Explicitly-typed, polymorphic OCaml

Concrete syntax:

$typeterm \rightarrow$ `int` | `bool` | $typeterm$ `->` $typeterm$ | id

$exp \rightarrow$ int | `true` | `false` | id | exp exp | `fun` id : $typeterm$ `->` exp
| `let` id : $typeterm$ = exp `in` exp | id`[`$typeterm$`]`

Abstract syntax:

```
type typeterm = IntType | BoolType | Typevar of string
              | FunType of typeterm * typeterm

type exp = Int of int | True | False | Var of string
         | Operation of exp * binary_operation * exp
         | App of exp * exp | Fun of string * typeterm * exp
         | Let of exp * typeterm * exp
         | Polyvar of string * typeterm
```

Explicitly-typed, polymorphic OCaml examples

```
let id:(alpha->alpha) = fun x:alpha -> x
in id[int->int] 4

let g:alpha->beta->alpha = fun a:alpha -> fun b:beta -> a
in let id:(gamma->gamma) = fun x:gamma -> x
   in g (id[int->int] 4) (id[bool->bool] true)
       [int -> bool -> int]

let incr:(int->int) = fun i:int -> i+1
in let double:((alpha->alpha)->(alpha->alpha)) =
           fun g:(alpha->alpha) -> fun x:alpha -> g (g x)
   in double[(int->int)->(int->int)] incr 4

let sub:(int->int->int) = fun i:int -> fun j:int -> i-j
in let reverse:((alpha->(beta->gamma))->(beta->(alpha->gamma))) =
      fun f:(alpha->(beta->gamma)) =
               fun x:alpha -> fun y:alpha -> f y x
   in reverse[(int->(int->int))->(int->(int->int))] sub 3 5
```

Explicitly-typed, polymorphic OCaml exercises

```
(let id = fun x -> x in id id) 5
```

$(\text{let } id : \alpha \to \alpha = \text{fun } x : \alpha \to x$
$\quad \text{in } id\,[(int \to int) \to (int \to int)]\ id\,[int \to int]\ )\ 5$

```
let g = fun a -> fun b -> a
in let incr = fun x -> x+1 in g incr (incr 5)
```

$\text{let } g : \alpha \to \beta \to \alpha = \text{fun } a : \alpha \to \text{fun } b : \beta \to a$
$\quad \text{in let } incr : int \to int = \text{fun } x : int \to x$
$\quad\quad \text{in } g\,[(int \to int) \to int \to (int \to int)]\ incr\ (incr\ 5)$

```
let apply = fun g -> fun x -> g x
in apply (fun x -> x) 3
```

$\text{let } apply : ((\alpha \to \beta) \to (\alpha \to \beta)) = \text{fun } g : (\alpha \to \beta) \to \text{fun } x : \alpha \to g\,x$
$\quad \text{in } apply\,[(int \to int) \to (int \to int)]\ (\text{fun } x : int \to x)\ 3$

Explicit polymorphic type system

- Γ is a map from variables to type schemes. τ, τ', τ'' are types.

(Const) $\Gamma \vdash$ **Int i : int**

(Var) $\Gamma \vdash a : \Gamma(a)$
$(\Gamma(a)$ **a type**$)$

(Fun) $\Gamma \vdash$ **fun** a:τ -> $e : \tau \rightarrow \tau'$
$\Gamma[a{:}\tau] \vdash e : \tau'$

(δ) $\Gamma \vdash e \oplus e' : \tau''$
$\Gamma \vdash e : \tau$
$\Gamma \vdash e' : \tau'$

(App) $\Gamma \vdash e\ e' : \tau'$
$\Gamma \vdash e : \tau \rightarrow \tau'$
$\Gamma \vdash e' : \tau$

(True) $\Gamma \vdash$ **true : bool**

(False) $\Gamma \vdash$ **false : bool**

(PolyVar) $\Gamma \vdash a[\tau] : \tau \leq \Gamma(a)$
$(\Gamma(a)$ **a type scheme**$)$

(Let) $\Gamma \vdash$ **let** a:τ = e in $e' : \tau'$
$\Gamma \vdash e : \tau$
$\Gamma[a{:}\mathbf{GEN}_\Gamma(\tau)] \vdash e' : \tau'$

Example

$\emptyset \vdash$
```
let id:(alpha->alpha) = fun x:alpha -> x
in id[int->int] 4
```
$: int$

$\emptyset \vdash fun\ x:\alpha \to x \quad : \alpha \to \alpha$

$\{x:\alpha\} \vdash x : \alpha$

$\{id : \forall\alpha.\ \alpha \to \alpha\} \vdash id[int \to int]\ 4 \quad : int$

$\{id : \forall\alpha.\alpha \to \alpha\} \vdash id[int \to int] : int \to int$

$(int \to int \leq \forall\alpha.\ \alpha \to \alpha)$

$\{id : \forall\alpha.\alpha \to \alpha\} \vdash 4 : int$

Example

$\emptyset \vdash$

```
let g:alpha->beta->alpha = fun a:alpha -> fun b:beta -> a
in let id:(gamma->gamma) = fun x:gamma -> x
   in g[int->bool->int] (id[int->int] 4) (id[bool->bool] true)
```

$: int$

$$\emptyset \vdash \text{fun } a:\alpha \to \text{fun } b:\beta \to a \; : \; \alpha \to \beta \to \alpha$$

$$\{a:\alpha\} \vdash \text{fun } b:\beta \to a \; : \; \beta \to \alpha$$

$$\{a:\alpha, b:\beta\} \vdash a \; : \; \alpha$$

$$\underbrace{\{g : \forall \alpha\beta.\, \alpha \to \beta \to \alpha\}}_{\Gamma_0} \vdash \text{let } id : \ldots \; : \; int$$

$$\Gamma_0 \vdash \text{fun } x:\gamma \to x \; : \; \gamma \to \gamma$$

$$\Gamma_0[x:\gamma] \vdash x \; : \; \gamma$$

$$\underbrace{\Gamma_0[id : \forall\gamma.\, \gamma \to \gamma]}_{\Gamma_1} \vdash g[int \to bool \to int] \cdots \; : \; int$$

(continued...)

$$\Gamma_1 \vdash g[\mathrm{int} \to \mathrm{bool} \to \mathrm{int}]\ (id[\mathrm{int} \to \mathrm{int}]\ 4) : \mathrm{bool} \to \mathrm{int}$$

$$\Gamma_1 \vdash g[\mathrm{int} \to \mathrm{bool} \to \mathrm{int}] : \mathrm{int} \to \mathrm{bool} \to \mathrm{int}$$

$$(\mathrm{int} \to \mathrm{bool} \to \mathrm{int} \leq \forall \alpha, \beta.\ \alpha \to \beta \to \alpha)$$

$$\Gamma_1 \vdash id[\mathrm{int} \to \mathrm{int}]\ 4 : \mathrm{int}$$

$$\Gamma_1 \vdash id[\mathrm{int} \to \mathrm{int}] : \mathrm{int} \to \mathrm{int}$$

$$(\mathrm{int} \to \mathrm{int} \leq \forall \alpha.\ \alpha \to \alpha)$$

$$\Gamma_1 \vdash 4 : \mathrm{int}$$

$$\Gamma_1 \vdash id[\mathrm{bool} \to \mathrm{bool}]\ \mathrm{true} : \mathrm{bool}$$

$$\Gamma_1 \vdash id[\mathrm{bool} \to \mathrm{bool}] : \mathrm{bool} \to \mathrm{bool}$$

$$(\mathrm{bool} \to \mathrm{bool} \leq \forall \alpha.\ \alpha \to \alpha)$$

$$\Gamma_1 \vdash \mathrm{true} : \mathrm{bool}$$

Example

$\emptyset \vdash$
```
let g:(int->beta)->beta = fun f:(int->beta) -> f 0
  in g[(int->bool)->bool] (fun x:int -> x>0)
```
$: \text{bool}$

$\emptyset \vdash \text{fun } f : \text{int} \to \beta \to f\ 0 \ : \ (\text{int} \to \beta) \to \beta$

$\{f : \text{int} \to \beta\} \vdash f\ 0 : \beta$

$\{f : \text{int} \to \beta\} \vdash f : \text{int} \to \beta$

$\{f : \text{int} \to \beta\} \vdash 0 : \text{int}$

$\Gamma_0 = \{g : \forall\beta.\ (\text{int} \to \beta) \to \beta\}$

$\{g : \forall\beta.\ (\text{int} \to \beta) \to \beta\} \vdash g[(\text{int} \to \text{bool}) \to \text{bool}] \ldots : \text{bool}$

$\Gamma_0 \vdash g[(\text{int} \to \text{bool}) \to \text{bool}] : (\text{int} \to \text{bool}) \to \text{bool}$

$((\text{int} \to \text{bool}) \to \text{bool} \leq \forall\beta.\ (\text{int} \to \beta) \to \beta)$

$\Gamma_0 \vdash \text{fun } x : \text{int} \to x > 0 \ : \ \text{int} \to \text{bool}$

$\Gamma_0[x : \text{int}] \vdash x > 0 : \text{bool}$

$\Gamma_0[x : \text{int}] \vdash x : \text{int}$

$\Gamma_0[x : \text{int}] \vdash 0 : \text{int}$

Technicality #1: Generalization

- **Ex: Prove this judgment (where `incr` has type int→int):**

$\emptyset \vdash$
```
let f:(a->a) = fun x:a ->
          let g:((a->b)->b) = fun y:(a->b) -> y x
          in g[(int->int)->int] incr
  in f[bool->bool] true                       : bool
```

This judgment is not provable.