Mini-ML

Expressions

$M ::= x|true|false|if\, M\, then\, M\, else\, M|\lambda x(M)|M\, M|let\, x = M\, in\, M|nil|M :: M|case\, M\, of\, nil \Rightarrow M|x :: x \Rightarrow M$

Types

$\tau ::= \alpha|bool|\tau \to \tau|\tau list$

$\sigma ::= \forall A(\tau)$ where $A \in \mathcal{P}(TyVar)$

Identify schemes up to α-equivalence

Say $\sigma \succ \tau$ where $\sigma = \forall \alpha_1, \ldots, \alpha_n(\tau')$ if $\tau = \tau'[\tau_1/\alpha_1, \ldots, \tau_n/\alpha_n]$ for some τ_i

Environments Γ are finite functions from variables to type schemes

Rules

$\Gamma \vdash x : \tau$ if $(x : \sigma) \in \Gamma$ and $\sigma \succ \tau$

$\Gamma \vdash B : bool$ if $B \in \{true, false\}$

$$\frac{\Gamma \vdash M_1 : bool \quad \Gamma \vdash M_2 : \tau \quad \Gamma \vdash M_3 : \tau}{\Gamma \vdash if\ M_1\ then\ M_2\ else\ M_3 : \tau}$$

$\Gamma \vdash nil : \tau\, list$

$$\frac{\Gamma \vdash M_1 : \tau \quad \Gamma \vdash M_2 : \tau\ list}{\Gamma \vdash M_1 :: M_2 : \tau\ list}$$

$$\frac{\Gamma \vdash M_1 : \tau_1\ list \quad \Gamma \vdash M_2 : \tau_2 \quad \Gamma, x_1 : \tau_1, x_2 : \tau_1\ list \vdash M_3 : T_2}{\Gamma \vdash case\ M_1\ of\ nil \Rightarrow M_2 | x_1 :: x_2 \Rightarrow M_3 : \tau_2}$$

if $x_1, x_2 \notin dom(\Gamma)$ and $x_1 \neq x_2$

$$\frac{\Gamma, x : \tau_1 \vdash M : \tau_2}{\Gamma \vdash \lambda x(M) : \tau_1 \to \tau_2}$$ if $x \notin dom(\Gamma)$

$$\frac{\Gamma \vdash M_1 : \tau_1 \to \tau_2 \quad \Gamma \vdash M_2 : \tau_1}{\Gamma \vdash M_1\ M_2 : \tau_2}$$

$$\frac{\Gamma \vdash M_1 : \tau \quad \Gamma, x : \forall A(\tau) \vdash M_2 : \tau'}{\Gamma \vdash let\ x = M_1\ in\ M_2 : \tau'}$$ if $x \notin dom(\Gamma)$ and $A = ftv(\tau) - ftv(\Gamma)$

We write $\Gamma \vdash M : \sigma$ if $A = ftv(\tau) - ftv(\Gamma)$, $\sigma = \forall A(\tau)$ and $\Gamma \vdash M : \tau$ is derivable

A closed type scheme $\forall A(\tau)$ is the principal type scheme of a closed expression M if:

- $\vdash M : \forall A(\tau)$
- For any other closed type scheme $\forall A'(\tau')$, if $\vdash M : \forall A'(\tau')$ then $\forall A(\tau) \succ \tau'$

Type Inference

There is an algorithm mgu which determines whether two types $\tau_1$and τ_2 are unifiable and its unifying substitution S so that:

- $S(\tau_1) = S(\tau_2)$
- For all $S' \in Sub$, if $S'(\tau_1) = S'(\tau_2)$ then $S' = TS$ for some $T \in Sub$

Principal solutions to typing problems $\Gamma \vdash M :?$ are pairs (S, σ) such that:

- $S\Gamma \vdash M : \sigma$
- For all (S', σ'), if $S'\Gamma \vdash M : \sigma'$ then there is some $T \in Sub$ so that $TS = S'$ and $T(\sigma) \succ \sigma'$

An algorithm exists to determine the principal type of a given expression. Some clauses are:

- Function abstraction $pt(\Gamma \vdash \lambda x(M) :?)$
  - $let\, \alpha = fresh\, in$
  - $let\, (S, \tau) = pt(\Gamma, x : \alpha \vdash M :?)\, in\, (S, S(\alpha) \to \tau)$
- Function application $pt(\Gamma \vdash M_1\, M_2 :?)$
  - $let\, (S_1, \tau_1) = pt(\Gamma \vdash M_1 :?)\, in$
  - $let\, (S_2, \tau_2) = pt(S_1\Gamma \vdash M_2 :?)\, in$
  - $let\, \alpha = fresh\, in$
  - $let\, S_3 = mgu(S_2\tau_1, \tau_2 \to \alpha)\, in\, (S_3S_2S_1, S_3(\alpha))$
- Conditionals $pt(\Gamma \vdash if\, M_1\, then\, M_2\, else\, M_3 :?)$
  - $let\, (S_1, \tau_1) = pt(\Gamma \vdash M_1 :?)\, in$
  - $let\, S_2 = mgu(\tau_1, bool)\, in$
  - $let\, (S_3, \tau_3) = pt(S_2S_1\Gamma \vdash M_2 :?)\, in$
  - $let\, (S_4, \tau_4) = pt(S_3S_2S_1\Gamma \vdash M_3 :?)\, in$
  - $let\, S_5 = mgu(S_4\tau_3, \tau_4)\, in\, (S_5S_4S_3S_2S_1, S_5\tau_4)$

Midi-ML

Expressions

$M ::= \ldots|()|ref\, M|!M|M := M$

Types

$\tau ::= \ldots|unit|\tau\, ref$

Rules

$\Gamma \vdash () : unit$

$\dfrac{\Gamma \vdash M : \tau}{\Gamma \vdash ref\ M : \tau\ ref}$

$\dfrac{\Gamma \vdash M : \tau\ ref}{\Gamma \vdash !M : \tau}$

$\dfrac{\Gamma \vdash M_1 : \tau\ ref \quad \Gamma \vdash M_2 : \tau}{\Gamma \vdash M_1 := M_2 : unit}$

Unfortunately this is type unsound. To restore type soundness we must modify the let rule:

$\dfrac{\Gamma \vdash M_1 : \tau_1 \quad \Gamma, x : \forall A(\tau_1) \vdash M_2 : \tau_2}{\Gamma \vdash let\ x = M_1\ in\ M_2 : \tau_2}$ if $x \notin dom(\Gamma)$ and $A = \begin{cases} \{\} & \text{if } M_1 \text{ is not a value} \\ ftv(\tau_1) - ftv(\Gamma) & \text{otherwise} \end{cases}$

This effectively forces references to be monomorphic, since references are not considered to be values.

Polymorphic Midi-ML

Types

$\pi ::= \alpha | bool | \pi \to \pi | \pi\, list | \forall \alpha(\pi)$

Now environments Γ are finite functions from variables to π types.

Rules

We replace the original variable typing rule with the following three in this language:

$\Gamma \vdash x : \pi$ if $(x : \pi) \in \Gamma$

$\dfrac{\Gamma \vdash M : \pi}{\Gamma \vdash M : \forall \alpha(\pi)}$if $\alpha \notin ftv(\Gamma)$

$\dfrac{\Gamma \vdash M : \forall \alpha(\pi)}{\Gamma \vdash M : \pi[\pi'/\alpha]}$

In this system, the type checking and typeability problems are equivalent and undecidable.

Polymorphic Lambda Calculus

Expressions

$M ::= x | \lambda x : \tau(M) | M\, M | \Lambda \alpha(M) | M\, \tau$

We now have beta-reduction on types: $(\Lambda \alpha(M))\tau \to M[\tau/\alpha]$

Types

$\tau ::= \alpha | \tau \to \tau | \forall \alpha(\tau)$

Now environments Γ are finite functions from variables to PLC τ types.

Rules

$\Gamma \vdash x : \tau$ if $(x : \tau) \in \Gamma$

$\dfrac{\Gamma, x : \tau_1 \vdash M : \tau_2}{\Gamma \vdash \lambda x : \tau_1(M) : \tau_1 \to \tau_2}$ if $x \in dom(\Gamma)$

$\dfrac{\Gamma \vdash M_1 : \tau_1 \to \tau_2 \quad \Gamma \vdash M_2 : \tau_1}{\Gamma \vdash M_1\ M_2 : \tau_2}$

$\dfrac{\Gamma \vdash M : \tau}{\Gamma \vdash \Lambda \alpha(M) : \forall \alpha(\tau)}$ if $\alpha \notin ftv(\Gamma)$ (important!)

$\dfrac{\Gamma \vdash M : \forall \alpha(\tau_1)}{\Gamma \vdash M\ \tau_2 : \tau_1[\tau_2/\alpha]}$

Now, for every PLC typing problem $\Gamma \vdash M :?$ there is at most one PLC type τ for which $\Gamma \vdash M : \tau$ is provable. An algorithm exists to determine this type, some clauses of which are:

- Function abstraction $typ(\Gamma \vdash \lambda x : \tau_1(M) :?)$
  - $let\, \tau_2 = typ(\Gamma, x : \tau_1 \vdash M :?)\, in\, \tau_1 \to \tau_2$
- Type generalization $typ(\Gamma \vdash \Lambda \alpha(M) :?)$
  - $let\, \tau = typ(\Gamma \vdash M :?)\, in\, \forall \alpha(\tau)$
- Type specialisation $typ(\Gamma \vdash M\, \tau_2 :?)$
  - $let\, \tau = typ(\Gamma \vdash M :?)\, in$
  - $case\, \tau\, of\, \forall \alpha(\tau_1) \to \tau_1[\tau_2/\alpha] | \_ \to FAIL$

Properties

Say $M \to M'$ if M beta-reduces to M' in one step up to alpha-conversion. Call $\to *$ the transitive reflexive closure of this. M is in *beta-normal* form if it contains no redexes.

If $\Gamma \vdash M : \tau$ then:

- Subject reduction: if $M \to M'$ then $\Gamma \vdash M' : \tau$
- Church-Rosser: if $M \to *M_1$ and $M \to *M_2$then there is M' with $M_1 \to *M'$ and $M_2 \to *M'$
- Strong normalization: there is no infinite chain of beta-reductions starting from M
- Hence we can test equality of any terms M, M'

Datatypes

Booleans

$bool \triangleq \forall\alpha(\alpha \to (\alpha \to \alpha))$

$True \triangleq \Lambda\alpha(\lambda x_1 : \alpha, x_2 : \alpha(x_1))$

$False \triangleq \Lambda\alpha(\lambda x_1 : \alpha, x_2 : \alpha(x_2))$

$if \triangleq \Lambda\alpha(\lambda b : bool, x_1 : \alpha, x_2 : \alpha(b\,\alpha\,x_1\,x_2))$

Lists

$\alpha\,list \triangleq \forall\alpha'(\alpha' \to (\alpha \to \alpha' \to \alpha') \to \alpha')$

$Nil \triangleq \Lambda\alpha, \alpha'(\lambda x' : \alpha', f : \alpha \to \alpha' \to \alpha'(x'))$

$Cons \triangleq \Lambda\alpha(\lambda x : \alpha, l : \alpha\,list(\Lambda\alpha'(\lambda x' : \alpha', f : \alpha \to \alpha' \to \alpha'(f\,x\,(l\,\alpha'\,x'\,f)))))$

$iter \triangleq \Lambda\alpha, \alpha'(\lambda x' : \alpha', f : \alpha \to \alpha' \to \alpha'(\lambda l : \alpha\,list(l\,\alpha'\,x'\,f)))$

Curry-Howard

Logic	$\leftrightarrow$	*Type system*
propositions, ϕ	$\leftrightarrow$	types, τ
constructive proofs, p	$\leftrightarrow$	expressions, M
"p is a proof of ϕ"	$\leftrightarrow$	"M is an expression of type τ"
simplification of proofs	$\leftrightarrow$	reduction of expressions

Dependent Types

$$\frac{\Gamma, x : \sigma \vdash M : \sigma'(x)}{\Gamma \vdash \lambda(x : \sigma)(M) : \Pi(x : \sigma)(\sigma'(x))}$$

$$\frac{\Gamma \vdash M : \Pi(x : \sigma)(\sigma'(x)) \quad \Gamma \vdash M' : \sigma}{\Gamma \vdash M\ M' : \sigma'(M')}$$