50.054 Static Semantics Part 2

ISTD, SUTD

Learning Outcomes

1. Apply type checking algorithm to type check a simply typed lambda calculus expression.
2. Apply Hindley Milner algorithm to type check lambda calculus expressions.
3. Apply Algorithm W to infer type for lambda calculus.

Recap

- Now you know how to type check and type inference SIMP programs
- How about Lambda Calculus

Simply Typed Lambda Calculus

We consider an adaptation

$$
\begin{array}{rccl}
\texttt{(Lambda Terms)} & t & ::= & x \mid \lambda x : T.t \mid t\ t \mid let\ x : T =\ t\ in\ t \mid if\ t\ then\ t\ else\ t \mid \\
\texttt{(Builtin Operators)} & op & ::= & + \mid - \mid * \mid / \mid == \\
\texttt{(Builtin Constants)} & c & ::= & 0 \mid 1 \mid ... \mid true \mid false \\
\texttt{(Types)} & T & ::= & int \mid bool \mid T \to T \\
\texttt{(Type Environments)} & \Gamma & \subseteq & (x \times T)
\end{array}
$$

- type annotation is introduced to lambda abstraction and let expression.
- $\to$ type operator is right associative, i.e. $T_1 \to T_2 \to T_3$ is parsed as $T_1 \to (T_2 \to T_3)$.

Type Checking

$$\Gamma \vdash t : T$$

- It's a relation!
- Given a type environment Γ and lambda term t and a type T, check whether t can be given a type T under Γ

Type Checking Rules

$$(\texttt{lctInt}) \quad \frac{c \texttt{ is an integer}}{\Gamma \vdash c : \mathit{int}}$$

$$(\texttt{lctBool}) \quad \frac{c \in \{\mathit{true}, \mathit{false}\}}{\Gamma \vdash c : \mathit{bool}}$$

$$(\texttt{lctVar}) \quad \frac{(x, T) \in \Gamma}{\Gamma \vdash x : T}$$

$$(\texttt{lctLam}) \quad \frac{\Gamma \oplus (x, T) \vdash t : T'}{\Gamma \vdash \lambda x : T.t : T \to T'}$$

$$(\texttt{lctApp}) \quad \frac{\Gamma \vdash t_1 : T_1 \to T_2 \qquad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1 \ t_2 : T_2}$$

$$(\texttt{lctLet}) \quad \frac{\Gamma \vdash t_1 : T_1 \qquad \Gamma \oplus (x, T_1) \vdash t_2 : T_2}{\Gamma \vdash \mathit{let} \ x : T_1 = t_1 \ \mathit{in} \ t_2 : T_2}$$

$$(\texttt{lctIf}) \quad \frac{\Gamma \vdash t_1 : \mathit{bool} \qquad \Gamma \vdash t_2 : T \qquad \Gamma \vdash t_3 : T}{\Gamma \vdash \mathit{if} \ t_1 \ \mathit{then} \ t_2 \ \mathit{else} \ t_3 : T}$$

$$(\texttt{lctOp1}) \quad \frac{\Gamma \vdash t_1 : \mathit{int} \qquad \Gamma \vdash t_2 : \mathit{int} \qquad \mathit{op} \in \{+, -, *, /\}}{\Gamma \vdash t_1 \ \mathit{op} \ t_2 : \mathit{int}}$$

$$(\texttt{lctOp2}) \quad \frac{\Gamma \vdash t_1 : \mathit{int} \qquad \Gamma \vdash t_2 : \mathit{int}}{\Gamma \vdash t_1 \ == \ t_2 : \mathit{bool}}$$

$$(\texttt{lctOp3}) \quad \frac{\Gamma \vdash t_1 : \mathit{bool} \qquad \Gamma \vdash t_2 : \mathit{bool}}{\Gamma \vdash t_1 \ == \ t_2 : \mathit{bool}}$$

$$(\texttt{lctFix}) \quad \frac{\Gamma \vdash t : (T_1 \to T_2) \to T_1 \to T_2}{\Gamma \vdash \mathit{fix} \ t : T_1 \to T_2}$$

Type Checking Example

$$let\ f : int \rightarrow int = (\lambda x : Int.(x + 1))\ inf\ 0$$

We will do it in the Jamboard

Simply Typed Calculus Type Checking Formal Properties

Progress

Let t be a simply typed lambda calculus term such that $fv(t) = \{\}$. Let T be a type such that $\{\} \vdash t : T$. Then t is either a value or there exists some t' such that $t \longrightarrow t'$.

Preservation

Let t and t' be simply typed lambda calculus terms such that $t \longrightarrow t'$. Let T be a type and Γ be a type environment such that $\Gamma \vdash t : T$. Then $\Gamma \vdash t' : T$.

Simply Typed Calculus Type Checking Formal Properties

Uniqueness

Let t be a simply typed lambda calculus term. Let Γ be a type environment such that for all $x \in fv(t)$, $x \in dom(\Gamma)$. Let T and T' be types such that $\Gamma \vdash t : T$ and $\Gamma \vdash t : T'$. Then T and T' must be the same.

Simply Typed Lambda Calculus limitation

- Verbosity
- Polymorphism is not supported

$$
\begin{aligned}
&\mathit{let}\ f = \lambda x : \alpha . x \\
&\mathit{in}\ \mathit{let}\ g = \lambda x : \mathit{int} . \lambda y : \mathit{bool} . x \\
&\quad \mathit{in}\ (g\ (f\ 1)\ (f\ \mathit{true}))
\end{aligned}
$$

Where α denotes some generic type.

Hindley Milner Type System

We re-define the lambda calculus syntax for Hindley Milner Type System as follows

$$
\begin{array}{rcll}
(\texttt{Lambda Terms}) & t & ::= & x \mid \lambda x.t \mid t\ t \mid \mathit{let}\ x =\ t\ \mathit{in}\ t \mid \mathit{if}\ t\ \mathit{then}\ t\ \mathit{else}\ t \mid t\ \mathit{op}\ t \mid c \mid \mathit{fix}\ t \\
(\texttt{Builtin Operators}) & \mathit{op} & ::= & + \mid - \mid * \mid / \mid == \\
(\texttt{Builtin Constants}) & c & ::= & 0 \mid 1 \mid ... \mid \mathit{true} \mid \mathit{false} \\
(\texttt{Types}) & T & ::= & \mathit{int} \mid \mathit{bool} \mid T \rightarrow T \mid \alpha \\
(\texttt{TypeScheme}) & \sigma & ::= & \forall \alpha.\sigma \mid T \\
(\texttt{Type Environments}) & \Gamma & \subseteq & (x \times \sigma) \\
(\texttt{Type Substitution}) & \Psi & ::= & [T/\alpha] \mid [] \mid \Psi \circ \Psi
\end{array}
$$

In the above grammar rules, we remove the type annotations from the lambda abstraction and let binding.

Hindley Milner Type System - Type Checking Mode

(**hmInt**) $$\frac{c \text{ is an integer}}{\Gamma \vdash c : int}$$

(**hmBool**) $$\frac{c \in \{true, false\}}{\Gamma \vdash c : bool}$$

(**hmVar**) $$\frac{(x, \sigma) \in \Gamma}{\Gamma \vdash x : \sigma}$$

(**hmLam**) $$\frac{\Gamma \oplus (x, T) \vdash t : T'}{\Gamma \vdash \lambda x.t : T \to T'}$$

(**hmApp**) $$\frac{\Gamma \vdash t_1 : T_1 \to T_2 \qquad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1 \ t_2 : T_2}$$

(**hmFix**) $$\frac{(fix, \forall\alpha.(\alpha \to \alpha) \to \alpha) \in \Gamma}{\Gamma \vdash fix : \forall\alpha.(\alpha \to \alpha) \to \alpha}$$

$\sigma_1 \sqsubseteq \sigma_2$ iff $\sigma_1 = \forall\alpha.\sigma_1'$ and there exists a type subsitution Ψ such that $\Psi(\sigma_1') = \sigma_2$

(**hmIf**) $$\frac{\Gamma \vdash t_1 : bool \qquad \Gamma \vdash t_2 : \sigma \qquad \Gamma \vdash t_3 : \sigma}{\Gamma \vdash if\ t_1\ \{t_2\}\ else\{t_3\} : \sigma}$$

(**hmOp1**) $$\frac{\Gamma \vdash t_1 : int \qquad \Gamma \vdash t_2 : int \qquad op \in \{+, -, *, /\}}{\Gamma \vdash t_1\ op\ t_2 : int}$$

(**hmOp2**) $$\frac{\Gamma \vdash t_1 : int \qquad \Gamma \vdash t_2 : int}{\Gamma \vdash t_1\ ==\ t_2 : bool}$$

(**hmOp3**) $$\frac{\Gamma \vdash t_1 : bool \qquad \Gamma \vdash t_2 : bool}{\Gamma \vdash t_1\ ==\ t_2 : bool}$$

(**hmLet**) $$\frac{\Gamma \vdash t_1 : \sigma_1 \qquad \Gamma \oplus (x, \sigma_1) \vdash t_2 : T_2}{\Gamma \vdash let\ x = t_1\ in\ t_2 : T_2}$$

(**hmInst**) $$\frac{\Gamma \vdash t : \sigma_1 \qquad \sigma_1 \sqsubseteq \sigma_2}{\Gamma \vdash t : \sigma_2}$$

(**hmGen**) $$\frac{\Gamma \vdash t : \sigma \qquad \alpha \notin ftv(\Gamma)}{\Gamma \vdash t : \forall\alpha.\sigma}$$

Free Type Variables

$$
\begin{array}{rcl}
\mathit{ftv}(\alpha) & = & \{\alpha\} \\
\mathit{ftv}(\mathit{int}) & = & \{\} \\
\mathit{ftv}(\mathit{bool}) & = & \{\} \\
\mathit{ftv}(T_1 \to T_2) & = & \mathit{ftv}(T_1) \cup \mathit{ftv}(T_2) \\
\mathit{ftv}(\forall\alpha.\sigma) & = & \mathit{ftv}(\sigma) - \{\alpha\}
\end{array}
$$

$\mathit{ftv}()$ is also overloaded to extra free type variables from a type environment.

$$
\mathit{ftv}(\Gamma) \;=\; \{\alpha \mid (x, \sigma) \in \Gamma \wedge \alpha \in \mathit{ftv}(\sigma)\}
$$

Type Substitution Application

The application of a type substitution can be defined as

$$
\begin{array}{rcll}
[]\sigma & = & \sigma & \\
[T/\alpha]\mathit{int} & = & \mathit{int} & \\
[T/\alpha]\mathit{bool} & = & \mathit{bool} & \\
[T/\alpha]\alpha & = & T & \\
[T/\alpha]\beta & = & \beta & \beta \neq \alpha \\
[T/\alpha]T_1 \to T_2 & = & ([T/\alpha]T_1) \to ([T/\alpha]T_2) & \\
[T/\alpha]\forall\beta.\sigma & = & \forall\beta.([T/\alpha]\sigma) & \beta \neq \alpha \wedge \beta \notin \mathit{ftv}(T) \\
(\Psi_1 \circ \Psi_2)\sigma & = & \Psi_1(\Psi_2(\sigma)) &
\end{array}
$$

Hindley Milner Type Checking Example

Let $\Gamma = \{\}$.

$$\begin{array}{l} let\ f = \lambda x.x \\ in\ let\ g = \lambda x.\lambda y.x \\ \quad in\ (g\ (f\ 1)\ (f\ true)) \end{array}$$

We will do it in the Jamboard.

Hindley Milner Type System Formal Properties

Progress

Let t be a lambda calculus term such that $fv(t) = \{\}$. Let σ be a type scheme such that $\Gamma_{init} \vdash t : \sigma$. Then t is either a value or there exists some t' such that $t \longrightarrow t'$.

Preservation

Let t and t' be lambda calculus terms such that $t \longrightarrow t'$. Let σ be a type scheme and Γ be a type environment such that $\Gamma \vdash t : \sigma$. Then $\Gamma \vdash t' : \sigma$.

Hindley Milner Type System Formal Properties

Uniqueness

The following property states that if a lambda term is typable, its type scheme must be unique modulo type variable renaming.

Let t be a lambda calculus term. Let Γ be a type environment such that for all $x \in fv(t)$, $x \in dom(\Gamma)$. Let σ and σ' be type schemes such that $\Gamma \vdash t : \sigma$ and $\Gamma \vdash t : \sigma'$. Then σ and σ' must be the same modulo type variable renaming.

Hindley Milner Inference Mode

In theory, we could use the Hindler Milner rules for type inference too, except

- *fix* type must be given in the initial type environment (not a big deal!)
- The (`hmGen`) and (`hmInst`) make the system non-syntax directed, i.e. huge search space.

Algorithm W

The rules are in form of $\Gamma, t \vDash T, \Psi$

Input

- Γ (starting) type environment
- t the lambda expression

Output

- T type
- Ψ a type substitution

Algorithm W

$$(\texttt{wInt})\quad \frac{c \texttt{ is an integer}}{\Gamma, c \vDash int, []}$$

$$(\texttt{wBool})\quad \frac{c \in \{true, false\}}{\Gamma, c \vDash bool, []}$$

$$(\texttt{wVar})\quad \frac{(x, \sigma) \in \Gamma \quad inst(\sigma) = T}{\Gamma, x \vDash T, []}$$

$$(\texttt{wFix})\quad \frac{\begin{array}{c}(fix, \forall\alpha.(\alpha \to \alpha) \to \alpha) \in \Gamma \\ T = inst(\forall\alpha.(\alpha \to \alpha) \to \alpha)\end{array}}{\Gamma, fix \vDash T, []}$$

$$(\texttt{wLam})\quad \frac{\alpha_1 = newvar \quad \Gamma \oplus (x, \alpha_1), t \vDash T, \Psi}{\Gamma, \lambda x.t \vDash: \Psi(\alpha_1 \to T), \Psi}$$

$$(\texttt{wLet})\quad \frac{\begin{array}{c}\Gamma, t_1 \vDash T_1, \Psi_1 \\ \Psi_1(\Gamma) \oplus (x, gen(\Psi_1(\Gamma), T_1)), t_2 \vDash T_2, \Psi_2\end{array}}{\Gamma, let\ x = t_1\ in\ t_2 \vDash T_2, \Psi_2 \circ \Psi_1}$$

$$(\texttt{wApp})\quad \frac{\begin{array}{c}\Gamma, t_1 \vDash T_1, \Psi_1 \quad \Psi_1(\Gamma), t_2 \vDash T_2, \Psi_2 \\ \alpha_3 = newvar \quad \Psi_3 = mgu(\Psi_2(T_1), T_2 \to \alpha_3)\end{array}}{\Gamma, (t_1\ t_2) \vDash \Psi_3(\alpha_3), \Psi_3 \circ \Psi_2 \circ \Psi_1}$$

$$(\texttt{wOp1})\quad \frac{\begin{array}{c}op \in \{+, -, *, /\} \\ \Gamma, t_1 \vDash T_1, \Psi_1 \quad \Psi_1(\Gamma), t_2 \vDash T_2, \Psi_2 \\ mgu(\Psi_2(T_1), T_2, int) = \Psi_3\end{array}}{\Gamma, t_1\ op\ t_2 \vDash int, \Psi_3 \circ \Psi_2 \circ \Psi_1}$$

$$(\texttt{wOp2})\quad \frac{\begin{array}{c}\Gamma, t_1 \vDash T_1, \Psi_1 \quad \Psi_1(\Gamma), t_2 \vDash T_2, \Psi_2 \\ mgu(\Psi_2(T_1), T_2) = \Psi_3\end{array}}{\Gamma, t_1\ ==\ t_2 \vDash bool, \Psi_3 \circ \Psi_2 \circ \Psi_1}$$

$$(\texttt{wIf})\quad \frac{\begin{array}{c}\Gamma, t_1 \vDash T_1, \Psi_1 \quad \Psi_1' = mgu(bool, T_1) \circ \Psi_1 \\ \Psi_1'(\Gamma), t_2 \vDash T_2, \Psi_2 \quad \Psi_1'(\Gamma), t_3 \vDash T_3, \Psi_3 \\ \Psi_4 = mgu(\Psi_3(T_2), \Psi_2(T_3))\end{array}}{\Gamma, if\ t_1\ then\ t_2\ else\ t_3 \vDash \Psi_4(\Psi_3(T_2)), \Psi_4 \circ \Psi_3 \circ \Psi_2 \circ \Psi_1'}$$

Algorithm W - Auxiliary functions

$$\Psi(\Gamma) \quad = \quad \{(x, \Psi(\sigma)) \mid (x, \sigma) \in \Gamma\}$$

$$\begin{array}{rcl} \mathit{inst}(T) & = & T \\ \mathit{inst}(\forall\alpha.\sigma) & = & [\beta_1/\alpha](\mathit{inst}(\sigma)) \ \mathit{where}\ \beta_1 = \mathit{newvar} \end{array}$$

$$\mathit{gen}(\Gamma, T) \quad = \quad \forall\overline{\alpha}.T \quad \mathit{where}\ \overline{\alpha} = \mathit{ftv}(T) - \mathit{ftv}(\Gamma)$$

Algorithm W - Unification

$$
\begin{array}{rcl}
mgu(\alpha, T) & = & [T/\alpha] \\
mgu(T, \alpha) & = & [T/\alpha] \\
mgu(int, int) & = & [] \\
mgu(bool, bool) & = & [] \\
mgu(T_1 \to T_2, T_3 \to T_4) & = & let\ \Psi_1 = mgu(T_1, T_3) \\
 & & in\ \ let\ \Psi_2 = mgu(\Psi_1(T_2), \Psi_1(T_4)) \\
 & & \qquad in\ \Psi_2 \circ \Psi_1
\end{array}
$$

Algorithm W - Examples

1. $\lambda x.x$
2. $\lambda x.\lambda y.x$
3. $let\ f = \lambda x.x\ in\ (let\ g = \lambda x.\lambda y.x\ in\ g\ (f\ 1)\ (f\ true))$

Will do it in Jamboard.

Summary

- Simply Typed Lambda Calculus
- Hindley Milner Type System
- Algorithm W