From Control Models to Real-Time Software

Paul Caspi

Verimag-CNRS

1. The synchronous approach

2. Simulink

The Synchronous Approach

Modelling and programming

- Control Systems
- Signal processing systems
- Hardware systems

Some shared features of these systems ?

Shared Features

- Reactive systems: flows of inputs and flows of outputs
- Parallelism: e.g., video and audio streams
  control several dimensions at the same time
  many inputs in parallel
- hard real time: because physics doesn't wait
- Dependability: often safety-critical systems
  need for sound tools
- Guaranteed bounds: on memory
  on execution times

Parallelism

Previous implementations were “naturally” parallel

- Analog devices
- Parallel hardware

What about programming ???

Parallelism in programming

Origins: time-sharing

several users using the same device :

examples:

- a printer
- a central unit
- Google ?

Needs ?

Parallelism in programming

Origins: time-sharing

several users using the same device :

examples:

- a printer
- a central unit
- Google ?

Needs ?

Processes should be as independent as possible

Yet need synchronisation

Programming concepts

- synchronisation: semaphores
  monitors
  processes, tasks, threads
  priorities

- communication: mailboxes
  shared memory

- communication and synchronisation: rendez-vous (hand-shake)
  queues
  remote procedure call
  client-server architecture

Difficulties of Parallel Software

- with debugging
  - deadlocks
  - non determinism
  - difficult to observe

- with formal verification:
  - non determinism

    $\Longrightarrow$ combinational explosion

- with time semantics:

Time Semantics of Asynchronous Programming

Java example:

is

```
     Thread.sleep(2000);
     Thread.sleep(3000);
```

equivalent to

```
     Thread.sleep(5000);
```

????

Time Semantics of Asynchronous Programming

Java example:

is

```
Thread.sleep(2000);
Thread.sleep(3000);
```

equivalent to

```
Thread.sleep(5000);
```

????

Why ??

What solutions to the problem???

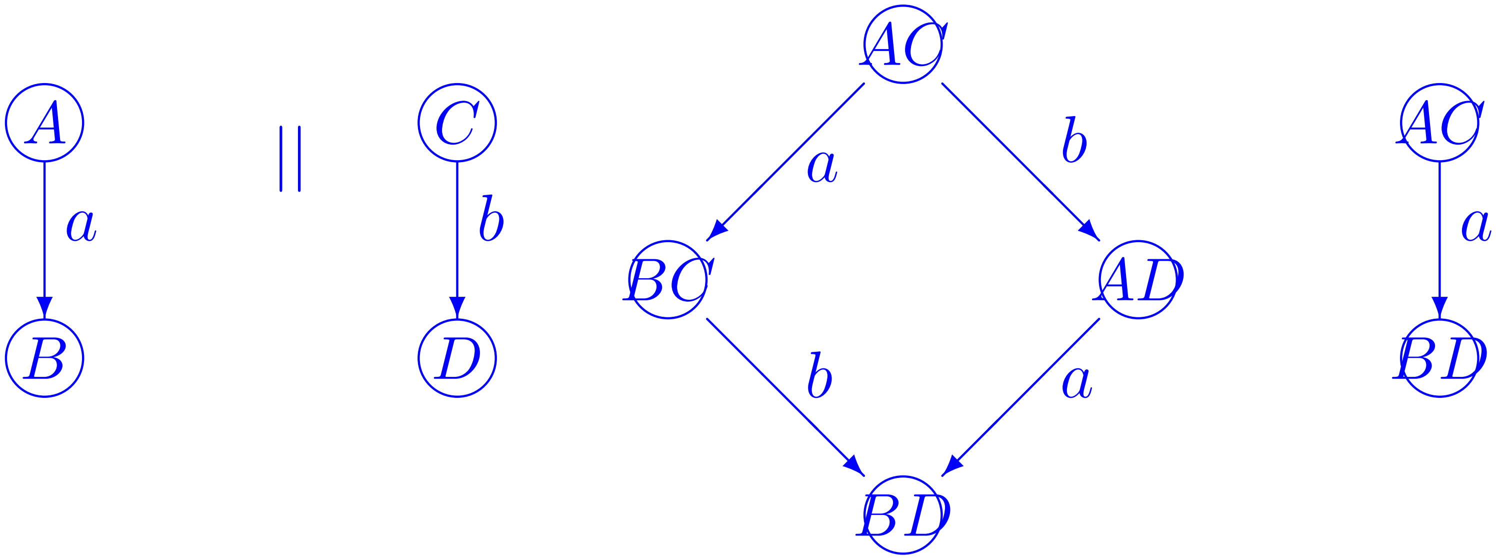
Quoting the Java documentation

"In any case, you cannot assume that invoking sleep will suspend the thread for precisely the time period specified."

Synchronous Approach

Based on the synchronous product of automata:

product asynchronous synchronous



CCS (asynchronous) is a sub-theory of SCCS

Provides a theoretical justification of practice:

Interest

- Synchronous primitives are stronger
  programming is easier
- No added non determinism:
  - easier debugging and test
  - less state explosion in formal verification
- Easier temporal reasoning:
  - synchronous steps provide a “natural” notion of logical time: Time flows equally within the different parallel tasks which thus share the same notion of time
  - Easier roll-back and recovery

Examples of the Synchronous Approach

- Simulation engines (VHDL, Simulink/Stateflow)

- Parallel and Hierachical Automata :

  Statecharts, Stateflow

- discrete time systems of equations

  Z-transforms, sampled-data control formalisms, Simulink

Simulink/Stateflow

- Presentation

- Examples

- Analysis

Some history

- Matlab simulation tool
- Simulink: a graphic interface to Matlab
- Stateflow: hierachical and parallel automata integrated to Simulink

  the first tool box allowing for this integration
- Real-Time Workshop automatic code generation for Simulink/Stateflow

  Several other code generators :
  - ASCET(ETAS), TargetLink(DSpace),
  - SCADE(Esterel-Technologies), based on Lustre (Verimag), the only one qualified for DO178B level A (safety critical Aerospace applications) and IEN 50128 SIL4 (safety critical railway applications)
- Many specialised libraries for control and architecture

Interest of Simulink

Interest:

- Allow for modelling both
  - Continuous-time dynamical systems :
    physical plants, analog controllers
  - Discrete-time dynamical systems:
    logical systems, computerised control
- Mostly based on sound mathematical principles:
  Laplace transform, Z-transform, differential equation solvers

The *defacto* standard in control modelling and implementation

Drawbacks

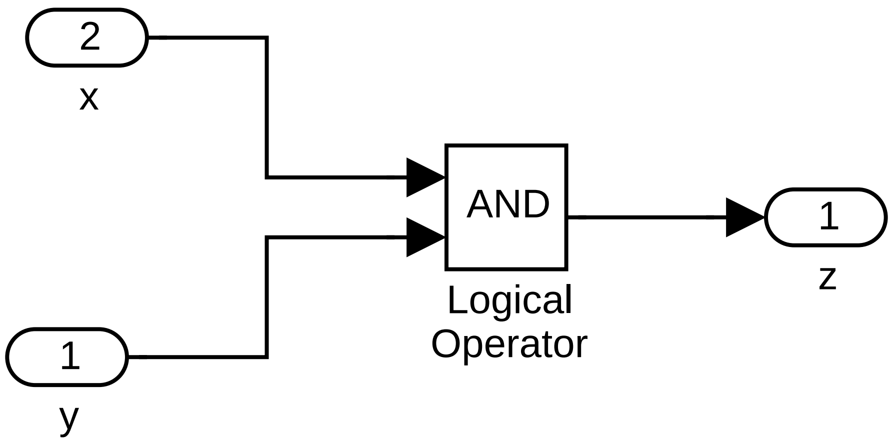
- But quite poor computer science principles:

  poor typing, poor static checks,

  unsafe side effects

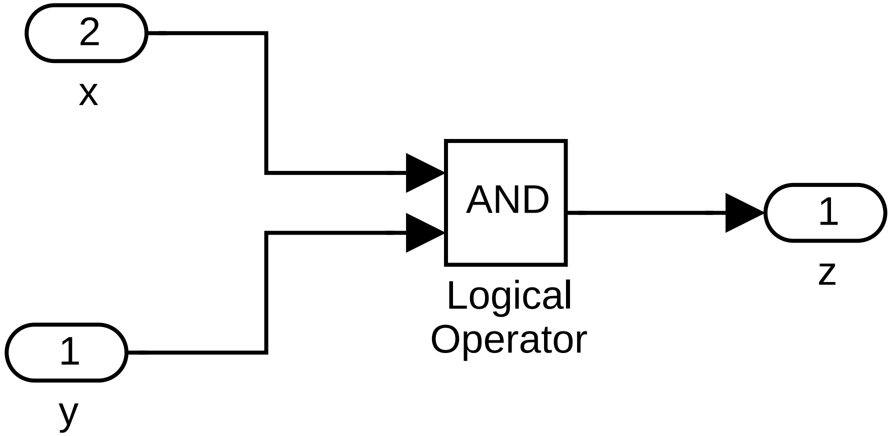
- This has to be taken into account when generating software!

Example: And gate



What does it mean?

Example: And gate



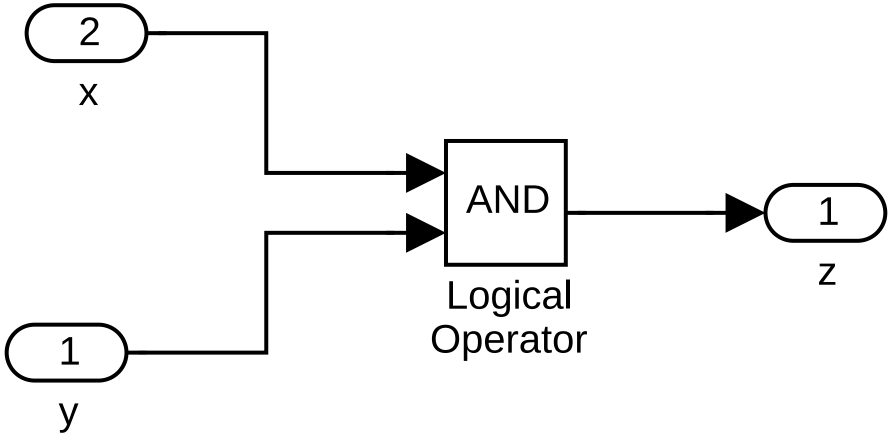
What does it mean?

Precise mathematical meaning:

$\forall t \in T : \ z(t) = x(t) \wedge y(t)$

But what is T ?

Example: And gate



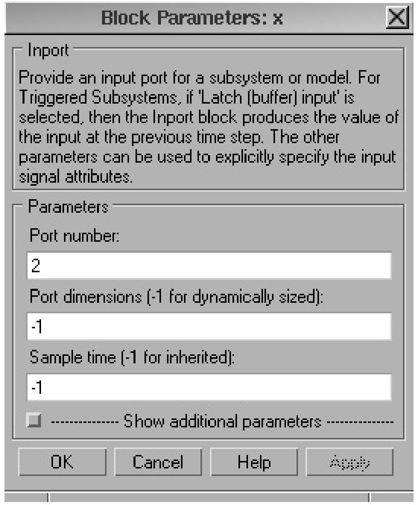
What does it mean?

Precise mathematical meaning:

$\forall t \in T: \;\; z(t) = x(t) \wedge y(t)$

But what is T ?

Two possible settings:



Block Parameters: x

Inport

Provide an input port for a subsystem or model. For Triggered Subsystems, if 'Latch (buffer) input' is selected, then the Inport block produces the value of the input at the previous time step. The other parameters can be used to explicitly specify the input signal attributes.

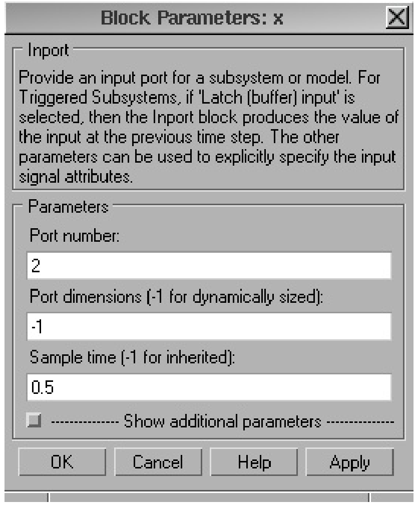
Parameters

Port number: 2

Port dimensions (-1 for dynamically sized): -1

Sample time (-1 for inherited): -1

□ Show additional parameters



Block Parameters: x

Inport

Provide an input port for a subsystem or model. For Triggered Subsystems, if 'Latch (buffer) input' is selected, then the Inport block produces the value of the input at the previous time step. The other parameters can be used to explicitly specify the input signal attributes.

Parameters

Port number: 2

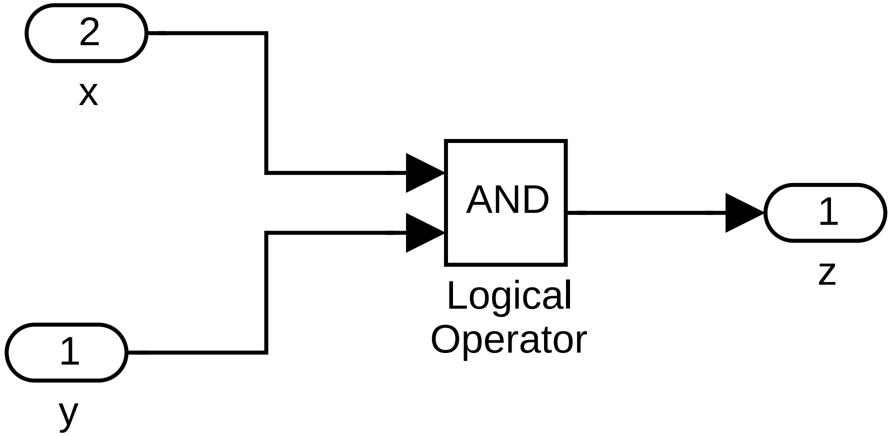
Port dimensions (-1 for dynamically sized): -1

Sample time (-1 for inherited): 0.5

□ Show additional parameters

takes the time scale of input   periodic sampling

Example: And gate



What does it mean?

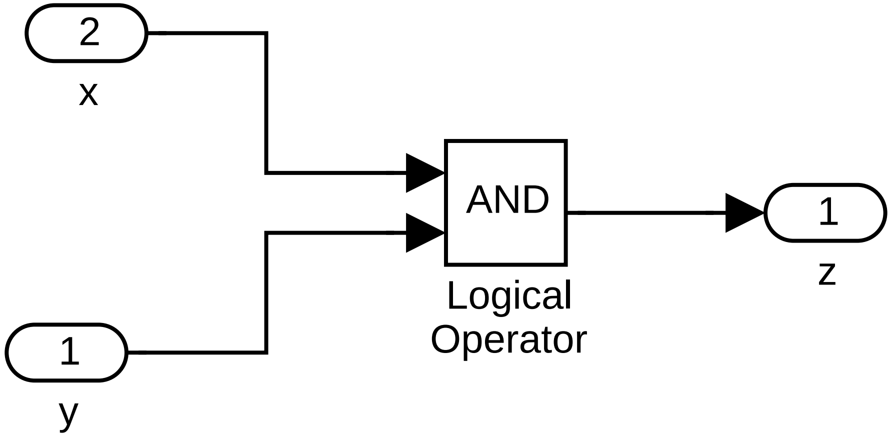
Precise mathematical meaning:

$\forall t \in T : \ z(t) = x(t) \wedge y(t)$

But what is T ?

What if x and y have different time scales?

Example: And gate



What does it mean?

Precise mathematical meaning:

$\forall t \in T : \; z(t) = x(t) \wedge y(t)$

But what is T ?

What if x and y have different time scales?

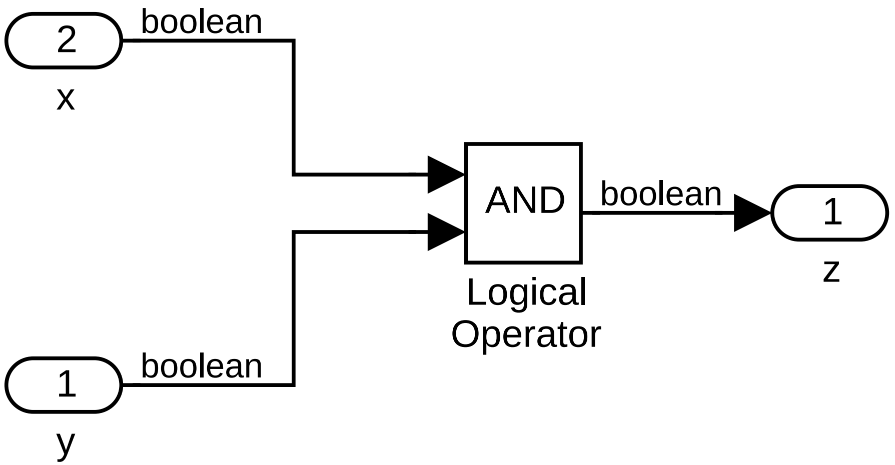
$$T = T_x \cup T_y$$

Each signal keeps its preceeding value when it is not updated

Entails a need for a “clock analysis” when generating code

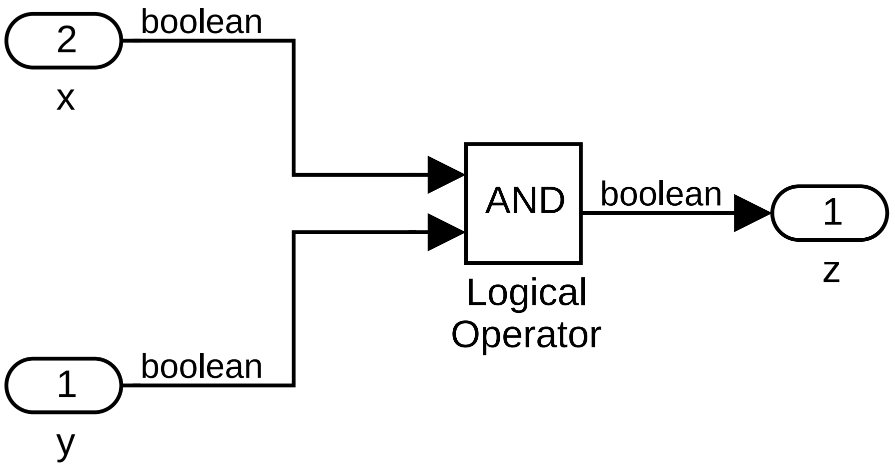
Typing Issues

We can ask for types and get this

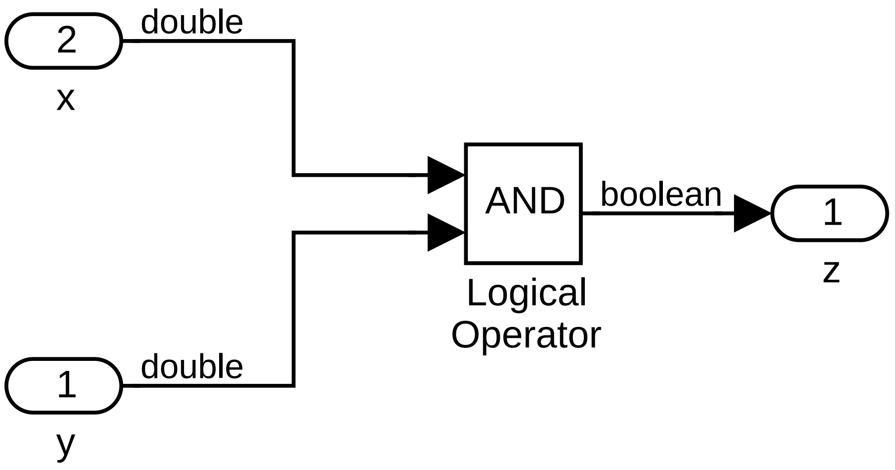


Typing Issues

We can ask for types and get this



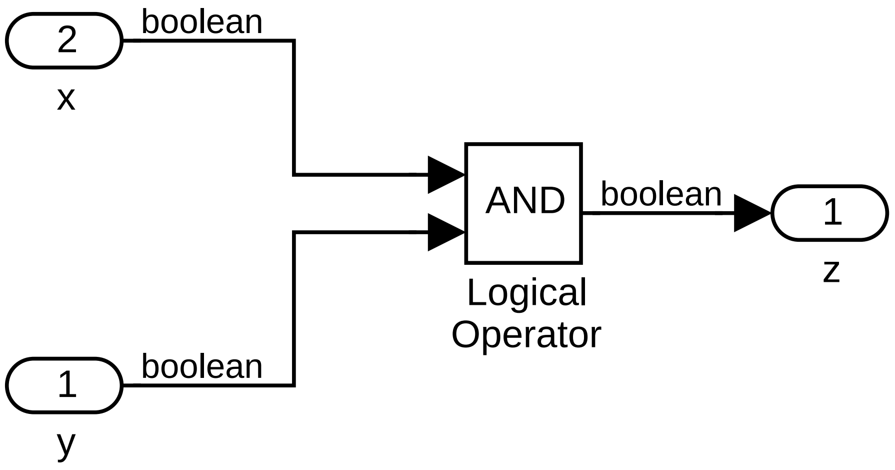
But we can also get this



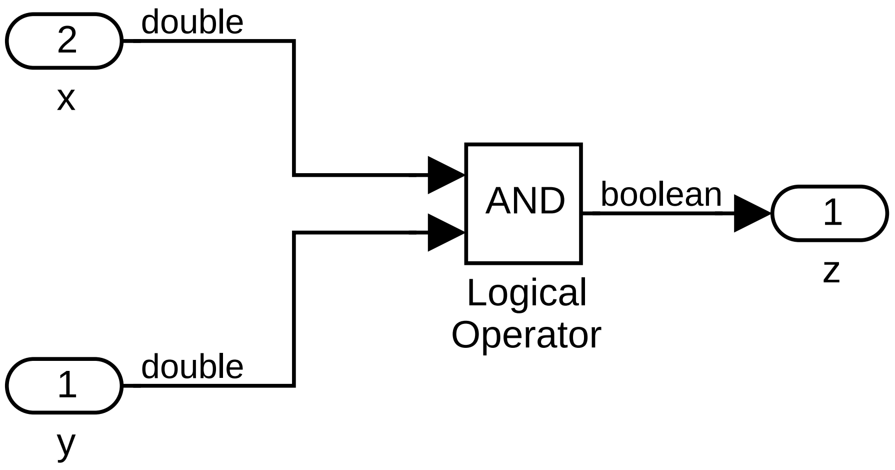
What does it mean ?

Typing Issues

We can ask for types and get this

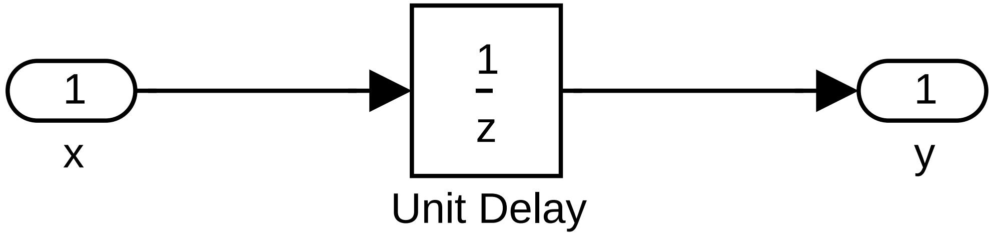


But we can also get this



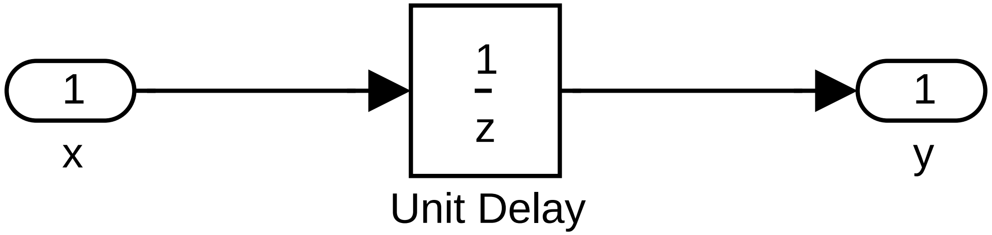
What does it mean ? Typing issues are important for safety purposes and should be carefully addressed when generating code

Example: Delay Block



What does it mean?

Example: Delay Block

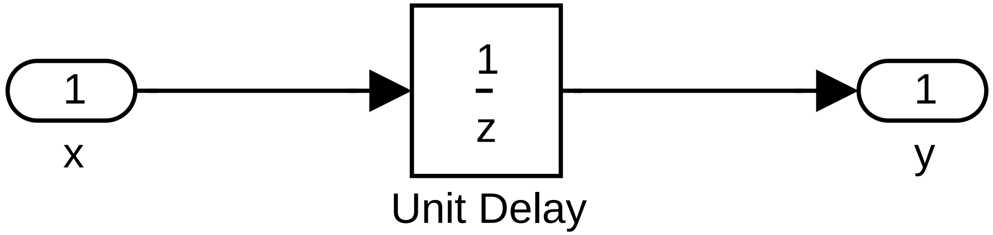


What does it mean?

Precise mathematical meaning:

$$\begin{cases} y(0) = \mathit{some\ setting} \\ \forall t \in T: \; y(t+1) = x(t) \end{cases}$$

Example: Delay Block



What does it mean?

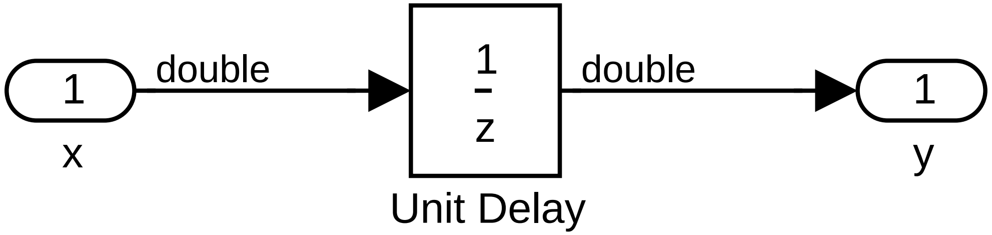
Precise mathematical meaning:

$$\begin{cases} y(0) = \mathit{some\ setting} \\ \forall t \in T: \ y(t+1) = x(t) \end{cases}$$

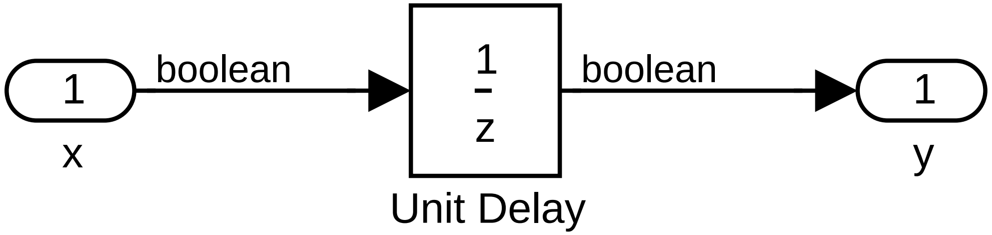
T needs to be a discrete time scale: “clock analysis” needed again

Typing Issues

We can get this:



Or this:



“Polymorphic” operator

Basics of Z-Transform

We associate with a discrete-time signal $x : N \rightarrow R$

its Z-transform: $$Z(x)(z) = \sum_{0}^{\infty} \frac{x(i)}{z^i}$$

This transform has several desirable properties:

- it is linear: $Z(ax + by) = aZ(x) + bZ(y)$
- it transforms delays into products :

  if $\begin{cases} x^-(0) = 0 \\ x^-(n+1) = x(n) \end{cases}$ then $Z(x^-)(z) = \dfrac{Z(x)(z)}{z}$

Why z-transform?

Basics of Z-Transform

We associate with a discrete-time signal $x : N \rightarrow R$

its Z-transform: $$Z(x)(z) = \sum_{0}^{\infty} \frac{x(i)}{z^i}$$

This transform has several desirable properties:

- it is linear: $Z(ax + by) = aZ(x) + bZ(y)$
- it transforms delays into products :

  if $\begin{cases} x^-(0) = 0 \\ x^-(n+1) = x(n) \end{cases}$ then $Z(x^-)(z) = \dfrac{Z(x)(z)}{z}$

Why z-transform?

Allows algebraic computations over difference equations

Proof of the Delay Operator

if $\begin{cases} x^{-}(0) = 0 \\ x^{-}(n+1) = x(n) \end{cases}$ then $Z(x^{-})(z) = \frac{1}{z} Z(x)(z)$

Proof:

$$Z(x)(z) = \sum_{0}^{\infty} \frac{x(i)}{z^{i}}$$

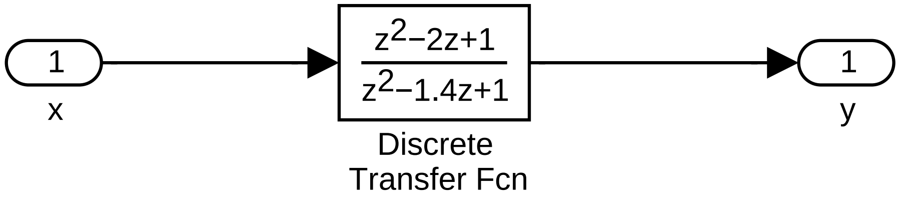
$$\frac{1}{z} Z(x)(z) = \frac{1}{z} \sum_{0}^{\infty} \frac{x(i)}{z^{i}} = \sum_{0}^{\infty} \frac{x(i)}{z^{i+1}} = \sum_{0}^{\infty} \frac{x^{-}(i+1)}{z^{i+1}}$$

$$\frac{1}{z} Z(x)(z) = \sum_{1}^{\infty} \frac{x^{-}(i)}{z^{i}} = \frac{0}{z^{0}} + \sum_{1}^{\infty} \frac{x^{-}(i)}{z^{i}} = \sum_{0}^{\infty} \frac{x^{-}(i)}{z^{i}}$$

$$\frac{1}{z} Z(x)(z) = Z(x^{-})(z)$$

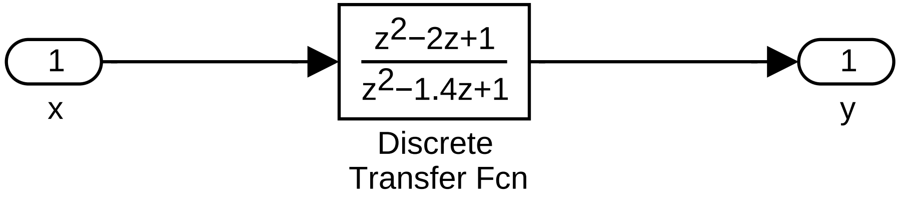
Example: Filters

A second-order linear filter:

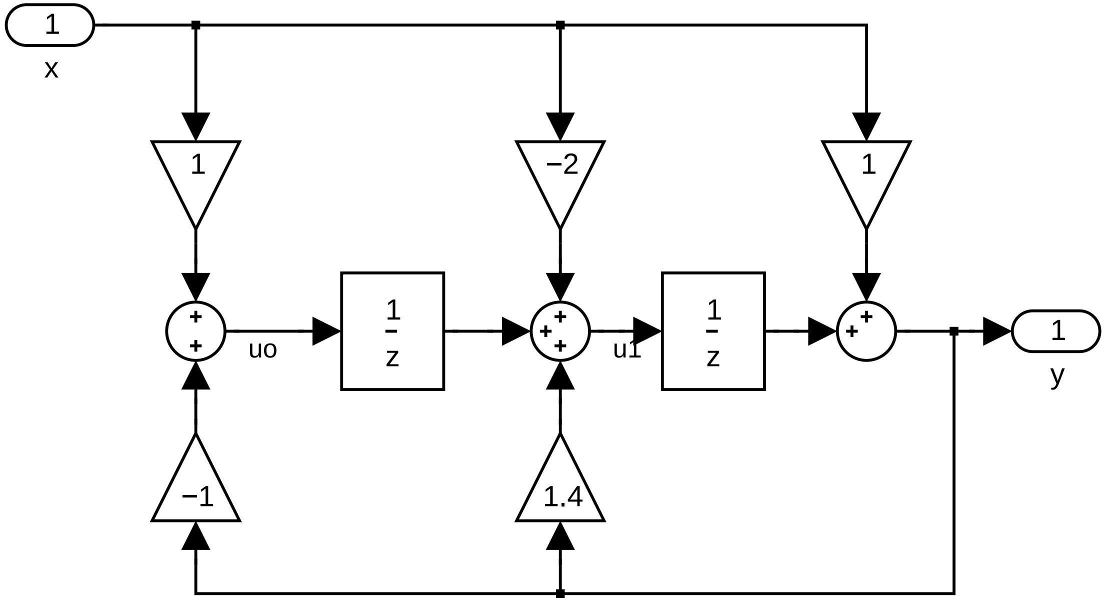


Example: Filters

A second-order linear filter:

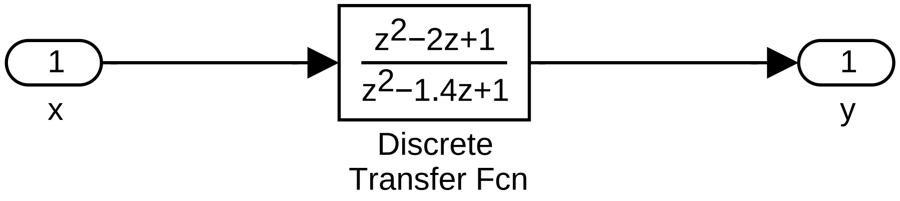


Another one:

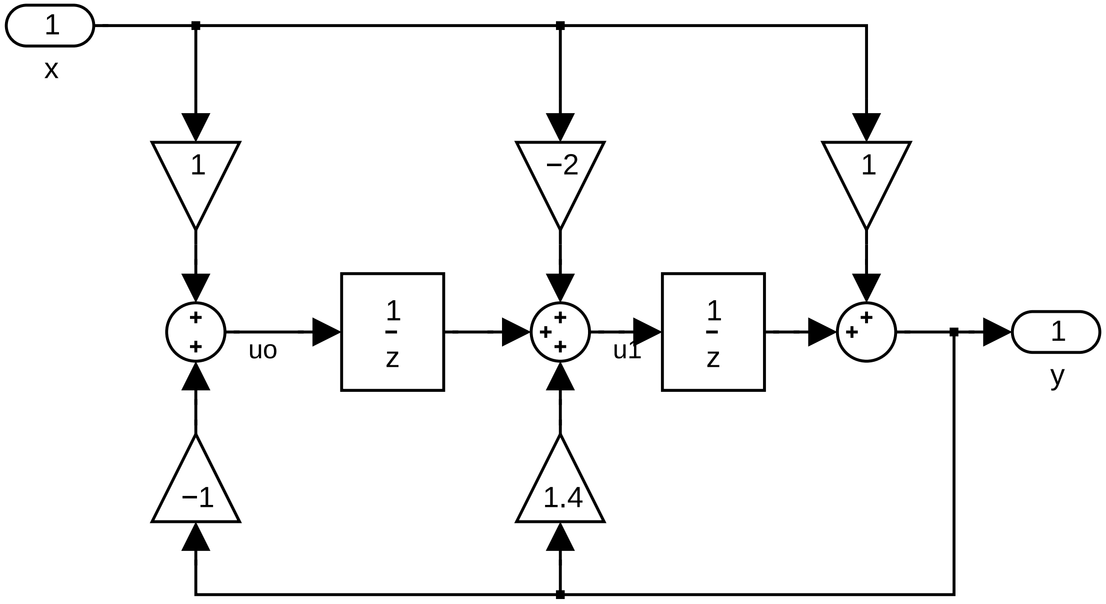


Example: Filters

A second-order linear filter:

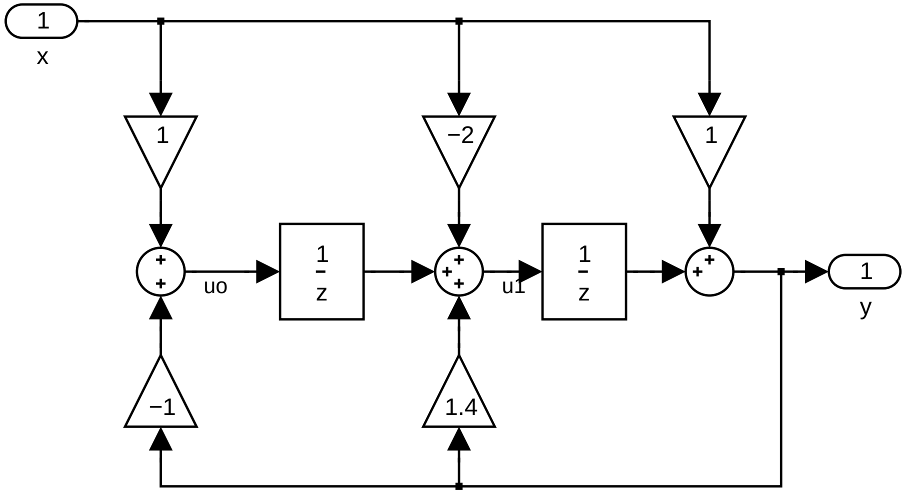


Another one:



They are the same ! Why ?

Proof



$$u_0 = x - y$$

$$u_1 = 1.4y - 2x + u_0/z$$

$$y = x + u_1/z$$

$$y = x + 1/z(1.4y - 2x + 1/z(x - y))$$

$$y = x - 2x/z + x/z^2 + 1.4y/z - y/z^2$$

$$y - 1.4y/z + y/z^2 = x - 2x/z + x/z^2$$

$$yz^2 - 1.4yz + y = xz^2 - 2xz + x$$

$$(z^2 - 1.4z + 1)y = (z^2 - 2z + 1)x$$

$$y = \frac{z^2 - 2z + 1}{z^2 - 1.4z + 1}x$$

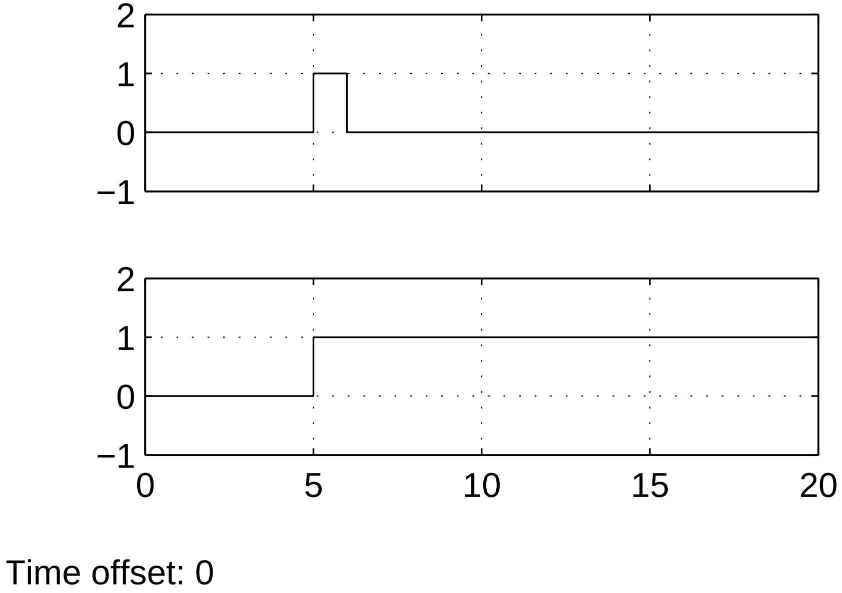
Programming in Simulink

Design a sub-system with one boolean input and one boolean output such that the output

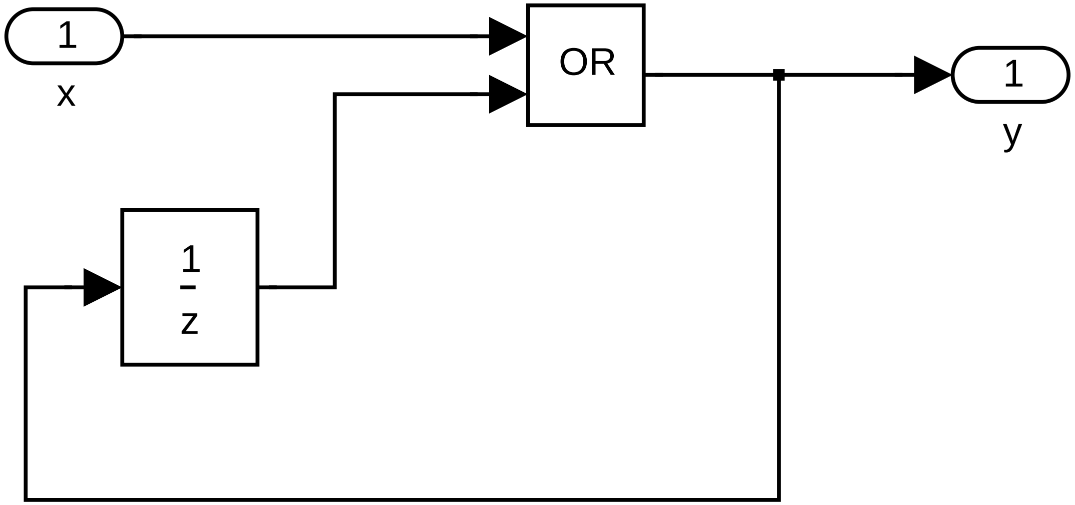
- is initially false
- becomes true as soon as the input becomes true
- stays true for ever as soon as it has become true once

input

output



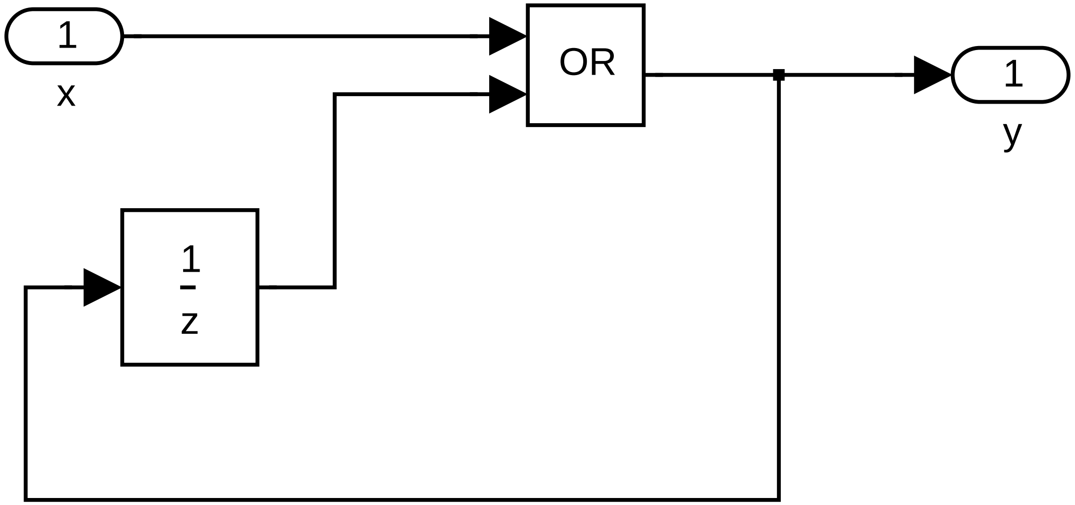
Solution



Why?

x	0	0	1	0	0	1	0	0
y/z	0	0						
$y = y/z \vee x$	0	0						

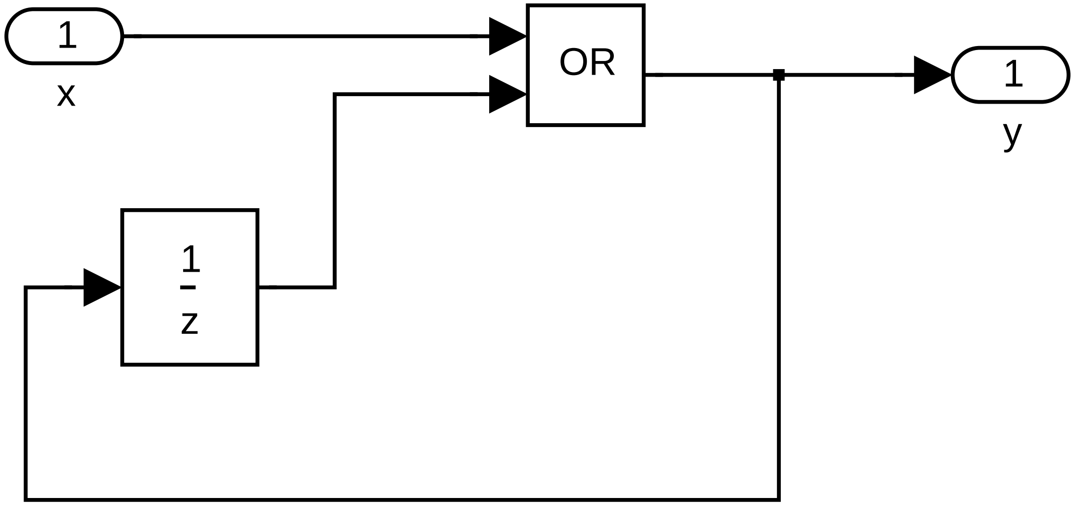
Solution



Why?

x	0	0	1	0	0	1	0	0
y/z	0	0	0					
$y = y/z \vee x$	0	0						

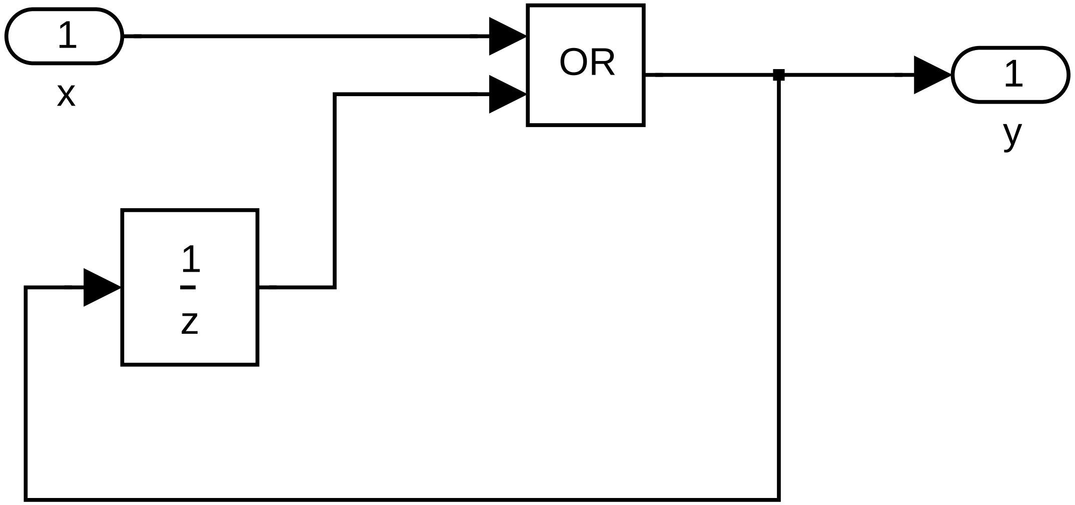
Solution



Why?

x	0	0	1	0	0	1	0	0
y/z	0	0	0					
$y = y/z \vee x$	0	0	1					

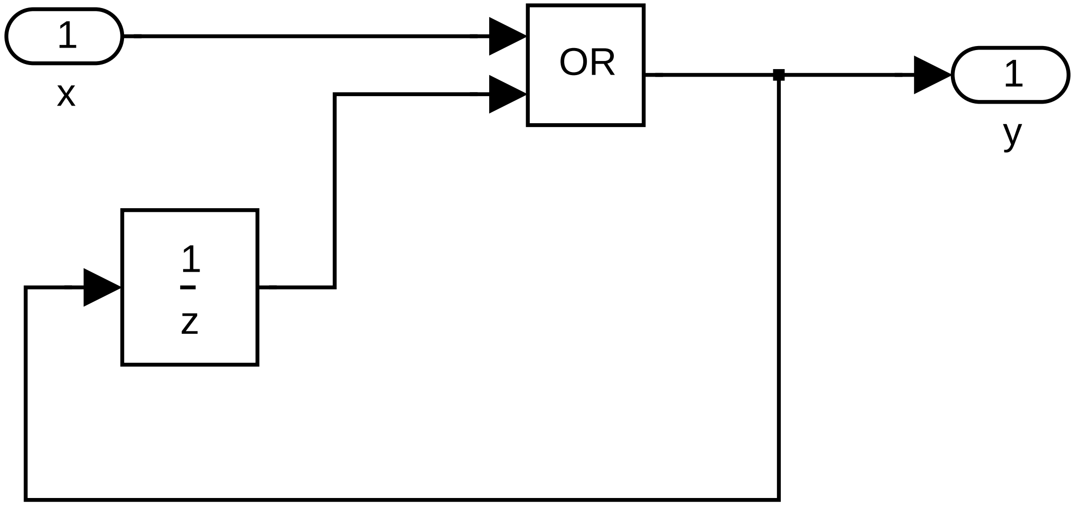
Solution



Why?

x	0	0	1	0	0	1	0	0
y/z	0	0	0	1				
$y = y/z \vee x$	0	0	1					

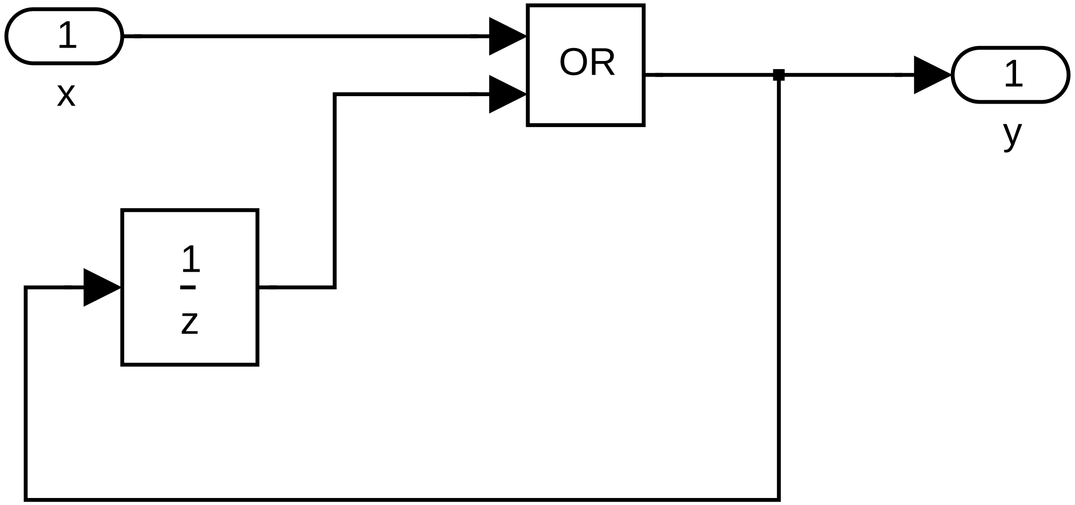
Solution



Why?

x	0	0	1	0	0	1	0	0
y/z	0	0	0	1				
$y = y/z \vee x$	0	0	1	1				

Solution



Why?

x	0	0	1	0	0	1	0	0
y/z	0	0	0	1	1	1	1	1
$y = y/z \vee x$	0	0	1	1	1	1	1	1

Programming in Simulink: Functional versus Imperative

Simulink is mainly a functional and component-based approach:

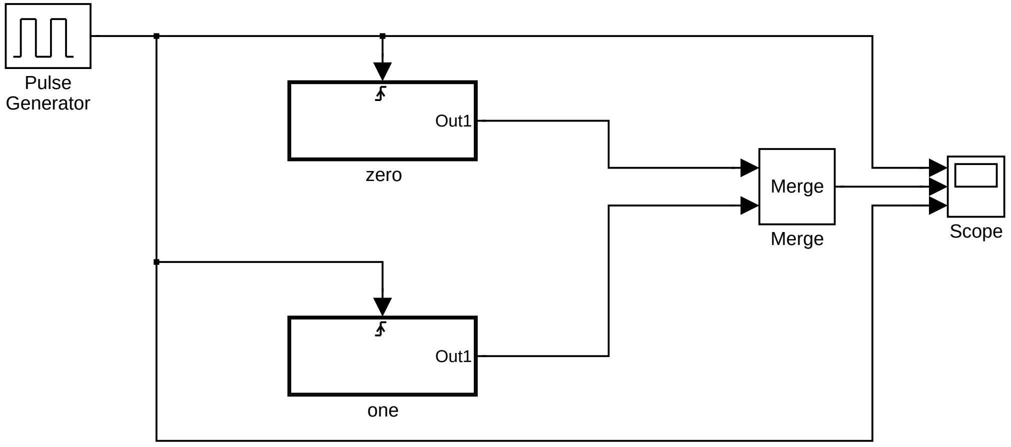
- most Simulink subsystems is are functions from input signals to output signals.

  As such, it is very safe: the behaviour of a functional subsystem depends only on its inputs and is the same whatever be the context in which it is used.

- It is similar to taking a component off the shelf and wiring it in a design.

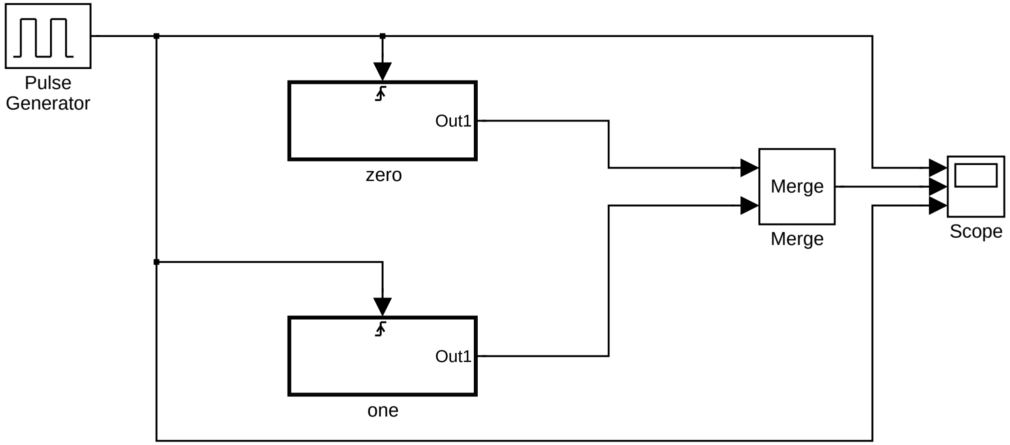
But there are still non functional blocks whose behaviour departs from this functional style. These operators should be used with care as they may produce side effects.

Imperative Features: the merge operator



What's wrong with it ?

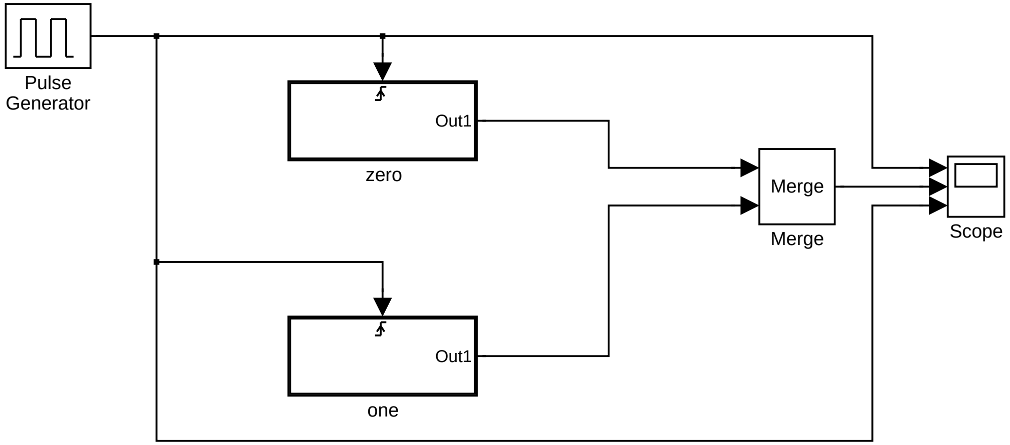
Imperative Features: the merge operator



What's wrong with it ?

Both subsystems are triggered at the same time: the result depends on the order into which the subsystems are created:

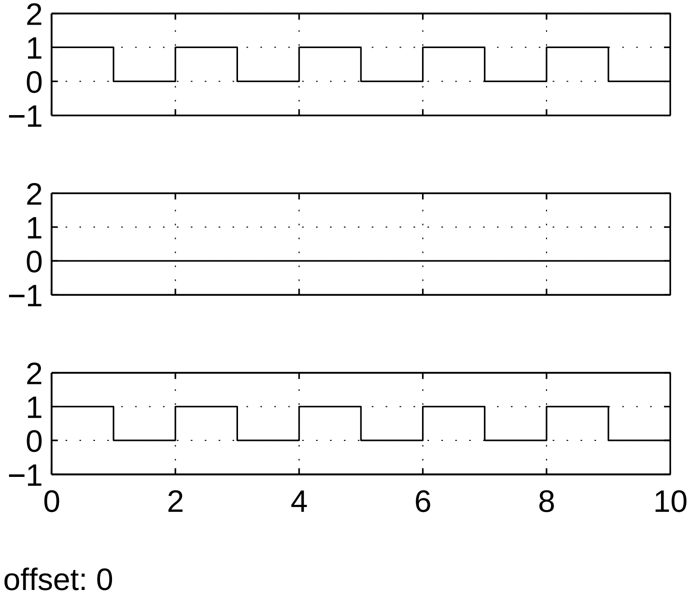
Imperative Features: the merge operator



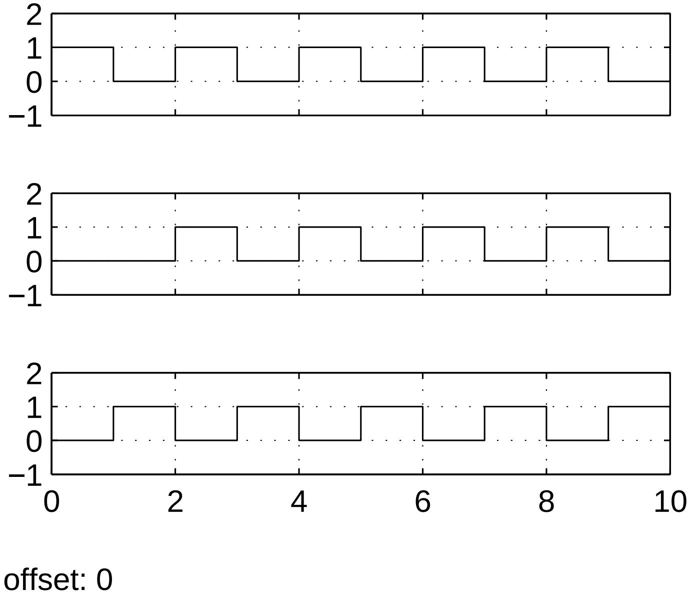
What's wrong with it ?

Both subsystems are triggered at the same time: the result depends on the order into which the subsystems are created:

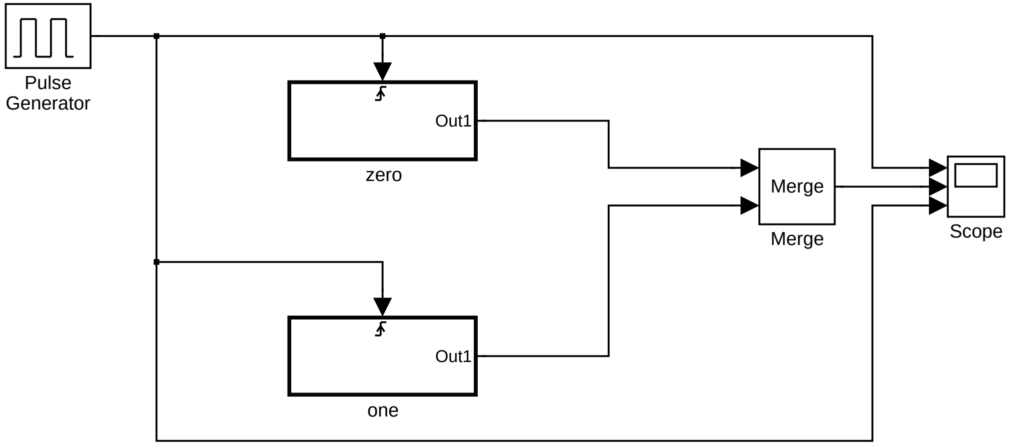
one created before zero



zero created before one

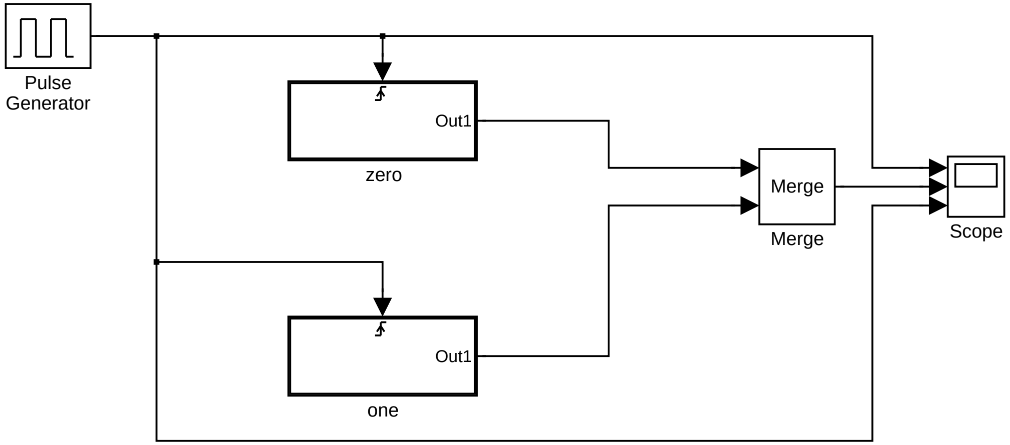


Imperative Features: the merge operator



How to make it order independent?

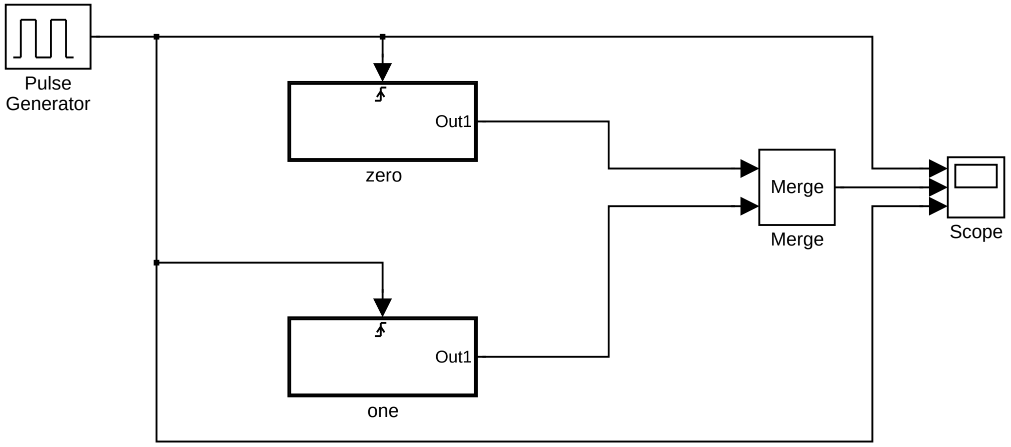
Imperative Features: the merge operator



How to make it order independent?

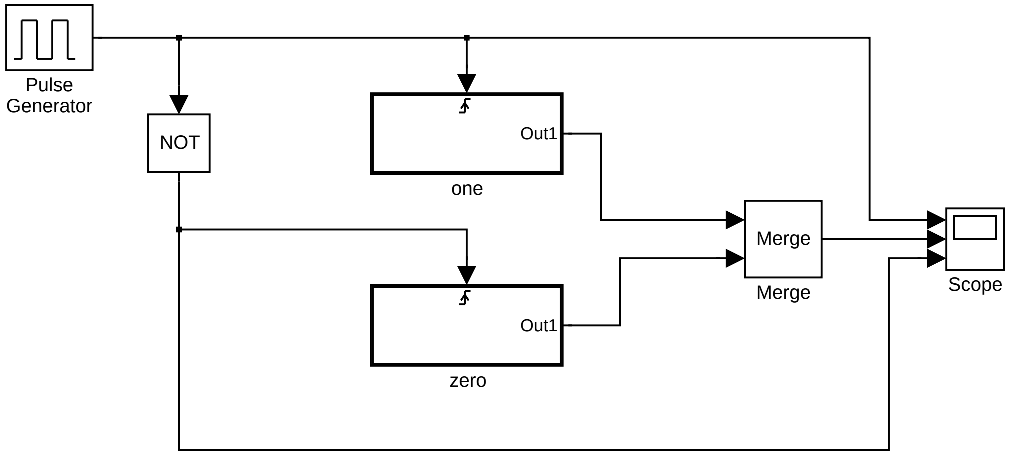
Ensure that the triggers be exclusive

Imperative Features: the merge operator

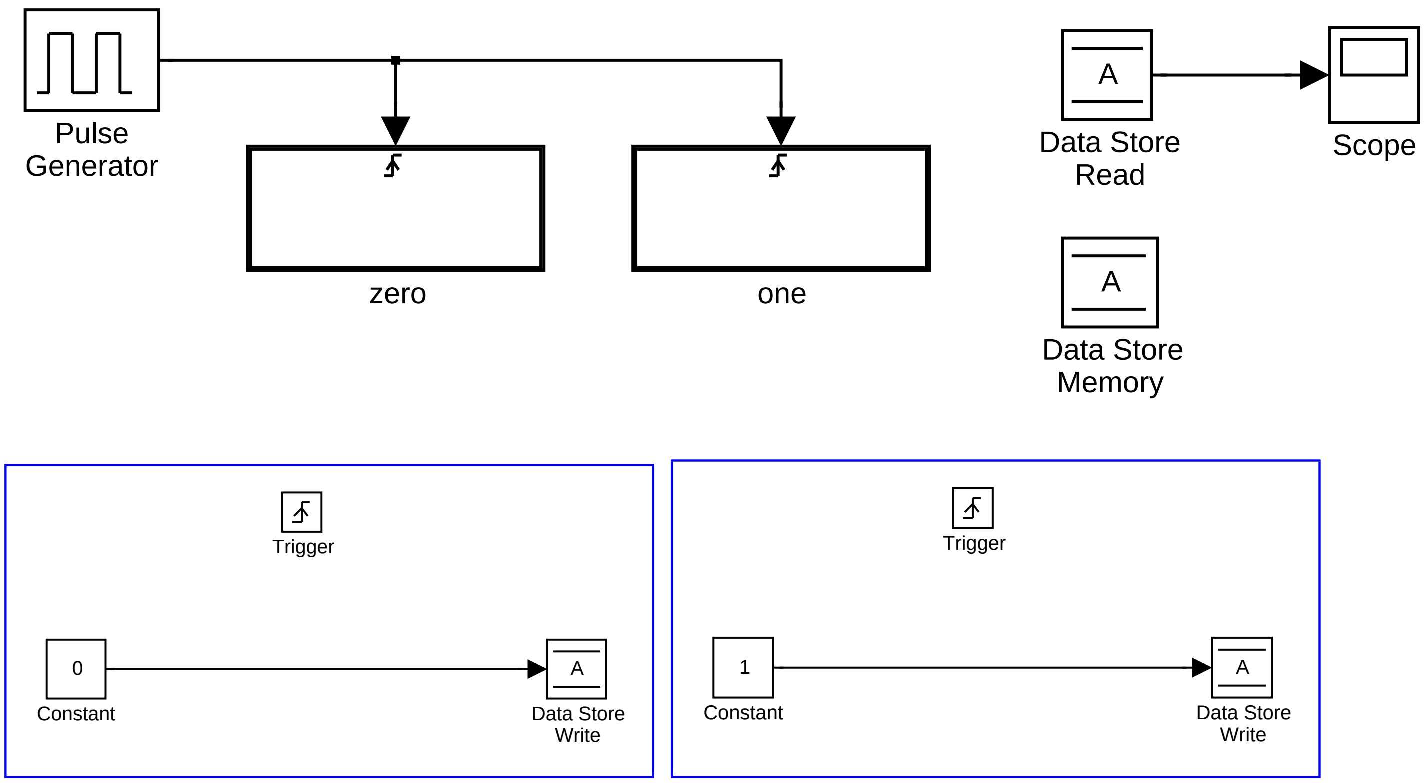


How to make it order independent?

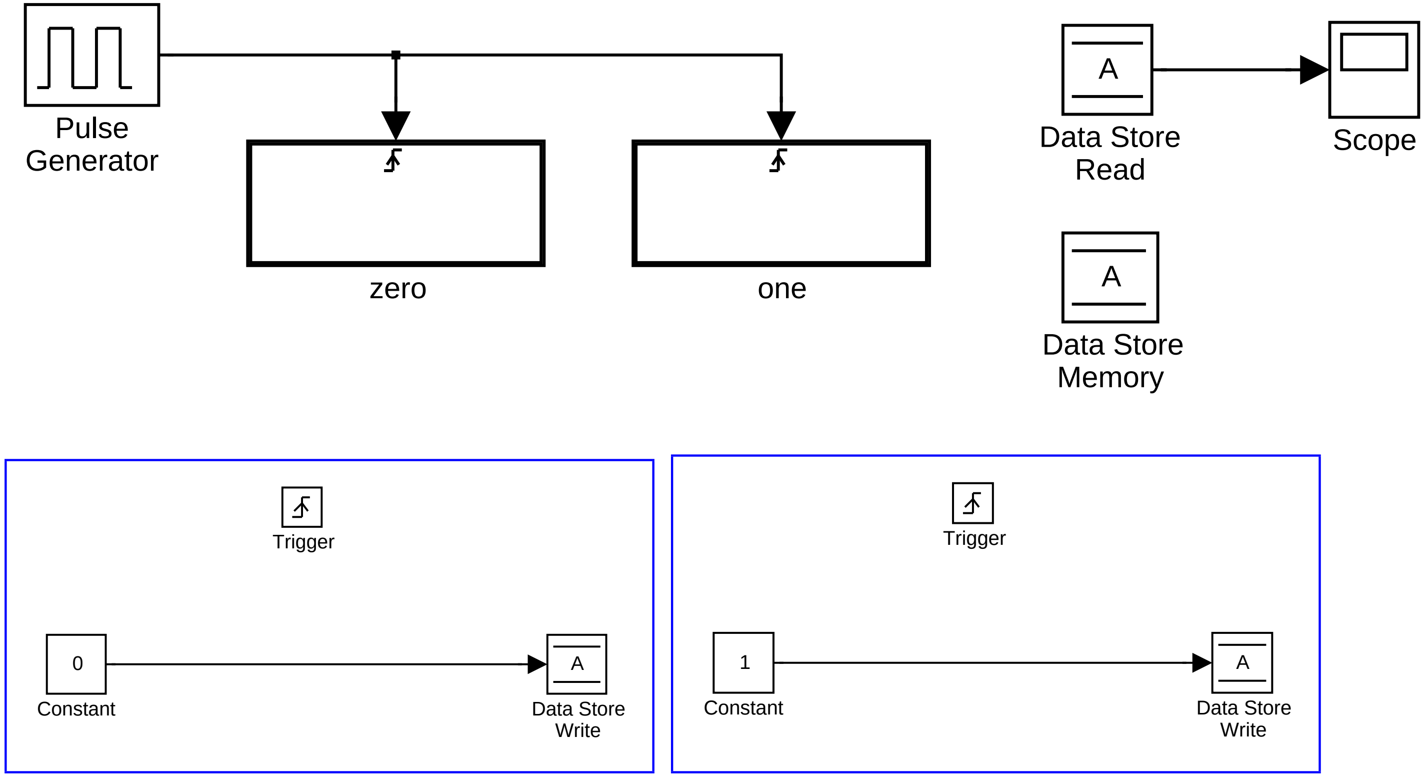
Ensure that the triggers be exclusive



Imperative Features: Memory Block

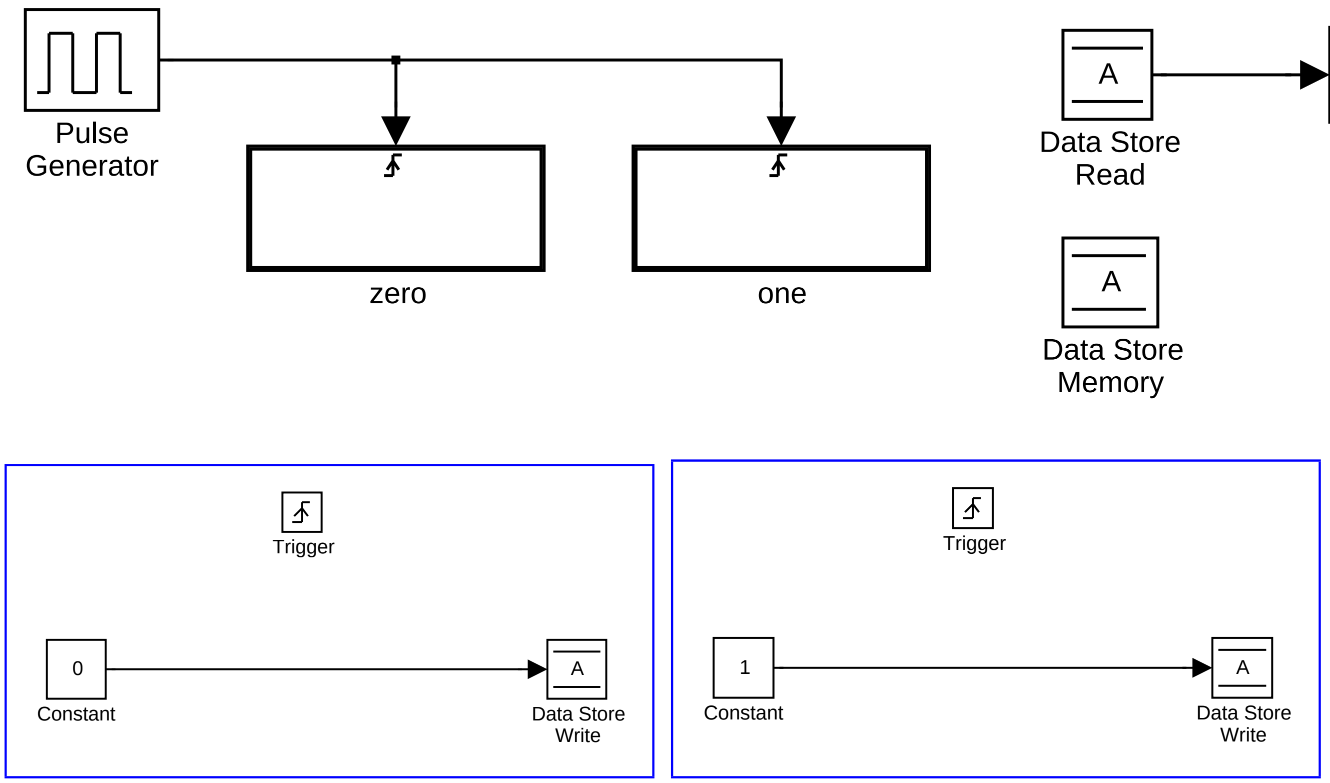


Imperative Features: Memory Block



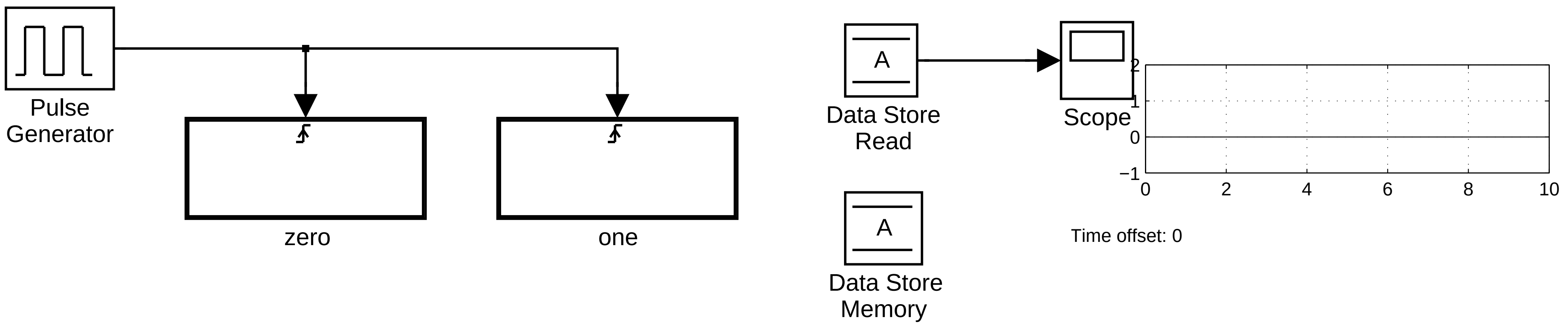
Same problem: when the triggers are non exclusive the execution order depends on some strange features.

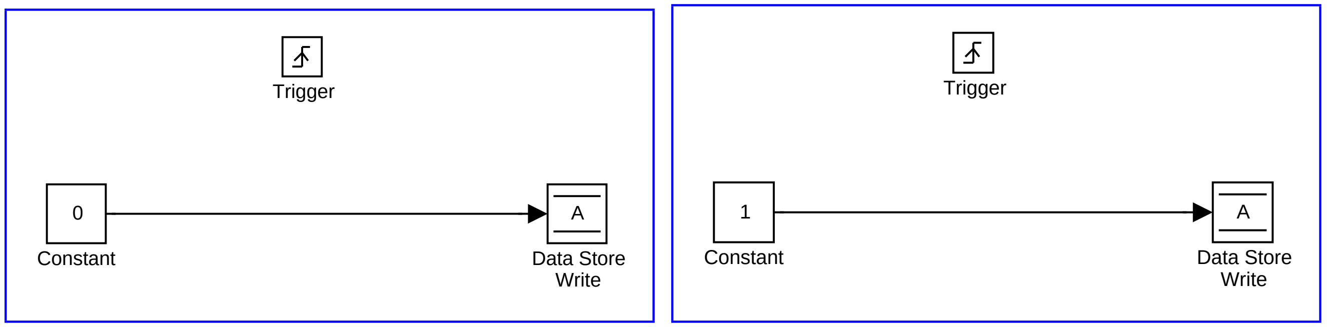
Imperative Features: Memory Block



Same problem: when the triggers are non exclusive the execution order depends on some strange features. What can we get ?

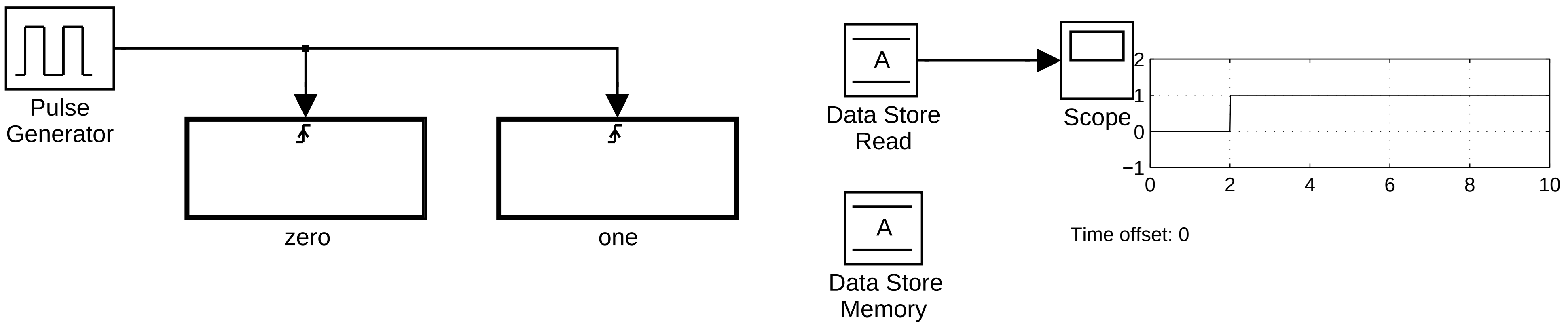
Imperative Features: Memory Block

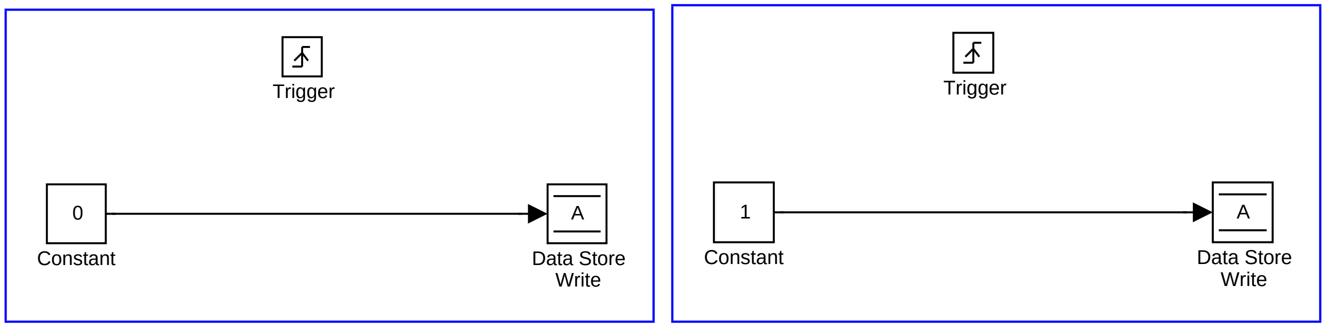




Same problem: when the triggers are non exclusive the execution order depends on some strange features. We can get . . .

Imperative Features: Memory Block





Same problem: when the triggers are non exclusive the execution order depends on some strange features. Or we can get. . .

Monitoring Properties

Simulink can also be used for monitoring properties of designs:

- testing
- feeding a model-checker or a theorem prover
- on-line monitoring

The idea is to design monitors as property observers: subsystems that output a signal carrying the “true value” of the property: the signal is initially true, stays true as long as the property is fulfilled, becomes false as soon as the property becomes false and then stays false for ever.

This is due to the fact that discrete-time Simulink is equivalent to a temporal logic of the past (allows modelling only “safety properties”)

Monitoring Properties: Example

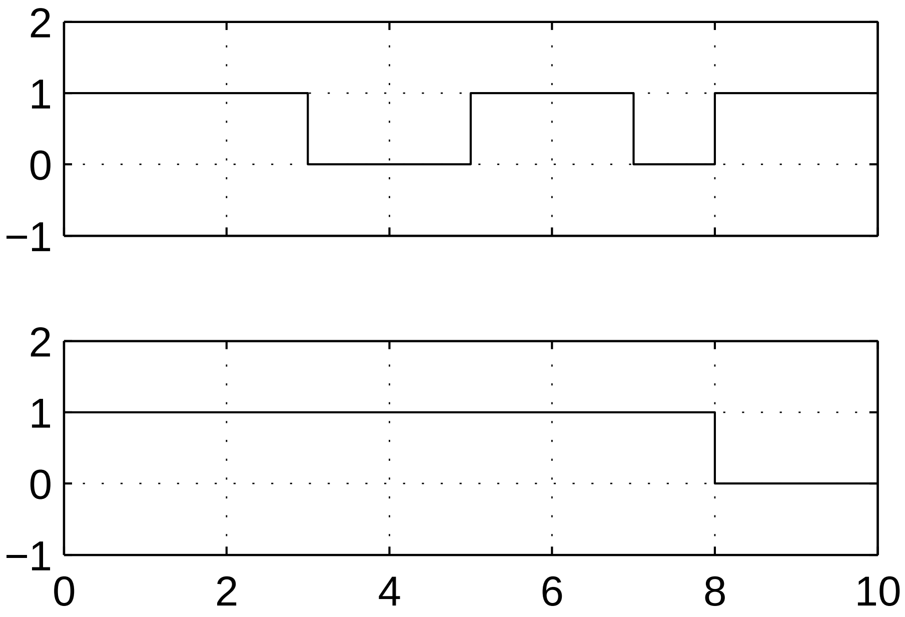
Design a property observer that monitors the property:

a signal doesn't change its value in two consecutive samples

Typical bevaviour:

signal

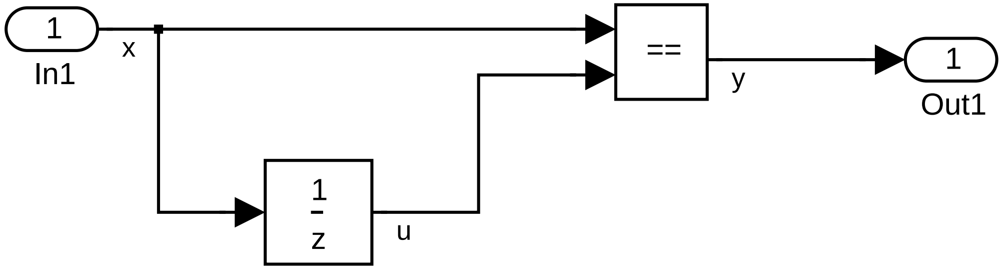
observer output



Time offset: 0

Monitoring Properties: Solution

The “stable” subsystem checks whether the current sample and the previous one are equal

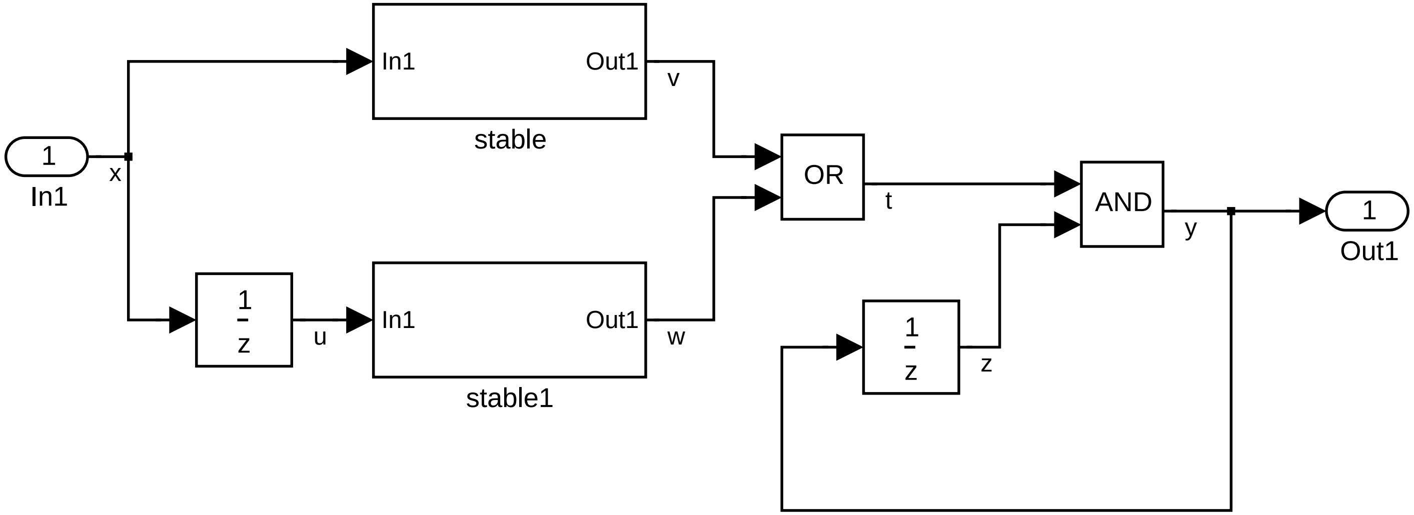


Monitoring Properties: Solution

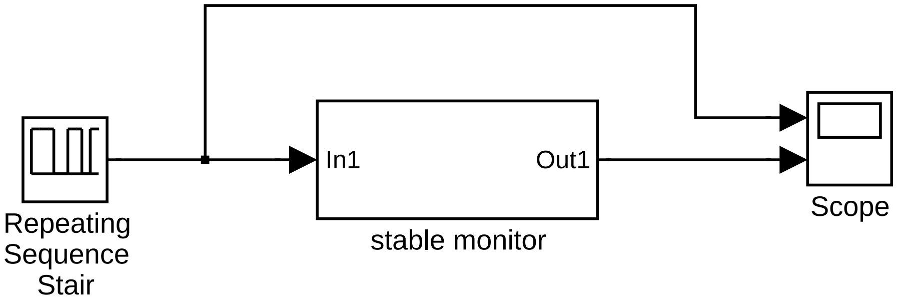
We cannot ask the signal to be always stable because it wouldn't be allowed to change.

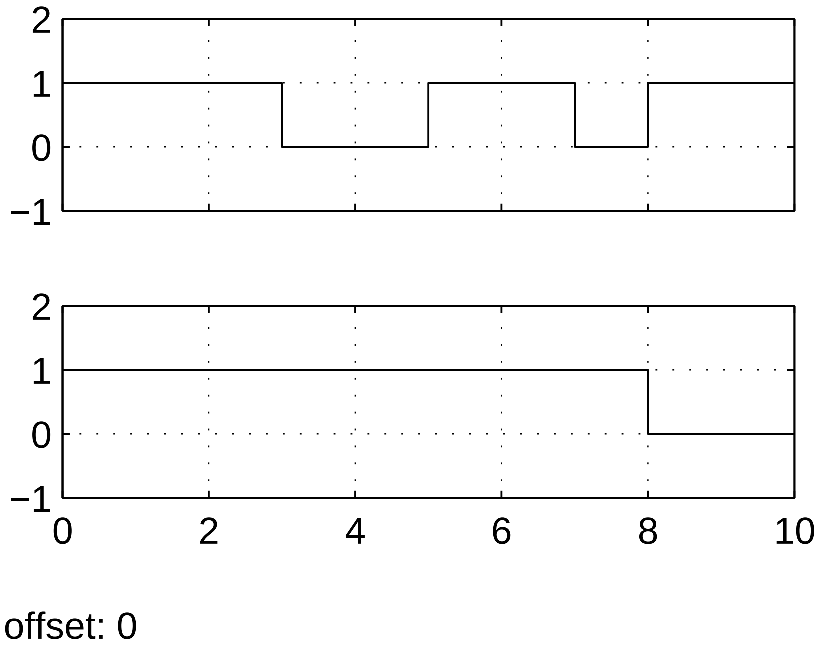
But if it is not stable now, at least it should have been stable at the previous instant.

Finally, when the property becomes false, the observer output should stay false for ever.



Monitoring Properties: Checking the Solution



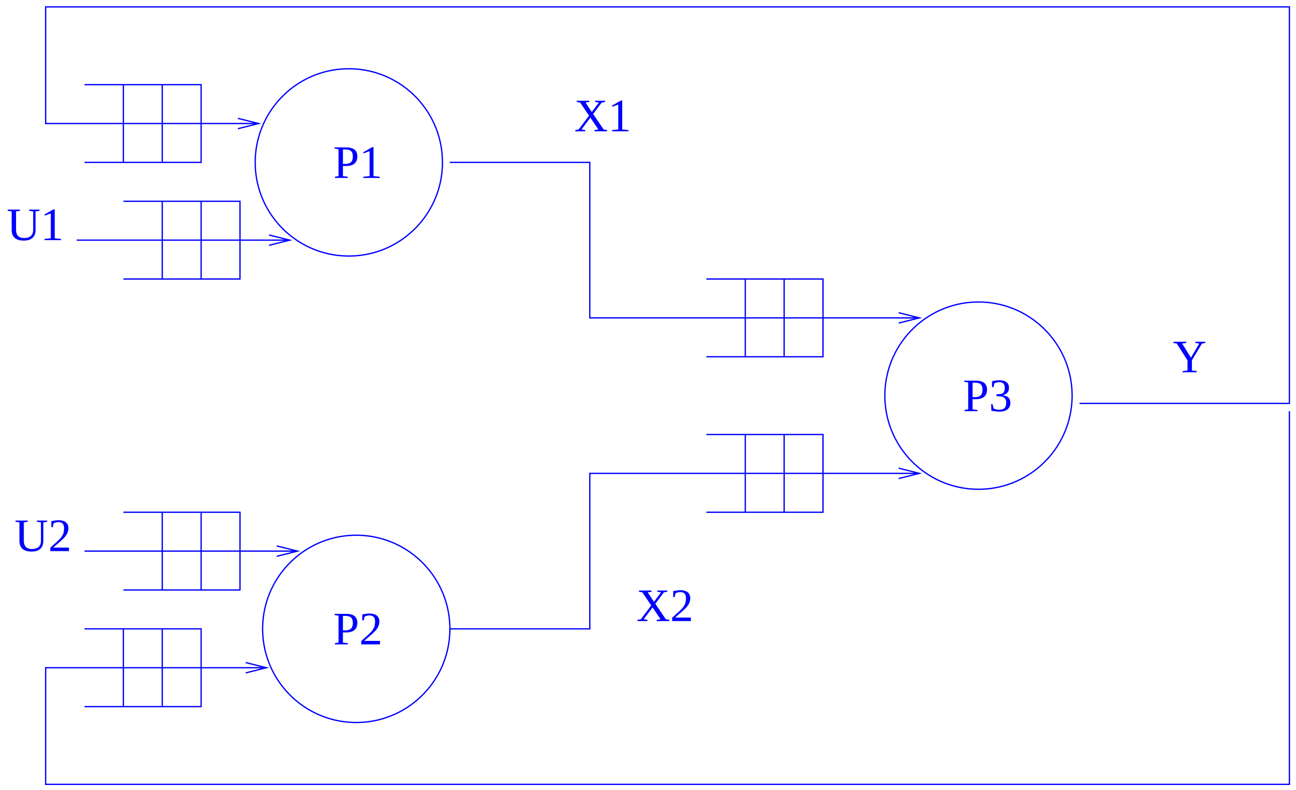


Simulink and Kahn Networks

- Kahn networks
- Their semantics
- Simulink as Kahn networks

Kahn Networks

Process networks *deterministic but asynchronous* communicating through FIFOs (Unix sockets for instance)



FIFO: reading blocks on empty queues

Kahn Semantics

An observer records the history of every queuee: $X = x_0, x_1, \ldots$

Processes are deterministic : $X_1 = P_1(Y, U_1)$

The system computes a solution of the system of equations:

$$\begin{aligned} X_1 &= P_1(Y, U_1) \\ X_2 &= P_2(Y, U_2) \\ Y &= P_3(X_1, X_2) \end{aligned}$$

What does it mean?

Domains

An history belongs to a domain D: $D^\infty = D^* + D^\omega$ (finite or infinite sequences) endowed with a partial order$\leq$ (prefix order)

$$x \leq y \Leftrightarrow \exists z : y = x@z$$

- minimum element ϵ ;

$$\forall x \in D^\infty : \epsilon \leq x$$

- any chain $C = \{x_0 \leq x_1 \leq \dots x_n \leq \dots\}$ has a least upper bound

$$\forall x \in C : x \leq \sup\ C$$

$$\forall y \in D^\infty : (\forall x \in C : x \leq y) \Rightarrow \sup\ C \leq y$$

$\Rightarrow (D^\infty, \leq, \epsilon)$ is a Complete Partial Order (CPO)

Least Fixpoints

- a function $f : D^\infty \Rightarrow D^\infty$ is continuous if for every chain $C = x_0 \leq x_1 \leq \ldots x_n \ldots$

$$f(\sup\ C) = \sup\{f(x_i)|x_i \in C\}$$

[continuous $\Rightarrow$ monotonic $: x \leq y \Rightarrow f(x) \leq f(y)$]

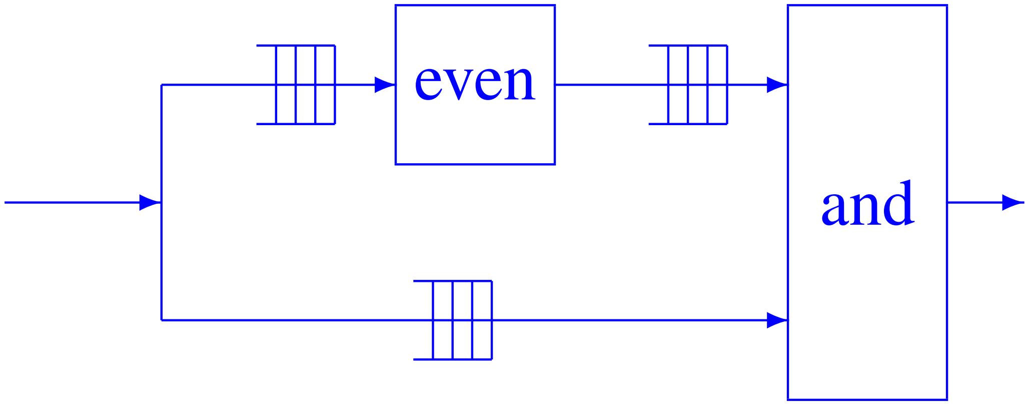
$\Rightarrow$ an equation $x = f(x)$ has a least solution (least fixpoint) and this can be extended to any order:

$$\mu f = \sup\{\epsilon \leq f(\epsilon) \leq \ldots f^n(\epsilon) \leq \ldots\}$$

Kahn claimed that this was what the network would compute!

And Simulink?

Simulink doesn’t allow bad behaved networks such that this one



whose queues will eventually overflow

Thus Simulink networks can also be executed asynchronously by connecting them via FIFOs.

But some care has to be taken when generating code for each process. (Benveniste’s work)

Conclusion about Simulink

- Allows modelling both controllers, environments (plants) and properties
- Mostly based on sound mathematical bases
- Provides means for parallelisation
- Some dangerous error-prone features which have to be handled with much care.