DTU Technical University of Denmark
Informatics and Mathematical Modelling
Computer Science and Technology

Sets

Contents

DTU Technical University of Denmark
Informatics and Mathematical Modelling
Computer Science and Technology

What is a set?

A set is: an unordered collection of distinct values of the same type.

Examples:

{1,3,5}
{"John","Peter","Ann"}

DTU Technical University of Denmark
Informatics and Mathematical Modelling
Computer Science and Technology

Set Type Expressions

- type_expr-**set** denotes the type consisting of all finite sets

  $\{v_1, \ldots, v_n\}$

  where $n \geq 0$, v_i : type_expr
- type_expr-**infset** denotes the type consisting of all sets

  $\{v_1, \ldots, v_n\}$,
  $\{v_1, \ldots, v_n, \ldots\}$

  where $n \geq 0$, v_i : type_expr

Operators

$\cup, \cap, \backslash, \in, \notin, \subset, \subseteq, \supset, \supseteq,$ **card**

card $\{n \mid n : \textbf{Nat}\} \equiv$ **chaos**

DTU Technical University of Denmark
Informatics and Mathematical Modelling
Computer Science and Technology

Set Type Expressions

Examples

Bool-set

denotes the type consisting of the sets:

{}
{**true**}
{**false**}
{**true**,**false**}

Nat-set

denotes the type consisting of the sets:

{}
{0}
{1}
...
{0,1}
...
{1,2,3}
...

Examples

Bool-infset = **Bool-set**

Nat-infset

denotes the type consisting of the sets:

{}
{0}
{1}
...
{0,1}
...
{1,2,3}
...
{0,1,2,3,4,...}
{2,3,5,7,...}
...

Set Value Expressions

Enumerated:

{1,2}
{1,2,1}

$\{\text{value\_expr}_1,\ldots,\text{value\_expr}_n\}$

Ranged:

$\{3 .. 7\} = \{3,4,5,6,7\}$
$\{3 .. 3\} = \{3\}$
$\{3 .. 2\} = \{\}$

$\{\text{value\_expr}_1 .. \text{value\_expr}_2\}$

Comprehended:

$\{2*n \mid n : \textbf{Nat} \cdot n \leq 3\}$

$\{\text{value\_expr}_1 \mid \text{typing}_1,\ldots,\text{typing}_n \cdot \text{value\_expr}_2\}$

Set Example I

```
scheme RESOURCE_MANAGER =
  class
    type
      Resource,
      Pool = Resource-set

    value
      obtain : Pool ~→ Pool × Resource
      obtain(p) as (p1,r1) post r1 ∈ p ∧ p1 = p\{r1}
      pre p ≠ {},

      release : Resource × Pool ~→ Pool
      release(r,p) ≡ p ∪ {r}
      pre r ∉ p
  end
```


Set Example II

```
scheme EQUIVALENCE_RELATION =
  class
    type
      Elem,
      Class = Elem-infset,
      Relation = Class-infset
    value
      is_wf_Relation : Relation → Bool
      is_wf_Relation(r) ≡
        {} ∉ r ∧
        (∀ e : Elem • ∃ c : Class • c ∈ r ∧ e ∈ c) ∧
        (∀ c1,c2 : Class •
          c1 ∈ r ∧ c2 ∈ r ∧ c1 ≠ c2 ⇒ c1 ∩ c2 = {}),
      initial : Relation = {{e} | e : Elem},
      make_equivalent : Elem × Elem × Relation → Relation
      make_equivalent(e1,e2,r) ≡
        {c | c : Class • c ∈ r ∧ {e1,e2} ∩ c = {}} ∪
        {c1 ∪ c2 | c1,c2 : Class •
          c1 ∈ r ∧ c2 ∈ r ∧ e1 ∈ c1 ∧ e2 ∈ c2},
      are_equivalent : Elem × Elem × Relation → Bool
      are_equivalent(e1,e2,r) ≡
        (∃ c : Class • c ∈ r ∧ e1 ∈ c ∧ e2 ∈ c)
  end
```

Set Example II — continued

Are the following properties true for EQUIVALENCE_RELATION ?

(1) is_wf_Relation(initial),

(2) $\forall$ e_1, e_1 : Elem, r : Relation •
is_wf_Relation(r) $\Rightarrow$
is_wf_Relation(make_equivalent(e_1,e_2,r))

Set Example III

scheme SET_DATABASE =
  class
    type
      Record = Key × Data,
      Database = {| rs : Record-**set** • is_wf_Database(rs) |},
      Key, Data
    value
      is_wf_Database : Record-**set** → **Bool**
      is_wf_Database(rs) ≡
        (∀ k : Key, d1, d2 : Data •
          ((k,d1) ∈ rs ∧ (k,d2) ∈ rs) ⇒ d1 = d2),
      empty : Database = {},
      insert : Key × Data × Database → Database
      insert(k,d,db) ≡ remove(k,db) ∪ {(k,d)},
      remove : Key × Database → Database
      remove(k,db) ≡ db \ {(k,d) | d : Data • **true**},
      defined : Key × Database → **Bool**
      defined(k,db) ≡ (∃ d : Data • (k,d) ∈ db),
      lookup : Key × Database $\xrightarrow{\sim}$ Data
      lookup(k,db) **as** d **post** (k,d) ∈ db
        pre defined(k,db)
  end

Set Example III — continued

Are the following properties true for SET_DATABASE?

is_wf_Database(empty),

∀ k : Key, d : Data, db : Database • is_wf_Database(insert(k,d,db)),

∀ k : Key, db : Database • is_wf_Database(remove(k,db))