



Mutually Recursive Definitions



Plan for today

- What is mutual recursion
- Example
 - Illustrating general approach
- Example
 - Illustrating and addressing technical issue with “list” based approach



Type Definition

```
; A parent is a structure
;   (make-parent l n)
; where l is a list-of-children,
;   n is a symbol

; A list-of-children is either
;   empty, or
;   (cons p l) where
; where p is a parent,
;   l is a list of children
```



Template

- **One** function on any of these types requires writing a set of functions for **all** the mutually recursive types
- Every different reference to a mutually recursive type **in definition** means we need a call to the appropriate function **in template**



Template (in class)

```
; A parent is a structure
;   (make-parent l n)
; where l is a list-of-children,
;   n is a symbol
(define-struct parent (loc name))
; f : parent -> ...
; (define (f x)
;   (... g (parent-loc x)
;         ... (parent-name x)))
```



Template (in class)

```
; A list-of-children is either
;   empty, or
;   (cons p l) where
; where p is a parent,
;   l is a list of children
; g : list-of-childer -> ...
; (define (g y)
;   (cond [(empty? y) ...]
;         [else ...(f (first y))...
;                   (g (rest y))...]))
```



Function calls in template

- Mutually recursive calls are part of template
 - Use of a mutually recursive type is just the same as a recursive use of a type itself
 - Mutually recursive type definitions are essentially the same as one big recursive type definition
- Where we put the function calls in the template is crucial for ensuring termination



More about termination

- For the recursive types we saw before today, recursive functions terminate if
 - They handle the base cases, and
 - They only make recursive calls on smaller values
- Mutually recursive definitions are the same
 - Example: Imagine a type box that can contain balloons, and a type balloon that can contain boxes.
 - Basic intuition for why template ensures termination:
 - Any box will be bigger than any box it contains
 - Similarly for balloons.



Code (in class)

- Write a function that counts descendants

```
; kids-parent : parent -> natural
; kids-loc    : loc -> natural
(define (kids-parent p)
  (kids-loc (parent-loc p)))
(define (kids-loc l)
  (cond [(empty? y) 0]
        [else (+ (+ 1 (kids-parent (first y)))
                  (kids-loc (rest y)))]))
; Discussed "0" vs. "1". Also, note the "+1"
; Note: Mutual "defines" should be contiguous
```



Alternate Representation

```
; A web-page is a structure
;   (make-wp s d)
; where s is a symbol and p is a web-document

; A web-document is
;   empty,
;   (cons s d), or
;   (cons p d),
; where s is a symbol, and p is a web-page,
;       d is a web-document
```



Template (in class)

```
; A web-page is a structure
;   (make-wp s d)
; where s is a symbol and p is a web-document

(define-struct wp (start document))

; f : web-page -> ... // this is the easy case
(define (f x)
  (... (wp-start x)
       ... (g (wp-document x))))
```



Template (in class)

```
; A web-document is
;   empty,
;   (cons s d), or
;   (cons p d),
; where s is a symbol, and p is a web-page,
;       d is a web-document

; g : web-document ->
; (define (g x)
; (cond [(empty? x) ...]
;       [else (cond [(symbol? (first x)) ... ]
;                   [else ...
```



Template (in class)

```
; A web-document is
;   empty,
;   (cons s d), or
;   (cons p d),
; where s is a symbol, and p is a web-page,
;       d is a web-document

; g : web-document ->
; (define (g x)
; (cond [(empty? x) ...]           // simplified cond
;       [(symbol? (first x)) ... ]
;       [else ...]))
```



Template (in class)

```
; Complete Template
;
; g : web-document ->
; (define (g x)
; (cond [(empty? x) ...]
;       [(symbol? (first x))
;        ... (first x) ... f (rest x) ... ]
;       [else
;        ... (first x) ... g (rest x) ... ]))
```



For Next Class

- No homework for this week
- Midterm:
 - Next lecture (Wednesday)
 - In class. Closed book / notes / computer.
 - Last year's exam, with key, is online
 - Chapters 1-13
- No quizzes until midterm