CSE 413: Programming Languages and their Implementation

Scheme - Lists

Hal Perkins
Spring 2011
(cons a b)

• Takes a and b as args, returns a compound data object that contains a and b as its parts
• We can extract the two parts with accessor functions car and cdr

(define a (cons 'x 'y))

(\text{a})

(\text{x})

(\text{y})
We can build arbitrary pairs with \texttt{cons}, but the workhorse data structures in Scheme are proper lists.
Lists

• By convention, a list is a sequence of linked pairs
  » \texttt{car} of each pair is the data element
  » \texttt{cdr} of each pair points to list tail or the empty list

\begin{figure}
\centering
\begin{tikzpicture}
\node (e) at (0,0) [draw, circle] {e};
\node (1) at (-1,-1) [draw] {1};
\node (2) at (-1,-2) [draw] {2};
\node (3) at (-1,-3) [draw] {3};
\node (4) at (1,-1) [draw] {};
\node (5) at (1,-2) [draw] {};
\node (6) at (1,-3) [draw] {};
\draw (e) -- (1);
\draw (1) -- (2);
\draw (2) -- (3);
\draw (4) -- (5);
\draw (5) -- (6);
\end{tikzpicture}
\end{figure}
nil

• if there is no element present for the car or cdr branch of a pair, we indicate that with the value nil

  » '() represents the empty list (quoted to prevent evaluation)

• (null? z) is true if z is '()

(define d (cons 'x '()))
(car d)
(cdr d)
(null? (car d))
(null? (cdr d))
List construction

(define e (cons 1 (cons 2 (cons 3 '()))))

(define e (list 1 2 3))
procedure list

(list a b c ...)

- list returns a newly allocated list of its arguments
  - the arguments can be atomic items like numbers or quoted symbols
  - the arguments can be other lists
- The backbone structure of a list is always the same
  - a sequence of linked pairs, ending with a pointer to null (the empty list)
  - the car element of each pair is the list item
  - the list items can be other lists
List structure

(define a (list 4 5 6))

(define b (list 7 a 8))
Examples of list building

(cons 1 (cons 2 '()))

(cons 1 (list 2))

(list 1 2)
How to process lists?

- A list is zero or more connected pairs
- Each node is a pair
- Thus the parts of a list (this pair, following pairs) are lists
- A natural way to express list operations?
(define (length m)
  (if (null? m)
      0
      (+ 1 (length (cdr m)))))

cdr down
sum the items in a list

(\texttt{add-items (list 2 5 4)})