Simple Type Inference, cont.
Announcements

- HW6 is a team homework

- Presentation guidelines are up, papers will be up on Schedule page by the end of week
  1. Select available paper/slot from list (first-come-first-serve)
  2. If available, I’ll assign and update, otherwise goto 1

- Quiz 5
Outline

- Simple type inference, cont.
  - Equality constraints, unification and substitution
  - Strategy 1: Constraint-based typing
  - Strategy 2: On-the-fly typing
  - The Let construct

- Examples in context of HW6 starter code

- Hindley Milner: next week
Unification
(essential for type inference!)

- **Unify**: tries to unify $\tau_1$ and $\tau_2$ and returns a principal unifier for $\tau_1 = \tau_2$ if unification is successful.

```python
def Unify(\tau_1, \tau_2) =
  \text{case } (\tau_1, \tau_2) \text{ of }
  (\tau_1, t_2) = [\tau_1 / t_2] \text{ provided } t_2 \text{ does not occur in } \tau_1
  (t_1, \tau_2) = [\tau_2 / t_1] \text{ provided } t_1 \text{ does not occur in } \tau_2
  (b_1, b_2) = \text{if } (\text{eq? } b_1 b_2) \text{ then } \square \text{ else } \text{fail}
  (\tau_{11} \rightarrow \tau_{12}, \tau_{21} \rightarrow \tau_{22}) = \text{let } S_1 = \text{Unify}(\tau_{11}, \tau_{21})
  S_2 = \text{Unify}(S_1(\tau_{12}), S_1(\tau_{22}))
  \text{in } S_2 S_1 // \text{compose substitutions}
  \text{otherwise } = \text{fail}
```

This is the occurs check!
Examples

- **Unify** (int→int, t₁→t₂) yields ?

- **Unify** (int, int→t₂) yields ?

- **Unify** (t₁, int→t₂) yields ?

\[ t₁ \rightarrow t₃ = \text{num} \rightarrow t₂ \rightarrow t₃^2 \]

\[ t₂ \rightarrow t₃ = \text{num} \rightarrow (t₂ \rightarrow t₃) \]

FAILS
because of occurs check
Unify Set of Constraints \( C \)

- **UnifySet**: tries to unify \( C \) and returns a principal unifier for \( C \) if unification is successful

```python
def UnifySet (C) =
    if C is Empty Set then []
    else let
        \( C = \{ \tau_1 = \tau_2 \} \cup C' \)
        \( S = \text{Unify} (\tau_1, \tau_2) \) // Unify returns a substitution \( S \)
in
        UnifySet ( S(C') ) ° S
    // Composition of substitutions
```

Program Analysis CSCI 4450/6450, A Milanova
Examples

- \{ t_1 = \text{int}, t_2 = t_1 \rightarrow t_1 \}\n
- \{ t_1 \rightarrow t_2 = t_2 \rightarrow t_3, t_3 = t_4 \rightarrow t_5 \}\n
\[ [t_1/t_2, t_1/t_3] \leftarrow \text{Unify} (t_1 \rightarrow t_2, t_2 \rightarrow t_3) \]

\[ S_1 = \begin{array}{l}
\text{Unify Set} (\{ t_2/t_2, t_1/t_3 \cup \{t_3 = t_4 \rightarrow t_5\}\})
\end{array} \]

\[ [t_4/t_5/t_2] \leftarrow \text{Unify} (t_4, t_4 \rightarrow t_5) \]

\[ \{ t_1 = t_2 \rightarrow t_1, t_1 = t_2 \rightarrow t_2 \}\]

\[ [t_1/t_2, t_1/t_3, t_4 \rightarrow t_5/t_2] \]
Type Inference, Strategy 1

- Aka constraint-based typing (e.g., Pierce)

- Traverse parse tree to derive a set of type constraints \( C \)
  - These are equality constraints
  - (Pseudo code in Lecture20)

- Solve type constraints offline
  - Use unification algorithm
  - (Pseudo code in Lecture20)
Outline

- Simple type inference, cont.
  - Equality constraints, Unification, Substitution
  - Strategy 1: Constraint-based typing
  - Strategy 2: On-the-fly typing
  - Let expressions

- Examples in context of HW6

- Hindley Milner: next week
Type Inference, Strategy 2

- Strategy 1 collects all constraints, then solves them offline

- Strategy 2 solves constraints on the fly
  - Builds the substitution map incrementally
  - Key reason: infers types as parser parses program!
Add a New Attribute, Substitution Map $S$

Grammar rule:

- $E ::= x$
- $E ::= c$
- $E ::= \lambda x. E_1$

Attribute rule:

- $T_E = \Gamma_E(x)$, $S_E = [ ]$
- $T_E = \text{int}$, $S_E = [ ]$
- $\Gamma_{E_1} = \Gamma_E; x:t_x$
- $T_E = S_{E_1}(t_x) \rightarrow T_{E_1}$, $S_E = S_{E_1}$

$T_{E_1} = T_{E_2} \rightarrow T_E$

$E ::= E_1 E_2$

$S = \text{Unify}(S_{E_2}(T_{E_1}), T_{E_2} \rightarrow t_E)$

$T_E = S(t_E)$, $S_E = S S_{E_2} S_{E_1}$

$T_E$ is a principal type of $E$.
$S_E$ is substitution map that results from inferring $T_E$.
$t_x, t_E$ are fresh type variables.

Program Analysis CSCI 4450/6450, A Milanova
Example: \((\lambda f. f\ 5)\ (\lambda x.\ x)\)

\((\lambda f. f\ 5)\ (\lambda x.\ x) : ?\)

1. App

- \(\Gamma_1 = []\)
- \(T_1 = \text{int}\)
- \(S_1 = [\ \text{int/t}_x,\ \text{int/t}_3,\ \text{int/t}_1,\ \text{int} \rightarrow \text{int/t}_f]\)

2. Abs

- \(\Gamma_2 = []\)
- \(T_2 = (\text{int} \rightarrow \text{t}_3) \rightarrow \text{t}_3\)
- \(S_2 = [\ \text{int} \rightarrow \text{t}_3 \rightarrow \text{t}_f]\)

3. App

- \(\Gamma_3 = [f:\text{t}_f]\)
- \(T_3 = \text{t}_3\)
- \(S_3 = [\ \text{int} \rightarrow \text{t}_3 \rightarrow \text{t}_f]\)

4. Abs

- \(\Gamma_4 = S_2(\Gamma_1) = []\)
- \(T_4 = \text{t}_x \rightarrow \text{t}_x\)
- \(S_4 = []\)

\(\Gamma = [x:\text{t}_x]\)

- \(T = \text{t}_x\)
- \(S = []\)

from Unify(\(\text{t}_f,\ \text{int} \rightarrow \text{t}_3\))