Start of Lecture 7: RSL: VALUES & OPERATIONS

A.2. Concrete RSL Types: Values and Operations

A.2.1. Arithmetic

type
  Nat, **Int**, **Real**
value
  $+,-,*$: **Nat**$\times$**Nat**$\to$**Nat** | **Int**$\times$**Int**$\to$**Int** | **Real**$\times$**Real**$\to$**Real**
  $/$: **Nat**$\times$**Nat**$\xrightarrow{\sim}$**Nat** | **Int**$\times$**Int**$\xrightarrow{\sim}$**Int** | **Real**$\times$**Real**$\xrightarrow{\sim}$**Real**
  $<,\leq,=,\neq,\geq,>$ (**Nat**|**Int**|**Real**) $\times$ (**Nat**|**Int**|**Real**) $\to$ **Bool**

A.2.2. Set Expressions

A.2.2.1. Set Enumerations

Let the below a's denote values of type A, then the below designate simple set enumerations:

$\{\{\}, \{a\}, \{e_1,e_2,...,e_n\}, ...\} \subseteq$ A-**set**
$\{\{\}, \{a\}, \{e_1,e_2,...,e_n\}, ..., \{e_1,e_2,...\}\} \subseteq$ A-**infset**

Example 34 – Set Expressions over Nets:

- We now consider hubs to abstract cities, towns, villages, etcetera.
- Thus with hubs we can associate sets of citizens.
- Let c:C stand for a citizen value c being an element in the type C of all such.
- Let g:G stand for any (group) of citizens, respectively the type of all such.
- Let s:S stand for any set of groups, respectively the type of all such.
- Two otherwise distinct groups are related to one another if they share at least one citizen, the liaisons.
- A network nw:NW is a set of groups such that for every group in the network one can always find another group with which it shares liaisons.

Solely using the set data type and the concept of subtypes, we can model the above:

type
  C
  $G' = C\textbf{-set},\ \ G = \{|\ g{:}G' \cdot g \neq \{\}\ |\}$
  $S = G\textbf{-set}$
  $L' = C\textbf{-set},\ \ L = \{|\ \ell{:}L' \cdot \ell \neq \{\}\ |\}$
  $NW' = S,\ \ NW = \{|\ s{:}S \cdot wf\_S(s)\ |\}$
value
  $wf\_S{:}\ S \to \textbf{Bool}$
  $wf\_S(s) \equiv \forall\ g{:}G \cdot g \in s \Rightarrow \exists\ g'{:}G \cdot\ g' \in s \wedge share(g,g')$
  $share{:}\ G \times G \to \textbf{Bool}$
  $share(g,g') \equiv g \neq g' \wedge g \cap g' \neq \{\}$
  $liaisons{:}\ G \times G \to L$
  $liaisons(g,g') = g \cap g'\ \textbf{pre}\ share(g,g')$

Annotations:

- L stands for proper liaisons (of at least one liaison).
- G', L' and N' are the "raw" types which are constrained to G, L and N.
- {| binding:type_expr · bool_expr |} is the general form of the subtype expression.
- For G and L we state the constraints "in-line", i.e., as direct part of the subtype expression.
- For NW we state the constraints by referring to a separately defined predicate.
- wf_S(s) expresses — through the auxiliary predicate — that s contains at least two groups and that any such two groups share at least one citizen.
- liaisons is a "truly" auxiliary function in that we have yet to "find an active need" for this function!

- The idea is that citizens can be associated with more than one city, town, village, etc.
- (primary home, summer and/or winter house, working place, etc.).
- A group is now a set of citizens related by some "interest"
- (Rotary club membership, political party "grassroots", religion, et.).
- The student is invited to define, for example, such functions as:
  - The set of groups (or networks) which are represented in all hubs [or in only one hub].
  - The set of hubs whose citizens partake in no groups [respectively networks].
  - The group [network] with the largest coverage in terms of number of hubs in which that group [network] is represented.

■ End of Example 34

A.2.2.2. Set Comprehension

- The expression, last line below, to the right of the $\equiv$, expresses set comprehension.
- The expression "builds" the set of values satisfying the given predicate.
- It is abstract in the sense that it does not do so by following a concrete algorithm.

type
  A, B
  $P = A \rightarrow \mathbf{Bool}$
  $Q = A \xrightarrow{\sim} B$
value
  comprehend: A-**infset** × P × Q → B-**infset**
  $\text{comprehend(s,P,Q)} \equiv \{\ Q(a) \mid a{:}A \cdot a \in s \wedge P(a)\}$

Example 35 – Set Comprehensions:

- Example 30 on page 171 illustrates, in the **Cartesians + Maps + Wellformedness** part the following set comprehensions in the *wf_N(hs,ls,g)* wellformedness predicate definition:
  - [d] ∪ {**dom** *g(hi)*|*hi:HI* · *hi* ∈ **dom** *g*}
    - It expresses the distributed union
    - of sets (**dom** *g(hi)*) of link identfiers
    - (for each of the *hi* indexed maps from (definition set, **dom**) link identiers
    - to (range set, **rng**) hub identifiers, where *hi:HI* ranges over **dom** *g*).

  - [e] ∪ {**rng** *g(hi)*|*hi:HI* · *hi* ∈ **dom** *g*}
    - It expresses the distributed union
    - of sets (**rng** *g(hi)*) of hub identfiers
    - (for each of the *hi* indexed maps from (definition set, **dom**) link identiers
    - to (range set, **rng**) hub identifiers, where *hi:HI* ranges over **deom** *g*).

■ End of Example 35

A.2.3. Cartesian Expressions
A.2.3.1. Cartesian Enumerations

- Let e range over values of Cartesian types involving $A, B, \ldots, C$,
- then the below expressions are simple Cartesian enumerations:

type
  A, B, ..., C
  A × B × ... × C
value
  (e1,e2,...,en)

Example 36 – Cartesian Net Types:

- So far we have abstracted hubs and links as sorts.
- That is, we have not defined their types concretely.
- Instead we have postulated some attributes such as:
  - observable hub identifiers of hubs and
  - sets of observable link identifiers of links connected to hubs.
- We now claim the following further attributes of hubs and links.

- Concrete links have
  - link identifiers,
  - link names – where two or more connected links may have the same link name,
  - two (unordered) hub identifiers,
  - lenghts,
  - locations – where we do not presently defined what we main by locations,
  - etcetera
- Concrete hubs have
  - hub identifiers,
  - unique hub names,
  - a set of one or more observable link identifiers,
  - locations,
  - etcetera.

type
LN, HN, LEN, LOC
cL = LI × LN × (HI × HI) × LOC × ...
cH = HI × HN × LI-**set** × LOC × ...

■ End of Example 36

A.2.4. List Expressions

A.2.4.1. List Enumerations

- Let a range over values of type A,
- then the below expressions are simple list enumerations:

$$\{\langle\rangle, \langle e\rangle, ..., \langle e1,e2,...,en\rangle, ...\} \subseteq A^*$$
$$\{\langle\rangle, \langle e\rangle, ..., \langle e1,e2,...,en\rangle, ..., \langle e1,e2,...,en,... \rangle, ...\} \subseteq A^\omega$$

$$\langle\ a\_i\ ..\ a\_j\ \rangle$$

- The last line above assumes a_i and a_j to be integer-valued expressions.
- It then expresses the set of integers from the value of e_i to and including the value of e_j.
- If the latter is smaller than the former, then the list is empty.

A.2.4.2. List Comprehension

- The last line below expresses list comprehension.

type
 A, B, P = A → **Bool**, Q = A $\stackrel{\sim}{\to}$ B
value
 comprehend: $A^\omega \times P \times Q \stackrel{\sim}{\to} B^\omega$
 comprehend(l,P,Q) ≡
  ⟨ Q(l(i)) | i **in** ⟨1..**len** l⟩ · P(l(i))⟩

Example 37 – Routes in Nets:

- A phenomenological (i.e., a physical) route of a net is a sequence of one or more adjacent links of that net.
- A conceptual route is a sequence of one or more link identifiers.
- An abstract route is a conceptual route
  - for which there is a phenomenological route of the net
  - for which the link identifiers of the abstract route
  - map one-to-one onto links of the phenomenological route.

type
 N, H, L, HI, LI
 PR′ = L*
 PR = {| pr:PR′ · ∃ n:N · wf_PR(pr)(n)|}
 CR = LI*
 AR′ = LI*
 AR = {| ar:AR′ · ∃ n:N · wf_AR(ar)(n) |}
value
 wf_PR: PR′ → N → **Bool**
 wf_PR(pr)(n) ≡
  ∀ i:**Nat** · {i,i+1}⊆**inds** pr ⇒
   ωHIs(l(i)) ∩ ωHIs(l(i+1)) ≠ {}
 wf_AR′: AR′ → N → **Bool**
 wf_AR(ar)(n) ≡
  ∃ pr:PR · pr ∈ routes(n) ∧ wf_PR(pr)(n) ∧ **len** pr=**len** ar ∧
   ∀ i:**Nat** · i ∈ **inds** ar ⇒ ωLI(pr(i))=ar(i)

- A single link is a phenomenological route.
- If r and r' are phenomenological routes
  - such that the last link r
  - and the first link of r'
  - share observable hub identifiers,

  then the concatenation $r^\frown r'$ is a route.

  This inductive definition implies a recursive set comprehension.
- A circular phenomenological route is a phenomenological route whose first and last links are distinct but share hub identifiers.
- A looped phenomenological route is a phenomenological route where two distinctly positions (i.e., indexed) links share hub identifiers.

value

$\mathsf{routes}\colon \mathsf{N} \to \mathsf{PR}\textbf{-infset}$

$\mathsf{routes(n)} \equiv$

$\quad \textbf{let}\ \mathsf{prs} = \{\langle \mathsf{l}\rangle | \mathsf{l{:}L{\cdot}\omega Ls(n)}\} \cup$

$\qquad\qquad \cup\ \{\mathsf{pr}\widehat{\ }\mathsf{pr'}|\mathsf{pr,pr'{:}PR{\cdot}}\{\mathsf{pr,pr'}\}\subseteq \mathsf{prs} \wedge \omega \mathsf{Hls(r(}\textbf{len}\ \mathsf{pr))} \cap \omega\mathsf{Hls(pr'(1))} \neq \{\}\}$

$\quad \mathsf{prs}\ \textbf{end}$

$\mathsf{is\_circular}\colon \mathsf{PR} \to \textbf{Bool}$

$\mathsf{is\_circular(pr)} \equiv \omega\mathsf{Hls(pr(1))} \cap \omega\mathsf{Hls(pr(}\textbf{len}\ \mathsf{pr))} \neq \{\}$

$\mathsf{is\_looped}\colon \mathsf{PR} \to \textbf{Bool}$

$\mathsf{is\_looped(pr)} \equiv \exists\ \mathsf{i,j}{:}\textbf{Nat} \cdot \mathsf{i} \neq \mathsf{j} \wedge \{\mathsf{i,j}\} \subseteq \mathsf{index\ pr} \Rightarrow \omega\mathsf{Hls(pr(i))} \cap \omega\mathsf{Hls(pr(j))} \neq \{\}$

- Straight routes are Phenomenological routes without loops.
- Phenomenological routes with no loops can be constructed from phenomenological routes by removing suffix routes whose first link give rise to looping.

value

$\mathsf{straight\_routes}\colon \mathsf{N} \to \mathsf{PR}\textbf{-set}$

$\mathsf{straight\_routes(n)} \equiv$

$\quad \textbf{let}\ \mathsf{prs} = \mathsf{routes(n)}\ \textbf{in}\ \{\mathsf{straight\_route(pr)}|\mathsf{pr{:}PR{\cdot}ps} \in \mathsf{prs}\}\ \textbf{end}$

$\mathsf{straight\_route}\colon \mathsf{PR} \to \mathsf{PR}$

$\mathsf{straight\_route(pr)} \equiv$

$\quad \langle \mathsf{pr(i)}|\mathsf{i}{:}\textbf{Nat}{\cdot}\mathsf{i}{:}[\,1..\textbf{len}\ \mathsf{pr}\,] \wedge \mathsf{pr(i)} \notin \textbf{elems}\langle \mathsf{pr(j)}|\mathsf{j}{:}\textbf{Nat}{\cdot}\mathsf{j}{:}[\,1..\mathsf{i}\,]\rangle\rangle$

■ End of Example 37

A.2.5. Map Expressions

A.2.5.1. Map Enumerations

- Let (possibly indexed) u and v range over values of type $T1$ and $T2$, respectively,
- then the below expressions are simple map enumerations:

type

$\mathrm{T1, T2}$

$\mathrm{M = T1 \ \overrightarrow{m}\ T2}$

value

$\mathrm{u,u1,u2,...,un{:}T1,\ v,v1,v2,...,vn{:}T2}$

$\{[\,],\ [\,\mathrm{u}\mapsto\mathrm{v}\,],\ ...,\ [\,\mathrm{u1}\mapsto\mathrm{v1,u2}\mapsto\mathrm{v2,...,un}\mapsto\mathrm{vn}\,],...\} \subseteq \mathrm{M}$

A.2.5.2. Map Comprehension

- The last line below expresses map comprehension:

type

$\mathrm{U, V, X, Y}$

$\mathrm{M = U \ \overrightarrow{m}\ V}$

$\mathrm{F = U \xrightarrow{\sim} X}$

$\mathrm{G = V \xrightarrow{\sim} Y}$

$\mathrm{P = U \to \textbf{Bool}}$

value

$\mathrm{comprehend\colon M{\times}F{\times}G{\times}P \to (X \ \overrightarrow{m}\ Y)}$

$\mathrm{comprehend(m,F,G,P)} \equiv$

$\quad [\ \mathrm{F(u)} \mapsto \mathrm{G(m(u))}\ |\ \mathrm{u{:}U} \cdot \mathrm{u} \in \textbf{dom}\ \mathrm{m} \wedge \mathrm{P(u)}\,]$

Example 38 – Concrete Net Type Construction:

- We Define a function *con[struct]_N_γ* (of the **Cartesians + Maps + Wellformedness** part of Example 30.
  - The base of the construction is the fully abstract sort definition of N_α in the **Sorts + Observers + Axioms** part of Example 30 – where the sorts of hub and link identifiers are taken from earlier examples.
  - The target of the construction is the N_γ of the **Cartesians + Maps + Wellformedness** part of Example 30.
  - First we recall the ssential types of that N_γ.

type
$N_\gamma = \mathsf{HUBS} \times \mathsf{LINKS} \times \mathsf{GRAPH}$
$\mathsf{HUBS} = \mathsf{HI} \ \overrightarrow{_m} \ \mathsf{H}$
$\mathsf{LINKS} = \mathsf{LI} \ \overrightarrow{_m} \ \mathsf{L}$
$\mathsf{GRAPH} = \mathsf{HI} \ \overrightarrow{_m} \ (\mathsf{LI} \ \overrightarrow{_m} \ \mathsf{HI})$
value
$\mathsf{con\_N}_\gamma\colon \mathsf{N}_\alpha \to \mathsf{N}_\gamma$
$\mathsf{con\_N}_\gamma(\mathsf{n}_\alpha) \equiv$
  let hubs $= [\,\omega\mathsf{HI}(h) \mapsto h \mid h{:}H \cdot h \in \omega\mathsf{Hs}(n_\alpha)\,]$,
    links $= [\,\omega\mathsf{LI}(h) \mapsto l \mid l{:}L \cdot l \in \omega\mathsf{Ls}(n_\alpha)\,]$,
    graph $= [\,\omega\mathsf{HI}(h) \mapsto [\,\omega\mathsf{LI}(l) \mapsto \iota(\omega\mathsf{HIs}(l)\backslash\{\omega\mathsf{HI}(h)\})$
      $\mid l{:}L \cdot l \in \omega\mathsf{Ls}(n_\alpha) \wedge li \in \omega\mathsf{LIs}(h)\,]$
      $\mid H{:}h \cdot h \in \omega\mathsf{Hs}(n_\alpha)\,]$ **in**
    (hubs.links,graph) **end**

ι: A-**set** $\xrightarrow{\sim}$ A [A could be LI-**set**]
ι(as) $\equiv$ **if card** as=1 **then let** {a}=as **in** a **else chaos end end**

theorem:
n_α **satisfies axioms** [2,5–8] for N of Example 1 $\Rightarrow$ $\mathsf{wf\_N}_\gamma\mathsf{con\_N}_\gamma(n_\alpha)$

■ End of Example 38

A.2.6. Set Operations
A.2.6.1. Set Operator Signatures

value
9 $\in$: A $\times$ A-**infset** $\to$ **Bool**
10 $\notin$: A $\times$ A-**infset** $\to$ **Bool**
11 $\cup$: A-**infset** $\times$ A-**infset** $\to$ A-**infset**
12 $\cup$: (A-**infset**)-**infset** $\to$ A-**infset**
13 $\cap$: A-**infset** $\times$ A-**infset** $\to$ A-**infset**
14 $\cap$: (A-**infset**)-**infset** $\to$ A-**infset**
15 $\backslash$: A-**infset** $\times$ A-**infset** $\to$ A-**infset**
16 $\subset$: A-**infset** $\times$ A-**infset** $\to$ **Bool**
17 $\subseteq$: A-**infset** $\times$ A-**infset** $\to$ **Bool**
18 =: A-**infset** $\times$ A-**infset** $\to$ **Bool**
19 $\neq$: A-**infset** $\times$ A-**infset** $\to$ **Bool**
20 **card**: A-**infset** $\xrightarrow{\sim}$ **Nat**

A.2.6.2. Set Examples

examples

$a \in \{a,b,c\}$
$a \notin \{\}$, $a \notin \{b,c\}$
$\{a,b,c\} \cup \{a,b,d,e\} = \{a,b,c,d,e\}$
$\cup\{\{a\},\{a,b\},\{a,d\}\} = \{a,b,d\}$
$\{a,b,c\} \cap \{c,d,e\} = \{c\}$
$\cap\{\{a\},\{a,b\},\{a,d\}\} = \{a\}$
$\{a,b,c\} \setminus \{c,d\} = \{a,b\}$
$\{a,b\} \subset \{a,b,c\}$
$\{a,b,c\} \subseteq \{a,b,c\}$
$\{a,b,c\} = \{a,b,c\}$
$\{a,b,c\} \neq \{a,b\}$
card $\{\} = 0$, **card** $\{a,b,c\} = 3$

A.2.6.3. Informal Explication

9. $\in$: The membership operator expresses that an element is a member of a set.
10. $\notin$: The nonmembership operator expresses that an element is not a member of a set.
11. $\cup$: The infix union operator. When applied to two sets, the operator gives the set whose members are in either or both of the two operand sets.
12. $\cup$: The distributed prefix union operator. When applied to a set of sets, the operator gives the set whose members are in some of the operand sets.
13. $\cap$: The infix intersection operator. When applied to two sets, the operator gives the set whose members are in both of the two operand sets.
14. $\cap$: The prefix distributed intersection operator. When applied to a set of sets, the operator gives the set whose members are in some of the operand sets.

15. $\setminus$: The set complement (or set subtraction) operator. When applied to two sets, the operator gives the set whose members are those of the left operand set which are not in the right operand set.
16. $\subseteq$: The proper subset operator expresses that all members of the left operand set are also in the right operand set.
17. $\subset$: The proper subset operator expresses that all members of the left operand set are also in the right operand set, and that the two sets are not identical.
18. $=$: The equal operator expresses that the two operand sets are identical.
19. $\neq$: The nonequal operator expresses that the two operand sets are *not* identical.
20. **card**: The cardinality operator gives the number of elements in a finite set.

A.2.6.4. Set Operator Definitions

value

$s' \cup s'' \equiv \{ a \mid a{:}A \cdot a \in s' \vee a \in s'' \}$
$s' \cap s'' \equiv \{ a \mid a{:}A \cdot a \in s' \wedge a \in s'' \}$
$s' \setminus s'' \equiv \{ a \mid a{:}A \cdot a \in s' \wedge a \notin s'' \}$
$s' \subseteq s'' \equiv \forall\, a{:}A \cdot a \in s' \Rightarrow a \in s''$
$s' \subset s'' \equiv s' \subseteq s'' \wedge \exists\, a{:}A \cdot a \in s'' \wedge a \notin s'$
$s' = s'' \equiv \forall\, a{:}A \cdot a \in s' \equiv a \in s'' \equiv s{\subseteq}s' \wedge s'{\subseteq}s$
$s' \neq s'' \equiv s' \cap s'' \neq \{\}$
card $s \equiv$
  if $s = \{\}$ **then** 0 **else**
  let $a{:}A \cdot a \in s$ **in** $1 +$ **card** $(s \setminus \{a\})$ **end end**
  pre s /* is a finite set */
card $s \equiv$ **chaos** /* tests for infinity of s */

A.2.7. Cartesian Operations

type
A, B, C
g0: G0 = A × B × C
g1: G1 = (A × B × C)
g2: G2 = (A × B) × C
g3: G3 = A × (B × C)

value
va:A, vb:B, vc:C, vd:D
(va,vb,vc):G0,
(va,vb,vc):G1
((va,vb),vc):G2
(va3,(vb3,vc3)):G3

decomposition expressions
let (a1,b1,c1) = g0,
(a1′,b1′,c1′) = g1 **in** .. **end**
let ((a2,b2),c2) = g2 **in** .. **end**
let (a3,(b3,c3)) = g3 **in** .. **end**

A.2.8. List Operations

A.2.8.1. List Operator Signatures

value
hd: $A^\omega \xrightarrow{\sim} A$
tl: $A^\omega \xrightarrow{\sim} A^\omega$
len: $A^\omega \xrightarrow{\sim}$ **Nat**
inds: $A^\omega \rightarrow$ **Nat-infset**
elems: $A^\omega \rightarrow$ A-**infset**
.(.): $A^\omega \times$ **Nat** $\xrightarrow{\sim} A$
$\widehat{\ }$: $A^* \times A^\omega \rightarrow A^\omega$
=: $A^\omega \times A^\omega \rightarrow$ **Bool**
≠: $A^\omega \times A^\omega \rightarrow$ **Bool**

A.2.8.2. List Operation Examples

examples
hd⟨a1,a2,...,am⟩=a1
tl⟨a1,a2,...,am⟩=⟨a2,...,am⟩
len⟨a1,a2,...,am⟩=m
inds⟨a1,a2,...,am⟩={1,2,...,m}
elems⟨a1,a2,...,am⟩={a1,a2,...,am}
⟨a1,a2,...,am⟩(i)=ai
⟨a,b,c⟩$\widehat{\ }$⟨a,b,d⟩ = ⟨a,b,c,a,b,d⟩
⟨a,b,c⟩=⟨a,b,c⟩
⟨a,b,c⟩ ≠ ⟨a,b,d⟩

A.2.8.3. Informal Explication

- **hd**: Head gives the first element in a nonempty list.
- **tl**: Tail gives the remaining list of a nonempty list when Head is removed.
- **len**: Length gives the number of elements in a finite list.
- **inds**: Indices give the set of indices from 1 to the length of a nonempty list. For empty lists, this set is the empty set as well.
- **elems**: Elements gives the possibly infinite set of all distinct elements in a list.
- $\ell(i)$: Indexing with a natural number, i larger than 0, into a list ℓ having a number of elements larger than or equal to i, gives the ith element of the list.

- $\widehat{\ }$: Concatenates two operand lists into one. The elements of the left operand list are followed by the elements of the right. The order with respect to each list is maintained.
- $=$: The equal operator expresses that the two operand lists are identical.
- $\neq$: The nonequal operator expresses that the two operand lists are *not* identical.

The operations can also be defined as follows:

A.2.8.4. List Operator Definitions

$$
\begin{array}{l}
\mathbf{value} \\
\quad \mathrm{is\_finite\_list}: A^{\omega} \to \mathbf{Bool} \\
\\
\quad \mathbf{len}\ q \equiv \\
\qquad \mathbf{case}\ \mathrm{is\_finite\_list}(q)\ \mathbf{of} \\
\qquad\quad \mathbf{true} \to \mathbf{if}\ q = \langle\rangle\ \mathbf{then}\ 0\ \mathbf{else}\ 1 + \mathbf{len}\ \mathbf{tl}\ q\ \mathbf{end}, \\
\qquad\quad \mathbf{false} \to \mathbf{chaos}\ \mathbf{end} \\
\\
\quad \mathbf{inds}\ q \equiv \\
\qquad \mathbf{case}\ \mathrm{is\_finite\_list}(q)\ \mathbf{of} \\
\qquad\quad \mathbf{true} \to \{\ i \mid i{:}\mathbf{Nat} \cdot 1 \leq i \leq \mathbf{len}\ q\ \}, \\
\qquad\quad \mathbf{false} \to \{\ i \mid i{:}\mathbf{Nat} \cdot i{\neq}0\ \}\ \mathbf{end} \\
\\
\quad \mathbf{elems}\ q \equiv \{\ q(i) \mid i{:}\mathbf{Nat} \cdot i \in \mathbf{inds}\ q\ \}
\end{array}
$$

$$
\begin{array}{l}
q(i) \equiv \\
\quad \mathbf{case}\ (q,i)\ \mathbf{of} \\
\qquad (\langle\rangle,1) \to \mathbf{chaos}, \\
\qquad (\_,1) \to \mathbf{let}\ a{:}A,q'{:}Q \cdot q{=}\langle a\rangle\widehat{\ }q'\ \mathbf{in}\ a\ \mathbf{end} \\
\qquad \_ \to q(i{-}1) \\
\quad \mathbf{end} \\
\\
fq\ \widehat{\ }\ iq \equiv \\
\qquad \langle\ \mathbf{if}\ 1 \leq i \leq \mathbf{len}\ fq\ \mathbf{then}\ fq(i)\ \mathbf{else}\ iq(i - \mathbf{len}\ fq)\ \mathbf{end} \\
\qquad\ \mid i{:}\mathbf{Nat} \cdot \mathbf{if}\ \mathbf{len}\ iq{\neq}\mathbf{chaos}\ \mathbf{then}\ i \leq \mathbf{len}\ fq{+}\mathbf{len}\ \mathbf{end}\ \rangle \\
\quad \mathbf{pre}\ \mathrm{is\_finite\_list}(fq) \\
\\
iq' = iq'' \equiv \\
\quad \mathbf{inds}\ iq' = \mathbf{inds}\ iq'' \wedge \forall\ i{:}\mathbf{Nat} \cdot i \in \mathbf{inds}\ iq' \Rightarrow iq'(i) = iq''(i) \\
\\
iq' \neq iq'' \equiv \sim(iq' = iq'')
\end{array}
$$

A.2.9. Map Operations

A.2.9.1. Map Operator Signatures and Map Operation Examples

$$
\begin{array}{l}
\mathbf{value} \\
\quad m(a): M \to A \xrightarrow{\sim} B,\ m(a) = b \\
\\
\quad \mathbf{dom}: M \to A\text{-}\mathbf{infset}\ [\text{domain of map}] \\
\qquad \mathbf{dom}\ [a1{\mapsto}b1,a2{\mapsto}b2,...,an{\mapsto}bn] = \{a1,a2,...,an\} \\
\\
\quad \mathbf{rng}: M \to B\text{-}\mathbf{infset}\ [\text{range of map}] \\
\qquad \mathbf{rng}\ [a1{\mapsto}b1,a2{\mapsto}b2,...,an{\mapsto}bn] = \{b1,b2,...,bn\} \\
\\
\quad \dagger: M \times M \to M\ [\text{override extension}] \\
\qquad [a{\mapsto}b,a'{\mapsto}b',a''{\mapsto}b''] \dagger [a'{\mapsto}b'',a''{\mapsto}b'] = [a{\mapsto}b,a'{\mapsto}b'',a''{\mapsto}b']
\end{array}
$$

$\cup$: M $\times$ M $\rightarrow$ M [merge $\cup$]
$[a\mapsto b, a'\mapsto b', a''\mapsto b''] \cup [a'''\mapsto b'''] = [a\mapsto b, a'\mapsto b', a''\mapsto b'', a'''\mapsto b''']$

$\backslash$: M $\times$ A-**infset** $\rightarrow$ M [restriction by]
$[a\mapsto b, a'\mapsto b', a''\mapsto b'']\backslash\{a\} = [a'\mapsto b', a''\mapsto b'']$

/: M $\times$ A-**infset** $\rightarrow$ M [restriction to]
$[a\mapsto b, a'\mapsto b', a''\mapsto b'']/\{a', a''\} = [a'\mapsto b', a''\mapsto b'']$

$=,\neq$: M $\times$ M $\rightarrow$ **Bool**

$\circ$: (A $\overrightarrow{m}$ B) $\times$ (B $\overrightarrow{m}$ C) $\rightarrow$ (A $\overrightarrow{m}$ C) [composition]
$[a\mapsto b, a'\mapsto b'] \circ [b\mapsto c, b'\mapsto c', b''\mapsto c''] = [a\mapsto c, a'\mapsto c']$

A.2.9.2. Map Operation Explication

- $m(a)$: Application gives the element that a maps to in the map m.
- **dom**: Domain/Definition Set gives the set of values which *maps to* in a map.
- **rng**: Range/Image Set gives the set of values which *are mapped to* in a map.
- †: Override/Extend. When applied to two operand maps, it gives the map which is like an override of the left operand map by all or some "pairings" of the right operand map.
- $\cup$: Merge. When applied to two operand maps, it gives a merge of these maps.

- $\backslash$: Restriction. When applied to two operand maps, it gives the map which is a restriction of the left operand map to the elements that are not in the right operand set.
- /: Restriction. When applied to two operand maps, it gives the map which is a restriction of the left operand map to the elements of the right operand set.
- =: The equal operator expresses that the two operand maps are identical.
- $\neq$: The nonequal operator expresses that the two operand maps are *not* identical.
- $\circ$: Composition. When applied to two operand maps, it gives the map from definition set elements of the left operand map, m_1, to the range elements of the right operand map, m_2, such that if a is in the definition set of m_1 and maps into b, and if b is in the definition set of m_2 and maps into c, then a, in the composition, maps into c.

Example 39 – Miscellaneous Net Expressions: Maps: Example 30 on page 171 left out defining the well-formedness of the map types:

value
  wf_HUBS: HUBS → **Bool**
 [a] wf_HUBS(hubs) ≡ ∀ hi:HI · hi ∈ **dom** hubs ⇒ ωHIhubs(hi)=hi
  wf_LINKS: LINKS → **Bool**
 [b] wf_LINKS(links) ≡ ∀ li:LI · li ∈ **dom** links ⇒ ωLIlinks(li)=li
 wf_N$_\gamma$: N$_\gamma$ → **Bool**
 wf_N$_\gamma$(hs,ls,g) ≡
   [c] **dom** hs = **dom** g ∧
   [d] ∪ {**dom** g(hi)|hi:HI · hi ∈ **dom** g} = **dom** links ∧
   [e] ∪ {**rng** g(hi)|hi:HI · hi ∈ **dom** g} = **dom** g ∧
   [f] ∀ hi:HI · hi ∈ **dom** g ⇒ ∀ li:LI · li ∈ **dom** g(hi) ⇒ (g(hi))(li)≠hi
   [g] ∀ hi:HI · hi ∈ **dom** g ⇒ ∀ li:LI · li ∈ **dom** g(hi) ⇒
      ∃ hi′:HI · hi′ ∈ **dom** g ⇒ ∃ ! li:LI · li ∈ **dom** g(hi) ⇒
       (g(hi))(li) = hi′ ∧ (g(hi′))(li) = hi

- [c] *HUBS* record the same hubs as do the net corresponding *GRAPHS* ($\mathbf{dom}\ hs = \mathbf{dom}\ g \wedge$).
- [d] *GRAPHS* record the same links as do the net corresponding *LINKS* ($\cup \{\mathbf{dom}\ g(hi) | hi{:}HI \cdot hi \in \mathbf{dom}\ g\} = \mathbf{dom}\ links$).
- [e] The target (or range) hub identifiers of graphs are the same as the domain of the graph ($\cup \{\mathbf{rng}\ g(hi) | hi{:}HI \cdot hi \in \mathbf{dom}\ g\} = \mathbf{dom}\ g$), that is none missing, no new ones !
- [f] No links emanate from and are incident upon the same hub ($\forall\ hi{:}HI \cdot hi \in \mathbf{dom}\ g \Rightarrow \forall\ li{:}LI \cdot li \in \mathbf{dom}\ g(hi) \Rightarrow (g(hi))(li) \neq hi$).
- [g] If there is a link from one hub to another in the *GRAPH*, then the same link also connects the other hub to the former ($\forall\ hi{:}HI \cdot hi \in \mathbf{dom}\ g \Rightarrow \forall\ li{:}LI \cdot li \in \mathbf{dom}\ g(hi) \Rightarrow \exists\ h\grave{i}{:}HI \cdot h\grave{i} \in \mathbf{dom}\ g \Rightarrow \exists\ !\ li{:}LI \cdot li \in \mathbf{dom}\ g(hi) \Rightarrow (g(hi))(li) = h\grave{i} \wedge (g(h\grave{i}))(li) = hi$).

■ End of Example 39

A.2.9.3. Map Operation Redefinitions

value

$$\mathbf{rng}\ m \equiv \{\ m(a)\ |\ a{:}A \cdot a \in \mathbf{dom}\ m\ \}$$

$$m1 \dagger m2 \equiv [\ a \mapsto b\ |\ a{:}A,b{:}B \cdot a \in \mathbf{dom}\ m1 \setminus \mathbf{dom}\ m2 \wedge b{=}m1(a) \vee a \in \mathbf{dom}\ m2 \wedge b{=}m2(a)\ ]$$

$$m1 \cup m2 \equiv [\ a \mapsto b\ |\ a{:}A,b{:}B \cdot a \in \mathbf{dom}\ m1 \wedge b{=}m1(a) \vee a \in \mathbf{dom}\ m2 \wedge b{=}m2(a)\ ]$$

$$m \setminus s \equiv [\ a \mapsto m(a)\ |\ a{:}A \cdot a \in \mathbf{dom}\ m \setminus s\ ]$$
$$m\ /\ s \equiv [\ a \mapsto m(a)\ |\ a{:}A \cdot a \in \mathbf{dom}\ m \cap s\ ]$$

$$m1 = m2 \equiv \mathbf{dom}\ m1 = \mathbf{dom}\ m2 \wedge \forall\ a{:}A \cdot a \in \mathbf{dom}\ m1 \Rightarrow m1(a) = m2(a)$$
$$m1 \neq m2 \equiv \sim(m1 = m2)$$

$$m^{\circ}n \equiv [\ a \mapsto c\ |\ a{:}A,c{:}C \cdot a \in \mathbf{dom}\ m \wedge c = n(m(a))\ ]\ \ \mathbf{pre}\ \mathbf{rng}\ m \subseteq \mathbf{dom}\ n$$

End of Lecture 7: RSL: VALUES & OPERATIONS