Funcons-beta: Maps

The PLanCompS Project

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

Maps

[*Type* maps
Funcon map
Funcon map-elements
Funcon map-lookup
Alias lookup
Funcon map-domain
Alias dom
Funcon map-override
Funcon map-unite
Funcon map-delete]

Meta-variables GT <: ground-values $T?$ <: values?

Built-in Type maps(GT, $T?$)

maps(GT, $T?$) is the type of possibly-empty finite maps from values of type GT to optional values of type $T?$.

Built-in Funcon map(_ : (tuples(GT, $T?$))*) : ⇒(maps(GT, $T?$))?

map(tuple(K_1, $V_1?$), $\cdots$, tuple(K_n, $V_n?$)) constructs a map from K_1 to $V_1?$, ..., K_n to $V_n?$, provided that K_1, ..., K_n are distinct, otherwise the result is ( ).

Note that map($\cdots$) is not a constructor operation.

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

The built-in notation $\{K_1 \mapsto V_1?, \cdots, K_n \mapsto V_n?\}$ is equivalent to $\mathsf{map}(\mathsf{tuple}(K_1, V_1?), \cdots, \mathsf{tuple}(K_n, V_n?))$. Note however that in general, maps cannot be identified with sets of tuples, since the values $V_i?$ are not restricted to $\mathsf{ground\text{-}values}$.

When $T? <: \mathsf{types}$, $\mathsf{maps}(GT, T?) <: \mathsf{types}$. The type $MT : \mathsf{maps}(GT, T?)$ represents the set of value-maps $MV : \mathsf{maps}(GT, \mathsf{values}?)$ such that $\mathsf{dom}(MV)$ is a subset of $\mathsf{dom}(MT)$ and for all K in $\mathsf{dom}(MV)$, $\mathsf{map\text{-}lookup}(MV, K) : \mathsf{map\text{-}lookup}(MT, K)$.

Built-in Funcon $\mathsf{map\text{-}elements}(\_ : \mathsf{maps}(GT, T?)) : \Rightarrow(\mathsf{tuples}(GT, T?))^*$

The sequence of tuples $(\mathsf{tuple}(K_1, V_1?), \cdots, \mathsf{tuple}(K_n, V_n?))$ given by $\mathsf{map\text{-}elements}(M)$ contains each mapped value K_i just once. The order of the elements is unspecified, and may vary between maps.

Assert $\mathsf{map}(\mathsf{map\text{-}elements}(M)) == M$

Built-in Funcon $\mathsf{map\text{-}lookup}(\_ : \mathsf{maps}(GT, T?), K : GT) : \Rightarrow(T?)?$

Alias $\mathsf{lookup} = \mathsf{map\text{-}lookup}$

$\mathsf{map\text{-}lookup}(M, K)$ gives the optional value to which K is mapped by M, if any, and otherwise $(\,)$.

Built-in Funcon $\mathsf{map\text{-}domain}(\_ : \mathsf{maps}(GT, T?)) : \Rightarrow \mathsf{sets}(GT)$

Alias $\mathsf{dom} = \mathsf{map\text{-}domain}$

$\mathsf{map\text{-}domain}(M)$ gives the set of values mapped by M.

$\mathsf{map\text{-}lookup}(M, K)$ is always $(\,)$ when K is not in $\mathsf{map\text{-}domain}(M)$.

Built-in Funcon $\mathsf{map\text{-}override}(\_ : (\mathsf{maps}(GT, T?))^*) : \Rightarrow \mathsf{maps}(GT, T?)$

$\mathsf{map\text{-}override}(\cdots)$ takes a sequence of maps. It returns the map whose domain is the union of their domains, and which maps each of those values to the same optional value as the first map in the sequence in whose domain it occurs . When the domains of the M^* are disjoint, $\mathsf{map\text{-}override}(M^*)$ is equivalent to $\mathsf{map\text{-}unite}(M^*)$.

Built-in Funcon $\mathsf{map\text{-}unite}(\_ : (\mathsf{maps}(GT, T?))^*) : \Rightarrow(\mathsf{maps}(GT, T?))?$

$\textsf{map-unite}(\cdots)$ takes a sequence of maps. It returns the map whose domain is the union of their domains, and which maps each of those values to the same optional value as the map in the sequence in whose domain it occurs, provided that those domains are disjoint - otherwise the result is $(\ )$.

Built-in Funcon $\textsf{map-delete}(\_ : \textsf{maps}(GT, T?), \_ : \textsf{sets}(GT)) : \Rightarrow \textsf{maps}(GT, T?)$

$\textsf{map-delete}(M, S)$ takes a map M and a set of values S, and returns the map obtained from M by removing S from its domain.

Assert $\textsf{map-domain}(\textsf{map-delete}(M, S)) == \textsf{set-difference}(\textsf{map-domain}(M), S)$