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 `ListOf-School`'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

```
(@HtDD School)
(define-struct school (name tuition))
;; Bar is (make-bar Natural String)
;; interp. properties of a university
;;         name is abbreviated name of the university
;;         tuition is yearly undergraduate tuition (CAD) of the university
(define S-UBC (make-school "UBC" 25000))
(define S-UOA (make-school "UAlberta" 16000))
(define S-UOC (make-school "UCalgary" 8500))

(@dd-template-rules compound)
(define (fn-for-school s)
  (... (school-tuition s)
       (school-name s)))



(@HtDD ListOfSchool)
;; ListOfSchool is one of:
;;  - empty
;;  - (cons School ListOfSchool)
;; interp. a list of schools
(define LOS1 empty)
(define LOS2 (cons S-UBC (cons S-UOA (cons S-UOC empty))))

(@dd-template-rules one-of          ; 2 cases
                    atomic-distinct ; empty
                    compound        ; (cons School ListOfSchool)
                    ref             ; (first los) is School
                    self-ref)       ; (rest los) is ListOfSchool
(define (fn-for-los los)
  (cond [(empty? los) (...)]
        [else
         (... (fn-for-school (first los))
              (fn-for-los (rest los)))]))
```