2.4.5 Predicate/Transition nets

Goal: compact representation of complex systems:
Example: *Dining philosophers.*
Boolean variables
d_i: i thinking, e_i: i eating, g_i: spoon i available.

Condition/event net:

Predicate/Transition net:
l(x) and r(x), left resp. right spoon of x.
Properties:

Tokens: individuals
Modell can be extended to arbitrary number of philosophers.
Semantics can be defined by replacing by the equivalent condition/event net.

2.4.6 Evaluation

Appropriate for distributed applications.
Well-known theory for formally proving properties.
For nets presented so far:
- no programming elements,
- no timing information,
- no hierarchy.
2.5 Message sequencing charts (MSCs)
Used for representing activities within distributed systems.
Example: Startup phase of an RS 232 connection:

Also used for representing train schedules.
MSCs only show examples.
Do not show tolerances and constraints.
Not useful for formal techniques.
2.6 UML

UML (Unified Modelling Language)
[Oesterreich97, Fowler98]
Language for modelling object-oriented software development.

UML includes:

1. **State diagrams:**
   UML includes StateCharts.

2. **activity diagrams:**
   activity diagrams = extended Petri nets.
   Extensions: symbols for decisions.
   SDL-like placement

3. **interaction diagrams:**
   Equivalent to message sequencing charts.

4. **class diagrams:**
   Represent inheritance of classes.

5. **package diagrams:**
   correspond to module charts of **StateMate**.

6. **use cases:** describe typical applications.

Checking of consistency?
2.7 ADA

Language designed for DoD.
Current version: ADA 1995
(object oriented extension of original language)
Salient feature:
Rendez-vous model of communication
(Processes have to meet to exchange information):
Example:

```ada
TASK screen_output IS
  ENTRY call_ch(val:character; x, y: integer);
  ENTRY call_int(z, x, y: integer);
END screen_out;
TASK BODY screen_output IS
...
SELECT
  ACCEPT call_ch ... DO ..
END call_ch;
OR
  ACCEPT call_int ... DO ..
END call_int;
END SELECT;
...

Sending a message:

BEGIN
  screen_out.call(‘Z’,10,20);
  EXCEPTION
    WHEN tasking_error => (exception handling)
END;
```
2.8 Java
Initially designed for embedded systems.
Advantages of using Java for embedded systems:

- **object orientation**
- **supports multiple threads**
- Java is **simple**.
- **short design cycles**
- Java is **compact**. Basic interpreter: 40 kBytes, Thread-Support, libraries: additional 175 kBytes.
- Java is **robust**: No memory leaks.
- Java is **platform independent**
- **dynamic loading of classes**.
- Java is **safe**: security checks etc.

Problems:

- **automatic garbage collection**
- **Unspecified dispatcher**

Working groups are trying to solve the problems (Nilsen and others).
Different subsets discussed

+Java 3D = 3D Java

+ Enterprise-Tools = Enterprise Java

Considered for set top boxes, Mercedes-Benz cars etc.
2.9 VHDL

VHDL = VHSIC Hardware Description Language
VHSIC = very high speed integrated circuit
Definition started by DoD in 1980.
1984: first version of the language defined.

Salient features:
- Syntax based on ADA
- flexible definition of value sets
- supports bitvectors
- supports time and time units
- processes are static, non-terminating and flat
- 2-phase simulation semantics to avoid non-determinism
- includes configuration mechanism
- Parallelism of hardware modelled by processes.
- Several processes per hardware component possible.
Example: full adder

![Full Adder Diagram]

Header:

ENTITY full_adder IS
  PORT (a, b, carry_in: IN Bit;
        sum, carry_out: OUT Bit);
END full_adder;

Behavioural body:

ARCHITECTURE behavior OF full_adder IS
  SIGNAL s: Bit;
BEGIN
  s <= a xor b AFTER 5 Ns;
  sum <= s xor carry_in AFTER 5 Ns;
  carry_out<= (a and b) or
               (s and carry_in) after 10 Ns;
END behavior;
ARCHITECTURE structure OF full_adder IS
  COMPONENT half_adder
    PORT (i1, i2:IN Bit; carry:OUT Bit; sum:OUT Bit);
  END component;
  COMPONENT or_gate
    PORT (i1, i2:IN Bit; o:OUT Bit);
  END component;
  SIGNAL x, y, z: Bit;
BEGIN
  s1: half_adder PORT MAP (a, b, x, y);
  s2: half_adder PORT MAP (y, carry_in, z, sum);
  s3: or_gate PORT MAP (x, z, carry_out);
END structure;
Simulation results:

A: 

B: 

CARRY_IN: 

SUM: 

CARRY_OUT: 

ns
0 50 100 150 200 250 300 350 400
SpecCharts [Gajski94] is a StateChart variant using VHDL to denote computations.

Characteristics of StateCharts:

- Developed for embedded system specification [NVG92]
- PSM (program-state machine) model + VHDL

**Characteristics supported**
- Behavioral hierarchy: sequential/concurrent behaviors
- State transitions: TOC (transition on completion) arcs
- Communication: shared memory, message passing
- Exceptions: TI (transition immediately) arcs

**Characteristics similar to VHDL**
- Programming constructs
- Structural hierarchy
- Synchronization and Timing

Two types of transitions: *transition on completion* and *transition immediately*.

SpecCharts descriptions can be easily translated into VHDL.
2. Estelle
Scope: description of communication protocols.
Standard defined by ISO.
Scope similar to SDL.
Attempts to integrate Estelle with SDL failed.
Communication based on channels + FIFO buffers.
FSM-based model of components.
Textual representation only. Example:

state disconnected, suspended, connected;
trans
{1} from disconnected to suspended
    when User.ICONreq
        begin output PDU.CR end;
{2} from suspended to connected
    when PDU.CC
        begin output User.ICONconf end;
{3} from suspended to disconnected
    when PDU.DR
        begin output User.IDISind end;
{4} from disconnected to connected ...
3. CSP [Hoare]

CSP = communicating sequential processes
One of the first languages containing language elements for describing interprocess communication. Communication based on named channels.

Example:

```
PROCESS A                PROCESS B

........                ........
VAR a ..                VAR b ..
a := 3;                  ...
c!a ;                  c?b  -- input from channel c
END;                  END;
```

Processes waiting for meeting at rendez-vous point.

Blocking communication.

CSP formed the basis for OCCAM (language for transputers [INMOS]).

Earlier transputers: 4 hardware communication channels
Within a single transputer, communication is handled by the operating system.

More recent transputers: virtual channels. Communic. with any number of processors.
4. **Verilog**
   Another hardware description language.
   Less flexible than VHDL, but usually simulates faster.
   Very popular in the US.

5. **Z**
   language for algebraic specification.

6. **LOTOS**
   Another language for algebraic specification.

7. **Esterel**
   Yet another language for algebraic specification.

8. **Silage**
   Special functional language for signal processing.

9. **HardwareC**
   Hardware description language with syntax similar to C.
   Simple extensions for describing concurrent and parallel execution and bitvectors.
## Summary

### Embedded System Features

<table>
<thead>
<tr>
<th>Language</th>
<th>State Transitions</th>
<th>Behavioral Hierarchy</th>
<th>Concurrency</th>
<th>Program Constructs</th>
<th>Exceptions</th>
<th>Behavioral Completion</th>
</tr>
</thead>
<tbody>
<tr>
<td>VHDL</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Verilog</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Esterel</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>SDL</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>CSP</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Statecharts</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>SpecCharts</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

- **Feature fully supported**
- **Feature partially supported**
- **Feature not supported**

### Remarks:
- Esterel can be used to describe state transitions.
- SDL contains programming language elements.
- Contains only criteria for which SpecCharts looks good (e.g. structural hierarchy not included, no dynamic process creation considered).
### Comparison according to Niemann:

<table>
<thead>
<tr>
<th></th>
<th>Standardization</th>
<th>Availability of Tools</th>
<th>Model Executability</th>
<th>Formal Analysis</th>
<th>Non-determinism</th>
<th>Timing Aspects</th>
<th>Exceptions</th>
<th>Synchron./Communication</th>
<th>Concurrency</th>
<th>Behavioural Completion</th>
<th>State-Transitions</th>
<th>Programming Constructs</th>
<th>Behavioural Hierarchy</th>
<th>Structural Hierarchy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lotos</td>
<td>+</td>
<td>0</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SDL</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>0</td>
<td>+3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Esterel</td>
<td>-</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>StateCharts</td>
<td>-</td>
<td>0</td>
<td>+</td>
<td>-1</td>
<td>-</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>High-Level Petri Net</td>
<td>-</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>-4</td>
<td>+6</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VHDL</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-2</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Verilog</td>
<td>-7</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SpecCharts</td>
<td>-</td>
<td>0</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>HardwareC</td>
<td>-</td>
<td>0</td>
<td>+</td>
<td>-</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>CSP/Occam</td>
<td>-</td>
<td>0</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Silage</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Remarks:

1. Formal semantics of StateCharts now available.
2. Possible formal semantics of VHDL described in book by Delgado-Kloos (ed.)
3. SDL supports only mix of structural and behavioural hierarchy.
4. Non-determinism included in Petri net model.
5. Same as for SDL.
6. There are timed Petri nets. Formal properties?
   Insufficient knowledge about variants of Petri nets.
7. IEEE standard for Verilog exists.

→ no single language that meets all requirements.
Mix of languages used in practice.

Translation between languages proposed by several researchers.