- Session types

$$\Delta \vdash T \leq S^{\ell} \quad \stackrel{\text{def}}{=} \quad \Delta \vdash T : K \wedge K \leq S^{\ell}$$

$$\frac{\Delta \vdash T \leq B^{\ell}}{\Delta ; \Gamma \vdash \text{new } T : ((T, \bar{T}) ; \Gamma)}$$

$$\frac{\Delta ; \Gamma_1 \vdash e_1 : (B, \Gamma_2) \quad \Delta ; \Gamma_2 \vdash e_2 : (T_1 ; \Gamma_3) \quad \Delta \vdash T_2 \leq S^{\ell} \quad \Delta \vdash T_1 \sim !B ; T_2}{\Delta ; \Gamma_1 \vdash \text{send } e_1\ e_2 : (B \to T_1 \to T_2 ; \Gamma_3)}$$

$$\frac{\Delta ; \Gamma_1 \vdash e : (T_1 ; \Gamma_2) \quad \Delta \vdash T_1 \sim ?B ; T_2 \quad \Delta \vdash T_2 \leq S^{\ell}}{\Delta ; \Gamma_1 \vdash \text{receive } e : (T_1 \to (B, T_2) ; \Gamma_2)}$$

- fork

$$\frac{\Delta ; \Gamma_1 \vdash e : (T ; \Gamma_2) \quad \Delta \vdash un(T)}{\Delta ; \Gamma_1 \vdash \text{fork } e : (\text{unit} ; \Gamma_2)}$$

- session types (choice)

$$\frac{\Delta;\Gamma_1 \vdash e : (T;\Gamma_2) \quad \Delta \vdash T \sim \&\{C_k : U_k\} \quad \Delta;\Gamma_2, x : U_k \vdash e_k : (T_k;E_k)}{\Delta;\Gamma_1 \vdash \underline{\text{case}}\ e\ \underline{\text{of}}\ C_k\ x \to e_k : (T_1; lub(E_k))}$$

$$\frac{\Delta;\Gamma_1 \vdash e : (T;\Gamma_2) \quad \Delta \vdash T_j \sim \oplus\{C : T_i\}}{\Delta;\Gamma_1 \vdash \underline{\text{select}}\ C\ e : (\oplus\{C : T\} : \Gamma_2)}$$

Notes

- case is overloaded : deconstructor (pattern matching) for datatypes and for session extend choice
- (Datatype ~~and session choice~~) constructors are unique : (see Haskell)

  data D = C | C } not valid
  data D = C }
  data E = C

- data declarations in a program are collected in a map $C \mapsto T$; Σ ConstructorEnv, Constructor : Type

  e.g., data IntMaybe = Nothing | Just Int

  yields Nothing : IntMaybe
  Just : Int → IntMaybe

- binding declarations, $x : T$, in a program are collected in a ~~Type~~ TermVarEnv (Γ)

  TermVar : Type

- Type declarations are collected in a ~~TypeVarEnv~~ ConstructorEnv

  ~~TypeVar : Type~~

- Expression declarations are collected in a

  EmpEnv TermVar : Exp

The initial Type environment Γ_0 contains entries for a) all primitive operators. E.g.,

"+" :: Int → Int → Int (2)

"Not" :: Bool → Bool.

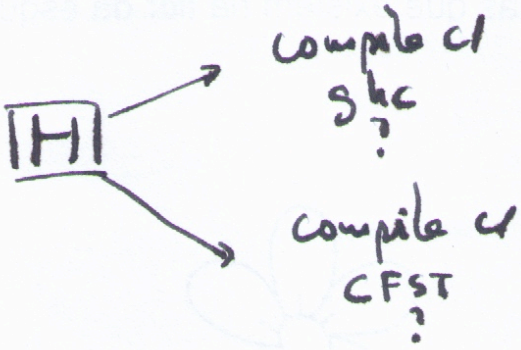
and b) all functions in the Haskell prelude

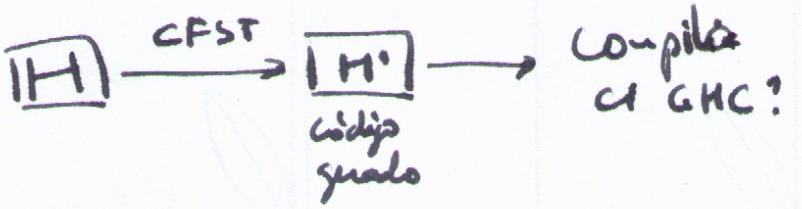
E.g.,

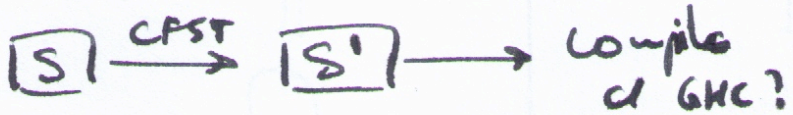
fst :: (a, b) → a (10)

map :: (a → b) → [a] → [b]

TESTS


H
compile c/ GHC ?
compile c/ CFST ?


H
CFST
H'
código gerado
compila c/ GHC?


S
CFST
S'
compila c/ GHC?