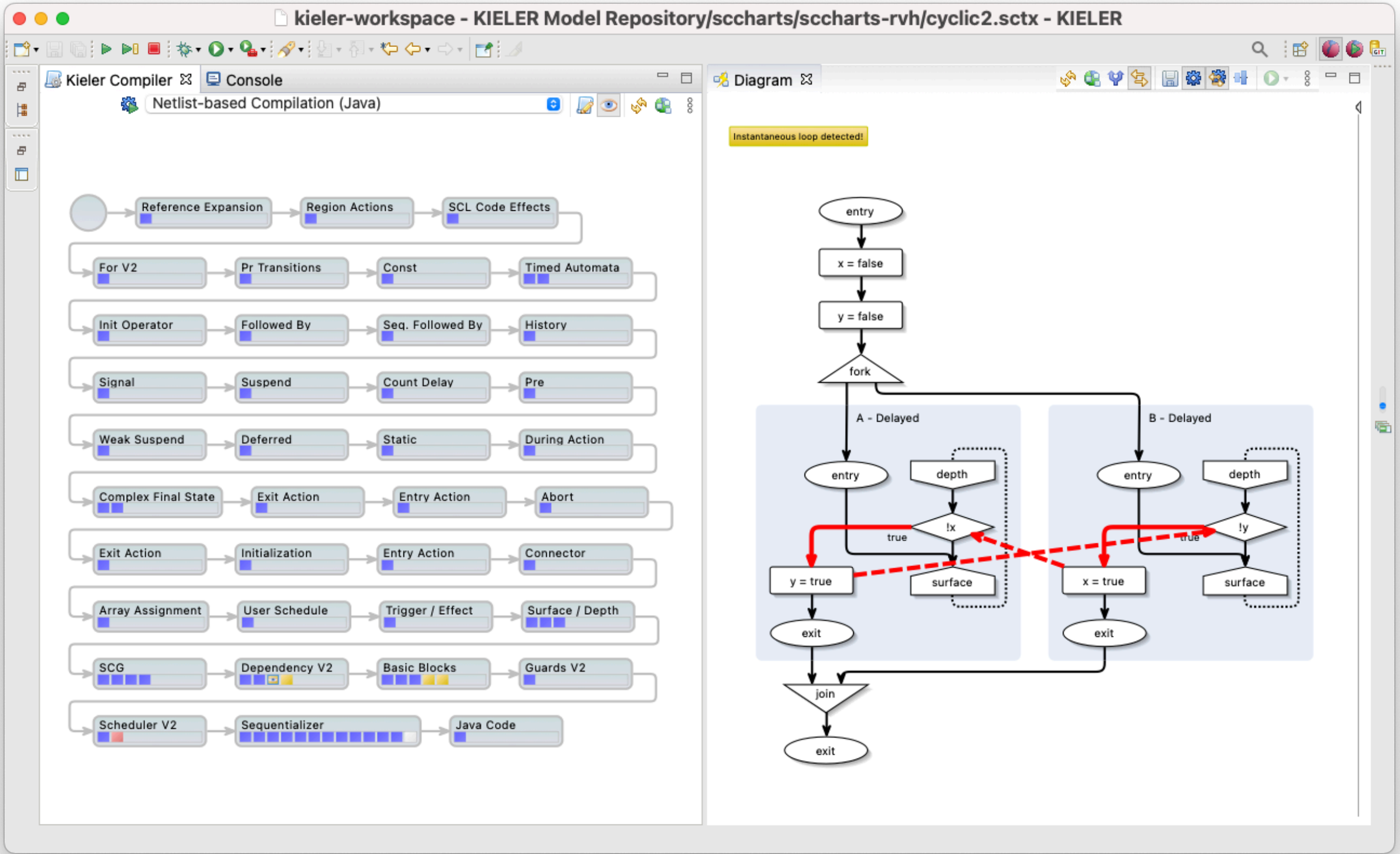
Another Example



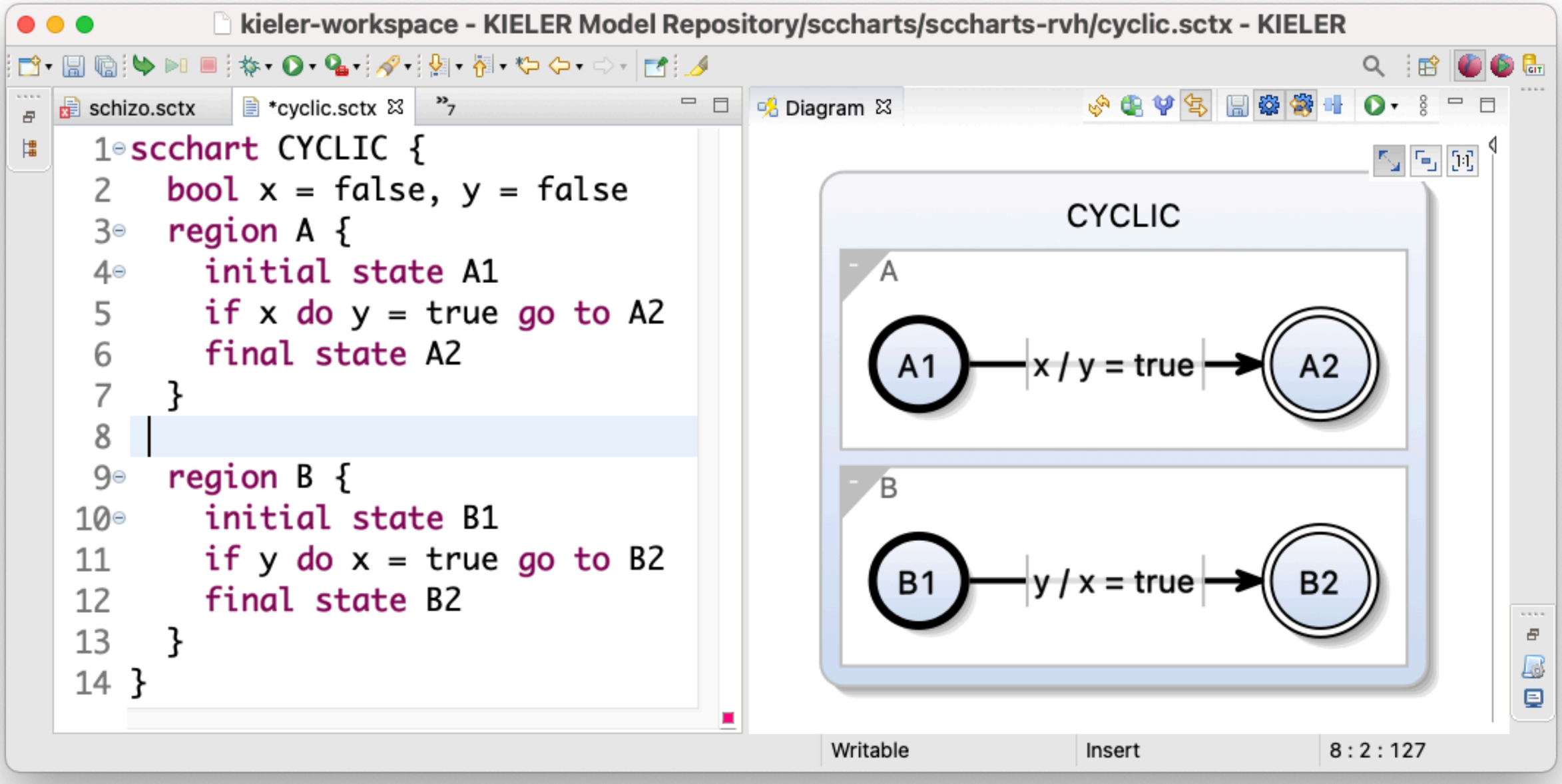
Another Example



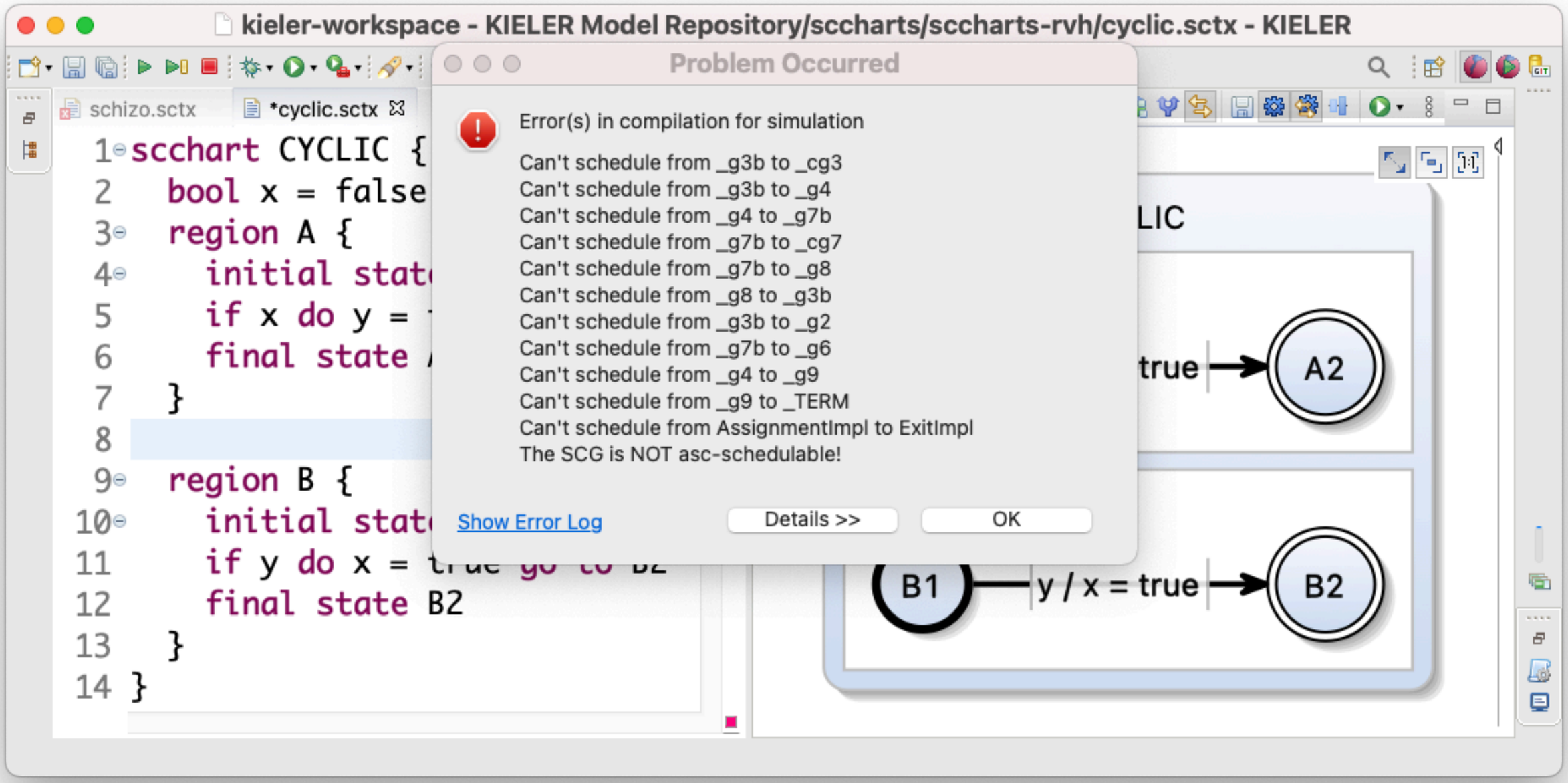
Another Example



Variation on Example

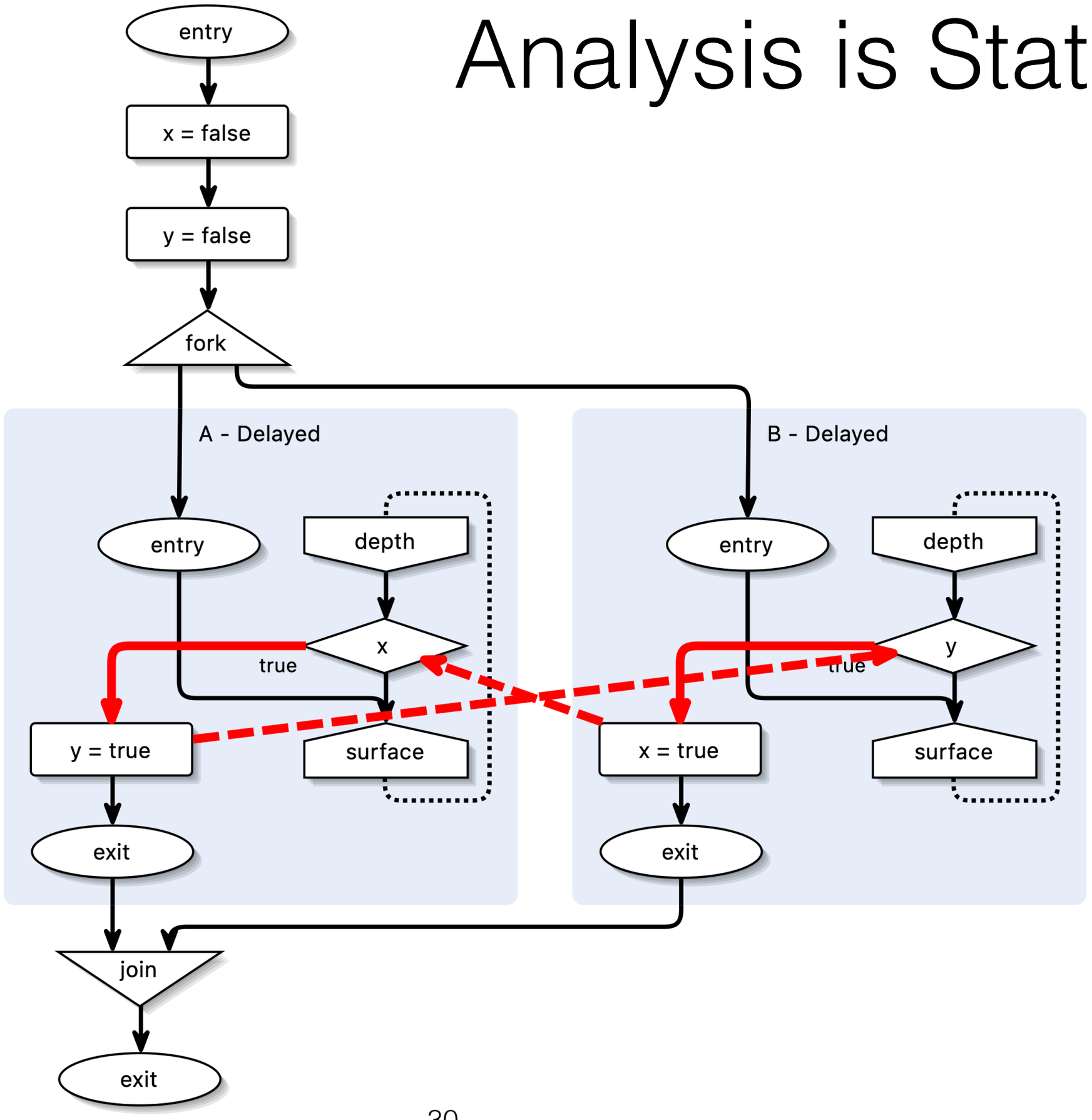


Variation on Example

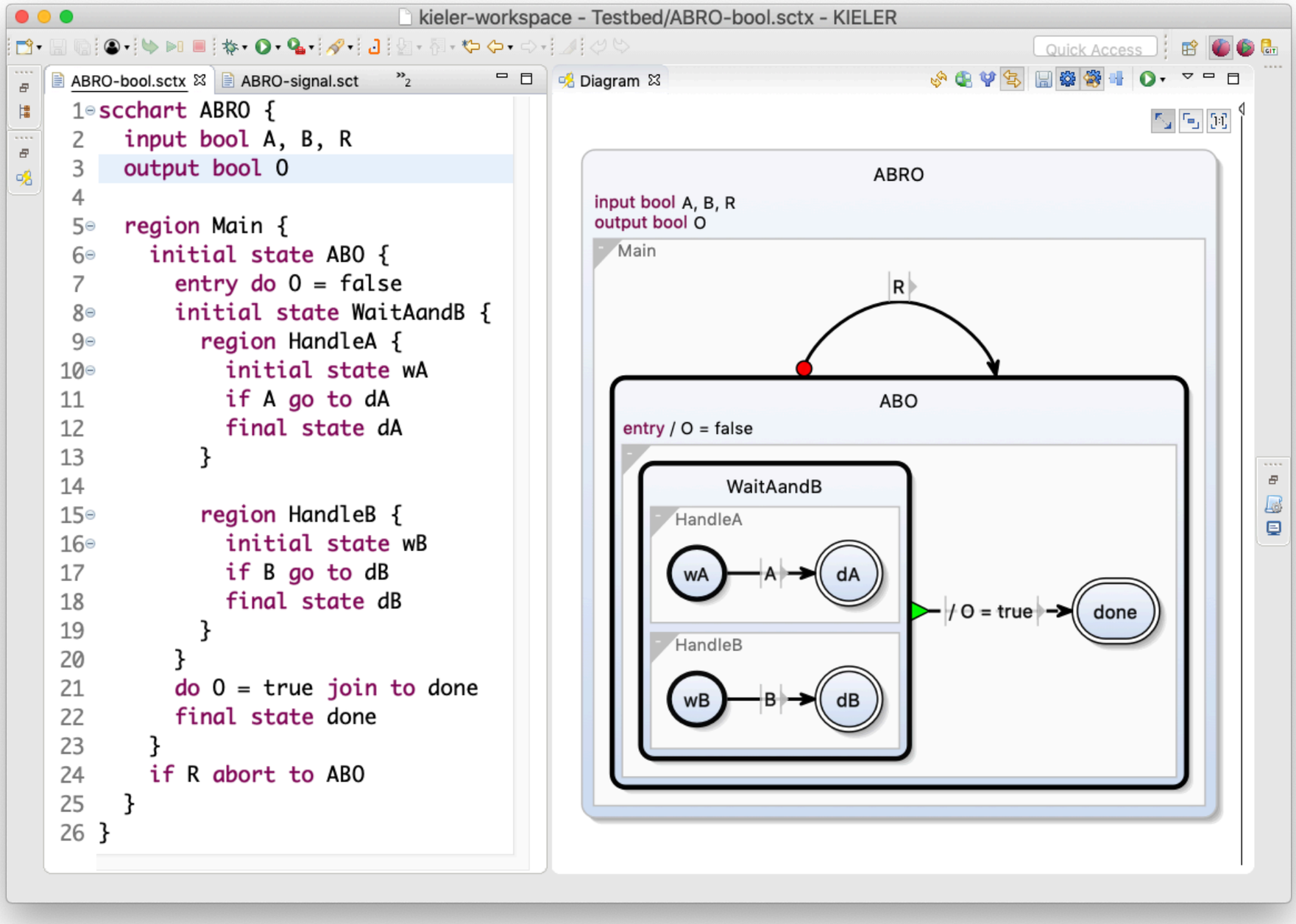


Instantaneous loop detected!

Analysis is Static!



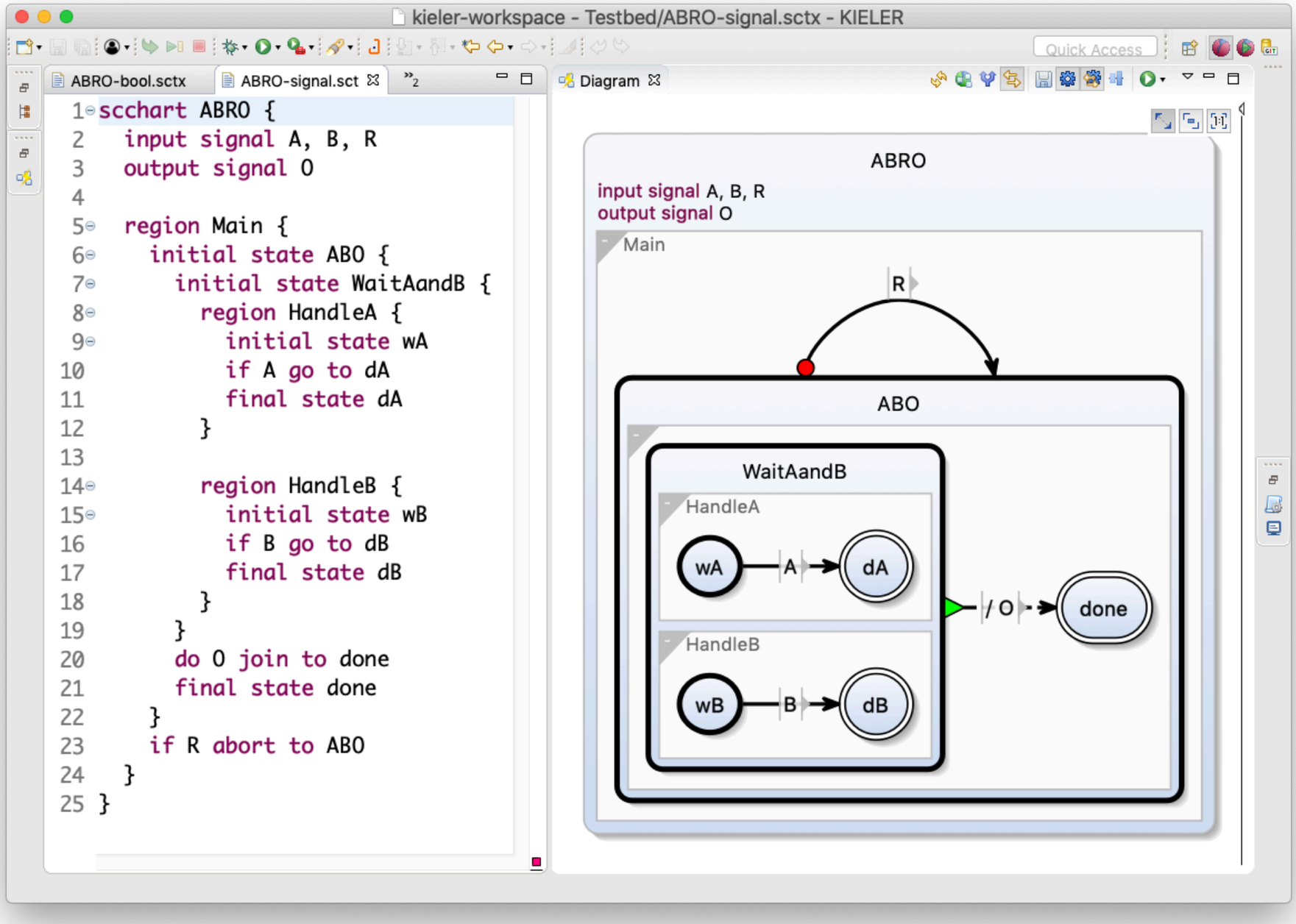
Booleans vs. Signals



`bool`:

- true or false
- Persistent across ticks

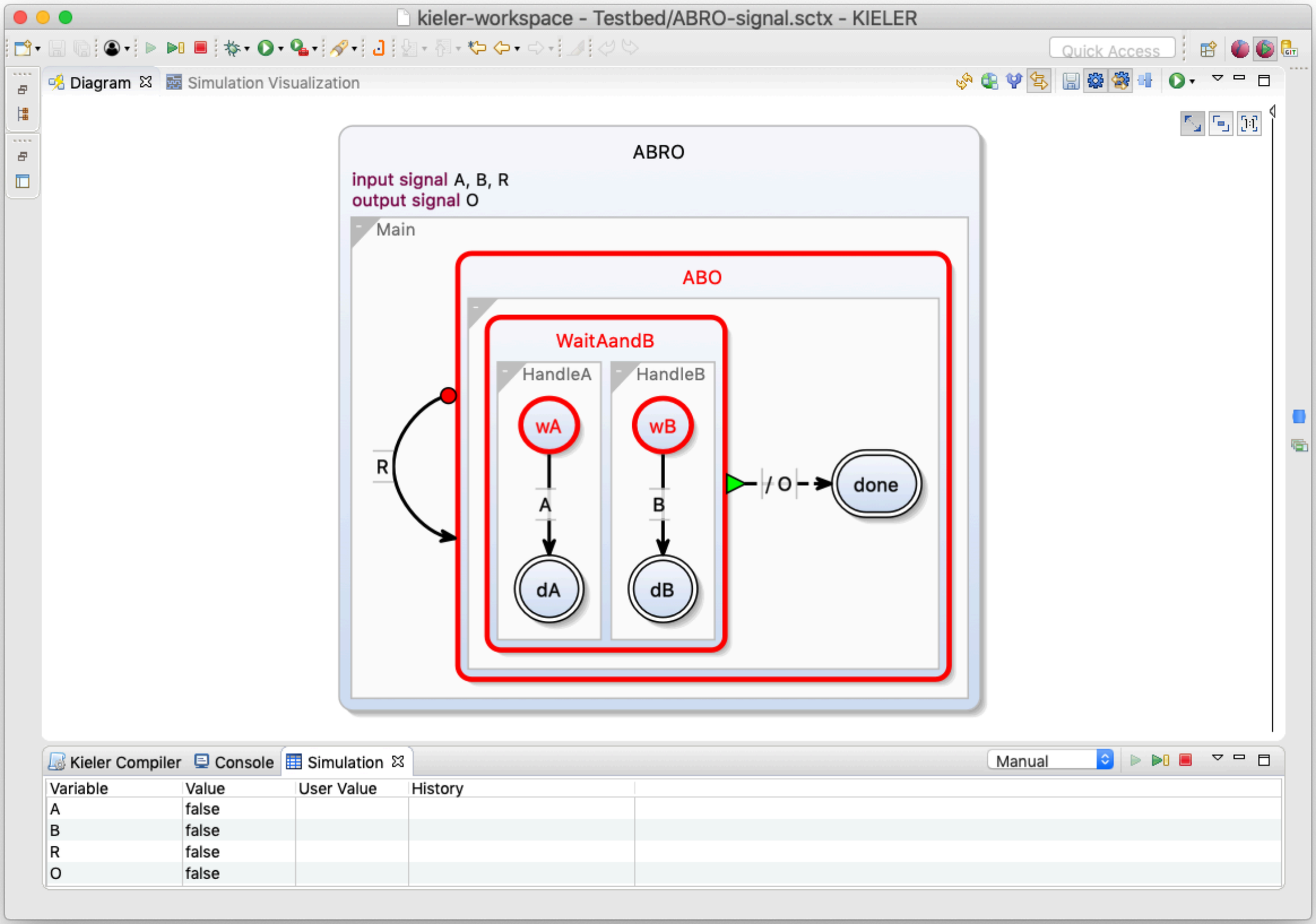
Booleans vs. Signals



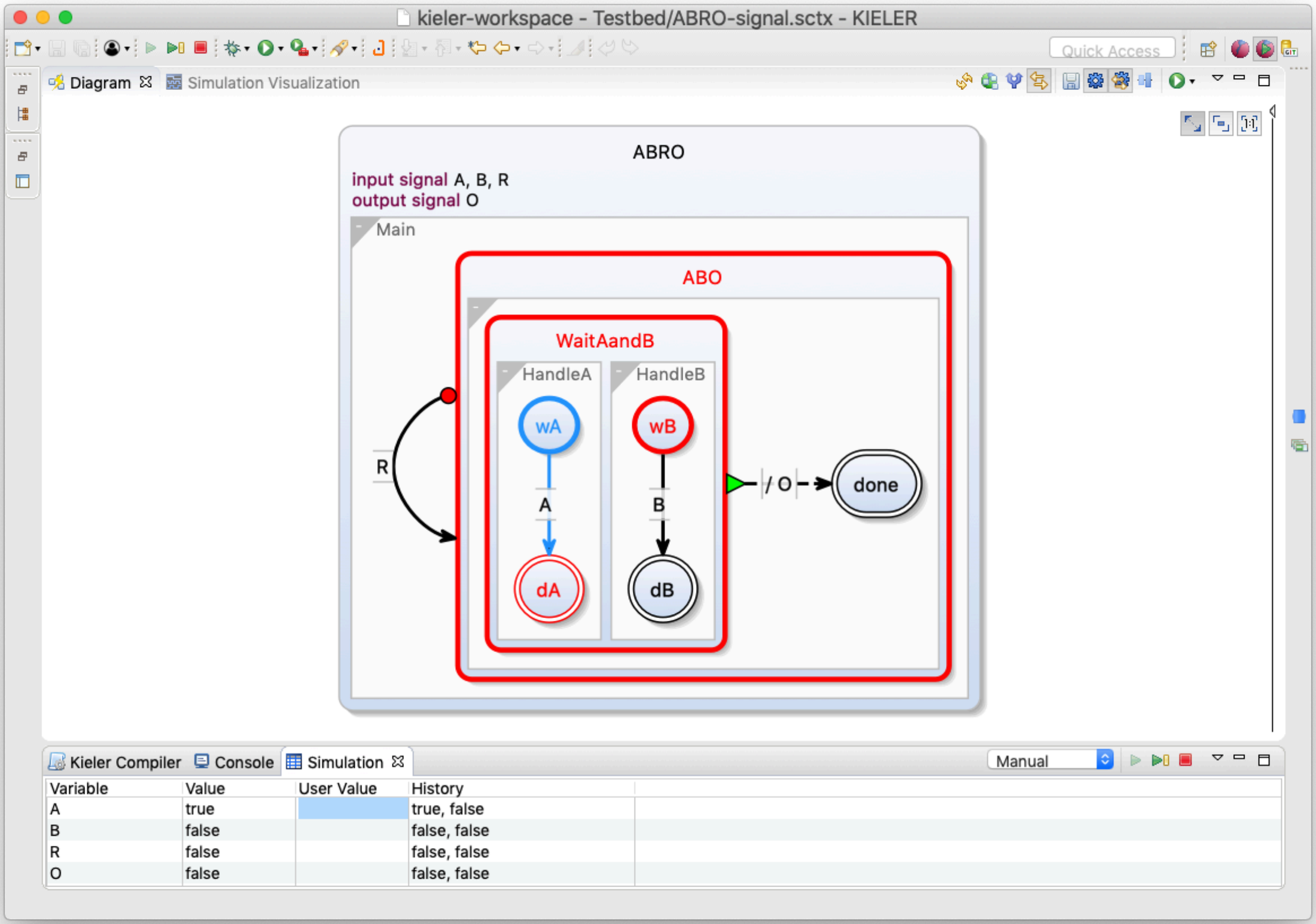
`signal`:

- true (“present”) or false (“absent”)
- Re-initialized to absent (unless input signal) at each tick
- Conceptually, correspond to *events*
- ... and beyond these *pure signals*, there are also *valued signals*, which carry a – persistent – value of some type (including bool) ...

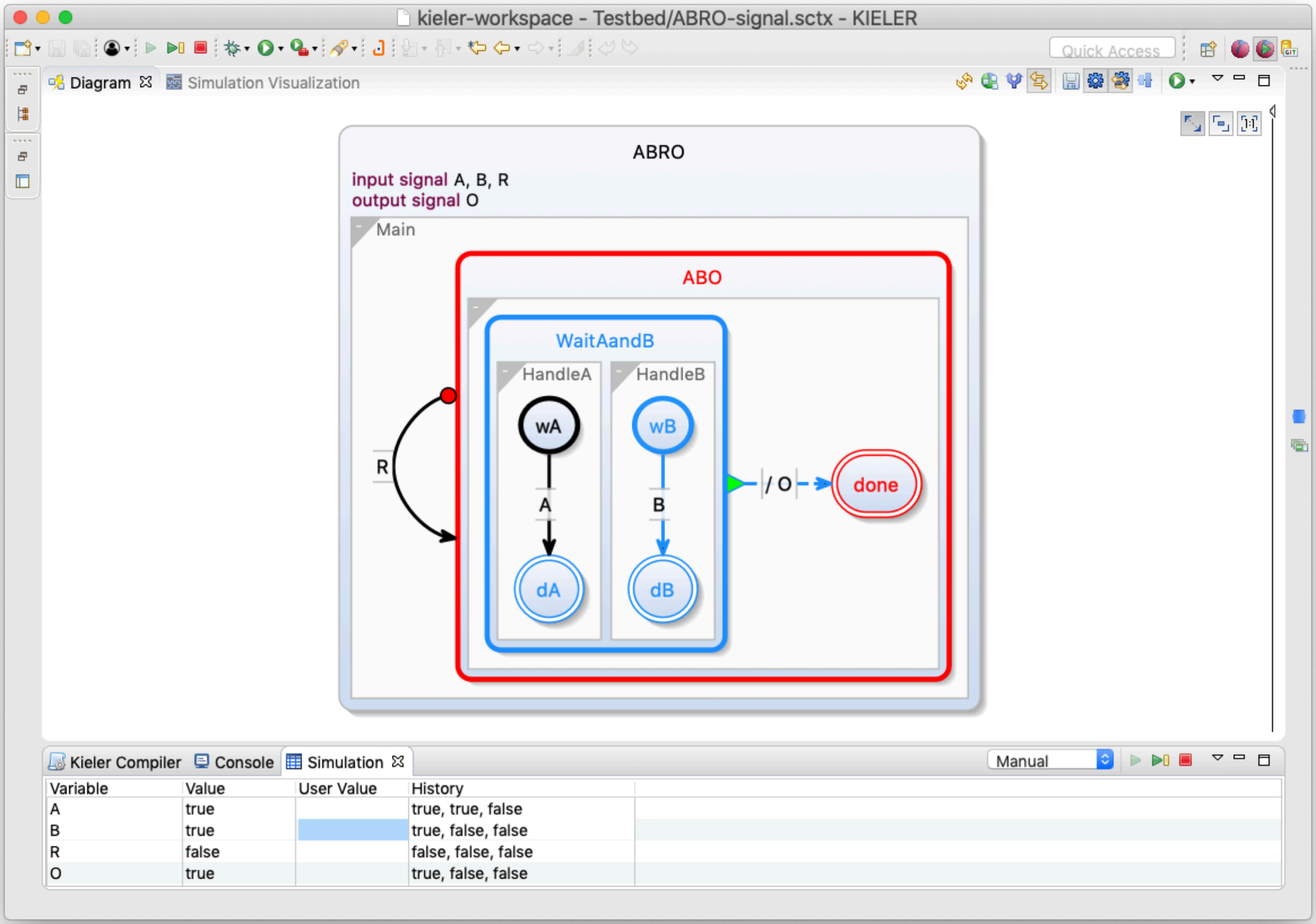
Simulation


kieler-workspace - Testbed/ABRO-signal.sctx - KIELER
Quick Access
Diagram
Simulation Visualization
ABRO
input signal A, B, R
output signal O
Main
ABO
WaitAandB
HandleA
HandleB
wA
wB
R
A
B
dA
dB
/ O
done
Kieler Compiler
Console
Simulation
Manual
Variable
Value
User Value
History
A
false
B
false
R
false
O
false

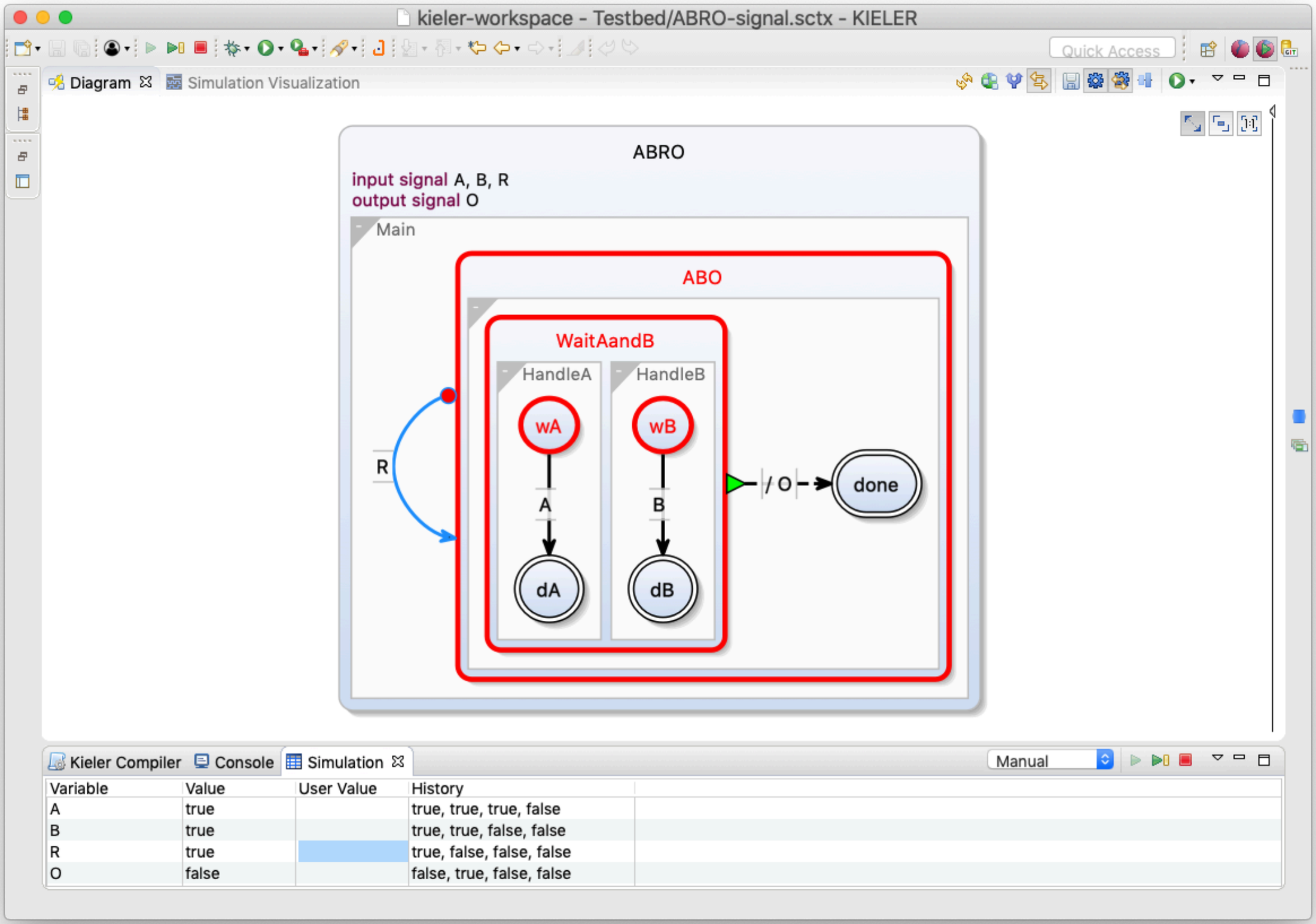
Simulation


kieler-workspace - Testbed/ABRO-signal.sctx - KIELER
Quick Access
Diagram
Simulation Visualization
ABRO
input signal A, B, R
output signal O
Main
ABO
WaitAandB
HandleA
HandleB
wA
wB
A
B
dA
dB
R
/ O
done
Kieler Compiler
Console
Simulation
Manual
Variable
Value
User Value
History
A
true
true, false
B
false
false, false
R
false
false, false
O
false
false, false

Simulation


kieler-workspace - Testbed/ABRO-signal.sctx - KIELER
Quick Access
Diagram
Simulation Visualization
ABRO
input signal A, B, R
output signal O
Main
ABO
WaitAandB
HandleA
HandleB
wA
wB
R
A
B
/ O
done
dA
dB
Kieler Compiler
Console
Simulation
Manual
Variable
Value
User Value
History
A
true
true, true, false
B
true
true, false, false
R
false
false, false, false
O
true
true, false, false

Simulation

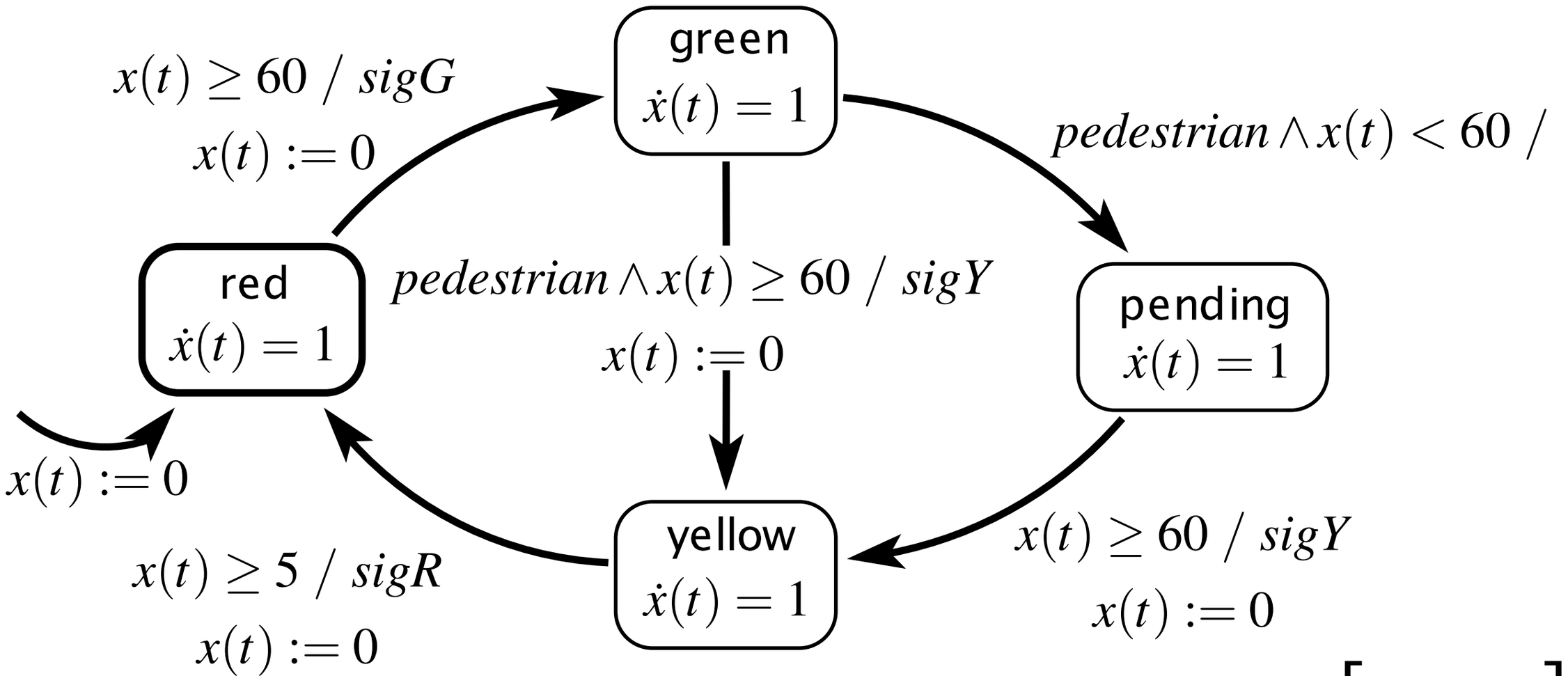

kieler-workspace - Testbed/ABRO-signal.sctx - KIELER
Quick Access
Diagram
Simulation Visualization
ABRO
input signal A, B, R
output signal O
Main
ABO
WaitAandB
HandleA
HandleB
wA
wB
R
A
B
dA
dB
/ O
done
Kieler Compiler
Console
Simulation
Manual
Variable
Value
User Value
History
A
true
true, true, true, false
B
true
true, true, false, false
R
true
true, false, false, false
O
false
false, true, false, false

More Language Features

- The semantic kernel of SCCharts has been stable since beginning
- Beyond kernel, have host language interaction, for regions, reference charts/inheritance, dataflow, ...
- Like other languages (e.g., Java), the feature set of SCCharts keeps evolving – also based on your input!
- When using SCCharts, should consult current documentation and experiment with different features to see what might suit you best
- See https://rtsys.informatik.uni-kiel.de/confluence/display/KIELER/Syntax

Traffic Light as Timed Automaton

continuous variable: $x(t) : \mathbb{R}$
inputs: *pedestrian* : pure
outputs: *sigR*, *sigG*, *sigY* : pure



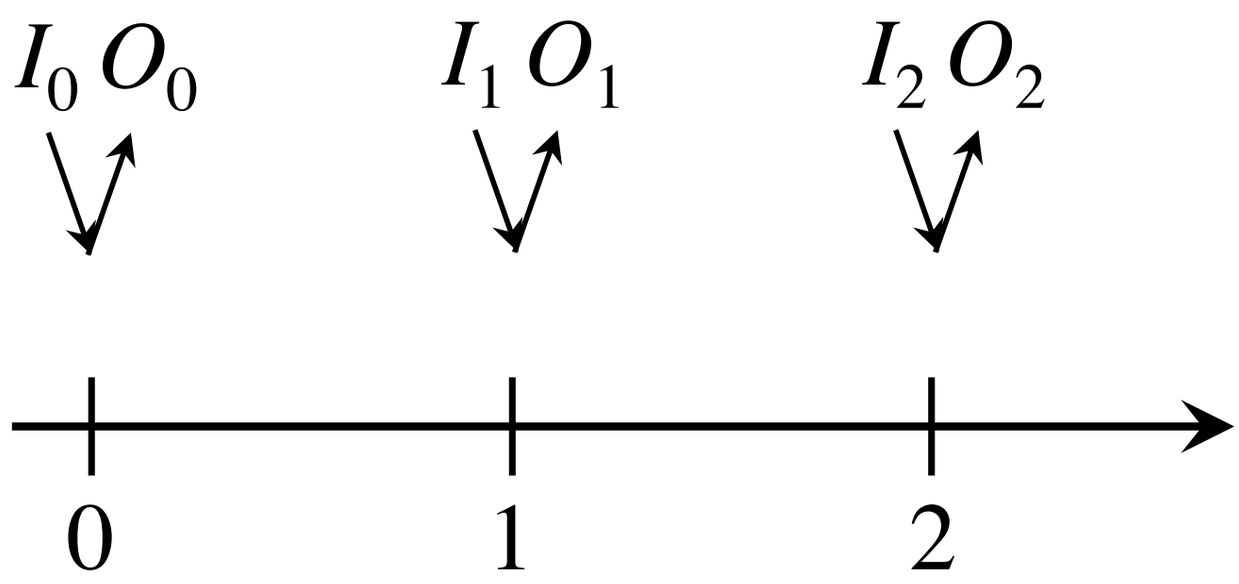
[Lee/Seshia]

Alur, Dill, *A theory of timed automata*, Theoretical Computer Science, 1994

Roadmap

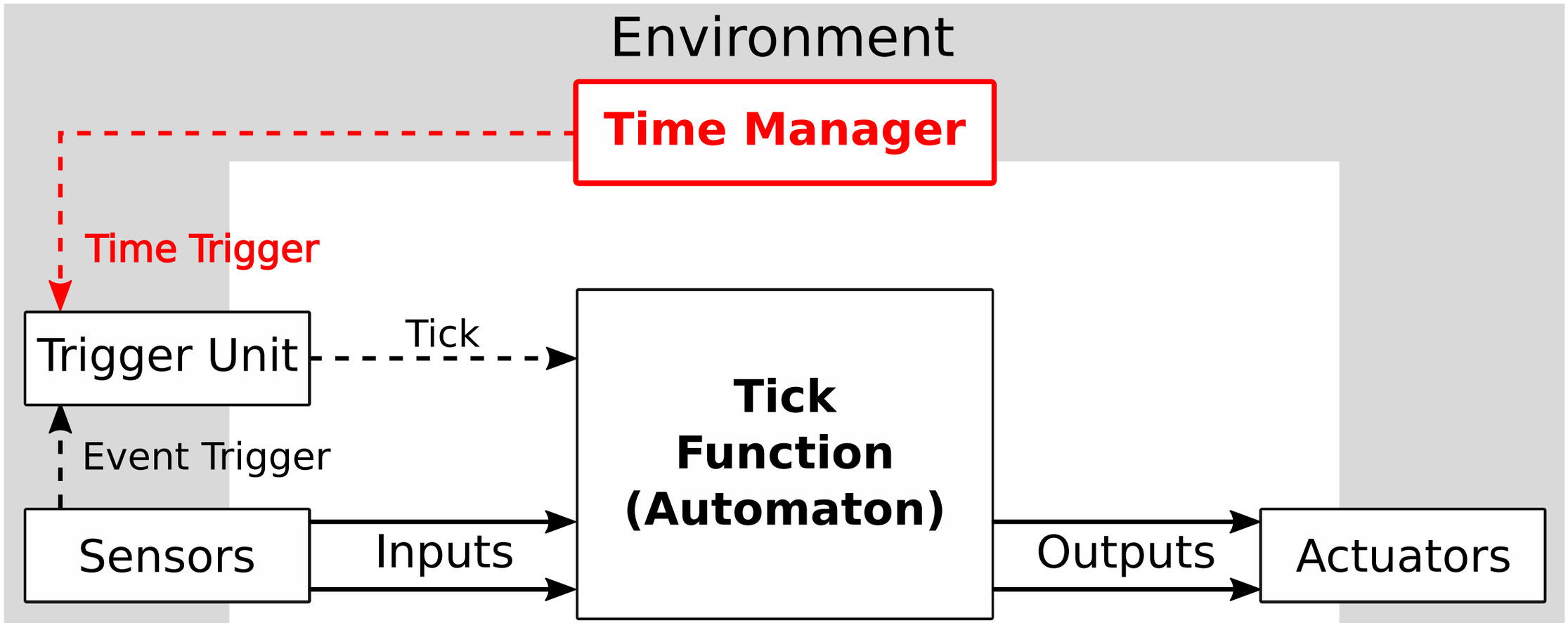
1. Traffic Light Example
2. Execution Models
3. Dynamic Ticks
4. Time in SCCharts: “clock”

Discrete (Logical) Time in Synchronous Programming

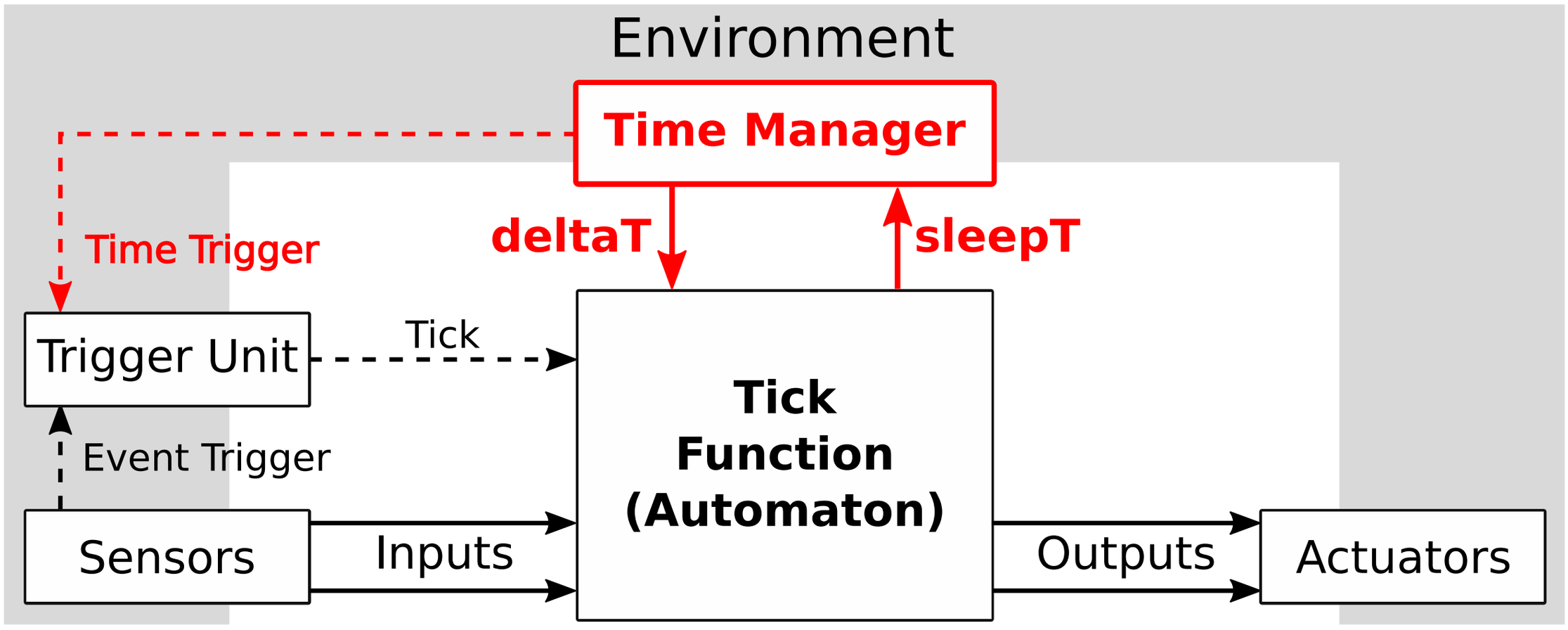


- Synchrony Hypothesis:
  Outputs are synchronous with inputs
- Computation "does not take time"
- Actual computation time does not influence result
- Sequence of outputs **determined** by inputs

Time-Triggered Execution



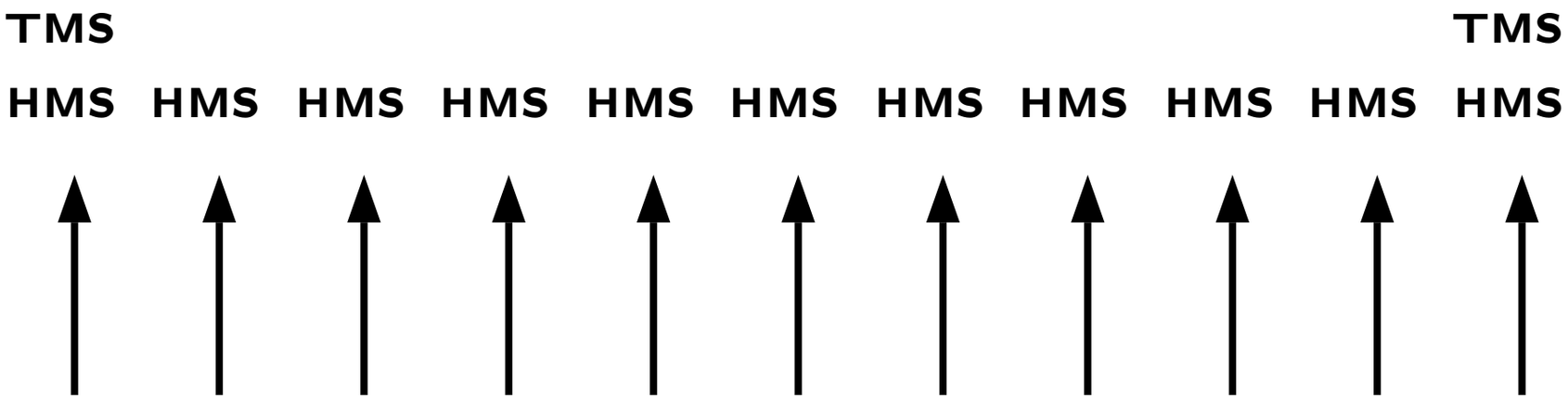
Eager Execution with Dynamic Ticks



deltaT: Time spent in the environment since the last tick

sleepT: Requested delay until next tick i.e. residual time for the reaction

Packaging Physical Time as Events



[Timothy Bourke, SYNCHRON 2009]

Event "HMS": 100 µsec have passed since last HMS

Event "TMS": 1000 µsec have passed since last TMS

A Problem With That …

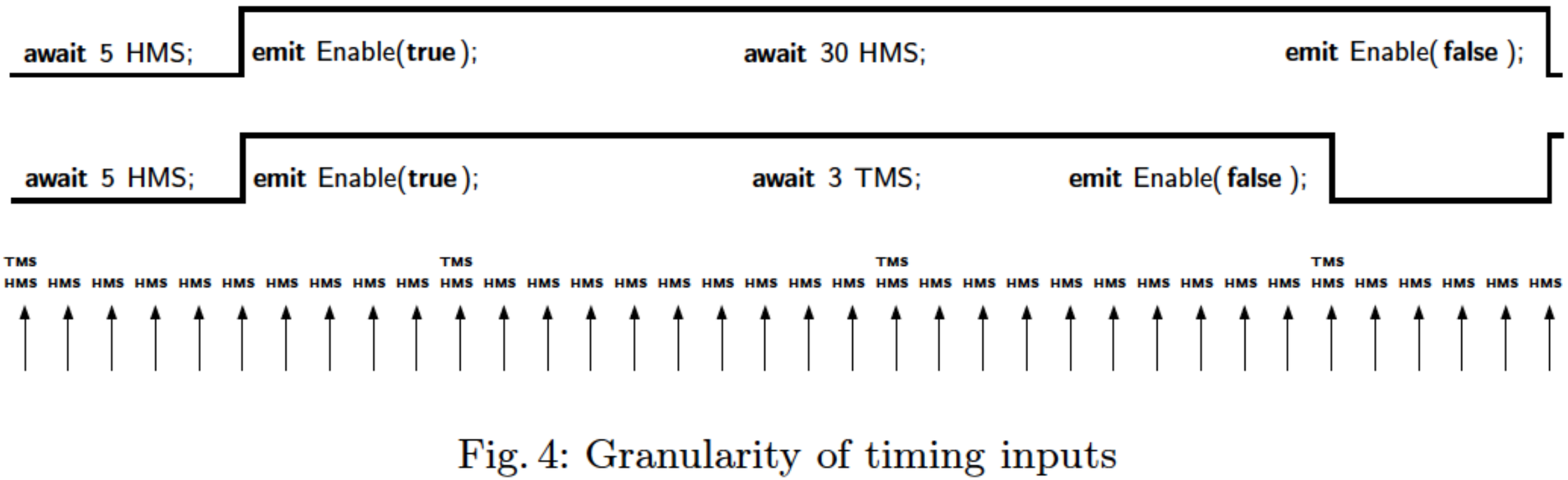


Fig. 4: Granularity of timing inputs

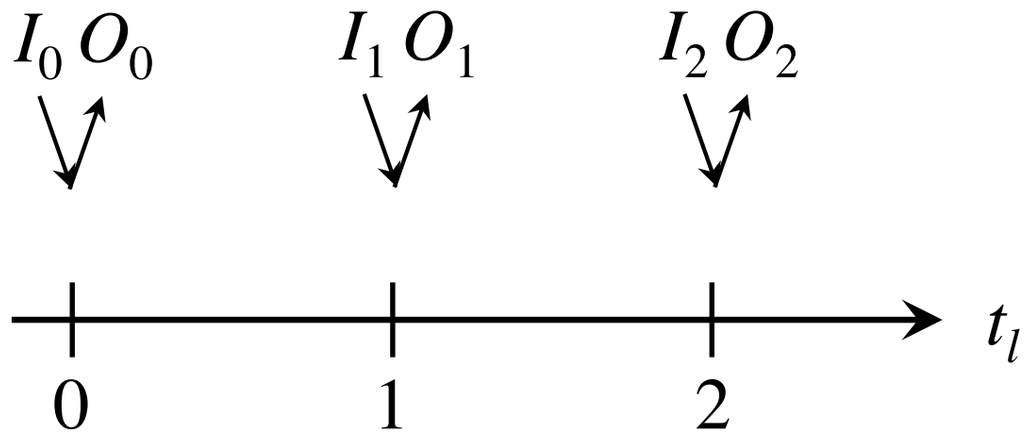
[Timothy Bourke, SYNCHRON 2009]

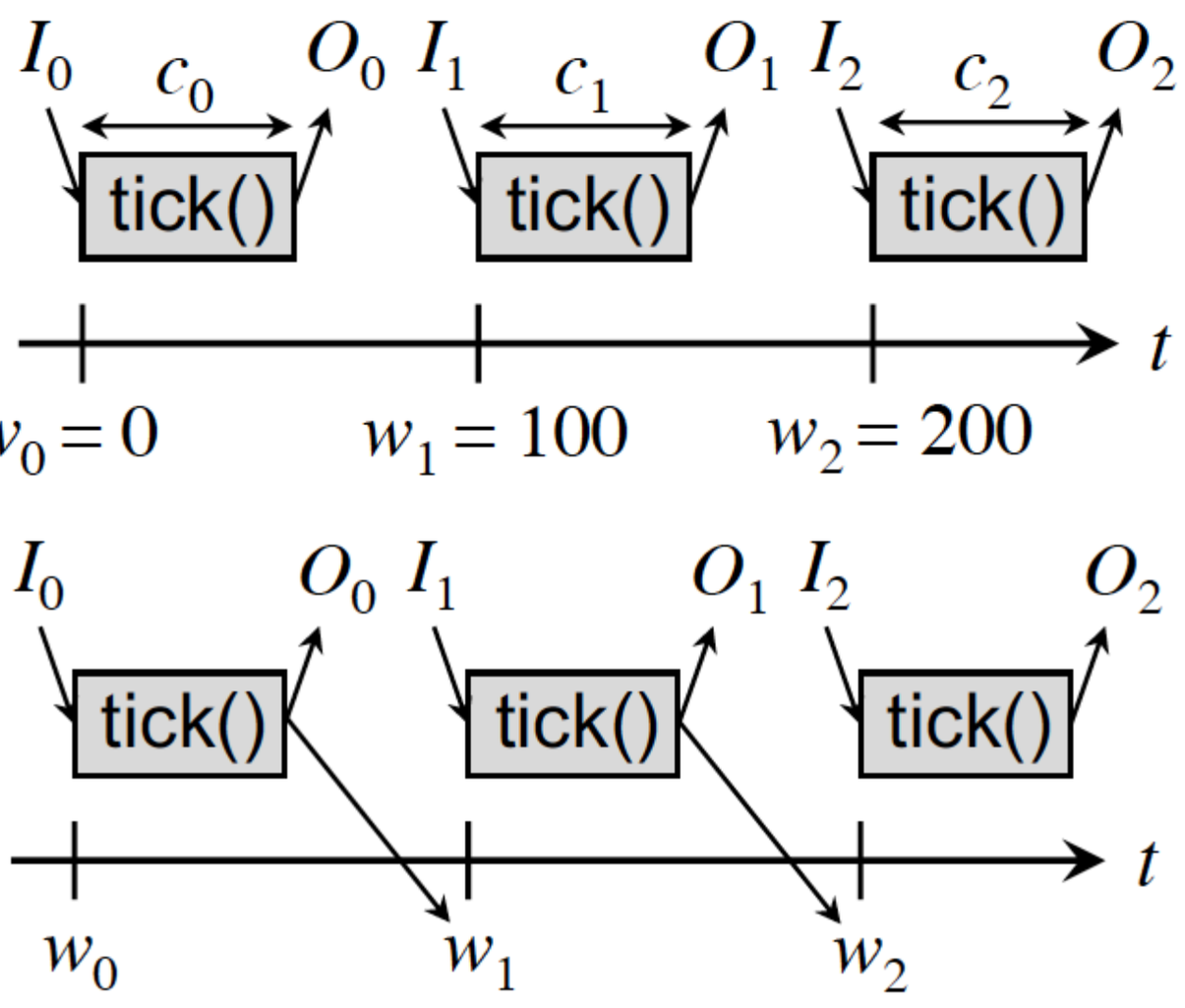
Roadmap

1. Traffic Light Example
2. Execution Models
3. Dynamic Ticks
4. Time in SCCharts: “clock”

Dynamic Ticks

- Recall logical time:





- Physical time, time-triggered:

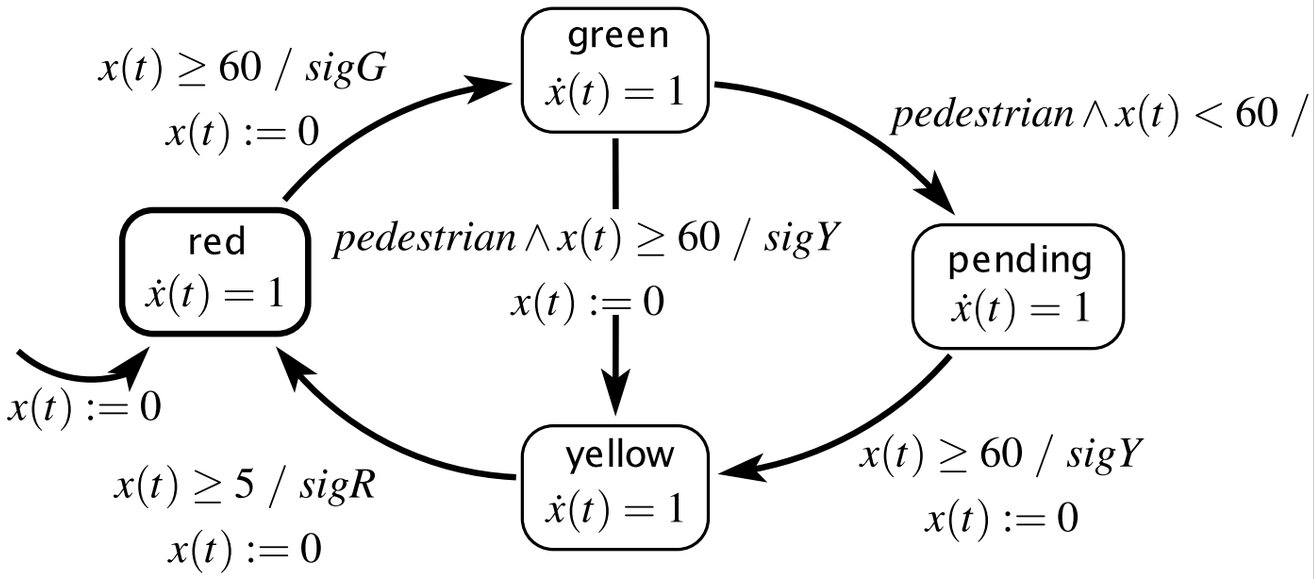
- Physical time, dynamic ticks:

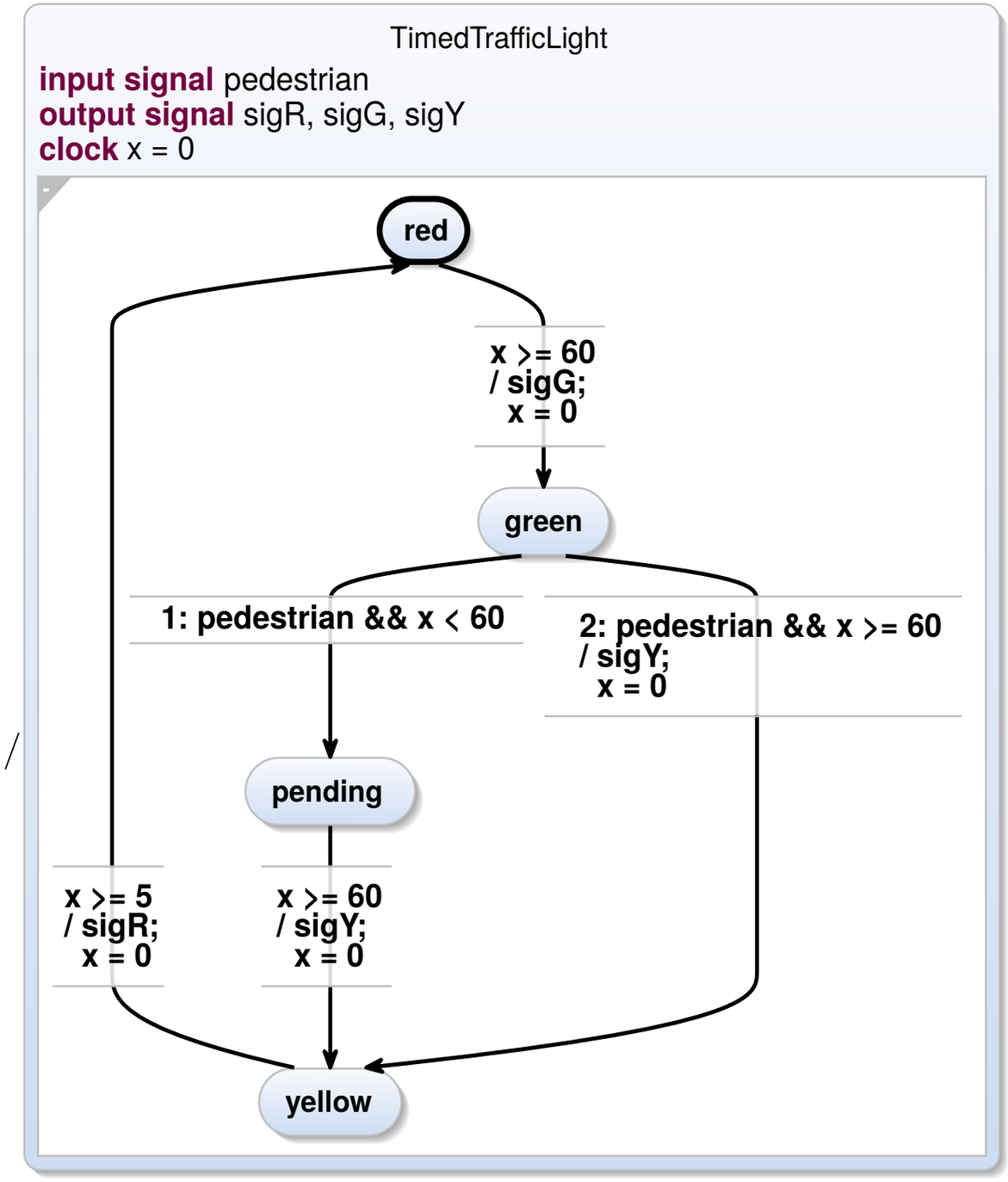
Roadmap

1. Traffic Light Example
2. Execution Models
3. Dynamic Ticks
4. Time in SCCharts: “clock”

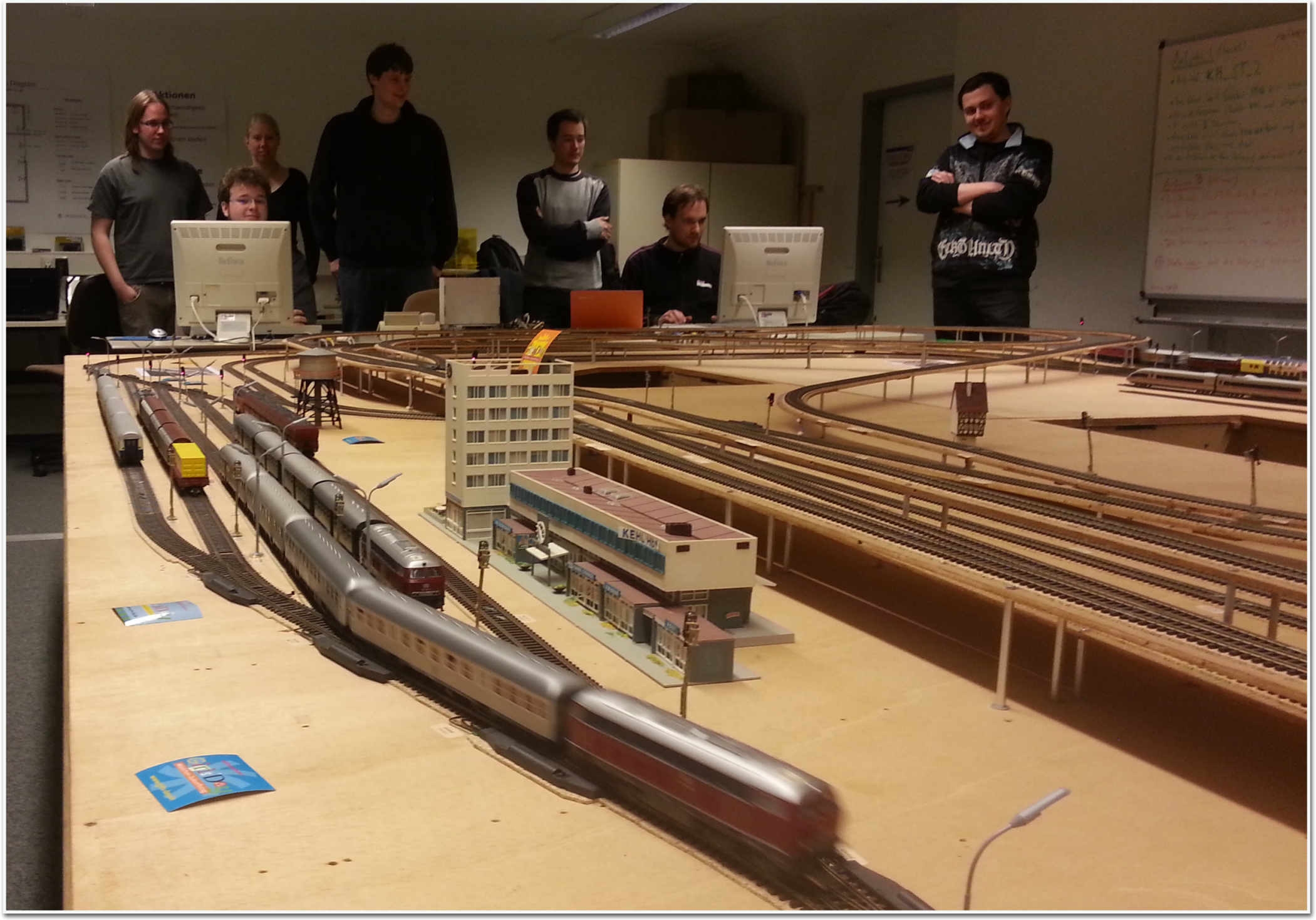
Traffic Light in SCCharts

continuous variable: $x(t) : \mathbb{R}$
inputs: *pedestrian* : pure
outputs: *sigR*, *sigG*, *sigY* : pure

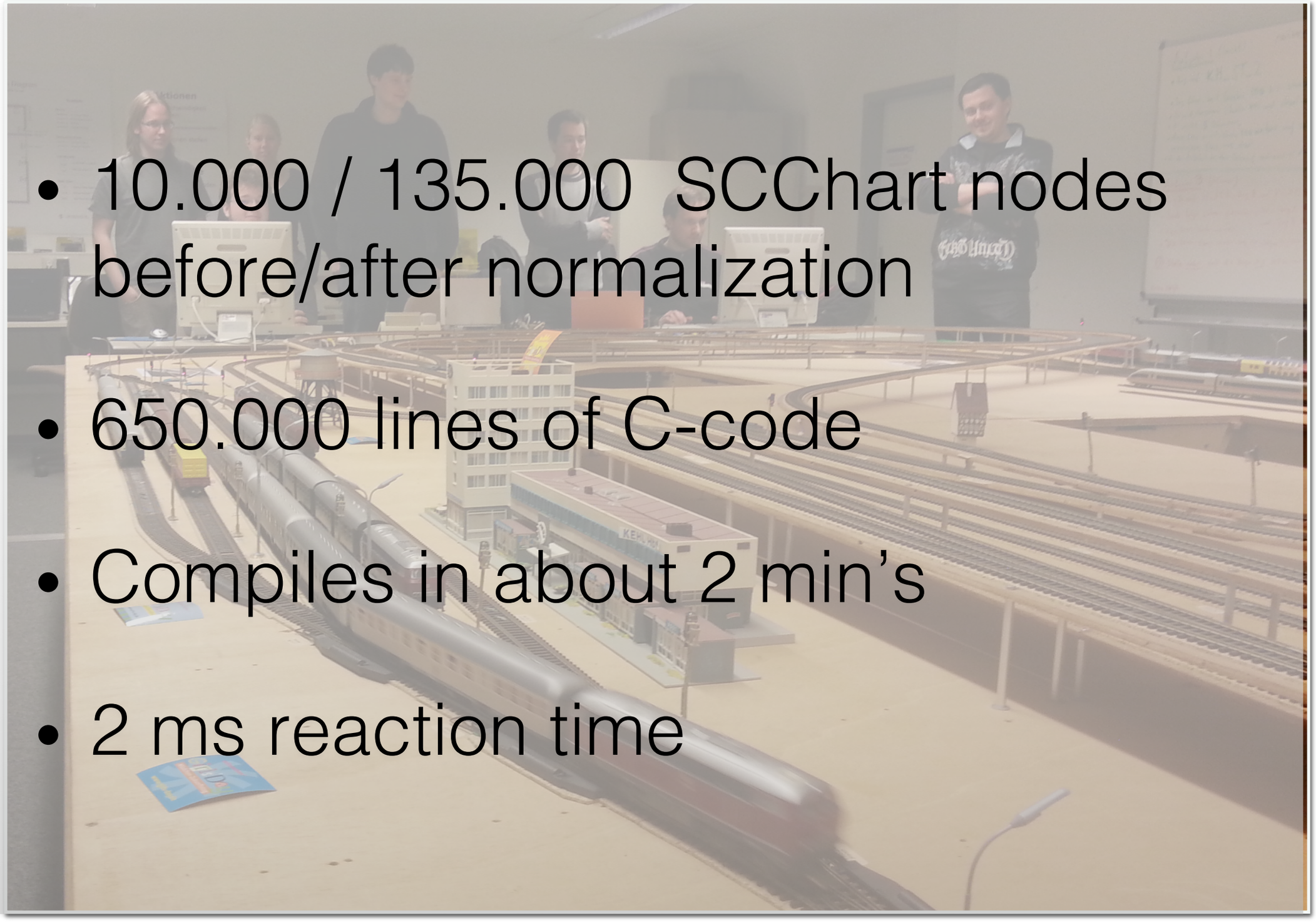




SCCharts – Classroom-Tested



SCCharts – Classroom-Tested



- 10.000 / 135.000 SCChart nodes before/after normalization
- 650.000 lines of C-code
- Compiles in about 2 min’s
- 2 ms reaction time