Type Inference

1 Introduction

You will undertake a few simple problems in order to develop basic skills with type inference.

2 Description

- For each of the following \((\text{val}\ \text{x}\ e)\) definitions, write down the constraints \(C\) and type \(\tau\) that are inferred for the expression \(e\) (use the derivation templates on the following pages), a most general substitution \(\theta\) that solves the constraint, and the type scheme \(\sigma\) that is inferred for the variable \(\text{x}\) when type checked with the initial environment

\[
\Gamma_0 = \{\text{nil} \mapsto \forall \alpha . \alpha\ \text{list}, \text{cons} \mapsto \forall \alpha . \alpha \times \alpha\ \text{list} \mapsto \alpha\ \text{list}, \\
\text{null?} \mapsto \forall \alpha . \alpha\ \text{list} \mapsto \text{bool}, \text{car} \mapsto \forall \alpha . \alpha\ \text{list} \mapsto \alpha, \text{cdr} \mapsto \forall \alpha . \alpha\ \text{list} \mapsto \alpha\ \text{list}, \\
* \mapsto \forall \alpha . \alpha\ \text{int} \times \alpha\ \text{int} \mapsto \alpha, \text{+} \mapsto \forall \alpha . \alpha\ \text{int} \times \alpha\ \text{int} \mapsto \alpha, \text{-} \mapsto \forall \alpha . \alpha\ \text{int} \times \alpha\ \text{int} \mapsto \alpha, \text{*} \mapsto \forall \alpha . \alpha\ \text{int} \times \alpha\ \text{int} \mapsto \alpha, \text{/} \mapsto \forall \alpha . \alpha\ \text{int} \times \alpha\ \text{int} \mapsto \alpha, \text{=} \mapsto \forall \alpha . \alpha\ \text{int} \times \alpha\ \text{int} \mapsto \text{bool}, \text{<} \mapsto \forall \alpha . \alpha\ \text{int} \times \alpha\ \text{int} \mapsto \text{bool}, \text{>} \mapsto \forall \alpha . \alpha\ \text{int} \times \alpha\ \text{int} \mapsto \text{bool}\}
\]

- (val singleton (lambda (x) (cons x nil)))
- (val mcons (lambda (p x l) (if (p x) (cons x l) l))

- (Adapted from Exercises 10 and 11 of Chapter 7 from Programming Languages: Build, Prove, and Compare (p. 534).)

- Find two constraints \(C_1\) and \(C_2\) and two substitutions \(\theta_1\) and \(\theta_2\) such that
  * \(C_1\) has one or more free (unification) type variables,
  * \(C_2\) has one or more free (unification) type variables,
  * \(C_1\) is satisfied by \(\theta_1\),
  * \(C_2\) is satisfied by \(\theta_2\),
  * \(C_1 \land C_2\) is satisfied by \(\theta_2 \circ \theta_1\).

- Find two constraints \(C_1\) and \(C_2\) and two substitutions \(\theta_1\) and \(\theta_2\) such that
  * \(C_1\) has one or more free (unification) type variables,
  * \(C_2\) has one or more free (unification) type variables,
  * \(C_1\) is satisfied by \(\theta_1\),
  * \(C_2\) is satisfied by \(\theta_2\),
  * \(C_1 \land C_2\) is not satisfied by any \(\theta\).

- Find two constraints \(C_1\) and \(C_2\) and three substitutions \(\theta_1\), \(\theta_2\), and \(\theta_3\) such that
  * \(C_1\) has one or more free (unification) type variables,
  * \(C_2\) has one or more free (unification) type variables,
  * \(C_1\) is satisfied by \(\theta_1\),
  * \(C_2\) is satisfied by \(\theta_2\),
  * \(C_1 \land C_2\) is satisfied by \(\theta_3\),
  * \(C_1 \land C_2\) is not satisfied by \(\theta_2 \circ \theta_1\).

3 Requirements and Submission

You may use the reference interpreter (see Appendix X), but there may only be one active laptop in each group.

At the end of class, submit the group’s solutions as hard-copy; be sure to include the names of all group members in the submission.
A Interpreter

A reference nano-ML interpreter is available on the CS Department Linux systems (e.g., glados.cs.rit.edu and queeg.cs.rit.edu and ICLs 1 and 2) at:

/usr/local/pub/mtf/plc/bin/nml

Use the reference interpreter to check your code.
B Constraint-based Typing Rules

\[\text{generalize}(\tau, A) = \forall \alpha_1, \ldots, \alpha_n. \tau|a_1 \mapsto \alpha_1, \ldots, a_n \mapsto \alpha_n]\] where \(\{a_1, \ldots, a_n\} = \text{futv}(\tau) - A\)

\(C\) solved by \(\theta\)

\[
\begin{array}{lcl}
T \text{ solved by } \{\} & \quad & C_1 \text{ solved by } \theta_1 \quad \theta_1(C_2) \text{ solved by } \theta_2 \\
\text{int} \sim \text{int} \text{ solved by } \{\} & \quad & \tau_1 \sim \tau'_1 \land \cdots \land \tau_n \sim \tau'_n \land \tau_r \sim \tau'_r \text{ solved by } \theta \\
\text{bool} \sim \text{bool} \text{ solved by } \{\} & \quad & \tau \sim \tau' \text{ solved by } \theta \\
\end{array}
\]

\(\tau \sim \tau'\) list solved by \(\theta\)

\(C, \Gamma \vdash e : \tau\)

\[
\begin{align*}
\text{If} & : \quad C_1, \Gamma \vdash e_1 : \tau_1 & C_2, \Gamma \vdash e_2 : \tau_2 & C_3, \Gamma \vdash e_3 : \tau_3 \\
\text{Lam} & : \quad C_1, \Gamma \vdash \lambda a_1, \ldots, a_n.x : \tau_n \\
\text{App} & : \quad C_f, \Gamma \vdash e_f : \tau_f & C_1, \Gamma \vdash e_1 : \tau_1 & \cdots & C_n, \Gamma \vdash e_n : \tau_n \\
\text{LetRec} & : \quad C, \Gamma \vdash \text{letrec}(\langle x_1, e_1, \ldots, x_n, e_n \rangle) : \tau_b
\end{align*}
\]

\(\langle d, \Gamma \rangle \rightarrow \Gamma'\)

\[
\begin{align*}
\text{Val} & : \quad C, \Gamma \vdash x : \sigma & C \text{ a fresh} & C \vdash e : \tau & C \& a \sim \tau \text{ solved by } \theta \\
\text{ValRec} & : \quad \langle \text{VAL}(x, e), \Gamma \rangle \rightarrow \Gamma \{x \mapsto \sigma\}
\end{align*}
\]
Derivation Templates

\[
\begin{align*}
\text{(LAM)} & \quad \text{(APP)} \\
\text{(VAR)} & \quad \text{a}_x \text{ fresh} \\
\Gamma_1(\text{cons}) = \forall \alpha. \alpha \times \alpha \text{ list} \to \alpha \text{ list} & \quad \text{a}_1 \text{ fresh} \\
01, \Gamma_1 \vdash \text{cons} : 02 & \\
\text{(VAR)} & \quad \Gamma_1(\text{x}) = \forall \alpha. \alpha_x \\
03, \Gamma_1 \vdash \text{x} : 04 & \\
\text{(VAR)} & \quad \Gamma_1(\text{nil}) = \forall \alpha. \alpha \text{ list} & \quad \text{a}_2 \text{ fresh} \\
05, \Gamma_1 \vdash \text{nil} : 06 & \\
\Gamma_1 = \Gamma_0 & \\
\text{C} = 09 & \\
\theta = 11 & \\
\theta(10) = 12 & \\
\end{align*}
\]

\[
\begin{align*}
\text{C}, \Gamma_0 \vdash \text{(lambda} (\text{x}) (\text{cons} x \text{ nil})) : 10 & \quad \text{C sat by } \theta \quad \text{generalize}(\theta(10), \emptyset) = 13 \\
\langle \Gamma_0, \text{(val singleton } (\text{lambda} (\text{x}) (\text{cons} x \text{ nil}))) \rangle \to \Gamma_0\{\text{singleton} \mapsto 13\} & \\
\end{align*}
\]
\[ \text{(Var)} \quad \Gamma_1(p) = \forall . a_p \]

\[ \text{Γ}_0 \vdash p : 02 \]

\[ \text{(Var)} \quad \Gamma_1(x) = \forall . a_x \]

\[ \text{Γ}_0 \vdash x : 04 \]

\[ \Gamma_1(l) = \forall . a_l \]

\[ \text{Γ}_0 \vdash l : 12 \]

\[ \text{Γ}_1 \vdash (\text{if} \ (p \ x) \ (\text{cons} \ x \ l) \ l) : 18 \]

\[ \text{Γ}_0 \vdash (\text{lambda} \ (p \ x \ l) \ (\text{if} \ (p \ x) \ (\text{cons} \ x \ l) \ l)) : 20 \]

\[ C, \Gamma_0 \vdash (\text{lambda} \ (p \ x \ l) \ (\text{if} \ (p \ x) \ (\text{cons} \ x \ l) \ l)) : 20 \]

\[ C \text{ sat by } \theta \quad \text{generalize}(\theta(20), \emptyset) = 23 \]

\[ (\Gamma_0, (\text{val} \ mcons \ (\text{lambda} \ (p \ x \ l) \ (\text{if} \ (p \ x) \ (\text{cons} \ x \ l) \ l)))) \rightarrow \Gamma_0 \{ \text{mcons} \mapsto 23 \} \]

\[ \Gamma_1 = \Gamma_0 \{ p \mapsto \forall . a_p, x \mapsto \forall . a_x, l \mapsto \forall . a_l \} \]

\[ C = 19 \]

\[ \theta = 21 \]

\[ \theta(20) = 22 \]