Joseph Camacho-Terrazas
11/17/2020
Lisp Programming

For this assignment, we were to write a Lisp program where given a circuit descriptor as input, we had to write a function that counted logical operators, listed the unique variables, and reduce the CD using tautologies. To count the logical operators, the function takes the CD as well as the user supplied operator argument. It will then compare the operators in the CD to the user input and will increase the count if there is a match. As for the unique function, it takes the CD and splits the list to obtain each variable using the findVariables function. We then use the unique function to get rid of any duplicate variables the previous function found. Lastly we call flatten in terminal to flatten our list and give us the final output. Finally, for the reduce function, we have the evalcd function which will make calls to the NOT, AND, and OR tautology functions in order to reduce the CD.

Joseph Camacho-Terrazas
11/17/2020
Lisp Programming

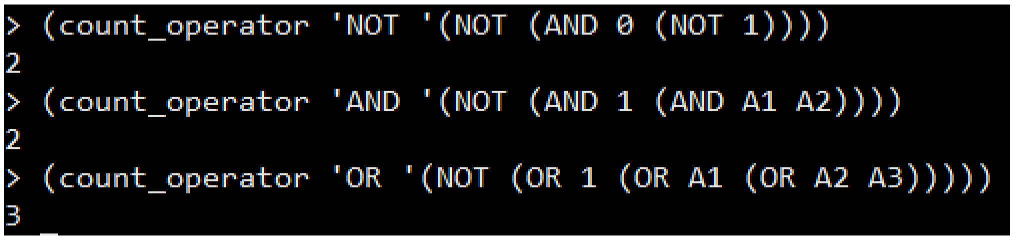
```
;;Joseph Camacho-Terrazas
;;11/17/2020
;;Input: A logical argument
;;Output: A reduced CD form, number of specified operators, or a list of all uniq
ue variables
;;Precondition: The user gives a proper CD as input
;;Postcondition: The program will give the correct output based on the specified
function.

;;Evaluates a circuit design

;;Counts the number of specified operators in the argument
;;Format is 'operator '(CD)
(define (count_operator x CD)
    (cond ((null? CD) 0);;Empty case
        ((not (list? CD));;Non-list case
        (if (eq? x CD) 1 0));;Perform count on the list
        (else (+ (count_operator x (car CD)) (count_operator x (cdr CD)))));;End
cond
);;End define

;;Finds all input variables in the argument
;;Format is '(CD)
(define (findVariables CD)
    (cond
        ((null? CD) '());;Empty case
        ((not (list? CD)) '());;Non-list case
        ((or (eq? (car CD) 1);;Find the variables in the list using recursion
                (eq? (car CD) 0)
                (eq? (car CD) 'AND)
                (eq? (car CD) 'OR)
                (eq? (car CD) 'NOT))
            (findVariables (cdr CD)))
        (else (cons (car CD) (findVariables (cdr CD)))));;End cond
);;End define

;;Finds all the unique variables in the argument
;;Format is '(CD)
(define (unique CD)
    (cond ((null? CD) '());;Empty Case
        ((not (list? CD)) '());;Non-list Case
        ((member (car CD) (cdr CD)) (unique (cdr CD)));;Split list to find unique
 variables
```

Joseph Camacho-Terrazas
11/17/2020
Lisp Programming

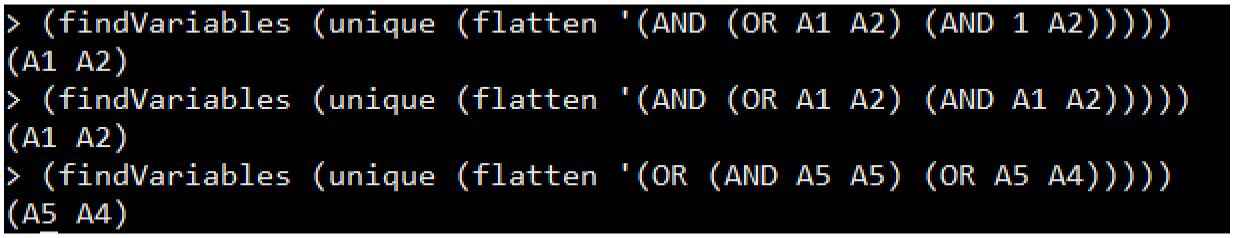
```
        (else (cons (car CD) (unique (cdr CD)))));;End cond
);;End define

;; NOT CD1
(define(evalcd CD)
    ;; Base Case
    (cond ((null? CD) '())
        ;; True, False, or A1....A1000
        ((not (list? CD)) CD)
        ((eq? (car CD) 'NOT) (evalcd_not CD))
        ((eq? (car CD) 'AND) (evalcd_and CD))
        ((eq? (car CD) 'OR ) (evalcd_or CD)));;End cond
);;End define

;;PRE:MUST be a (NOT CD) form (CAR CD) => NOT
;;Reduce the Argument and see if we can reduce it
(define (evalcd_not CD)
    (cond ((eq? (evalcd (cadr CD)) 0) 1)
        ((eq? (evalcd (cadr CD)) 1) 0)
        (else (cons 'NOT (list (evalcd (cadr CD))))));;End cond
);;End define

;;PRE: MUST be (AND CD1 CD2) format
;;POST: Apply simple tautologies to the CD1 and CD2 and maybe reduce
;;AND
(define (evalcd_and CD)
    (cond ((eq? (evalcd (cadr CD)) 0) 0)
        ((eq? (evalcd (caddr CD)) 0) 0)
        ((eq? (evalcd (cadr CD)) 1) (evalcd (caddr CD)))
        ((eq? (evalcd (caddr CD)) 1) (evalcd (cadr CD)))
        (else (cons 'AND
            (list (evalcd (cadr CD))
            (evalcd (caddr CD))))));;End cond
);;End define

;;PRE: MUST be (OR CD1 CD2) format
;;POST: Apply simple tautologies to the CD1 and CD2 and maybe reduce
;;OR
(define (evalcd_or CD)
    (cond ((eq? (evalcd (cadr CD)) 1) 1)
        ((eq? (evalcd (caddr CD)) 1) 1)
        ((eq? (evalcd (cadr CD)) 0) (evalcd (caddr CD)))
        ((eq? (evalcd (caddr CD)) 0) (evalcd (cadr CD)))
        (else (cons 'OR
```

Joseph Camacho-Terrazas
11/17/2020
Lisp Programming

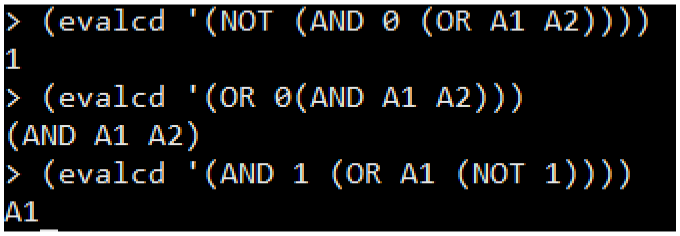
```
            (list (evalcd (cadr CD))
            (evalcd (caddr CD))))));;End cond
);;End define
```



```
> (count_operator 'NOT '(NOT (AND 0 (NOT 1))))
2
> (count_operator 'AND '(NOT (AND 1 (AND A1 A2))))
2
> (count_operator 'OR '(NOT (OR 1 (OR A1 (OR A2 A3)))))
3
```



```
> (findVariables (unique (flatten '(AND (OR A1 A2) (AND 1 A2)))))
(A1 A2)
> (findVariables (unique (flatten '(AND (OR A1 A2) (AND A1 A2)))))
(A1 A2)
> (findVariables (unique (flatten '(OR (AND A5 A5) (OR A5 A4)))))
(A5 A4)
```



```
> (evalcd '(NOT (AND 0 (OR A1 A2))))
1
> (evalcd '(OR 0(AND A1 A2)))
(AND A1 A2)
> (evalcd '(AND 1 (OR A1 (NOT 1))))
A1
```