

CMSC 330

Homework 2

Due: November 5, 2008 in discussion section

Lambda Calculus Problems

0. Here is the ambiguous grammar for lambda calculus given in class:

$$E \rightarrow x \mid \lambda x . E \mid E E$$

where we use x to stand for any variable. The two rules for parsing with this (ambiguous) grammar are:

1. Application is left associative. So “ $e_1 e_2 e_3$ ” is parsed as “ $(e_1 e_2) e_3$ ”
2. Lambda extends as far to the right as possible. So “ $\lambda x . e_1 e_2$ ” is parsed as “ $\lambda x . (e_1 e_2)$ ”

Bearing that in mind, draw parse trees for the following lambda terms:

- (a) $\lambda x . x z \lambda y . x y$
- (b) $(\lambda x . x z) \lambda y . w \lambda w . w y z x$
- (c) $\lambda x . x y \lambda x . y x$

If you have any doubt about how to parse these terms, please come to office hours, since misunderstandings this will likely cause you to make mistakes on the rest of the homework.

1. Find the set of free variables in the following expressions :

- (a) $\lambda x . x z \lambda y . x y$
- (b) $(\lambda x . x z) \lambda y . w \lambda w . w y z x$
- (c) $\lambda x . x y \lambda x . y x$

2. Apply β -reduction to each of the following λ -expressions until no more beta reductions can be applied:

- (a) $(\lambda x . x x)(\lambda y . y x) z$
- (b) $(\lambda x . (\lambda y . (x y)) y) z$
- (c) $((\lambda x . x x)(\lambda y . y))(\lambda y . y)$
- (d) $((\lambda x . \lambda y . (x y)) (\lambda y . y)) w$

3. Consider the λ -expression $(\lambda x . y)((\lambda y . y y y)(\lambda x . x x x))$

- (a) Show that this expression has multiple reduction sequences, some infinite and some finite.
- (b) What is the reduced value of a finite reduction sequence? (All finite reduction sequences have the same result.)

4. Here are three important *combinators*, which are lambda terms with no free variables:

$$I = \lambda x . x$$

$$K = \lambda x . \lambda y . x$$

$$S = \lambda x . \lambda y . \lambda z . (x z (y z)).$$

I is the identity function and K is the function of two arguments that ignores its second argument and returns its first. It can be shown that any combinator can be generated from S and K using only function application.

- (a) Show that the expression SKK reduces to the identity function I .
- (b) Show the result of applying beta reduction to the expression $(S (K S) K) x y z$ until no more beta reductions can be applied.

5. Using the definitions of **true**, **false**, and **or** given in class (see slides), show that **or** behaves correctly as a logical “or” function.