CS 565, Jens Palsberg May 2, 2001

Final Exam + Qualifying Exam

Each of the four questions is worth five points, for a total of 20 points. Write your name and id number on the first page you submit. Write solutions for different questions on different sheets of paper. Don't submit this handout.

1. [Pointer Analysis] Here is an extension of the λ-calculus:

$$e ::= x \mid \lambda x.e \mid e_1 e_2 \mid \mathsf{malloc}() \mid l$$

We use x to range over identifiers, and we use l to range over heap addresses. We extend the usual small-step call-by-value semantics with the rule:

$$\mathsf{malloc}() \rightarrow l \qquad \text{where } l \text{ is fresh.}$$

The idea is that malloc() can be used to allocate one word of heap storage, and that l will be the address of that newly allocated storage.

Suppose we have a program E and a subexpression e of E, and that we want to know whether any two of the free variables of e can refer to the *same* words of storage. Present a static analysis which gives a conservative answer to this question. Here "conservative" means that the analysis should answer either "no" or "don't know."

2. [CPS] Consider the following transformation $[\![\cdot]\!]$ of simply-typed λ-terms:

$$\begin{array}{rcl} \text{Types:} & & t ::= t \rightarrow t \mid \mathsf{int} \\ [\![x]\!] & = & x \\ [\![\lambda x.e]\!] & = & \lambda k.(k(\lambda x.[\![e]\!])) \\ [\![e_1 e_2]\!] & = & \lambda k.([\![e_1]\!](\lambda v.((v[\![e_2]\!])k))) \end{array}$$

Let o be a type of answers. Define t^* inductively:

$$\begin{array}{rcl} \mathsf{int}^* & = & \mathsf{int} \\ (s \rightarrow t)^* & = & ((s^* \rightarrow o) \rightarrow o) \rightarrow ((t^* \rightarrow o) \rightarrow o) \end{array}$$

Define $A^*(x) = (t^* \rightarrow o) \rightarrow o$ if $A(x) = t$.

Prove the following result in reasonable detail.

Theorem If $A \vdash e : t$, then $A^* \vdash [\![e]\!] : (t^* \rightarrow o) \rightarrow o$.

3. [Fast Control-Flow Analysis] Consider the following lambda-term:

$$G = (\lambda^1 f.((f\,f)(\lambda^2 x.x)))(\lambda^3 y.y)$$

Show the graph for G that is used by the Heintze/McAllester $O(n^2)$ time flow analysis algorithm. What is the label set for G produced by the algorithm?

4. [Subtyping] Here is a grammar for a λ-calculus. We use c to range over integer constants.

$$\begin{array}{rcl} e & ::= & x \mid \lambda x.e \mid e_1 e_2 \mid c \\ t & ::= & t \to t \mid \mathsf{int} \mid \top. \end{array}$$

The semantics is the usual small-step, call-by-value semantics. Consider the following type rules:

$$A \vdash x : t \quad (A(x) = t) \tag{1}$$

$$\frac{A[x : s] \vdash e : t}{A \vdash \lambda x.e : s \to t} \tag{2}$$

$$\frac{A \vdash e_1 : s \to t \quad A \vdash e_2 : s}{A \vdash e_1 e_2 : t} \tag{3}$$

$$A \vdash c : \mathsf{int} \tag{4}$$

$$\frac{A \vdash e : s \quad s \leq t}{A \vdash e : t} \tag{5}$$

A partial ordering $\leq$ of the types is defined by the two rules:

$$t \leq \top \text{ , for all } t \qquad\qquad \frac{s \leq s' \quad t \leq t'}{s \to t \leq s' \to t'}$$

Notice the subtyping rule for function types; it is unusual and problematic. As a consequence of that rule, type soundness does not hold: there is at least one program that type checks but which *can* go wrong. Present such a program, together with a type derivation, and argue that it can go wrong.