CS 565, Jens Palsberg May 6, 1999

Final Exam + Qualifying Exam

Submit your solutions on separate paper (don't submit this handout). Write your name and id number on the top of the first page you submit. I will stable your pages together. This exam is intended to take you 2 hours. Budget your time wisely. There are 4 questions on the exam. Each question is worth 5 points, for a total of 20 points maximum. Write the solutions for different questions on different sheets of paper. Make clear what problem each solution is for. I have spare paper, just ask for it.

To those who are taking the qualifying exam: to pass the exam you must get an excellent score.

1. [**Simple Types**] Consider the language of λ-terms generated by the grammar

$$e ::= x \mid \lambda x.e \mid e_1 e_2 \mid \texttt{succ}\ e \mid c$$

where `succ` denotes the successor function on integers, and c ranges over integer constants. Let us equip this language with a standard simple type system, where the types are generated by the grammar:

$$t ::= \texttt{int} \mid t \to t.$$

Present an algorithm which given any type t will produce a closed λ-term which has that type t and no other type. (Note: if the given type is

$$\texttt{int} \to \texttt{int},$$

then $\lambda x.x$ is *not* a correct output for the algorithm because $\lambda x.x$ also has type

$$(\texttt{int} \to \texttt{int}) \to (\texttt{int} \to \texttt{int}),$$

and many other types.)

2. [**Flow Analysis**] Let E be the λ-term $(\lambda^1 x.xx)(\lambda^2 y.yy)$. What is the flow set for E if we execute the $O(n^3)$ time flow analysis algorithm on E? What is the flow set for E if we execute the Heintze/McAllester flow analysis algorithm on E? Justify your answers.

3. [Security] Here is a variant of Scheme with two security primitives:

$$e \ ::= \ x \mid \texttt{(lambda (}x\texttt{) } e\texttt{)} \mid \texttt{(} e_1\, e_2 \texttt{)} \mid c \mid \texttt{(display } e\texttt{)} \mid$$
$$\texttt{(encrypt } e\texttt{)} \mid \texttt{(decrypt } e\texttt{)}$$

The result of evaluating `(display` e`)` is unspecified. We let c range over (plaintext) integer constants. It is possible to `encrypt` something which is already encrypted. We have that `encrypt` and `decrypt` satisfy the equation:

$$\texttt{(decrypt (encrypt } e\texttt{))} = e.$$

Suggest a type system which guarantees that `display` is only applied to expressions that evaluate to unencrypted (plaintext) integers. Present a type derivation for the following expression:

```
(((lambda (f) (lambda (x) (f (decrypt x))))
  (lambda (y) (display y)))  (encrypt 5))
```

4. [Useless Variables] Consider the Scheme program

```
(letrec ((loop (lambda (a bogus j)
                 (if (> j 10)
                     a
                     (loop (+ a j) (+ bogus 2) (+ j 1))))))
  (loop 0 13 0))
```

It produces 55. Notice that the second parameter to `loop` does not contribute to the final outcome of the computation; such parameters are called *useless*. Useless parameters are unlikely to occur in human-produced code, but may be introduced by program transformations. We want to eliminate useless parameters from procedures and eliminate the actual parameters from their call sites. So, we need a static analysis which estimates the set of useless parameters in a program. Present such an analysis for the following subset of Scheme:

$$e ::= x \mid \texttt{(lambda (}x\texttt{) } e\texttt{)} \mid \texttt{(} e_1\, e_2 \texttt{)} \mid c$$

We let c range over integer constants. Your analysis should determine that `y,a` are useless in the expression below. Justify that this is the case.

```
((((lambda (x) (lambda (y) (x y)))
   (lambda (a) (lambda (b) b)))
  5) 6)
```