

Programming Languages and Types

Homework 12

Yi Dai

February 4, 2013

1 Simply-Typed λ -Calculus

1.1 Typing Derivation

Tell whether each of the following terms in the simply-typed λ -calculus with all the extensions introduced in the lecture is well-typed in the empty typing context. If it is, give a typing derivation for it; if not, give the reason. For very large terms, you can name their sub-terms and type them individually.

1. **pred (succ false)**
2. $\lambda f : \mathbf{Nat} \rightarrow \mathbf{Nat} . \lambda n : \mathbf{Nat} . f \ (f \ (\mathbf{succ} \ n))$
3. **if (iszero (succ 0)) then true else 0**
4. $\{one = \mathbf{succ} \ 0, tru = \mathbf{true}\} \text{ as } \{tru : \mathbf{Bool}, one : \mathbf{Nat}\}$
5. **let $b = \mathbf{false}$ in (iszero b)**
6. **let $p = (0, \mathbf{succ} \ 0)$ in (snd p , fst p)**
7. **case (inl 0) of inl $x \Rightarrow \mathbf{false}$ | inr $x \Rightarrow \mathbf{true}$**

8.

```
fix ( $\lambda$  fise : Nat  $\rightarrow$  Bool .  
   $\lambda$  n : Nat .  
    if (iszero n)  
      then true  
      else if (iszero (pred n))  
        then false  
        else fise (pred (pred n)) )
```

1.2 Programming with Extensions

1. Complete the addition function $add : \mathbf{Nat} \rightarrow \mathbf{Nat} \rightarrow \mathbf{Nat}$ in the simply-typed λ -calculus with base type **Nat** and extension the fixed-point operator **fix**.

$$add = \mathbf{fix} (\lambda fadd : \mathbf{Nat} \rightarrow \mathbf{Nat} \rightarrow \mathbf{Nat} . ?)$$

2 System- \mathcal{F}

2.1 Parametric Polymorphism

2.2 Typing Church-Encodings