Start of Lecture 8: RSL: PREDICATE CALCULUS and λ–CALCULUS

A.3. The RSL Predicate Calculus

A.3.1. Propositional Expressions

- Let identifiers (or propositional expressions) a, b, ..., c designate Boolean values (**true** or **false** [or **chaos**]).
- Then:

false, **true**
a, b, ..., c $\sim$a, a$\wedge$b, a$\vee$b, a$\Rightarrow$b, a=b, a$\neq$b

- are propositional expressions having Boolean values.
- $\sim$, $\wedge$, $\vee$, $\Rightarrow$, = and $\neq$ are Boolean connectives (i.e., operators).
- They can be read as: *not*, *and*, *or*, *if then* (or *implies*), *equal* and *not equal*.

A.3.2. Simple Predicate Expressions

- Let identifiers (or propositional expressions) a, b, ..., c designate Boolean values,
- let x, y, ..., z (or term expressions) designate non-Boolean values
- and let i, j, . . ., k designate number values,
- then:

false, **true**
a, b, ..., c
$\sim$a, a$\wedge$b, a$\vee$b, a$\Rightarrow$b, a=b, a$\neq$b
x=y, x$\neq$y,
i<j, i$\leq$j, i$\geq$j, i$\neq$j, i$\geq$j, i>j

- are simple predicate expressions.

A.3.3. Quantified Expressions

- Let X, Y, . . ., C be type names or type expressions,
- and let $\mathcal{P}(x)$, $\mathcal{Q}(y)$ and $\mathcal{R}(z)$ designate predicate expressions in which x, y and z are free.
- Then:

$\forall$ x:X $\cdot$ $\mathcal{P}(x)$
$\exists$ y:Y $\cdot$ $\mathcal{Q}(y)$
$\exists\,!$ z:Z $\cdot$ $\mathcal{R}(z)$

- are quantified expressions — also being predicate expressions.

Example 40 – Predicates Over Net Quantities:

- From earlier examples we show some predicates:
- Example 28: Right hand side of function definition *is_two_way_link(l)*:
  $\exists\ l\sigma{:}L\Sigma \cdot l\sigma \in \omega H\Sigma(l) \wedge \mathbf{card}\ l\sigma{=}2$

- Example 30:
  - The **Sorts + Observers + Axioms** part:
    - Right hand side of the wellformedness function *wf_N(n)* definition:
      $\forall\ n{:}N \cdot \mathbf{card}\,\omega Hs(n){\geq}2 \wedge \mathbf{card}\,\omega Ls(n){\geq}1 \wedge [\,5{-}{-}8\,]$ *of example 1*
    - Right hand side of the wellformedness function *wf_N(hs,ls)* definition:
      $\mathbf{card}\ hs{\geq}2 \wedge \mathbf{card}\ ls{\geq}1\ \ldots$

- The **Cartesians + Maps + Wellformedness** part:
  - Right hand side of the *wf_HUBS* wellformedness function definition:
    $\forall\ hi{:}HI \cdot hi \in \mathbf{dom}\ hubs \Rightarrow \omega HIhubs(hi){=}hi$
  - Right hand side of the *wf_LINKS* wellformedness function definition:
    $\forall\ li{:}LI \cdot li \in \mathbf{dom}\ links \Rightarrow \omega LIlinks(li){=}li$
  - Right hand side of the *wf_N(7 hs,ls,g)* wellformedness function definition:
    $[\,c\,]\ \mathbf{dom}\ hs = \mathbf{dom}\ g \wedge$
    $[\,d\,] \cup \{\mathbf{dom}\ g(hi) | hi{:}HI \cdot hi \in \mathbf{dom}\ g\} = \mathbf{dom}\ links \wedge$
    $[\,e\,] \cup \{\mathbf{rng}\ g(hi) | hi{:}HI \cdot hi \in \mathbf{dom}\ g\} = \mathbf{dom}\ g \wedge$
    $[\,f\,]\ \forall\ hi{:}HI \cdot hi \in \mathbf{dom}\ g \Rightarrow \forall\ \ li{:}LI \cdot li \in \mathbf{dom}\ g(hi) \Rightarrow (g(hi))(li){\neq}hi$
    $[\,g\,]\ \forall\ hi{:}HI \cdot hi \in \mathbf{dom}\ g \Rightarrow \forall\ li{:}LI \cdot li \in \mathbf{dom}\ g(hi) \Rightarrow$
    $\exists\ hi'{:}HI \cdot hi' \in \mathbf{dom}\ g \Rightarrow \exists\,!\ li{:}LI \cdot li \in \mathbf{dom}\ g(hi) \Rightarrow$
    $(g(hi))(li) = hi' \wedge (g(hi'))(li) = hi$

■ End of Example 40

A.4. λ-Calculus + Functions

A.4.1. The λ-Calculus Syntax

type /∗ A BNF Syntax: ∗/
  ⟨L⟩ ::= ⟨V⟩ | ⟨F⟩ | ⟨A⟩ | (⟨A⟩)
  ⟨V⟩ ::= /∗ variables, i.e. identifiers ∗/
  ⟨F⟩ ::= λ⟨V⟩ · ⟨L⟩
  ⟨A⟩ ::= (⟨L⟩⟨L⟩)
value /∗ Examples ∗/
  ⟨L⟩: e, f, a, ...
  ⟨V⟩: x, ...
  ⟨F⟩: λ x · e, ...
  ⟨A⟩: f a, (f a), f(a), (f)(a), ...

A.4.2. Free and Bound Variables

Let x, y be variable names and e, f be λ-expressions.

- $\langle$V$\rangle$: Variable x is free in x.
- $\langle$F$\rangle$: x is free in $\lambda y \cdot e$ if $x \neq y$ and x is free in e.
- $\langle$A$\rangle$: x is free in $f(e)$ if it is free in either f or e (i.e., also in both).

A.4.3. Substitution

- **subst**([N/x]x) $\equiv$ N;
- **subst**([N/x]a) $\equiv$ a,

  for all variables a$\neq$ x;
- **subst**([N/x](P Q)) $\equiv$ (**subst**([N/x]P) **subst**([N/x]Q));
- **subst**([N/x]($\lambda x \cdot P$)) $\equiv$ λ y·P;
- **subst**([N/x](λ y·P)) $\equiv$ $\lambda y \cdot$ **subst**([N/x]P),

  if x$\neq$y and y is not free in N or x is not free in P;
- **subst**([N/x](λy·P)) $\equiv$ λz·**subst**([N/z]**subst**([z/y]P)),

  if y$\neq$x and y is free in N and x is free in P

  (where z is not free in (N P)).

A.4.4. α-Renaming and β-Reduction

- α-renaming: λx·M

  If x, y are distinct variables then replacing x by y in λx·M results in λy·**subst**([y/x]M). We can rename the formal parameter of a λ-function expression provided that no free variables of its body M thereby become bound.
- β-reduction: (λx·M)(N)

  All free occurrences of x in M are replaced by the expression N provided that no free variables of N thereby become bound in the result. (λx·M)(N) $\equiv$ **subst**([N/x]M)

A.4.5. An Example

Example 41 – Network Traffic:

- We model traffic by introducing a number of model concepts.
- We simplify
  - – without loosing the essence of this example, namely to show the use of λ–functions –
  - by omitting consideration of dynamically changing nets.
- These are introduced next:
  - Let us assume a net, *n:N*.
  - There is a dense set, *T*, of times – for which we omit giving an appropriate definition.
  - There is a sort, *V*, of vehicles.
  - *TS* is a dense subset of *T*.
  - For each *ts:TS* we can define a minimum and a maximum time.

- The $\mathcal{MIN}$ and $\mathcal{MAX}$ functions are meta-linguistic.
- At any moment some vehicles, *v:V*, have a *pos:Pos*ition on the net and *VP* records those.
- A *Pos*ition is either on a link or at a hub.
- An *onL*ink position can be designated by the link identifier, the identifiers of the from and to hubs, and the fraction, *f:F*, of the distance down the link from the from hub to the to hub.
- An *atH*ub position just designates the hub (by its identifier).
- Traffic, *tf:TF*, is now a continuous function from *T*ime to *NP* ("recordings").
- Modelling traffic in this way entails a ("serious") number of well-formedness conditions. These are defined in *wf_TF* (omitted: ...).

value
  n:N
type
  T, V
  TS = T-**infset**
axiom
  ∀ ts:TS • ∃ tmin,tmax:T: tmin ∈ ts ∧ tmax ∈ ts ∧ ∀ t:T • t ∈ ts ⇒ tmin ≤ t ≤ tmax
  [that is: ts = {$\mathcal{MIN}$(ts)..$\mathcal{MAX}$(ts)}]
type
  VP = V $\overrightarrow{m}$ Pos
  TF′ = T → VP, TF = {|tf:TF′•wf_TF(tf)(n)|}
  Pos = onL | atH
  onL == mkLPos(hi:HI,li:LI,f:F,hi:HI), atH == mkHPos(hi:HI)
value
  wf_TF: TF→ N → **Bool**
  wf_TF(tf)(n) ≡ ...
  $\mathcal{DOMAIN}$: TF → TS
  $\mathcal{MIN}$, $\mathcal{MAX}$: TS → T

- We have defined the continuous, composite entity of traffic.
- Now let us define an operation of inserting a vehicle in a traffic.
- To insert a vehicle, v, in a traffic, tf, is prescribable as follows:
  - the vehicle, v, must be designated;
  - a time point, t, "inside" the traffic tf must be stated;
  - a traffic, vtf, from time t of vehicle v must be stated;
  - as well as traffic, tf, into which vtf is to be "merged".
- The resulting traffic is referred to as tf'.

value
  insert_V: V × T × TF → TF → TF
  insert_V(v,t,vtf)(tf) **as** tf′

- The function *insert_V* is here defined in terms of a pair of pre/post conditions.
- The pre-condition can be prescribed as follows:
  - The insertion time t must be within to open interval of time points in the traffic tf to which insertion applies.
  - The vehicle v must not be among the vehicle positions of tf.
  - The vehicle must be the only vehicle "contained" in the "inserted" traffic vtf.

pre: $\mathcal{MIN}$($\mathcal{DOMAIN}$(tf)≤t≤$\mathcal{MAX}$($\mathcal{DOMAIN}$(tf)) ∧
  ∀ t′:T • t′ ∈ $\mathcal{DOMAIN}$(tf) ⇒ v ∉ **dom** tf(t′) ∧
  $\mathcal{MIN}$($\mathcal{DOMAIN}$(vtf)) = t ∧
  ∀ t′:T•t′ ∈ $\mathcal{DOMAIN}$(vtf) ⇒ **dom** vtf(t′)={v}

- The post condition "defines" tf', the traffic resulting from merging vtf with tf:
  - Let ts be the time points of tf and vtf, a time interval.
  - The result traffic, tf', is defines as a λ-function.
  - For any t'' in the time interval
  - if t'' is less than t, the insertion time, then tf' is as tf;
  - if t'' is t or larger then tf' applied to t'', i.e., $tf'(t'')$
    - for any $v' : V$ different from v yields the same as $(tf(t))(v')$,
    - but for v it yields $(vtf(t))(v)$.

post: tf′ = λt″·
    let ts = $\mathcal{DOMAIN}$(tf) ∪ $\mathcal{DOMAIN}$(vtf) **in**
    if $\mathcal{MIN}$(ts) ≤ t″ ≤ $\mathcal{MAX}$(ts)
     then
      ((t″<t) → tf(t″),
      (t″≥t) → [v′↦ **if** v′≠v **then** (tf(t))(v′) **else** (vtf(t))(v) **end**])
     else chaos end
    end
assumption: wf_TF(vtf)∧wf_TF(tf)
theorem: wf_TF(tf′)

- We leave it as an exercise for the student to define functions for:
  - removing a vehicle from a traffic,
  - changing to course of a vehicle from an originally (or changed) vehicle traffic to another.
  - etcetera. ■ End of Example 41

A.4.6. Function Signatures

For sorts we may want to postulate some functions:

type
  A, B, ..., C
value
  ωB: A → B
  ...
  ωC: A → C

- These functions cannot be defined.
- Once a domain is presented
  - in which sort A and sorts or types B, ... and C occurs
  - these observer functions can be demonstrated.

Example 42 – Hub and Link Observers:

- Let a net with several hubs and links be presented.
- Now observer functions
  - ωHs and
  - ωLs

  can be demonstrated:
  - one simply "walks" along the net, pointing out
  - this hub and
  - that link,
  - one-by-one
  - until all the net has been visited.

- The observer functions
  - ωHI and
  - ωLI

  can be likewise demonstrated, for example:
  - when a hub is "visited"
  - its unique identification
  - can be postulated (and "calculated")
  - to be the unique geographic position of the hub
  - one which is not overlapped by any other hub (or link),
- and likewise for links. ■ End of Example 42

A.4.7. Function Definitions

Functions can be defined explicitly:

type
  A, B
value
  f: A → B [a total function]
  f(a_expr) ≡ b_expr

  g: A $\stackrel{\sim}{\to}$ B [a partial function]
  g(a_expr) ≡ b_expr
  pre P(a_expr)
  P: A → **Bool**

- a_expr, b_expr are
- A, respectively B valued expressions
- of any of the kinds illustrated in earlier and later sections of this primer.

Or functions can be defined implicitly:

value
  f: A→B
  f(a_expr) **as** b
  post P(a_expr,b)
  P: A×B→**Bool**

  g: A$\stackrel{\sim}{\to}$B
  g(a_expr) **as** b
  pre P′(a_expr)
  post P(a_expr,b)
  P′: A→**Bool**

where b is just an identifier.

- Finally functions, f, g, ..., can be defined in terms of axioms
- over function identifiers, f, g, ..., and over identiers of function arguments and results.

type
  A, B, C, D, ...
value
  f: A → B
  g: C → D
  ...
axiom
  ∀ a:A, b:B, c:C, d:D, ...
    $\mathcal{P}_1$(f,a,b) ∧ ... ∧ $\mathcal{P}_m$(f,a,b)
    ...
    $\mathcal{Q}_1$(g,c,d) ∧ ... ∧ $\mathcal{Q}_n$(g,c,d)

Example 43 – Axioms over Hubs, Links and Their Observers:

- Example 1 on page 39 Items [4]–[8]
- clearly demonstrates how a number of entities and observer functions are constrained
- (that is, partially defined)
- by function signatures and axioms. ■ End of Example 43

End of Lecture 8: RSL: PREDICATE CALCULUS and λ–CALCULUS