

CS357 Homework 3

Due Monday, Dec. 12

As this assignment is due very shortly before we will need to turn in grades, please do not be late. You can either email me (Aiken) your solutions or drop them at my office, 411 Gates. Because everyone is working on projects, there is no programming on this assignment.

Type Inference with Set Constraints

Consider the following type inference rules for the lambda calculus.

$$A, x : t \vdash x : t$$
$$A, x : t \vdash e : t'$$
$$\frac{}{A \vdash \lambda x. e : t \rightarrow t'}$$
$$A \vdash e : t$$
$$A \vdash e' : t'$$
$$t \leq a \rightarrow b \text{ (where } a \text{ and } b \text{ are fresh variables)}$$
$$t' \leq a$$
$$\frac{}{A \vdash e e' : b}$$

Note that the first two rules are just the standard ones for the simply typed lambda calculus; only the third rule is different. Here \leq is ASCII for subset, \vdash is turnstile, and λx is pronounced “lambda x”.

A type is a type and associated constraints “ t where C ”, defined by the following grammar:

$$t := a \mid t \rightarrow t \mid 0 \mid 1$$
$$C := t_1 \leq t_1' \ \& \ \dots \ \& \ t_n \leq t_n'$$

where “ a ” stands for a family of type variables and “ $\&$ ” is conjunction of constraints. Note that types don’t have unique representations in general. For example

a

and

a where $0 \leq 1$

are equivalent. The meaning of a type “t where C” is

$\{S \mid S \text{ is a solution of } C\}$

As a consequence, any type with inconsistent constraints (constraints with no solutions) is equal to 1.

1. Every term in a type *t where C* can be labeled as *positive* or *negative*. The term *t* and all lower bounds in *C* are positive; all upper bounds in *C* are negative. In a function type $f = t_1 \rightarrow t_2$, if *f* is positive then t_1 is negative and t_2 is positive, and if *f* is negative then t_1 is positive and t_2 is negative. Prove that if a variable *x* appears only in positive positions then it can be replaced by 0 without changing the meaning of the type. (A symmetric argument shows that a variable that appears only in negative positions can be replaced by 1.)

2. Give types for the following lambda terms. You should simplify the constraints in your solutions as much as you can, including applying the result of problem 1 to remove variables. Try to give the most natural types you can, which will generally have the smallest number of constraints and variables.

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

Context Sensitive Analysis

3. Design a fully context-sensitive closure analysis for the lambda calculus using the techniques in lecture 20 (these lecture notes are available before that class in case you want to try to do this problem early). Your answer should be in the form of a set of inference rules, one for each kind of lambda expression, a list of the constructors that you use, and a description of how your solution works.

4. Show the result of applying your analysis from question 3 to the program

$((\lambda x.\lambda f. f (f x)) (\lambda y.y)) (\lambda a.\lambda b.b)$