CSCI-344
Programming Language Concepts (Section 3)

Lecture 26
Type Inference Rules
Instructor: Hossein Hojjat

November 11, 2016

Where we are

Done:

- Formalize Type Inference: Substitution, Unification

This session:

- Type Inference Rules

Solving Constraints

- Grammar for constraints:

$$C^{\sim} ::= \tau_1 \sim \tau_2 \mid C_1^{\sim} \wedge C_2^{\sim} \mid \boldsymbol{T}$$

- $\tau_1 \sim \tau_2$: types τ_1 and τ_2 are equal

- We solve a constraint C by finding a substitution θ such that the constraint θC is satisfied

$$\frac{\tau_1 = \tau_2}{\tau_1 \sim \tau_2 \text{ is satisfied}} \qquad \frac{C_1^{\sim} \text{ is satisfied} \quad C_2^{\sim} \text{ is satisfied}}{C_1^{\sim} \wedge C_2^{\sim} \text{ is satisfied}} \qquad \frac{}{\boldsymbol{T} \text{ is satisfied}}$$

- Substitutions distribute over constraints

$$\theta(\tau_1 \sim \tau_2) = \theta\tau_1 \sim \theta\tau_2$$

$$\theta(C_1^{\sim} \wedge C_2^{\sim}) = \theta C_1^{\sim} \wedge \theta C_2^{\sim}$$

$$\theta(\boldsymbol{T}) = \boldsymbol{T}$$

Solving Constraints

Which of these have solutions?

$$\texttt{int} \sim \texttt{bool}$$

$$\texttt{int list} \sim \texttt{bool list}$$

$$'a \sim \texttt{int}$$

$$'a \sim \texttt{int list}$$

$$'a \sim \texttt{int} \rightarrow \texttt{int}$$

$$'a \sim' a$$

$$'a \texttt{ list} \sim' b \rightarrow' b$$

$$'a \times \texttt{int} \sim \texttt{bool} \times' b$$

$$'a \times \texttt{int} \sim \texttt{bool} \rightarrow' b$$

$$'a \sim' b \texttt{ list} \wedge' b \sim' a \texttt{ list}$$

$$'a \sim ('a, \texttt{int})$$

$$'a \sim \tau \qquad \text{(arbitrary tau)}$$

Solving Constraints

Which of these have solutions?

$$\texttt{int} \sim \texttt{bool}$$

$$\texttt{int list} \sim \texttt{bool list}$$

$$'a \sim \texttt{int}$$

$$'a \sim \texttt{int list}$$

$$'a \sim \texttt{int} \rightarrow \texttt{int}$$

$$'a \sim' a$$

$$'a\ \texttt{list} \sim' b \rightarrow' b$$

$$'a \times \texttt{int} \sim \texttt{bool} \times' b$$

$$'a \times \texttt{int} \sim \texttt{bool} \rightarrow' b$$

only if we allow infinite solutions

$$'a \sim' b\ \texttt{list} \wedge' b \sim' a\ \texttt{list}$$

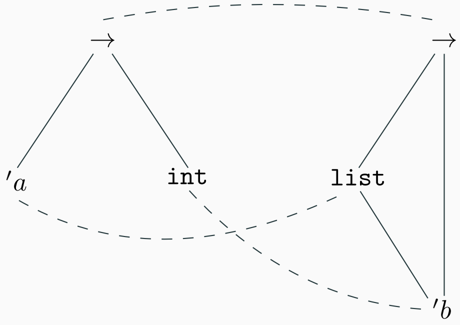
$$'a \sim ('a, \texttt{int})$$

$$'a \sim \tau \qquad \text{(arbitrary tau)}$$

Unification Algorithm

- Walk down both graphs in parallel until either:
    1. hit a contradiction (different type constructors)
    2. hit a type variable: the variable must be equal to corresponding subtree

Example: Unify $'a \to$ `int` and b `list` $\to' b$



From type rules to type inference

Judgment: $C^{\sim}, \Gamma \vdash e : \tau$

Assuming constraint $C^{\sim}$ is satisfied, in environment Γ term e has type τ

IF Rule

Type rule for conditional expression:

$$\frac{\Gamma \vdash e_1 : \texttt{bool} \qquad \Gamma \vdash e_2 : \tau \qquad \Gamma \vdash e_3 : \tau}{\Gamma \vdash \mathrm{IF}(e_1, e_2, e_3) : \tau}\ (\mathrm{IF})$$

becomes

$$\frac{C_1^{\sim}, \Gamma \vdash e_1 : \tau_1 \qquad C_2^{\sim}, \Gamma \vdash e_2 : \tau_2 \qquad C_3^{\sim}, \Gamma \vdash e_3 : \tau_3}{C_1^{\sim} \wedge C_2^{\sim} \wedge C_3^{\sim} \wedge\ \tau_1 \sim \texttt{bool}\ \wedge\ \tau_2 \sim \tau_3, \Gamma \vdash \mathrm{IF}(e_1, e_2, e_3) : \tau_2}\ (\mathrm{IF})$$

Bookkeeping

Abbreviation for n separate judgments

$$\frac{C_1^{\sim}, \Gamma \vdash e_1 : \tau_1 \quad \cdots \quad C_n^{\sim}, \Gamma \vdash e_n : \tau_n}{C_1^{\sim} \wedge \cdots \wedge C_n^{\sim}, \Gamma \vdash e_1, \cdots, e_n : \tau_1, \cdots, \tau_n} \text{ (TYPESOF)}$$

Simplification of IF rule:

$$\frac{C^{\sim}, \Gamma \vdash e_1, e_2, e_3 : \tau_1, \tau_2, \tau_3}{C^{\sim} \wedge \ \tau_1 \sim \texttt{bool} \ \wedge \ \tau_2 \sim \tau_3, \Gamma \vdash \mathrm{IF}(e_1, e_2, e_3) : \tau_2} \text{ (IF)}$$

Function Application

$$\frac{C^{\sim}, \Gamma \vdash e, e_1, \cdots, e_n : \hat{\tau}, \tau_1, \cdots, \tau_n \qquad \alpha \text{ is fresh}}{C^{\sim} \wedge\ \hat{\tau} \sim \tau_1 \times \cdots \times \tau_n \to \alpha, \Gamma \vdash \mathrm{APPLY}(e, e_1, \cdots, e_n) : \alpha} \ (\mathrm{APPLY})$$

Function Application

$$\frac{C^{\sim}, \Gamma \vdash e, e_1, \cdots, e_n : \hat{\tau}, \tau_1, \cdots, \tau_n \qquad \alpha \text{ is fresh}}{C^{\sim} \wedge\ \hat{\tau} \sim \tau_1 \times \cdots \times \tau_n \rightarrow \alpha, \Gamma \vdash \mathrm{APPLY}(e, e_1, \cdots, e_n) : \alpha} \ (\mathrm{APPLY})$$

Example: Infer a type for `(+ 2 5)`:

$$\frac{\overline{\boldsymbol{T}, \Gamma \vdash \texttt{+}: \texttt{int} \times \texttt{int} \rightarrow \texttt{int}} \quad \overline{\boldsymbol{T}, \Gamma \vdash \texttt{2}: \texttt{int}} \quad \overline{\boldsymbol{T}, \Gamma \vdash \texttt{5}: \texttt{int}}}{\boldsymbol{T} \wedge \boldsymbol{T} \wedge \boldsymbol{T} \wedge \texttt{int} \times \texttt{int} \rightarrow \texttt{int} \sim \texttt{int} \times \texttt{int} \rightarrow \alpha_1, \Gamma \vdash \texttt{(+ 2 5)}}$$

Substitute `int` for α_1:

$$\frac{\overline{\boldsymbol{T}, \Gamma \vdash \texttt{+}: \texttt{int} \times \texttt{int} \rightarrow \texttt{int}} \quad \overline{\boldsymbol{T}, \Gamma \vdash \texttt{2}: \texttt{int}} \quad \overline{\boldsymbol{T}, \Gamma \vdash \texttt{5}: \texttt{int}}}{\boldsymbol{T} \wedge \boldsymbol{T} \wedge \boldsymbol{T} \wedge \texttt{int} \times \texttt{int} \rightarrow \texttt{int} \sim \texttt{int} \times \texttt{int} \rightarrow \texttt{int}, \Gamma \vdash \texttt{(+ 2 5)}}$$

From Type Scheme to Types

- Non-deterministic rule

$$\frac{\Gamma(x) = \delta \qquad \tau <: \delta}{\Gamma \vdash x : \tau}\ (\mathrm{VAR})$$

- Deterministic rule:
  VAR rule instantiates type schema with fresh and distinct type variables

$$\frac{\begin{array}{c}\Gamma(x) = \forall \alpha_1, \cdots \alpha_n.\ \tau \\ \alpha'_1, \cdots, \alpha'_n \text{ are fresh and distinct}\end{array}}{\boldsymbol{T}, \Gamma \vdash x : ((\alpha_1 \mapsto \alpha'_1) \circ \cdots \circ (\alpha_n \mapsto \alpha'_n))\, \tau}\ (\mathrm{VAR})$$

From Types to Type Scheme

```
-> (val fst (lambda (x y) x))
fst : (forall ('a 'b) ('a * 'b -> 'a))
```

- First derive:

$$\boldsymbol{T}, \varnothing \vdash \texttt{(lambda (x y) x)} : \alpha \times \beta \to \alpha$$

- Abstract over free type vars and add to environment

$$\texttt{fst} : \forall \alpha, \beta .\ \alpha \times \beta \to \alpha$$

Generalize Function

- Useful tool for calculating quantification set

$$\texttt{generalize}(\tau, A) = \forall \alpha_1, \cdots, \alpha_n.\ \tau$$

- where

$$\{\alpha_1, \cdots, \alpha_n\} = \mathit{ftv}(\tau) - A$$

- **Example:**

$$\texttt{generalize}(\alpha \times \beta \to \alpha, \varnothing) = \forall \alpha, \beta . \alpha \times \beta \to \alpha$$

Summary

This Lecture

- Type Inference

Next Lecture

- Control operators and reduction semantics