Monomorphic, explicitly-typed OCaml

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

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

Type rules (where Γ is a mapping from variables to types, and each binary operation $\oplus$ is assumed to have a given type $\tau \longrightarrow \tau' \longrightarrow \tau''$):

(Const) $\Gamma \vdash \textbf{Int i} : \textbf{int}$

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

(Fun) $\Gamma \vdash \textbf{fun } a{:}\tau \texttt{ -> } e : \tau \longrightarrow \tau'$
$\quad \Gamma[a{:}\tau] \vdash e : \tau'$

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

(App) $\Gamma \vdash e\ e' : \tau'$
$\quad \Gamma \vdash e : \tau \longrightarrow \tau'$
$\quad \Gamma \vdash e' : \tau$

(True) $\Gamma \vdash \textbf{true} : \textbf{bool}$

(False) $\Gamma \vdash \textbf{false} : \textbf{bool}$

Examples

$\emptyset \vdash$ `fun f:(int->int) -> f 3` : **(int→ int) → int**

$\{f : \text{int} \to \text{int}\} \vdash f\ 3 : \text{int}$

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

$\{f : \text{int} \to \text{int}\} \vdash 3 : \text{int}$

$\emptyset \vdash$ `fun f:(int->int) -> fun x:int -> f (f x)`
: **(int→ int) → (int→ int)**

$\{f : \text{int} \to \text{int}\} \vdash \text{fun } x : \text{int} \to f\ (f\ x) : \text{int} \to \text{int}$

$\Gamma_0 = \boxed{\{f : \text{int} \to \text{int},\ x : \text{int}\}} \vdash f\ (f\ x) : \text{int}$

$\Gamma_0 \vdash f : \text{int} \to \text{int}$

$\Gamma_0 \vdash f\ x : \text{int}$

$\Gamma_0 \vdash f : \text{int} \to \text{int}$

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

Examples of untypable terms

- **These are terms for which no type declarations can be given that would allow the term to be typed:**

$\emptyset \vdash$ `fun f:  ?  -> f f`

Requires that f have type $\alpha \to \beta$ and requires that f have type α.
So f must have type α and $\alpha \to \beta$, which is impossible.

$\emptyset \vdash$ `fun f: ? -> fun g: ?  -> g (f 1) (f true)`

f must have type $int \to \alpha$ f must have type $bool \to \beta$
But $int \to \alpha$ and $bool \to \beta$ cannot be the same type.

$\emptyset \vdash$ `(fun f: ?  -> f f)(fun i:  ?  -> i)`

Example of type inference

$\emptyset \vdash$ `fun f -> 1 + (f 3)`

Annotate: $(\text{fun } f_\alpha \to (1_\beta + (f_\alpha\ 3_\gamma)_\delta)_\epsilon)_\theta$

Generate E: $\beta = int,\ \gamma = int,\ \delta = int,\ \alpha = \gamma \to \delta,\ \theta = \alpha \to \epsilon,\ \epsilon = int$

Solve E

$$\begin{array}{l} \beta = int \\ \gamma = int \\ \delta = int \\ \alpha = \gamma \to \delta \\ \theta = \alpha \to \epsilon \\ \epsilon = int \end{array} \overset{\gamma}{\Rightarrow} \begin{array}{l} \beta = int \\ \gamma = int \\ \delta = int \\ \alpha = int \to \delta \\ \theta = \alpha \to \epsilon \\ \epsilon = int \end{array} \overset{\delta}{\Rightarrow} \begin{array}{l} \beta = int \\ \gamma = int \\ \delta = int \\ \alpha = int \to int \\ \theta = \alpha \to \epsilon \\ \epsilon = int \end{array} \overset{\alpha}{\Rightarrow} \begin{array}{l} \beta = int \\ \gamma = int \\ \delta = int \\ \alpha = int \to int \\ \theta = (int \to int) \to \epsilon \\ \epsilon = int \end{array} \overset{\epsilon}{\Rightarrow} \begin{array}{l} \beta = int \\ \gamma = int \\ \delta = int \\ \alpha = int \to int \\ \theta = (int \to int) \to int \\ \epsilon = int \end{array}$$

Example of type inference

$\emptyset \vdash$ `fun f -> f 3`

Annotate: $(\text{fun } f_\alpha \to (f_\alpha\ 3_\beta)_\gamma)_\delta$

Generate E: $\beta = int,\ \alpha = \beta \to \gamma,\ \delta = \alpha \to \gamma$

Solve E

$$\begin{array}{l} \beta = int \\ \alpha = \beta \to \gamma \\ \delta = \alpha \to \gamma \end{array} \overset{\beta}{\Rightarrow} \begin{array}{l} \beta = int \\ \alpha = int \to \gamma \\ \delta = \alpha \to \gamma \end{array} \overset{\alpha}{\Rightarrow} \begin{array}{l} \beta = int \\ \alpha = int \to \gamma \\ \delta = (int \to \gamma) \to \gamma \end{array}$$

Example of type inference

$\emptyset \vdash$ `fun f -> f (f 3)`

Annotate: $(\text{fun } f_\alpha \to (f_\alpha\ (f_\alpha\ 3_\beta)_\gamma)_\delta)_\epsilon$

Generate E: $\beta = \text{int},\ \epsilon = \alpha \to \delta,\ \alpha = \beta \to \gamma,\ \alpha = \gamma \to \delta$

Solve E

$$\begin{array}{l} \beta = \text{int} \\ \epsilon = \alpha \to \delta \\ \alpha = \beta \to \gamma \\ \alpha = \gamma \to \delta \end{array} \overset{\beta}{\Rightarrow} \begin{array}{l} \beta = \text{int} \\ \epsilon = \alpha \to \delta \\ \alpha = \text{int} \to \gamma \\ \alpha = \gamma \to \delta \end{array} \Rightarrow \begin{array}{l} \beta = \text{int} \\ \epsilon = \alpha \to \delta \\ \alpha = \text{int} \to \text{int} \\ \gamma = \text{int} \\ \delta = \text{int} \end{array} \overset{\delta}{\Rightarrow} \begin{array}{l} \beta = \text{int} \\ \epsilon = \alpha \to \text{int} \\ \alpha = \text{int} \to \text{int} \\ \gamma = \text{int} \\ \delta = \text{int} \end{array} \overset{\alpha}{\Rightarrow} \begin{array}{l} \beta = \text{int} \\ \epsilon = (\text{int} \to \text{int}) \to \text{int} \\ \alpha = \text{int} \to \text{int} \\ \gamma = \text{int} \\ \delta = \text{int} \end{array}$$

unify $\text{int} \to \gamma$
with $\gamma \to \delta$

Example of type inference

$\emptyset \vdash$ `fun f -> fun g -> g (f 1) (f true)`

Annotate: $(\text{fun } f_\alpha \rightarrow (\text{fun } g_\beta \rightarrow ((g_\beta\ (f_\alpha\ 1_\delta)_\epsilon)_\gamma\ (f_\alpha\ \text{true}_\psi)_\varphi)_\pi)_\tau)_\eta$

Generate E (partially): $\alpha = \delta \rightarrow \epsilon,\ \alpha = \psi \rightarrow \varphi,\ \delta = \text{int},\ \psi = \text{bool}, \ldots$

Solve E

$$\begin{array}{l} \alpha = \delta \rightarrow \epsilon \\ \alpha = \psi \rightarrow \varphi \\ \delta = \text{int} \\ \psi = \text{bool} \\ \vdots \end{array} \overset{\delta}{\Rightarrow} \begin{array}{l} \alpha = \text{int} \rightarrow \epsilon \\ \alpha = \psi \rightarrow \varphi \\ \delta = \text{int} \\ \psi = \text{bool} \\ \vdots \end{array} \overset{\psi}{\Rightarrow} \begin{array}{l} \alpha = \text{int} \rightarrow \epsilon \\ \alpha = \text{bool} \rightarrow \varphi \\ \delta = \text{int} \\ \psi = \text{bool} \\ \vdots \end{array} \Rightarrow$$

unify $\text{int} \rightarrow \epsilon$ with $\text{bool} \rightarrow \varphi$ $\Rightarrow$ impossible

Unification examples

- **Will present algorithm for unification next. These are examples to be solved by inspection.**

unify($\alpha \rightarrow \beta$**, int**$\rightarrow \gamma$ **)**

See lecture 22

unify($\alpha \rightarrow$**(int**$\rightarrow \beta$**), (int**$\rightarrow$**int)**$\rightarrow \gamma$ **)**