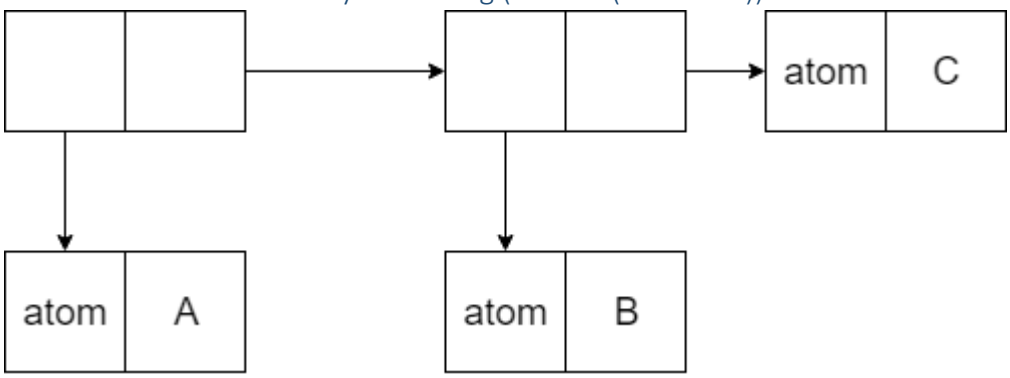
Programming Languages Hw3

Arya Banaeizadeh 9431029

John Mitchell, Concepts in Programming Languages

3.1 Cons Cell Representations

(a) Draw the list structure created by evaluating (cons 'A (cons 'B 'C)).



(b) Write apure Lisp expression that will result in this representation, with no sharing of the (B .C) cell. Explain why your expression produces this structure.

(cons ((cons 'A (cons 'B 'C)) (cons 'B 'C)))

First cons creates a cons cell containing a cons which contains atom A and Cons B C and another cons of atom B and atom C each time (cons 'B 'C) is mentioned it's stored in a new place of memory and thus resulting in the desired shape.

(c) Write a pure Lisp expression that will result in this representation, with sharing of the (B . C) cell. Explain why your expression produces this structure.

((lambda (x)(cons (cons 'A x)x))(cons 'B 'C))

For this image, we should define a lambda function with x as argument and apply (cons 'B 'C) to the function to reference both of them to the same place in memory.

3.2 Conditional Expressions in Lisp

(a) Suppose McCarthy suggests that the value of (cond (*p*1 *e*1) . . . (*pn en*)) should be the value of ek if pk is true and if, for every i<k, the value of expression pi is either false or undefined. Is it possible to implement this interpretation? Why or why not? (Hint: Remember the halting problem.)

No because there's no way to tell if an expression is undefined, if during execution pi is undefined the program will halt and we can't identify the expression.

(b) Another design for conditional might allow any of several values if more than one of the guards (p1, . . . , pn) is true. More specifically (and be sure to read carefully), suppose someone suggests the following meaning for conditional:

i. The conditional's value is undefined if none of the pk is true.

ii. If some pk are true, then the implementation must return the value of ej for some j with pj true. However, it need not be the first such ej . Note that in (cond (a b) (c d) (e f)), for example, if a runs forever, c evaluates to true, and e halts in error, the value of this expression should be the value of d, if it has one. Briefly describe a way to implement conditional so that properties i and ii are true. You need to write only two or three sentences to explain the main idea.

For achieving this we should acquire a machine capable of parallel computing. The first to return true will determine the value of cond. If none return true then the value would be undefined

(c) Under the original interpretation, the function would give us t for odd numbers and nil for even numbers. Modify this expression so that it would always give us t for odd numbers and nil for even numbers under the alternative interpretation described in part (b).

(define odd(x) cond((eq x 0) nill)

((eq x 1) t)

((> x 1) (odd(- x 2)))

((< x 0) (odd(+ x 2)))

(odd(+ x 1) nil))

)

(d) Of the original interpretation, the interpretation in part (a), and the interpretation in part (b), which ones would allow us to implement Scor most easily?What about Por?Which interpretation would make implementations of short-circuiting or difficult?Which interpretation would make implementation of parallel or difficult?Why?

For SCOR (a) is a better method since (a) will run sequentially (if the first condition isn't true it'll jump to the 2nd)

For POR however (b) is a better choice since expressions are evaluated simultaneously.

It's difficult to implement POR using (a), consider e1 is undefined and e2 is true the return value should be true and implementing this case would be hard if (a) is used.

(b) for SCOR results in redundant conditional checks.

3.4 Lisp and Higher-Order Functions

(a) Fill in the missing code in the following definition. The correct answer is short and fits here easily. You may also want to answer parts (b) and (c) first.

f(h x)

(b) When (compose2 maplist car) is evaluated, the result is a function defined by (lambda (f xs) (g . . .)) above, with

i. which function replacing g?

maplist

ii. and which function replacing h?

car

(c) We could also write the subexpression (lambda (xs) (. . .)) as (compose (. . .) (. . .)) for two functions. Which two functions are these? (Write them in the correct order.)

Compose (f h)

3.5 Definition of Garbage

(a) If a memory location is garbage according to our definition, is it necessarily garbage according to McCarthy's definition? Explain why or why not.

No, it's possible that a list can not be accessed by our program but accessible by other programs or the case that there is a pointer assigned to a place of memory but it's not used in the runtime.

Another case would be inline functions; the pointer exists after using them but there is no way to refer to them.

(b) If a location is garbage according to McCarthy's definition, is it garbage by our definition? Explain why or why not.

Yes, program execution will be able to continue with different arrangements of car and cdr. so if it's not able to run on different arrangements then it's not able to be continued using car and cdr.

(c) There are garbage collectors that collect everything that is garbage according to McCarthy's definition. Would it be possible to write a garbage collector to collect everything that is garbage according to our definition? Explain why or why not?

No, determining this before runtime would result in a massive overhead in the best case. However there are some alternate ways to estimate the places in the memory that are no longer needed (like reference integer etc)

3.6 Reference Counting

(a) Describe how reference counting could be used for garbage collection in evaluating the following expression: (car (cdr (cons (cons a b) (cons c d)))) where a, b, c, a nd d are previously defined names for cells. Assume that the reference counts for a, b, c, a nd d are initially set to some numbers greater than 0, so that these do not become garbage. Assume that the result of the entire expression is not garbage. How many of the three cons cells generated by the evaluation of this expression can be returned to the free-storage list?

The result is: atom C

All the other parts are considered garbage and will be released.

Car(Cdr(cons (cons a b)(cons c d)))

Cons(a b) is garbage cell (flag = 0)

Car(cons (c d)) : d is garbage

(b) The "impure" Lisp function rplaca takes as arguments a cons cell c and a value v and modifies c's address field to point to v. Note that this operation does not produce a new cons cell; it modifies the one it receives as an argument. The function rplacd performs the same function with respect the decrement portion of its argument cons cell. Lisp programs that use rplaca or rplacd may create memory structures that cannot be garbage collected properly by reference counting. Describe a configuration of cons cells that can be created by use of operations of pure Lisp and rplaca and rplacd. Explain why the reference-counting algorithm deos not work properly on this structure.

Consider this:

(lamda(x) (cons (replaca x ('a 'b)) (car x)))) (cons ('c 'd) ('e 'f))

3.8 Concurrency in Lisp

(a) Assuming an unbounded number of processors, how much time would you expect the evaluation of the following fib function to take on positive-integer argument n?

We are interested only in time up to a multiplicative constant; you may use "big Oh" notation if you wish. If two instructions are done at the same time by two processors, count that as one unit of time.

t(n) = O(1) + MAX(t(n-1) , t(n-2)) =

O(1) + t(n-1) => O(n)

(b) At first glance, we might expect that two expressions which differ only because an occurrence of a subexpression e is replaced with (future e), would be equivalent. However, there are some circumstances in which the result of evaluating one might differ from the other. More specifically, side effects may cause problems. To demonstrate this, write an expression of the form (. . . e . . .) so that when the e is changed to (future e), the expression's value or behavior might be different because of side effects, and explain why. Do not be concerned with the efficiency of either computation or the degree of parallelism.

(Lambda (e) (cons (car e) (rplaca e c))) (cons a b) = (cons (a (rplaca e c)))

(Lambda (e) (cons (car future(e)) (rplaca e c))) (cons a b) = (cons (c (rplaca e c)))

(c) Side effects are not the only cause for different evaluation results. Write a pure Lisp expression of the form (. . . e' . . .) so that when the e' is changed to (future e'), the expression's value or behavior might be different, and explain why.

I. (cond (a b) (c d))

II. (cond (future(a) b) (c d))

for (I) d executes if a is fls and c is tru (if a is tru b executes and program is over). However in II, 1st cond will execute parallel to 2nd cond so if future(a) is tru then 2nd cond may execute as well.

(d) Suppose you are part of a language design team that has adopted futures as an approach to concurrency. The head of your team suggests an error-handling feature called a try block. The syntactic form of a try block is This construct would have the following characteristics:

i. Errors are programmer defined and occur when an expression of the form (raise error-i) is evaluated inside e, the main expression of the try block.

ii. If no errors occur, then (try e (error-1 handler-1) . . .) is equivalent to e.

iii. If the error named error-i occurs during the evaluation of e, the rest of the computation of e is aborted, the expression handler-i is evaluated, and this becomes the value of the try block.

The other members of the team think this is a great idea and, claiming that it is a completely straightforward construct, ask you to go ahead and implement it. You think the construct might raise some tricky issues. Name two problems or important interactions between error handling and concurrency that you think need to be considered. Give short code examples or sketches to illustrate your point(s). (Note: You are not being asked to solve any problems associated with futures and try blocks; just identify the issues.) Assume for this part that you are using pure Lisp (no side effects).

For one because of the parallel execution different order of commands may get executed and therefore different errors could happen in different runtimes. Which leads to different handlers getting executed and producing a different result. Also the errors can't be traced to their origin.