1. **An RSL Primer**

1.1. **Types**

1.1.1. **Type Expressions**

- Type expressions are expressions whose values are types, that is, possibly infinite sets of values (of “that” type).

1.1.1.1. **Atomic Types**

- Atomic types have (atomic) values.
- That is, values which we consider to have no proper constituent (sub-)values,
- i.e., cannot, to us, be meaningfully “taken apart”.

---

**Example 1**

Basic Net Attributes:

- For safe, uncluttered traffic, hubs and links can ‘carry’ a maximum of vehicles.
- Links have lengths. (We ignore hub (traversal) lengths.)
- One can calculate whether a link is a two-way link.

---

**type**

1. The Boolean type of truth values `false` and `true`.
2. The integer type on integers ..., –2, –1, 0, 1, 2, ...
3. The natural number type of positive integer values 0, 1, 2, ...
4. The real number type of real values, i.e., values whose numerals can be written as an integer, followed by a period (“.”), followed by a natural number (the fraction).
5. The character type of character values ‘a’, ‘b’, ...
6. The text type of character string values ‘aa’, ‘aaa’, ..., ‘abc’, ...

---
1. An RSL Primer

1.1. Types

1.1.1. Type Expressions

1.1.1.1. Atomic Types

\[
\text{type } \begin{align*}
\text{MAX} &= \text{Nat} \\
\text{LEN} &= \text{Real} \\
\text{is\_Two\_Way\_Link} &= \text{Bool}
\end{align*}
\]

value

\[
\begin{align*}
\text{obs\_Max}: (H|L) &\to \text{MAX} \\
\text{obs\_Len}: L &\to \text{LEN} \\
\text{is\_Two\_Way\_Link}: L &\to \text{is\_Two\_Way\_Link} \\
\text{is\_Two\_Way\_Link}(l) &\equiv \exists l\sigma: L \Sigma \cdot l\sigma \in \text{obs\_H\Sigma}(l) \land \text{card } l\sigma = 2
\end{align*}
\]

\textbf{End of Example 1}

Example 2

\textbf{Composite Net Type Expressions:}

- The type clauses of function signatures:

\[
\begin{align*}
\text{value} \\
\text{f}: A &\to B
\end{align*}
\]

- often have the type expressions \(A\) and/or \(B\)

- be composite type expressions:

\[
\begin{align*}
\text{value} \\
\text{obs\_Hls}: L &\to \text{HI-set} \\
\text{obs\_Lls}: H &\to \text{LI-set} \\
\text{obs\_HΣ}: H &\to \text{HT-set} \\
\text{set\_HΣ}: H \times H\Sigma &\to H
\end{align*}
\]

\textbf{End of Example 2}
1.1.2. Type Definitions

1.1.2.1. Concrete Types

- Types can be concrete
- in which case the structure of the type
- is specified by type expressions:

\[
\text{type} \quad A = \text{Type}\_\text{expr}
\]

schematic examples:

\[
\begin{align*}
A1 &= \text{B1-set}, \quad A2 = \text{B1-infset} \\
A3 &= \text{B2} \times \text{C1} \times \text{D1} \\
B1 &= \text{E}^*, \quad B2 = \text{E}^\omega \\
C1 &= \text{F}^m \to \text{G} \\
D1 &= \text{H} \to \text{J}, \quad D2 = \text{H} \sim \to \text{J} \\
K &= \text{L} | \text{M}
\end{align*}
\]

**Composite Net Types:**

- There are many ways in which nets can be concretely modelled:

**Sorts + Observers + Axioms:** First we show an example of type definitions without right-hand side, that is, of sort definitions.

From a net one can observe many things.

Of the things we focus on are the hubs and the links.

A net contains two or more hubs and one or more links.

\[
\begin{align*}
\text{type} \\
\left[\text{sorts}\right] \quad N_\alpha, \text{H, L, HI, LI} \\
\text{value} \\
\text{obs}_\text{Hs}: N_\alpha \to \text{H-set} \\
\text{obs}_\text{Ls}: N_\alpha \to \text{L-set} \\
\text{axiom} \\
\forall n: N_\alpha \cdot \text{card obs}_\text{Hs}(n) > 0 \Rightarrow \text{card obs}_\text{Ls}(n) \geq 1 \\
\end{align*}
\]

**Cartesians + Wellformedness:** A net can be considered as a Cartesian of sets of two or more hubs and sets of one or more links.

\[
\begin{align*}
\text{type} \\
\left[\text{sorts}\right] \quad \text{H, L} \\
N_\beta &= \text{H-set} \times \text{L-set} \\
\text{value} \\
\text{wf}_{N_\beta}: N_\beta \to \text{Bool} \\
\text{wf}_{N_\beta}(\text{hs,ls}) &\equiv \text{card hs} > 1 \Rightarrow \text{card ls} > 0 \\
\text{inject}_{N_\beta}: N_\alpha \sim N_\beta \preceq \text{wf}_{N_\beta}(\text{hs,ls}) \\
\text{inject}_{N_\beta}(n_\alpha) &\equiv (\text{obs}_\text{Hs}(n_\alpha), \text{obs}_\text{Ls}(n_\alpha))
\end{align*}
\]

**Cartesians + Maps + Wellformedness:** Or a net can be described

a as a triple of b-c-d:

- b hubs (modelled as a map from hub identifiers to hubs),
- c links (modelled as a map from link identifiers to links), and
- d a graph from hub identifiers \(h_i\) connected to maps from identifiers \(l_{ij}\) of linked hubs to the identifiers \(h_j\) of link connected hubs.
type

[sorts] H, HI, L, LI
[a] N_3 = HUBS \times LINKS \times GRAPH
[b] HUBS = HI \times H
[c] LINKS = LI \times L
[d] GRAPH = HI \times (LI - m > HI)

[b,c] hs:HUBS and ls:LINKS are maps from hub (link) identifiers to hubs (links) where one can still observe these identifiers from these hubs (link).

\[ a_1(a) = s \land \ldots \land a_2(a) = s \land \ldots \land a_i(a) = s \]
\[ a_1(a) \land a_2(a) \land \ldots \land a_i(a) = s \]

\[ \forall a:A_1, a_2:A_2, \ldots, a_i:A_i \cdot \]
\[ s_a(mk_id_1(a_1, a_2, \ldots, a_i)) = a_1 \land s_a(mk_id_1(a_1, a_2, \ldots, a_i)) = a_2 \land \ldots \land s_a(mk_id_1(a_1, a_2, \ldots, a_i)) = a_i \land \]
\[ \forall a:A \cdot \text{let} \ mk_id_1(a_1, a_2, \ldots, a_i) = a \ \text{in} \]
\[ a' = s_a(a) \land a_2 = s_a(a_2) \land \ldots \land a_i = s_a(a_i) \ \text{end} \]

\[ \text{End of Example 3} \]

8. From a net one can remove a link.\(^5\) The removal command specifies a link identifier.

\[ \text{Example 4} \]

**Net Record Types: Insert Links:**

7. To a net one can insert a new link in either of three ways:

(a) Either the link is connected to two existing hubs — and the insert operation must therefore specify the new link and the identifiers of two existing hubs;

(b) or the link is connected to one existing hub and to a new hub — and the insert operation must therefore specify the new link, the identifier of an existing hub, and a new hub;

(c) or the link is connected to two new hubs — and the insert operation must therefore specify the new link and two new hubs.

(d) From the inserted link one must be able to observe identifier of respective hubs.

8. From a net one can remove a link.\(^5\) The removal command specifies a link identifier.

\[ = \text{provided that what remains is still a proper net} \]
1.1.2. Subtypes

- In RSL, each type represents a set of values. Such a set can be delimited by means of predicates.
- The set of values \( b \) which have type \( B \) and which satisfy the predicate \( \mathcal{P} \), constitute the subtype \( A \):

\[
A = \{ | b: B \cdot \mathcal{P}(b) | \}
\]

Example 17 on page 356 presents the semantics functions for \textit{int\_Insert} and \textit{int\_Remove}.

... End of Example 4 

Example 5 ............................. Net Subtypes:

- In Example 3 on page 273 we gave three examples.
  - For the first we gave an example, \textit{Sorts + Observers + Axioms}, “purely” in terms of sets, see \textit{Sorts — Abstract Types} below.
  - For the second and third we gave concrete types in terms of Cartesians and Maps.

We now redefine this as follows:
In the **Cartesians + Maps + Wellformedness** part of Example 3
- a net was a triple of hubs, links and a graph,
- each with their wellformedness predicates.

We now redefine this as follows:

```plaintext
type
[sorts] L, H, LI, HI
N = \{n:N \cdot wf_N(n)\}
value
wf_N: N → Bool
wf_N(n) ≡ ∀ n:N \cdot card\ obs\_Hs(n) ≥ 0 ∧ card\ obs\_Ls(n) ≥ 0 ∧
axioms 2.–3., 5.–6., and 8., (Page 14)
```

```
• In the **Cartesians + Wellformedness** part of Example 3
  - a net was a Cartesian of a set of hubs and a set of links
  - with the wellformedness that there were at least two hubs and at least one link
  - and that these were connected appropriately (treated as ...).

We now redefine this as follows:

type
N = H-set × L-set
N = \{n:N \cdot wf_N(n)\}
```

```
• In the **Cartesians + Maps + Wellformedness** part of Example 3
  - a net was a triple of hubs, links and a graph,
  - each with their wellformedness predicates.

We now redefine this as follows:

```
```
1.1.2.3. **Sorts — Abstract Types**

- Types can be (abstract) sorts
- in which case their structure is not specified:

  ```
  type
  A, B, ..., C
  ```

**Example 6**  
- Net Sorts:
  - In formula lines of Examples 3–5
  - we have indicated those type clauses which define sorts,
  - by bracketed [sorts] literals.

**End of Example 6**

---

**End of Lecture 7: RSL: Types**