Funcons-beta: Sets

The PLanCompS Project

`Funcons-beta/Values/Composite/Sets/Sets.cbs`*

Sets

[*Type* sets
Funcon set
Funcon set-elements
Funcon is-in-set
Funcon is-subset
Funcon set-insert
Funcon set-unite
Funcon set-intersect
Funcon set-difference
Funcon set-size
Funcon some-element
Funcon element-not-in]

Meta-variables GT <: ground-values

Built-in Type sets(GT)

sets(GT) is the type of possibly-empty finite sets $\{V_1, \cdots, V_n\}$ where $V_1 : GT$, ..., $V_n : GT$.

Built-in Funcon set(_ : $(GT)^*$) : ⇒ sets(GT)

The notation $\{V_1, \cdots, V_n\}$ for set($V_1, \cdots, V_n$) is built-in.

*Suggestions for improvement: `plancomps@gmail.com`.
Issues: `https://github.com/plancomps/CBS-beta/issues`.

Assert $\{V^* : (GT)^*\} == \mathsf{set}(V^*)$

Note that $\mathsf{set}(\cdots)$ is not a constructor operation. The order and duplicates of argument values are ignored (e.g., $\{1, 2, 1\}$ denotes the same set as $\{1, 2\}$ and $\{2, 1\}$).

Built-in Funcon $\mathsf{set\text{-}elements}(\_ : \mathsf{sets}(GT)) : \Rightarrow (GT)^*$

For each set S, the sequence of values V^* returned by $\mathsf{set\text{-}elements}(S)$ contains each element of S just once. The order of the values in V^* is unspecified, and may vary between sets (e.g., $\mathsf{set\text{-}elements}\ \{1, 2\}$ could be $(1, 2)$ and $\mathsf{set\text{-}elements}\ \{1, 2, 3\}$ could be $(3, 2, 1)$).

Assert $\mathsf{set}(\mathsf{set\text{-}elements}(S)) == S$

Built-in Funcon $\mathsf{is\text{-}in\text{-}set}(\_ : GT, \_ : \mathsf{sets}(GT)) : \Rightarrow \mathsf{booleans}$

$\mathsf{is\text{-}in\text{-}set}(GV, S)$ tests whether GV is in the set S.

Assert $\mathsf{is\text{-}in\text{-}set}(GV : GT, \{\ \}) == \mathsf{false}$
Assert $\mathsf{is\text{-}in\text{-}set}(GV : GT, \{GV\} : \mathsf{sets}(GT)) == \mathsf{true}$

Built-in Funcon $\mathsf{is\text{-}subset}(\_ : \mathsf{sets}(GT), \_ : \mathsf{sets}(GT)) : \Rightarrow \mathsf{booleans}$

$\mathsf{is\text{-}subset}(S_1, S_2)$ tests whether S_1 is a subset of S_2.

Assert $\mathsf{is\text{-}subset}(\{\ \}, S : \mathsf{sets}(GT)) == \mathsf{true}$
Assert $\mathsf{is\text{-}subset}(S : \mathsf{sets}(GT), S) == \mathsf{true}$

Built-in Funcon $\mathsf{set\text{-}insert}(\_ : GT, \_ : \mathsf{sets}(GT)) : \Rightarrow \mathsf{sets}(GT)$

$\mathsf{set\text{-}insert}(GV, S)$ returns the set union of $\{GV\}$ and S.

Assert $\mathsf{is\text{-}in\text{-}set}(GV : GT, \mathsf{set\text{-}insert}(GV : GT, S : \mathsf{sets}(GT))) == \mathsf{true}$

Built-in Funcon set-unite(_ : (sets(GT))*) : ⇒ sets(GT)

set-unite(···) unites a sequence of sets.

Assert set-unite(S : sets(GT), S) == S
Assert set-unite(S1 : sets(GT), S2 : sets(GT)) == set-unite(S2, S1)
Assert set-unite(S1 : sets(GT), set-unite(S2 : sets(GT), S3 : sets(GT))) == set-unite(set-unite(S1, S2), S3)
Assert set-unite(S1 : sets(GT), S2 : sets(GT), S3 : sets(GT)) == set-unite(S1, set-unite(S2, S3))
Assert set-unite(S : sets(GT)) == S
Assert set-unite( ) == { }

Built-in Funcon set-intersect(_ : (sets(GT))+) : ⇒ sets(GT)

set-intersect(GT, ···) intersects a non-empty sequence of sets.

Assert set-intersect(S : sets(GT), S) == S
Assert set-intersect(S1 : sets(GT), S2 : sets(GT)) == set-intersect(S2, S1)
Assert set-intersect(S1 : sets(GT), set-intersect(S2 : sets(GT), S3 : sets(GT))) == set-intersect(set-intersect
Assert set-intersect(S1 : sets(GT), S2 : sets(GT), S3 : sets(GT)) == set-intersect(S1, set-intersect(S2, S3))
Assert set-intersect(S : sets(GT)) == S

Built-in Funcon set-difference(_ : sets(GT), _ : sets(GT)) : ⇒ sets(GT)

set-difference(S1, S2) returns the set containing those elements of S1 that are not in S2.

Built-in Funcon set-size(_ : sets(GT)) : ⇒ natural-numbers

Assert set-size(S : sets(GT)) == length(set-elements(S))

Funcon some-element(_ : sets(GT)) : ⇒ GT?
Assert some-element(S : sets(GT)) == index(1, set-elements(S))
Assert some-element { } == ( )

Built-in Funcon element-not-in(GT : types, _ : set(GT)) : ⇒ GT?

element-not-in(GT, S) gives an element of the type GT not in the set S, or ( ) when S is empty. When the set of elements of GT is infinite, element-not-in(GT, S) never gives ( ).