HW 6 – Polymorphic Type Inference

CS 421 – Sprint 2014

Revision 1.1

Assigned Thursday, February 27, 2014

Due Sunday, March 16, 2014, 11:59 pm

1 Change Log

1.0 Initial Release.

1.1 Additional notes.

2 Turn-In Procedure

This assignment is named `hw6`. Using your favorite tool(s), you should put your solution in a file named `hw6-solution.pdf`. Your answers to the following questions are to be submitted using the svn repository as described in the section `Instruction for Solving and Submitting Assignments` on the web-page: `http://courses.engr.illinois.edu/cs421/sp2014/mps/index.html`

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 functional 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

(32 points) Give a complete type derivation for the following typing judgment.

```
let rec f = fun x -> fun n -> if n <= 0 then [] else x::(f x (n - 1)) in
  (f 3 2, f "a" 4) : int list * string list
```

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.

Note 1 The proof rules provided with the midterm are for monomorphic type derivation only. This assignment requires polymorphic type derivation.

Note 2 According to the course slide 54 from lecture 12, "`::`" (cons), "`[]`" (nil), and "`(,)`" (pair) are polymorphic constants. You may *not* treat them as primitive operations. Treat them as functions instead. You may only treat "+" and "<=" as primitive operations.

Note 3 Please refer the course slides from lecture 12 (slides 46 all the way to the end).