Haskell Exercises 7: Tuples

Antoni Diller

26 July 2011

(1) An association list is a list of 2-tuples. For example, `[("temp", 34), ("height", 80), ("weight", 180), ("depth", 7)]`. Define a function *domain* :: $Eq\ a \Rightarrow [(a, b)] \to [a]$ which takes an association list and returns the list of all those things that occur in the first component of each tuple. Make sure that the value of *domain* does not contain any duplicates.

(2) Define a function *range* :: $Eq\ a \Rightarrow [(a, b)] \to [b]$ which takes an association list and returns the list of all those things that occur in the second component of each tuple. Make sure that the value of *range* does not contain any duplicates.

(3) Define a function *compose* :: $[(a, b)] \to [(b, c)] \to [(a, c)]$ such that a tuple (x, z) is in the list returned as the value of the function *compose ass1 ass2* iff (x, y) is in *ass1* and (y, z) is in *ass2*. For example, `compose [(1, 2), (7, 11)] [(2, 3), (11, 14)]` is `[(1, 3), (7, 14)]`.

(4) Define a function *inverse* :: $[(a, b)] \to [(b, a)]$ such that a tuple (y, x) is in *inverse ass* iff (x, y) is in *ass*.

(5) A *homogeneous* association list is one whose tuples contain elements belonging to the same type. Define a function *reflexive* :: $Eq\ a \Rightarrow [(a, a)] \to Bool$ which tests to see if a homogeneous association list is reflexive, that is to say, if either (x, y) or (y, x) is in *ass*, then (x, x) is also in *ass*.

(6) Define a function *symmetric* :: $Eq\ a \Rightarrow [(a, a)] \to Bool$ which tests to see if a homogeneous association list is symmetric, that is to say, if (x, y) is in *ass*, then so is (y, x).

(7) Define a function *transitive* :: $Eq\ a \Rightarrow [(a, a)] \to Bool$ which tests to see if a homogeneous association list *ass* is transitive, that is to say, if (x, y) and (y, z) are in *ass*, then so is (x, z).

(8) Define a function *closure* :: $[(a, a)] \to [(a, a)]$ which takes an arbitrary association list *ass* and produces the reflexive, transitive closure of *ass*, that is to say, if (x, y) and (y, z) are in *ass*, then (x, y), (y, z) and (x, z) are all in *closure ass*.

(9) Define the function *pairs* such that *pairs i* is the list of all distinct pairs of integers (x, y) such that $1 \leq x, y \leq i$. For example,

$$\textit{pairs}\ 3 = [(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)].$$

(10) Using the function *zip* define the infinite list *factlist* of factorials.

(11) A curried function f of n Boolean arguments is called tautologous if it returns *True* for every one of the 2^n possible combinations of Boolean arguments. Write a function *taut* so that *taut* n f is *True* is and only if f is tautologous.