
HW 3 – Type Judgments

CS 421 – Summer 2013

Revision 1.0

Assigned June 13, 2013

Due Jun 21, 2013, 11:59 PM

Extension 48 hours (20% penalty)

1 Change Log

1.0 Initial Release.

2 Turn-In Procedure

Your answers to the following questions are to be submitted electronically via the `handin` script as though an MP.

3 Objectives and Background

The purpose of this HW is to test your understanding of

- how to use typing rules to perform type derivations in simplified OCaml

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 Problem

(25 pts) Give a complete type derivation for the following type judgment:

```
⊢ let id = fun x -> x
  in fun x -> if x > 0 then id [] else (id x) :: [] : int -> int list
```

You should use the *polymorphic* typing rules for `let` and `let rec`. Assume that `::` is a constant with type $\forall\alpha. \alpha \rightarrow \alpha \text{ list} \rightarrow \alpha \text{ list}$ (where $x :: y$ is parsed as $(::) x y$) and `[]` is a constant with type $\forall\alpha. \alpha \text{ list}$. Be sure to check each time you use the `Const` or `Var` rule for a polymorphic constant or variable that the type at which it is checked is an instance of its polymorphic type. As a suggestion on formatting, you may want to name subtrees of the proof and write them out separately. Note that the problem asks you to perform type checking, not type inference.