COMPUTER SCIENCE TRIPOS Part II – 2012 – Paper 9

9 Optimising Compilers (AM)

(*a*) Explain the core ideas of strictness analysis, including the abstract values used for abstracting non-function values and what concrete values they represent. Briefly explain how program functions f are abstracted to strictness functions $f^\sharp$. Give the abstractions of `λ(x,y).x+y` and `λ(x,y). if random() then x else y`. [5 marks]

(*b*) Justify or correct the following statements: (*i*) "since abstract interpretation replaces real-world computation with a directly corresponding abstract computation then strictness analysis fails to terminate on non-terminating programs"; and (*ii*) "when a strict function is applied to an expression e then e is necessarily evaluated during the call". [4 marks]

(*c*) We now wish to extend strictness analysis from simple `int` expressions to allow also (lazy) `int list` expressions. These represent lists whose head and tail components are only evaluated when required. Wadler suggested capturing strictness-like properties on lazy lists using an abstract interpretation with four abstract values for `int list` concrete values:

0: non-termination

∞: a chain of cons cells, either infinite or having some tail component which does not terminate

$0\in$: a chain of cons cells ending in `nil` but having at least one member which does not terminate

$1\in$: a possibly empty chain of cons cells ending in `nil` every member of which terminates

By analogy with ordinary strictness functions, give abstract interpretations in truth-table form (noting that values of type `int list` have four values rather than the standard two) for the following functions involving lazy list values:

(*i*) `λ(x:int list). nil` [1 mark]

(*ii*) `λ(x:int list). cons(42,x)` [1 mark]

(*iii*) `λ(x,y:int list). if random() then x else y`
Explain how you resolved any choice which arose. [3 marks]

(*iv*) `hd` [2 marks]

(*v*) `tl` [1 mark]

(*vi*) `append` [2 marks]

(*vii*) `reverse` [1 mark]