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

Last Time

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

Is this what we want?

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

```
;; 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 2π)
;; 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
```

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

```
(define/contract (deg->rad deg)
  (-> AngleD? AngleR?)
  (* deg (/ pi 180)))
```

Step 5: Code

Not allowed by data def, but should be ok?

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

Step 4: Examples

```
(check-equal? (deg->rad 360) 0) ; ???
(check-equal? (deg->rad 360) (* 2 pi)) ; ???
```

Step 6: Tests

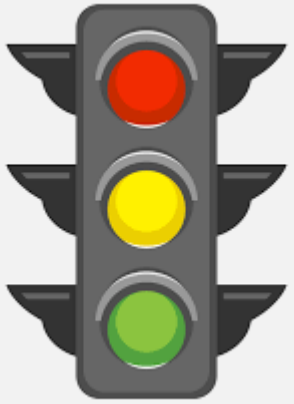
Kinds of Data Definitions

- 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")
```

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



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

```
(define (TrafficLight? x)
  (or (red-light? x)
      (green-light? x)
      (yellow-light? x)))
```

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")
```

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

(keep order the same)

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

cond is multi-arm `if` (expression)

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

Step 5: Code

Designing **data first** makes **writing function** (code) **easier**!

```
(check-equal? (next-light RED-LIGHT) GREEN-LIGHT)
(check-equal? (next-light GREEN-LIGHT) YELLOW-LIGHT)
(check-equal? (next-light YELLOW-LIGHT) RED-LIGHT)
```

Step 4: Examples

Last Time

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))
```

A **function's template** is completely **determined by the input's Data Definition**

```
;; 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) ....]))
```

(keep order the same)

Step 5: ~~Code~~ **Template**

Step 6: **Code** (fill in the "...." with arithmetic)

Kinds of Data Definitions

- Basic data
- Intervals
- Enumerations
- Itemizations

Itemization Data Definitions (Generalized enumeration)

2023 Federal Income Tax Brackets

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

Source: Internal Revenue Service

```
;; 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)))
;; ...
```

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

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

`else` is fallthrough case

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

Some Pre-defined Enumerations

```
; A KeyEvent is one of:
; – 1String
; – "left"
; – "right"
; – "up"
; – ...
```

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

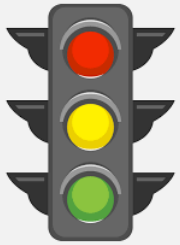
Template

```
; 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) ...]
    ...))
```

```
;; 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?

```
;; 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? li) (= li RED-L))
(define (green-L? li) (= li GREEN-L))
(define (yellow-L? li) (= li YELLOW-L))
```

Check-In Quiz 9/20

on gradescope

(due 1 minute before midnight)