Tutorial-11

TA - Sinong

Quiz-10

1. Which is not correct about polymorphism?

a. A term can be used in many concrete contexts with different concrete types.

b. It is the ability of an object to take on many forms.

c. It makes typed constructs useful in more contexts.

d. Existential polymorphism is about code reuse.

Response to Criticisms of Typed Languages

- Types overly constrain functions & data
  - Polymorphism makes typed constructs useful in more contexts
    - universal polymorphism => code reuse
      - \x.x : ‘a → ‘a (* ’a is any type *)
      - reverse : ‘a list → ‘a list (* ’a is any type *)
    - existential polymorphism => modules & abstract data types
      - $T = \exists X\ \{a{:}\,X;\ f{:}\,X \rightarrow bool\}$
      - intT = {a: int; f: int → bool}
      - boolT = {a: bool; f: bool → bool}
- Types clutter programs and slow down programmer productivity
  - Type inference.
    - uninformative annotations may be omitted

1. Which is not correct about polymorphism?

a. A term can be used in many concrete contexts with different concrete types.

b. It is the ability of an object to take on many forms.

c. It makes typed constructs useful in more contexts.

d. Existential polymorphism is about code reuse.

2. Typed language need type inference.

a. True

b. False

2. Typed language need type inference.

a. True

b. False

in typed language the type is already annotated.

3. Which one is not a step of type inference?

a. Add type schemas

b. Generate type constraints

c. Determine subtypes

d. Solve type constraints

Bonus Point: Which four steps?

Please describe without handout

- STEP 1: ADD TYPE SCHEMES
- STEP 2: GENERATE CONSTRAINTS
- STEP 3: SOLVE CONSTRAINTS
- STEP 4: GENERATE TYPES

3. Which one is not a step of type inference?

a. Add type schemas

b. Generate type constraints

c. Determine subtypes

d. Solve type constraints

4. In the step of constraint generation, which simple rule is not totally correct?

a. G |-- x ==> x : s, {}

b. G |-- 2 ==> 2 : int, {}

c. G |-- false ==> false : bool, {}

d. G |-- true ==> true : bool, {}

Constraint Generation

- Simple rules:
  - G |-- x ==> x : s, { }   (if G(x) = s)
    - If G(x) is not defined then x is free variable

  - G |-- 3 ==> 3 : int, { }   (same for other ints)

  - G |-- true ==> true : bool, { }

  - G |-- false ==> false : bool, { }

4. In the step of constraint generation, which simple rule is not totally correct?

a. G |-- x ==> x : s, {}

b. G |-- 2 ==> 2 : int, {}

c. G |-- false ==> false : bool, {}

d. G |-- true ==> true : bool, {}

5. Try to write down the constraint generation rules of function application. (Here is the rule of + operation)

<Bonus Point> Write on the white board without handout

5. Try to write down the constraint generation rules of function application. (Here is the rule of + operation)

```
G |-- u1 ==> e1 : t1, q1
G |-- u2 ==> e2 : t2, q2
---------------------------------------------------------
G |-- u1 u2==> e1 e2: a, q1 U q2 U {t1 = t2 -> a}
```

6. If type variable a is not in the domain of substitution S, then S(a) = ?

6. If type variable a is not in the domain of substitution S, then S(a) = ?

a

7. What is the application order of (U o S) (a)

U(S(a))

8. What is the principal solution for the following equations?

$q = \{a = b, b = c\text{->}c, c = int\}$

<Bonus Point>

8. What is the principal solution for the following equations?
q = {a = b, b = c->c, c = int}

S(c)=int, S(b)=S(a)=int->int,
S(d)=d (for all d other than a, b, c

Homework-10

Problem1 - 30%

Prove the Lemma: If $(S, q) \rightarrow (S', q')$ then:

- T is complete for (S, q) iff T is complete for (S', q')
- T is principal for (S, q) iff T is principal for (S', q')

$$\frac{}{(S,\{int=int\} \cup q) \rightarrow (S, q)}$$

$$\frac{}{(S,\{s11 \rightarrow s12 = s21 \rightarrow s22\} \cup q) \rightarrow (S, \{s11 = s21, s12 = s22\} \cup q)}$$

$$\frac{}{(S,\{bool=bool\} \cup q) \rightarrow (S, q)}$$

$$\frac{}{(S,\{a=s\} \cup q) \rightarrow ([a=s] \circ S, q[s/a])} \ (a \text{ not in } FV(s))$$

$$\frac{}{(S,\{a=a\} \cup q) \rightarrow (S, q)}$$

$$\frac{}{(S,\{s=a\} \cup q) \rightarrow ([a=s] \circ S, q[s/a])} \ (a \text{ not in } FV(s))$$

(S is a solution to the constraints q)

$$\frac{}{S \models \{\,\}}$$

$$\frac{S(s1) = S(s2) \qquad S \models q}{S \models \{s1 = s2\} \cup q}$$

Complete Solutions

- A complete solution for (S, q) is a substitution T such that
  1. T <= S
  2. T |= q
  - intuition: T extends S and solves q

- A principal solution T for (S, q) is complete for (S, q) and
  3. for all T' such that 1. and 2. hold, T' <= T
  - intuition: T is the most general solution (it's the least restrictive)

- **Case:** $\frac{}{(S,\{int=int\}\cup q)\rightarrow(S,q)}$**(u-int)**

  Need to prove: T is complete for $(S, \{int = int\} \cup q)$ iff T is complete for (S, q)

  a) $\rightarrow$

  $$\begin{aligned}
  &(1)\ T\ is\ complete\ for\ (S, \{int = int\} \cup q) && (by\ assumption)\\
  &(2)\ T <= S,\\
  &\quad T| = \{int = int\} \cup q && (by\ (1))\\
  &(3)\ T| = q && (by\ (2)\ and\ inversion\ of\ S-equal)\\
  &(4)\ T\ is\ complete\ for\ (S, q) && (by\ (2)\ and\ (3))
  \end{aligned}$$

  b) $\leftarrow$

  $$\begin{aligned}
  &(1)\ T\ is\ complete\ for\ (S, q) && (by\ assumption)\\
  &(2)\ T <= S,\\
  &\quad T| = q && (by\ (1))\\
  &(3)\ T(int) = T(int)\\
  &(4)\ T| = \{int = int\} \cup q && (by\ (2), (3)\ and\ S-equal)\\
  &(5)\ T\ is\ complete\ for\ (S, \{int = int\} \cup q) && (by\ (2)\ and\ (4))
  \end{aligned}$$

Need to prove: T is principal for $(S, \{int = int\} \cup q)$ iff T is principal for (S', q')

a) $\rightarrow$

$$
\begin{array}{lr}
(1)\ T\ is\ principal\ for\ (S, \{int = int\} \cup q) & (by\ assumption) \\
(2)\ T\ is\ complete\ for\ (S, \{int = int\} \cup q) & (by\ (1)) \\
(3)\ T\ is\ complete\ for\ (S', q') & (by\ (2)) \\
(4)\ For\ any\ complete\ solution\ T'\ for\ (S', q'), & \\
\quad T'\ is\ complete\ for\ (S, \{int = int\} \cup q) & \\
(5)\ T' <= T & (by\ (1)) \\
(6)\ T\ is\ principal\ for\ (S', q') & (by\ (3)\ and\ (5))
\end{array}
$$

b) $\leftarrow$

$$
\begin{array}{lr}
(1)\ T\ is\ principal\ for\ (S', q') & (by\ assumption) \\
(2)\ T\ is\ complete\ for\ (S', q') & (by\ (1)) \\
(3)\ T\ is\ complete\ for\ (S, \{int = int\} \cup q) & (by\ (2)) \\
(4)\ For\ any\ complete\ solution\ T'\ for\ (S, \{int = int\} \cup q), & \\
\quad T'\ is\ complete\ for\ (S', q') & \\
(5)\ T' <= T & (by\ (1)) \\
(6)\ T\ is\ principal\ for\ (S, \{int = int\} \cup q) & (by\ (3)\ and\ (5))
\end{array}
$$

- **Case:** $\overline{(S,\{bool=boolt\}\cup q)\rightarrow(S,q)}$ **(u-bool)**

  Similar to u-int.

- **Case:** $\overline{(S,\{a=a\}\cup q)\rightarrow(S,q)}$ **(u-eq)**

  Similar to u-int.

- **Case:** $\frac{}{(S,\{s_{11} \to s_{12} = s_{21} \to s_{22}\} \cup q) \to (S,\{s_{11} = s_{21}, s_{12} = s_{22}\} \cup q)}$ **(u-fun)**

  Need to prove: T is complete for $(S, \{s_{11} \to s_{12} = s_{21} \to s_{22}\} \cup q)$ iff T is complete for $(S, \{s_{11} = s_{21}, s_{12} = s_{22}\} \cup q)$

  a) $\to$

  $$\begin{array}{lr}
(1)\ T\ is\ complete\ for\ (S, \{s_{11} \to s_{12} = s_{21} \to s_{22}\} \cup q) & (by\ assumption) \\
(2)\ T <= S, & \\
\quad T| = \{s_{11} \to s_{12} = s_{21} \to s_{22}\} \cup q & (by\ (1)) \\
(3)\ T(s_{11} \to s_{12}) = T(s_{21} \to s_{22}) & \\
\to T(s_{11}) \to T(s_{12}) = T(s_{21}) \to T(s_{22}) & (by\ (2)\ and\ inversion\ of\ S-equal) \\
(4)\ T(s_{11}) = T(s_{21}), T(s_{12}) = T(s_{22}) & (by\ (3)) \\
(5)\ T| = \{s_{11} = s_{21}, s_{12} = s_{22}\} \cup q & (by\ (4)\ and\ S-equal) \\
(6)\ T\ is\ complete\ for\ (S, \{s_{11} = s_{21}, s_{12} = s_{22}\} \cup q) & (by\ (2)\ and\ (5))
\end{array}$$

  b) $\leftarrow$

  $$\begin{array}{lr}
(1)\ T\ is\ complete\ for\ (S, \{s_{11} = s_{21}, s_{12} = s_{22}\} \cup q) & (by\ assumption) \\
(2)\ T <= S, & \\
\quad T| = \{s_{11} = s_{21}, s_{12} = s_{22}\} \cup q & (by\ (1)) \\
(3)\ T| = q & \\
\quad T(s_{11}) = T(s_{21}) & \\
\quad T(s_{12}) = T(s_{22}) & (by\ (2)\ and\ inversion\ of\ S-equal) \\
(4)\ T(s_{11} \to s_{12}) = T(s_{11}) \to T(s_{12}) & \\
= T(s_{21}) \to T(s_{22}) = T(s_{21} \to s_{22}) & (by\ (3)) \\
(5)\ T| = \{s_{11} \to s_{12} = s_{21} \to s_{22}\} \cup q & (by\ (3), (4)\ and\ S-equal)) \\
(6)\ T\ is\ complete\ for\ (S, \{int = int\} \cup q) & (by\ (2)\ and\ (5))
\end{array}$$

Need to prove: T is principal for $(S, \{s_{11} \to s_{12} = s_{21} \to s_{22}\} \cup q)$ iff T is principal for $(S, \{s_{11} = s_{21}, s_{12} = s_{22}\} \cup q)$

Similar to u-int.

- **Case:** $\frac{}{(S,\{a=s\}\cup q)\rightarrow([a=s]\circ S, q[s/a])}$ **(a not in FV(s))(u-var1)**

  Need to prove: T is complete for $(S, \{a = s\} \cup q)$ iff T is complete for $([a = s] \circ S, q[s/a])$

  a) $\rightarrow$

  $$\begin{array}{llr} (1) & T \textit{ is complete for } (S, \{a = s\} \cup q) & (\textit{by assumption}) \\ (2) & T <= S, & \\ & T| = \{a = s\} \cup q & (\textit{by } (1)) \\ (3) & T(a) = T(s) & \\ & T| = q & (\textit{by } (2) \textit{ and inversion of } S - \textit{equal}) \\ (4) & T| = q[s/a] & (\textit{by } (3) \textit{ and lemma} \boxed{1}) \\ (5) & T <= [a = s] \circ S & (\textit{by } (2), (3) \textit{ and lemma} \boxed{2}) \\ (6) & T \textit{ is complete for } ([a = s] \circ S, q[s/a]) & (\textit{by } (4) \textit{ and } (5)) \end{array}$$

  b) $\leftarrow$

  $$\begin{array}{llr} (1) & T \textit{ is complete for } ([a = s] \circ S, q[s/a]) & (\textit{by assumption}) \\ (2) & T <= [a = s] \circ S, & \\ & T| = q[s/a] & (\textit{by } (1)) \\ (3) & T = U \circ [a = s] \circ S <= S & (\textit{by } (2)) \\ (4) & a \notin dom(S), s \notin dom(S) & \\ (5) & T(a) = T(s) & (\textit{by } (4)) \\ (6) & T| = q & (\textit{by } (2), (5) \textit{ and inversion of lemma} \boxed{1}) \\ (7) & T| = \{a = s\} \cup q & (\textit{by } (5), (6) \textit{ and } S - \textit{equal}) \\ (7) & T \textit{ is complete for } (S, \{a = s\} \cup q) & (\textit{by } (3) \textit{ and } (6)) \end{array}$$

Lemma 1. *If* $T(m) = T(n), T| = q$, *then* $T| = q[n/m]$

Proof. Prove: By induction on the derivation of $S| = q$
case S-empty: obviously
case S-equal: If m=a or m=b .... else .... (Here we skip the proof steps)

And it's easy to prove the inversion lemma is also right, which is
If $T(m) = T(n), T| = q[n/m]$, then $T| = q$

□

Lemma 2. *If* $T(a) = T(s), T <= S$, *then* $T <= [a = s] \circ S$

Proof. Prove:Suppose $T = U \circ S$
Let $S' = U \circ [a = s] \circ S$, for all variables x
If $x \neq a$, T(x) = U(S(x)), S'(x)=U(S(x)) = T(x)
If $x = a$,
    if $a \in dom(S)$, S'(a)=U(S(a)) = T(a).
    if $a \notin dom(S)$, S'(a) = U([a=s](S(a))) = U([a=s](a)) = U(s)
        if $s \notin dom(S)$, T(a) = T(s) = U(S(s)) = U(s) = S'(a)
        if $s \in dom(S)$ T(a) = T(s) = U(S(s)), let $S' = U \circ [s = S(s)] \circ [a = s] \circ S$,
            S'(a) = U(S(s)) = T(a)
So $T = S'$. Because $S' <= [a = s] \circ S$, so $T <= [a = s] \circ S$

Need to prove: T is principal for $(S, \{a = s\} \cup q)$ iff T is principal for $([a = s] \circ S, q[s/a])$

Similar to u-int.

- **Case:** $\frac{}{(S, \{s=a\} \cup q) \rightarrow ([a=s] \circ S, q[s/a])}$ **(a not in FV(s))(u-var2)**

  Similar to u-var2

Problem 3. Show why type checking let expression using [t-LetPoly] is exponential in time and give an amortised linear implementation of let polymorphism instead.

Solution. Suppose the length of the input term e_0 is n. e_0 is a let expression like $let\ x = e_1\ in\ x\ x\ x\ x...$ and $e_1 = let\ x = e_2\ in\ x\ x\ x....$ The length of e_1 is $n/2$. Repeat this step so that e_1, e_2, e_3 have the same formulations as e_0. In this case the time complexity is $O(n/2) * O(n/4) * O(n/8).... = O(n^{\log n})$, which is exponential.

We can solve $let\ x = e_1\ in\ e_2$ in this way:

1. Once we get the principal type t_1 of e_1, we don't bind it with x in context Γ. We find all free variables in t_1. Suppose they are $x_1, .., x_n$. Now we bind x with a special type scheme $\forall x_1...x_n.t_1$.

2. We do typecheck for e_2. Each time we encounter an occurrence of x in e_2, we generate type variables $y_1, ...y_n$ and use them to instantiate $\forall x_1...x_n.t_1$, yielding $t_1[y_1/x_1, ..., y_n/x_n]$

□