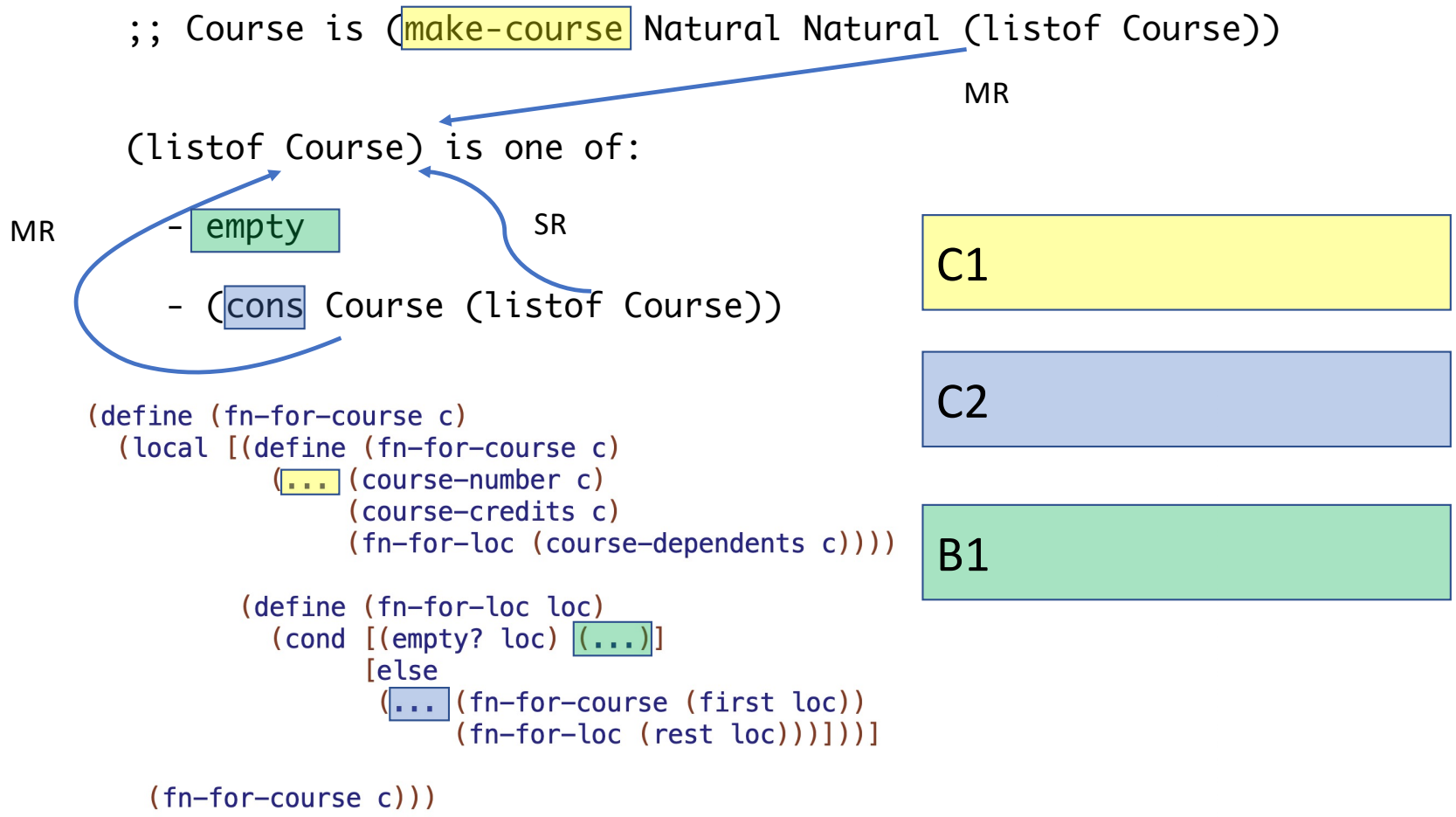
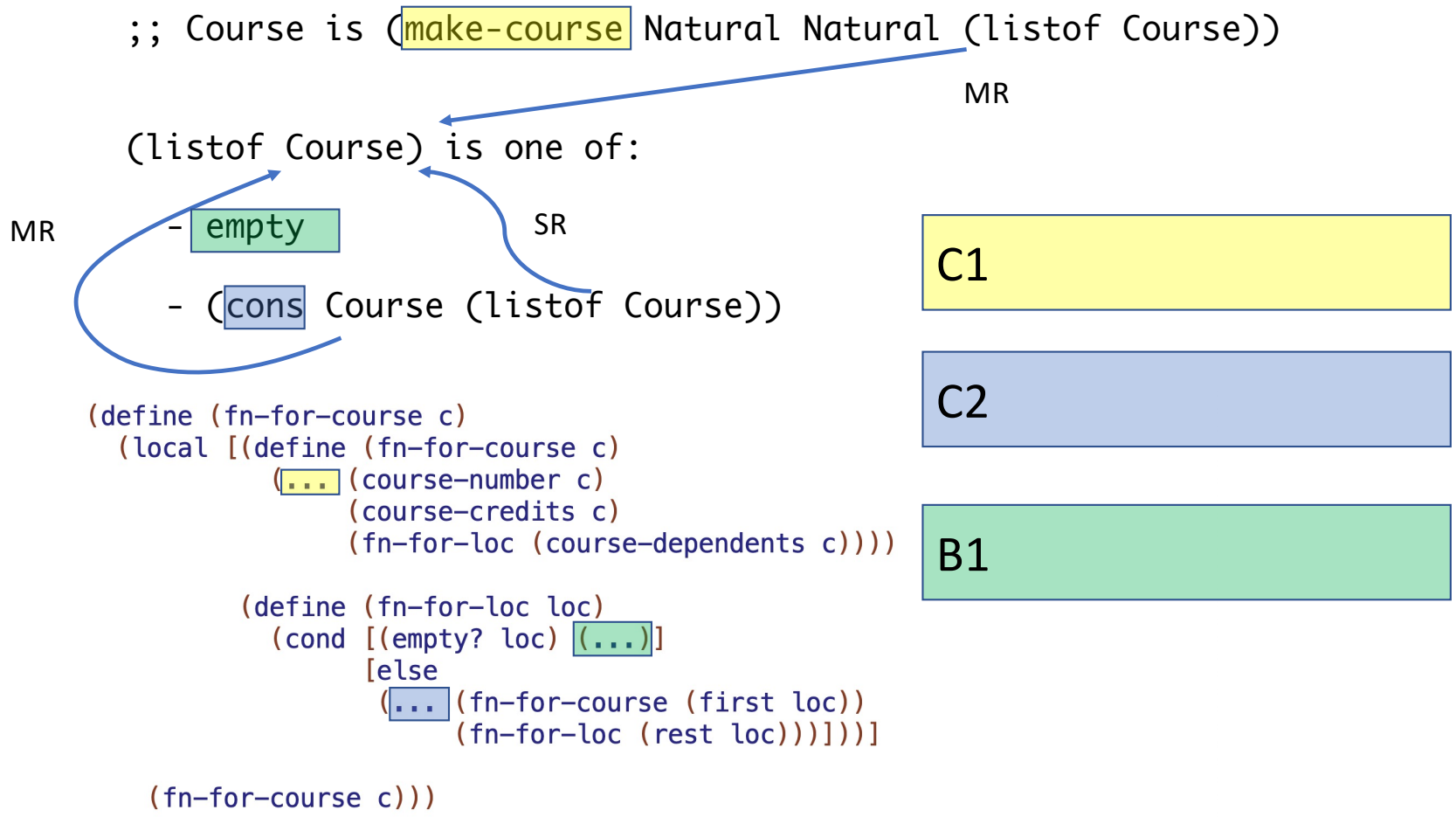
L14

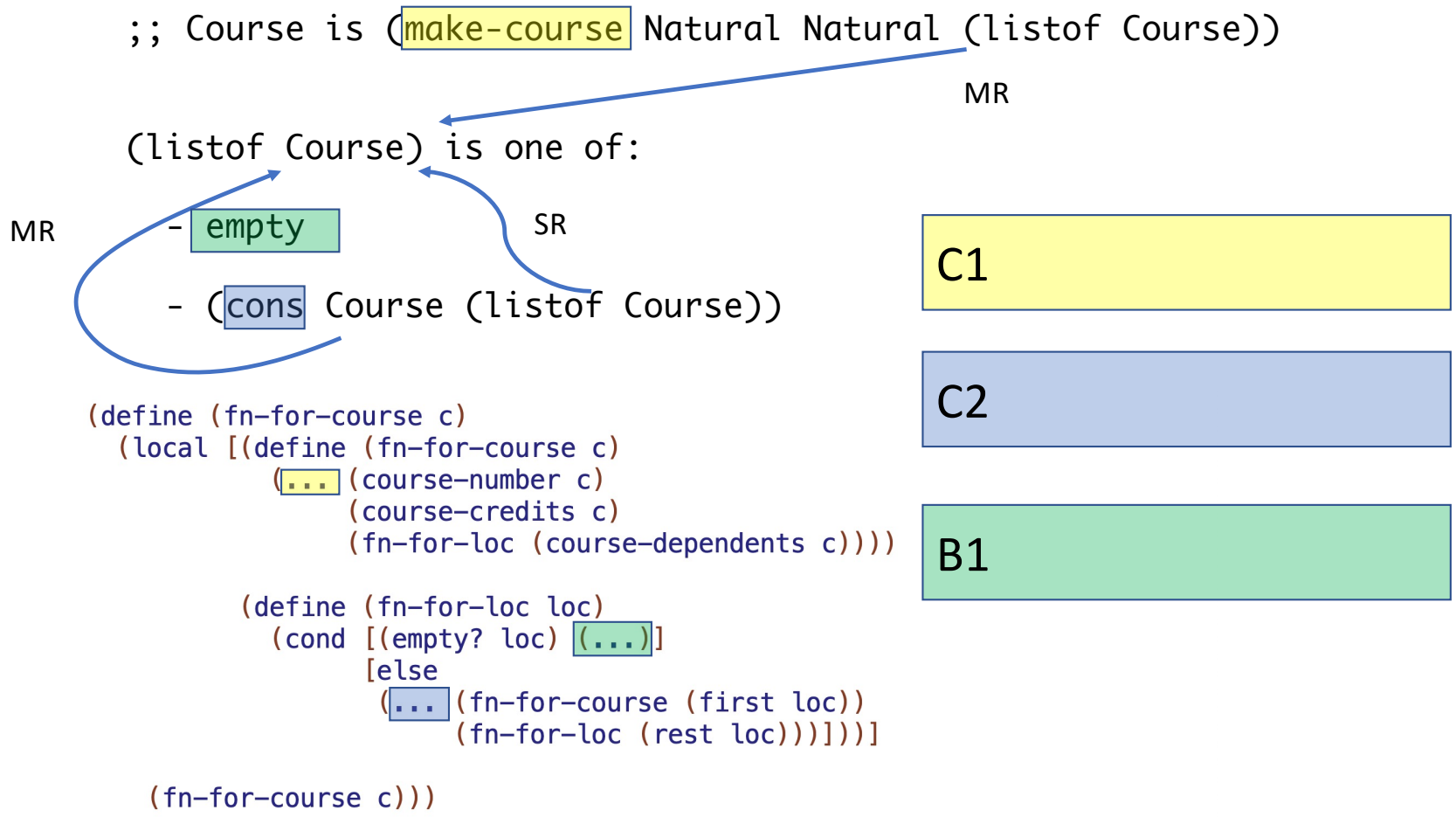
- Clicker questions on using built-in abstract functions
  - match the argument and result of your function
  - to argument and result of one or more built-in abstract functions

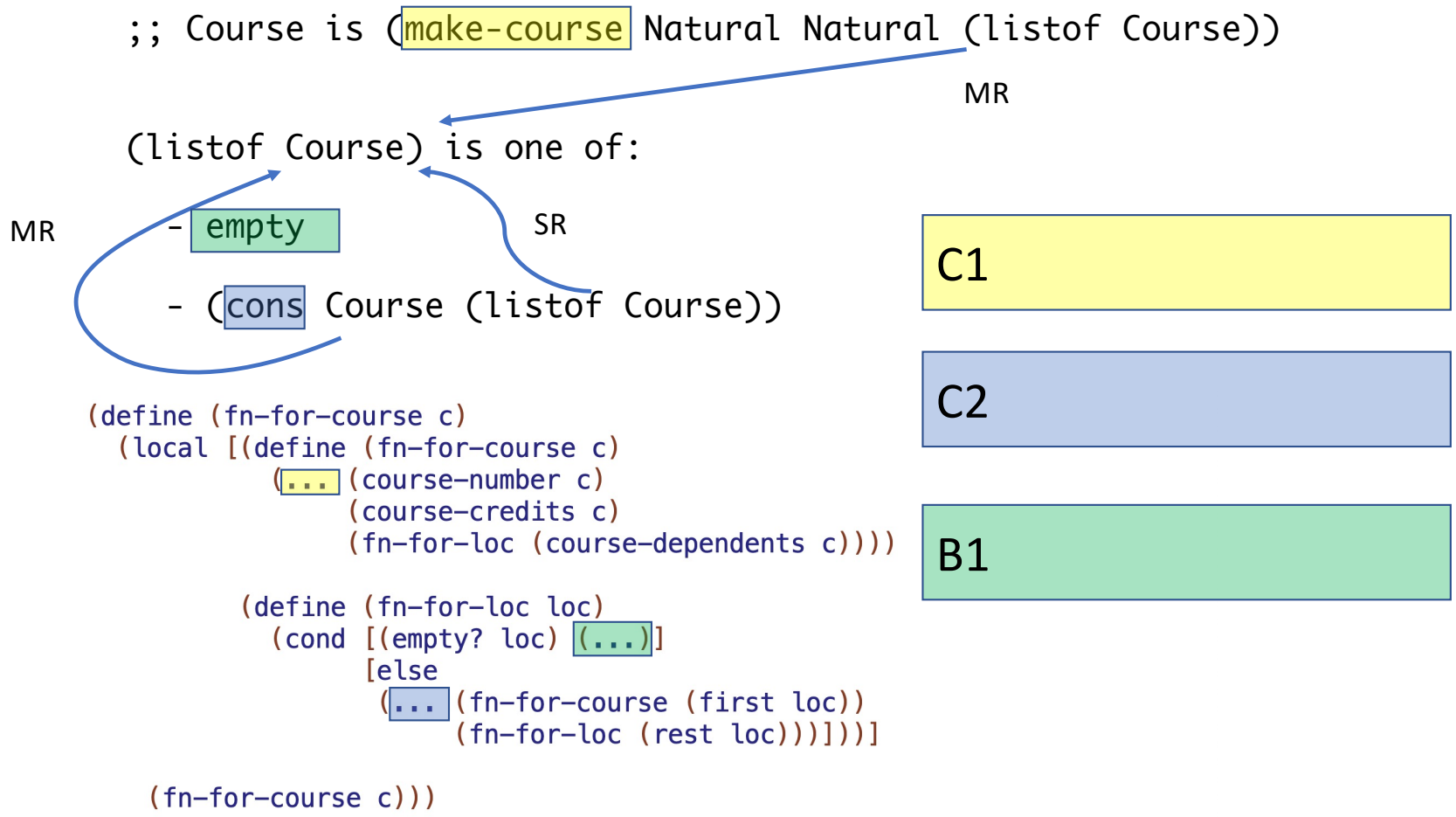
- multi-type fold functions
  - https://en.wikipedia.org/wiki/Fold_(higher-order_function)
  - easy data traversal shows up in most languages (visitor pattern etc.)
  - designing
  - using
  - complications

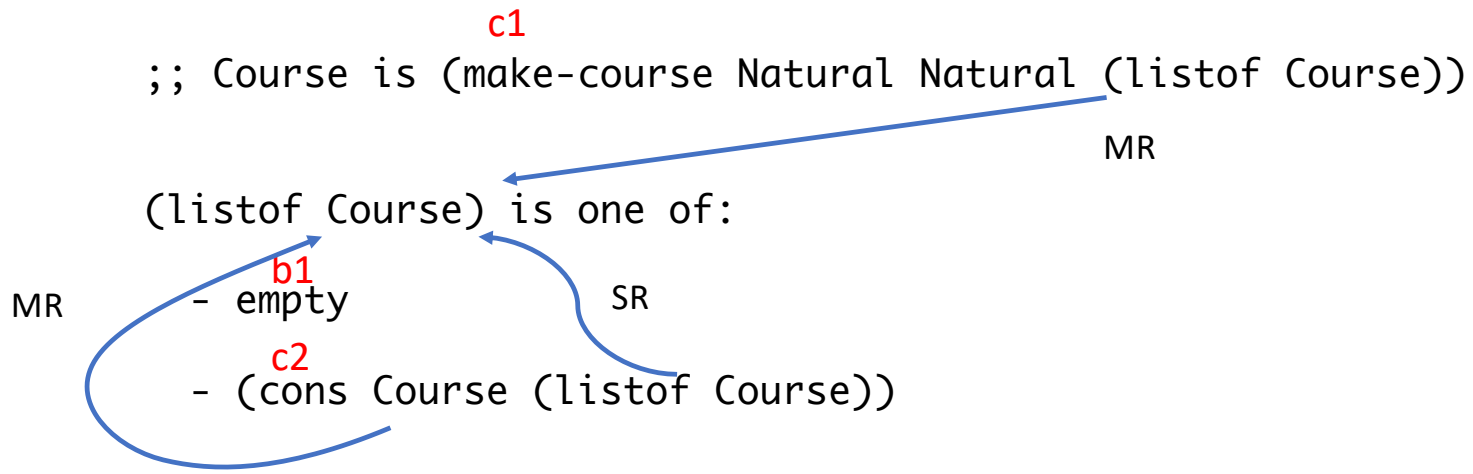
```
;; Course is (make-course Natural Natural (listof Course))

(listof Course) is one of:
  - empty
  - (cons Course (listof Course))
```


;; Course is (make-course Natural Natural (listof Course))
MR
(listof Course) is one of:
MR
- empty
SR
- (cons Course (listof Course))
(define (fn-for-course c)
(local [(define (fn-for-course c)
(... (course-number c)
(course-credits c)
(fn-for-loc (course-dependents c))))
(define (fn-for-loc loc)
(cond [(empty? loc) (...)]
[else
(... (fn-for-course (first loc))
(fn-for-loc (rest loc)))]))]
(fn-for-course c)))
C1
C2
B1

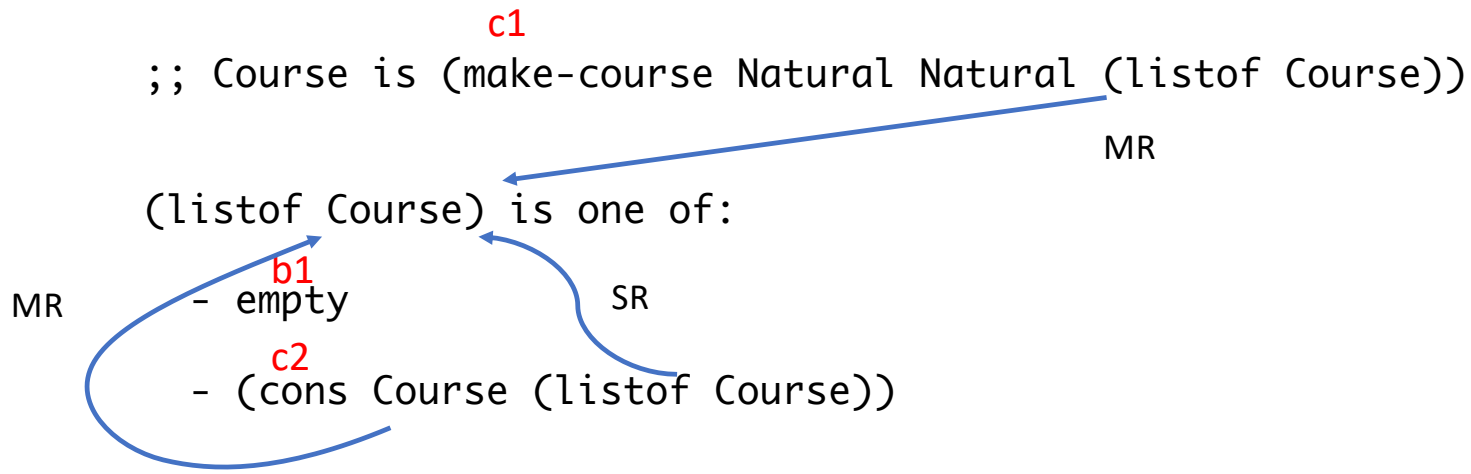

;; Course is (make-course Natural Natural (listof Course))
MR
(listof Course) is one of:
MR
- empty
SR
- (cons Course (listof Course))
(define (fn-for-course c)
(local [(define (fn-for-course c)
(... (course-number c)
(course-credits c)
(fn-for-loc (course-dependents c))))
(define (fn-for-loc loc)
(cond [(empty? loc) (...)]
[else
(... (fn-for-course (first loc))
(fn-for-loc (rest loc)))]))]
(fn-for-course c)))
C1
C2
B1


;; Course is (make-course Natural Natural (listof Course))
MR
(listof Course) is one of:
MR
- empty
SR
- (cons Course (listof Course))
(define (fn-for-course c)
(local [(define (fn-for-course c)
(... (course-number c)
(course-credits c)
(fn-for-loc (course-dependents c))))
(define (fn-for-loc loc)
(cond [(empty? loc) (...)]
[else
(... (fn-for-course (first loc))
(fn-for-loc (rest loc)))]))]
(fn-for-course c)))
C1
C2
B1

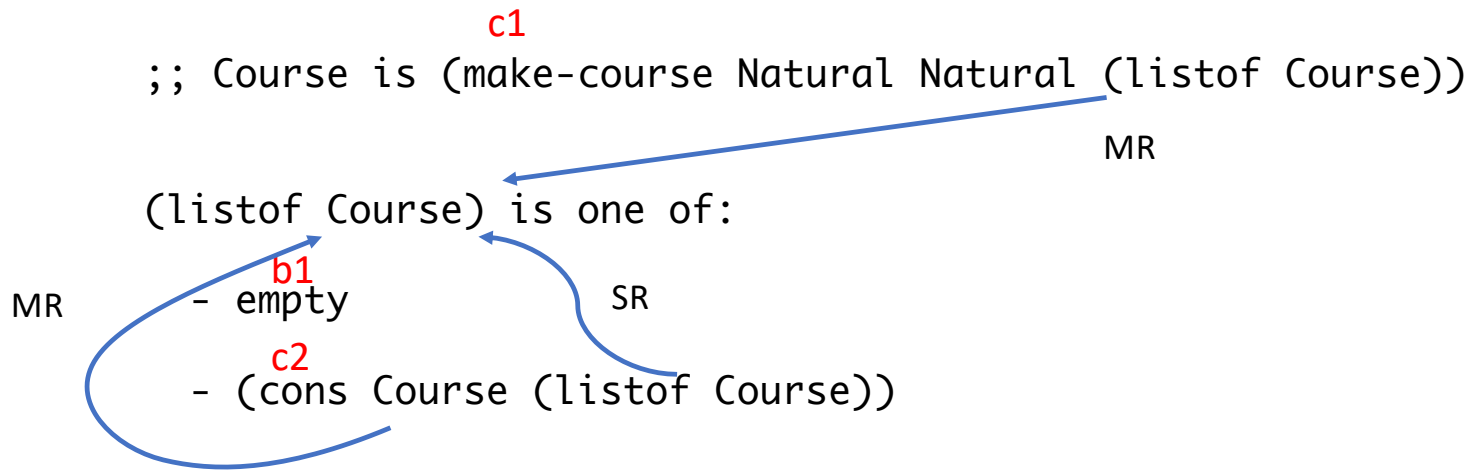

;; Course is (make-course Natural Natural (listof Course))
MR
(listof Course) is one of:
MR
- empty
SR
- (cons Course (listof Course))
(define (fn-for-course c)
(local [(define (fn-for-course c)
(... (course-number c)
(course-credits c)
(fn-for-loc (course-dependents c))))
(define (fn-for-loc loc)
(cond [(empty? loc) (...)]
[else
(... (fn-for-course (first loc))
(fn-for-loc (rest loc)))]))]
(fn-for-course c)))
C1
C2
B1


c1
;; Course is (make-course Natural Natural (listof Course))
MR
(listof Course) is one of:
b1
MR
- empty
SR
c2
- (cons Course (listof Course))

C1	C2	B1



	C1	C2	B1
All nums	(cons num rmr)	append	empty
Total credits	(+ cr rmr)	+	0



	C1	C2	B1
All nums	(cons num rmr)	append	empty
Total credits	(+ cr rmr)	+	0
Courses w/ credits	(if (>= cr n) (cons <C> rmr) rmr)	append	empty no access to actual course
find			whole tree every time