Scheme Lists

Abhinav Ashar
CS 61A: Structure and Interpretation of Computer Programs

January 18, 2019

1 Scheme Lists

a) Scheme lists are analogous to linked lists in Python, with *first* as a value and *rest* as a pointer to another list. In Scheme, however, anything can technically be in the *rest*.

b) (cons 2 3) is represented as (2 . 3) in dot notation

c) Whenever a . is followed by a (), they cancel each other out (Ex. (1 . (2 . (3))) → (1 2 3). Therefore, dots only show up for pairs where the second element is not a pair or a nil.

d) When converting something to dot notation, add parentheses when you see cons, write the *first* value, add a . , and write the *rest* in parentheses, and do that for the whole expression. At the end, eliminate any adjacent . () pairs.

e) **Well-formed list:** *first* is a value and *rest* is another Scheme list or null. Otherwise, it is not a well-formed list.
Example:
(1 2 3) → well-formed
(cons 1 (cons 2 (cons 3 nil) → well-formed
(1 . 2 3) → not well-formed
(cons 1 (cons 2 3)) → not well-formed

f) **Hint:** Well-formed Scheme lists can be easily converted in your head to a Python linked list, but doing the same with mal-formed Scheme lists is not easy.

g) **Pair:** has a *first* and a *rest* (does not need to be well-formed)

h) (car lst) is like lst.first and (cdr lst) is like lst.rest

i) **list?** - checks if something is a well-formed list

j) All lists are pairs but not all pairs are lists (Ex. '(1 2 3 . 4) is a pair but not a list because it is not well-formed)

k) *quote* avoids initial evaluation of the argument that follows it