CS 162 Programming languages

Lecture 9: Type Inference

Yu Feng

Winter 2020

H&M

Hindley-Milner type inference algorithm

Robin Milner

1934-2010

Awarded 1991 Turing Award for "*…ML, the first language to include polymorphic type inference and a type-safe exception handling mechanism…*"

Type checking

- (Static) type-checking can reject a program before it runs to prevent the possibility of some errors
  - A feature of statically typed languages
- Dynamically typed languages do little (none?)
  - So might try to treat a number as a function at run-time
- ML (and Java, C#, Scala, C, C++) is statically typed
  - Every binding has one type, determined “at compile-time”

Implicitly typed

- ML is statically typed
- ML is implicitly typed: rarely need to write down types

```
fun f x = (* infer val f : int -> int *)
    if x > 3
    then 42
    else x * 2

fun g x = (* report type error *)
    if x > 3
    then true
    else x * 2
```

- Explicitly typed: like Java or C++!

Type inference

- Type inference problem: Give every binding/expression a type such that type-checking succeeds
  - Fail if and only if no solution exists
- In principle, could be a pass before the type-checker
  - But often implemented together
- Type inference can be easy, difficult, or impossible
  - Easy: Accept all programs
  - Easy: Reject all programs
  - Subtle, elegant, and not magic: ML

Hindley-Milner algorithm

For each top-level definition, traverse each AST to:

- **decorate** each AST node with preliminary type variable
- collect type constraints from each AST node
- use unification to solve constraints and produce a substitution
- use substitution to infer type of definition

Simple-typed lambda calculus

$$e ::= x \mid \lambda x : \tau \,.\, e \mid e_1\ e_2 \mid n$$

$$\tau ::= \textbf{int} \mid X \mid \tau_1 \rightarrow \tau_2$$

To formally define type inference, we introduce a new typing relation:

$$\Gamma \vdash e : \tau \triangleright C$$

Type environment

Meaning: Expression e has type τ provided that every constraint in the set C is satisfied.

Typing rules for STLC

$$\text{CT-VAR}\ \frac{}{\Gamma \vdash x : \tau \triangleright \emptyset}\ x : \tau \in \Gamma$$

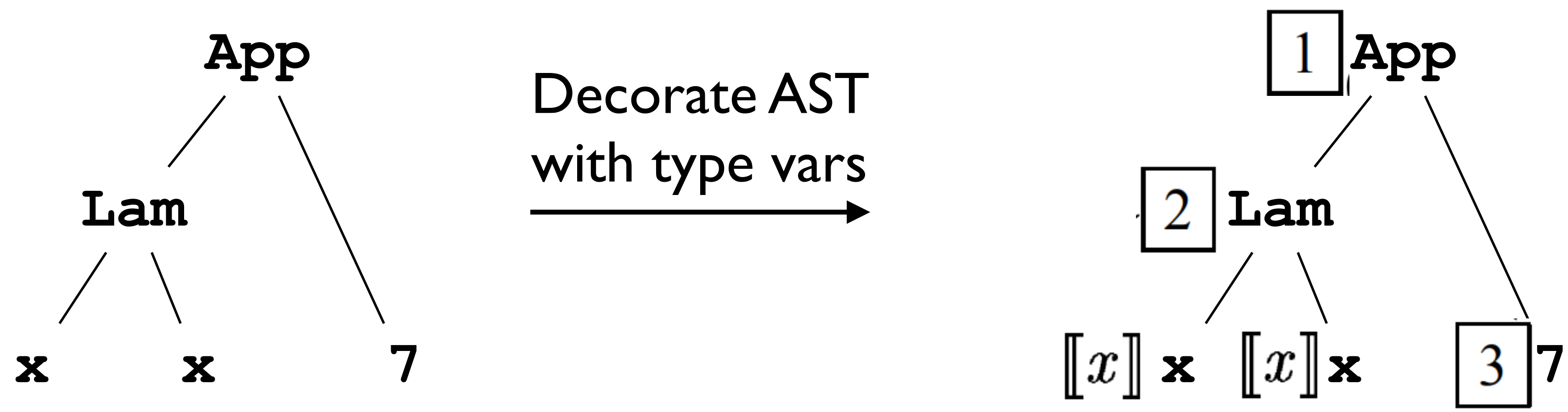
$$\text{CT-INT}\ \frac{}{\Gamma \vdash n : \textbf{int} \triangleright \emptyset}$$

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

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

Example

What is the type of ((\x. x) 7)?



Generate constraints using typing rules

$$[\![\boxed{2}]\!] = [\![\boxed{3}]\!] \rightarrow [\![\boxed{1}]\!]$$

$$[\![\boxed{2}]\!] = [\![x]\!] \rightarrow [\![x]\!]$$

$$[\![\boxed{3}]\!] = \text{number}$$

How to solve those constraints?

Unification algorithm

$$
\begin{array}{lcl}
\mathit{unify}(C) & = & \text{if } C = \emptyset\text{, then } \{\,\} \\
& & \text{else let } \{\mathtt{S} = \mathtt{T}\} \cup C' = C \text{ in} \\
& & \quad \text{if } \mathtt{S} = \mathtt{T} \\
& & \qquad \text{then } \mathit{unify}(C') \\
& & \quad \text{else if } \mathtt{S} = \mathtt{X} \text{ and } \mathtt{X} \notin FV(\mathtt{T}) \\
& & \qquad \text{then } \mathit{unify}(\{\mathtt{X} \mapsto \mathtt{T}\}C') \circ \{\mathtt{X} \mapsto \mathtt{T}\} \\
& & \quad \text{else if } \mathtt{T} = \mathtt{X} \text{ and } \mathtt{X} \notin FV(\mathtt{S}) \\
& & \qquad \text{then } \mathit{unify}(\{\mathtt{X} \mapsto \mathtt{S}\}C') \circ \{\mathtt{X} \mapsto \mathtt{S}\} \\
& & \quad \text{else if } \mathtt{S} = \mathtt{S}_1 \to \mathtt{S}_2 \text{ and } \mathtt{T} = \mathtt{T}_1 \to \mathtt{T}_2 \\
& & \qquad \text{then } \mathit{unify}(C' \cup \{\mathtt{S}_1 = \mathtt{T}_1, \mathtt{S}_2 = \mathtt{T}_2\}) \\
& & \quad \text{else} \\
& & \qquad \text{fail}
\end{array}
$$

Occurrence check:
make sure T does not
contain type variable X

Types and Programming Languages p326

Example

How to solve those constraints?

$$\llbracket \boxed{2} \rrbracket = \llbracket \boxed{3} \rrbracket \rightarrow \llbracket \boxed{1} \rrbracket$$

$$\llbracket \boxed{2} \rrbracket = \llbracket x \rrbracket \rightarrow \llbracket x \rrbracket$$

$$\llbracket \boxed{3} \rrbracket = \mathsf{number}$$

Action	Stack	Substitution
Initialize	$\llbracket \boxed{2} \rrbracket = \llbracket \boxed{3} \rrbracket \rightarrow \llbracket \boxed{1} \rrbracket$ $\llbracket \boxed{2} \rrbracket = \llbracket x \rrbracket \rightarrow \llbracket x \rrbracket$ $\llbracket \boxed{3} \rrbracket = \mathsf{number}$	empty
Step 3	$\llbracket \boxed{3} \rrbracket \rightarrow \llbracket \boxed{1} \rrbracket = \llbracket x \rrbracket \rightarrow \llbracket x \rrbracket$ $\llbracket \boxed{3} \rrbracket = \mathsf{number}$	$\llbracket \boxed{2} \rrbracket \mapsto \llbracket \boxed{3} \rrbracket \rightarrow \llbracket \boxed{1} \rrbracket$
Step 5	$\llbracket \boxed{3} \rrbracket = \llbracket x \rrbracket$ $\llbracket \boxed{1} \rrbracket = \llbracket x \rrbracket$ $\llbracket \boxed{3} \rrbracket = \mathsf{number}$	$\llbracket \boxed{2} \rrbracket \mapsto \llbracket \boxed{3} \rrbracket \rightarrow \llbracket \boxed{1} \rrbracket$
Step 3	$\llbracket \boxed{1} \rrbracket = \llbracket x \rrbracket$ $\llbracket x \rrbracket = \mathsf{number}$	$\llbracket \boxed{2} \rrbracket \mapsto \llbracket x \rrbracket \rightarrow \llbracket \boxed{1} \rrbracket$ $\llbracket \boxed{3} \rrbracket \mapsto \llbracket x \rrbracket$
Step 3	$\llbracket x \rrbracket = \mathsf{number}$	$\llbracket \boxed{2} \rrbracket \mapsto \llbracket x \rrbracket \rightarrow \llbracket x \rrbracket$ $\llbracket \boxed{3} \rrbracket \mapsto \llbracket x \rrbracket$ $\llbracket \boxed{1} \rrbracket \mapsto \llbracket x \rrbracket$
Step 3	empty	$\llbracket \boxed{2} \rrbracket \mapsto \mathsf{number} \rightarrow \mathsf{number}$ $\llbracket \boxed{3} \rrbracket \mapsto \mathsf{number}$ $\llbracket \boxed{1} \rrbracket \mapsto \mathsf{number}$ $\llbracket x \rrbracket \mapsto \mathsf{number}$

https://cs.brown.edu/courses/cs173/2002/Lectures/2002-11-13.pdf