Maps
Contents

What is a map?

A map is: an unordered collection of pair of values.

Examples:

$[\texttt{"Klaus"} \mapsto 7, \texttt{"John"} \mapsto 2, \texttt{"Mary"} \mapsto 7]$
$[1 \mapsto 2, 5 \mapsto 10]$

Maps my be:

- infinite
- partial
- non-deterministic on applications

Map Type Expressions

- $\text{type\_expr}_1 \ \widetilde{\overrightarrow{m}} \ \text{type\_expr}_2$

  Denote the type consisting of the map values:

  $[v_1 \mapsto w_1, \ldots, v_n \mapsto w_n]$,
  $[v_1 \mapsto w_1, \ldots, v_n \mapsto w_n, \ldots]$,

  where $n \geq 0$, $v_i : \text{type\_expr}_1$, $w_i : \text{type\_expr}_2$

- $\text{type\_expr}_1 \ \overrightarrow{m} \ \text{type\_expr}_2$

  $T1 \ \overrightarrow{m} \ T2 \simeq$
  $\{| \ m : T1 \ \widetilde{\overrightarrow{m}} \ T2 \bullet$
  $\quad \textbf{card dom } m \textbf{ post true} \wedge$
  $\quad (\forall x : T1 \bullet x \in \textbf{dom } m \Rightarrow (m(x) \textbf{ post true}))$
  $| \}$

NOTE: this differs from the RSL book!

1. $\widetilde{\overrightarrow{m}}$ is used instead of $\overrightarrow{m}$
2. $\overrightarrow{m}$ is now used for a subtype of maps that have finite domains and are deterministic on application

Map Type Expressions
Example

Bool $\widetilde{\overrightarrow{m}}$ **Bool**

denote the type consisting of the maps:

$[\,]$

$[\textbf{true} \mapsto \textbf{true}]$
$[\textbf{true} \mapsto \textbf{false}]$
$[\textbf{false} \mapsto \textbf{true}]$
$[\textbf{false} \mapsto \textbf{false}]$

$[\textbf{true} \mapsto \textbf{true}, \textbf{false} \mapsto \textbf{true}]$
$[\textbf{true} \mapsto \textbf{false}, \textbf{false} \mapsto \textbf{true}]$
$[\textbf{true} \mapsto \textbf{true}, \textbf{false} \mapsto \textbf{false}]$
$[\textbf{true} \mapsto \textbf{false}, \textbf{false} \mapsto \textbf{false}]$

$[\textbf{true} \mapsto \textbf{true}, \textbf{true} \mapsto \textbf{false}]$
$[\textbf{false} \mapsto \textbf{true}, \textbf{false} \mapsto \textbf{false}]$
$\vdots$
$[\textbf{true} \mapsto \textbf{true}, \textbf{true} \mapsto \textbf{false},$
$\textbf{false} \mapsto \textbf{true}, \textbf{false} \mapsto \textbf{false}]$

Map Type Expressions
Example

Bool $\overrightarrow{m}$ **Bool**

denote the type consisting of the maps:

[ ]

[**true** ↦ **true**]
[**true** ↦ **false**]
[**false** ↦ **true**]
[**false** ↦ **false**]

[**true** ↦ **true**, **false** ↦ **true**]
[**true** ↦ **false**, **false** ↦ **true**]
[**true** ↦ **true**, **false** ↦ **false**]
[**true** ↦ **false**, **false** ↦ **false**]

Map Type Expressions
Example

Nat $\overrightarrow{m}$ **Bool**

denote the type consisting of the maps:

[ ]
[0 ↦ **true**]
⋮
[1 ↦ **true**]
⋮
[0 ↦ **true**, 1 ↦ **true**]
⋮
[0 ↦ **true**, 1 ↦ **true**, . . .]
⋮

Map Value Expressions

Enumerated:

[3 ↦ **true**, 5 ↦ **false**]
[″`Klaus`″ ↦ 7, ″`John`″ ↦ 2, ″`Mary`″ ↦ 7]

[$\text{expr}_1 \mapsto \text{expr}_1'$,. . . ,$\text{expr}_n \mapsto \text{expr}_n'$]

Comprehended:

[n ↦ 2∗n | n : **Nat** • n ≤ 2] = [0 ↦ 0, 1 ↦ 2, 2 ↦ 4]
[n ↦ 2∗n | n : **Nat**] = [0 ↦ 0, 1 ↦ 2, 2 ↦ 4, . . .]

[$\text{expr}_1 \mapsto \text{expr}_2$ | typing_1,...,typing_n • *logical*-expr_3]

Map Application

Basic form:

map-$\text{expr}(\text{expr}_1)$

Examples:

[″`Klaus`″ ↦ 7, ″`John`″ ↦ 2, ″`Mary`″ ↦ 7](″`John`″) = 2

[3 ↦ **true**, 3 ↦ **false**](3) = **true** [] **false**

Derived form:

map-$\text{expr}(\text{expr}_1)$...$(\text{expr}_n)$

Example:

[1 ↦ [″`Per`″ ↦ 5, ″`Jan`″ ↦ 7] , 2 ↦ [ ]](1)(″`Jan`″)

Built-in Map Operators

dom : $(T_1 \xrightarrow[m]{\sim} T_2) \to T_1$**-infset**
rng : $(T_1 \xrightarrow[m]{\sim} T_2) \to T_2$**-infset**
$\dagger : (T_1 \xrightarrow[m]{\sim} T_2) \times (T_1 \xrightarrow[m]{\sim} T_2) \to (T_1 \xrightarrow[m]{\sim} T_2)$
$\cup : (T_1 \xrightarrow[m]{\sim} T_2) \times (T_1 \xrightarrow[m]{\sim} T_2) \to (T_1 \xrightarrow[m]{\sim} T_2)$
$\setminus : (T_1 \xrightarrow[m]{\sim} T_2) \times T_1$**-infset** $\to (T_1 \xrightarrow[m]{\sim} T_2)$
$/ : (T_1 \xrightarrow[m]{\sim} T_2) \times T_1$**-infset** $\to (T_1 \xrightarrow[m]{\sim} T_2)$
$\circ : (T_2 \xrightarrow[m]{\sim} T_3) \times (T_1 \xrightarrow[m]{\sim} T_2) \to (T_1 \xrightarrow[m]{\sim} T_3)$

Map Operator — Examples and Definitions

Let m be a deterministic map.

rng $m = \{m(d) \mid d : T_1 \cdot d \in$ **dom** $m\}$

$[3 \mapsto$ **true**, $5 \mapsto$ **false**$] \dagger [5 \mapsto$ **true**$]$
$\equiv [3 \mapsto$ **true**, $5 \mapsto$ **true**$]$
$[3 \mapsto$ **true**, $5 \mapsto$ **false**$] \cup [5 \mapsto$ **true**$]$
$\equiv [3 \mapsto$ **true**, $5 \mapsto$ **false**, $5 \mapsto$ **true**$]$

$m \setminus s = [d \mapsto m(d) \mid d : T_1 \cdot d \in$ **dom** $m \wedge d \notin s]$
$m / s = [d \mapsto m(d) \mid d : T_1 \cdot d \in$ **dom** $m \wedge d \in s]$

$[3 \mapsto$ **true**, $5 \mapsto$ **false**$] \setminus \{5, 7\} = [3 \mapsto$ **true**$]$
$[3 \mapsto$ **true**, $5 \mapsto$ **false**$] / \{3, 5, 7\}$
$= [3 \mapsto$ **true**, $5 \mapsto$ **false**$]$

$m_1 \circ m_2 =$
$[x \mapsto m_1(m_2(x)) \mid x : T_1 \cdot$
$x \in$ **dom** $m_2 \wedge m_2(x) \in$ **dom** $m_1]$

$[3 \mapsto$ **true**, $5 \mapsto$ **false**$] \circ [$"Klaus" $\mapsto 3,$ "John" $\mapsto 7]$
$= [$"Klaus" $\mapsto$ **true**$]$

Map Example 1
Database Representations

type Database = (Key × Data)**-set**

$\{\}$
$\{(k_1,d_1), (k_2,d_2)\}$

type Database = (Key × Data)*

$\langle\rangle$
$\langle(k_1,d_1), (k_2,d_2)\rangle$

type Database = Key $\xrightarrow[m]{}$ Data

$[\,]$
$[k_1 \mapsto d_1, k_2 \mapsto d_2]$

Map Example 1

scheme MAP_DATABASE =
class
type
Database = Key $\xrightarrow[m]{}$ Data,
Key, Data
value
empty : Database = [ ],

insert : Key × Data × Database → Database
insert(k,d,db) ≡ db † [k ↦ d],

remove : Key × Database → Database
remove(k,db) ≡ db \ {k},

defined : Key × Database → **Bool**
defined(k,db) ≡ k ∈ **dom** db,

lookup : Key × Database $\xrightarrow{\sim}$ Data
lookup(k,db) ≡ db(k)
pre defined(k,db)
end

Map Example 2

scheme EQUIVALENCE_RELATION =
  class
    type
      Elem, Partition_Id,
      Relation = Elem $\xrightarrow[m]{}$ Partition_Id
    value
      is_wf_Relation : Relation → **Bool**
      is_wf_Relation(r) ≡
      (∀ e : Elem • e ∈ **dom** r ∧ (∃! p : Partition_Id • r(e) ≡ p)),

      initial : Relation,

      make_equivalent : Elem × Elem × Relation ~→ Relation
      make_equivalent(e1,e2,r) ≡
        r † [e ↦ r(e2) | e : Elem • r(e) = r(e1)]
        pre {e1,e2} ⊆ **dom** r,

      are_equivalent : Elem × Elem × Relation ~→ **Bool**
      are_equivalent(e1,e2,r) ≡ r(e1) = r(e2)
      pre {e1,e2} ⊆ **dom** r

    axiom
      is_wf_Relation(initial),
      ∀ e1, e2 : Elem • e1 ≠ e2 ⇒ initial(e1) ≠ initial(e2)
  end

Map Example 3

scheme BILL_OF_PRODUCTS =
  class
    type
      Product,
      Bop = Product $\xrightarrow[m]{}$ Product-**set**
    value
      is_wf_Bop : Bop → **Bool**
      is_wf_Bop(bop) ≡
        (∀ ps : Product-**set** •
          ps ∈ **rng** bop ⇒ ps ⊆ **dom** bop) ∧
        (∀ p : Product •
          p ∈ **dom** bop ⇒ p ∉ sub_products(p,bop)),

      sub_products : Product × Bop ~→ Product-**infset**
      sub_products(p,bop) ≡
        {p_sub | p_sub : Product • depends_on(p,p_sub,bop)}
      pre p ∈ **dom** bop,

      depends_on : Product × Product × Bop ~→ **Bool**,
      depends_on(p1,p2,bop) ≡
        p2 ∈ bop(p1) ∨
        (∃ p : Product •
          (p ∈ bop(p1) ∧ depends_on(p,p2,bop)))
      pre p1 ∈ **dom** bop

Map Example 3 — Continued

  empty : Bop = [ ],

  enter : Product × Product-**set** × Bop ~→ Bop
  enter(p,ps,bop) ≡ bop ∪ [p ↦ ps]
  pre p ∉ **dom** bop ∧ ps ⊆ **dom** bop,

  delete : Product × Bop ~→ Bop
  delete(p,bop) ≡ bop\{p}
  pre p ∈ **dom** bop ∧
    ∼(∃ ps : Product-**set** • ps ∈ **rng** bop ∧ p ∈ ps),

  add : Product × Product × Bop ~→ Bop
  add(p1,p2,bop) ≡ bop † [p1 ↦ bop(p1) ∪ {p2}]
  pre
    {p1,p2} ⊆ **dom** bop ∧ p1 ≠ p2 ∧ p2 ∉ bop(p1) ∧
    p1 ∉ sub_products(p2,bop),

  erase : Product × Product × Bop ~→ Bop
  erase(p1,p2,bop) ≡ bop † [p1 ↦ bop(p1)\{p2}]
  pre p1 ∈ **dom** bop ∧ p2 ∈ bop(p1)
end

Cycles in a bill-of-products

The well-formedness requirement that there should be no cycles in a bill-of-products (page 81 in the book) could be specified in a shorter way:

value
  is_wf_Bop: Bop → **Bool**
  is_wf_Bop(bop) ≡
    (∀ s : Product-**set** • s ∈ **rng** bop ⇒ s ⊆ **dom** bop) ∧
    ∼cyclic(bop)

  cyclic : Bop → **Bool**
  cyclic(bop) ≡
    (
      ∃ ps : Product-**set** •
        ps ⊆ **dom** bop ∧
        card ps > 0 ∧
        (∀ p : Product • p ∈ ps ⇒ bop(p) ∩ ps ≠ {})
)