CS 232, Jens Palsberg Jun 12, 2007

Final Exam

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

1. [**Flow analysis**] Consider a core of Java with classes, class extension, fields, methods, statements, and expressions. In a notation of your choice, phrase a simplified version of 0-CFA which (1) has one set variable for each field, (2) has just one set variable for each method, (3) has no separate set variable for each expression, and (4) has the property that the least solution maps the set variable for a field to a set of classes that all are subtypes of the declared type of that field.

2. [**Flow-insensitive points-to analysis**] Consider programs of the form $s_1; s_2; \ldots; s_n$, where each statement s is an assignment of the form $e_1 = e_2$. Each expression e is derived from the grammar

$$e ::= x \mid \&e \mid *e$$

where x is an integer variable. Points-to information is of the form (x, y), where x, y are variables, meaning that the value of x is the address of y. An instance of the flow-insensitive points-to analysis problem is a program P and a points-to fact (x, y), and the problem is to determine whether there is a sequence of some of the statements in P (in any order), the execution of which leads to a state in which (x, y) is true. What is the complexity of the flow-insensitive points-to analysis problem? Justify your answer.

3. [**Heintze-McAllester style flow analysis**] Here are the rules for Heintze-McAllester style flow analysis of a λ-calculus program P:

$$\frac{}{x \rightarrow dom(\lambda^\ell x.e)} \text{ (if } \lambda^\ell x.e \in P) \qquad \frac{}{ran(\lambda^\ell x.e) \rightarrow e} \text{ (if } \lambda^\ell x.e \in P)$$

$$\frac{}{dom(e_1) \rightarrow e_2} \text{ (if } (e_1\ e_2) \in P) \qquad \frac{}{(e_1\ e_2) \rightarrow ran(e_1)} \text{ (if } (e_1\ e_2) \in P)$$

$$\frac{n_1 \rightarrow n_2 \qquad n \rightarrow dom(n_2)}{dom(n_2) \rightarrow dom(n_1)} \qquad \frac{n_1 \rightarrow n_2 \qquad n \rightarrow ran(n_1)}{ran(n_1) \rightarrow ran(n_2)}$$

Phrase a Heintze-McAllester style flow analysis for programs derived from the grammar:

$$e \ ::= \ x \mid \lambda^\ell x.e \mid e_1 e_2 \mid nil \mid cons\ e\ e \mid car\ e \mid cdr\ e$$

where *nil* is the empty list, *cons* e_1 e_2 constructs a list with head e_1 and tail e_2, *car* e returns the head of the list e, and *cdr* e returns the tail of the list e.

4. [**Dead code**] Phrase a whole-program flow analysis for a core of Java, particularly excluding dynamic class loading and reflection, which determines a set of methods, as large as possible, that cannot be called.