

list<T> is the type of lists whose members are all of type T.

 alist<A, B> is the type of assoc lists with (A B) pairs

 hash<A, B> is the type of hash tables from A to B

 conj<T> is the type of conjunctions of values of type T,
 e.g. (and foo bar) : conj<symbol>

 disj<T> is similar for disjunctions (or).

 A | B is the type that includes values from both A and B
 (i.e. each value is either an A or a B)
 e.g.
 1 : symbol | integer
 'foo : symbol | integer

 maybe<T> is equivalent to T | null

 <enum>s here are really just types of symbols,
 see symbol-types.lisp

Conjunctions and disjunctions of concepts are themselves concepts of the same type. They can be named, defined, related to other concepts, and used in restrictions. Conj is implemented with inheritance by an appropriately-typed concept with empty slots, and disj is implemented with lists starting with OR. Feature/map lists can be merged as appropriate.

