2.3 SDL

Language for the specification of distributed systems.

Dates back to early 70s; formal semantics late 80s.

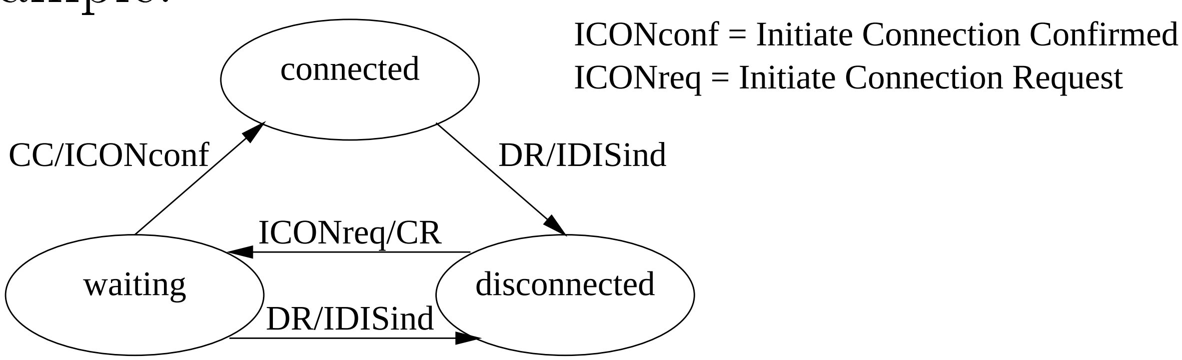
Language defined by CCITT (Committee Consultativ International Telegraphique et Telephonique).

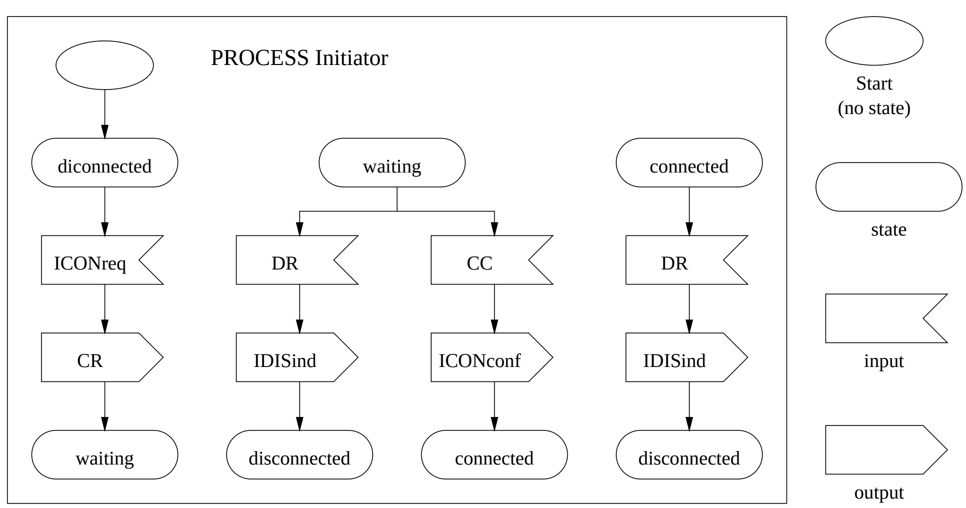
2.3.1 Language elements

Graphical and textual format.

Basic element: process (= extended FSM);
Extension: Operations on data.

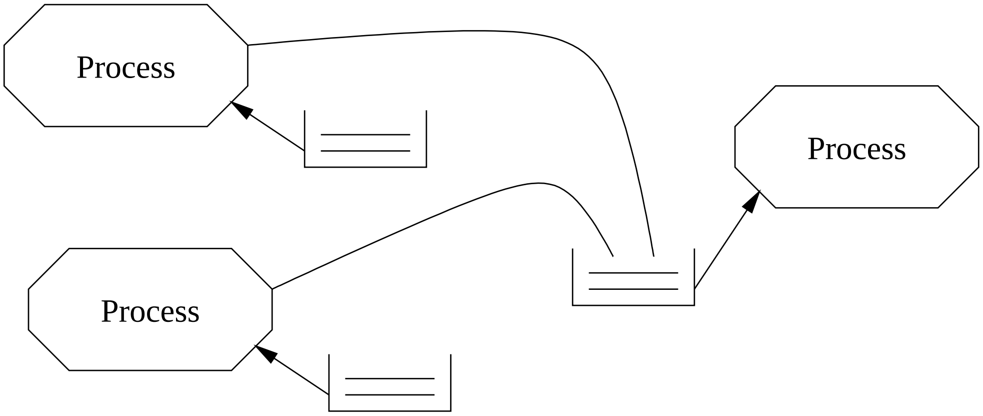
Example:





Semantics

Based on implicit input queues:



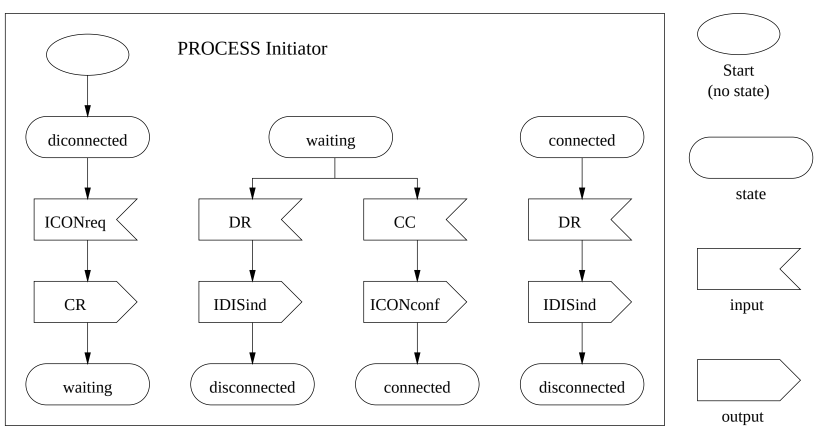
For every state:

first signal from input queue is removed and analyzed whether it is relevant for any state transition.

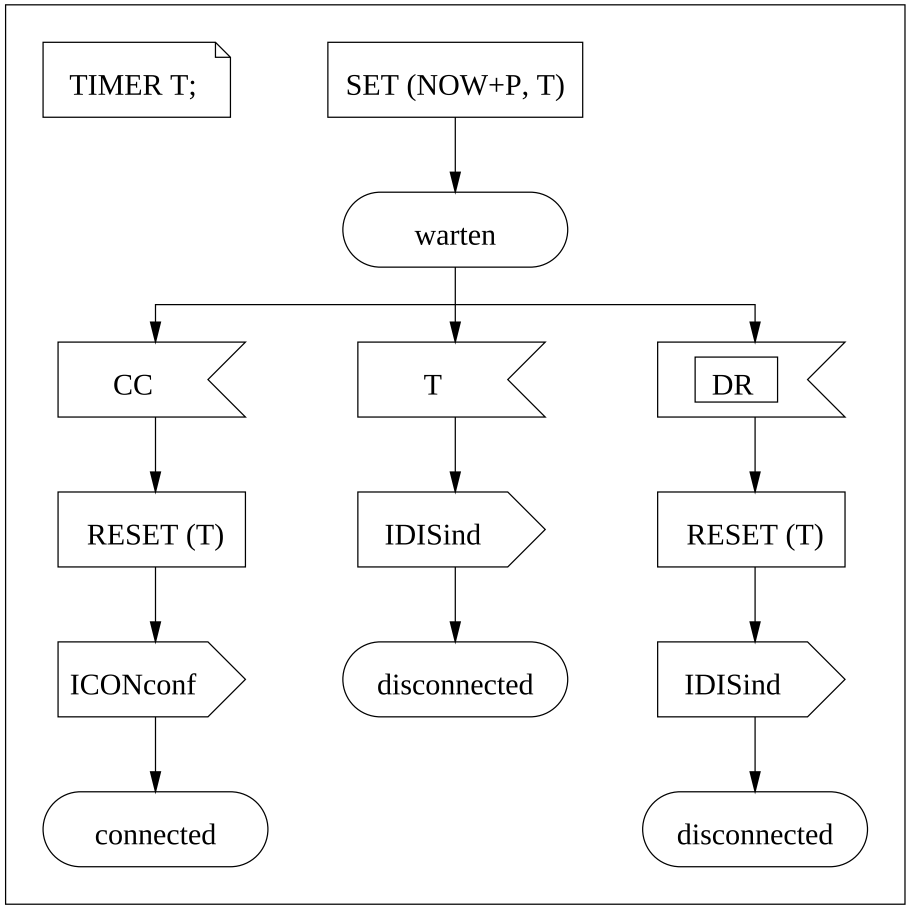
Signals or not stored (exception: SAVE mechanism)

Input queues are assumed to have infinite capacity.

Problem: how large should input buffers be for any physical implementation?



Timers



`SET(NOW+P,T)` sets timer to 'now + P'.

After the specified time, an input event `T` will be entered into the input queue.

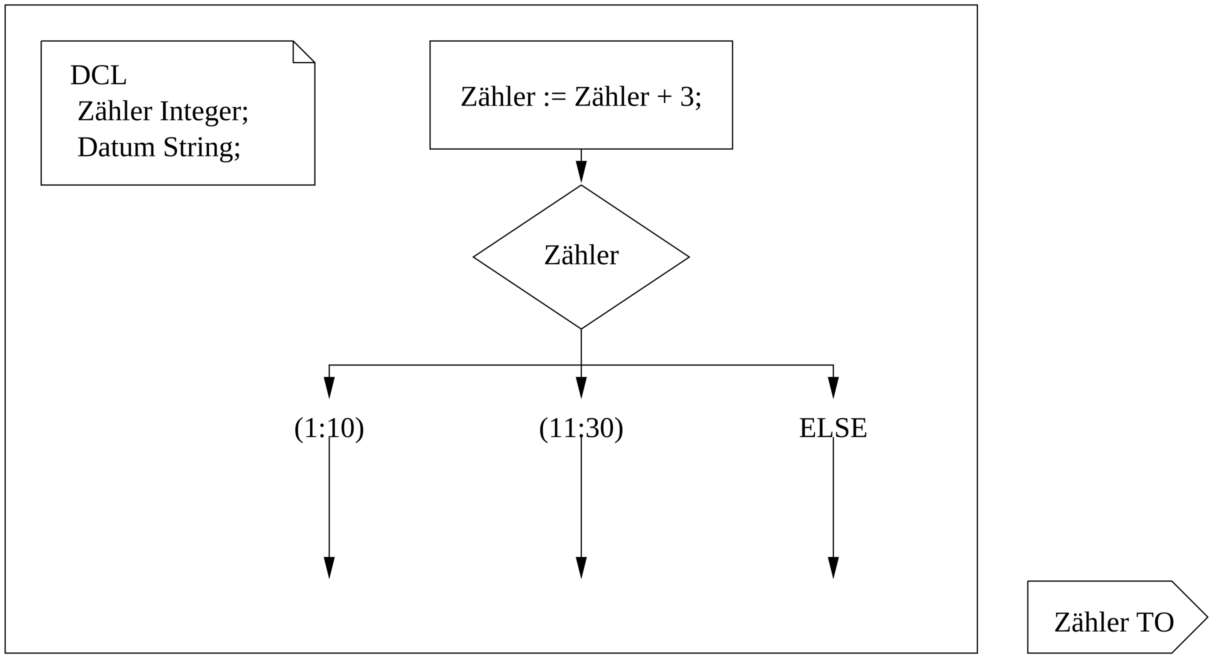
No immediate action, if other events are still in the queue.

If timer already in the queue, `RESET` will remove timer from input queue.

Timer mechanism sufficient for telecom applications, not appropriate for hard real-time constraints.

Operations on data

Variables can be declared and used in input/output.



Concept of data types based on **abstract data types (ADTs)**. Syntax for operations just like in usual programming languages.

Hierarchy

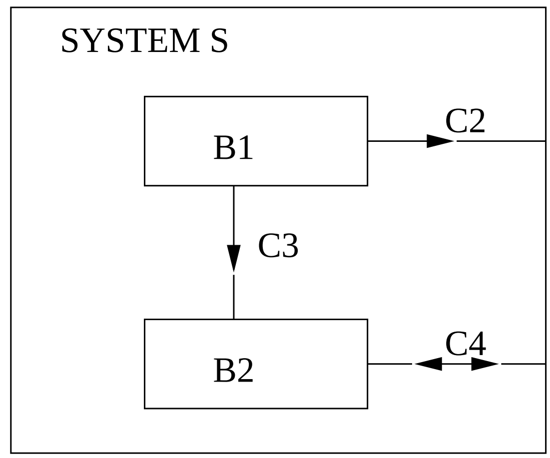
Processes can obtain process identifiers of offspring; no other form of hierarchical processes.

However, **blocks** can be used to describe process interaction.

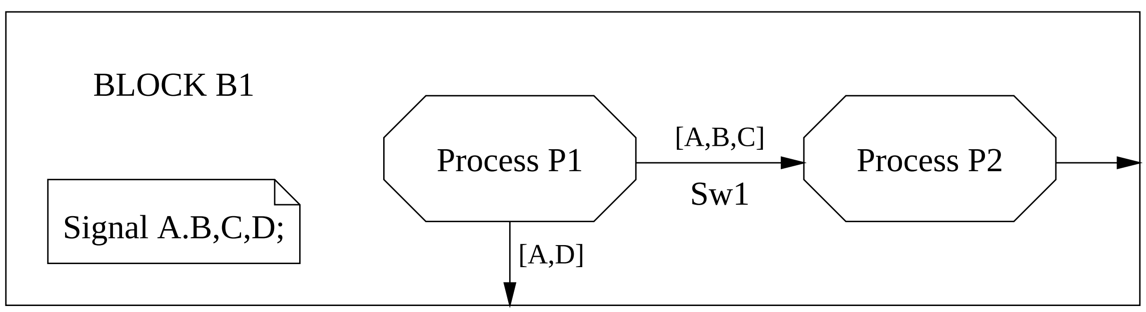
Edges = channels; labels: channel name and/or signal names.

Blocks can be hierarchical.

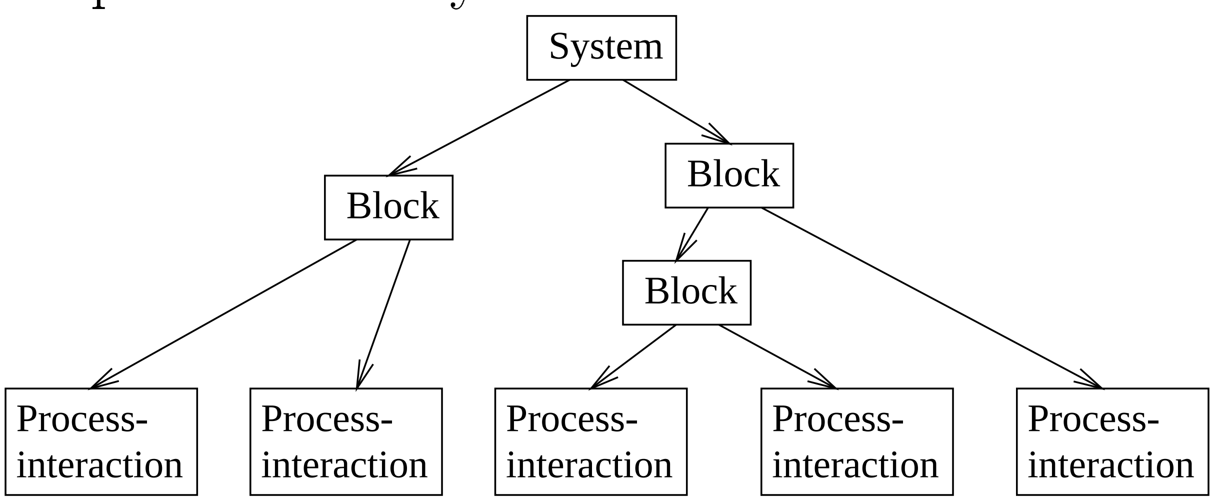
- Top-level block = **system**;



- Lowest level = **process interaction diagramm**:



- Complete hierarchy:



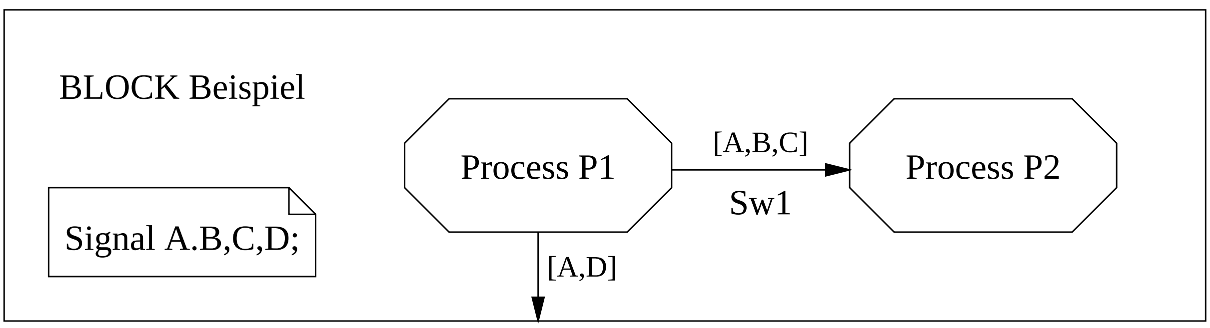
Inter-process communication

Methods for addressing recipient:

1. **Explicit destination address**

2. **By indirect addressing**:

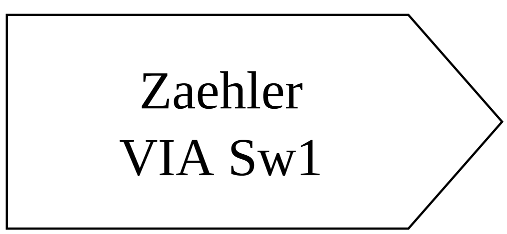
   Recipient determined by context:



   Signal `B` will always go to process P2.

3. **Addressing of the channel**

   For the same example:

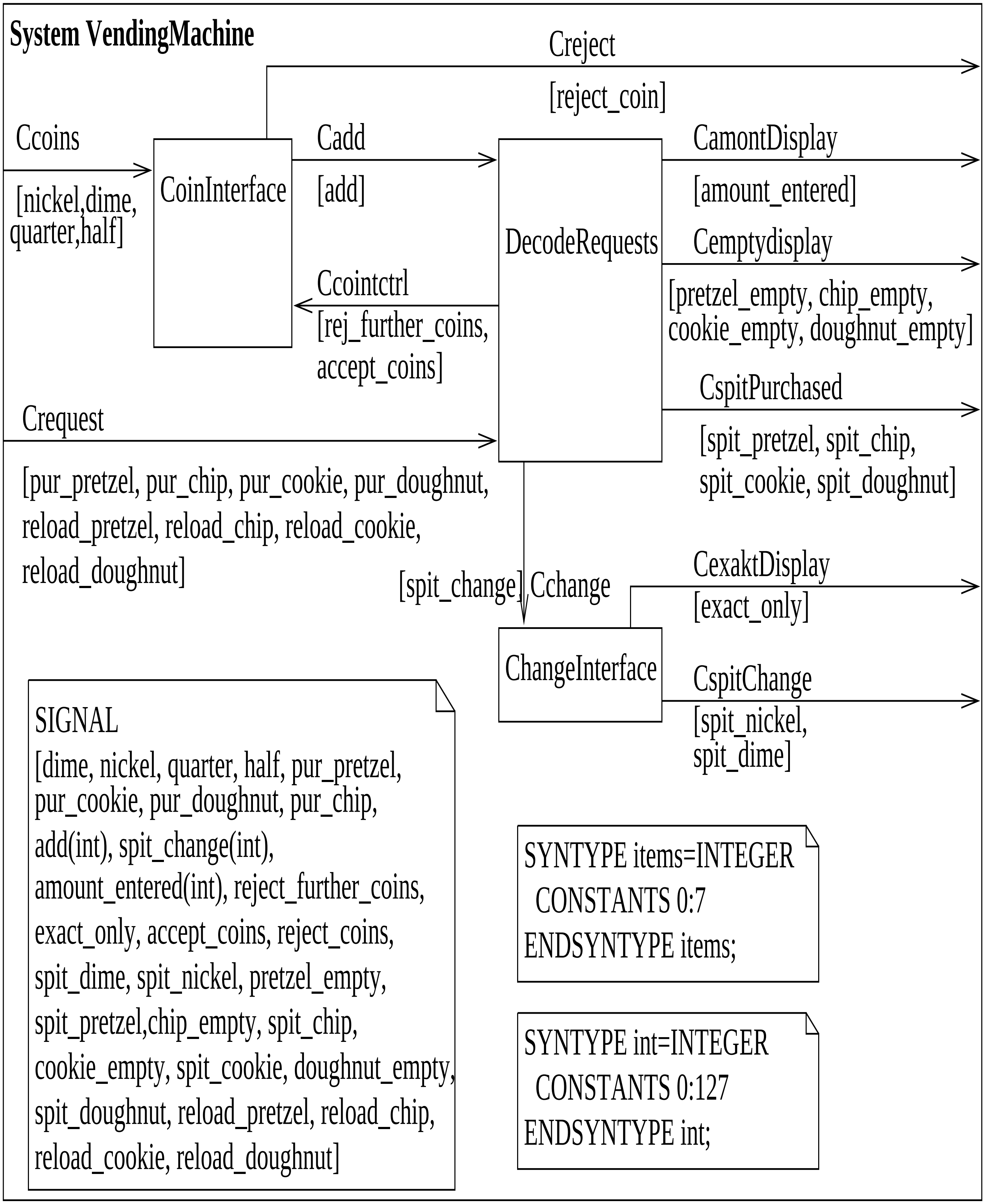


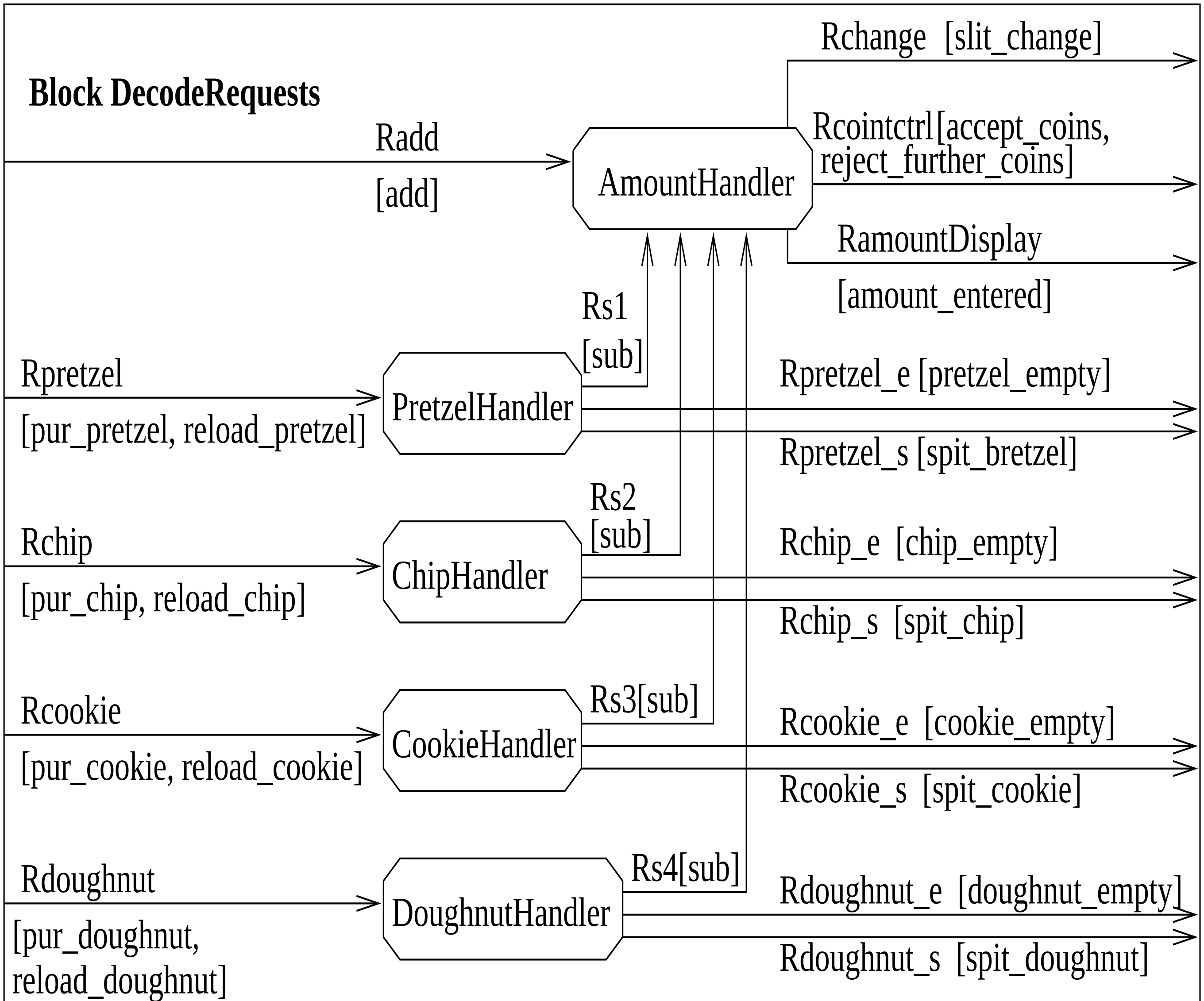
Evaluation

Salient features of SDL:

- No general broadcast mechanism;
  suited for distributed systems
- Adequate for telecommunications
- Problems for hard time constraints
- Processes can be generated dynamically; processes can terminate themselves
- No hierarchical processes; hierarchy limited to blocks
- Size of input buffers difficult to estimate.
- Non-deterministic behaviour in case several messages arrive at an input queue at the same time.

Complex example: vending machine for cookies, potato chips, doughnuts and pretzels

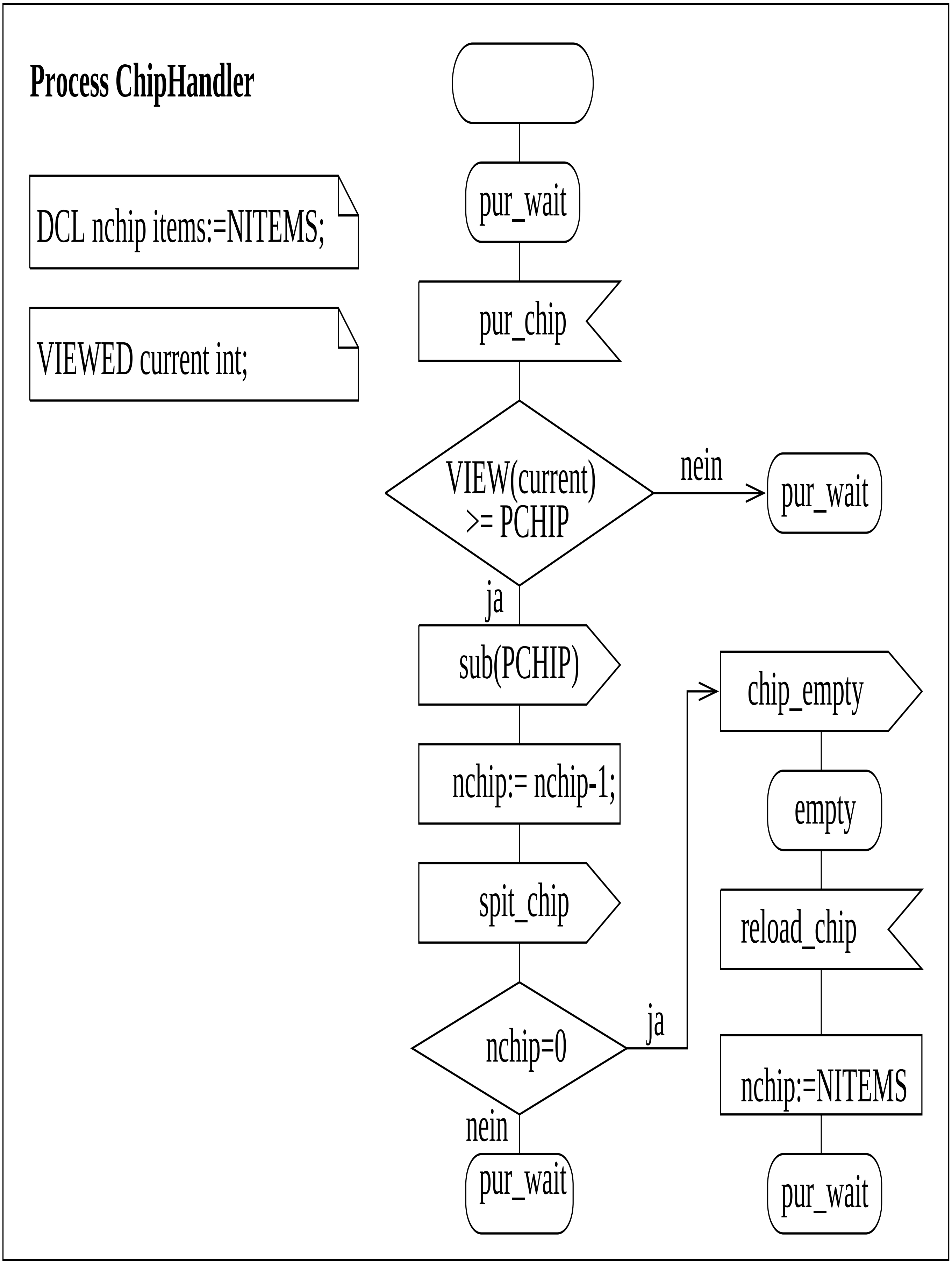




CONNECT Cadd AND Radd;
CONNECT Ccoinctrl AND Rcoinctrl;
CONNECT Cchange AND Rchange;
CONNECT CAmountDisplay AND RamountDisplay;
CONNECT Crequest AND Rpretzel,Rchip,Rcookie,
Rdoughnut;
CONNECT CemptyDisplay AND Rpretzel_e,Rchip_e,
Rcookie_e,Rdoughnut_e;
CONNECT CspitPurchased AND Rpretzel_s,
Rchip_s,Rcookie_s,Rdoughnut_s;

SYNONYM PRETZEL int=50
SYNONYM PCHIP int=15;
SYNONYM PCOOKIE int=55;
SYNONYM PDOUGHNUT
int=60;
SYNONYM PMAX int=60;
SYNONYM NITEMS items=7;

SIGNAL sub(int);


Process ChipHandler
DCL nchip items:=NITEMS;
VIEWED current int;
pur_wait
pur_chip
VIEW(current)
>= PCHIP
nein
pur_wait
ja
sub(PCHIP)
nchip:= nchip-1;
spit_chip
nchip=0
ja
nein
pur_wait
chip_empty
empty
reload_chip
nchip:=NITEMS
pur_wait

2.4 Petri nets

2.4.1 Introduction

Carl Adam Petri, 1962.

Modelling of causal dependence.

No explicit reference to time.

No global synchronization.

Appropriate for modelling of distributed systems.

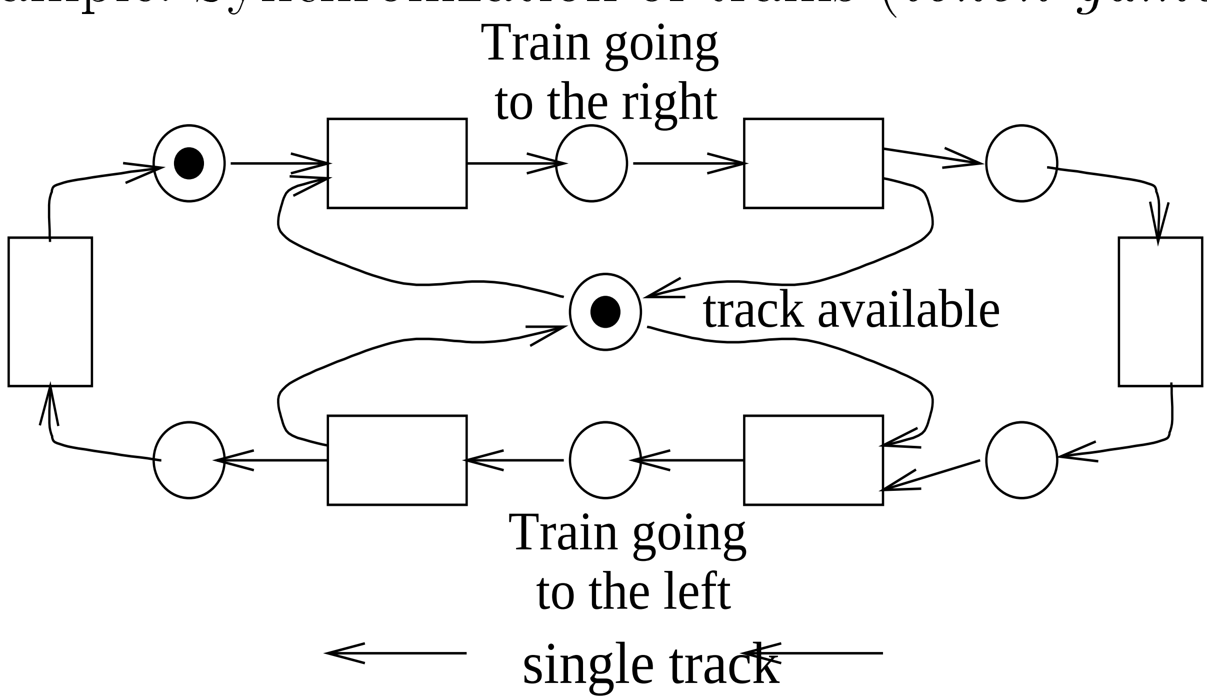
Condition: can be met; represented by circles.

Token within circle: condition is met.

Events take no time; represented by boxes.

Dependencies modelled by edges.

Example: Synchronization of trains (*token game*):



2.4.2 Condition/Event nets

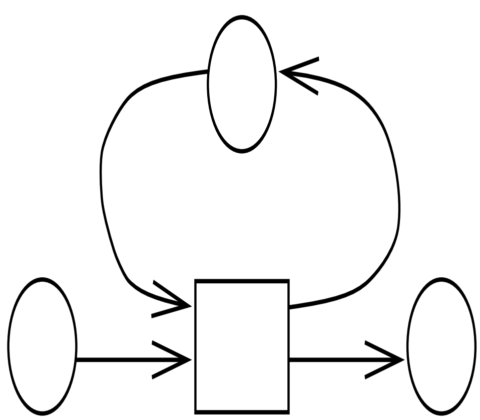
Def.: Tripel $N = (B, E, F)$ called *net*, iff:

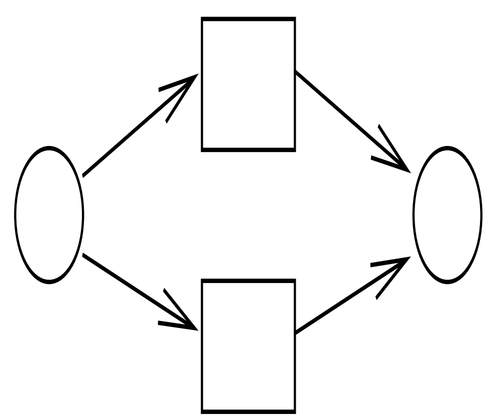
1. B and E are disjoint sets
2. $F \subseteq (E \times B) \cup (B \times E)$ is a binary relation, (flow of N).

Def.: Let N be a net, $x \in (B \cup E)$.
${}^\bullet x := \{y | yFx\}$ is called *pre-condition* and
$x^\bullet := \{y | xFy\}$ is called *post-condition*

Def.: $(b, e) \in B \times E$ is called ______________
iff $bFe \wedge bFe$.
N is called **pure**, if F has no ____________.

Def.: N is called **simple**, if different elements don't have the same pre- and post-conditions.





Condition/event nets are simple nets with no isolated elements and additional properties.

Condition/event nets are bipartite graphs.

Condition/event nets: maximum of 1 token per condition.

2.4.3 Place/transition nets

Several tokens per condition (called places P in this case); weighted edges.

Def.: Mapping $M : S \rightarrow I\!N \cup \{\omega\}$ is called **marking**.

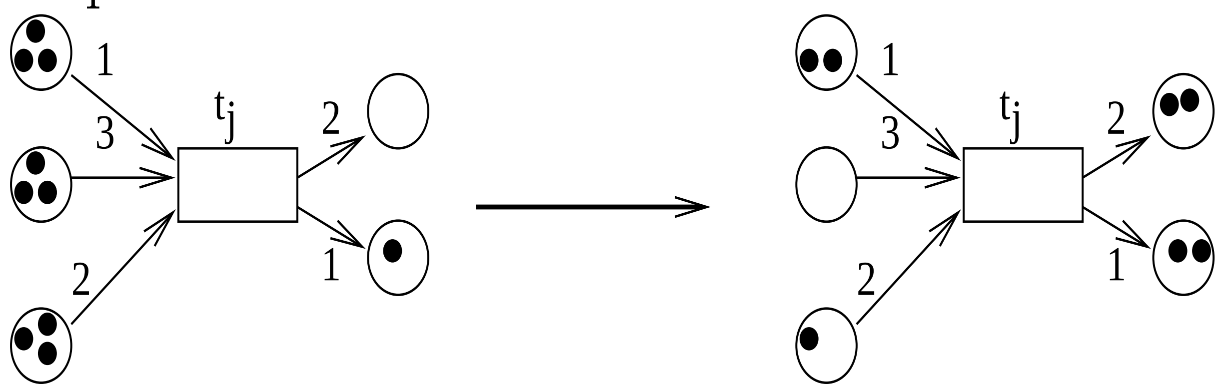
Def.: (S, T, F, K, W, M_0) is called place/transition net $\iff$

1. $N = (S, T, F)$ is a net, S is the set of places, T is the set of transitions.
2. $K : S \rightarrow (I\!N \cup \{\omega\}) \setminus \{0\}$ is called capacity of places ($\omega =$ unlimited).
3. $W : F \rightarrow (I\!N \setminus \{0\})$ is the weight of edges.
4. $M_0 : S \rightarrow I\!N \cup \{\omega\}$ is called initial marking.

Def.: Switching transition results in new marking M', generated from current marking M by:

$$M'(s) = \begin{cases} M(s) - W(s,t), & \text{if } s \in {}^\bullet t \setminus t^\bullet \\ M(s) + W(t,s), & \text{if } s \in t^\bullet \setminus {}^\bullet t \\ M(s) - W(s,t) + & \\ W(s,t), & \text{if } s \in {}^\bullet t \cap t^\bullet \\ M(s) & \text{otherwise} \end{cases}$$

Example:



Default: $W(f) = 1, K(s) = \omega$.

Transition t can take place if t is activated.

Def.: Transition $t \in T$ ist called M-activated $\iff$

$(\forall s \in {}^\bullet t : M(s) \geq W(s,t)) \wedge$
$(\forall s \in t^\bullet : M(s) \leq K(s) - W(t,s))$

Def.: Let $\underline{t} : S \to \mathbb{Z}$ be defined as:

$$\underline{t}(s) = \begin{cases} -W(s,t), & \text{if } s \in {}^{\bullet}t \setminus t^{\bullet} \\ +W(t,s), & \text{if } s \in t^{\bullet} \setminus {}^{\bullet}t \\ -W(s,t) + W(s,t), & \text{if } s \in {}^{\bullet}t \cap t^{\bullet} \\ 0 & \text{otherwise} \end{cases}$$

A transition t taking place will generate a new marking from the current one as follows:

$$\forall s \in S : M'(s) = M(s) + \underline{t}(s)$$

If we consider M and $\underline{t}$ to be vectors
and '+' to denote vector addition, then

$$M' = M + \underline{t}$$

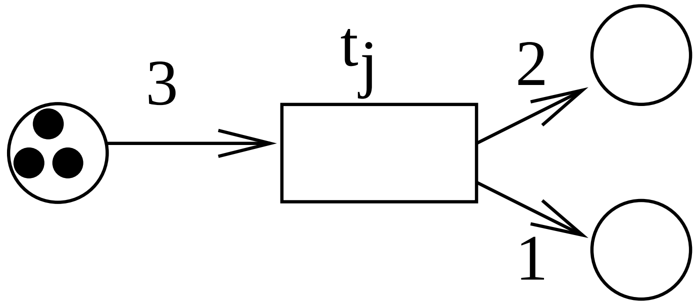
Def.: $\underline{N} : S \times T \to \mathbb{Z}$, $\forall t \in T : \underline{N}(s,t) = \underline{t}(s)$ is called incidence matrix.

For pure nets: W can be computed from $\underline{\mathrm{N}}$.

Invariants

Total number of tokens in $R \subseteq S$ remains constant for $t_j \in T$ iff $\sum_{s \in R} \underline{t}_j(s) = 0$

Example:



$$\underline{c}_R(s) = \begin{cases} 1 \text{ if } s \in R \\ 0 \text{ if } s \notin R \end{cases}$$

is called **characteristic vector** of set $R \subseteq S$.

Sum of tokens can be represented as scalar product:

$$\sum_{s \in R} \underline{t}_j(s) = \underline{t}_j \cdot \underline{c}_R$$

If total number of tokens is constant for all $\underline{t}_j \in T$:

$$\begin{aligned} \underline{t}_1 \cdot \underline{c}_R &= 0 \\ \ldots \; .. &\; .. \\ \underline{t}_n \cdot \underline{c}_R &= 0 \end{aligned}$$

$$\begin{aligned} \underline{t}_1 \cdot \underline{c}_R &= 0 \\ \ldots \quad .. &\quad .. \\ \underline{t}_n \cdot \underline{c}_R &= 0 \end{aligned}$$

Number of tokens is constant for sets of places satisfying

$$\underline{N}^T \cdot \underline{c}_R = 0 \tag{1}$$

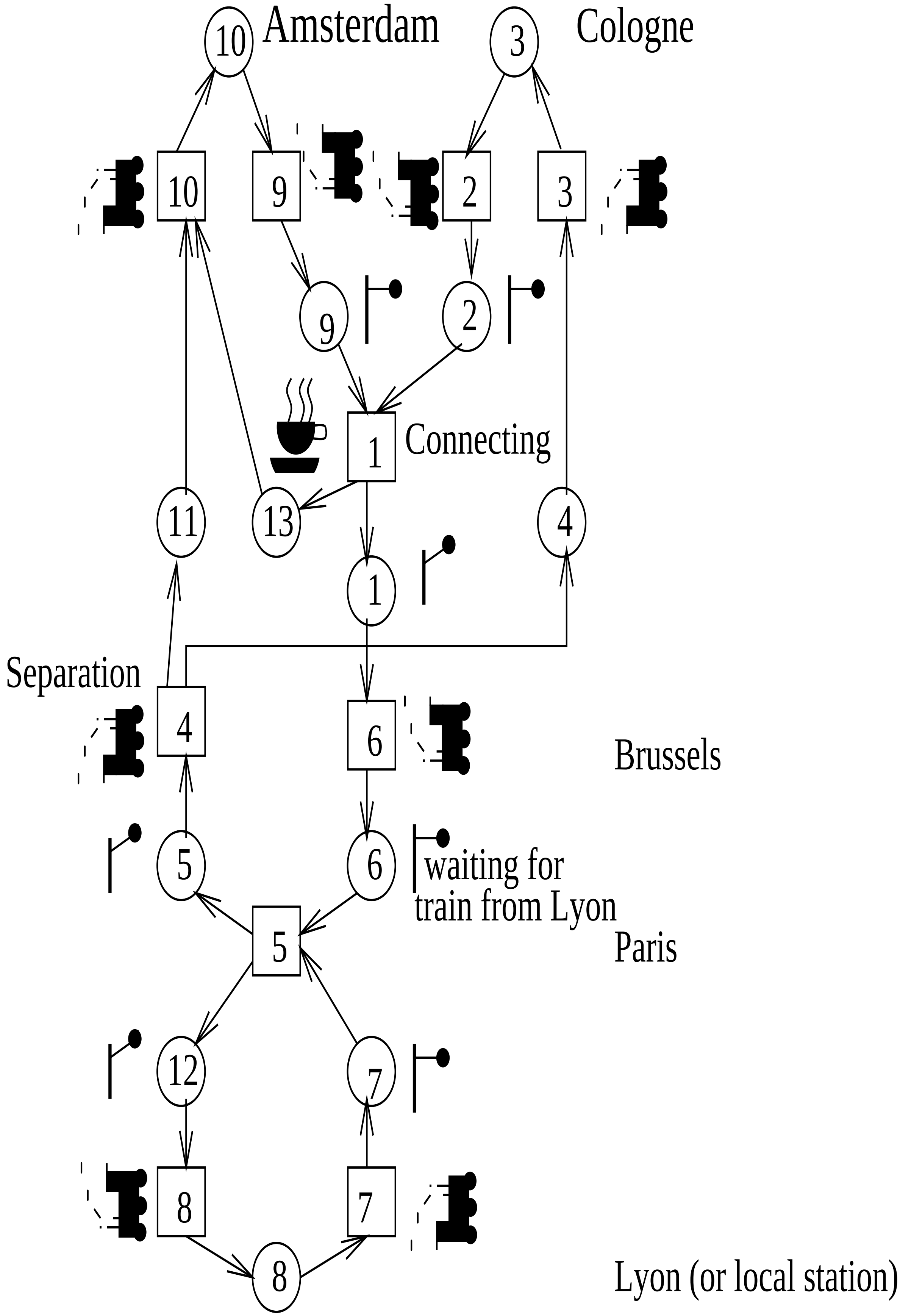
where

$$\underline{N}^T = \begin{pmatrix} \underline{t}_1 \\ .. \\ .. \\ \underline{t}_n \end{pmatrix}$$

Linear equation system.

Only 0 and 1 accepted as results.

Example:



10
Amsterdam
3
Cologne
10
9
2
3
9
2
1
Connecting
11
13
4
1
Separation
4
6
Brussels
5
6
waiting for
train from Lyon
5
Paris
12
7
8
7
8
Lyon (or local station)

Matrix:

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}
t_1	1	-1							-1				1
t_2		1	-1										
t_3			1	-1									
t_4				1	-1						1		
t_5					1	-1	-1					1	
t_6	-1					1							
t_7							1	-1					
t_8								1				-1	
t_9									1	-1			
t_{10}										1	-1		-1
$b_1 = \underline{c}_{R_1}$	1	1	1	1	1	**1**	0	0	0	0	**0**	**0**	**0**
b_2	0	-1	-1	-1	0	**0**	0	0	1	1	**1**	**0**	**0**
$b_3 = \underline{c}_{R_3}$	0	0	0	0	0	**0**	1	1	0	0	**0**	**1**	**0**
$b_4 = \underline{c}_{R_4}$	0	0	0	0	0	**0**	0	0	1	1	**0**	**0**	**1**
$b_1 + b_2 = \underline{c}_{R_2}$	1	0	0	0	1	1	0	0	1	1	1	0	0

S invariants: