CS 565, Jens Palsberg May 4, 2000

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. [Effect Analysis] Consider the language of λ-terms generated by the grammar

$$e ::= x \mid \lambda^l x.e \mid e_1 e_2$$

where x ranges over variable names, and l ranges over labels. Present an algorithm which given a closed expression gives a non-trivial, conservative estimate of the set of labels of λ-abstractions which will be called during a call-by-value evaluation of that expression. For example, for the program

$$(\lambda^1 x.x)(\lambda^2 y.y)$$

the best possible result is $\{1\}$ because $\lambda^2 y.y$ is not called during an evaluation.

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

$$G = (\lambda^1 f.(f(\lambda^2 x.xx))f)(\lambda^3 a.a)$$

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?

3. [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 infinite series of simply-typed λ-terms

$$e_1, e_2, \ldots, e_i, \ldots$$

such that the size of e_n is $O(n)$, and the size of the smallest type of e_n is exponential in n.

4. [Reachability] Below is a grammar for classes in a subset of Java called TJ (Tiny Java). Method overloading is not allowed in TJ.

$$\begin{array}{rcl} D & ::= & \texttt{class}\ C\ \texttt{extends}\ C\{M^*\} \\ M & ::= & \texttt{public}\ C\ m(C\ x)\{\texttt{return}\ e;\} \\ e & ::= & x \mid \mathit{this} \mid e.m(e) \mid \texttt{new}\ C() \\ C, m, x & \in & \mathit{Identifier} \end{array}$$

One of the methods is designated as the main method, that is, the place where a computation begins. We want to analyze TJ programs, and because of space constraints, we are only allowed to use a total of two data structures R and S:

$$\begin{array}{rcl} R & : & \text{represents a set of methods} \\ S & : & \text{represents a set of class names.} \end{array}$$

Present a static analysis which when given a TJ program uses R and S to determine a conservative estimate of the set of reachable methods, that is, methods that will be executed during a run starting in the main method. When the analysis terminates, the estimated set of reachable methods should be represented by R. To avoid being overly conservative, the analysis should take as much advantage as possible of static type information and the extends relation among classes.