Module 4b: Reference

CPSC 110

Peyton Seigo

2018-09-30

Module 4b: Reference

Learning goal

- Be able to predict and identify the correspondence between references in a data definition and helper function calls in functions that operate on the data.

Notes

- When a data definition uses the reference rule, the `@dd-template-rules` tag is `ref`

Terminology

- **Reference relationship**: data definition that refers to a different type of data (that's not primitive!)
- **Reference rule**: for data definitions with a reference to another data definitions that you've defined
    - Rule: must wrap calls to referenced definition in that definition's template function (called a **natural helper**)
- **Natural helper**: a referenced data definition's template function due to the reference rule
    - A natural helper in a template says "do something complicated in a helper function that consumes the referred to type. do NOT do it here!"
    - HtDF: create a **helper function** for the natural helper
        * wish list entry: `@HtDF`, `@signature`, purpose, stub, and `!!!`
- **Helper function**: actual function written when doing HtDF
- **complicated? rule**: if it would take more than 1 function that operates on the referenced type, make a helper function instead.

Lists containing non-primitive (user-defined) data

Example of a list data definition containing non-primitive data defined by the user. Pay attention to

- How the self-reference rule applies to `ListOfSchool`
    - the `dd-template-rules` tag is `self-ref`
    - `(rest los)` is non-primitive and is a `ListOfSchool`, so it is wrapped in `ListOfSchool`'s template function
        * this is **Natural Recursion**
        * when writing a function that consumes `ListOfSchool`, this will be a **Recursive Call**
- How the reference rule applies to `ListOfSchool`

- the `dd-template-rules` tag is `ref`
- (`first los`) is non-primitive and is a `School`, so it is wrapped in `School`'s template function
  * this is a **Natural Helper**
  * when writing a function that consumes `ListOfSchool`, you must write a **Helper Function** (unless you're not performing any operations on that data)
- There are **two** data definitions, not just one

```
1   (@HtDD School)
2   (define-struct school (name tuition))
3   ;; Bar is (make-bar Natural String)
4   ;; interp. properties of a university
5   ;;         name is abbreviated name of the university
6   ;;         tuition is yearly undergraduate tuition (CAD) of the
        university
7   (define S-UBC (make-school "UBC" 25000))
8   (define S-UOA (make-school "UAlberta" 16000))
9   (define S-UOC (make-school "UCalgary" 8500))
10
11  (@dd-template-rules compound)
12  (define (fn-for-school s)
13    (... (school-tuition s)
14         (school-name s)))
15
16
17  (@HtDD ListOfSchool)
18  ;; ListOfSchool is one of:
19  ;;  - empty
20  ;;  - (cons School ListOfSchool)
21  ;; interp. a list of schools
22  (define LOS1 empty)
23  (define LOS2 (cons S-UBC (cons S-UOA (cons S-UOC empty))))
24
25  (@dd-template-rules one-of          ; 2 cases
26                      atomic-distinct ; empty
27                      compound        ; (cons School ListOfSchool)
28                      ref             ; (first los) is School
29                      self-ref)       ; (rest los) is ListOfSchool
30  (define (fn-for-los los)
31    (cond [(empty? los) (...)]
32          [else
```

```
33          (... (fn-for-school (first los))
34              (fn-for-los (rest los)))])) 
```