Optimising Compilers 2005 – Paper 9 Question 7 (AM)
Solution Notes

[This is a question on strictness analysis (Section 11 of the notes).]

(a) Bookwork, main points:

- Apply to lazy languages, to determine when a sub-calculation fails to terminate implies the whole calculation fails to terminate.
- Lectures restrict to first-order functions; so will we. Given a fn $f : D^k \to D$ we calculate a strictness fn $f^\sharp : 2^k \to 2$ where $2 = \{0, 1\}$.
- For built-in functions we pre-calculate using

$$\begin{array}{rcl} a^\sharp(x_1, \ldots, x_r) & = & 0 \text{ if } (\forall d_1, \ldots, d_r \in D s.t.\ (x_i = 0 \Rightarrow d_i = \perp)\ a(d_1, \ldots, d_r) = \perp \\ & = & 1 \text{ otherwise} \end{array}$$

  For user functions we deterine strictness fns $f^\sharp$ in terms of the same composition and recursion as the definition of f.
- If $f^\sharp(1, \ldots, 1, 0, 1, \ldots 1) = 0$ then f is strict in its ith argument so we can optimise f to calculate its ith argument before calling f with no change of semantics, i.e. change CBN to CBV.

(b)

(i) $f^\sharp(x) = 1$

(ii) $g^\sharp(x) = 0$

(iii) $h^\sharp(y, z) = y \vee z$ (because we've only used the strictness property of f, not its definition).

(iv) $k^\sharp(x, y, z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z) = (x \vee y) \wedge (x \vee z) \wedge (y \vee z)$

(c) One construction is as follows. Given *be* put it in DNF $t_1 \vee \cdots \vee t_n$. Suppose $t_i = (v_{i1} \wedge \cdots \wedge v_{im_i})$ then put $e_i = v_{i1} + \cdots + v_{im_i}$ to make an expr e_i with strictness t_i (as $+$ is strict). Now define

$$\begin{array}{rcl} u(x_1, \ldots, x_k) & = & \\ & & \text{if } f(1) \text{ then } e_1 \text{ else} \\ & & \text{else if } f(2) \text{ then } e_2 \\ & & \cdots \\ & & \text{else } e_n \end{array}$$

using if-then-else to give 'or' (expoiting b(iii)). The smartest students might notice that cases b(i) and b(ii) above represent the cases when the DNF expression degenerates into is 1 or 0 and hence isn't really covered by the the main bit of the answer to part (c)!