1.6. Imperative Constructs

1.6.1. Statements and State Changes

Unit value
stmt: Unit → Unit
stmt()

- The Unit clause, in a sense, denotes “an underlying state”
- which we, for simplicity, can consider as
- a mapping from identifiers of declared variables into their values.

- Statements accept no arguments and, usually, operate on the state
- through “reading” the value(s) of declared variables and
- through “writing”, i.e., assigning values to such declared variables.

- Statement execution thus changes the state (of declared variables).

- Unit → Unit designates a function from states to states.

- Statements, stmt, denote state-to-state changing functions.

- Affixing () as an “only” arguments to a function “means” that () is an argument of type Unit.

1.6.2. Variables and Assignment

0. variable v:Type := expression
1. v := expr

1.6.3. Statement Sequences and skip

2. skip
3. stm_1;stm_2;...;stm_n

1.6.4. Imperative Conditionals

4. if expr then stm_c else stm_a end
5. case e of: p_1→S_1(p_1),...,p_n→S_n(p_n) end

6. while expr do stmt end
7. do stmt until expr end

1.6.5. Iterative Conditionals

8. for i in list · P(list(i)) do S(list(i)) end
9. for e in set · P(e) do S(e) end

1.6.6. Iterative Sequencing
1.7. Process Constructs

1.7.1. Process Channels

Let A, B and C stand for three types of (channel) messages and \( i: \text{IIdx}, j: \text{JIdx} \) for channel array indexes, then:

\[
\text{channel } c: A \{ k[i]: \text{IIdx} \}: B \{ \text{ch}[i,j]: \text{IIdx}, j: \text{JIdx} \}: C
\]

Example

Modelling Connected Links and Hubs:

• Examples (17–20) are building up a model of one form of meaning of a transport net.
  – We model the movement of vehicles around hubs and links.
  – We think of each hub, each link and each vehicle to be a process.
  – These processes communicate via channels.

• We need some auxiliary quantities in order to be able to express subsequent channel declarations.

• Given that we assume a net, \( n: N \) and a set \( vs: V \), of vehicles,
  – the sets of hubs, \( hs \), and links, \( ls \) of the net;
  – the set, \( ivhs \), of indices between vehicles and hubs,
  – the set, \( ivls \), of indices between vehicles and links,
  – the set, \( ihls \), of indices between hubs and links.

value

\[
\begin{align*}
\text{hs: } H-\text{set} &= \text{obs}_H(n), \text{ls: } L-\text{set} = \text{obs}_L(n) \\
\text{his: } Hl-\text{set} &= \{ \text{obs}_H(h) | h: H \in hs \}, \text{lis: } Ll-\text{set} = \{ \text{obs}_L(l) | l: L \in ls \}, \\
\text{ivhs: } IVH-\text{set} &= \{ (\text{obs}_V(v), \text{obs}_H(h)) | v: V, h: H \in vs \land h \in hs \}, \\
\text{ivls: } IVL-\text{set} &= \{ (\text{obs}_V(v), \text{obs}_L(l)) | v: V, l: L \in vs \land l \in ls \}, \\
\text{ihls: } IHL-\text{set} &= \{ (hi, li) | h: H, (hi, li): IHL, h: H \in hs \land hi = \text{obs}_H(h) \land li \in \text{obs}_L(ls(h)) \}
\end{align*}
\]
1.7. Process Definitions

- A process definition is a function definition.
- The below signatures are just examples.
- They emphasise that process functions must somehow express,
  - in their signature,
  - via which channels they wish to engage in input and output events.
- Processes $P$ and $Q$ are to interact, and to do so “ad infinitum”.
- Processes $R$ and $S$ are to interact, and to do so “once”, and then yielding $B$, respectively $D$ values.

Example 18 . . . . . Communicating Hubs, Links and Vehicles:

- Hubs interact with links and vehicles:
  - with all immediately adjacent links,
  - and with potentially all vehicles.
- Links interact with hubs and vehicles:
  - with both adjacent hubs,
  - and with potentially all vehicles.
- Vehicles interact with hubs and links:
  - with potentially all hubs.
  - and with potentially all links.
Example 19
Transport Net: Modeling Transport Nets

1.7.3. Process Composition

1.7.2. Process Constructs

1.7.1. Process Composition

1.7.0. Process Composition

Parallel composition

\[ \{ P \parallel Q \} \]

Concurrent parallel composition

\[ \{ P \parallel Q \} \]

Sequential parallel composition

\[ \{ P \parallel Q \} \]

Non-deterministic parallel composition

\[ \{ P \parallel Q \} \]

Non-deterministic parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]

Nested parallel composition

\[ \{ P \parallel Q \} \]
The three auxiliary processes:

- $E_{li}$ update the hub with respect to (wrt.) connected link, $li$, information $m$,
- $E_{vi}$ update the hub with wrt. vehicle, $vi$, information $m$,
- $E_{own}$ update the hub with wrt. whatever the hub so decides. An example could be signalling dependent on previous link-to-hub communicated information, say about traffic density.

$E_{li}: LI \rightarrow M \rightarrow H \rightarrow H$
$E_{vi}: VI \rightarrow M \rightarrow H \rightarrow H$
$E_{own}: H \rightarrow H$

The student is encouraged to sketch/define similarly schematic link and vehicle processes.

End of Example 19

Example 20

Modelling Vehicle Movements:

Whereas hubs and links are modelled as basically static, passive, that is, inert, processes we shall consider vehicles to be “highly” dynamic, active processes.

We assume that a vehicle possesses knowledge about the road net.
- The road net is here abstracted as an awareness of
  - which links, by their link identifiers,
  - are connected to any given hub, designated by its hub identifier,
  - the length of the link,
  - and the hub to which the link is connected “at the other end”, also by its hub identifier.

A vehicle is further modelled by its current position on the net in terms of either hub or link positions
- designated by appropriate identifiers
- and, when “on a link” “how far down the link”, by a measure of a fraction of the total length of the link, the vehicle has progressed.

Let $c$ and $k[i]$ designate channels of type $A$
and $e$ expression values of type $A$, then:

- [1] $c?, k[i]?$ input $A$ value
- [2] $c!e, k[i]!e$ output $A$ value
- [3] $P: ... \rightarrow \text{out} c ..., P(...) \equiv ... c!e ...$ offer an $A$ value,
- [4] $Q: ... \rightarrow \text{in} c ..., Q(...) \equiv ... c? ...$ accept an $A$ value
- [5] $S: ... \rightarrow ..., S(...) = P(...)||Q(...) \equiv ...$ synchronise and communicate

- [5] expresses the willingness of a process to engage in an event that
  - [1,3] “reads” an input, respectively

Net  $= HI \rightarrow m \rightarrow (LI \rightarrow m \rightarrow HI)$
Pos  $= \text{atH}\mid \text{onL}$
atH  $= \text{mk}\_\text{atH}(hi:HI)$
onL  $= \text{mk}\_\text{onL}(fhi:HI,li:LI,f:F,thi:HI)$
F   $= \{|f: \text{Real} | 0 \leq f \leq 1\}$
• We first assume that the vehicle is at a hub.
• There are now two possibilities ([1–2] versus [4–8]).
  – Either the vehicle remains at that hub
    » [1] which is expressed by some non-deterministic `wait`
    » [2] followed by a resumption of being that vehicle at that location.
  – [3] Or the vehicle (driver) decides to “move on”:
    » [5] Onto a link, \( li \),
    » [4] among the links, \( lis \), emanating from the hub,
    » [6] and towards a next hub, \( hi \).
  – [4,6] The \( lis \) and \( hi \) quantities are obtained from the vehicle’s own knowledge of the net.
  – [7] The hub and the chosen link are notified by the vehicle of its leaving the hub and entering the link,
  – [8] whereupon the vehicle resumes its being a vehicle at the initial location on the chosen link.

• We then assume that the vehicle is on a link and at a certain distance “down”, \( f \), that link.
• There are now two possibilities ([1–2] versus [4–7]).
  – Either the vehicle remains at that hub
    » [1] which is expressed by some non-deterministic `wait`
    » [2] followed by a resumption of being that vehicle at that location.
  – [3] Or the vehicle (driver) decides to “move on”:
    » [4] among the links, \( lis \), emanating from the hub,
    » [6] and towards a next hub, \( hi \).
  – [4,6] The \( lis \) and \( hi \) quantities are obtained from the vehicle’s own knowledge of the net.
  – [7] The hub and the chosen link are notified by the vehicle of its leaving the hub and entering the link,
  – [8] whereupon the vehicle resumes its being a vehicle at the initial location on the chosen link.

• The vehicle chooses between these two possibilities by an internal non-deterministic choice ([3]).

**type**
\[ M \equiv mk_LH(li:LI,hi:HI) \mid mk_HL(hi:HL,li:LI) \]

**value**
\[ \delta: Real = move(h,f) \text{ axiom } 0 < \delta < 1 \]
vehicle\((vi)(mk_{onL}(hi,li,f,hi),net)(v) \equiv \]
\[ 1' \text{ (wait)} \]
\[ 2' \text{ vehicle}(vi)(mk_{onL}(hi,li,f,hi),net)(v)) \]
\[ 3' \text{ end} \]
\[ 4' \text{ case } f \text{ of} \]
\[ 5' 0 \rightarrow ((vl[vi,hi]mk_LH(li,hi))\text{vh}[vi,li]mk_LH(li,hi)); \]
\[ 6' \text{ vehicle}(vi)(mk_{atH}(hi),net)(v)), \]
\[ 7' \text{ end )} \]
move: \( H \times F \rightarrow F \)

*End of Example 20*
1.8. **Simple RSL Specifications**

- Besides the above constructs RSL also possesses module-oriented constructs.

  - scheme, class and object

- We shall not cover these here.

- An RSL specification is then simply a sequence of one or more clusters of:
  - zero, one or more sort and/or type definitions,
  - zero, one or more variable declarations,
  - zero, one or more channel declarations,
  - zero, one or more value definitions (including functions) and
  - zero, one or more axioms.

- We can illustrate these specification components schematically:

- The ordering of these clauses is immaterial.

- Intuitively the meaning of these definitions and declarations are the following.

  - The **type** clause introduces a number of user-defined type names;
    * the type names are visible anywhere in the specification;
    * and either denote sorts or concrete types.

  - The **variable** clause declares some variable names;
    * a variable name denote some value of decalred type;
    * the variable names are visible anywhere in the specification:
      - assigned to (‘written’) or
      - values ‘read’.

  - The **channel** clause declares some channel names;
    * either simple channels or arrays of channels of some type;
    * the channel names are visible anywhere in the specification.

- The **value** clause bind (constant) values to value names.

  - These value names are visible anywhere in the specification.

- The specification

  **type**

  A, B, C, D, E, F, G

  **value**

  va:A, vb:B, ... ve:E

  Hf = A-set, Hi = A-infset

  J = B×C×...×D

  Kf = E', Ki = E''

  L = F ↦ G

  Mt = J → Kf, Mp = J ⊆ Ki

  N == alpha | beta | ... | omega

  O == mk_Hf(as:Hf)

  P = Hf | Kf | L | ...

  variable

  vhf:Hf := ()

  channel

  chf:F, chg:G, {chb[i] | i:A} : B

- The ordering of these clauses is immaterial.

  - The **type** clause introduces a number of user-defined type names;
    * the type names are visible anywhere in the specification;
    * and either denote sorts or concrete types.

  - The **variable** clause declares some variable names;
    * a variable name denote some value of decalred type;
    * the variable names are visible anywhere in the specification:
      - assigned to (‘written’) or
      - values ‘read’.

  - The **channel** clause declares some channel names;
    * either simple channels or arrays of channels of some type;
    * the channel names are visible anywhere in the specification.

- The **value** clause bind (constant) values to value names.

  - These value names are visible anywhere in the specification.

  - The specification

  **type**

  A

  **value**

  a:A

  f1: A → B, f2: C ⊆ D

  f1(a) ≡ E_f1(a)

  f2: E → in/out chf F

  f2(e) ≡ E_f2(e)

  f3: Unit → in chf out chg Unit

  P_i(f1, va),

  P_i(f2, vb),

  P_i(f3, ve)
Example 21 A Neat Little “System”:

- We present a self-contained specification of a simple system:
  - The system models
    * vehicles moving along a net, vehicle,
    * the recording of vehicles entering links, enter_sensor,
    * the recording of vehicles leaving links, leave_sensor, and
    * the road_pricing payment of a vehicle having traversed (entered and left) a link.
  - Note
    * that vehicles only pay when completing a link traversal;
    * that ‘road pricing’ only commences once a vehicle enters the first link after possibly having left an earlier link (and hub); and
    * that no road_pricing payment is imposed on vehicles entering, staying-in (or at) and leaving hubs.

- We assume the following:
  * that each link is somehow associated with two pairs of sensors:
    · a pair of enter and leave sensors at one end, and
    · a pair of enter and leave sensors at the other end;
  * a road pricing process
    · which records pairs of link enterings and leavings,
    · first one, then, after any time interval, the other,
    · with leavings leading to debiting of traversal fees;

- Our first specification
  - define types,
  - assume a net value,
  - declare channels and
  - state signatures of all processes.

• ves stand for vehicle entering (link) sensor channels,
• vls stand for vehicle leaving (link) sensor channels,
• rp stand for ‘road pricing’ channel
• enter_sensor(hi,li) stand for vehicle entering [sensor] process from hub hi to link (li).
• leave_sensor(li,hi) stand for vehicle leaving [sensor] process from link li to hub (hi).
• road_pricing() stand for the unique ‘road pricing’ process.
• vehicle(vi)(...) stand for the vehicle vi process.
• To understand the sensor behaviours let us review the vehicle behaviour.

• In the vehicle behaviour defined in Example 20, in two parts, Slide 381 and Slide 383 we focus on the events
  – [7] where the vehicle enters a link, respectively
  – [5'] where the vehicle leaves a link.

• These are summarised in the schematic reproduction of the vehicle behaviour description.
  – We redirect the interactions between vehicles and links to become
    – interactions between vehicles and enter and leave sensors.

value
\[ \delta: \text{Real} = \text{move}(h,f) \text{ axiom } 0 < \delta \leq 1 \]
move: \( H \times F \to F \)

• As mentioned on Slide 389 link behaviours are associated with two pairs of sensors:
  – a pair of enter and leave sensors at one end, and
  – a pair of enter and leave sensors at the other end;

value
\[
\text{link}(li)(l) \equiv \\
\text{let } \{hi,li\} = \text{obs\_Hls}(l) \text{ in} \\
\text{enter\_sensor}(hi,li) \parallel \text{leave\_sensor}(li,hi) \parallel \\
\text{enter\_sensor}(hi,li) \parallel \text{leave\_sensor}(li,hi) \text{ end} \\
\text{enter\_sensor}(hi,li) \equiv \\
\text{let } vi = \text{ves}[hi,li]? \text{ in } \text{rp!mk\_Enter\_LL}(vi,li); \text{enter\_sensor}(hi,li) \text{ end} \\
\text{leave\_sensor}(li,hi) \equiv \\
\text{let } vi = \text{ves}[li,hi]? \text{ in } \text{rp!mk\_Leave\_LL}(vi,li); \text{enter\_sensor}(li,hi) \text{ end}
\]

• The LVS component of the road_pricing behaviour serves,
  – to record whether the movement of a vehicles “originates” along a link or not.

• Otherwise we leave it to the student to carefully read the formulas.

value
\[
\text{payment}: \text{VI} \times \text{LI} \to (\text{ACC} \times \text{FEE}) \to \text{ACC} \\
\text{payment}(vi,li)(\text{fee},\text{acc}) \equiv \\
\text{let } \text{bal} = \text{if } vi \in \text{dom acc} \text{ then } \text{add}(\text{acc}(vi),\text{fee(li)}) \text{ else } \text{fee(li)} \text{ end} \\
\text{in } \text{acc} \uparrow [vi \mapsto \text{bal}] \text{ end} \\
\text{add}: \text{FEE} \times \text{Bal} \to \text{Bal} [\text{add fee to balance}]
\]
road\_pricing(lvs,fee,acc) \equiv in\ rp

let m = rp? in

case m of

\begin{itemize}
\item[\text{mk\_Enter\_Ll}(vi,li) \rightarrow]
road\_pricing(lvs[\{li\mapsto lvs(li)\cup\{vi\} ],fee,acc),

\item[\text{mk\_Leave\_Ll}(vi,li) \rightarrow]
let lvs' = if vi \in lvs(li) then lvs[\{li\mapsto lvs(li)\setminus\{vi\} ] else lvs end,
acc' = payment(vi,li)(fee,acc) in
road\_pricing(lvs',fee,acc)
\end{itemize}
end end end

\end{example}

\end{lecture}

\end{document}