

# Type Inference

## Lecture 3a

# Review of last class ...

---

- Programming language

$$e = x \mid \lambda x : \tau . e \mid e \ e \mid i \mid e + e \mid \text{if } e \ e \ e$$

- Semantics  $(\lambda x : \tau . e) \ e' \ , \ \beta \ e[x * e']$

- Type language  $\tau = \alpha \mid \tau \rightarrow \tau \mid \text{int}$

- Some programs have types  $A \vdash e : \tau$ 
  - (according to type rules)

- **Soundness Theorem:** Evaluation preserves types
  - If  $A \vdash e : \tau$  and  $e \ , \ \beta \ d$ , then  $A \vdash d : \tau$

- Untypable program:  $3 + (\lambda x : \tau . e)$

# Challenge Problem

---

- Consider the program
  - `if 1 (3 + 2) (4 3)`
- What is its execution behavior?
- What is its type?

# Type Inference

---

- The *type erasure* of  $e$  is  $e$  with all type information removed
  - the untyped term
- Is an untyped term the erasure of some simply typed term?
  - And what are the types?
- This is a *type inference* problem.
  - We must infer, rather than check, the types.

# Outline

---

- We develop the inference algorithm in steps:
  - recast the type rules in an equivalent form
  - show typing in the new rules reduces to a constraint satisfaction problem
  - show the constraint problem is solvable
    - In this case, via term unification.
- We will use this outline again.

# The Problems

---

$$\begin{array}{c}
 \frac{A(x) = \tau}{A \vdash x : \tau} \quad \frac{A, x : \tau \vdash e : \tau'}{A \vdash \lambda x : \tau. e : \tau \rightarrow \tau'} \quad \frac{A \vdash e_1 : \tau \rightarrow \tau' \quad A \vdash e_2 : \tau}{A \vdash e_1 e_2 : \tau'} \\
 \\
 \frac{}{A \vdash i : \text{int}} \quad \frac{A \vdash e_1 : \text{int} \quad A \vdash e_2 : \text{int}}{A \vdash e_1 + e_2 : \text{int}} \quad \frac{A \vdash e_1 : \text{int} \quad A \vdash e_2 : \tau \quad A \vdash e_3 : \tau}{A \vdash \text{if } e_1 e_2 e_3 : \tau}
 \end{array}$$

- There are three problems in developing an algorithm
  - How do we construct the right type assumptions?
  - How do we ensure types match in applications?
  - How do we ensure types match in if-then-else?

# New Rules

---

$$\begin{array}{c}
 \frac{A(x) = \alpha_x}{A \vdash x : \alpha_x} \quad \frac{A, x : \alpha_x \vdash e : \tau}{A \vdash \lambda x. e : \alpha_x \rightarrow \tau} \quad \frac{A \vdash e_1 : \tau_1 \quad A \vdash e_2 : \tau_2 \quad \tau_1 = \tau_2 \rightarrow \beta}{A \vdash e_1 e_2 : \beta} \\
 \\
 \frac{}{A \vdash i : \text{int}} \quad \frac{A \vdash e_1 : \tau_1 \quad A \vdash e_2 : \tau_2 \quad \tau_1 = \tau_2 = \text{int}}{A \vdash e_1 + e_2 : \text{int}} \quad \frac{A \vdash e_1 : \tau_1 \quad A \vdash e_2 : \tau_2 \quad A \vdash e_3 : \tau_3 \quad \tau_1 = \text{int} \quad \tau_2 = \tau_3}{A \vdash \text{if } e_1 e_2 e_3 : \tau_2}
 \end{array}$$

- Sidestep the problems by introducing explicit unknowns and constraints

# New Rules

---

$$\frac{A(x) = \alpha_x}{A \vdash x : \alpha_x} \quad \frac{A, x : \alpha_x \vdash e : \tau}{A \vdash \lambda x. e : \alpha_x \rightarrow \tau} \quad \frac{A \vdash e_1 : \tau_1 \quad A \vdash e_2 : \tau_2 \quad \tau_1 = \tau_2 \rightarrow \beta}{A \vdash e_1 e_2 : \beta}$$

$$\frac{}{A \vdash i : \text{int}} \quad \frac{A \vdash e_1 : \tau_1 \quad A \vdash e_2 : \tau_2 \quad \tau_1 = \tau_2 = \text{int}}{A \vdash e_1 + e_2 : \text{int}} \quad \frac{A \vdash e_1 : \tau_1 \quad A \vdash e_2 : \tau_2 \quad A \vdash e_3 : \tau_3 \quad \tau_1 = \text{int} \quad \tau_2 = \tau_3}{A \vdash \text{if } e_1 e_2 e_3 : \tau_2}$$

- Type assumption for variable  $x$  is a fresh variable  $\alpha_x$

# New Rules

$$\frac{A(x) = \alpha_x}{A \vdash x : \alpha_x} \quad \frac{A, x : \alpha_x \vdash e : \tau}{A \vdash \lambda x. e : \alpha_x \rightarrow \tau} \quad \frac{A \vdash e_1 : \tau_1 \quad A \vdash e_2 : \tau_2 \quad \tau_1 = \tau_2 \rightarrow \beta}{A \vdash e_1 e_2 : \beta}$$

$$\frac{}{A \vdash i : \text{int}} \quad \frac{A \vdash e_1 : \tau_1 \quad A \vdash e_2 : \tau_2 \quad \tau_1 = \tau_2 = \text{int}}{A \vdash e_1 + e_2 : \text{int}} \quad \frac{A \vdash e_1 : \tau_1 \quad A \vdash e_2 : \tau_2 \quad A \vdash e_3 : \tau_3 \quad \tau_1 = \text{int} \quad \tau_2 = \tau_3}{A \vdash \text{if } e_1 e_2 e_3 : \tau_2}$$

- Equality conditions represented as side constraints

# New Rules

---

$$\frac{A(x) = \alpha_x}{A \vdash x : \alpha_x} \quad \frac{A, x : \alpha_x \vdash e : \tau}{A \vdash \lambda x. e : \alpha_x \rightarrow \tau} \quad \frac{A \vdash e_1 : \tau_1 \quad A \vdash e_2 : \tau_2 \quad \tau_1 = \tau_2 \rightarrow \beta}{A \vdash e_1 e_2 : \beta}$$

$$\frac{A \vdash i : \text{int}}{A \vdash i : \text{int}} \quad \frac{A \vdash e_1 : \tau_1 \quad A \vdash e_2 : \tau_2 \quad \tau_1 = \tau_2 = \text{int}}{A \vdash e_1 + e_2 : \text{int}} \quad \frac{A \vdash e_1 : \tau_1 \quad A \vdash e_2 : \tau_2 \quad A \vdash e_3 : \tau_3 \quad \tau_1 = \text{int} \quad \tau_2 = \tau_3}{A \vdash \text{if } e_1 e_2 e_3 : \tau_2}$$

- Hypotheses are all arbitrary
  - Can always complete a derivation, pending constraint resolution

# Notes

---

- The introduction of unknowns and constraints works only because the shape of the proof is already known.
  - This tells us where to put the constraints and unknowns.
- The revised rules are trivial to implement, except for handling the constraints.

# Solutions of Constraints

---

- The new rules generate a system of type equations.
- Intuitively, a solution of these equations gives a derivation.
- A solution is a substitution  $\text{Vars}$  ,  $\text{Types}$  such that the equations are satisfied.

# Example

---

$$\alpha = \beta \rightarrow \gamma$$

$$\alpha = \gamma \rightarrow \beta$$

$$\beta = \text{int}$$

- A solution is

$$\alpha = \text{int} \rightarrow \text{int}, \beta = \text{int}, \gamma = \text{int}$$

# Solving Type Equations

---

- Term equations are a unification problem.
  - Solvable in near-linear time using a union-find based algorithm.
- No solutions  $\alpha = T[\alpha]$  are permitted
  - The *occurs check*.
  - The check is omitted if we allow infinite types.

# Unification

---

- Close constraints under four rules.
- If no inconsistency or occurs check violation found, system has a solution.
  - $\text{int} = x, y$

$$\mathcal{S} \cup \{\alpha = \alpha\} \Rightarrow \mathcal{S}$$

$$\mathcal{S} \cup \{\alpha = \tau\} \Rightarrow \mathcal{S}[\tau / \alpha] \cup \{\alpha \cong \tau\}$$

$$\mathcal{S} \cup \{\tau_1 \rightarrow \tau_2 = \tau_3 \rightarrow \tau_4\} \Rightarrow \mathcal{S} \cup \{\tau_1 = \tau_3, \tau_2 = \tau_4\}$$

$$\mathcal{S} \cup \{\text{int} = \text{int}\} \Rightarrow \mathcal{S}$$

# Syntax

---

- We distinguish *solved* equations  $\alpha \cong \tau$
- Each rule manipulates only unsolved equations.

$$\mathcal{S} \cup \{\alpha = \alpha\} \Rightarrow \mathcal{S}$$

$$\mathcal{S} \cup \{\alpha = \tau\} \Rightarrow \mathcal{S}[\tau / \alpha] \cup \{\alpha \cong \tau\}$$

$$\mathcal{S} \cup \{\tau_1 \rightarrow \tau_2 = \tau_3 \rightarrow \tau_4\} \Rightarrow \mathcal{S} \cup \{\tau_1 = \tau_3, \tau_2 = \tau_4\}$$

$$\mathcal{S} \cup \{\text{int} = \text{int}\} \Rightarrow \mathcal{S}$$

# Rules 1 and 4

---

- Rules 1 and 4 eliminate trivial constraints.
- Rule 1 is applied in preference to rule 2
  - the only such possible conflict

$$\mathcal{S} \cup \{\alpha = \alpha\} \Rightarrow \mathcal{S}$$

$$\mathcal{S} \cup \{\alpha = \tau\} \Rightarrow \mathcal{S}[\tau / \alpha] \cup \{\alpha \cong \tau\}$$

$$\mathcal{S} \cup \{\tau_1 \rightarrow \tau_2 = \tau_3 \rightarrow \tau_4\} \Rightarrow \mathcal{S} \cup \{\tau_1 = \tau_3, \tau_2 = \tau_4\}$$

$$\mathcal{S} \cup \{\text{int} = \text{int}\} \Rightarrow \mathcal{S}$$

## Rule 2

---

- Rule 2 eliminates a variable from all equations but one (which is marked as solved).
  - Note the variable is eliminated from all unsolved as well as solved equations

$$\mathcal{S} \cup \{\alpha = \alpha\} \Rightarrow \mathcal{S}$$

$$\mathcal{S} \cup \{\alpha = \tau\} \Rightarrow \mathcal{S}[\tau / \alpha] \cup \{\alpha \cong \tau\}$$

$$\mathcal{S} \cup \{\tau_1 \rightarrow \tau_2 = \tau_3 \rightarrow \tau_4\} \Rightarrow \mathcal{S} \cup \{\tau_1 = \tau_3, \tau_2 = \tau_4\}$$

$$\mathcal{S} \cup \{\text{int} = \text{int}\} \Rightarrow \mathcal{S}$$

# Rule 3

---

- Rule 3 applies structural equality to non-trivial terms.

$$\mathcal{S} \cup \{\alpha = \alpha\} \Rightarrow \mathcal{S}$$

$$\mathcal{S} \cup \{\alpha = \tau\} \Rightarrow \mathcal{S}[\tau / \alpha] \cup \{\alpha \cong \tau\}$$

$$\mathcal{S} \cup \{\tau_1 \rightarrow \tau_2 = \tau_3 \rightarrow \tau_4\} \Rightarrow \mathcal{S} \cup \{\tau_1 = \tau_3, \tau_2 = \tau_4\}$$

$$\mathcal{S} \cup \{\text{int} = \text{int}\} \Rightarrow \mathcal{S}$$

# Correctness

---

- Each rule preserves the set of solutions.
  - Rules 1 and 4 eliminate trivial constraints.
  - Rule 2 substitutes equals for equals.
  - Rule 3 is the definition of equality on function types.

$$\mathcal{S} \cup \{\alpha = \alpha\} \Rightarrow \mathcal{S}$$

$$\mathcal{S} \cup \{\alpha = \tau\} \Rightarrow \mathcal{S}[\tau / \alpha] \cup \{\alpha \cong \tau\}$$

$$\mathcal{S} \cup \{\tau_1 \rightarrow \tau_2 = \tau_3 \rightarrow \tau_4\} \Rightarrow \mathcal{S} \cup \{\tau_1 = \tau_3, \tau_2 = \tau_4\}$$

$$\mathcal{S} \cup \{\text{int} = \text{int}\} \Rightarrow \mathcal{S}$$

# Termination

---

- Rules 1 and 4 reduce the number of equations.
- Rule 2 reduces the number of variables in unsolved equations.
- Rule 3 decreases the height of terms.

$$\mathcal{S} \cup \{\alpha = \alpha\} \Rightarrow \mathcal{S}$$

$$\mathcal{S} \cup \{\alpha = \tau\} \Rightarrow \mathcal{S}[\tau / \alpha] \cup \{\alpha \cong \tau\}$$

$$\mathcal{S} \cup \{\tau_1 \rightarrow \tau_2 = \tau_3 \rightarrow \tau_4\} \Rightarrow \mathcal{S} \cup \{\tau_1 = \tau_3, \tau_2 = \tau_4\}$$

$$\mathcal{S} \cup \{\text{int} = \text{int}\} \Rightarrow \mathcal{S}$$

# Termination (Cont.)

---

- Rules 1, 3, and 4 always terminate
  - because terms must eventually be reduced to height 0.
- Eventually rule 2 is applied, reducing the number of variables.

$$\mathcal{S} \cup \{\alpha = \alpha\} \Rightarrow \mathcal{S}$$

$$\mathcal{S} \cup \{\alpha = \tau\} \Rightarrow \mathcal{S}[\tau / \alpha] \cup \{\alpha \cong \tau\}$$

$$\mathcal{S} \cup \{\tau_1 \rightarrow \tau_2 = \tau_3 \rightarrow \tau_4\} \Rightarrow \mathcal{S} \cup \{\tau_1 = \tau_3, \tau_2 = \tau_4\}$$

$$\mathcal{S} \cup \{\text{int} = \text{int}\} \Rightarrow \mathcal{S}$$

# A Nitpick

---

- We really need one more operation
- $\tau = \alpha$  should be flipped to  $\alpha = \tau$  if  $\tau$  is not a variable
  - Needed to ensure rule 2 applies whenever possible.
  - We just assume equations are maintained in this "normal form".

# Solutions

---

- The final system is a solution.
  - There is one equation  $\alpha \cong \tau$  for each variable.
  - This is a substitution with all the solutions of the original system
- Must also perform occurs check to guarantee there are no recursive constraints.

# Example

---

*rewrites*

$$\alpha = \beta \rightarrow \gamma, \alpha = \gamma \rightarrow \beta, \beta = \text{int}$$

# An Example of Failure

---

$$\alpha = \beta \rightarrow \gamma, \alpha = \gamma \rightarrow (\beta \rightarrow \beta), \beta = \text{int}$$

# Notes

---

- The algorithm produces the *most general unifier* of the equations.
  - All solutions are preserved.
- Less general solutions are always substitution instances of the most general solution.

## Next Time ...

---

- Lackwit: A Program Understanding Tool Based on Type Inference
- Type-Based Race Detection for Java