Tutorial 7: Trees and Mutually Recursive Data

CS 135 Fall 2007

October 24-26, 2007

Today's tutorial covers topics from lecutre module 7 on trees and mutually recursive data definitions. One area we're not touching today is binary search trees — these should be pretty well covered by your assignment, but the general techniques we use here will apply for those structures as well. As usual, the starter code (t7-starter.scm) is, well, a good place to start.

1 Family Trees

The data definition for family trees (`ft`'s) is the first one presented in lecture module 7. It is repeated in the starter code. Here we'll write some useful functions for family trees.

1.1 Drawing the tree

The code given below (also in the starter code) defines a simple family tree that will later be useful for testing our functions. First, draw the family three represented below with box-and-arrows notation. Which node is the 'root'?

```
(define patrick1 (make-child empty empty 'Patrick 1928 'brown))
(define emma (make-child empty empty 'Emma 1933 'brown))
(define patrick2 (make-child patrick1 emma 'Patrick 1956 'brown))
(define mary (make-child empty empty 'Mary 1908 'green))
(define doris (make-child empty mary 'Doris 1935 'blue))
(define bob (make-child empty empty 'Bob 1930 'blue))
(define susan (make-child bob doris 'Susan 1956 'blue))
(define emily (make-child patrick2 susan 'Emily 1987 'green))
```

1.2 Counting

Write a function `ancestors-named` which consumes a family tree and a symbol for a name and produces the number of ancestors in the tree with the given name.

1.3 Mothers' Names

Write a function `mother-names` that produces a list of the names of everyone in a given family tree that is a mother (with duplicates).

1.4 We don't mean radioactive...

Some sociologists use the term 'nuclear family' to refer to a family with some children and exactly two parents. For our purposes, we'll say a family tree represents a nuclear family iff all children in the tree have exactly two or zero parents.

Write a predicate function `nuclear-family?` which returns true iff the given ft represents a nuclear family.

1.5 Respect your elders!

Note that each `child` structure contains a field for the year the child was born. Use this information to write a function `oldest-ancestor` which consumes an ft ***which is not empty*** and produces a `child` node corresponding to the oldest ancestor in the tree. If there is a tie, return any one of the oldest.

2 Food, Glorious Food

This half of the tutorial will make use of a new mutually-recursive data definition, which is in the starter code and repeated here for convenience:

An expression of type `food` is of the form `(make-food nme veg ilst)`, where `nme` is a string for the name of the food, `veg` is a boolean (true iff the food is vegetarian), and `ilst` is an ingredients-list.

An ingredients-list (or 'il') is either `empty` or `(cons rat (cons f ilst))`, where `rat` is a number (for the ratio of that ingredient), `f` is a food, and `ilst` is an ingredients-list.

2.1 Template

Write a template for a function that consumes an expression of type *food*.

2.2 Contains?

Write a function `contains-food?`which consumes a string (for the name of a food) and a food, and returns true iff the given food contains an ingredient (or sub-ingredient etc.) with the given name.

2.3 Vegeterian check

Notice that each food has a field which indicates whether it is vegetarian. But suppose we want to be really sure that a food contains no non-vegetarian products. To accomplish this, write a function `veg-check?` which consumes a food and produces true iff the food and all foods which it contains are all vegetarian.

2.4 Ingredient weight

Write a function `ingredient-weight` which consumes a food, the total weight of that food, and the name of a single ingredient, and produces the weight of that ingredient in the original food.

2.5 Almost vegetarian

If one per cent or less of a food is not vegetarian, then probably no one can tell. In this case, we'll call the food 'almost vegetarian'. Write a function `almost-veg?` which consumes a food and produces true iff at least 99 per cent of that food is vegetarian.