LISP

Matej Stanic

Specialization Seminar

January 21st 2013

Overview

- LISP (**LIS**t **P**rocessing) one of the oldest programming languages
- invented by John McCarthy (MIT) in 1958
- first implementation by Steve Russell

History

- 1960-1965: Lisp 1.5
- 1970s: two main dialects, MacLisp and InterLisp
- 1970-1985: several Lisp machines (optimized)
  -> Artificial Intelligence
- 1981-1986: Common Lisp
- 1986-1994: ANSI Common Lisp (standard)
- today: **Common Lisp** and Scheme

What does LISP make unique?

- LISP is a functional programming language (lambda calculus)
- List Processing vs. (Lots of ((Irritating Superfluous) (Parentheses)))
  -> S-Expressions: code = data, data = code !
- LISP is free of types (optional types)
- LISP is a interpreter-language
  -> "read-eval-print-loop"
- LISP is a "programmable programming language"
  -> Macros

Important features

- simple, regular syntax
- powerful macros
- dynamic typing
- garbage collection

Syntax

Programs are represented by **S-Expressions**
S-Expressions consist of **atoms** and **lists** (syntax-tree!)

Atoms

- numbers, strings...
  and
- symbols

e.g. 91, "Text", `write`

Lists

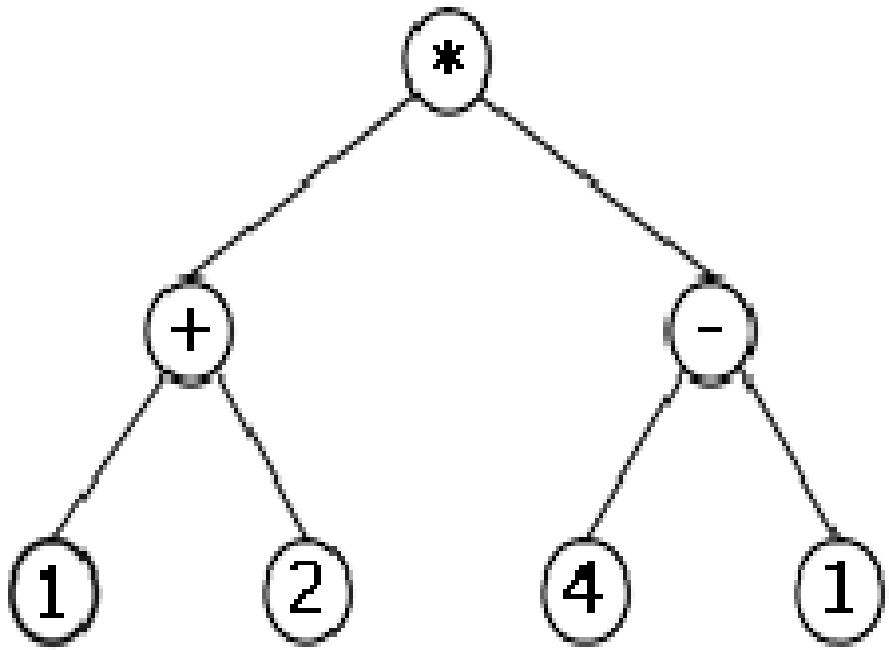
e.g. `(1 2 3)`, `(5 (5 6) 7)`

- also used for function calls (prefix notation)

e.g. `( + 1 2 3)`, `(* (+ 1 2) (- 4 1))`
-> code = list = data!

Syntax-tree of a S-Expression

Example: (* (+ 1 2) (- 4 1))



Important functions

- `setf` - assign value to specified symbol

```
> (setf x 5)
5
```

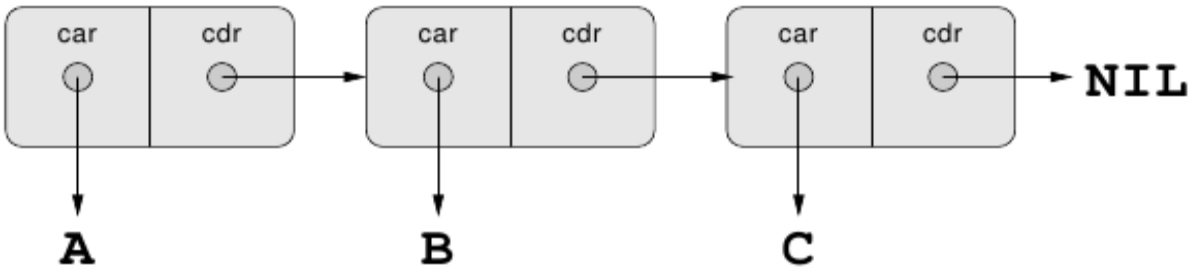
- `quote(')` - prevents the quoted arguments from being evaluated

```
> (quote (+ 2 3))
(+ 2 3)
> '(+ 2 3)
(+ 2 3)
```

Lists

Linked lists are the main data structure of Lisp

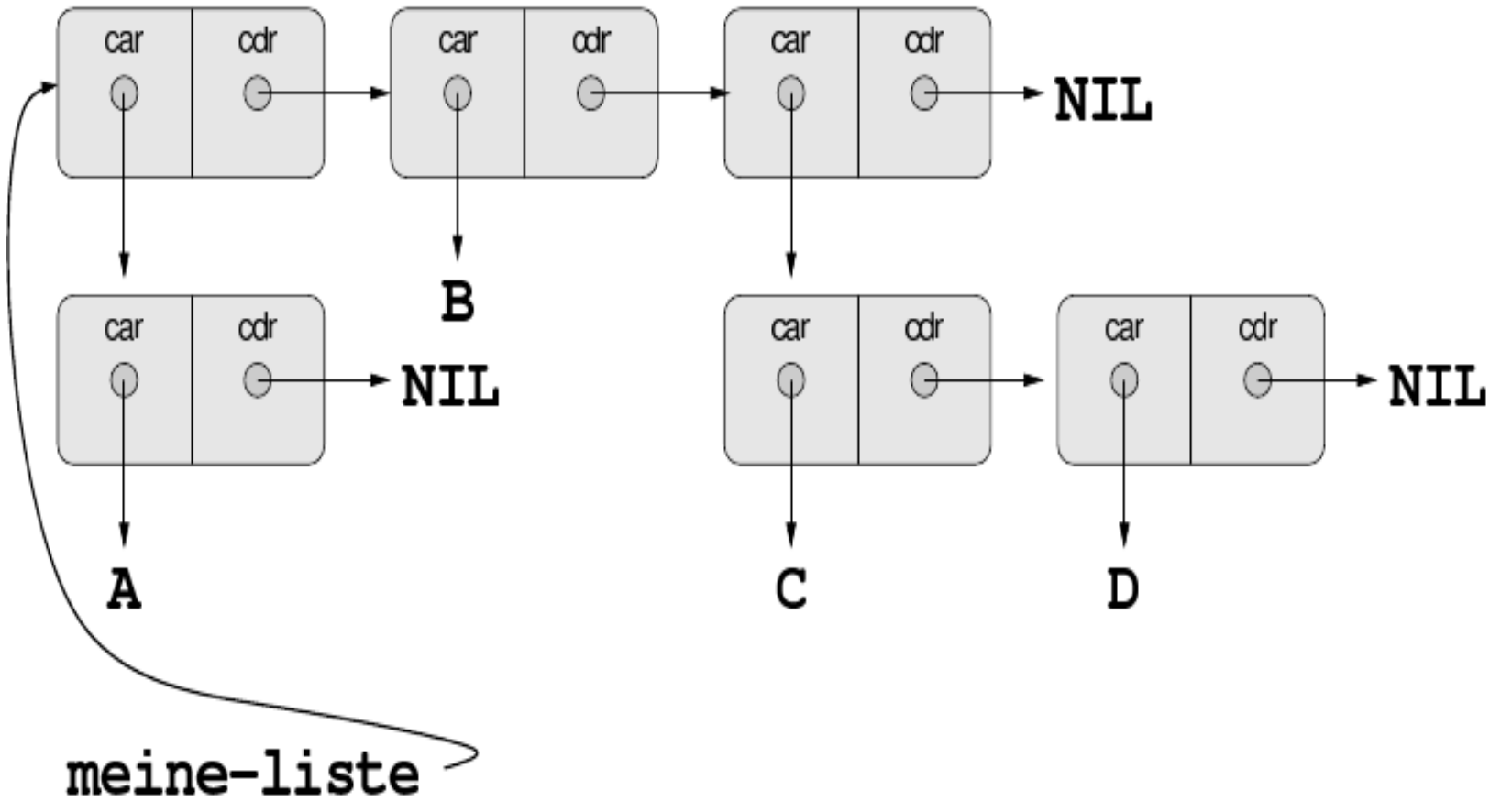
Example: (A B C)



- `car` - value pointer
  -> content of address register
- `cdr` - rest of the list
  -> content of decrement register
- `nil` represents empty list

Lists

Example: ((A) B (C D))



Basic Functions

- access to elements with `car` and `cdr`
- `list` - create new list

```
> (list 'a 'b 'c)
(A B C)
```

- `cons` - add element to list

```
> (cons 'd '(e f))
(D E F)
```

- `dolist` - iterate over list
- and many other functions...

Other featured data structures

- Arrays
- Hashtables
- ...

Variables

- lexical variables ("local variables")

Example

```
> (let ((x 5))
    (print x))
5
> x
Error: Attempt to take the value of unbound
variable 'X'.
```

- dynamic variables ("global variables")

Example

```
> (defparameter *globalvar* 5)
*GLOBALEVAR*
> *globalevar*
5
```

Functions

- define own functions with `defun`:

Example

```
> (defun add2 (x) (+ x 2))
> (add2 5)
7
```

Recursion

recursion is also supported:

Example

```
> (defun print-elements (list)
  (if (null list)
    (print nil))
  (print (car list))
  (print-elements (cdr list)))
> (print-elements '(a b c))
A
B
C
NIL
```

λ-Expressions

Similar notation in Lisp
Example: a function, which increases its argument by 2

λ-calculus

$\lambda x.x + 2$

Common Lisp

```
(lambda (x) (+ x 2))
```

-> anonymous funcions, used to construct complex functions

Higher-order functions

- Functions which either take functions as input or output a function
- Remember: function = data !

Example

```
> (apply #'+ '(1 2 3))
6
```

-> reader macro for executing code during runtime

Control Structures

- Booleans: `T` -> true, `NIL` -> false
- Conditionals: `if`, `cond`
- Loops: `do`, `dotimes`
- Blocks: `progn`

Macros

- Powerful functions which change the code of a program and influence its evaluation directly.
- Macros are created with `defmacro`

Example

```
> (defmacro set-nil (var)
.    (list 'setf var nil))
> (set-nil x)
NIL
```

Macros

What happens during runtime?

- Parser reads macro `(set-nil x)`
- New code is generated (macro expansion)
  `(list 'setf var nil)` is transformed to `(setf var nil)`
- New code `(setf var nil)` gets evaluated instead of the macro

Macros can contain other macros, which contain other macros...
-> Many possibilities to manipulate code during runtime, very powerful!

Usage

- Lisp is the second-oldest high-level programming language in widespread use today
- Historically one of the languages of Artificial Intelligence
- USA: often first language learned
- Lisp dialect Clojure is growing in popularity

Example: 99 Bottles of Beer

```
;; Bottles by Rebecca Walpole (walpolr@cs.orst.edu)
;; Note: the  p takes care of plurals.
;;
(defun bottles (n)
"Prints the lyrics to '99 Bottles of Beer'"
(if (< n 1)
  (format t -%Time to go to the store. %")
  (progn (format t -%  a bottle :p of beer on the wall."n)
    (format t -%  a bottle :p of beer."n)
    (format t -% Take one down, pass it around.")
    (format t -%  a bottle :p of beer on the wall. %"(- n
1))
    (bottles (- n 1))
    )
  )
)
(bottles 99)
```

Summary

- Lisp is functional (but also imperative and OO)
  -> λ-calculus
- Code = Lists = Data
- Lisp is a "programmable programming language"