Type Inference

CS 152 (Spring 2021)

Harvard University

Tuesday, March 23, 2021

Today, we will learn about

- Type inference
  - Type-checking vs type-inference
  - Constraint-based typing
  - Unification

Type annotations

Type inference

- Infer (or reconstruct) the types of a program
- Example: $\lambda a.\, \lambda b.\, \lambda c.\, \text{if } a\ (b+1) \text{ then } b \text{ else } c$

Constraint-based Type Inference

- *Type variables* $X, Y, Z, \ldots$: placeholders for types.
- Judgment $\Gamma \vdash e : \tau \triangleright C$
  - Expression e has type τ provided every constraint in set C is satisfied
  - Constraints are of the form $\tau_1 \equiv \tau_2$

Language

$$
\begin{aligned}
e &::= x \mid \lambda x\!:\!\tau.\, e \mid e_1\ e_2 \mid n \mid e_1 + e_2 \\
\tau &::= \textbf{int} \mid X \mid \tau_1 \to \tau_2
\end{aligned}
$$

Inference rules

$$\textsc{CT-Var}\ \frac{}{\Gamma \vdash x : \tau \triangleright \emptyset}\ x : \tau \in \Gamma$$

$$\textsc{CT-Int}\ \frac{}{\Gamma \vdash n : \textbf{int} \triangleright \emptyset}$$

$$\textsc{CT-Add}\ \frac{\Gamma \vdash e_1 : \tau_1 \triangleright C_1 \qquad \Gamma \vdash e_2 : \tau_2 \triangleright C_2}{\Gamma \vdash e_1 + e_2 : \textbf{int} \triangleright C_1 \cup C_2 \cup \{\tau_1 \equiv \textbf{int}, \tau_2 \equiv \textbf{int}\}}$$

Inference rules, ctd.

$$\textsc{CT-Abs}\ \frac{\Gamma, x : \tau_1 \vdash e : \tau_2 \triangleright C}{\Gamma \vdash \lambda x : \tau_1 .\, e : \tau_1 \to \tau_2 \triangleright C}$$

$$\textsc{CT-App}\ \frac{\begin{array}{c}\Gamma \vdash e_1 : \tau_1 \triangleright C_1 \\ \Gamma \vdash e_2 : \tau_2 \triangleright C_2 \\ C' = C_1 \cup C_2 \cup \{\tau_1 \equiv \tau_2 \to X\}\end{array}}{\Gamma \vdash e_1\ e_2 : X \triangleright C'}\quad X \text{ is fresh}$$

Example

Unification

- What does it mean for a set of constraints to be satisfied?
- How do we find a solution to a set of constraints (i.e., infer the types)?
- To answer these questions: we define *type substitutions* and *unification*

Type subsitutions (aka substitutions)

- Map from type variables to types
- Substitution of type variables, formally:

$$\sigma(X) = \begin{cases} \tau & \text{if } X \mapsto \tau \in \sigma \\ X & \text{if } X \text{ not in the domain of } \sigma \end{cases}$$

$$\sigma(\textbf{int}) = \textbf{int}$$

$$\sigma(\tau \to \tau') = \sigma(\tau) \to \sigma(\tau')$$

Substitution in constraints

- Extended to substitution of constraints, and set of constrains:

$$\begin{aligned} \sigma(\tau_1 \equiv \tau_2) &= \sigma(\tau_1) \equiv \sigma(\tau_2) \\ \sigma(C) &= \{\sigma(c) \mid c \in C\} \end{aligned}$$

Unification

- Constraints are of form $\tau_1 \equiv \tau_2$
- Substitution σ *unifies* $\tau_1 \equiv \tau_2$ if $\sigma(\tau_1)$ is the same as $\sigma(\tau_2)$
- Substitution σ *unifies* (or *satisfies*) set of constraints C if it unifies every constraint in C
- So given $\vdash e : \tau \triangleright C$, want substitution σ that unifies C
  - Moreover, type of e is $\sigma(\tau)$

Unification algorithm

$$unify(C) = \sigma$$

$unify(\emptyset)$

[]

$unify(\{\tau \equiv \tau'\} \cup C)$

if $\tau = \tau'$ then

$\quad unify(C)$

else if $\tau = X$ and X not a free variable of τ' then

$\quad$ let $\sigma = [X \mapsto \tau']$ in

$\quad unify(\sigma(C)) \circ \sigma$

else if $\tau' = X$ and X not a free variable of τ then

$\quad$ let $\sigma = [X \mapsto \tau]$ in

$\quad unify(\sigma(C)) \circ \sigma$

else if $\tau = \tau_o \to \tau_1$ and $\tau' = \tau'_o \to \tau'_1$ then

$\quad unify(C \cup \{\tau_0 \equiv \tau'_0, \tau_1 \equiv \tau'_1\})$

else *fail*