UMass Boston Computer Science

CS450 High Level Languages (section 2)

More Kinds of Data Definitions

Wednesday, September 20, 2023
Logistics

• HW 1 out
  • due: Sun 9/24 11:59 pm EST

• Course web site:
  • See The Design Recipe section
  • Lecture code (see lecture03.rkt) may occasionally be posted
Design Recipe, Step 1: Data Design

Create **Data Definitions**

- Describes the **types of data** that the program operates on
- Has 4 parts:
  1. **Name**
  2. Description of all possible values of the data
  3. **Interpretation** explaining the real world concepts the data represents
  4. **Predicate** returning true if the given value is in the data definition
Kinds of Data Definitions

- Basic data
- Intervals
- Enumerations
- Itemizations
Interval Data Definitions

;; An AngleD is a number in \([0, 360)\)
;; interp: An angle in degrees
(define (AngleD? deg)
  (and (>= deg 0) (< deg 360)))

;; An AngleR is a number in \([0, 2\pi)\)
;; interp: An angle in radians
(define (AngleR? r)
  (and (>= r 0) (< r (* 2 pi))))

;; deg->rad: AngleD -> AngleR
;; Converts the given angle in degrees to radians
(define (deg->rad deg)
  (* deg (/ pi 180)))

(check-equal? (deg->rad 0) 0)
(check-equal? (deg->rad 90) (/ pi 2))
(check-equal? (deg->rad 180) pi)

It depends! (Data representations are crucial because they determine what the rest of the program looks like)

Function Recipe Steps 1-3: name, signature, description

Step 5: Code

Step 6: Tests

Not allowed by data def, but should be ok?
Kinds of Data Definitions

- Basic data
- Intervals
- Enumerations
- Itemizations
Enumeration Data Definitions

;; A TrafficLight is one of:
;;  - RED-LIGHT
;;  - GREEN-LIGHT
;;  - YELLOW-LIGHT

;; Interpretation: Represents possible colors of a traffic light
(define RED-LIGHT "RED")
(define GREEN-LIGHT "GREEN")
(define YELLOW-LIGHT "YELLOW")

(define (red-light? x) (string=? x RED-LIGHT))
(define (green-light? x) (string=? x GREEN-LIGHT))
(define (yellow-light? x) (string=? x YELLOW-LIGHT))

NOTE: this is not the only possible data definition. Is there a better one?

Need to add an extra step to Data Design Recipe
Design Recipe, Step 1: Data Design

Create **Data Definitions**

- Describes the *types of data* that the program operates on
- Has 4 parts:
  1. **Name**
  2. **Description of all possible values** of the data
  3. **Interpretation** explaining the real world concepts the data represents
  4. **Predicate** returning true if the given value is in the data definition
     • If needed, also define predicates for each *enumeration* or *itemization*
Enumeration Data Definitions

;; A TrafficLight is one of:
;; - RED-LIGHT
;; - GREEN-LIGHT
;; - YELLOW-LIGHT
;; Interpretation: Represents possible colors of a traffic light
(define RED-LIGHT "RED")
(define GREEN-LIGHT "GREEN")
(define YELLOW-LIGHT "YELLOW")

;; next-light: TrafficLight -> TrafficLight
;; Computes the next light after the given one
(define (next-light light)
  (cond
    ;; cond is multi-arm if (expression)
    ; [ ]
    [(red-light? light) GREEN-LIGHT]
    [(green-light? light) YELLOW-LIGHT]
    [(yellow-light? light) RED-LIGHT]))

;; check-equal: TrafficLight TrafficLight -> Boolean
;; Checks if the two lights are equal
(define (check-equal? light1 light2)
  (cond
    ;; cond is multi-arm if (expression)
    ; [ ]
    [(red-light? light1) (red-light? light2)]
    [(green-light? light1) (green-light? light2)]
    [(yellow-light? light1) (yellow-light? light2)]))

The data and function have the same structure!
(keep order the same)

Designing data first makes writing function (code) easier!
Function Design Recipe

1. Name

2. **Signature** – *types* of the function input(s) and output

3. **Description** – *explain* (in English prose) the function behavior

4. **Examples** – *show* (using `rackunit`) the function behavior

5. **Code** – *implement* the rest of the function (arithmetic)

6. **Tests** – *check* (using `rackunit`) the function behavior
Function Design Recipe

1. **Name**
2. **Signature** – *types* of the function input(s) and output
3. **Description** – *explain* (in English prose) the function behavior
4. **Examples** – *show* (using rackunit) the function behavior
5. **Template** – *sketch out* the function structure (using input’s Data Definition)
6. **Code** – *implement* the rest of the function (arithmetic)
7. **Tests** – *check* (using rackunit) the function behavior
Enumeration Data Definitions

;; A TrafficLight is one of:
(define RED-LIGHT "RED")
(define GREEN-LIGHT "GREEN")
(define YELLOW-LIGHT "YELLOW")

;; Interpretation: Represents possible colors of a traffic light
(define (red-light? x) (string=? x RED-LIGHT))
(define (green-light? x) (string=? x GREEN-LIGHT))
(define (yellow-light? x) (string=? x YELLOW-LIGHT))

;; next-light: TrafficLight -> TrafficLight
;; Computes the next light after the given one
(define (next-light light)
  (cond
    [(red-light? light) ....]
    [(green-light? light) ....]
    [(yellow-light? light) ....]))
Kinds of Data Definitions

• Basic data
• Intervals
• Enumerations
• Itemizations
Itemization Data Definitions

The data and function have the **same structure**!

Design Recipe allows combining cases if they are handled the same.

### 2023 Federal Income Tax Brackets

<table>
<thead>
<tr>
<th>Tax Rate</th>
<th>For Single Filers</th>
<th>For Married Individuals Filing Joint Returns</th>
<th>For Heads of Households</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>$0 to $11,000</td>
<td>$0 to $22,000</td>
<td>$0 to $15,700</td>
</tr>
<tr>
<td>12%</td>
<td>$11,000 to $44,725</td>
<td>$22,000 to $89,450</td>
<td>$15,700 to $59,850</td>
</tr>
<tr>
<td>22%</td>
<td>$44,725 to $95,375</td>
<td>$89,450 to $190,750</td>
<td>$59,850 to $95,350</td>
</tr>
<tr>
<td>24%</td>
<td>$95,375 to $182,100</td>
<td>$190,750 to $364,200</td>
<td>$95,350 to $182,100</td>
</tr>
<tr>
<td>32%</td>
<td>$182,100 to $231,250</td>
<td>$364,200 to $462,500</td>
<td>$182,100 to $231,250</td>
</tr>
<tr>
<td>35%</td>
<td>$231,250 to $578,125</td>
<td>$462,500 to $693,750</td>
<td>$231,250 to $578,100</td>
</tr>
<tr>
<td>37%</td>
<td>$578,125 or more</td>
<td>$693,750 or more</td>
<td>$578,100 or more</td>
</tr>
</tbody>
</table>

Source: Internal Revenue Service

---

```scheme
;; A Salary is one of:
;; [0, 11000)
;; [11000 44725)
;; [44725, 95375)
;; ...
;; Interp: Salary in US Dollars,
;; split by 2023 tax bracket
(define (10%-bracket? salary)
  (and (>= salary 0)
       (< salary 11000)))
(define (12%-bracket? salary)
  (and (>= salary 11000)
       (< salary 44725)))
;; ...

;; taxes-owed: Salary -> TaxBalance
;; computes federal income tax owed in 2023
(define (taxes-owed salary)
  (cond
   [(10%-bracket? salary) ....]
   [(12%-bracket? salary) ....]
   [else ....])))
```
Some Pre-defined Enumerations

; A **KeyEvent** is one of:
; - 1String
; - "left"
; - "right"
; - "up"
; - ...

; **WorldState** KeyEvent -> ...
(define (handle-key-events w ke)
  (cond
    [(= (string-length ke) 1) ...]
    [(string=? "left" ke) ...]
    [(string=? "right" ke) ...]
    [(string=? "up" ke) ...]
    [(string=? "down" ke) ...]
    ...))

; A **MouseEvt** is one of these Strings:
; - "button-down"
; - "button-up"
; - "drag"
; - "move"
; - "enter"
; - "leave"

;; handle-mouse: WorldState Coordinate Coordinate MouseEvt -> WorldState
;; Produces the next WorldState
;; from the given Worldstate, mouse position, and mouse event
(define (handle-mouse w x y evt)
  (cond
    [(string=? evt "button-down") ....]
    [(string=? evt "button-up") ....]
    [else ....]]))

; A 1String is a String of length 1, including
; - "\" (the backslash),
; - " " (the space bar),
; - "\t" (tab),
; - "\r" (return), and
; - "\b" (backspace).

; **interpretation** represents keys on the keyboard
In-class exercise: **big-bang** practice

- Create a big-bang traffic light simulator that changes on a mouse click (“button-down” event)

**Data Definition choice?**
- Pros?
- Cons?

```scheme
;; A TrafficLight is one of:
(define RED-LIGHT "RED")
(define GREEN-LIGHT "GREEN")
(define YELLOW-LIGHT "YELLOW")

;; Interpretation: Represents possible colors of a traffic light
(define (red-light? x) (string=? x RED-LIGHT))
(define (green-light? x) (string=? x GREEN-LIGHT))
(define (yellow-light? x) (string=? x YELLOW-LIGHT))

;; A TrafficLight2 is one of:
(define GREEN-L 0)
(define YELLOW-L 1)
(define RED-L 2)

;; Interp: represents a traffic light state
(define (red-L? ll) (= ll RED-L))
(define (green-L? ll) (= ll GREEN-L))
(define (yellow-L? ll) (= ll YELLOW-L))
```
Check-In Quiz 9/20

on gradescope

(due 1 minute before midnight)