Introduction to SDL

TSP

Stéphane Maag

TELECOM SudParis

Objectives

This course intends to make the participants discover:

- SDL (with MSC) notations
- Use of SDL'88
- Edit on RTDS

Specification Description Language

Outline

- SDL, a FDT for complex system specification
- MSC to SDL
- SDL system
- SDL notations
- SDL process
- From the specification to the simulation
- RTDS

... and conclusion.

SDL - a Formal Description Technique

- FDTs (also called *specification language*):
  - specify the functional properties of a system according to its environment
  - are conceived to describe distributed systems composed by processes that are executed in parallel, synchronize themselves and communicate by messages

- Other techniques: process algebra (CCS), finite state machines, temporal logic, Petri networks, ...

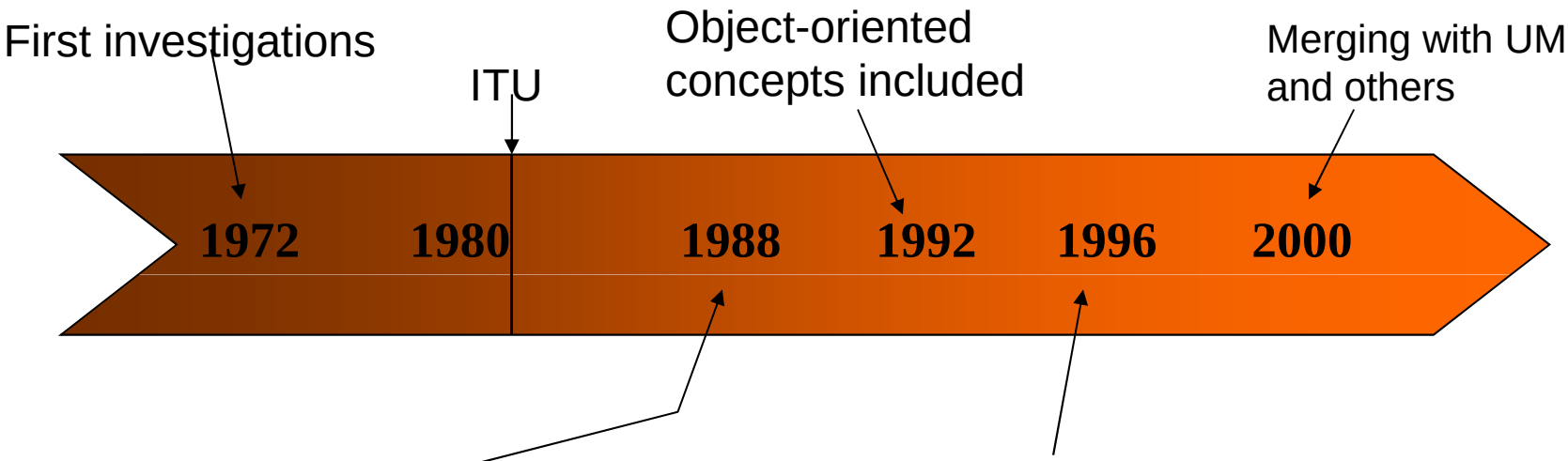
Briefly, SDL

- SDL (Specification Description Language):
  - Define and normalized by ITU(-T) (1988, 1992, 1996, 2000)
  - based on the Extended Finite State Machines (EFSM), asynchronous
  - 2 visions: SDL-GR (graphical) and SDL-PR (textual)
  - Abstract data types, ASN.1

Let's go with SDL … in details …

- To specify, to describe without ambiguities telecommunication systems
- To represent functional properties of a system:
  - structural properties: system architecture, its decomposition into interconnected functional blocks
  - behavioral properties: system reactions after stimuli coming from the environment
- The architecture ≠ The behavior

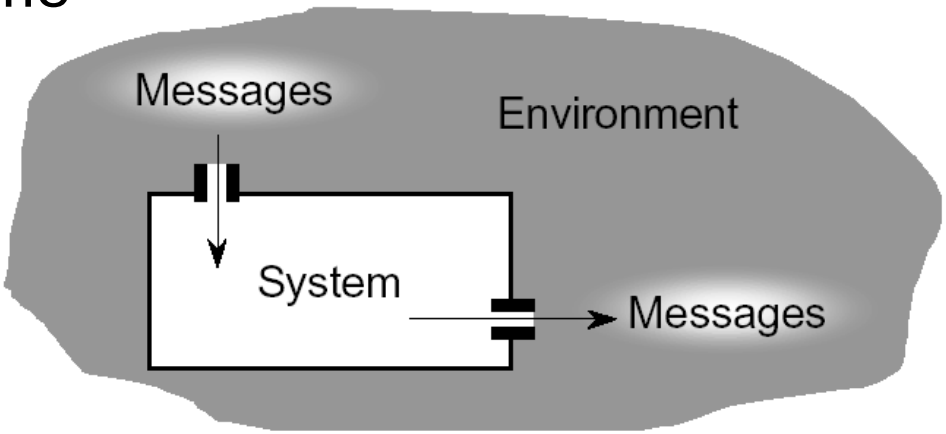
History


First investigations
ITU
Object-oriented concepts included
Merging with UML and others
1972
1980
1988
1992
1996
2000
first tool supported version,
hierarchical description of the structure,
description of the communication and behaviors (EFSM),
formal semantic,
basic data.
Few modifications

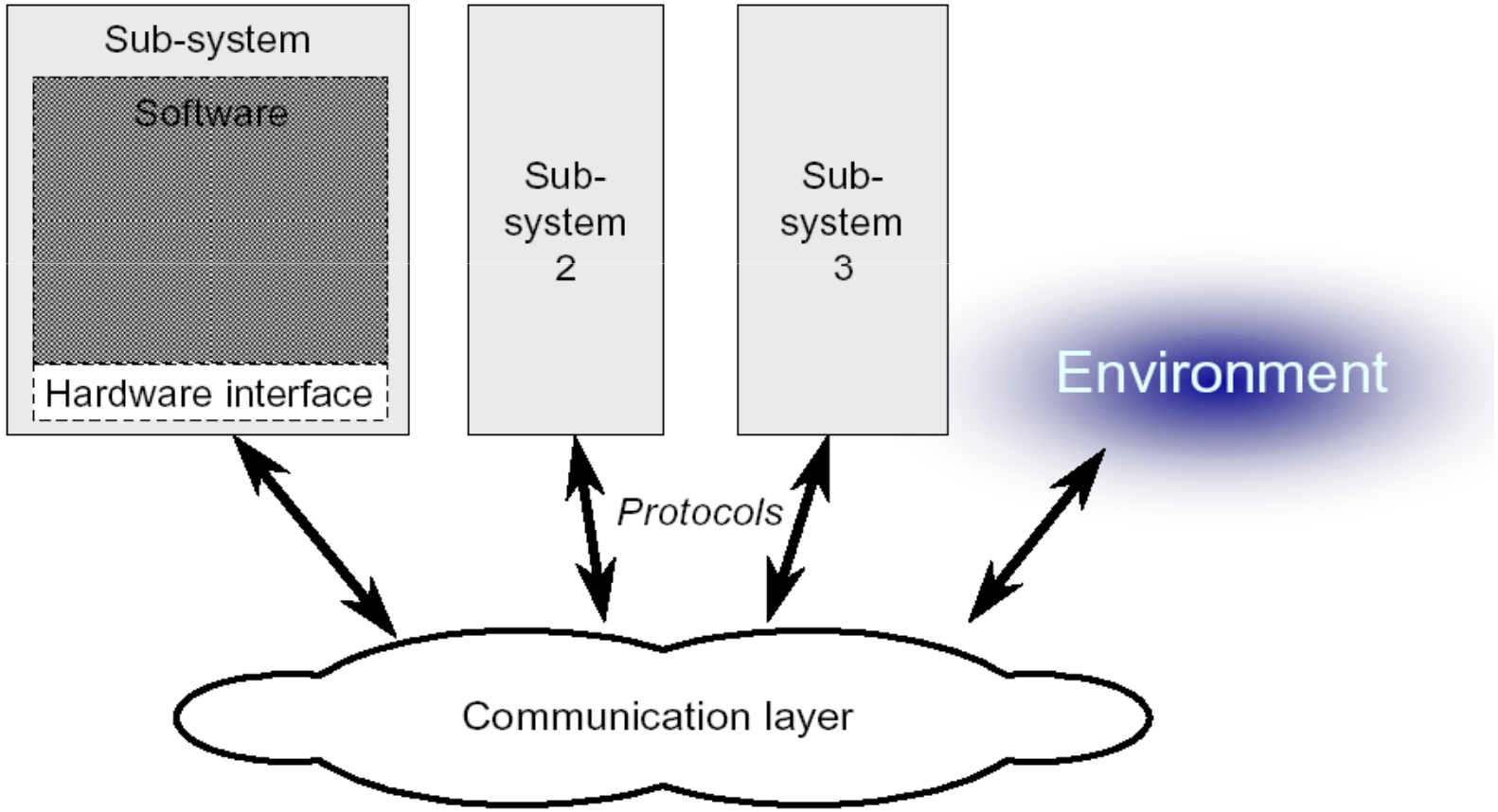
What is a Real-time system ?

A software-based system:

- Capable of interacting with its environment...
- According to a response time compatible with the dynamics of the environment
- In order to supervise, to command or to communicate with the environment at any time



Distributed System



SDL for Reactive and Discrete Systems

- Communication:
  - Message exchanges between the system and its environment
  - Mainly asynchronous interactions, but synchronous ones also supported

- Nevertheless:
  - SDL is not adapted to cyclic data-driven inputs
  - SDL is unable to describe non real-time aspects, such as:
    - Data bases
    - GUIs

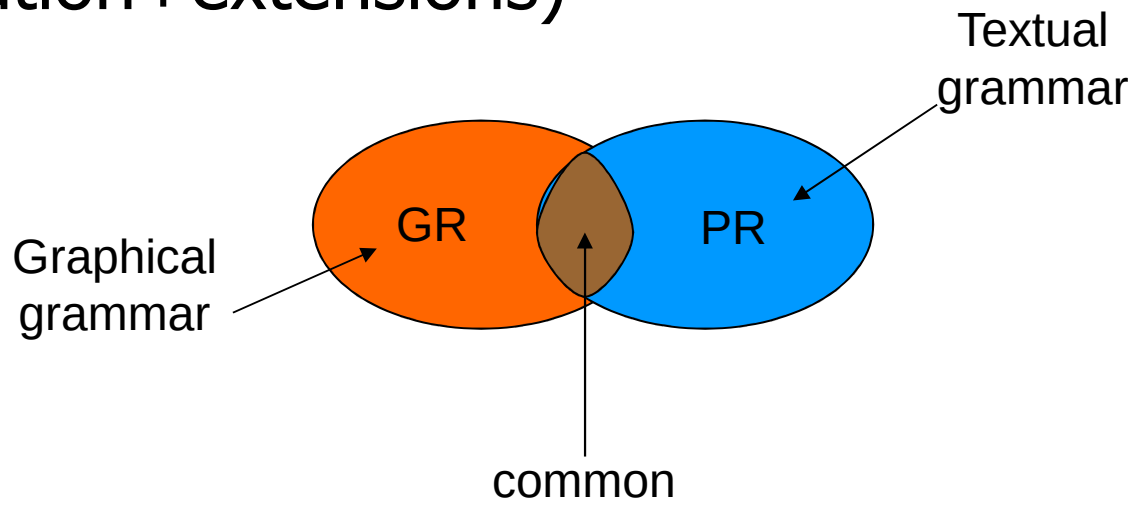
SDL applications

- Wide range of applications
  - safety and mission critical communicating systems
  - real-time applications

- Wide range of architectures
  - workstation-based distributed system, 32-bits communication board, 8-bits micro-controller embedded system

SDL
Two normalized representations

- Graphical representation: GR
- Textual representation: PR
- Exchange format: PR+CIF (information+extensions)



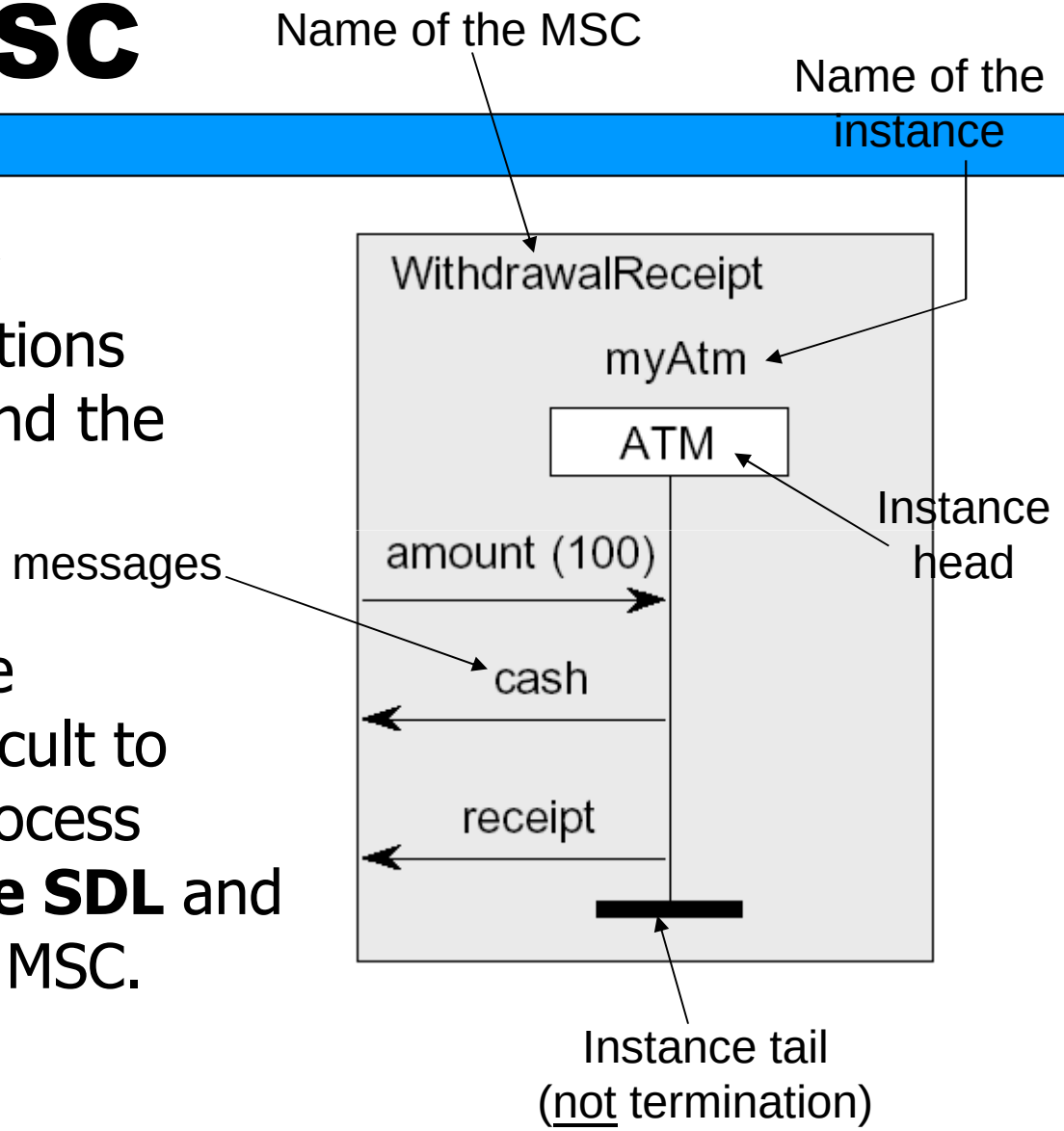
MSC - to provide the behaviors

Message Sequence Chart

- Z.120 Recommendation managed by the ITU
- "*is to provide a trace language for the specification and description of the communication behavior of system components and their environment by means of message interchange*"

SDL with MSC

- To describe cases by sequences of interactions between instances and the environment
- allows to observe the interactions, but difficult to assign values and process operations … **we use SDL** and we may control with MSC.



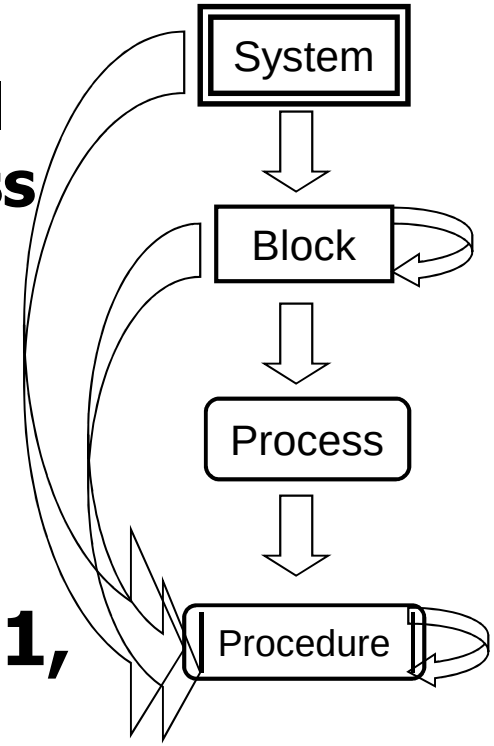
System specification

Three aspects in order to specify:

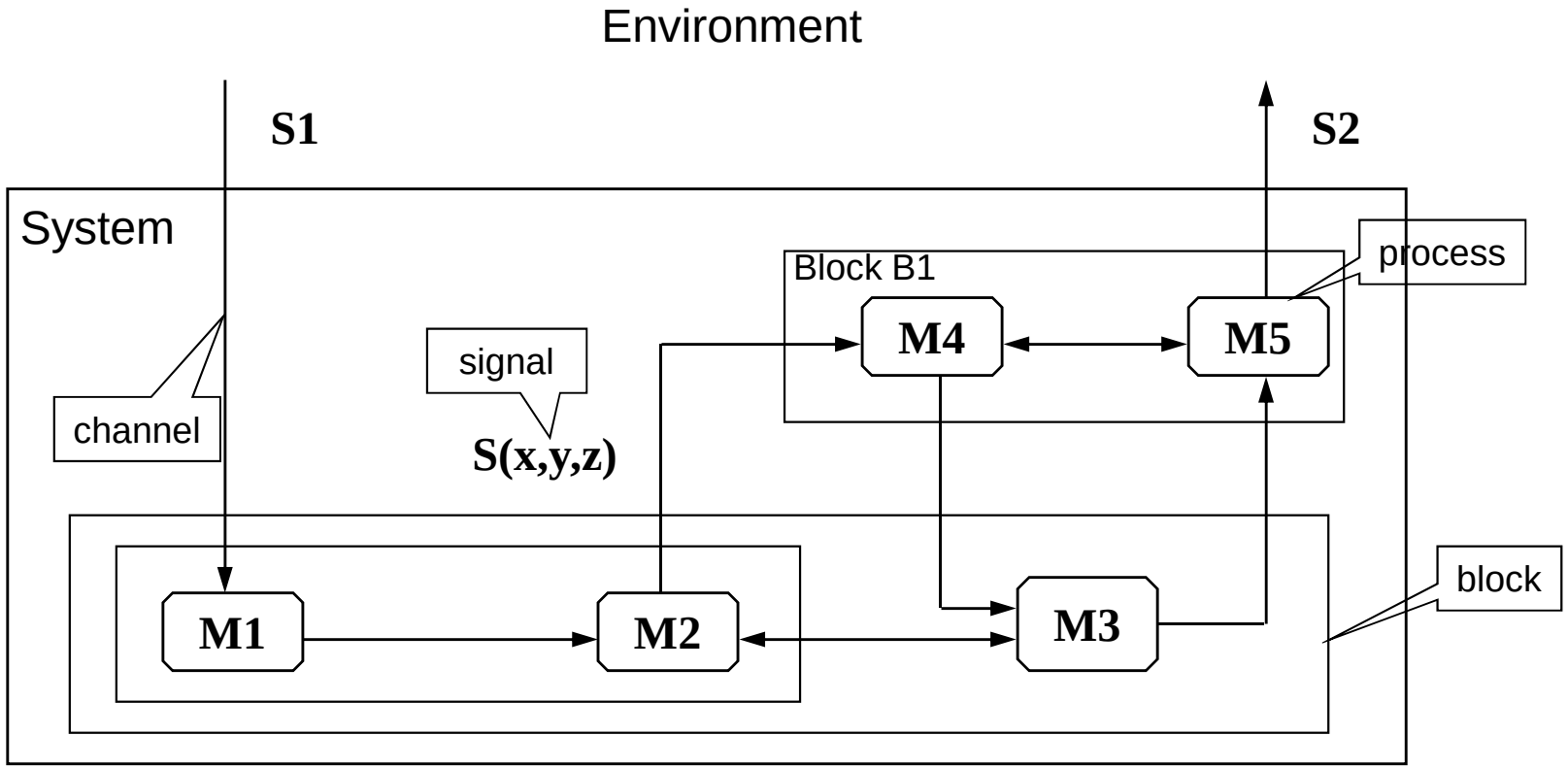
- The definition of the system structure with the interconnections
- The dynamic behavior of each process (or machines) and their interaction with the other processes and the environment
- operations on data (into the processes)

Semantic models - Hierarchy

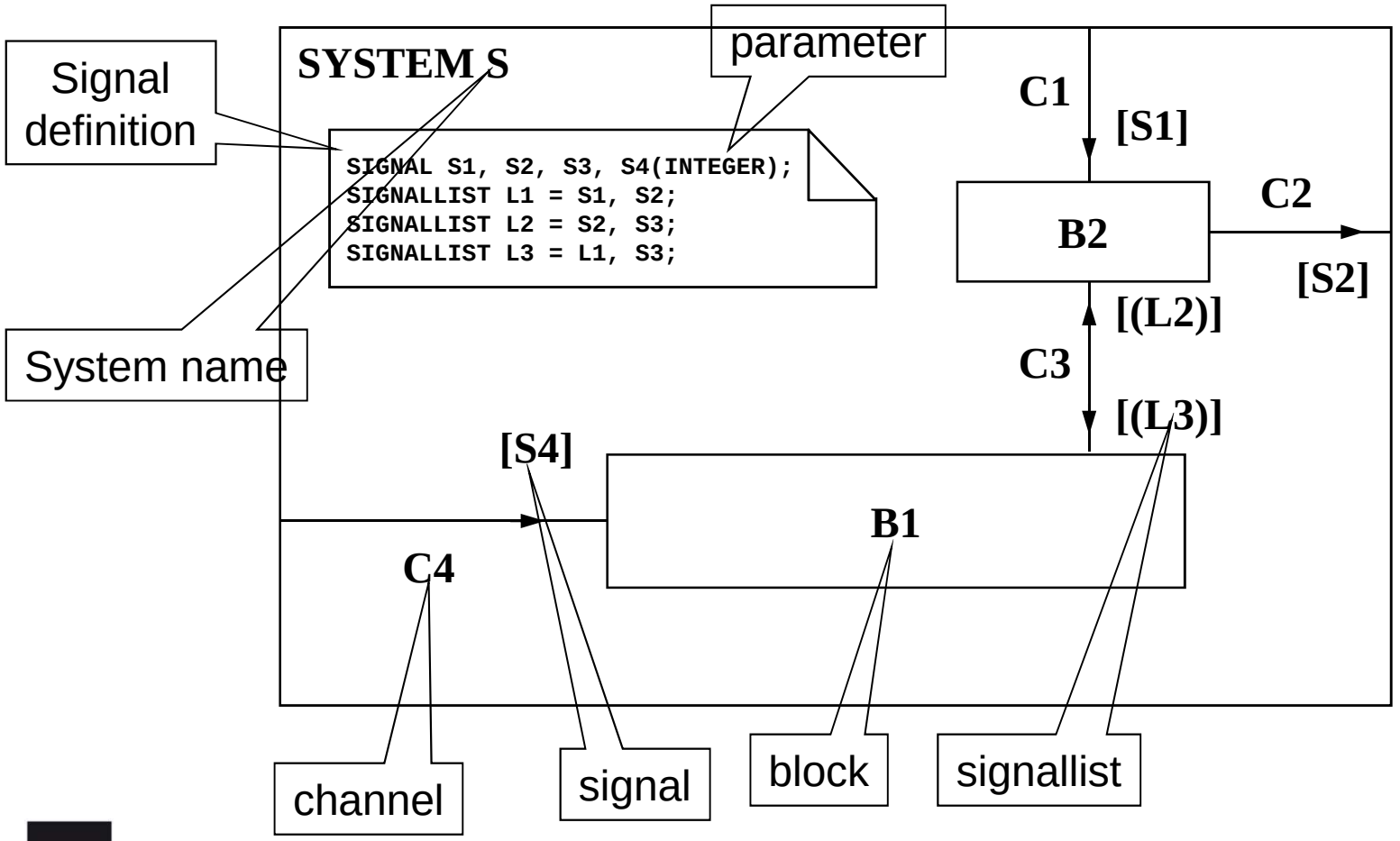
- System architecture:
  - Decomposition by interconnected structural entities: **system**, **block, channel, process**
- System behavior:
  - communicating processes: **signals, variables as inputs/outputs: EFSM**
- Data: **variables, signals, sorts, ASN.1, ...**



System architecture

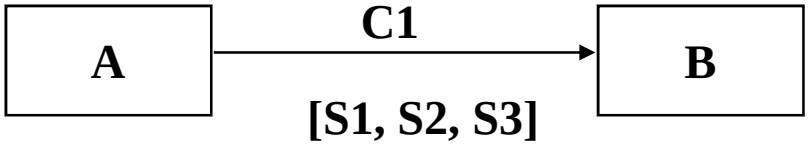


System SDL: example

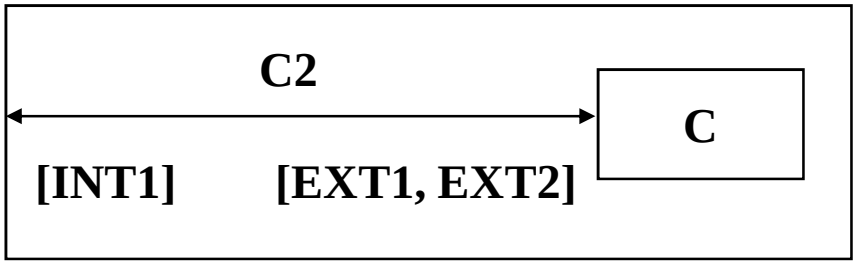


Channels

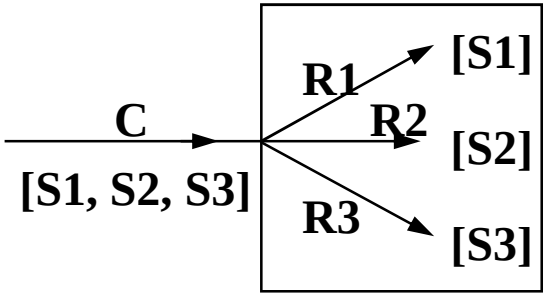
unidirectional

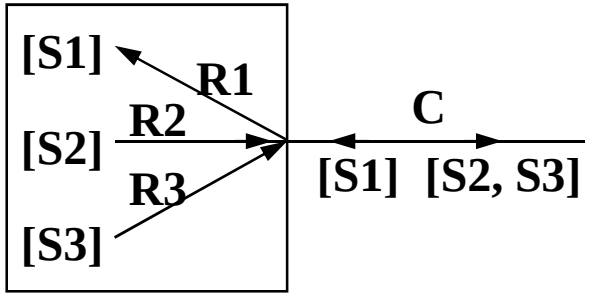


bi-directional

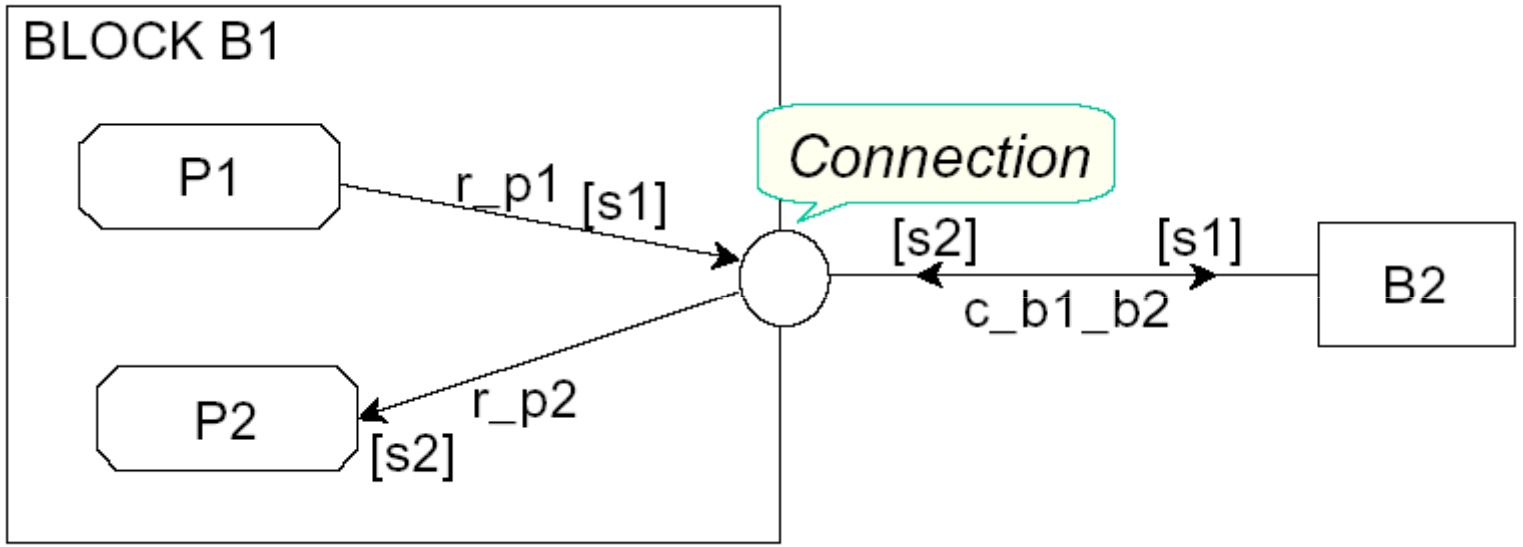


Multi-connections





Connections between blocks



The connections must be defined,
that which channels are linked, and which signals are transmitted.

SDL predefined types

- *INTEGER* signed integer
- *REAL* real
- *NATURAL* positive or null integer
- *CHARACTER* 1 character
- *CHARSTRING* charstring (string of characters)
- *BOOLEAN* boolean
- *TIME* absolute time (syntype of REAL)
- *DURATION* duration (syntype of REAL)
- *PID* to identify a process instance

Operators on predefined types

- All types
  - =, /=
- INTEGER and NATURAL
  - -, +, *, /, >, <, >=, <=, Float, Mod, Rem
- REAL
  - -, +, *, /, >, <, >=, <=, Fix

CONSTANTS

- They can be defined at any level of the SDL hierarchy

SYNONYM maxusers **INTEGER** = 10**;**

Basic user-defined types

- Enumerated types

```
NEWTYPE WeekDay
LITERALS mon, tue, wed, thu, fri, sat, sun;
ENDNEWTYPE;
```

- Range types (often used to index arrays)

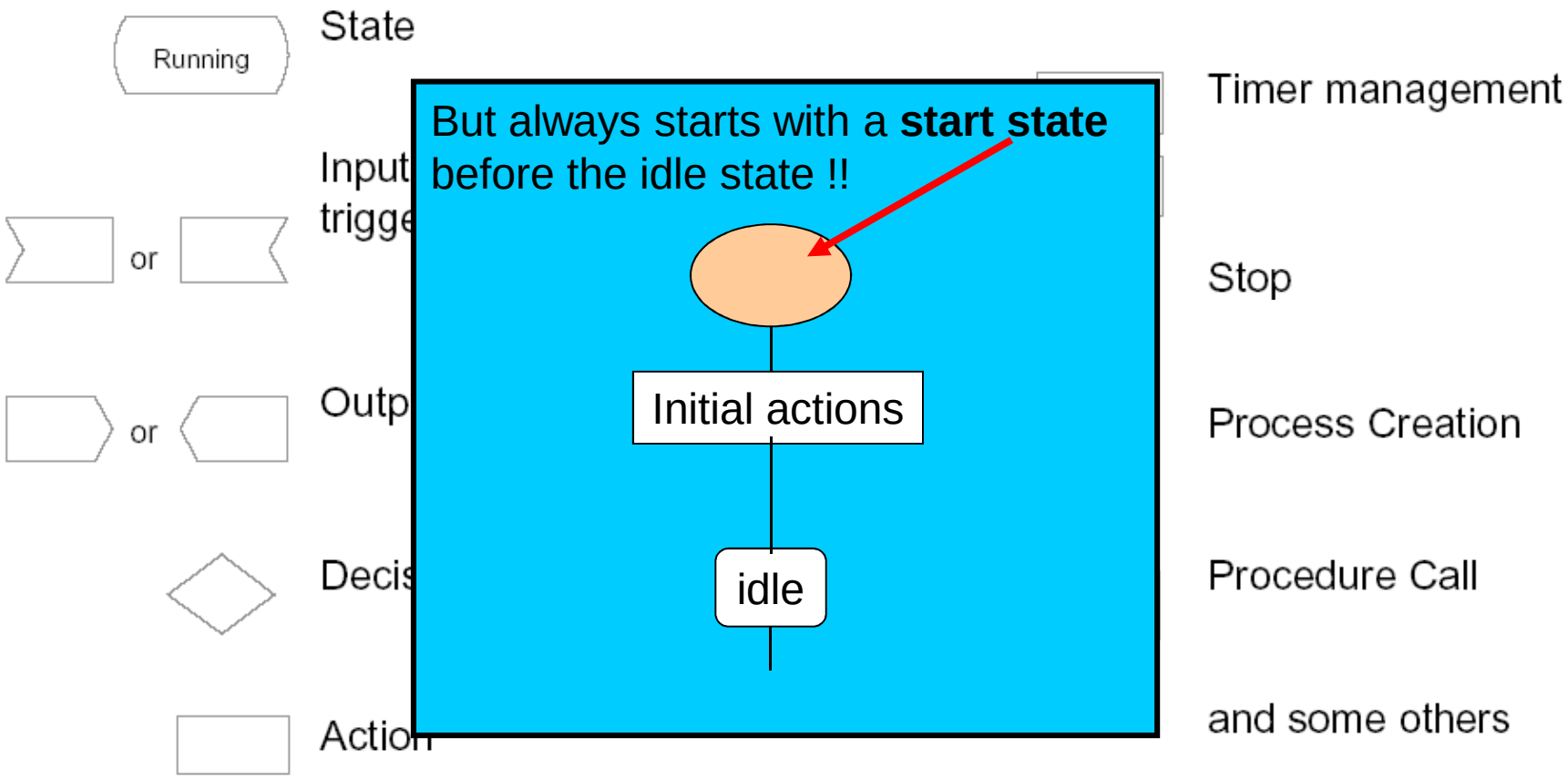
```
SYNTYPE Index_T = Natural
CONSTANTS 1:12
ENDSYNTYPE;

SYNTYPE Digit_T = Character
CONSTANTS '0':'9'
ENDSYNTYPE;

SYNTYPE WeekEnd = WeekDay
DEFAULT sun;  CONSTANTS sat:sun
ENDSYNTYPE;
```

The SDL process

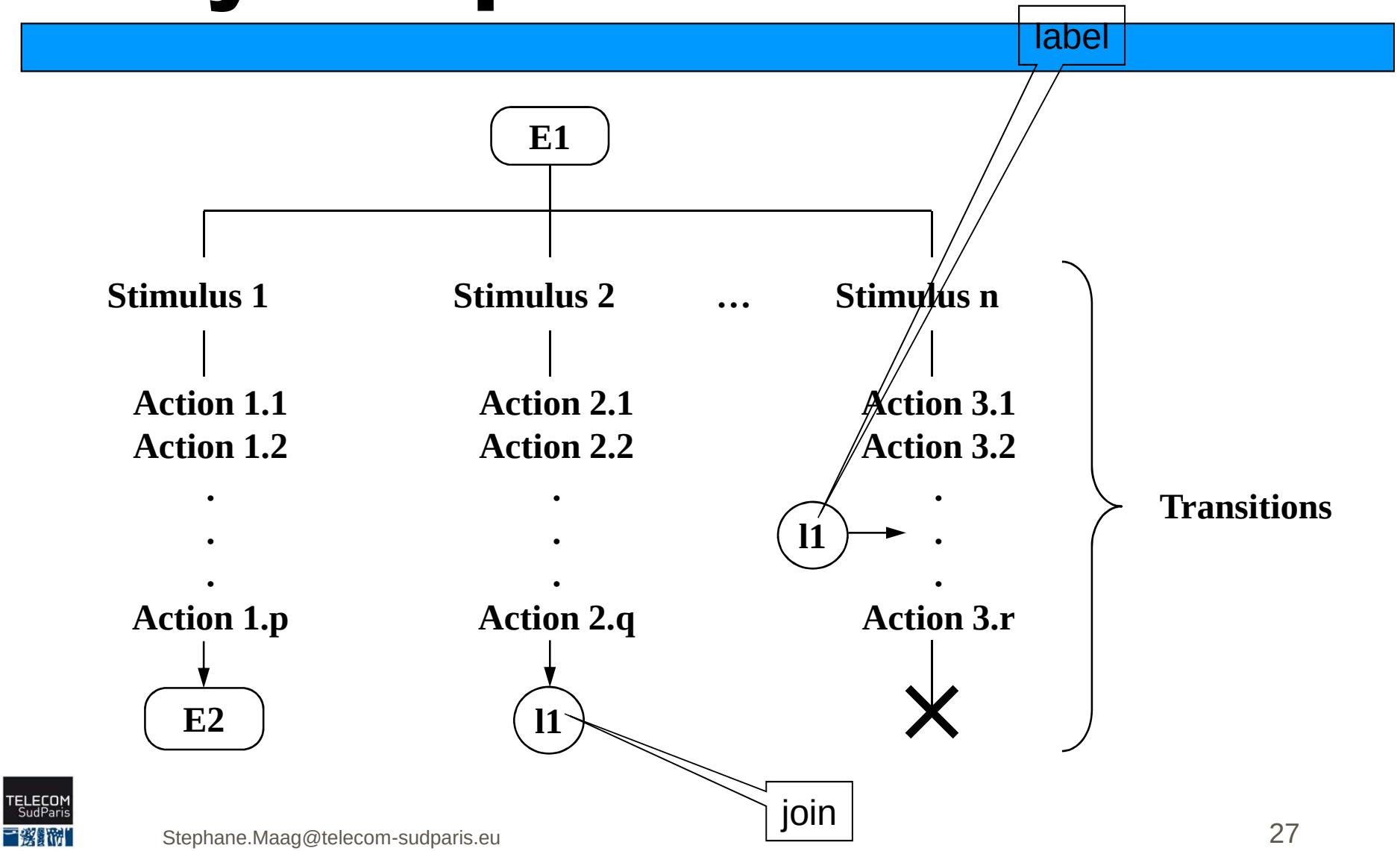
It describes the behavior and extends the FSM concept:

- the queue associated to each process is not necessarily a FIFO.
- A transition (not necessarily of a null length) may contain:
  - receiving and sending data
  - analyzing variables to determine the next transition
  - execution of tasks
  - procedure call
  - dynamic creation of process
  - triggered timers

Major SDL elements in a process



Body of a process

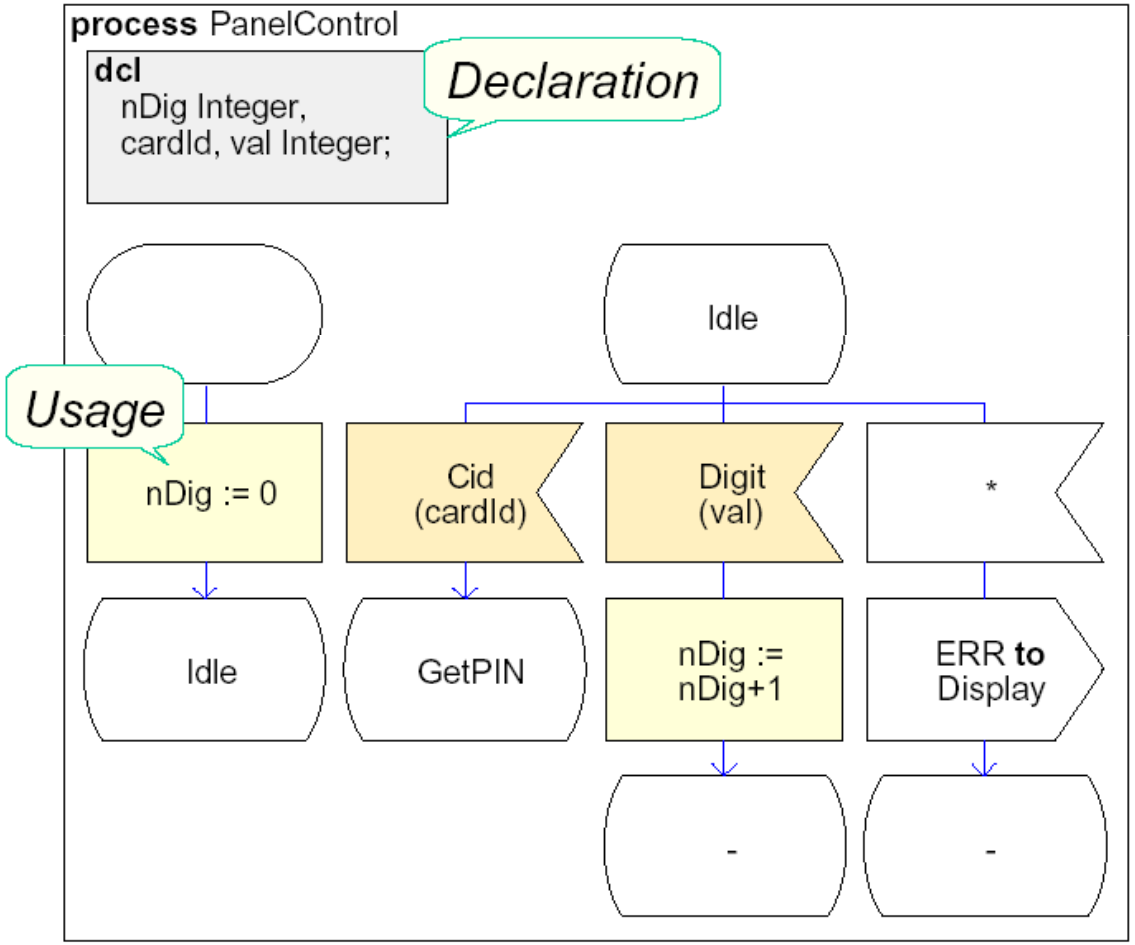


Declaration in processes

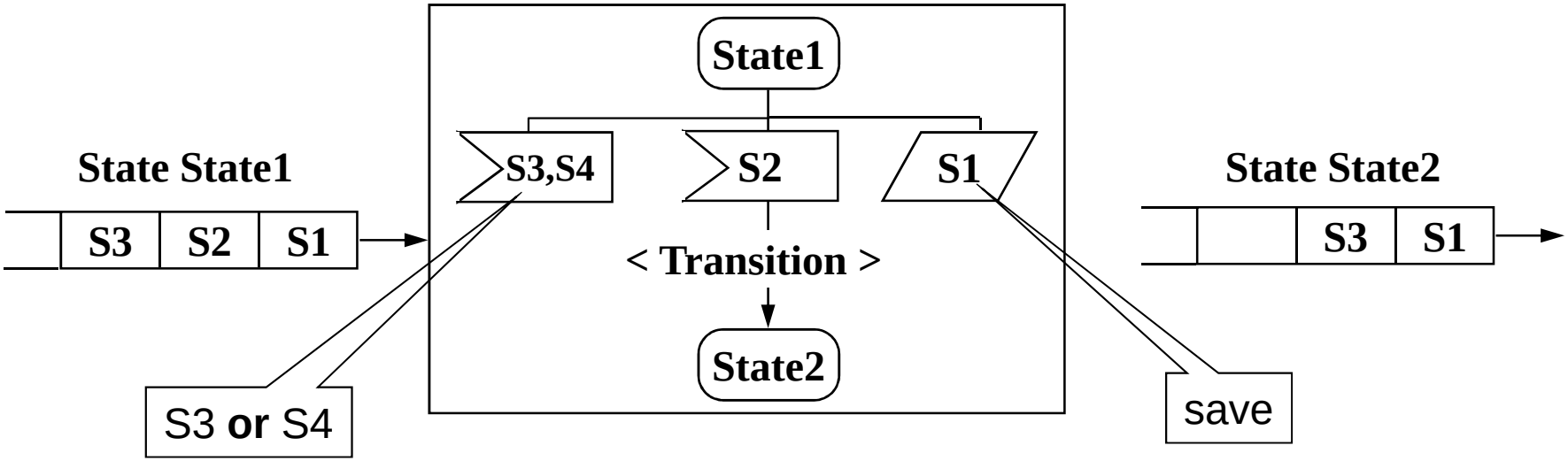
Variables

- declared in a Text symbol of a process, service, procedure
- no global variables at system or block level
- can be initialized:

```
DCL
   nbTransactions Integer := 0,
   v1, v2 MyType;
```

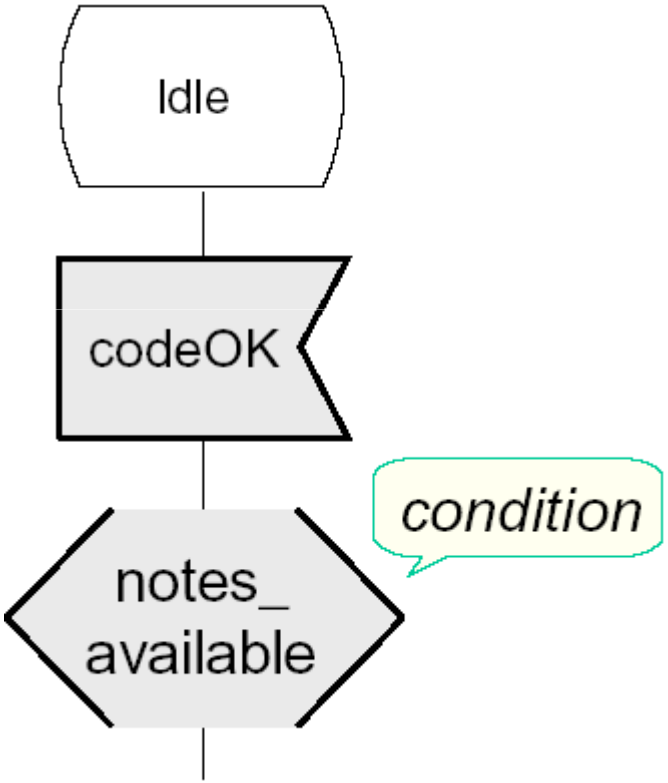


Stimuli types - inputs



"save" allows to save a signal and keeps it in the queue until the next state … waiting for the next signal.

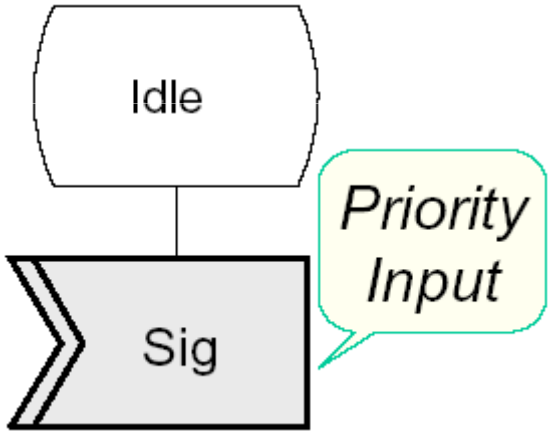
Input - Condition



Boolean expression

- signal can only be consumed if the condition is true, otherwise it is saved.
- ! The expression may not depend on current input signal parameters: only the *previous* value is accessible

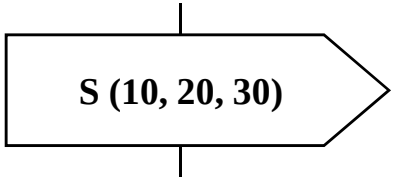
Input - priority



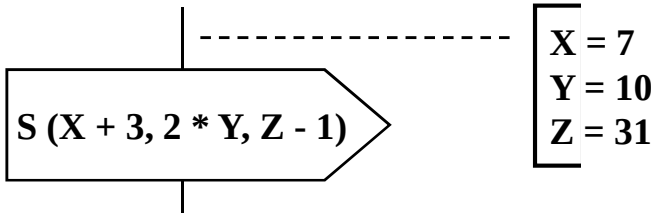
Priority signals are processed
prior to the other signals in the queue

Outputs

- Signal S with three associated values

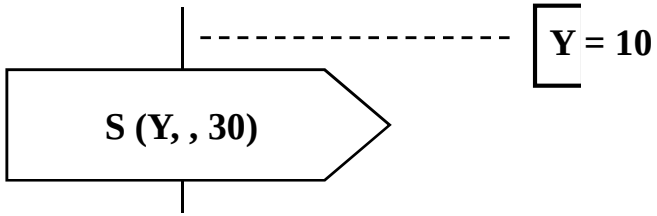


- Signal S avec three expressions to be evaluated



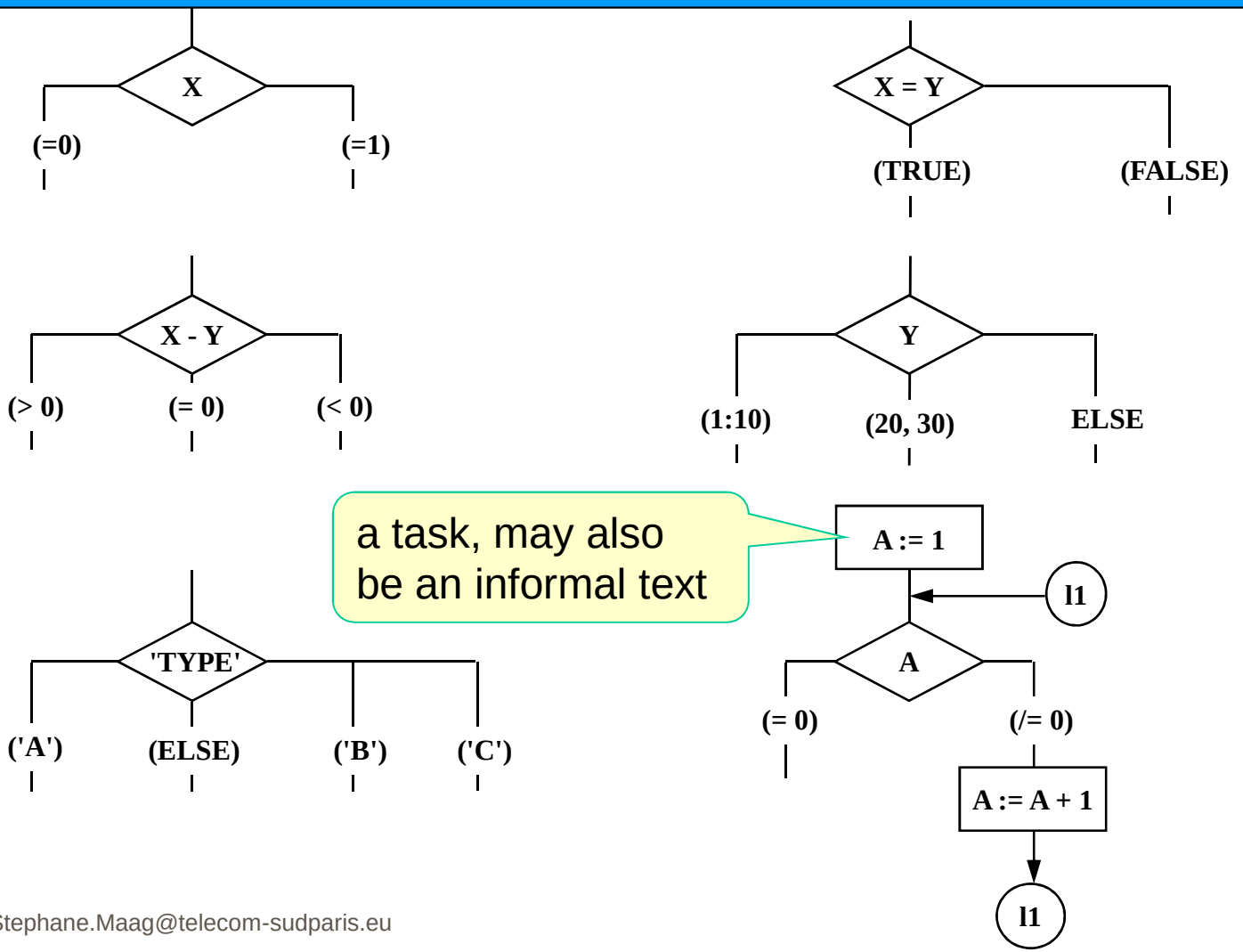
The transmitted signal contains the values : 10, 20, 30

- Signal S with a undefined value

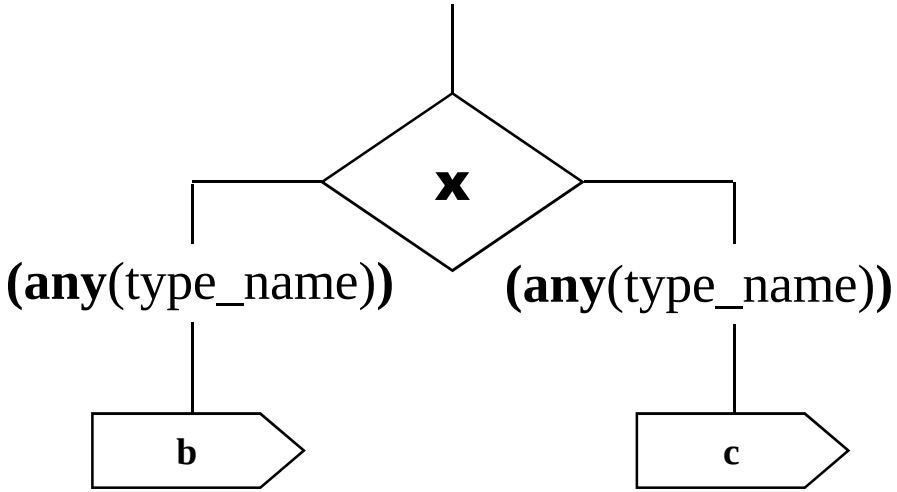


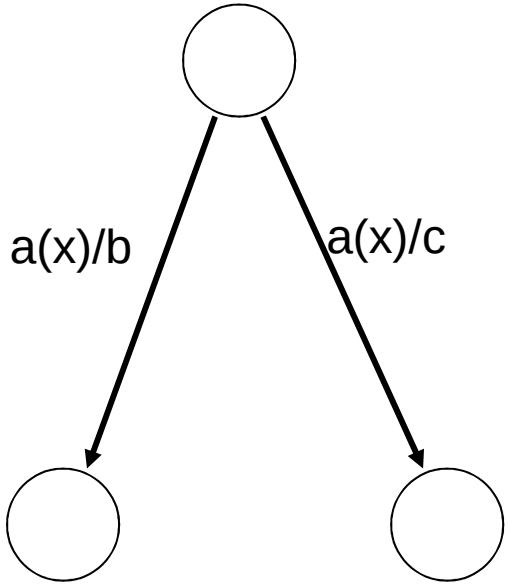
The transmitted signal contains the values : 10, undefined, 30

Decisions


X
(=0)
(=1)
X = Y
(TRUE)
(FALSE)
X - Y
(> 0)
(= 0)
(< 0)
Y
(1:10)
(20, 30)
ELSE
a task, may also
be an informal text
A := 1
l1
'TYPE'
('A')
(ELSE)
('B')
('C')
A
(= 0)
(/= 0)
A := A + 1
l1

Non-deterministic transitions

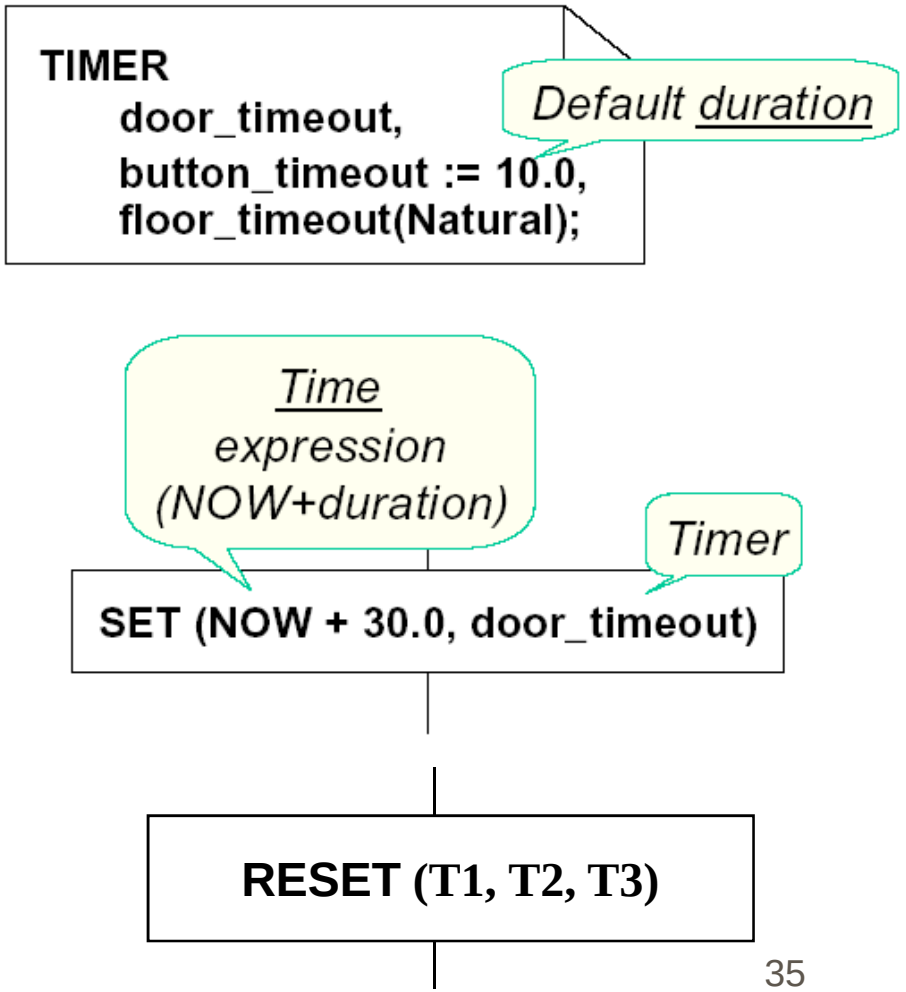




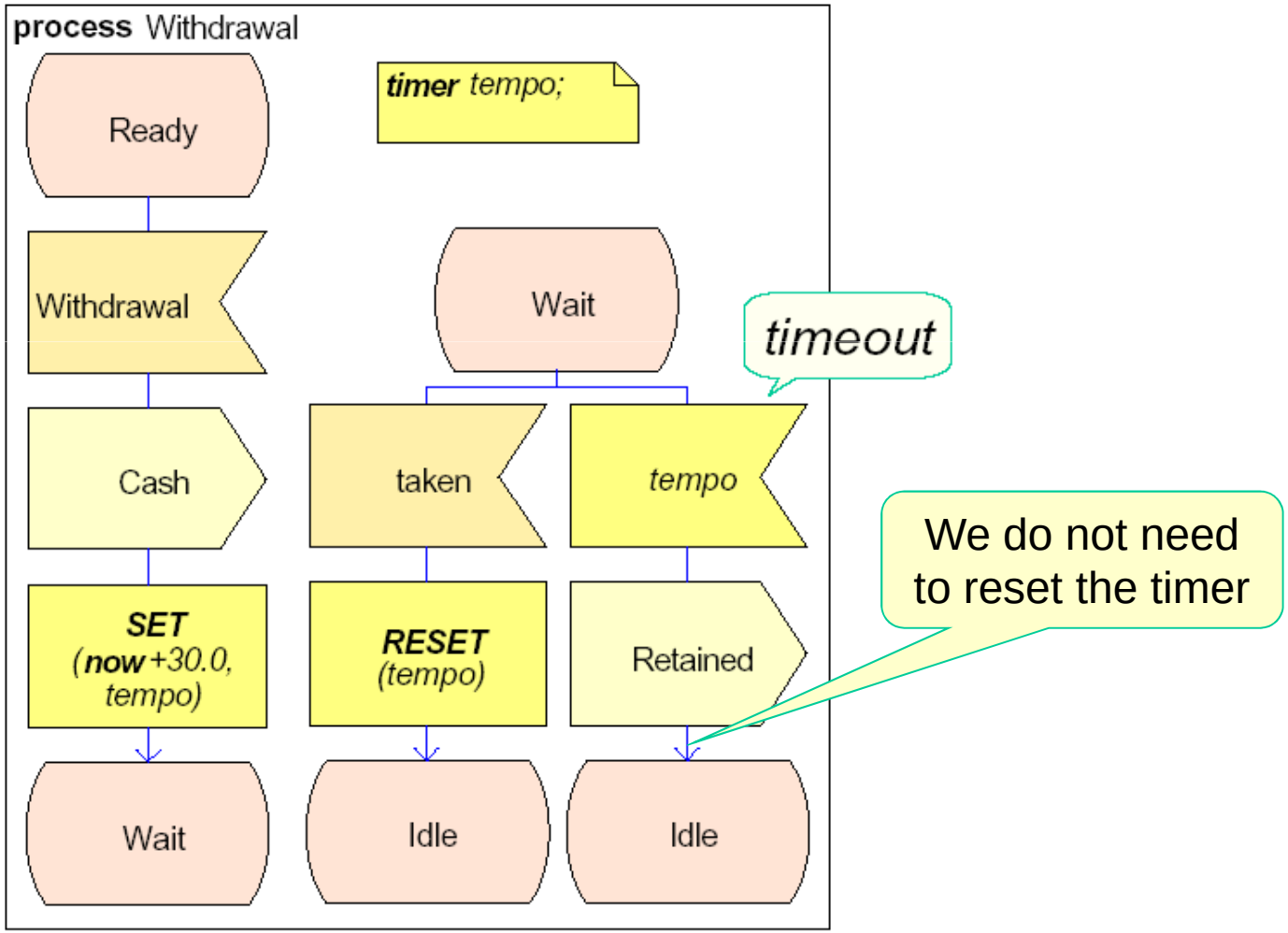
Non-deterministic transitions
are used to describe random events

Express the Time in SDL

- A Timer is a meta-process able to transmit signals on demand to the process.
- The current time is given by the variable **NOW.**
- The RESET also removes the corresponding signal from the process queue (case of an expired TIMER, but the signal is not consumed yet.

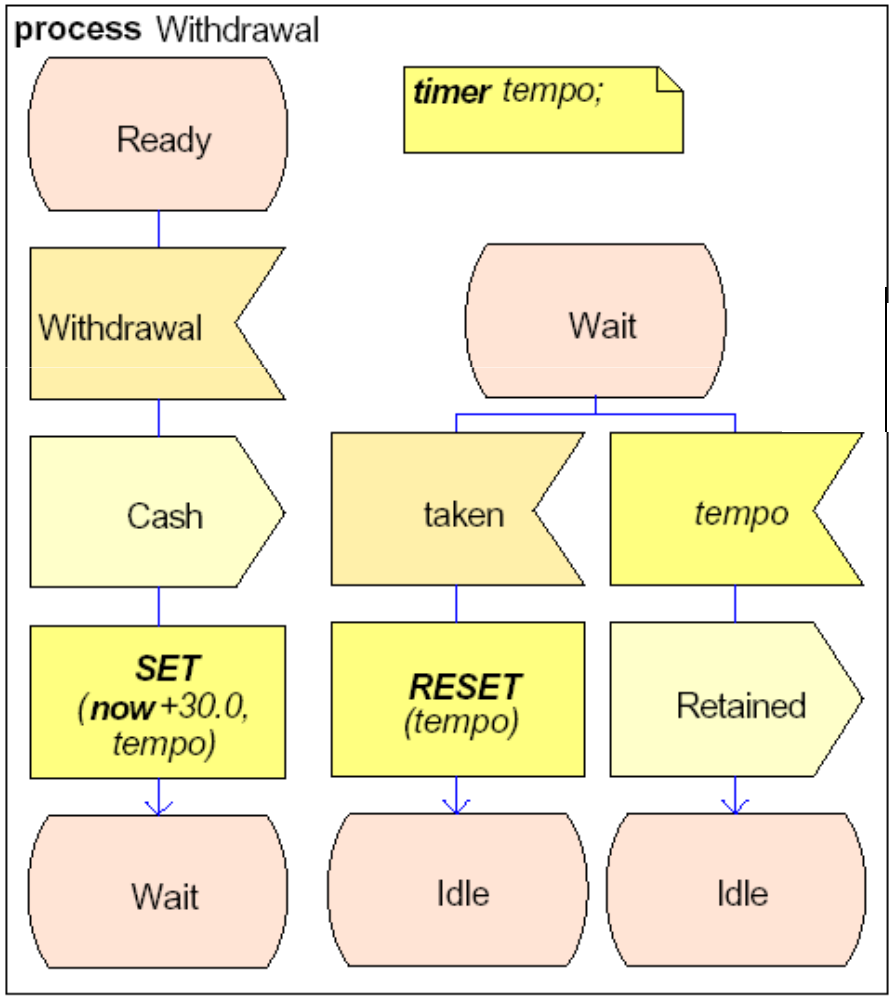


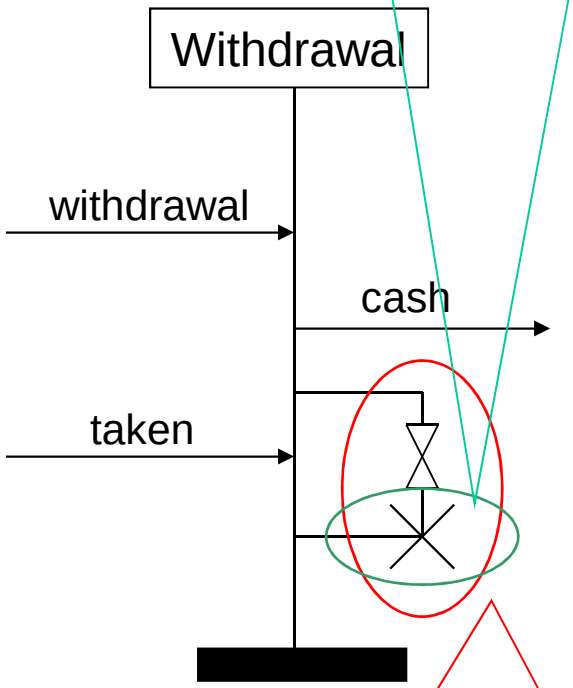
Use of Timers



Mapping with MSC

•Delete the process
•signals remaining in the queue are lost
•future messages to thies process are lost

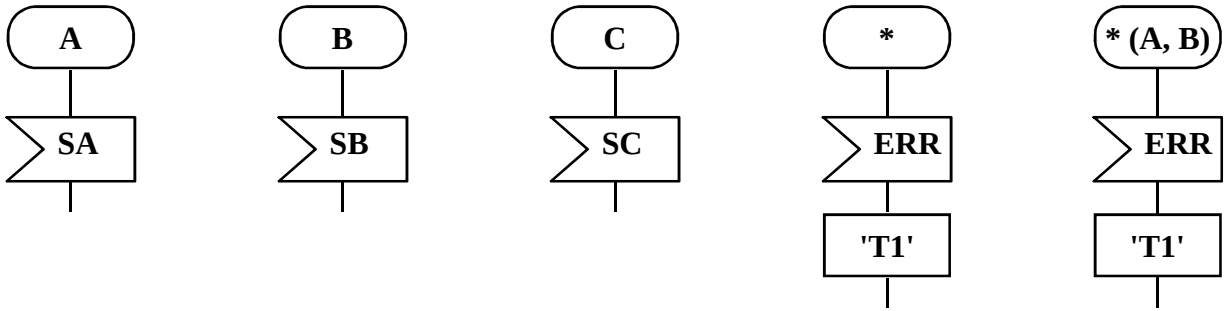




The timer is set and reset because the arriving of signal *taken*

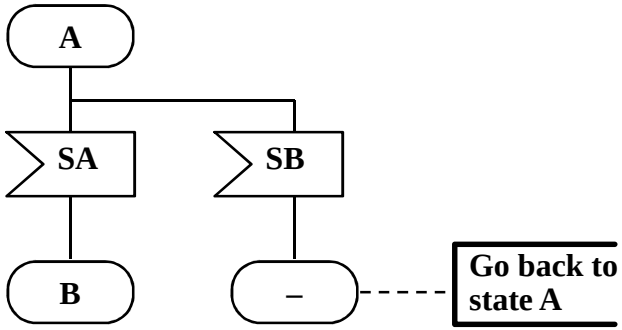
To ease the writing (1/2)

The transition associated to the state * is applicable with all the states, while the state *(A,B) is also applicable with all the states **except** A and B

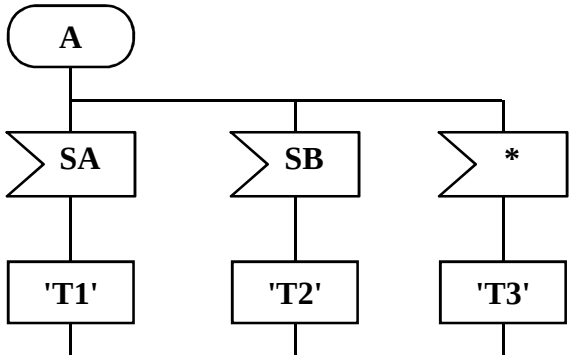


To ease the writing (2/2)

- To go back to the previous state



- Input *: represents all other signals



System simulation - Objectives

The model is now syntactically correct and semantically consistent. But it is good ?

From low costs to high quality:

- debugging
- evaluation of alternative solutions
- verification, detection of errors, comparison with MSC requirements.
- Test generation

$\Rightarrow$ to minimize the final costs

Two kind of simulation

Interactive

- step-by-step (debugging)
- access to all data
- MSC generation
- SDL tracking

Exhaustive

- fully automatic
- measures state and transitions coverage
- check properties
- reachability graph generation

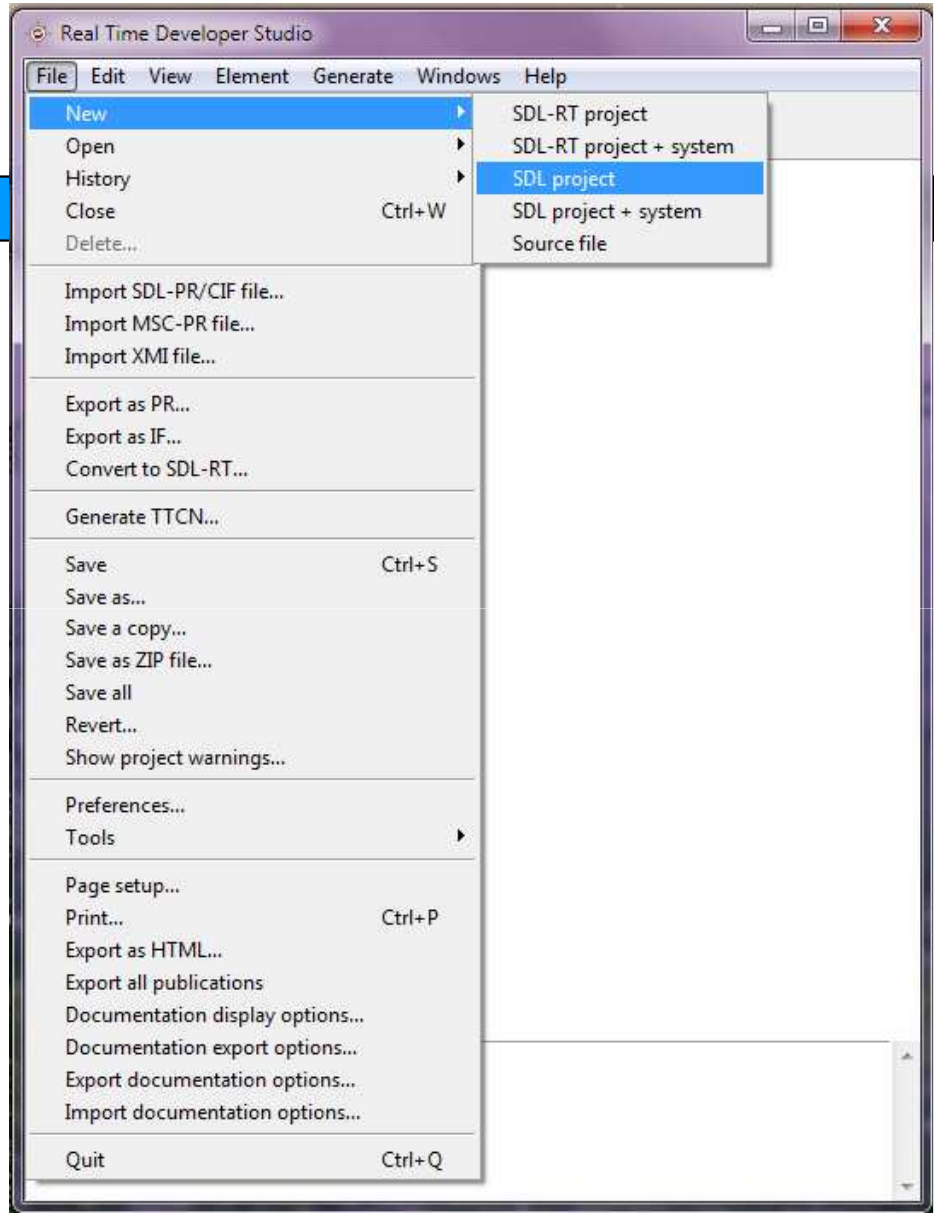
Real Time Developer Studio (RTDS)

- A Pragmadev tool
- The tool allowing the edition from the requirements
- Architectural and behavioral design
- Model checking capabilities,
- Traceability information.
- Code generation
- Testing
- TTCN3

GUI - RTDS

Graphical User Interface

Then:
- Save As (in your Home dir!)
- Right click -> add component (system, then block then process)



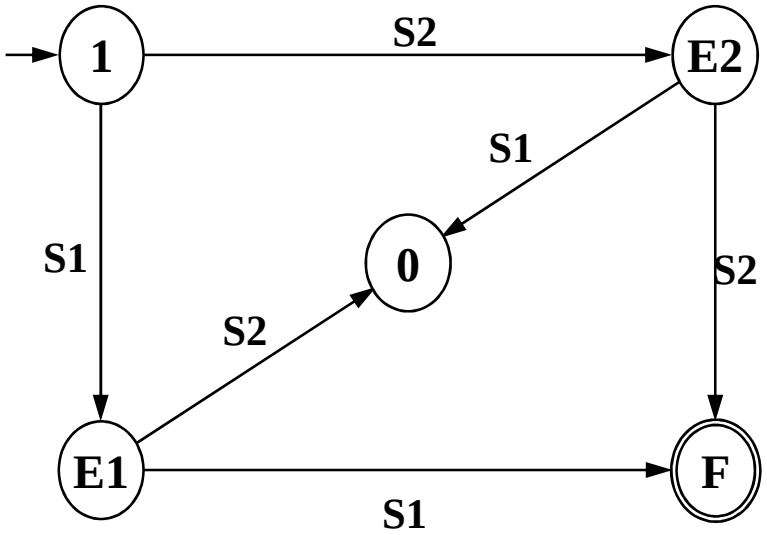
Conclusion

- SDL, a language to specify complex systems. User-friendly with its PR/GR
- Powerful to express important protocols
- Allows to simulate system behaviors

- In the following : *on the road of instantiating and testing … on the road ..*

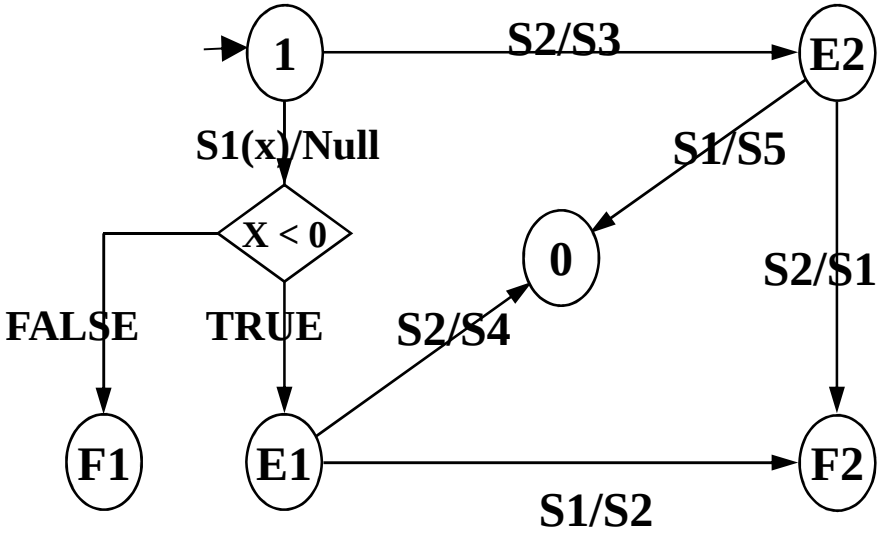
FSM - EFSM

FSM



I/O EFSM

Integer x;



Short Exercices
Specification using FSM/EFSM

Create a deterministic FSM representing the language based on the words {0, 1} that contains all the words in which sequences containing **no more** than 4 consecutive ‘1’ may be read.