Funcons-beta: Sets *

The PLanCompS Project

`Sets.cbs` | PLAIN | PRETTY

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 $V_1 : GT, \ldots, V_n : GT$.

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

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

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

Note that 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 set-elements($\_ :$ sets(GT)) : $\Rightarrow (GT)^*$

For each set S, the sequence of values V^* returned by set-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., set-elements $\{1, 2\}$ could be $(1, 2)$ and set-elements $\{1, 2, 3\}$ could be $(3, 2, 1)$).

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

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

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

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

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

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

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

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

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

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

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

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

$\text{set-unite}(\cdots)$ unites a sequence of sets.

Assert $\text{set-unite}(S : \text{sets}(GT), S) == S$
Assert $\text{set-unite}(S_1 : \text{sets}(GT), S_2 : \text{sets}(GT)) == \text{set-unite}(S_2, S_1)$
Assert $\text{set-unite}(S_1 : \text{sets}(GT), \text{set-unite}(S_2 : \text{sets}(GT), S_3 : \text{sets}(GT)))$
$== \text{set-unite}(\text{set-unite}(S_1, S_2), S_3)$
Assert $\text{set-unite}(S_1 : \text{sets}(GT), S_2 : \text{sets}(GT), S_3 : \text{sets}(GT))$
$== \text{set-unite}(S_1, \text{set-unite}(S_2, S_3))$
Assert $\text{set-unite}(S : \text{sets}(GT)) == S$
Assert $\text{set-unite}(\ ) == \{\ \}$

Built-in Funcon $\text{set-intersect}(\_ : (\text{sets}(GT))^+) : \Rightarrow \text{sets}(GT)$

$\text{set-intersect}(GT, \cdots)$ intersects a non-empty sequence of sets.

Assert $\text{set-intersect}(S : \text{sets}(GT), S) == S$
Assert $\text{set-intersect}(S_1 : \text{sets}(GT), S_2 : \text{sets}(GT)) == \text{set-intersect}(S_2, S_1)$
Assert $\text{set-intersect}(S_1 : \text{sets}(GT), \text{set-intersect}(S_2 : \text{sets}(GT), S_3 : \text{sets}(GT)))$
$== \text{set-intersect}(\text{set-intersect}(S_1, S_2), S_3)$
Assert $\text{set-intersect}(S_1 : \text{sets}(GT), S_2 : \text{sets}(GT), S_3 : \text{sets}(GT))$
$== \text{set-intersect}(S_1, \text{set-intersect}(S_2, S_3))$
Assert $\text{set-intersect}(S : \text{sets}(GT)) == S$

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

set-difference(S_1, S_2) returns the set containing those elements of S_1 that are not in S_2.

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 ( ).