Lists of Structures

CS 5010 Program Design Paradigms "Bootcamp"

Lesson 4.3

Introduction

- Lists of structures occur all the time
- Programming with these is no different:
  - write down the data definition, including interpretation and template
  - Follow the Recipe!

Learning Objectives

- At the end of this lesson you should be able to:
  - write down a template for lists of compound data
  - use the template to write simple functions on lists of compound data

Tweaking the Template Recipe

- Programming with lists of compound data is no different from programming with lists of scalars, except that we make one small change in the recipe for templates

The template recipe, updated

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.

Observe that this is just what we did for self-references, because a list is a kind of mixed data.

Books, again

```
(define-struct book (author title on-hand price))

;; A Book is a
;;  (make-book String String NonNegInt NonNegInt)
;; Interpretation:
;; --author is the author's name
;; --title is the title
;; --on-hand is the number of copies on hand
;; --price is the price in USD*1

;; book-fn : Book -> ??
;; (define (book-fn b)
;;   (... (book-author b)
;;        (book-title b)
;;        (book-on-hand b)
;;        (book-price b)))
```

Here is the data definition for a book in a bookstore, with structure definition, data definition, interpretation, and template.

Notice that the data definition doesn't say WHICH list of books this is. It could be all the books in the bookstore, just the paperbacks, the ones that have been ordered in the last 30 days, etc. etc.

Template for ListofBooks

```
;; A ListOfBooks (LOB) is either
;; -- empty
;; -- (cons Book LOB)

;; lob-fn : LOB -> ??
;; (define (lob-fn lob)
;;   (cond
;;     [(empty? lob) ...]
;;     [else (...
;;             (book-fn (first lob))
;;             (lob-fn (rest lob)))]))
```

When dealing with a list of structures, you should insert a call to a function here.

(rest lob) is a LOB, so we wrap it in a lob-fn.

Similarly, (first lob) is a Book, so we wrap it in a book-fn.

Example: if book-fn is just a selector, you can put it in directly

```
;; books-authors : LOB -> ListOfString
;; STRATEGY: Use template for LOB on lob
(define (books-authors lob)
  (cond
    [(empty? lob) empty]
    [else (cons
           (book-author (first lob))
           (books-authors (rest lob)))]))
```

book-author is certainly a **book-fn**!

Same thing for lists of other non-scalar data

```
;; A ListOfKeyEvents (LOKE) is either
;; -- empty
;; -- (cons KeyEvent LOKE)

;; loke-fn : LOKE -> ??
;; (define (loke-fn loke)
;;   (cond
;;     [(empty? loke) ...]
;;     [else (...
;;             (kev-fn (first loke))
;;             (loke-fn (rest loke)))]))
```

(rest loke) is a LOKE, so we wrap it in a loke-fn.

Similarly, **(first loke)** is a KeyEvent, so we wrap it in a kev-fn.

Module Summary: Self-Referential or Recursive Information

- Represent arbitrary-sized information using a *self-referential* (or *recursive*) 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.
- Writing functions on this kind of data is easy: just Follow The Recipe!
- But get the template right!

Summary

- At the end of this lesson you should be able to:
  - write down a template for lists of compound data
  - use the template to write simple functions on lists of compound data
- The Guided Practices will give you some exercise in doing this.

Next Steps

- Study 04-2-books.rkt in the Examples file
- If you have questions about this lesson, ask them on the Discussion Board
- Do Guided Practice 4.4
- Go on to the next lesson