CS 565, Jens Palsberg — Dec 16, 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. [Flow Analysis] Consider constraints of the forms:

$$\begin{array}{rcl} c & \in & V \\ V & = & V' \\ (c \in V) & \Rightarrow & (V' = V'') \end{array}$$

where V, V', V'' are flow variables ranging over sets of class names, and c is a class name. Use such constraints to specify flow constraints for the following Java expressions and statements:

$$\begin{array}{rcl} e & ::= & x \mid \texttt{this} \mid \texttt{new}\ A\texttt{()} \mid e_1\texttt{.}m\texttt{(}e_2\texttt{)} \\ s & ::= & x \texttt{ = } e \end{array}$$

Your analysis should be at least as conservative as the subset-based $O(n^3)$ time flow analysis. To demonstrate that, show a fragment of a Java program in which there is a particular method call $e_1\texttt{.}m\texttt{(}e_2\texttt{)}$ where the receiver expression e_1 has a strictly bigger flow set with the new flow analysis compared to its flow set with the $O(n^3)$ time flow analysis. Argue in reasonable detail why the two flow sets are different.

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

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

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. [Security] Consider the following language:

$$
\begin{array}{rcl}
p & ::= & d^* \; s^* \\
d & ::= & x : t \\
t & ::= & L \mid H \\
s & ::= & x \texttt{ = } e \mid \texttt{if (}e\texttt{ == 0) } s \texttt{ else } s \\
e & ::= & c \mid x \mid e \texttt{ + } e \\
c & ::= & \mathit{IntegerConstant}
\end{array}
$$

All variables are integer variables which get their initial values from the user when a program execution begins. A variable is declared to be either low security (L) or high security (H). Give type rules that guarantee that a well-typed program will have the non-interference property:

> **Non-interference:** A program has the non-interference property if we can arbitrarily change the initial values of the H variables *without* thereby affecting the final values of the L variables.

Based on your type rules, argue that the following program type checks.

```
x:L   y:H   z:L
x=5
y=x
if (y == 0) y=1 else y=2
if (x == 0) z=x else y=2
```

4. [Java Security] Below is a grammar for classes in a Java-like language called SFJ (Secure Flyweight Java). Method overloading is not allowed in SFJ.

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

The notation $[\alpha]$ indicates that α is optional. Suppose we are given a package of SFJ classes, and that their superclasses also belong to the same package, including Object. Give type rules that guarantee that no object of a confined type will escape the package.