Computational Models

- The concept of a computational model
- Basic computational models
- The von Neumann computational model
- Key concepts relating to computational models
The concept of a computational model

- Model: Foundation or paradigm
- Level of abstraction
- Computational Model
  - Computer architecture
  - Computer language

Interpretation of concept of a computational model

- Computational Model
  - (1) Basic items of computation
  - (2) Problem description model
  - (3) Execution model
(1) **Basic items of computation**

- e.g. data, object, argument and functions, element of sets and the predicates

(2) **Problem description model**

- Problem description model
  - Style
  - Method
- Problem description style
  - Procedural
    ✤ (algorithm for solving the problem is stated)
  - Declarative
    ✤ (all the facts and relationships relevant to the given problem is stated)
**Problem description style**

(e.g.)

Calculate n factorial, n!

- **Procedural style**
  
  int nfac (int n) {
  int fac = 1;
  if (n > 0)
    for ( int i = 2; i <= n; i++ )  
  fac = fac * i;
  return fac; }

- **Declarative style**
  
  fac (0) = 1;
  fac ( n>0 ) = n * fac ( n-1 );

**Declarative style**

- Using functions
  - in a model called applicative, (Pure Lisp)

- Using predicates
  - in a model called predicate logic-based, (Prolog)
Problem description method

- Procedural method
  - how a solution of the given problem has to be described
    - e.g. sequence of instructions
- Declarative method
  - how the problem itself has to be described
    - e.g. set of functions

(3) Execution Model

- Interpretation of how to perform the computation related to the problem description method
- Execution semantics
  - rule that prescribes how a single execution step is to be performed
- Control of the execution sequence
  - ordering of execution sequence
Execution semantic

- State transition semantics
  - Turing model
  - von Neumann model
  - object-based model
- Dataflow semantics
  - dataflow model
- Reduction semantics
  - applicative model (Pure Lisp)
- SLD-resolution
  - Predicate logic-based model (Prolog)

Control of the execution sequence

- Control driven
  - assumed that there exists a program consisting of sequence of instructions
  - execution sequence is then implicitly given by the order of the instruction
  - explicit control instructions to change the order
- Data driven
  - an operation is activated as soon as all the needed input data is available (eager evaluation)
- Demand driven
  - an operation is activated only when execution is needed to achieve the final result
Concepts of computational model, programming language, and architecture

Typical Evolution

- Computation model
- Corresponding programming language
- Corresponding architecture
Basic computational models

- Turing
- von Neumann
- object based
- dataflow
- applicative
- predicate logic based

<table>
<thead>
<tr>
<th>Computational model</th>
<th>Turing</th>
<th>von Neumann</th>
<th>Object-based</th>
<th>Dataflow</th>
<th>Applicative</th>
<th>Predicate logic-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic items of computation</td>
<td>Elements of the top symbol set are mapped defined on them</td>
<td>Data assigned to named entities (variables) Operations performed on data</td>
<td>Objects, which can be manipulated by a set of messages or computations</td>
<td>Data assigned to named entities (variables) Operations performed on data</td>
<td>Arguments of functions to their arguments</td>
<td>Elements of data Predicates declared to them</td>
</tr>
<tr>
<td>Style</td>
<td>Procedural</td>
<td>Declarative</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Input data</td>
<td>Input data</td>
<td>Initial states of objects</td>
<td>Input data</td>
<td>Arguments of the functions to be evaluated</td>
<td>Goal statement</td>
</tr>
<tr>
<td>Interpretation of how to perform the computation</td>
<td>Transition function δ</td>
<td>Sequence of instructions</td>
<td>Sequence of messages sent to created objects</td>
<td>Flow data through dataflow diagram and, as a consequence, output data is produced</td>
<td>Evaluated functions</td>
<td>Logical inference</td>
</tr>
<tr>
<td>Advanced execution model</td>
<td>State transition semantics</td>
<td>Control driven</td>
<td>Control driven</td>
<td>Control driven</td>
<td>Reduction semantics</td>
<td>SLD-resolution</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Sequential</td>
<td>Parallel</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1.13 A summary of the key features of the basic computational models.
The von Neumann computational model

- Basic items of computation are data
  - variables (named data entities)
  - memory or register locations whose addresses correspond to the names of the variables
  - data container
  - multiple assignments of data to variables are allowed
- Problem description model is procedural (sequence of instructions)
- Execution model is state transition semantics
  - Finite State Machine

von Neumann model vs. finite state machine

- As far as execution is concerned the von Neumann model behaves like a finite state machine (FSM)
- FSM = \{ I, G, \delta, G_0, G_f \}
- I: the input alphabet, given as the set of the instructions
- G: the set of the state (global), data state space D, control state space C, flags state space F, \( G = D \times C \times F \)
- \( \delta \): the transition function: \( \delta: I \times G \rightarrow G \)
- \( G_0 \): the initial state
- \( G_f \): the final state
Key characteristics of the von Neumann model

- Consequences of multiple assignments of data
  - history sensitive
  - side effects
- Consequences of control-driven execution
  - computation is basically a sequential one
  - ++ easily be implemented
- Related language
  - allow declaration of variables with multiple assignments
  - provide a proper set of control statements to implement the control-driven mode of execution

Extensions of the von Neumann computational model

- new abstraction of parallel execution
- communication mechanism allows the transfer of data between executable units
  - unprotected shared (global) variables
  - shared variables protected by modules or monitors
  - message passing, and
  - rendezvous
- synchronization mechanism
  - semaphores
  - signals
  - events
  - queues
  - barrier synchronization
Key concepts relating to computational models

- **Granularity**
  - complexity of the items of computation
  - size
  - fine-grained
  - middle-grained
  - coarse-grained

- **Typing**
  - data based type ~ Tagged
  - object based type (object classes)