Subtypes

Contents

- what is a subtype? 2
- subtype expressions 2
- subtype relation 3
- maximal types 4
- type checking 7
- examples of specification using subtypes 8
- subtypes in quantified expressions 10
- subtypes in value definitions 11
- empty subtypes 14

Subtypes

A *type* is a collection of related values.

In RSL one type T_1 is said to be a *subtype* of another type T_2, if T_1 is a *subset* of T_2.

Subtype Expressions

In RSL, a subtype expression defines a type as a subtype of another type.

Examples:

$\{|\ l : \mathbf{Int}^* \bullet \mathbf{len}\ l > 0\ |\}$
$\{|\ rs : \mathrm{Record}\textbf{-set} \bullet \mathrm{is\_wf\_Database}(rs)\ |\}$

General form:

$\{|\ \mathrm{binding} : \mathrm{type\_expr} \bullet \textit{logical}\text{-}\mathrm{value\_expr}\ |\}$

Subtype Relation

If T_1 is a subtype of T_2, we write $T_1 \preceq T_2$.

$\preceq$ is reflexive and transitive

$\mathbf{Nat} \preceq \mathbf{Int}$

$T^* \preceq T^\omega$

Proving Subtype Relations

$T_1 = \{|\ id : T \bullet p_1(id)\ |\}$
$T_2 = \{|\ id : T \bullet p_2(id)\ |\}$

$T_1 \preceq T_2$?

$\forall\ id : T \bullet p_1(id) \Rightarrow p_2(id)$

Examples:

$\llcorner\{|\ j : \mathbf{Int} \bullet j \geq 7 |\} \preceq \{|\ j : \mathbf{Int} \bullet j \geq 0 |\}\lrcorner$

$\llcorner\forall\ j : \mathbf{Int} \bullet j \geq 7 \Rightarrow j \geq 0\lrcorner$

$\llcorner\mathbf{true}\lrcorner$

Maximal Types

Definition: T is *maximal* iff

$\forall\ T2 \bullet T \preceq T2 \Rightarrow T2 = T$

Maximal types:

- **Unit**, **Bool**, **Int**, **Real**, **Char** (not **Nat**, **Text**)
- composite types built by application of $\times$, **-infset**, $^\omega$, $\xrightarrow[m]{\sim}$ and $\xrightarrow{\sim}$ (not **-set**, *, $\overrightarrow{m}$, $\rightarrow$) on maximal types.

Maximal Type of

Definition:

The *maximal type of* a type T is the unique type *max*(T) for which

1. *max*(T) is maximal
2. $\mathrm{T} \preceq \mathit{max}(\mathrm{T})$

Rules for calculating the maximal type of a type:

$\mathit{max}(\mathbf{Nat}) = \mathbf{Int}$
$\mathit{max}(\mathbf{Text}) = \mathit{max}(\mathbf{Char}^*)$
$\mathit{max}(\mathbf{Int}) = \mathbf{Int}$
$\vdots$
$\mathit{max}(\mathrm{T}_1 \times \ldots \times \mathrm{T}_n) = \mathit{max}(\mathrm{T}_1) \times \ldots \times \mathit{max}(\mathrm{T}_n)$
$\mathit{max}(\mathrm{T}\text{-}\mathbf{set}) = \mathit{max}(\mathrm{T})\text{-}\mathbf{infset}$
$\mathit{max}(\mathrm{T}\text{-}\mathbf{infset}) = \mathit{max}(\mathrm{T})\text{-}\mathbf{infset}$
$\mathit{max}(\mathrm{T}^*) = \mathit{max}(\mathrm{T})^\omega$
$\mathit{max}(\mathrm{T}^\omega) = \mathit{max}(\mathrm{T})^\omega$
$\mathit{max}(\{|\ \mathrm{id} : \mathrm{T} \bullet \mathrm{expr}\ |\}) = \mathit{max}(\mathrm{T})$
$\mathit{max}(\mathrm{T}_1 \underset{m}{\rightarrow} \mathrm{T}_2) = \mathit{max}(\mathrm{T}_1) \underset{m}{\stackrel{\sim}{\rightarrow}} \mathit{max}(\mathrm{T}_2)$
$\mathit{max}(\mathrm{T}_1 \underset{m}{\stackrel{\sim}{\rightarrow}} \mathrm{T}_2) = \mathit{max}(\mathrm{T}_1) \underset{m}{\stackrel{\sim}{\rightarrow}} \mathit{max}(\mathrm{T}_2)$
$\mathit{max}(\mathrm{T}_1 \stackrel{\sim}{\rightarrow} \mathrm{T}_2) = \mathit{max}(\mathrm{T}_1) \stackrel{\sim}{\rightarrow} \mathit{max}(\mathrm{T}_2)$
$\mathit{max}(\mathrm{T}_1 \rightarrow \mathrm{T}_2) = \mathit{max}(\mathrm{T}_1) \stackrel{\sim}{\rightarrow} \mathit{max}(\mathrm{T}_2)$

Maximal Type of

Fact:

Any type can be expressed as a subtype of its maximal type:

$\mathrm{T} = \{|\ \mathrm{t} : \mathit{max}(\mathrm{T}) \bullet \mathrm{p}(\mathrm{t})\ |\}$ for some p

Examples:

$\mathbf{Nat} = \{|\ \mathrm{i} : \mathbf{Int} \bullet \mathrm{i} \geq 0\ |\}$
$\mathrm{S} \rightarrow \mathrm{T} \ = \ \{|\ \mathrm{f} : \mathrm{S} \stackrel{\sim}{\rightarrow} \mathrm{T} \bullet \forall\, \mathrm{x} : \mathrm{S} \bullet \mathrm{f}(\mathrm{x})\ \mathbf{post\ true}\ |\}$

Maximal Type Checking

- subtype checking is not statically decidable
- the type checker performs only maximal type checking

Example:

value $\mathrm{n} : \{|\ \mathrm{i} : \mathbf{Int} \bullet \mathrm{i} \geq 2\ |\}$
axiom $\mathrm{n} = 1$

is statically ok, but inconsistent!

Subtype Example

scheme EQUIVALENCE_RELATION =
  class
    type
      Elem,
      Partition_Id,
      Relation = $\{|\ \mathrm{r} : \mathrm{Elem} \underset{m}{\stackrel{\sim}{\rightarrow}} \mathrm{Partition\_Id} \bullet \mathrm{is\_wf\_Relation(r)}\ |\}$
    value
      is_wf_Relation : $(\mathrm{Elem} \underset{m}{\stackrel{\sim}{\rightarrow}} \mathrm{Partition\_Id}) \rightarrow \mathbf{Bool}$
      is_wf_Relation(m) $\equiv$
        $(\forall\ \mathrm{e} : \mathrm{Elem} \bullet \mathrm{e} \in \mathbf{dom}\ \mathrm{m} \wedge$
          $(\exists!\ \mathrm{p} : \mathrm{Partition\_Id} \bullet \mathrm{r(e)} \equiv \mathrm{p}\ )),$

      initial : Relation,

      make_equivalent : $\mathrm{Elem} \times \mathrm{Elem} \times \mathrm{Relation} \rightarrow \mathrm{Relation}$
      make_equivalent(e1,e2,r) $\equiv$
        $\mathrm{r} \dagger [\,\mathrm{e} \mapsto \mathrm{r(e2)} \mid \mathrm{e} : \mathrm{Elem} \bullet \mathrm{r(e)} = \mathrm{r(e1)}\,],$

      are_equivalent : $\mathrm{Elem} \times \mathrm{Elem} \times \mathrm{Relation} \rightarrow \mathbf{Bool}$
      are_equivalent(e1,e2,r) $\equiv \mathrm{r(e1)} = \mathrm{r(e2)}$

    axiom
      $\forall\ \mathrm{e1, e2} : \mathrm{Elem} \bullet\ \mathrm{e1} \neq \mathrm{e2} \Rightarrow \mathrm{initial(e1)} \neq \mathrm{initial(e2)},$
  end

Subtype Example

scheme QUEUE =
  class
    type
      Element,
      Queue = {| q : Element* • **len** q ≤ max |},
      Extensible_Queue = {| q : Queue • **len** q < max |},
      Reducible_Queue = {| q : Queue • q ≠ empty |}

    value
      max : **Nat** • max > 0,

      empty : Extensible_Queue = ⟨⟩,

      put : Element × Extensible_Queue → Reducible_Queue
      put(e,eq) ≡ eq ⌢ ⟨e⟩,

      get : Reducible_Queue → Extensible_Queue × Element
      get(rq) ≡ (**tl** rq,**hd** rq)
  end

Subtypes in Quantified Expressions

Examples

∀ x : {| id : **Int** • id ≥ 2 |} • x ≠ 0
equivalent to:
∀ x : **Int** • x ≥ 2 ⇒ x ≠ 0

∃ x : {| id : **Int** • id ≥ 2 |} • x = 3
equivalent to:
∃ x : **Int** • x ≥ 2 ∧ x = 3

Generally

Assume p[id] convergent for id : T.

∀ x : {| id : T • p[id] |} • e[x]
equivalent to:
∀ x : T • p[x] ⇒ e[x]

∃ x : {| id : T • p[id] |} • e[x]
equivalent to:
∃ x : T • p[x] ∧ e[x]

Subtypes in Constant Definitions

Example

value c : {| i : **Int** • i ≥ 1 |}
equivalent to
value c : **Int**
axiom c ≥ 1 (or **axiom** c ∈ { i | i : **Int** • i ≥ 1 })

value c : {| i : **Int** • i ≥ 1 |} = 2

consistent as 2 ≥ 1

Generally

Assume p[id] convergent for id : T.

value c : {| id : T • p[id] |}
equivalent to
value c : T
axiom p[c] (or **axiom** c ∈ { id | id : T • p[id] })

value c : {| id : T • p[id] |} = e

consistent only if p[e] is true

Subtypes in Function Definitions

Example

value f : {| i : **Int** • i ≥ 1 |} → {| i : **Int** • i ≥ 2 |}

equivalent to

value f : **Int** $\xrightarrow{\sim}$ **Int**
axiom ∀ x : **Int** • x ≥ 1 ⇒ (f(x) **post true** ∧ f(x) ≥ 2)

equivalent to

value f : **Int** $\xrightarrow{\sim}$ **Int**
axiom ∀ x : **Int** • f(x) **as** y **post** y ≥ 2 **pre** x ≥ 1

Generally

Assume p1[id] convergent for id : T1.

value f : {| id : T1 • p1[id] |} → {| id : T2 • p2[id] |}

equivalent to

value f : T1 $\xrightarrow{\sim}$ T2
axiom ∀ x : T1 • p1[x] ⇒ (f(x) **post true** ∧ p2[f(x)])

Subtypes in Explicit Function Definitions

Example

value f : {| i : **Int** • i ≠ 0 |} → **Int**
  f(x) ≡ 1 / x

equivalent to

value f : **Int** $\stackrel{\sim}{\rightarrow}$ **Int**
  f(x) ≡ 1 / x
  pre x ≠ 0

Generally

value f : {| id : T1 • is_wff(id) |} → T2
  f(x) ≡ e[x]

equivalent to

value f : T1 $\stackrel{\sim}{\rightarrow}$ T2
  f(x) ≡ e[x]
  pre is_wff(x)

when ∀ x : T1 • e[x] **post true pre** is_wff(x) is true

Empty Subtypes

type NoInt = {| i : **Int** • **false** |}

value no_constant : NoInt
equivalent to
value no_constant : **Int**
axiom no_constant ∈ { i | i : **Int** • **false** }

value no_fun : **Unit** → NoInt
equivalent to
value no_fun : **Unit** → **Int**
axiom no_fun() ∈ { i | i : **Int** • **false** }

value no_arg_fun : NoInt → **Int**
equivalent to
value no_arg_fun : Int $\stackrel{\sim}{\rightarrow}$ **Int**
axiom ∀ x : **Int** • **false** ⇒ f(x) **post true**

axiom
  ∀ x : NoInt • **false**, ∼(∃ x : NoInt), { x | x : NoInt } = {}
equivalent to
axiom true, **true**, **true**