Homework 1
Type Rules and Types Rule

98-317: Hype for Types

Due: 23 January 2018 at 6:30 PM

1 Introduction

In class, we discussed the basic notation for writing about types. In this homework, your mission, if you choose to accept it, is to translate some plain English descriptions of types into that notation, and then use typing rules to figure out the types of expressions.

This homework is divided into four parts: Required, Useful, Fun, and Completely Unnecessary But Also Fun. You will receive credit for this homework if you turn in something for the "required" portion.

Turning in the Homework: Bring a paper copy of your homework solutions to class on January 23rd.

You *do not* have to typeset your solutions, but of course you may if you want to. A handwritten solution is perfectly acceptable.

2 Required

In the next couple of problems, you'll be translating English descriptions of typing rules into inference rules. We've provided an example solution to give you an idea of what you should write.

Example Task A function, which we denote $\lambda(x : \tau_1)e$ has type, $\tau_1 \to \tau_2$ if e has type τ_2 (assuming x has type τ_1).

Solution: $$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda(x : \tau_1)e : \tau_1 \to \tau_2}$$

LaTeX If you're interested in typesetting your solutions, the LaTeX for the above solution is provided below:

```
\usepackage{proof}

$\infer[]{\Gamma \vdash \lambda(x:\tau_1)e : \tau_1 \rightarrow \tau_2}
{\Gamma, x:\tau \vdash e : \tau_2}$
```

Req Task 1 A tuple, which we denote "(x, y)" has type $\tau_1 \times \tau_2$ if x has type τ_1 and y has type τ_2. Write an inference rule describing the type of a tuple.

Req Task 2 An if expression, denoted "`if b then` e_1 `else` e_2" has type τ if b has type bool, e_1 has type τ, and e_2 has type τ. Write an inference rule describing the type of an if expression.

Req Task 3 An option case expression, denoted

`case opt of NONE =>` e_1 | `SOME x =>` e_2

has type τ if e_2 has type τ (assuming x has type τ_1), e_1 has type τ, and opt has type τ_1 option. Write an inference rule describing the type of an option case expression.

3 Useful

Tasks in this section and below are completely optional. You do not have to do these tasks to get credit for the homework.

In the next couple of problems, you will use the following set of inference rules to determine the types of a few expressions.[1]

$$\frac{}{\Gamma, x : \tau \vdash x : \tau} \qquad \frac{}{\Gamma \vdash \texttt{num[n]} : \texttt{nat}}$$

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda(\texttt{x} : \tau_1)\ \texttt{e} : \tau_1 \to \tau_2} \qquad \frac{\Gamma \vdash e_1 : \tau_1 \to \tau_2 \quad \Gamma \vdash e_1 : \tau_1}{\Gamma \vdash \texttt{e}_1(\texttt{e}_2) : \tau}$$

$$\frac{\Gamma \vdash \texttt{e} : \tau_1}{\Gamma \vdash \texttt{inl}[\tau_1 + \tau_2](\texttt{e}) : \tau_1 + \tau_2} \qquad \frac{\Gamma \vdash \texttt{e} : \tau_2}{\Gamma \vdash \texttt{inr}[\tau_1 + \tau_2](\texttt{e}) : \tau_1 + \tau_2}$$

$$\frac{\Gamma \vdash e : \tau_1 + \tau_2 \quad \Gamma, x : \tau_1 \vdash e_1 : \tau \quad y : \tau_2 \vdash e_2 : \tau}{\Gamma \vdash \texttt{case e of inl(x)}\ \Rightarrow\ \texttt{e}_1 \mid \texttt{inr(y)}\ \Rightarrow\ \texttt{e}_2 : \tau}$$

Useful Task 1 $\lambda(\texttt{x} : \texttt{nat})\texttt{x}$

Useful Task 2 $\texttt{inl}[\texttt{nat} \to \texttt{nat} + \texttt{nat}](\lambda(\texttt{x} : \texttt{nat})\texttt{x})$

Useful Task 3

```
case (inl[(nat → nat) + nat](λ(x : nat)x)) of
    inl(x) ⇒ x (num[10])
  | inr(y) ⇒ y
```

[1] Don't worry if you've never seen some of these types. The purpose here is to be able to read and understand the rules. Plus, we'll be going over all the ones you don't recognize next class!

4 Fun

Fun Task 1 Using the rules in the useful section, write an expression of type $(\texttt{nat} \to \texttt{nat}) \to \texttt{nat} \to (\texttt{nat} + (\texttt{nat} + \texttt{nat}))$

Fun Task 2 What's the type of the following expression? Or, if the expression does not have a type, explain why.

$$\lambda(f)(\lambda(x)f(x(x)))(\lambda(x)f(x(x)))$$

We've left off the types to make it tricker!

5 Completely Unnecessary but Also Fun

Unnecessary Task 1 What's your favorite type?

Unnecessary Task 2 Are you hype for types?