Property lists

York University
Department of Computer Science and Engineering
Overview

• Properties for symbols
• Library example
• Components of a symbol
• Using setf with symbol components

[ref.: chap 7- Wilensky]
• Objects have properties, e.g. color, weight, etc.

• Symbols can have properties as well.
  – To get the value of the property ‘color’ for symbol ‘chair’:
    > (get ‘chair ‘color)
    NIL
  – To set the value of the property ‘color’ for symbol ‘chair’:
    > (setf (get ‘chair ‘color) ‘blue)
    BLUE
    > (get ‘chair ‘color)
    BLUE
Not set, or set to nil?

• Assume we set the following properties:

\[
\begin{align*}
&> (\text{setf} \ (\text{get} \ ‘\text{food1} \ ‘\text{taste}) \ ‘\text{sour}) \\
&SOUR \\
&> (\text{setf} \ (\text{get} \ ‘\text{food2} \ ‘\text{taste}) \ ‘\text{sweet}) \\
&SWEET \\
&> (\text{setf} \ (\text{get} \ ‘\text{food2} \ ‘\text{peanutfree}) \ \text{nil}) \\
&NIL
\end{align*}
\]

• If we access the value of properties:

\[
\begin{align*}
&> (\text{get} \ ‘\text{food1} \ ‘\text{peanutfree}) \\
&NIL \\
&> (\text{get} \ ‘\text{food2} \ ‘\text{peanutfree}) \\
&NIL
\end{align*}
\]

Nil means “not set”.

Nil means “set to nil”.

Nil means "not set". Nil means "set to nil".
Not set, or set to nil? (cont.)

• To distinguish, we can use `get` with 3 arguments

  > (get 'food1 'peanutfree 'unknown)
  UNKNOWN

  > (get 'food2 'peanutfree 'unknown)
  NIL

If third parameter is returned, it means “not set”.

Nil means actually “set to nil”.

• The third argument is an **optional** argument, while the first two arguments are **required** arguments.
Library example

• We can use a global variable `library` to store the list of books.

• A function to **add a book**:

  ```lisp
  > (defun addbook (bookref newtitle newauthor)
          (setf (get bookref 'title) newtitle)
          (setf (get bookref 'author) newauthor)
          (setq library (cons bookref library))
          bookref)

  ADDBOOK
  ```
Library example (cont.)

> (setq library nil)
NIL

> (addbook 'book1 '(common lispcraft) '(robert wilensky))
BOOK1

> (addbook 'book2 '(programming in prolog) '(william clocksin))
BOOK2

> library
(book2 book1)

> (get 'book1 'author)
(ROBERT WILENSKY)

Three argument, last two are lists
Adding to the front of list library
Properties are set globally! (we will see why shortly)
Library example (cont.)

- A function to retrieve information, for example:

  > (retrieveby ‘author ‘(robert wilensky))
  (BOOK1)

  > (defun retrieveby (property value)
      (do ((lst library (cdr lst))
          (result nil (if (equal (get (car lst) property) value)
                     (cons (car lst) result)
                     result)))
          ((null lst) result)))

  - Two index variables: lst and result
  - Variable lst is initially set to library. In each loop, the head is checked, and then it is set to the tail
  - Variable result is initially set to nil (empty list). In each loop, if a relevant book is found it will be added to the front of result.
Library example (cont.)

- Exercises:

  1. Write a function that deletes from the library.
  2. Write a retrieving function that works if we have the value of the property partially, for example:
     > (retrieveby2 'author 'robert)
        (BOOK1)
  3. Write a function that retrieves books by searching in values of all properties, e.g.
     > (retrieveall 'robert)
        (BOOK1)
Uniqueness of symbols

- **Symbols** can refer to different **variables**.
- Properties are attributes of the symbol, **not** the variables it can refer to!
- Unlike the variables they refer to, symbols are **unique**.
- Therefore changes to properties of a symbol are not local, but are **global**.
Example

> (setq x 5)
5
> (setf (get 'x 'color) 'red)
RED

> (defun f1 (x) (setq x (+ x 2)) (setf (get 'x 'color) 'blue) 'done)
F1

> (f1 2)
DONE
> x
5
> (get 'x 'color)
BLUE

x: a global variable here

x: a formal parameter, therefore bound and local here

Changes to x inside f1 were local, value of global variable x not changed.

Changes to properties of x are global!
Four components of a symbol

[ http://xahlee.org/elisp/Symbol-Components.html]

• Each symbol in LISP has
  – Print name:
    a string, for reading and printing the symbol’s name
  – Value:
    The current value of the symbol as a variable
  – Function:
    The function definition for the symbol
  – Property list:
    The property list of the symbol
Four components of a symbol (cont.)

> (setq x 5)
5
> (defun x (y) (* 100 y))
X
> (setf (get 'x 'comment) '(this is a comment))
(THIS IS A COMMENT)

> (symbol-name 'x)
"X"
> (symbol-value 'x)
5
> (symbol-function 'x)
#<FUNCTION X (Y) (DECLARE (SYSTEM::IN-DEFUN X)) (BLOCK X (* 100 Y))>
> (symbol-plist 'x)
(COMMENT (THIS IS A COMMENT) SYSTEM::DEFINITION ((DEFUN X (Y) (* 100 Y)) ).
Using setf with symbol components

• Setf to set value (instead of setq)
  (setq x 20)  
  = (setf (symbol-value ‘x) 20)

• Setf to set property list
  (setf (get ‘chair ‘color) red)  
  (setf (get ‘chair ‘height) 50)  
  = (setf (symbol-plist ‘chair)  
        ‘(height 50 color red))

• Setf to set function definition (instead of defun)
  (defun f1 (x) (* x 100))  
  = (setf (symbol-function ‘f1)  
         (lambda (x) (* x 100)))