

# 2017 Midterm Exam Lecture Notes

October 4, 2018

## Pair and List

$$\left[ \begin{array}{l} (\text{Cons } 1 \ 2) \\ \quad \quad \quad \# \\ (\text{List } 1 \ 2) = (\text{Cons } 1 \ (\text{list } 2)) \end{array} \right]$$

List ::= (empty list) | (Cons

a list is a pair

a pair is not necessarily a list

list?

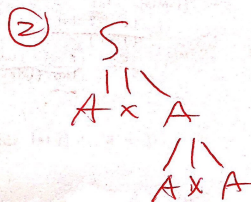
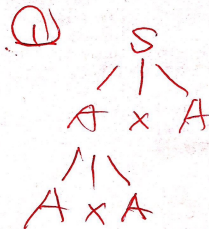
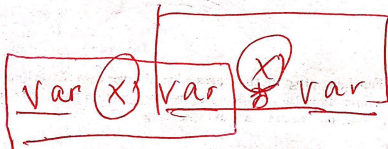
if  $(= (\text{list } 1 \ 2) (\text{list } 3 \ 4))$

Q2

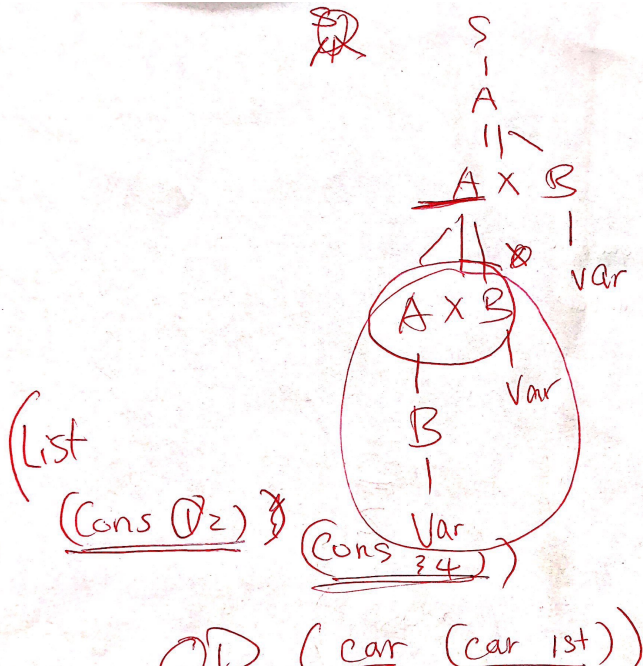
old:  $A \rightarrow A \times A \mid A \oslash A \mid B$

new:  $A \rightarrow \underline{A \times B} \mid A \oslash B \mid B$

$B \rightarrow \neg B \mid C$



Q2



4. (8 pt) Identify free and bound variables in the following expression. Write F (for free variables) or B (for bound variables) under each variable in the description.

`(let ((foo (lambda (a b) (+ a c))) (c d) (d e)) (foo (+ a b c) (+ d e)))`

Handwritten annotations: A red box labeled "def" encloses the lambda expression. A red box labeled "body" encloses the body of the let expression. Red arrows indicate variable binding: from 'a' to 'a', from 'b' to 'b', from 'c' to 'c', from 'd' to 'd', and from 'e' to 'e'. Red letters 'F' and 'B' are written under each variable: 'F' under 'foo', 'B' under 'a', 'B' under 'b', 'F' under 'c', 'F' under 'd', 'F' under 'e', 'B' under 'foo' in the body, 'F' under 'a' in the body, 'F' under 'b' in the body, 'F' under 'c' in the body, 'F' under 'd' in the body, and 'F' under 'e' in the body.

Q7

(c) (2 pt) Convert the above FuncLang program into the curried form

Sol

```
(a) (define myList (list (cons 1 3)
                          (cons 4 2)
                          (cons 5 6)))
```

$(\text{List } 1\ 2)$   
 $\hookrightarrow \text{List } (\underline{\text{cons } 1\ 2})$

```
(define apply-on-nth
  (lambda (op lst n)
    (if (null? lst) -1
        (if (= n 1)
            (op (car (car lst))
                (cdr (car lst)))
            (apply-on-nth op (cdr lst) (- n 1))))))
```

$\text{Cdr } 2$

$\hookrightarrow (\text{list } 2)$

```
(b) (define apply-on-nth
      (lambda (op)
        (lambda (lst)
          (lambda (n)
            (if (null? lst) -1
                (if (= n 1)
                    (op (car (car lst))
                        (cdr (car lst)))
                    (apply-on-nth op (cdr lst) (- n 1))))))))
```

$op (\text{car } (\text{car } \text{lst}))$   
 $\text{car } \text{cdr } (\text{car } \text{lst})$