Schedule

SCHEDULE

w	date	topic	w	date	topic
~~1~~	~~March 2~~	~~introduction I~~	~~8~~	~~May 4~~	~~lazy programming~~
~~2~~	~~March 9~~	~~introduction II~~	~~9~~	~~May 11~~	~~interpreter I~~
~~3~~	~~March 16~~	~~introduction III~~	10	May 18	type system I
~~4~~	~~March 23~~	~~program reasoning~~	11	June 1	interpreter II
~~5~~	~~March 30~~	~~combinatorics~~	12	June 8	type system II
~~6~~	~~April 6~~	~~abstract data types~~	13	June 22	type system III
~~7~~	~~April 27~~	~~symbolic computation~~	14	June 29	exam

1. type check
2. type reconstruction

Type Checking

Type Checking

- types $\tau ::= \alpha \mid \tau \to \tau \mid g(\tau_1, \ldots, \tau_n)$
- typing environments $A ::= \varnothing \mid A, z : \tau$ where $z ::= x \mid c$
- typing judgements $A \vdash e : \tau$
- typing rules

$$\frac{z \in \mathrm{dom}(A)}{A \vdash x : A(z)} \qquad \frac{A, x : \tau_1 \vdash e : \tau_2}{A \vdash (\mathsf{fun}\ x \to e) : \tau_1 \to \tau_2}$$

$$\frac{A \vdash e_1 : \tau_2 \to \tau \quad A \vdash e_2 : \tau_2}{A \vdash e_1 e_2 : \tau} \qquad \frac{A \vdash e_1 : \tau_1 \quad A, x : \tau_1 \vdash e_2 : \tau_2}{A \vdash \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 : \tau}$$

EXERCISES

$A(\mathsf{true}) = \mathsf{bool}$, $A(+) = \mathsf{int} \to \mathsf{int} \to \mathsf{int}$

- $A \vdash (\mathsf{fun}\ x \to x)\ \mathsf{true} : \mathsf{bool}$
- $A \vdash (\mathsf{fun}\ x \to x + x) : \mathsf{int} \to \mathsf{int}$

Type Inference

Type Inference

- typing with unknown variables
- solve by unification

Type Substitutions

type substitution θ is

- function from type variables to types
- $\mathrm{dom}(\theta) = \{\alpha \mid \theta(\alpha) \neq \alpha\}$ is finite

DEFINITIONS

- application to types

$$\alpha\theta = \theta(\alpha)$$
$$(\tau_1 \to \tau_2)\theta = \tau_1\theta \to \tau_2\theta$$
$$g(\tau_1, \ldots, \tau_n)\theta = g(\tau_1\theta, \ldots, \tau_n\theta)$$

- application to type environments

$$(x_1 : \tau_1, \ldots, x_n : \tau_n)\theta = x_1 : \tau_1\theta, \ldots, x_n : \tau_n\theta$$

- composition

$$\theta_1\theta_2 = \{\alpha \mapsto (\alpha\theta_1)\theta_2 \mid \alpha \text{ is type variable}\}$$

Type Inference

PROBLEM

- input: $A \rhd e : \tau$
- output: θ such that $A\theta \vdash e : \tau\theta$

solution θ is principal if every solution is of the form $\theta\theta'$ for some θ'

APPROACH

1. to find τ such that $A \vdash e : \tau$, use fresh variable α to solve $A \rhd e : \alpha$ $\tau = \alpha\theta$ for its principal solution θ
2. $A \rhd e : \tau \quad \rightsquigarrow^* \quad C$
3. principal solution of $A \rhd e : \tau$ = most general unifier of C

Type Inference

- $\mathrm{ftv}(\tau)$ is set of all free type variables in τ

$$\mathrm{ftv}(A) = \bigcup_{z \in \mathrm{dom}(A)} \mathrm{ftv}(A(z))$$

- simplification of type inference problems

$$\begin{array}{lll}
A \rhd z : \tau & \rightsquigarrow \bot & \text{if } z \notin \mathrm{dom}(A) \\
A \rhd z : \tau & \rightsquigarrow A(z) \doteq \tau & \text{if } z \in \mathrm{dom}(A) \\
A \rhd (\text{fun } x \to e) : \tau & \rightsquigarrow \exists\alpha_1, \alpha_2.(A, x : \alpha_1 \rhd e : \alpha_2 \ \wedge \ \tau \doteq \alpha_1 \to \alpha_2) & \\
A \rhd e_1 e_2 : \tau & \rightsquigarrow \exists\alpha.(A \rhd e_1 : \alpha \to \tau \ \wedge \ A \rhd e_2 : \alpha) & \\
A \rhd \text{let } x = e_1 \text{ in } e_2 : \tau & \rightsquigarrow \exists\alpha.(A \rhd e_1 : \alpha \ \wedge \ A, x : \alpha \rhd e_2 : \tau) &
\end{array}$$

where $\alpha_1, \alpha_2, \alpha \notin \mathrm{ftv}(\tau) \cup \mathrm{ftv}(A)$, i.e., fresh variables

EXERCISES rewrite to unification problems

$A(\mathsf{true}) = \mathsf{bool}$, $A(+) = \mathsf{int} \to \mathsf{int} \to \mathsf{int}$

- $A \rhd (\text{fun } x \to x)\ \mathsf{true} : \alpha$
- $A \rhd (\text{fun } x \to x + x) : \alpha$

Unification Algorithm

$$\{\tau_1 \doteq \tau_1', \ldots, \tau_n \doteq \tau_n'\}\theta = \{\tau_1\theta \doteq \tau_1'\theta, \ldots, \tau_n\theta \doteq \tau_n'\theta\}$$

$$\begin{array}{ll}
U(\theta, \varnothing) & = \theta \\
U(\theta, \{\tau \doteq \tau\} \uplus C) & = U(\theta, C) \\
U(\theta, \{\alpha \doteq \tau\} \uplus C) & = U(\theta\{\alpha \mapsto \tau\}, C\{\alpha \mapsto \tau\}) \\
U(\theta, \{\tau \doteq \alpha\} \uplus C) & = U(\theta\{\alpha \mapsto \tau\}, C\{\alpha \mapsto \tau\}) \\
U(\theta, \{g(\tau_1, \ldots, \tau_n) \doteq g(\tau_1', \ldots, \tau_n')\} \uplus C) & = U(\theta, \{\tau_1 \doteq \tau_1', \ldots, \tau_n \doteq \tau_n'\} \cup C) \\
U(\theta, \{\tau \doteq \tau'\} \uplus C) & = \text{raise } (\mathrm{Unify}(\tau, \tau'))
\end{array}$$

where $\alpha \neq \tau$ and τ does not contain α

Properties

- type system : simple types

THEOREM safety

- (progress) if e is typable then $e \longrightarrow e'$ or e is value
- (preservation) if e has type τ and $e \longrightarrow e'$ then e' has type τ

THEOREM strong normalization

simply typed term is always terminating wrt $\longrightarrow_\beta$

Homework

1. implement unification algorithm
2. implement constraint generator
3. implement type inference

PROBLEMS

- fixed point combinator is not typable (why?)
- how to allow polymorphic types?