Trees

CS 5010 Program Design Paradigms "Bootcamp"

Lesson 6.2

Introduction/Outline

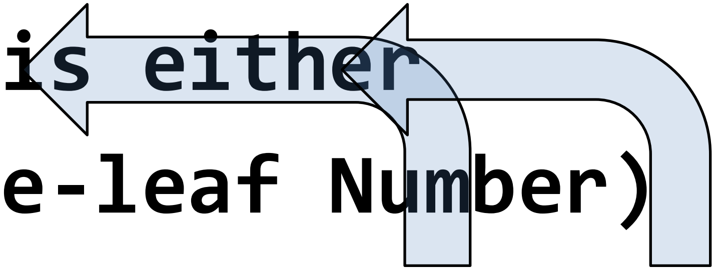
- We've now learned about two ways to represent sequence information.
- Many examples of information have a natural branching structure.
- These are represented as *trees*, which you should have learned about in your data structures course.
- In this lesson, we'll study how to apply the Design Recipe to trees.

Learning Objectives

- At the end of this lesson you should be able to:
  - Write a data definition for tree-structured information
  - Write a template for tree-structured information
  - Write functions that manipulate that data, using the template

Binary Trees

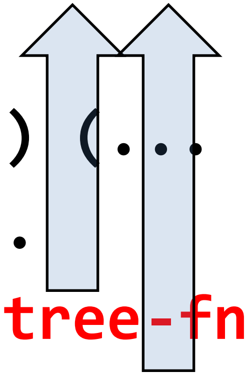
```
(define-struct leaf (datum))
(define-struct node (lson rson))

;; A Tree is either
;; -- (make-leaf Number)
;; -- (make-node Tree Tree)
```



There are many ways to define binary trees. We choose this one because it is clear and simple.

Template

```
tree-fn : Tree -> ???
(define (tree-fn t)
  (cond
    [(leaf? t) (... (leaf-datum t))]
    [else (...
            (tree-fn (node-lson t))
            (tree-fn (node-rson t)))]))
```



Here's the template for this data definition. Observe that we have two self-references in the template, corresponding to the two self-references in the data definition.

Self-reference in the data definition leads to self-reference in the template; Self-reference in the template leads to self-reference in the code.

The template questions

```
tree-fn : Tree -> ???
(define (tree-fn t)
  (cond
    [(leaf? t) (... (leaf-datum t))]
    [else (...
            (tree-fn (node-lson t))
            (tree-fn (node-rson t) ))]))
```

What's the answer for a leaf?

If you knew the answers for the 2 sons, how could you find the answer for the whole tree?

And here are the template questions. When we write a function using the template, we fill in the template with the answers to these questions.

The template recipe

Question	Answer
Does the data definition distinguish among different subclasses of data?	Your template needs as many cond clauses as subclasses that the data definition distinguishes.
How do the subclasses differ from each other?	Use the differences to formulate a condition per clause.
Do any of the clauses deal with structured values?	If so, add appropriate selector expressions to the clause.
Does the data definition use self-references?	Formulate ``natural recursions'' for the template to represent the self-references of the data definition.
Do any of the fields contain compound or mixed data?	If the value of a field is a foo, add a call to a foo-fn to use it.

The template recipe doesn't need to change

Next we'll do some examples of functions on binary trees.

leaf-sum

What's the answer for a leaf?

```
leaf-sum : Tree -> Number
(define (leaf-sum t)
  (cond
    [(leaf? t) (leaf-datum t)]
    [else (+
            (leaf-sum (node-lson t))
            (leaf-sum (node-rson t)))]))
```

If you knew the answers for the 2 sons, how could you find the answer for the whole tree?

leaf-max

```
leaf-max : Tree -> Number
(define (leaf-max t)
  (cond
    [(leaf? t) (leaf-datum t)]
    [else (max
            (leaf-max (node-lson t))
            (leaf-max (node-rson t)))]))
```

What's the answer for a leaf?

If you knew the answers for the 2 sons, how could you find the answer for the whole tree?

leaf-min

```
leaf-min : Tree -> Number
(define (leaf-min t)
  (cond
    [(leaf? t) (leaf-datum t)]
    [else (min
            (leaf-min (node-lson t))
            (leaf-min (node-rson t)))]))
```

What's the answer for a leaf?

If you knew the answers for the 2 sons, how could you find the answer for the whole tree?

Summary

- You should now be able to:
  - Write a data definition for tree-structured information
  - Write a template for tree-structured information
  - Write functions that manipulate that data, using the template

Next Steps

- Study the file 06-2-trees.rkt in the Examples folder.
- If you have questions about this lesson, ask them on the Discussion Board
- Do Guided Practice 6.2
- Go on to the next lesson