HW 5 – Polymorphic Type Inference

CS 421 – Fall 2012
Revision 1.0

Assigned October 2, 2012
Due October 16, 2012, 11:59 pm
Extension 48 hours (20% penalty)

1 Change Log

1.0 Initial Release.

2 Turn-In Procedure

Answer the problem below, save your work as a PDF (either scanned if handwritten or converted from a program), and hand in the PDF. Your file should be named `hw5.pdf`.

3 Objectives and Background

The purpose of this HW is to test your understanding of how to use typing rules to perform polymorhic type derivations in a funtional programming language (here with OCaml syntax). Another purpose of HWs is to provide you with experience answering non-programming written questions of the kind you may experience on the midterms and final.

4 Problems

1. (25 points) Give a complete type derivation for the following typing judgment's.

$$\left\{\begin{array}{l}\texttt{tl}:\forall\texttt{'a. 'a list -> 'a list}\\ \texttt{(::)}:\forall\texttt{'a. 'a -> 'a list -> 'a list}\\ \texttt{(=)}:\forall\texttt{'a. 'a -> 'a -> bool}\end{array}\right\}\vdash\begin{array}{l}\texttt{(let rec length =}\\ \texttt{\ \ fun lst -> if lst = [] then 0}\\ \texttt{\ \ \ \ \ \ \ \ \ \ \ \ \ else 1 + length (tl(lst))}\\ \texttt{\ in}\\ \texttt{\ \ length (("a" :: []) :: [])}\\ \texttt{\ ): int}\end{array}$$

As a suggestion for formatting, you may want to name subtrees of the proof and write them out separately. Note, we are asking for a type judgment not the intermediate state of a type inferencing algorithm.

Solution:

Let

$$\begin{aligned}
\Gamma_1 &= \{\texttt{tl}:\forall\texttt{'a. 'a list -> 'a list};\ \texttt{(::)}:\forall\texttt{'a. 'a -> 'a list -> 'a list};\\
&\qquad \texttt{(=)}:\forall\texttt{'a. 'a -> 'a -> bool}\}\\
\Gamma_2 &= \{\texttt{length}:\texttt{'a list -> int};\ \texttt{tl}:\forall\texttt{'a. 'a list -> 'a list};\\
&\qquad \texttt{(::)}:\forall\texttt{'a. 'a -> 'a list -> 'a list};\ \texttt{(=)}:\forall\texttt{'a. 'a -> 'a -> bool}\}\\
\Gamma_3 &= \{\texttt{lst}:\texttt{'a list};\ \texttt{length}:\texttt{'a list -> int};\ \texttt{tl}:\forall\texttt{'a. 'a list -> 'a list};\\
&\qquad \texttt{(::)}:\forall\texttt{'a. 'a -> 'a list -> 'a list};\ \texttt{(=)}:\forall\texttt{'a. 'a -> 'a -> bool}\}\\
\Gamma_4 &= \{\texttt{length}:\forall\texttt{'a. 'a list -> int};\ \texttt{tl}:\forall\texttt{'a. 'a list -> 'a list};\\
&\qquad \texttt{(::)}:\forall\texttt{'a. 'a -> 'a list -> 'a list};\ \texttt{(=)}:\forall\texttt{'a. 'a -> 'a -> bool}\}
\end{aligned}$$

Also let $InTree =$

$$
\dfrac{
  \dfrac{\text{where } \texttt{'a} \mapsto \texttt{string list}}{\Gamma_4 \vdash \texttt{length:string list list -> int}}
  \quad
  \dfrac{
    \dfrac{
      \dfrac{\text{where } \texttt{'a} \mapsto \texttt{string list}}{\Gamma_4 \vdash \texttt{(::):string list -> string list list -> string list list}}\;\textsc{Var}
      \quad
      \dfrac{
        \dfrac{
          \dfrac{\text{where } \texttt{'a} \mapsto \texttt{string}}{\Gamma_4 \vdash \texttt{(::) :string -> string list-> string list}}\;\textsc{Var}
          \quad
          \dfrac{}{\Gamma_4 \vdash \texttt{"a"::string}}\;\textsc{Const}
        }{\Gamma_4 \vdash \texttt{((::)"a") :string list -> string list}}\;\textsc{App}
        \quad
        \dfrac{\text{where } \texttt{'a} \mapsto \texttt{string}}{\Gamma_4 \vdash \texttt{[]:string list}}\;\textsc{Const}
      }{\Gamma_4 \vdash \texttt{("a"::[]):string list}}\;\textsc{App}
    }{\Gamma_4 \vdash \texttt{((::)("a"::[])) :string list list -> string list list}}\;\textsc{App}
    \quad
    \dfrac{\text{where } \texttt{'a} \mapsto \texttt{string list}}{\Gamma_4 \vdash \texttt{[]:string list list}}\;\textsc{Const}
  }{\Gamma_4 \vdash \texttt{(("a"::[])::[]) :string list list}}\;\textsc{App}
}{\Gamma_4 \vdash \texttt{length(("a"::[])::[]):int}}\;\textsc{App}
$$

The proof is then:

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{
        \dfrac{
          \dfrac{\text{where } \texttt{'a} \mapsto \texttt{'a list}}{\Gamma_3 \vdash \texttt{(=) 'a list -> 'a list -> bool}}\;\textsc{Var}
          \quad
          \dfrac{}{\Gamma_3 \vdash \texttt{lst :'a list}}\;\textsc{Var}
        }{\Gamma_3 \vdash \texttt{((=)lst) 'a list -> bool}}\;\textsc{App}
        \quad
        \dfrac{\text{where 'a} \mapsto \texttt{'a}}{\Gamma_3 \vdash \texttt{[] :'a list}}\;\textsc{Const}
      }{\Gamma_3 \vdash \texttt{(lst = []):bool}}\;\textsc{App}
      \quad
      \dfrac{}{\Gamma_3 \vdash \texttt{0:int}}\;\textsc{Const}
      \quad
      \dfrac{
        \dfrac{}{\Gamma_3 \vdash \texttt{1:int}}\;\textsc{Const}
        \quad
        \dfrac{
          \dfrac{\text{where } \texttt{'a} \mapsto \texttt{'a}}{\Gamma_3 \vdash \texttt{length :'a list -> int}}\;\textsc{Var}
          \quad
          \dfrac{
            \dfrac{\text{where } \texttt{'a} \mapsto \texttt{'a}}{\Gamma_3 \vdash \texttt{tl :'a list -> 'a list}}\;\textsc{Var}
            \quad
            \dfrac{}{\Gamma_3 \vdash \texttt{lst :'a list}}\;\textsc{Var}
          }{\Gamma_3 \vdash \texttt{tl(lst) :'a list}}\;\textsc{App}
        }{\Gamma_3 \vdash \texttt{length (tl(lst)) :int}}\;\textsc{App}
      }{\Gamma_3 \vdash \texttt{1 + length (tl(lst)) :int}}\;\textsc{PrimOp}
    }{\Gamma_3 \vdash \texttt{(if lst = [] then 0 else 1 + length (tl(lst)):int}}\;\textsc{If}
  }{\Gamma_2 \vdash \texttt{(fun lst -> if lst = [] then 0 else 1 + length (tl(lst)) :'a list -> int}}\;\textsc{Fun}
  \quad
  InTree
}{\Gamma_1 \vdash \texttt{(let rec length = fun lst -> if lst = [] then 0 else 1 + length (tl(lst)) in length (("a" :: []) :: [])):int}}\;\textsc{LetRec}
$$