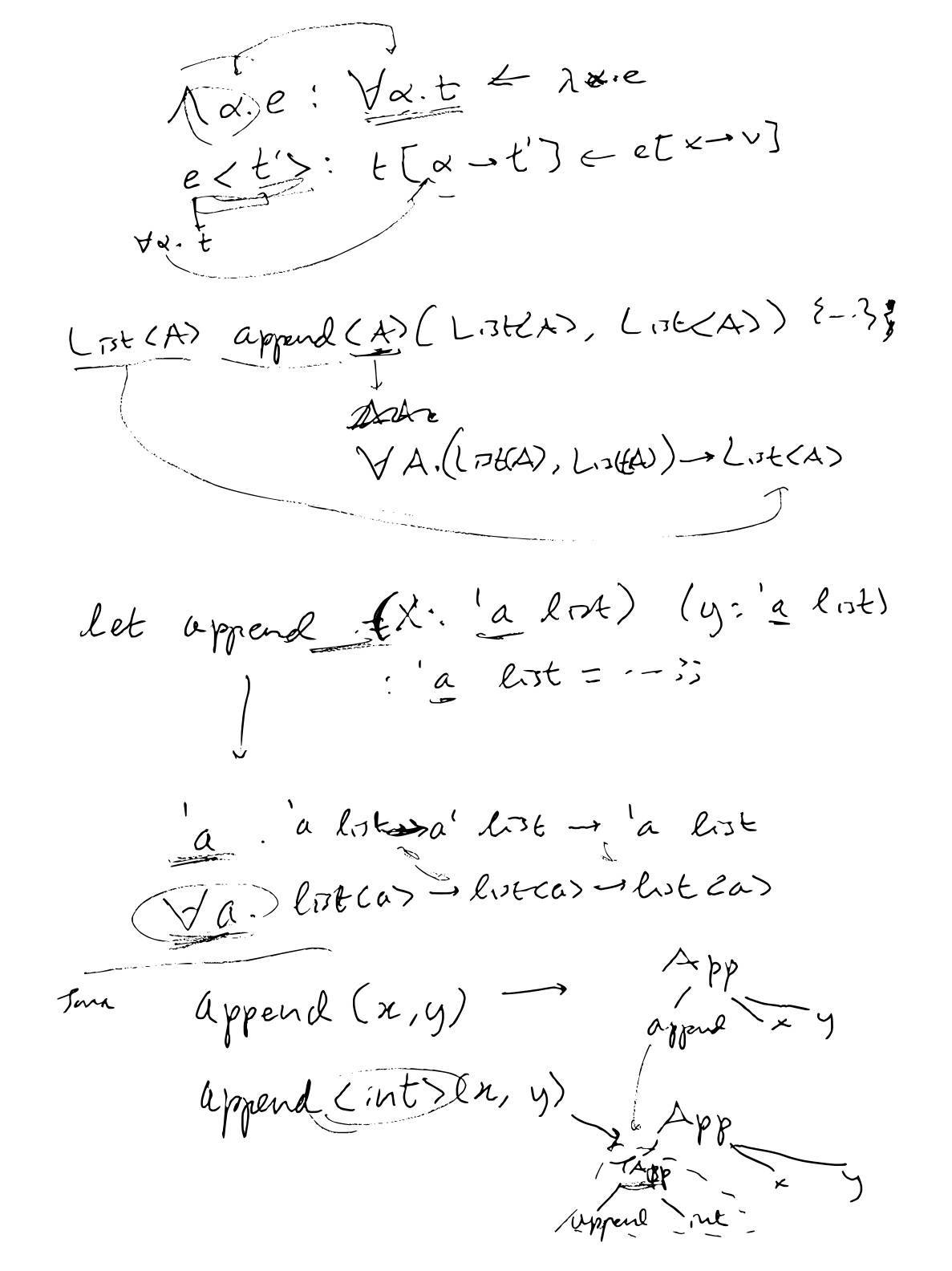

Λα.e : ∀α.t ← λα.e
e<t'> : t[α → t'] ← e[x → v]
∀α. t
List<A> append<A>(List<A>, List<A>) {-};
∀A.(List<A>, List<A>) → List<A>
let append (x: 'a list) (y: 'a list)
: 'a list = --;;
'a . 'a list → 'a list → 'a list
∀a. list<a> → list<a> → list<a>
Term
append (x, y)
append <int> (x, y)
App
append
x
y
App
TApp
x
y
append
int

append : 'a 'a list -> ---

append x : 'b list -> 'b list

int list

'b list

append <'b> x

let append x y = ---

let appnd = Λ'a ↓ λx (λy (. ---

$\Gamma; \Delta \vdash e : \forall \alpha . t$

$\Gamma; \Delta \vdash e\langle t' \rangle : t[\alpha \to t']$

↓ a list of available types

$\lambda x . \; x + 5$

$\lambda x . \; y + 5$

int → int

let f = $\Lambda \alpha . \lambda x : \alpha . \; x$ in f<int>

$\alpha \to \alpha$

e, e'

$\lambda x . e'$

$\forall \alpha . \alpha \to \alpha$

$(\lambda x : \alpha . \; x) \; 3$

$$\frac{\Gamma; \Delta, \alpha \vdash e : t}{\Gamma; \Delta \vdash \Lambda \alpha.\, e : \forall \alpha.\, t}$$

$\lambda x.x : \forall \alpha.\, \alpha \to \alpha$

let f = ---

$\Lambda \alpha.\, \Lambda \alpha_1 \cdots$

$t_1 = [t_2]$

let f x y = match x with

| a :: b → f b (a :: y)

| [] → y

t_2 $[t_2]$ t_3 $t_3 = [t_2]$ t_1

$\forall \alpha.\, [\alpha] \to [\alpha] \to [\alpha]$

$\forall t_2.\, [t_2] \to [t_2] \to [t_2]$

add <T extends Arith> (T a , T b)

append <T> (List<T>, List<T>) → List<T>

$\forall T.\ List\ T \to List\ T \to List\ T$

union {
  int n; -
  student s; -
}

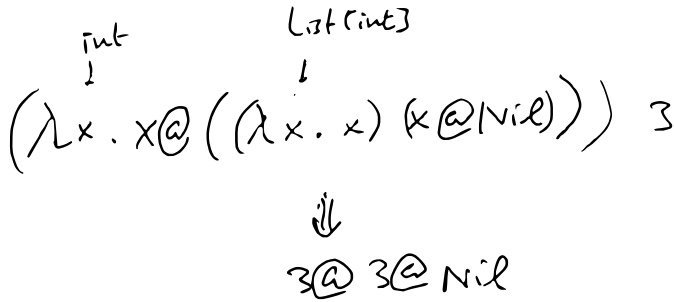
type sorl =
  n of int
| s of student

match -- with
  n n' → [ ]
| s s' → [ ]

list<int> -
list<T> -
T -
A → B

[x : t]

Context ← Env
Constraint


int
List[int]
(λx . x@((λx . x) (x@Nil))) 3
3@ 3@ Nil