2.3 SDL

Language for the specification of distributed systems. Dates back to early 70s; formal semantics late 80s. Language defined by CCITT (Committee Consultatif International Telegraphique et Telephonique).

2.3.1 Language elements

Graphical and textual format.

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

Example:

- ICONconf = Initiate Connection Confirmed
- ICONreq = Initiate Connection Request

PROCESS Initiator

- Start (no state)
- disconnected
- waiting
- connected
- ICONreq
  - CR
  - waiting
- disconnected
- connected
- DR
  - IDISind
  - disconnected
- connected
- DR
  - IDISind
  - connected
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?
**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.

DCL
Zähler Integer;
Datum String;

Zähler := Zähler + 3;

Zähler

(1:10) (11:30) ELSE

Zähler TO

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 = \textit{system};

- Lowest level = \textit{process interaction diagram}: 

- Complete hierarchy:
Inter-process communication
Methods for addressing recipient:

1. **Explicit destination address**
   
   ![](image1.png)

2. **By indirect addressing:**
   
   Recipient determined by context:

   ![](image2.png)

   Signal B will always go to process P2.

3. **Addressing of the channel**
   
   For the same example:

   ![](image3.png)
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

```
SYNTYPE items=INTEGER
CONSTANTS 0:7
ENDSYNTYPE items;

SYNTYPE int=INTEGER
CONSTANTS 0:127
ENDSYNTYPE int;
```

```
Ccoins
[nickel,dime, quarter,half]

CoinInterface
[add]

Cadd

[reject_coin]

CamontDisplay
[amount_entered]

Cemptydisplay
[pretzel_empty, chip_empty, cookie_empty, doughnut_empty]

CspitPurchased
[spit_pretzel, spit_chip, spit_cookie, spit_doughnut]

CspitChange
[spit_nickel, spit_dime]

Crequest
[pur_pretzel, pur_chip, pur_cookie, pur_doughnut, reload_pretzel, reload_chip, reload_cookie, reload_doughnut]

CexaktDisplay
[exact_only]

CspitPurchased
[spit_pretzel, spit_chip, spit_cookie, spit_doughnut]

SIGNAL
[dime, nickel, quarter, half, pur_pretzel, pur_cookie, pur_doughnut, pur_chip, add(int), spit_change(int), amount_entered(int), reject_further_coins, exact_only, accept_coins, reject_coins, spit_dime, spit_nickel, pretzel_empty, spit_pretzel, chip_empty, spit_chip, cookie_empty, spit_cookie, doughnut_empty, spit_doughnut, reload_pretzel, reload_chip, reload_cookie, reload_doughnut]
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);
DCL nchip items:=NITEMS;

VIEWED current int;

Process ChipHandler

pur_wait

pur_chip

VIEW(current) >= PCHIP

ja

sub(PCHIP)

nchip:= nchip-1;

spit_chip

ja

nchip=0

nein

pur_wait

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*):

- Train going to the right
- Track available
- Train going to the left
- Single track
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, $x \in (B \cup E)$.

$x^\bullet := \{y | yFx\}$ 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 $M : S \rightarrow \mathbb{N} \cup \{\omega\}$ 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. $K : S \rightarrow (\mathbb{N} \cup \{\omega\}) \setminus \{0\}$ is called capacity of places ($\omega = \text{unlimited}$).
3. $W : F \rightarrow (\mathbb{N} \setminus \{0\})$ is the weight of edges.
4. $M_0 : S \rightarrow \mathbb{N} \cup \{\omega\}$ is called initial marking.
**Def.** Switching transition results in new marking $M'$, generated from current marking $M$ by:

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

**Example:**

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

Transition $t$ can take place if $t$ is activated.

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

\[(\forall s \in \cdot t : M(s) \geq W(s, t)) \land (\forall s \in t^* : M(s) \leq K(s) - W(t, s))\]
Def.: Let $t : S \rightarrow \mathbb{Z}$ be defined as:

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

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

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

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

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

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

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

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

Example:

\[
\begin{align*}
\mathcal{C}_R(s) &= \begin{cases}
1 & \text{if } s \in R \\
0 & \text{if } s \notin R
\end{cases} \\
\text{is called \textbf{characteristic vector} of set } R \subseteq S.
\end{align*}
\]

Sum of tokens can be represented as scalar product:

\[
\sum_{s \in R} t_j(s) = t_j \cdot \mathcal{C}_R
\]

If total number of tokens is constant for all $t_j \in T$:

\[
t_1 \cdot \mathcal{C}_R = 0 \\
\vdots \\
\vdots \\
\vdots \\
t_n \cdot \mathcal{C}_R = 0
\]
\[ t_1 \cdot c_R = 0 \]

\[ \ldots \ldots \ldots \]

\[ t_n \cdot c_R = 0 \]

Number of tokens is constant for sets of places satisfying

\[
(1) \quad N^T \cdot c_R = 0
\]

where

\[
N^T = \begin{pmatrix}
  t_1 \\
  \vdots \\
  t_n
\end{pmatrix}
\]

Linear equation system.
Only 0 and 1 accepted as results.
Example:

Amsterdam

Cologne

Connecting

Separation

Brussels

Waiting for train from Lyon

Paris

Lyon (or local station)
Matrix:

<table>
<thead>
<tr>
<th></th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
<th>$s_4$</th>
<th>$s_5$</th>
<th>$s_6$</th>
<th>$s_7$</th>
<th>$s_8$</th>
<th>$s_9$</th>
<th>$s_{10}$</th>
<th>$s_{11}$</th>
<th>$s_{12}$</th>
<th>$s_{13}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$t_2$</td>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_3$</td>
<td></td>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_4$</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_6$</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_7$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_8$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_9$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_{10}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>$b_1 = \mathcal{C}r_1$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$b_2$</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$b_3 = \mathcal{C}r_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$b_4 = \mathcal{C}r_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$b_1 + b_2 = \mathcal{C}r_2$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
$S$ invariants:

Brussels

Amsterdam

Paris

Lyon

Cologne