System Design and Methodology / Embedded Systems Design

V. Discrete Event Models

TDTS07/TDDI08
VT 2026

Ahmed Rezine

(Based on material by Petru Eles and Soheil Samii)

Institutionen för datavetenskap (IDA)
Linköpings universitet

DISCRETE EVENT MODELS

1. What Is a Discrete Event Model?

2. Discrete Event Simulation

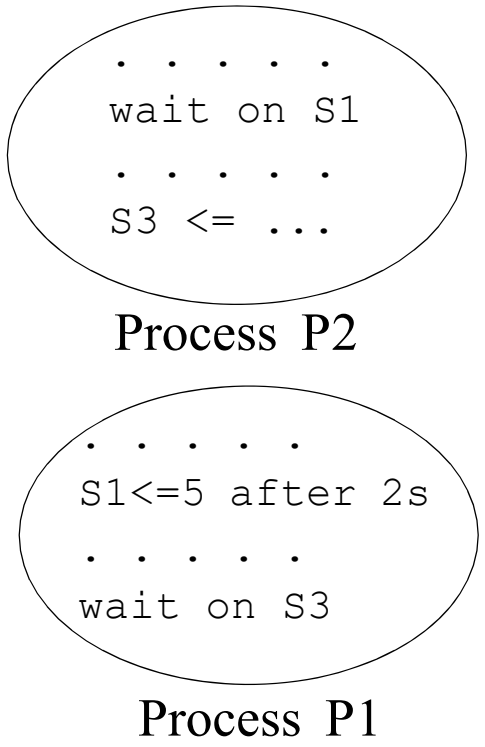
3. Efficiency of Discrete Event Simulation

4. Potential Ambiguities in Discrete Event Simulation

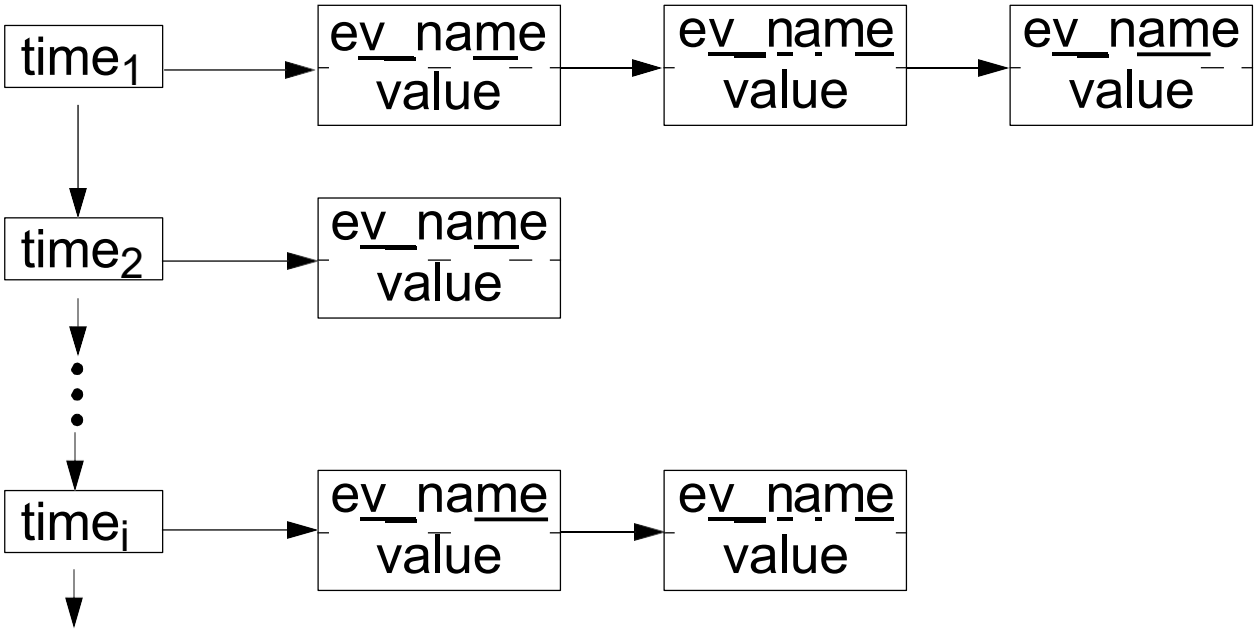
Discrete Event Models

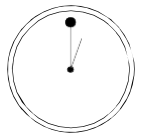
- The system is a collection of processes that respond to events.
- *Each event carries a time-stamp indicating the time at which the event occurs*.
- *Time-stamps are totally ordered*.
- A Discrete Event (DE) simulator maintains a *global event queue sorted by the time-stamps*. The simulator also keeps a single global time.

Discrete Event Simulator



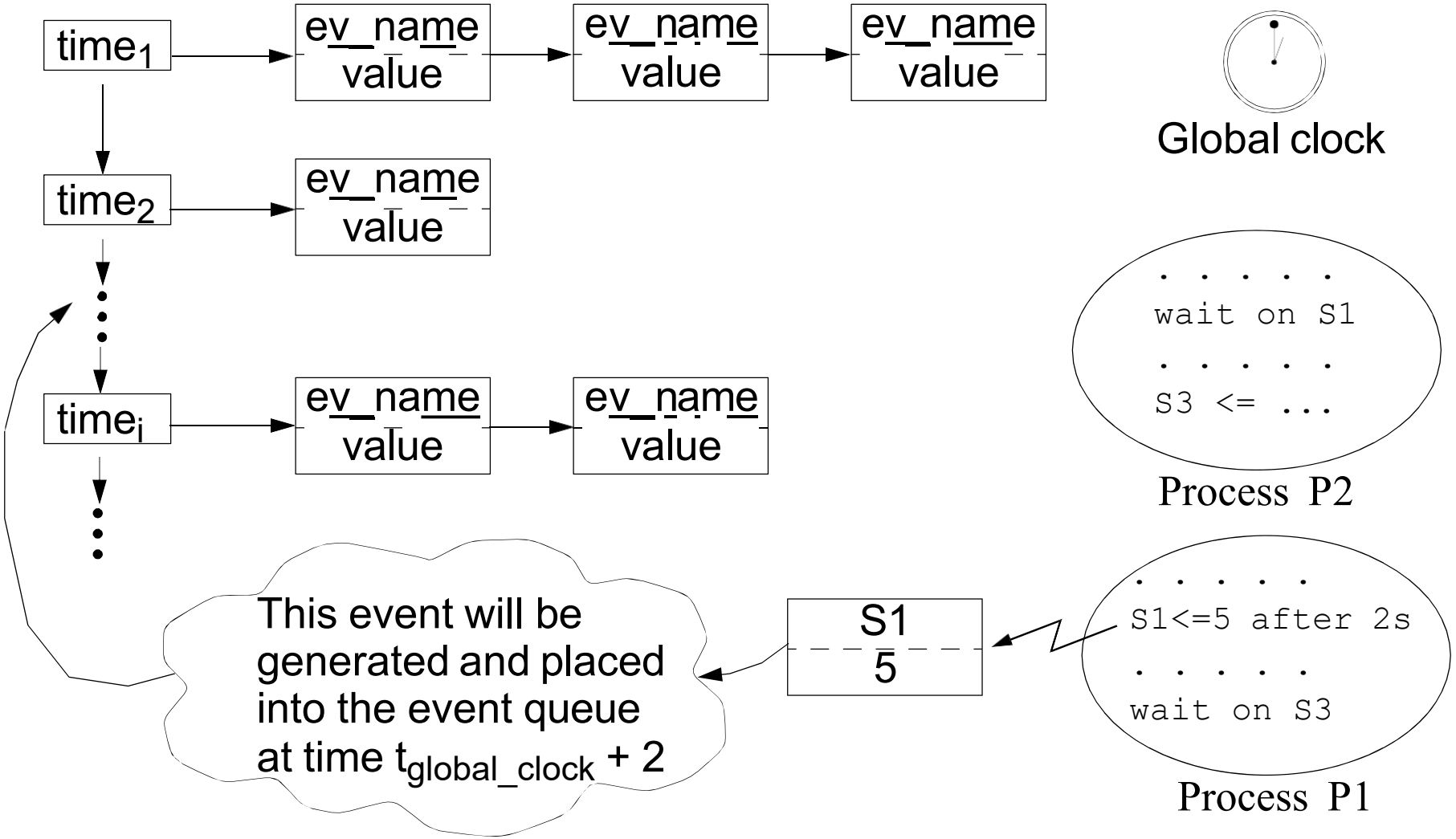
Discrete Event Simulator



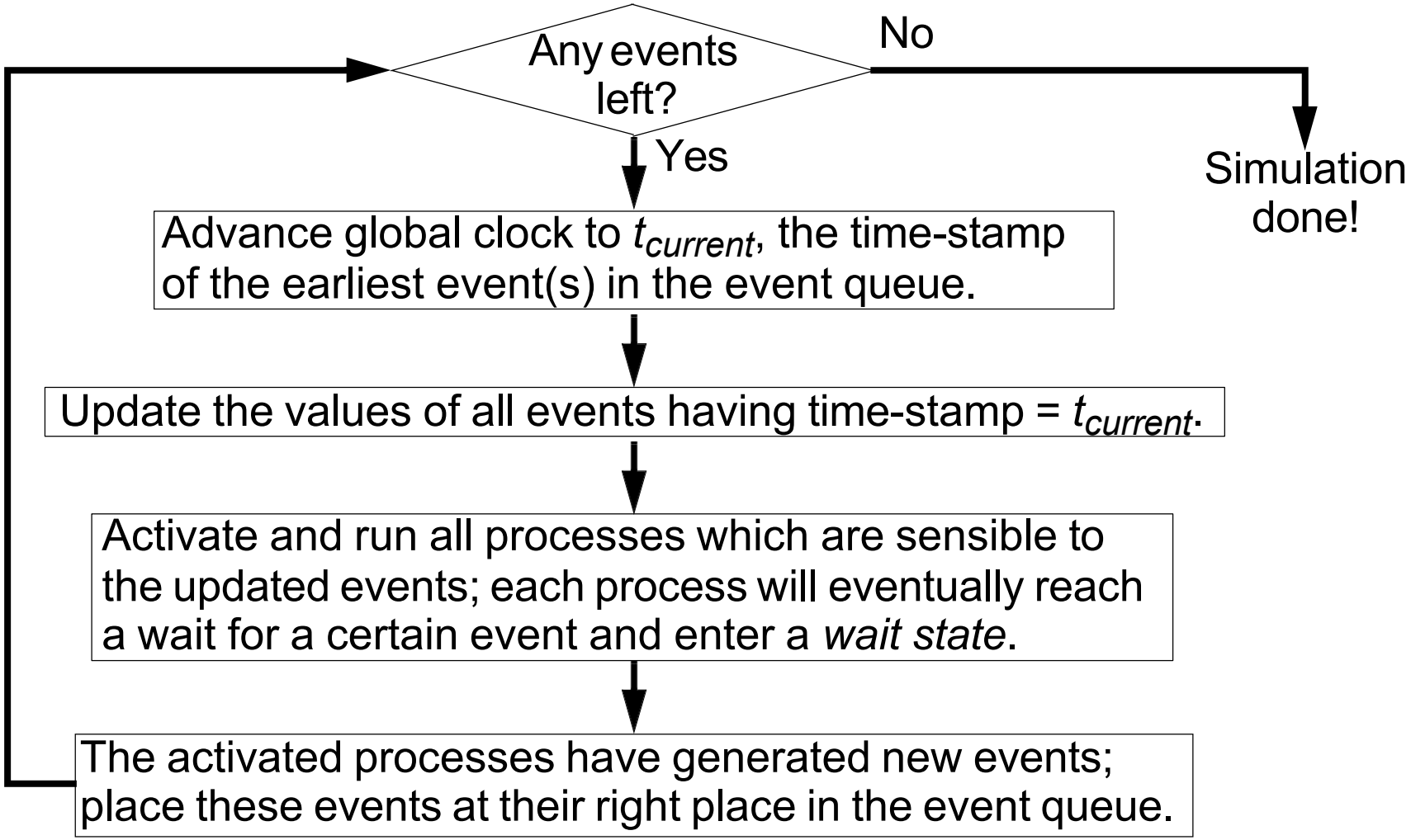


Global clock

Discrete Event Simulator



Discrete Event Simulator



Discrete Event Simulation

- The discrete event model has been mainly used for system simulation.
  - Several languages have been developed for system modeling based on the discrete event model. Most well known:
    - VHDL, Verilog (both used for hardware modeling), SystemC

- Efficient way to simulate distributed systems.

  In general, efficient for large systems with autonomous components, with relatively large idle times. Systems with non-regular, possibly long times between different activities.

  Why is this the case?

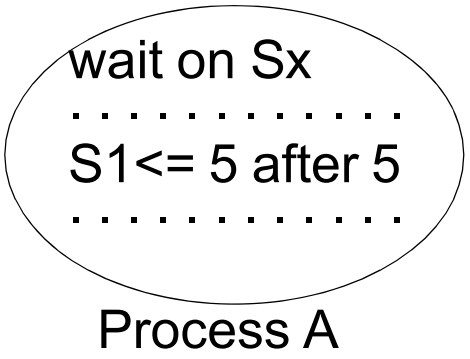
  Because DE simulation will only consider the particular times when a change in the system (an event) occurs. This is opposed to, for example, cycle-based models, where *all clock-ticks* are considered.

Discrete Event Simulation

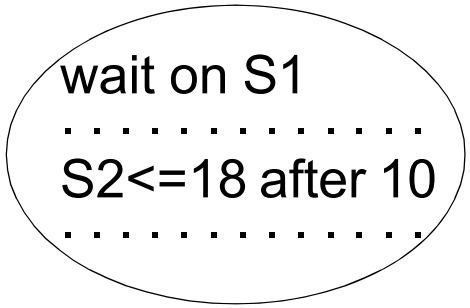
- Event driven models are primarily employed for simulation.
  - Functional verification
  - Performance evaluation

- Both synthesis and formal verification are very complex with DE models.
  - The classical trade-off between expressive power and the possibility of formal reasoning and efficient synthesis.

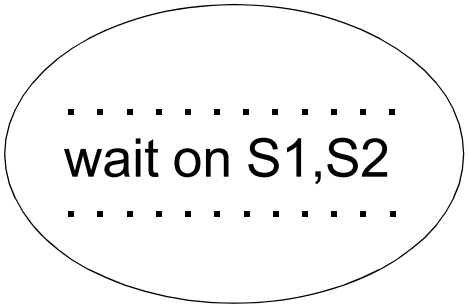
An Example



Process A



Process B

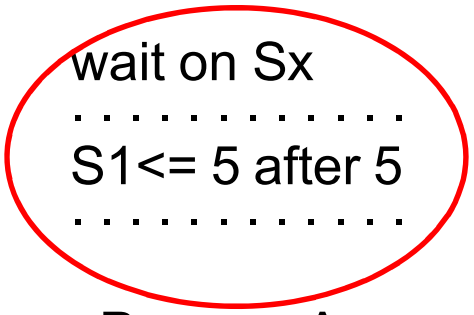


Process C

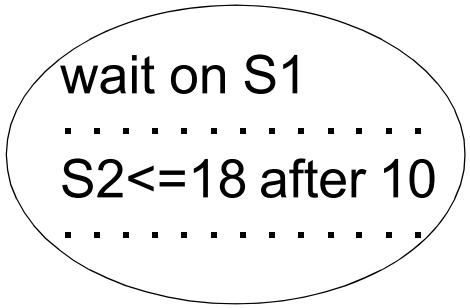
S1	S2
100	0

Initial values

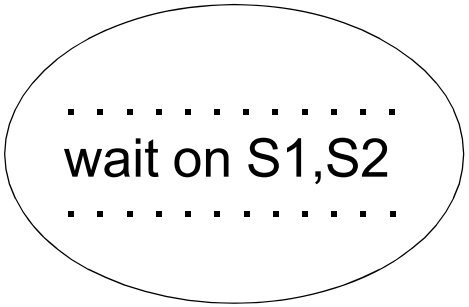
An Example



Process A

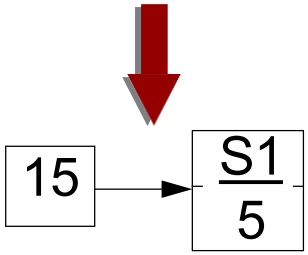


Process B

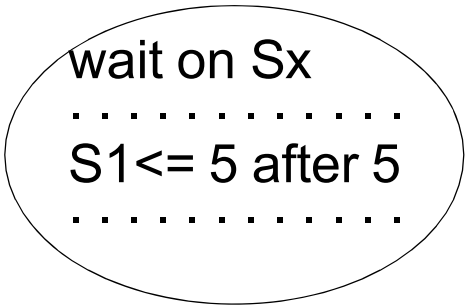


Process C

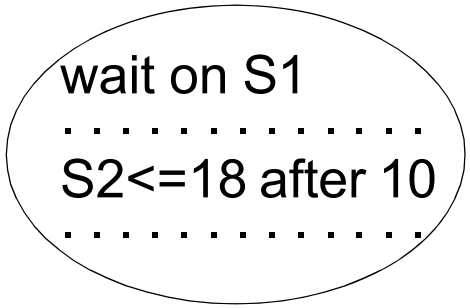
At time 10 Process A executes



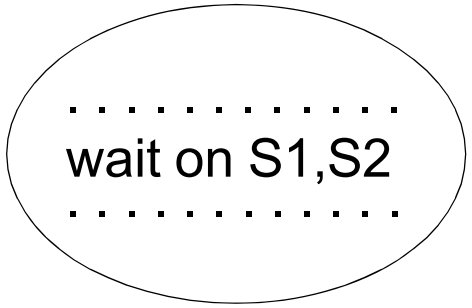
An Example



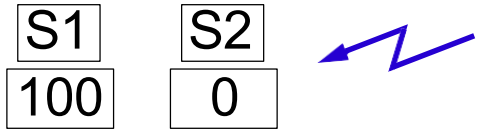
Process A



Process B

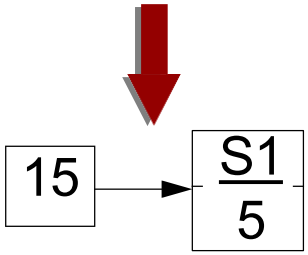


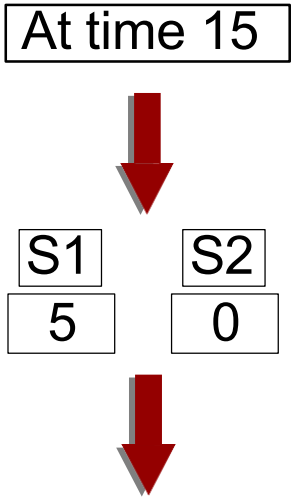
Process C



Initial values

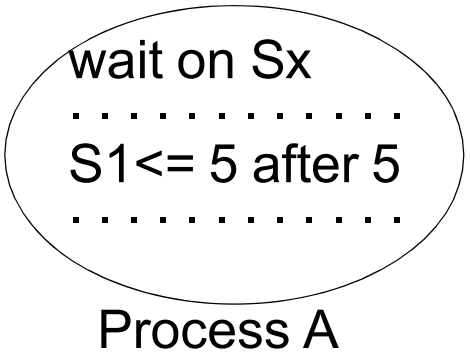
At time 10 Process A executes

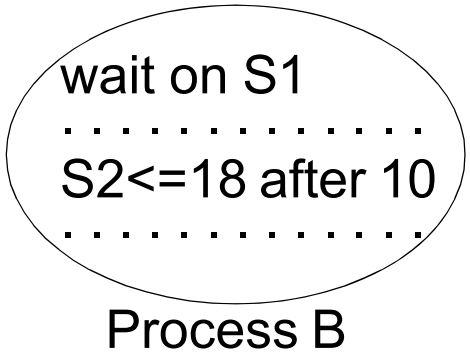


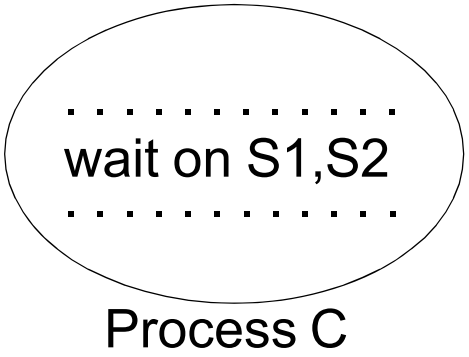


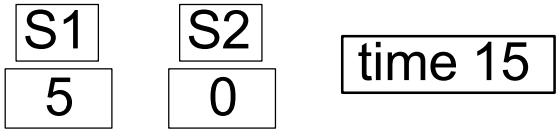
Processes B and C are ready to execute

An Example

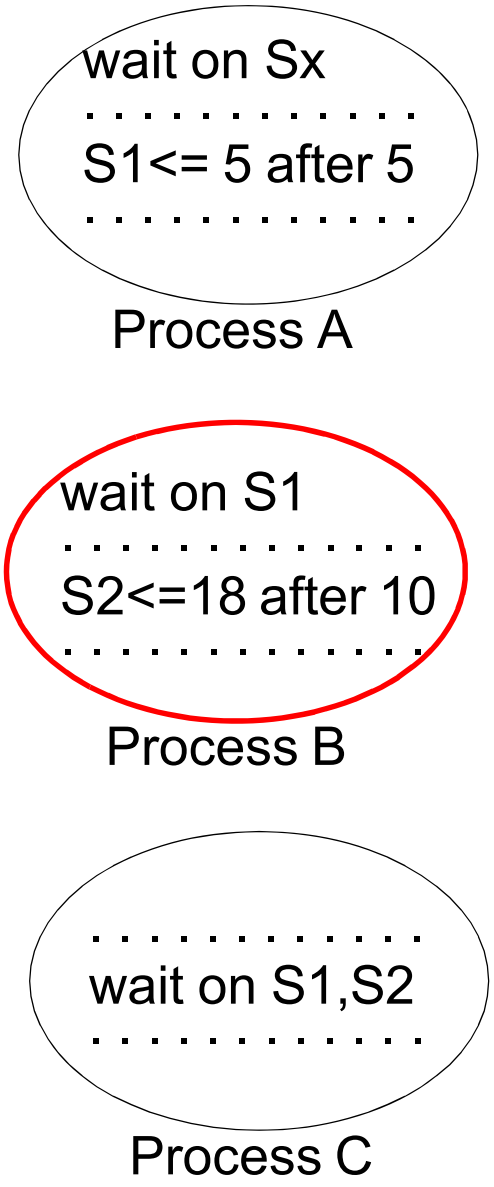

wait on Sx
S1<= 5 after 5
Process A

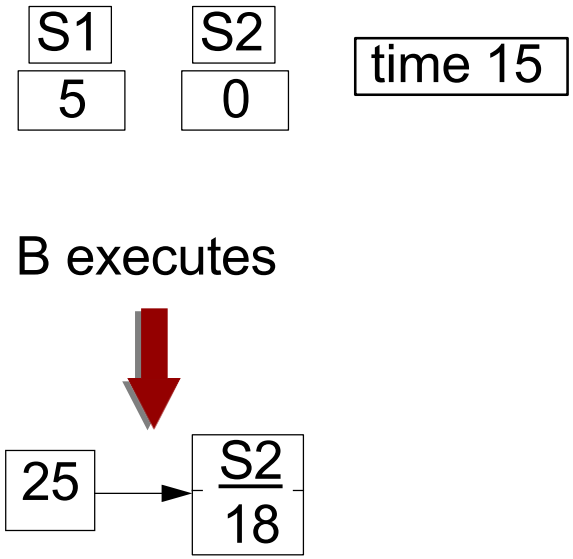

wait on S1
S2<=18 after 10
Process B


wait on S1,S2
Process C

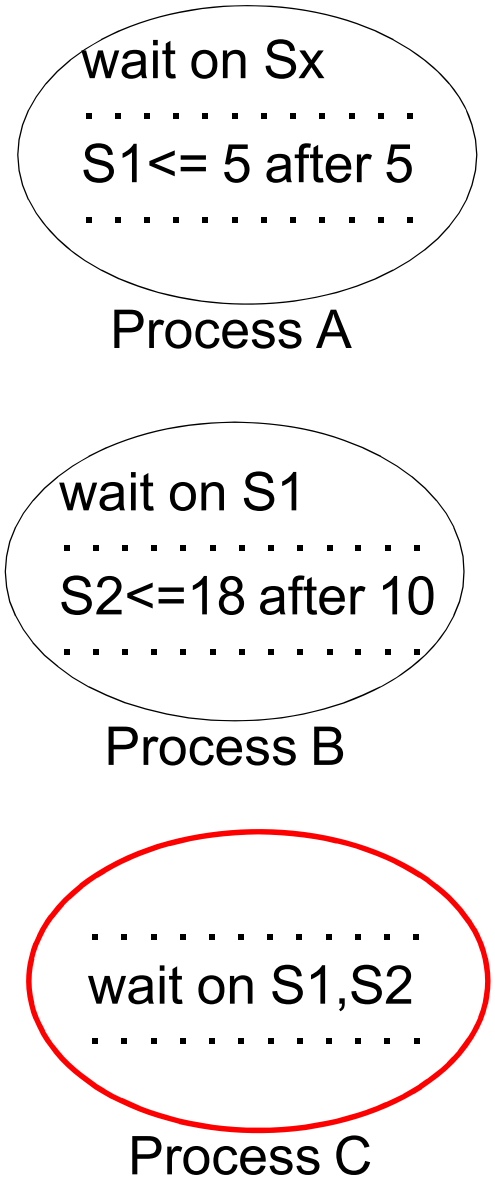

S1
5
S2
0
time 15

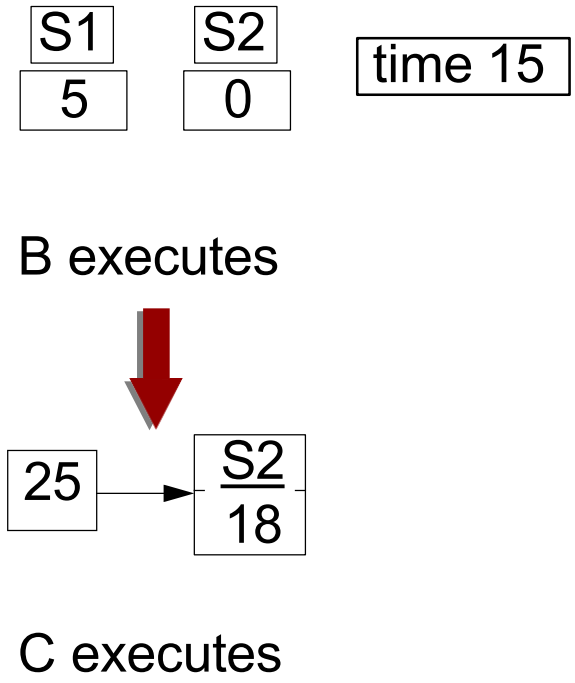
An Example


wait on Sx
S1<= 5 after 5
Process A
wait on S1
S2<=18 after 10
Process B
wait on S1,S2
Process C

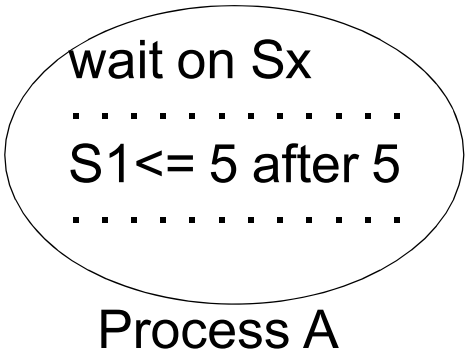

S1
5
S2
0
time 15
B executes
25
S2
18

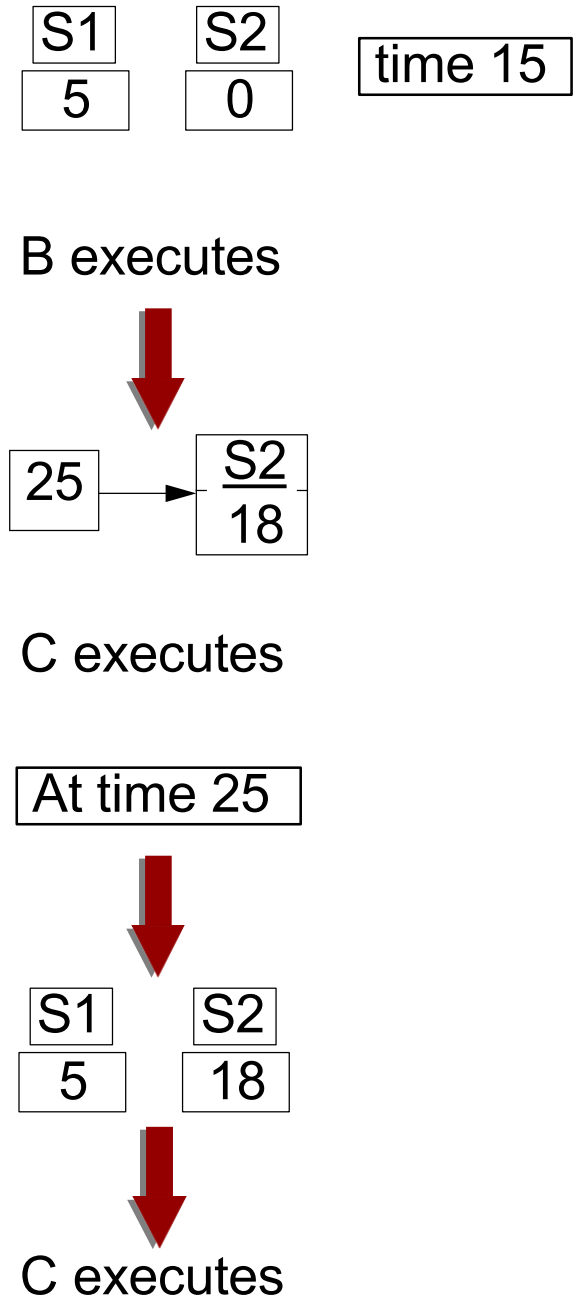
An Example


wait on Sx
S1<= 5 after 5
Process A
wait on S1
S2<=18 after 10
Process B
wait on S1,S2
Process C


S1
5
S2
0
time 15
B executes
25
S2
18
C executes

An Example


wait on Sx
S1<= 5 after 5
Process A
wait on S1
S2<=18 after 10
Process B
wait on S1,S2
Process C

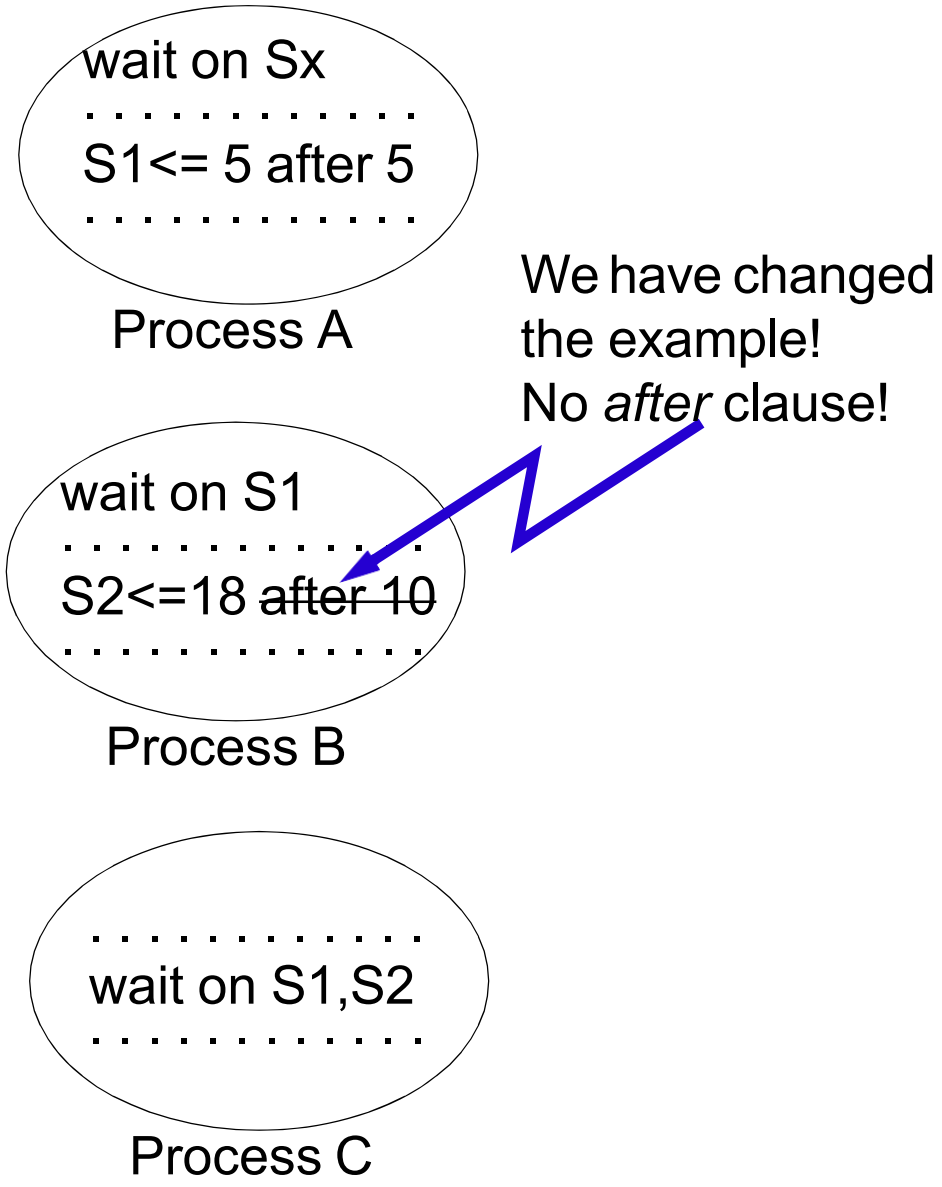

S1
5
S2
0
time 15
B executes
25
S2
18
C executes
At time 25
S1
5
S2
18
C executes

Delta Delay/Delta Cycles

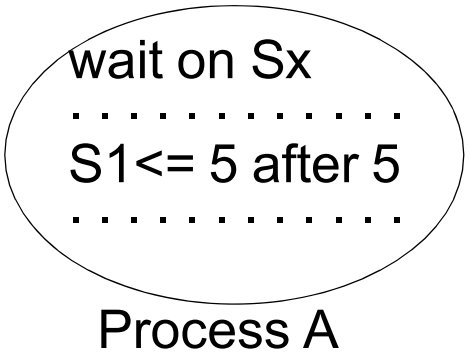
- A zero delay event will be registered at a time which is infinitesimally delayed relative to the current time.

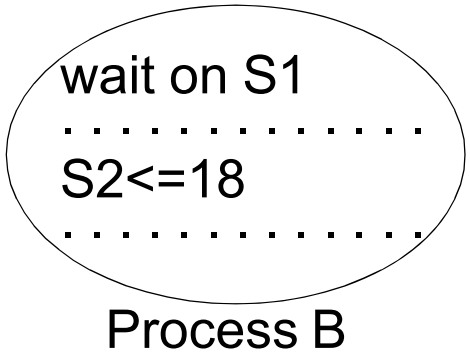
A *delta delay* will be introduced on the event $\Rightarrow$ the new event will be consumed in the following simulation cycle and not the current one.

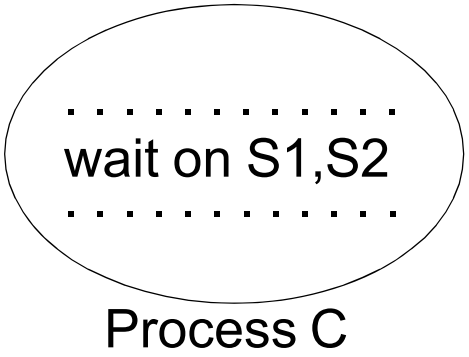
An Example


wait on Sx
S1<= 5 after 5
Process A
We have changed
the example!
No *after* clause!
wait on S1
S2<=18 ~~after 10~~
Process B
wait on S1,S2
Process C

An Example

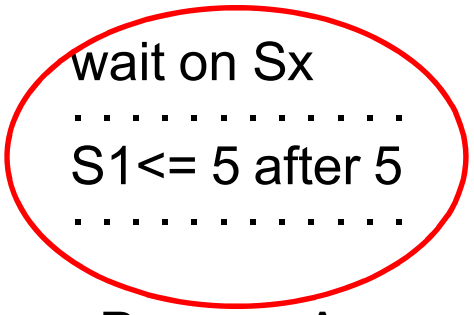

wait on Sx
S1<= 5 after 5
Process A


wait on S1
S2<=18
Process B

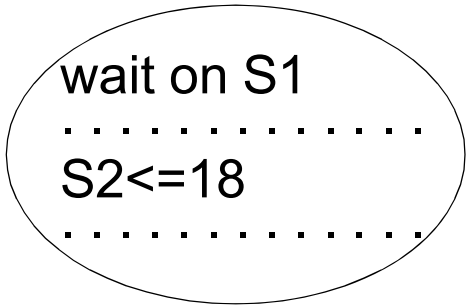

wait on S1,S2
Process C


S1
S2
100
0
Initial values

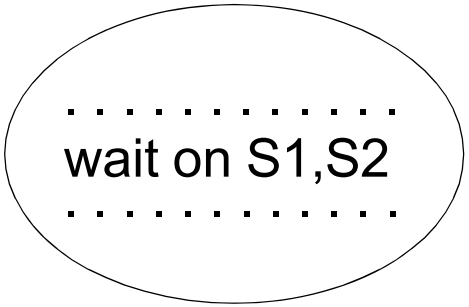
An Example



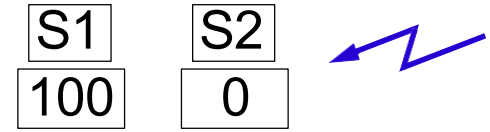
Process A



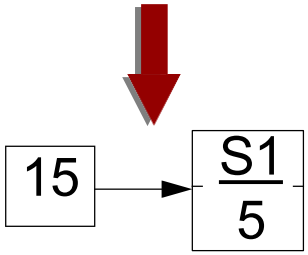
Process B



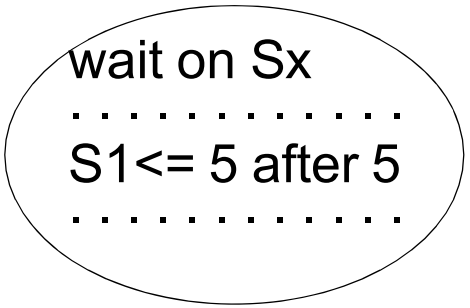
Process C



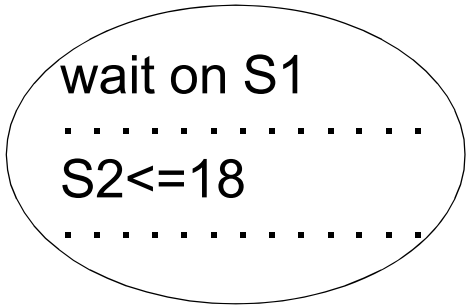
At time 10 Process A executes



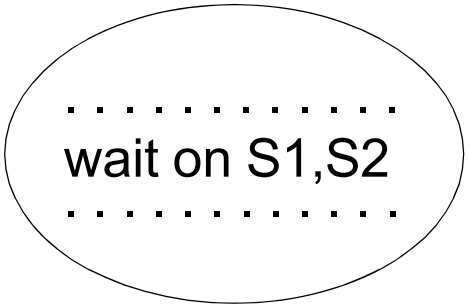
An Example



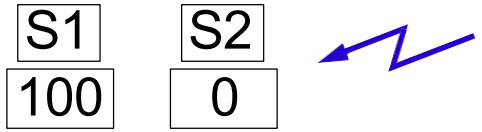
Process A



Process B

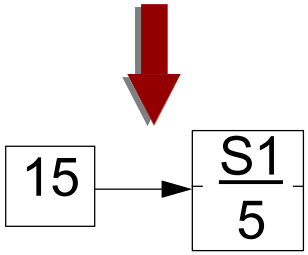


Process C

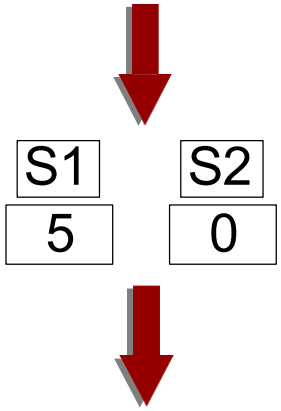


Initial values

At time 10 Process A executes

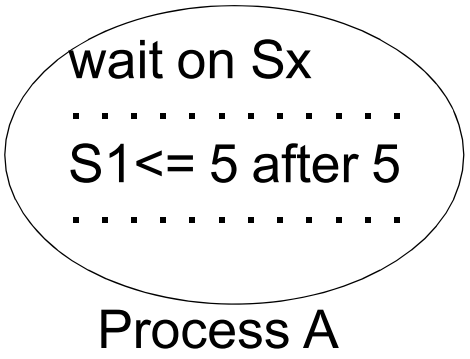


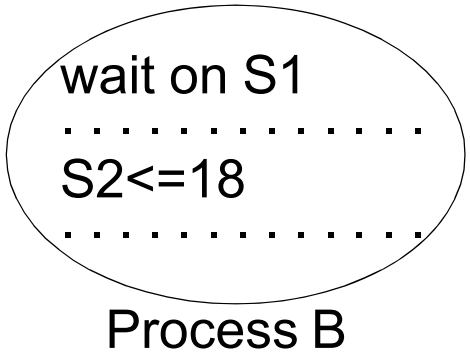
At time 15

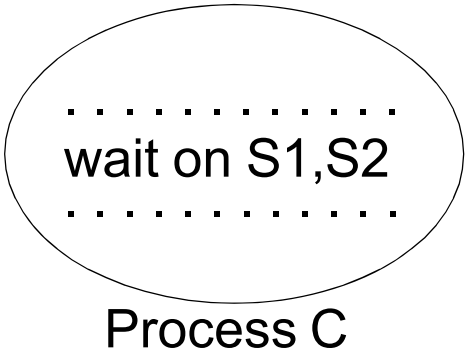


Processes B and C are ready to execute

An Example

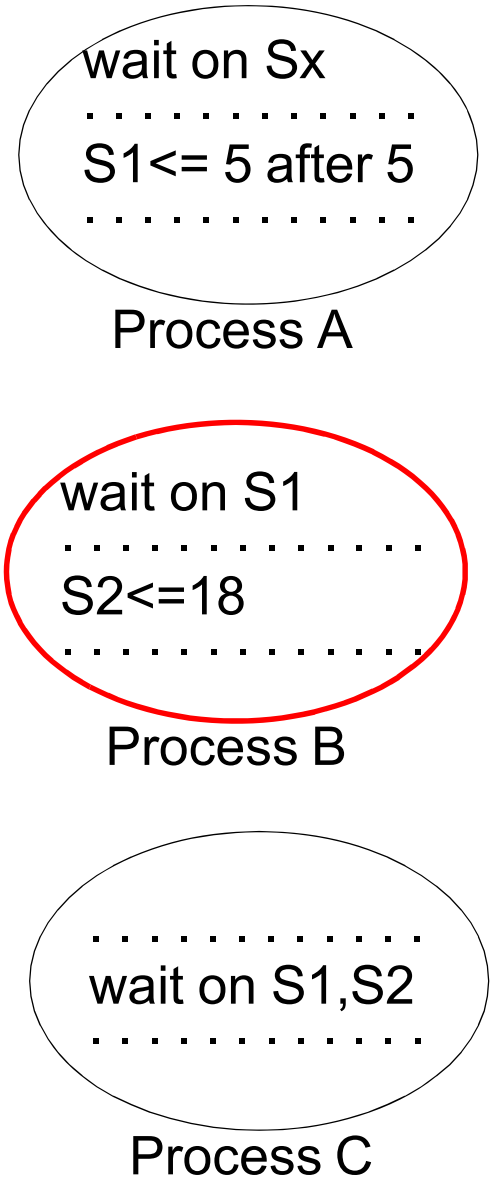


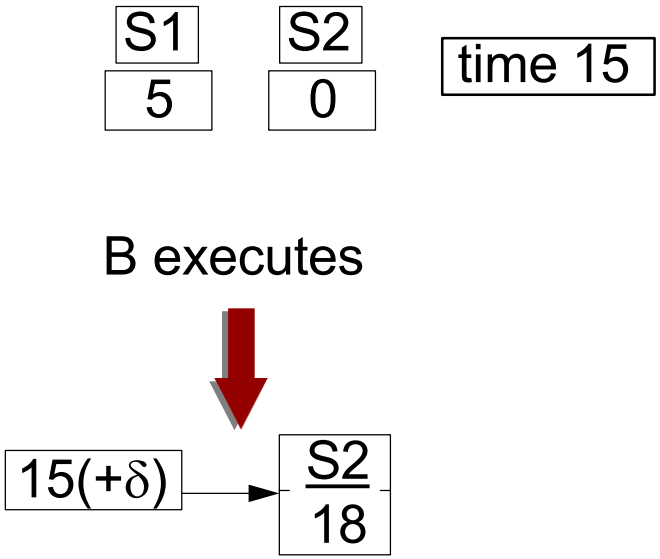




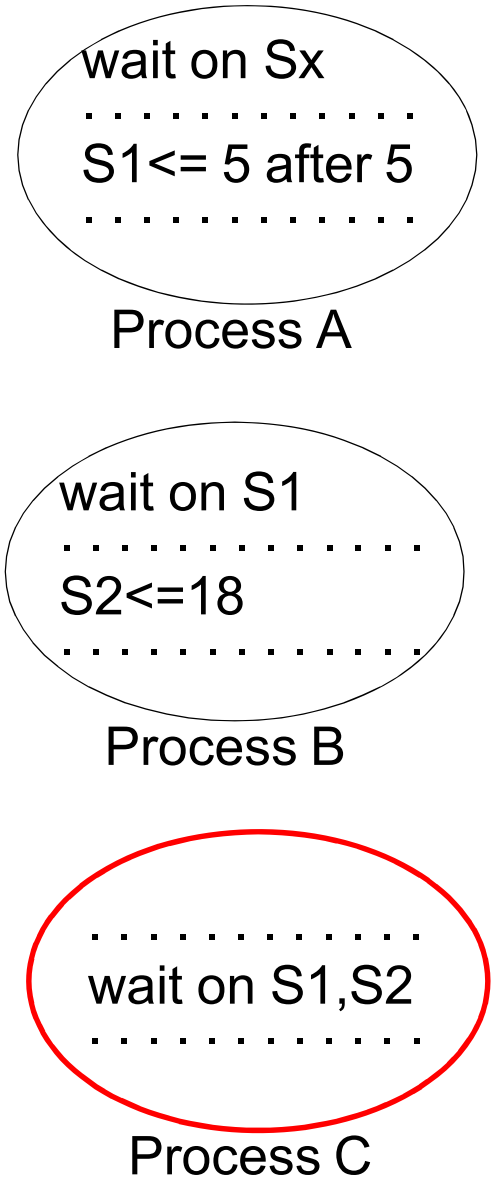
S1	S2	time 15
5	0	

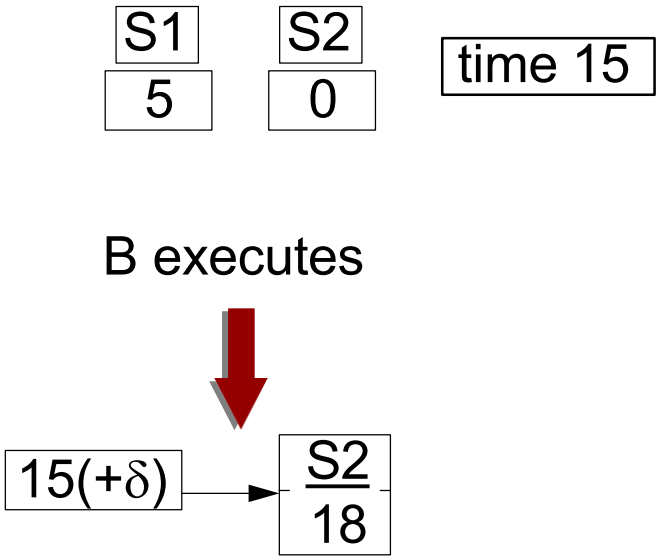
An Example


wait on Sx
S1<= 5 after 5
Process A
wait on S1
S2<=18
Process B
wait on S1,S2
Process C

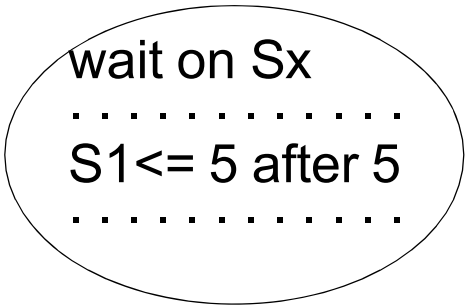

S1
5
S2
0
time 15
B executes
15(+δ)
S2
18

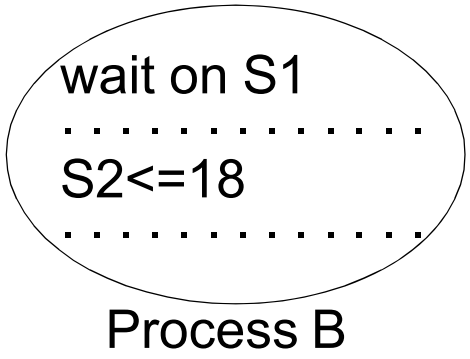
An Example

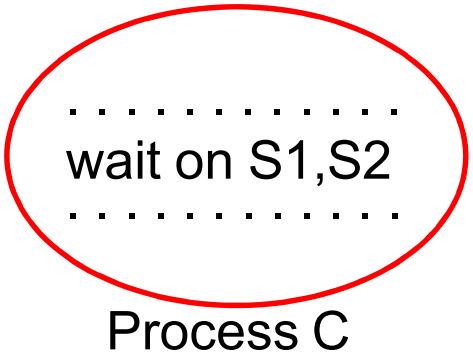

wait on Sx
S1<= 5 after 5
Process A
wait on S1
S2<=18
Process B
wait on S1,S2
Process C

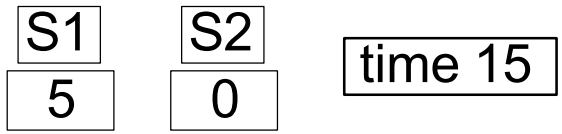

S1
5
S2
0
time 15
B executes
15(+δ)
S2
18
C executes

An Example

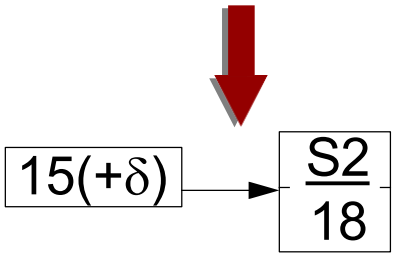






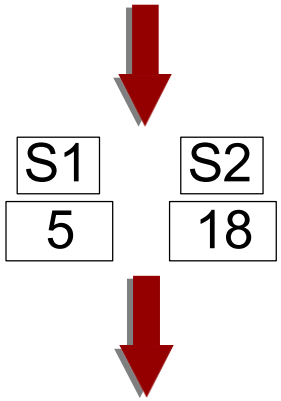


B executes



C executes

In the following simulation cycle: time 15(+δ)



C executes