Listen als Bäume

Geschachtelte Listen können als Bäume aufgefaßt werden. Mit **copy-tree** läßt sich eine Kopie eines solchen Baumes anlegen. Im Gegensatz zu **copy-list** werden dabei nicht nur die obersten **cons**-Strukturen neu angelegt, sondern rekursiv für jede Teilstruktur neue Verbindungen erzeugt.

Die Funktion **subst** kann dazu verwendet werden, die sämtlichen Auftreten einer Teilstruktur innerhalb eines Baumes durch eine andere Struktur zu ersetzen.

Listen als Mengen

Eine Menge läßt sich sehr einfach als Liste darstellen. Zur Behandlung solcher Strukturen stellt LISP spezielle Funktionen bereit.

Das Prädikat **member** testet, ob ein Objekt als Element in einer Menge auftritt. Der Konstruktor **adjoin** fügt ein Objekt neu in eine Menge ein, unter der Bedingung, daß das Objekt noch nicht Element der Menge ist. Daneben stehen wichtige Mengenoperationen zur Verfügung: **subsetq**, **union**, **intersection**, **set-difference**, **set-exclusive-or**.

Assoziationslisten

Eine Assoziationsliste (kurz *alist*) ist eine Liste von eingebetteten Sublisten, deren jeweils erstes Element als *Schlüssel* dient. Üblicherweise sind die Sublisten Paare, bestehend aus Schlüssel und Datum, die als **cons** realisiert sind.

acons Fügt ein neues Schlüssel-Datum-Paar in eine Assoziationsliste ein.

pairlis Aus einer Liste von Schlüsseln und einer separaten Liste korrespondierender Daten wird eine Assoziationsliste aufgebaut bzw. ergänzt.

assoc Schlüssel-orientierter Zugriff auf die entsprechende Subliste einer Assoziationsliste.

rassoc Wert-orientierter Zugriff, die Rolle von Schlüssel und Datum sind gerade vertauscht.

Beispiel:

```
(setq udo '((age . 28) (name udo p))
      maria (pairlis '(age friend) '(18 udo)))
(assoc 'age udo)  -->  (AGE . 28)
(setq udo (acons 'age 29 udo))
  -->  ((AGE . 29) (AGE . 28) (NAME UDO P))
(assoc 'age udo)  -->  (AGE . 29)
(cdr (assoc 'name udo))  -->  (UDO P)
(rassoc 'udo maria)  -->  (FRIEND . UDO)
(cdr (assoc 'age maria))  -->  18
```