Computing Science (CMPUT) 325

Nonprocedural Programming

Department of Computing Science
University of Alberta

Symbolic Expressions

- Symbolic Expression (also known as S-Expression, s-expr, sexpr)
- The “universal” data structure for Lisp
- A generalization of atoms and lists that we had in Fun
- Adds the notion of a “dotted pair” `(x . y)`

Symbolic Expression - Definition

- (Compare with Fun)
- An atom is an s-expression
- If x1, ..., xn are s-expressions, then (x1 ... xn) is an s-expression (list)
- If x1 and x2 are s-expressions, then (x1 . x2) is an s-expression (dotted pair)
- Examples: hello, (a b c), (a (b) ((()))),
  (a . b)
  (a . (b . c))
  (1 2 3 (4 . 5))

Car and Cdr with Dotted Pairs

- `car` of `(x . y)` gives `x`
- `cdr` of `(x . y)` gives `y`
- In Lisp, we get the following identities:
  - `(car (cons 'x 'y)) = x`
  - `(cdr (cons 'x 'y)) = y`
- It is the same definition as with Lists before, but we no longer require the second argument of `cons` to be a list. Now, it can be **any** s-expression.

Warning about Dotted Pair in Lisp

- **WARNING: dots and many other special characters can be part of names in Lisp**
- `(a.b)` is NOT what you want - need whitespace to separate atoms from dots

```
* (car '(a.b))          WRONG! NOT a dotted pair!
A.B                     (a.b) is a list containing
* (cdr '(a.b))          one atom a.b
NIL
* (car '(a . b))       Correct
A
* (cdr '(a . b))
B
```

Dotted Pairs vs Lists

- Some dotted pairs are identical to some lists (more later)
- How to check? Just check the `car` and `cdr` parts
- Example:

```
* (car '(a))
A
* (cdr '(a))
NIL
* (car '(a . nil))
A
* (cdr '(a . nil))
NIL
* (equal '(a) '(a . nil))
T
```

Machine Level Representation

- How are s-expr represented in Lisp?
- Only two cases:
- Atoms (not discussed further here)
- Dotted pairs

Dotted Pair Representation

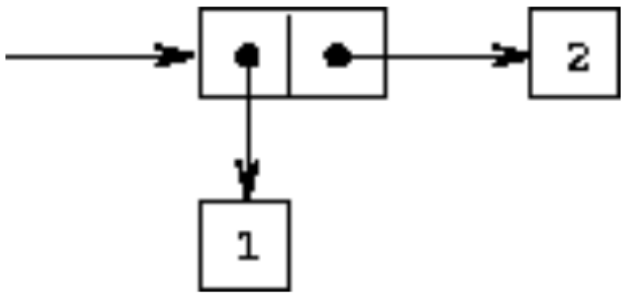


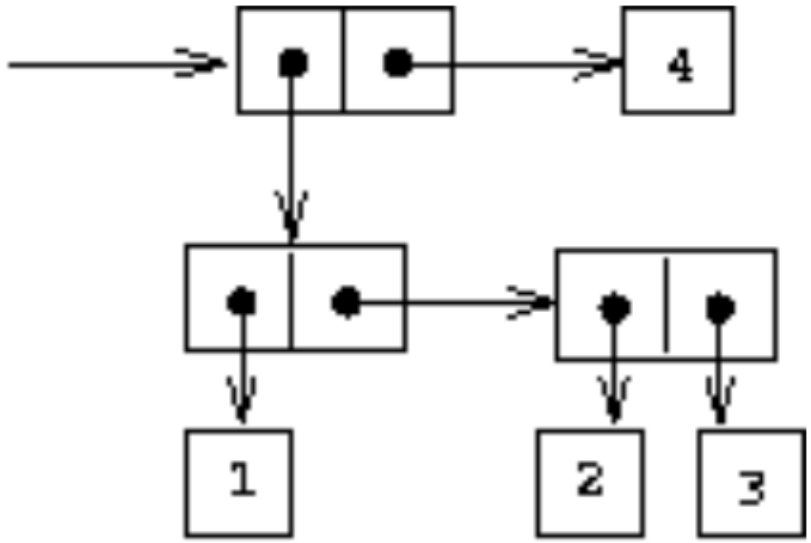
Image source:

`https://xuanji.appspot.com/isicp/2-2-closure.html`

- We need a data structure that can give us the car-part and the cdr-part
- Think of a “box” with two “cells” for car and cdr
- Each cell holds a pointer, either to an atom or another dotted pair
- Example: `(cons 1 2)`
- Constructs `(1 . 2)`

Example 2

Example: `((1 . (2 . 3)) . 4)`




```
(cons (cons 1
            (cons 2 3))
      4)
```

Image source:

`https://xuanji.appspot.com/isicp/2-2-closure.html`

Example 3

Example: `((1 . 2) . (3 . 4))`

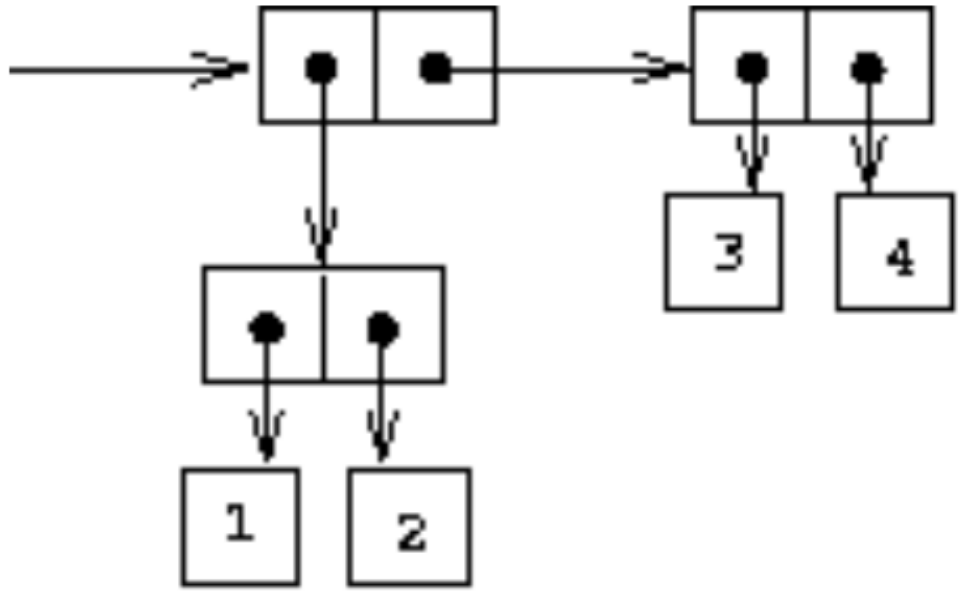


Image source:

`https://xuanji.appspot.com/isicp/2-2-closure.html`

Car and Cdr Examples with Dotted Pair

```
* (car '((1 . 2) . (3 . 4)))
(1 . 2)
* (cdr '((1 . 2) . (3 . 4)))
(3 . 4)
* (caar '((1 . 2) . (3 . 4)))
1
* (cdar '((1 . 2) . (3 . 4)))
2
* (cadr '((1 . 2) . (3 . 4)))
3
* (cddr '((1 . 2) . (3 . 4)))
4
```

Simple List Representation

- Simple lists are represented as linear chains
- Example: `(1 2 3 4)`
- Convention: a crossed-out box represents `nil`

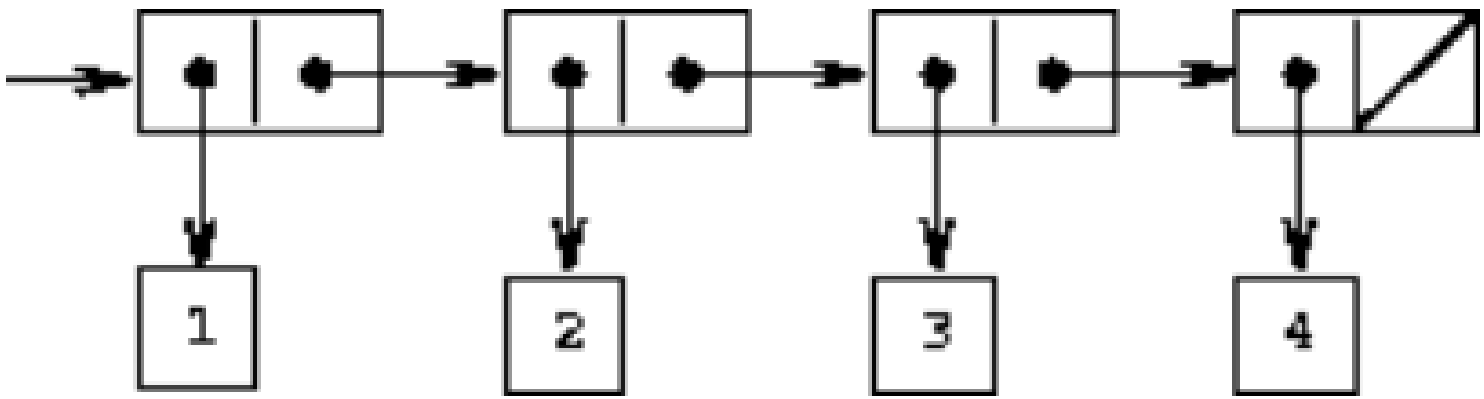


Image source:
`https://xuanji.appspot.com/isicp/2-2-closure.html`

More on Dotted Pair vs List

- We already saw that `(a . nil) = (a)`
- Reason: both have same car and cdr.
  - `(cdr '(a . nil)) = nil`
  - `(cdr '(a)) = nil`
- Every list can be rewritten as dotted pairs by repeatedly using the car, cdr definitions
- Example from last slide:

```
(1 2 3 4) = (1 . (2 3 4)) =
(1 . (2 . (3 4))) =
(1 . (2 . (3 . (4)))) =
(1 . (2 . (3 . (4 . nil))))
```

Dotted Pair vs List Example

- Example: all the following s-expr are identical to (a b c)
- `(a b c)` “simplest form” (fewest dots)
- `(a b c . nil)`
- `(a b . (c . nil))`
- `(a . (b . (c . nil)))` “full dotted pair form”
- `(a b . (c))`
- `(a . (b c))`

More Examples

- Lisp prints s-expr in their simplest form

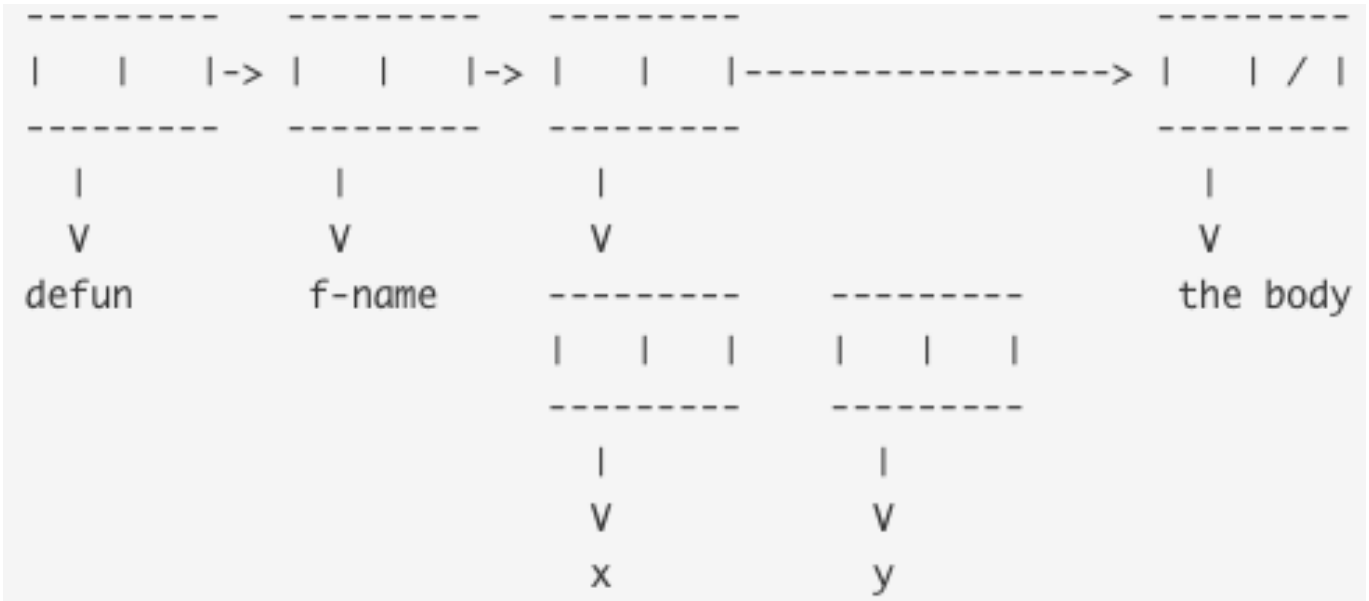
```
* (cons (cons 1 (cons 2 3)) 4)
((1 2 . 3) . 4)
* '(a . (b . (c . nil)))
(A B C)
* '(a b . (c))
(A B C)
* '(a . (b c))
(A B C)
* '(a b . (c . nil))
(A B C)
```

What is The Simplest Form?

- Least amount of dots
- Rule of thumb: can eliminate a dot followed by an open parenthesis
- Also eliminate the matching open/closing parentheses
  - Note: also works for `nil` if you write it as `()`
  - Exercise - simplify examples from the previous slides

Last example: Function Definition

- A function definition can be stored like any other s-expr
- This makes it really easy to write higher-order functions, which process other functions
- Example: `(defun f-name (x y) (body))`



Summary

- Introduced s-expressions and its machine representation
- Lists are special case of s-expr
- Many different-looking expressions are the same s-expr
- Can always check by “taking both apart” by computing car and cdr
- Can also check by drawing the diagrams for both
- Can also check by typing them into Lisp to convert to simplest form