2004 Paper 9 Question 3

Optimising Compilers

Assume that a program consists of a sequence of declarations `Object` o`;` where o is an object name, followed by a sequence of function definitions $f(x_1, \ldots, x_k) = e$ where expressions, e, have syntax

$$e ::= n \mid o \mid x \mid f(e_1, \ldots, e_k) \mid \texttt{let}\ x = e_1\ \texttt{in}\ e_2 \mid \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3.$$

where n ranges over integer constants and x over variables (which may contain integers or object *references*). Variables may not contain function values.

Alias Analysis is a technique which will determine that, during evaluation of e within

$$\texttt{let}\ x = o\ \texttt{in}\ \texttt{let}\ y = o\ \texttt{in}\ e$$

x and y alias because they are both references to the same object o.

(a) Show how to associate a flow variable with each variable and (sub-)expression of the program. State the values which flow variables might reasonably take in such an analysis. [4 marks]

(b) Show how, given a program, we can generate a set of constraint-style equations (analogously to control-flow analysis for λ-expressions) whose solution gives a superset of the values which might be returned from each (sub-)expression of the program. [Hint: suppose that each function definition has flow variables representing the value ranges of each of its arguments and of its result.] [8 marks]

(c) Explain what happens in, and give modifications to part (b) for, the generalisation whereby variables can also reference functions and be called by the syntax

$$e ::= e_0(e_1, \ldots, e_k)$$

[4 marks]

(d) Explain how you would respond to the criticism that your analysis may fail to terminate if your language is extended with arithmetic expressions because a single expression may give rise to an infinite set of values. [2 marks]

(e) Briefly describe any optimisation whereby knowing that x and y cannot alias is necessary for the optimisation to be safe. [2 marks]