Optimising Compilers

Briefly summarize the main concepts of strictness analysis including the kind of languages to which it applies, and the way in which both system-provided and user-defined functions \( f \) yield strictness properties \( f^\# \) (relate the types of \( f \) and \( f^\# \)). [6 marks]

Give the strictness functions corresponding to the following ternary functions:

\[(a) \quad f_1(x,y,z) = x\cdot y + z\]
\[(b) \quad f_2(x,y,z) = \text{if } x=9 \text{ then } y \text{ else } z\]
\[(c) \quad f_3(x,y,z) = \text{pif } x=9 \text{ then } y \text{ else } z\]

where \( \text{pif } e_1 \text{ then } e_2 \text{ else } e_3 \) is the parallel conditional: it behaves similarly to the standard conditional in that if \( e_1 \) evaluates to \( \text{true} \) or \( \text{false} \) then it yields \( e_2 \) or \( e_3 \) as appropriate; however, evaluation of \( e_2 \) and \( e_3 \) occurs concurrently with \( e_1 \) to allow the \( \text{pif} \) construct also to terminate with the value of \( e_2 \) when \( e_2 \) and \( e_3 \) both terminate with equal values (even if \( e_1 \) computes forever).

Comment briefly how your strictness property for \( f_1 \) would change if the multiplication returned zero without evaluating the other argument in the event that one argument were zero. [7 marks]

Let \( g, h_1 \) and \( h_2 \) be binary functions and recall the definition of function composition:

\[g \circ (h_1, h_2) = \lambda(x,y).g(h_1(x,y), h_2(x,y)).\]

Define three such functions in an ML-like syntax (whose arguments and results are integers) and which have the property that

\[(g \circ (h_1, h_2))^\# \neq g^\# \circ (h_1^\#, h_2^\#).\]

[Hint: you might find it helpful to think of a solution where \( g \) may ignore one of its arguments but \( always \ does \) when composed with \( (h_1, h_2) \).] Comment whether this inequality means that \( g^\# \circ (h_1^\#, h_2^\#) \) fails to be a safe strictness property for \( g \circ (h_1, h_2) \). [7 marks]