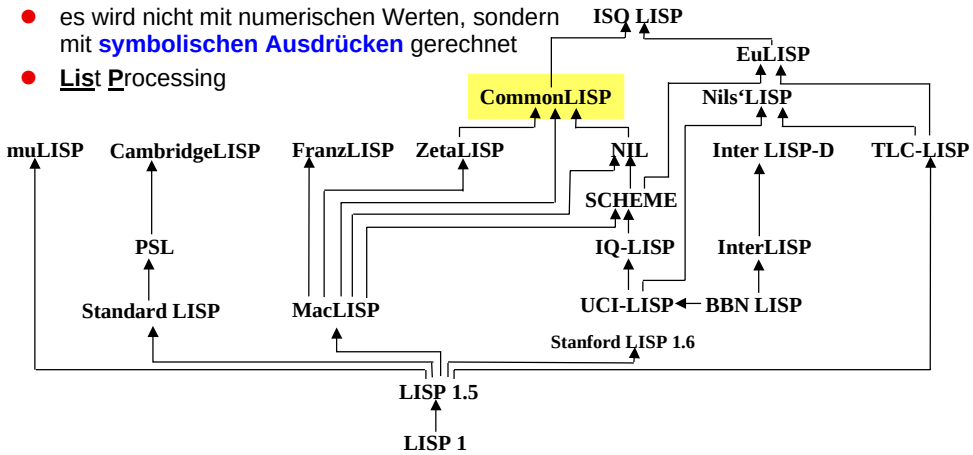
Applikative Programmierung in LISP I

- Allgemeines
- Listen für Daten und Programme
- Wertzuweisungen, -ermittlung, -interpretation
- Listenverarbeitung
- Prädikate

Historie

- **LISP**
  - die bekannteste **applikative** Sprache
  - 1958 von John McCarthy entwickelt
  - für Probleme der **KI** entwickelt
  - es wird nicht mit numerischen Werten, sondern mit **symbolischen Ausdrücken** gerechnet
  - **List Processing**



Allgemeines

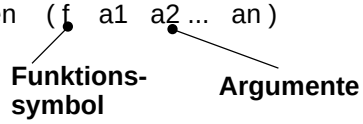
Charakterisierung

- **Vielfältiger Einsatz in der KI**
  - **Wissensverarbeitung**
  - **Bildverarbeitung / Bildverstehen**
  - **Übersetzung natürlicher Sprachen**
  - **Automatisches Beweisen**
  - **Symbolische Algebra etc.**
- **Interaktive Arbeitsweise**
  - Interpreter statt Compiler
- **LISP Programme und Daten haben dieselbe Form**
  - LISP-Programm kann andere Programme als Daten benutzen
- **Literatur:**
  - **Anderson/Corbett/Reisner: Essential LISP**
  - **Winston/Horn: LISP**
  - **Stoyan/Görz: LISP, eine Einführung in die Programmierung**
  - **Mayer: Programmieren in COMMON Lisp**

Listen für Daten und Programme

LISP-Programm

- **LISP – Programm:** **Folge von Ausdrücken, die nacheinander ausgewertet werden**
- **Ausdrücke**
  - Atome mit Wert
  - oder Listen (f a1 a2 ... an)



Wert des Ausdrucks ist der Funktionswert für diese Argumente

  - Präfixnotation, klammerlos

Listen für Daten und Programme

Beispiele für Ausdrücke

Prompt Ausdruck Kommentare

Ergebnis

```
----> (+  11  6)              ;  Addition
17
----> (*  7  8)
56
----> ( /  10.0  4)
2.5
----> (-  4711 )              ;  unäres Minus
-4711
----> (GCD  91  49)
7
----> (MAX 13  8  25)
25
----> (+ (+ 17.0  4) (EXPT  2  3))
29.0
```


Listen für Daten und Programme

Atom, Liste, Form

- **Atome**
  - Zahlenliterale
  - Symbole (Literale, Konstante, Variablen)
    `T, NIL, ANTON, ARGUMENT-1, X_17`
  - Zeichenketten(literale) „a b c d“ „~ A“
- **Liste**

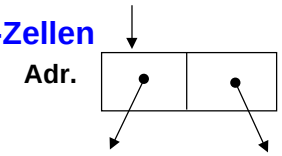
  (listelem listelem ... listelem)

  ( ) } leere Liste
  NIL }
- **Ausdrücke**
  - Atome oder Liste
- **Form**
  - auswertbarer Ausdruck

Listen für Daten und Programme

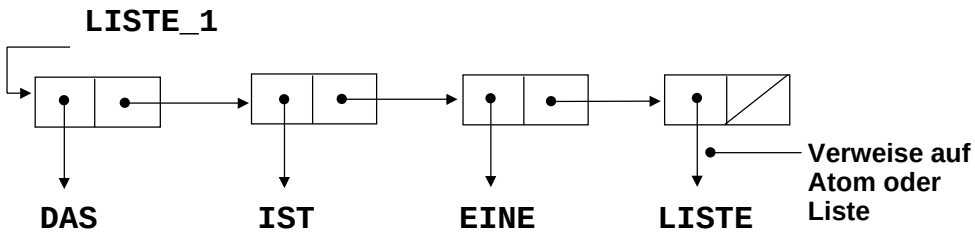
Interne Listendarstellung

- **Listen können durch CONS-Zellen repräsentiert werden.**



2 Inhalte jeweils Adressen (Zeiger)

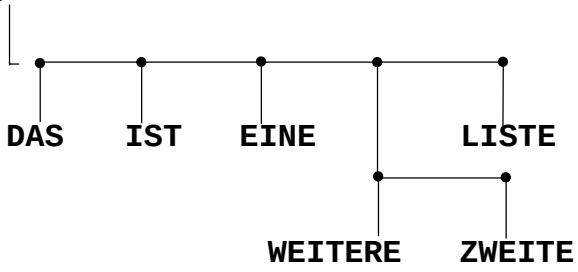
- **Liste**
  - Folge von CONS-Zellen, die über den rechten Zeiger verknüpft sind.
- **Beispiel**
  - `(DAS IST EINE LISTE)` sei Wert des Atoms `LISTE_1`



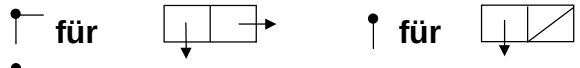
Listen für Daten und Programme

Liste für Daten

- **`(DAS IST EINE (WEITERE ZWEITE) LISTE)` abgekürzt notiert als**



- **Abkürzungen / Bedeutung**



Referenz auf den Wert: Atom, Liste

Übergang zu nächsten Listenelement

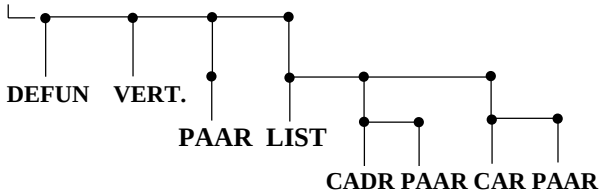
Listen für Daten und Programme

Liste für Funktion

- `(DEFUN VERTAUSCHE (PAAR)`

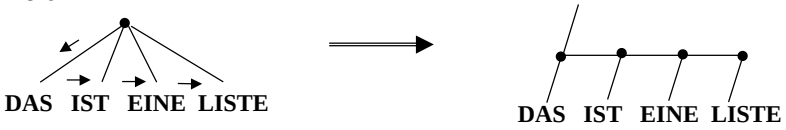
Liste für Funktion

`(LIST(CADR PAAR)(CAR PAAR))))`



- **Lisp - Liste und allgemeine Liste**

Lisp – Liste ist erster Sohn – nächster Bruder– Darstellung (Binärbaum-Darstellung) eines n-ären Baums (→ Datenstrukturen): Wert ins linke Zeigerfeld



Funktionen

Funktionen: Definition und Aufruf

- **Definition (Deklaration)**
  - `(DEFUN FName [Parlist] [Rumpf] )`
  - Auswertung:
    - Wert Funktionsname
      Seiteneffekt Zuordnung FName → Rumpf
- **Aufruf**
  - `(FName` a_1 a_2 … a_n `)` a_i Ausdrücke
  - Auswertung:
    - a_1, a_2 ... werden ausgewertet
    - Bindung an formale Parameter (der Reihenfolge nach)
    - Auswertung des Rumpfs
    - Wert des letzten Ausdrucks ist der Wert der Funktion
    - falls 0 ist Wert NIL
    - Bindungen werden aufgelöst

Wertzuweisung, -ermittlung, Interpretation

Identitätsfunktion

- **QUOTE ist die Identitätsfunktion**
- **Beispiel**
  - --> `(QUOTE (DA DA DA))`
  - `(DA DA DA)`
- **Achtung**
  - QUOTE wertet seine Argumente nicht aus!
  - Sonst wäre das Resultat von (QUOTE (DA DA DA)) nicht (DA DA DA). Sondern das Resultat
    - der Anwendung der Funktion DA
    - auf die Argumente DA DA
- **Abkürzung**
  - `(QUOTE A)` kann kürzer dargestellt werden als `'A`

Wertzuweisung, -ermittlung, Interpretation

Wertzuweisung

- **Die Wertzuweisung an ein symbolisches Atom geschieht durch die Systemfunktion** `SETQ`

```
(SETQ L '( A  B ))
```

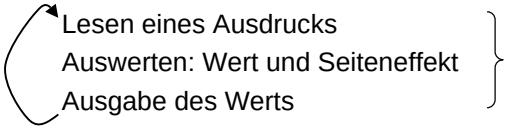
Quote-Zeichen für `(QUOTE(A B))`, Funktion mit Seiteneffekt

- **Beispiele**
  - `(SETQ x 5)`
  - `(SETQ y x)`
  - `(QUOTE x)`
  - `(SETQ y 'x)`
  - `y`
  - `(+ 'x 7)`
  - `'(+ 5 7)`
  - `('+ 5 7)`
  - `(+ '7 '4)`

Wertzuweisung, -ermittlung, Interpretation

Ausführung eines Programms

- **Nacheinanderauswertung der Ausdrücke durch Interpreter**



- **Seiteneffekte**
  - Zustand z: Gesamtheit von Bindungen von Symbolen an Werte
  - nach Ausdrucksermittlung, neuer Zustand z'
- **Änderungen von Bindungen nur über Seiteneffekte**

Wertzuweisung, -ermittlung, Interpretation

EVAL als Systemfunktion

```
----> (SETQ A 'B)   ; Zuweisung des Symbols und nicht des
B                   ; Werts von B an A
----> (SETQ B   '(+ 3 4))
(+ 3 4)
----> A
B
----> B
(+ 3 4)
----> (EVAL  A)
7
```

Auswertung von A

damit EVAL B

A
|
B
|
7

Listen verarbeitung

Motivation und Operationen

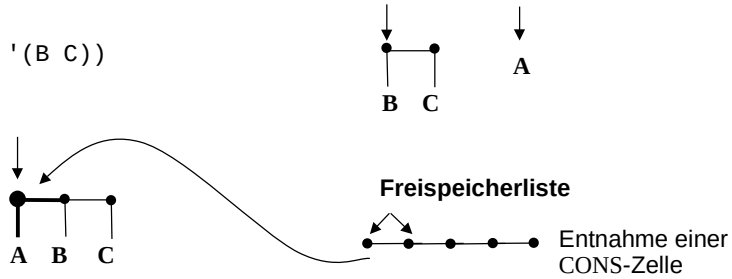
- **LISP Symbolverarbeitung**
  - abgebildet auf Listenverarbeitung
  - Grundmaschinerie jedes LISP-Systems
  - Notwendigkeit entsprechender Listenoperationen
- **Listenoperationen**
  - Aufbau
  - Zerlegung
  - Erweiterung
  - Spiegelung
  - etc.

Listen verarbeitung

Listenaufbau mit CONS

- **Aufbau**
  - `(CONS Ausdruck1 Ausdruck2)`
    Wert: Listenel1 Liste1
- **Ergebnis**
  - Die Liste, die durch Anhängen von Listenel1 am vorderen Ende von Liste1 entsteht
  - kein Seiteneffekt !!
- **Beispiel**

```
----> (CONS 'A '(B C))
```



Listen verarbeitung

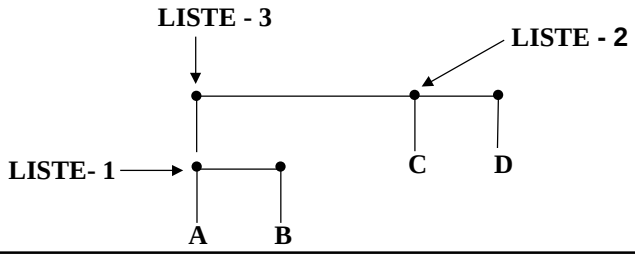
Weitere Beispiele mit CONS

```
-----> (CONS '(A B)'(C D))
((A B) C D)

-----> (SETQ LISTE_1 '(A B))
(A B)

-----> (SETQ LISTE_2 '(C D))
(C D)

-----> (SETQ LISTE_3 (CONS LISTE_1 LISTE_2))
((A B) C D)
```



Listen verarbeitung

Zerlegen von Listen mit CAR, CDR

- (CAR L)
  - liefert erstes Element, falls L nicht leer sonst ()
- (CDR L)
  - liefert Rest als Liste, leere Liste falls L leer war oder nur aus einem Element bestand
- **Beispiele:**

Keine Seiten-effekte

```
----> (CAR  '( (A B)  C))

----> (CDR  '((A B)  C))

----> (SETQ  SPRUCH  '(WISSEN IST MACHT))

----> (CAR  SPRUCH)

----> (CDR  SPRUCH)

----> (CAR  'SPRUCH)
```

Listen verarbeitung

Weitere Beispiele mit CONS, CAR, CDR

```
----> (CDR  '(CAR  '((A B)  C)))      ; Quotierung schützt '(CAR...)
((QUOTE  ((A B)  C)))                  ; vor Auswertung

----> (CAR  (CONS  'A  '(B C)))       ; Listen man. auf Programme
A                                      ; als Listen

----> (CDR  (CONS  'A  '(B C)))
(B C)

----> (CAR   (+  17  4))              ; Wert von (+ 17  4) ist Atom
ERROR

----> (CAR  '(+  17  4))
+

----> (CDR  '(+  17  4))
(17  4)
```


Listen verarbeitung

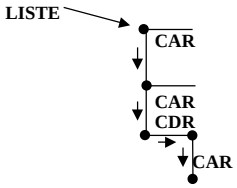
Zusammengesetzte Zerlegung: Zugriffspfade

- **Erweiterungen durch zusammengesetzte Zugriffsoperationen**

  `(CAR (CDR (CAR (CAR LISTE))))`

  abgek. zu `(CADAAR LISTE)`

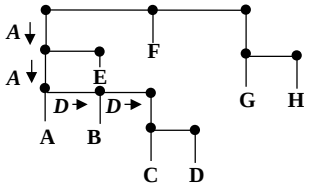
  bis zur Tiefe 4, Lesen von rechts nach links



- **Beispiel:**

  `(CDDAAR '(((A B (C D )) E) F (G H)))`

  `((C D))`



Listen verarbeitung

Zusammenhang

- (CONS 'Π '(1 2 ... n)) Wert (Π 1 2 ... n)
  (CAR '(Π 1 2 ... n)) Wert Π
  (CDR '(Π 1 2 ... n)) Wert (1 ... n)
- **LISP bietet bequemere Listenoperationen**
- **Diese sind alle mit Hilfe von `CONS, CAR, CDR` implementierbar**

Listen verarbeitung

Zusammenfügen durch APPEND

- `(APPEND L1 L2 .... Ln)` **Li hat als Wert eine Liste**
- **Ergebnis:**
  - Die Liste, die durch Aneinanderfügen der Elemente von Li gewonnen wird.
- **Beispiele:**

```
-->(APPEND '(A B) '(C D))
(ABCD)
-->(SETQ L '(A B))
(A B)
-->(CONS L L)
((A B) A B)
-->(APPEND L L)
(A B A B)
```

Listen verarbeitung

Zusammenfügen durch LIST

- `(LIST Ausdr-1 Ausdr-2 ... Ausdr-n)`
- **Ergebnis**
  - Liste der Werte der Argumente in der gegebenen Reihenfolge
- **Beispiele:**

```
--->(LIST '(A B) '(C D))
((A B) (C D))

--->(LIST L L)
((A B) (A B))
```


Listen verarbeitung

Weitere Beispiele

```
-----> ( CONS  'L  L)
( L  A  B)

-----> (APPEND 'L  L)            ; 'L keine Liste
ERROR

-----> (LIST  'L  L)
(L  (A B))

-----> (APPEND '(C)'() L '(D E))
(C A B D E)

-----> (LIST  L L  '(C D ))
(( A B  (AB)  (C D))

-----> (APPEND '() '() )
NIL

-----> (LIST   '() '() )
( NIL  NIL)
```

Listen verarbeitung

Weitere Funktionen zur Listenmanipulation

- **(REVERSE Liste)**

  Wert: Liste, die auf oberstem Niveau die Reihenfolge der Elemente invertiert, die Elemente bleiben unverändert

  ```
  --->(REVERSE '(A B))
  (B A)

  --->(REVERSE '((A B) (C D)))
  ((C D) (A B))

  --->(REVERSE (APPEND '(A B) '(C D)))
  (D C B A)
  ```

- **(LAST Liste1)**

  Wert: Liste, die das letzte Element von Liste1 als einziges Element enthält

  ```
  --->(LAST '(((A B) (C D)) (A B)))
  ((A B))
  ```


Prädikate und Strukturermittlung

Prädikate

- **Bedingungen**
  - werden durch Prädikate, die durch Junktoren verknüpft sind realisiert
- **Prädikat ist eine Funktion, die einen Wahrheitswert liefert**
- **Wahrheitswerte**
  - falsch NIL
  - wahr T
- **Strukturprädikate für Atome, CONS-Paare, Listen**
  - ist Atom
  - ist leer
  - ist Lisp-Ausdruck
  - Übereinstimmung von Listen

Prädikate und Strukturermittlung

Strukturprädikate für Atome

- `(ATOM Ausdruck)`

  Wert: T, falls der Wert von Ausdruck nicht mit CONS aufgebaut werden kann, NIL sonst

- `(CONSP Ausdruck)`

  Wert: T, falls der Wert von Ausdruck mit CONS aufgebaut werden kann, NIL sonst

  damit Präzisierung des Begriffs „Atom“

- **Beispiel:**

```
--->(ATOM 'NIL)
T
--->(ATOM T)
T
--->(ATOM 'ZWILLINGE)
T
--->(SETQ ZWILLINGE '(MAX MORITZ))
(MAX MORITZ)
--->(ATOM ZWILLINGE)
NIL
```

```
--->(CONSP ZWILLINGE)
T
--->(CONSP '(MAX MORITZ)
T
--->(ATOM '(+ 17 4))
NIL
--->(CONSP '(A B C))
T
--->(CONSP '17)
NIL
```


Prädikate und Strukturermittlung

Strukturprädikate für Ausdrücke

- **(NULL Ausdruck)**

Wert: T, falls Ausdruck als Wert die leere Liste hat, NIL sonst

```
-----> (NULL '() )
T
-----> (NULL NIL)
T
-----> (NULL 0)
NIL
-----> (NULL (CDDR '(A B)))
T
```

- **(LISTP Ausdruck)**

Wert: T, falls der Wert von Ausdruck das Prädikat CONSP erfüllt oder NIL ist, NIL sonst

```
-----> (LISTP 'T)
NIL
-----> (LISTP 'NIL)
T
-----> (LISTP '( ))
T
```

Prädikate und Strukturermittlung

Strukturprädikate für Listen

- **Übereinstimmung von LISP-Strukturen**

  Vergleich bzgl. interner Speicherstrukturen : EQ
  Vergleich bzgl. externer Darstellung : EQUAL

- `(EQ Ausdr1 Ausdr2)`

  Wert T, falls dem Wert der beiden Argumente dieselbe Speicherstruktur zugrundeliegt, NIL sonst

- `(EQUAL Ausdr1 Ausdr2)`

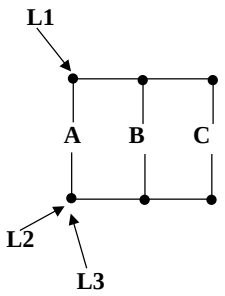
  Wert T, falls der Wert der beiden Argumente die gleiche externe Repräsentation besitzt, NIL sonst

Prädikate und Strukturermittlung

Beispiele zu Listenvergleich

- ------► (SETQ L1 (LIST 'A 'B 'C))
  (A B C)
  ------► (SETQ L2 (LIST 'A 'B 'C))
  (A B C)
  ------► (SETQ L3 L2)
  (A B C)

- ------► (EQ 'A 'A)
  T
  ------► (EQ 'A 'B)
  NIL
  ------► (EQ L1 L2)
  NIL
  ------► (EQ L2 L3)
  T
  ------► (EQUAL L1 L2)
  T
  ------► (EQUAL L2 L3)
  T
  ------► (EQ (CAR L1) (CAR L2))
  T



haben die gleiche Struktur, sind aber nicht die gleichen Speicherstrukturen

haben beide das Atom A als Wert; das hat eindeutige Speicherrepräsentation

Prädikate und Strukturermittlung

Leere Liste

- **Bemerkung**

  NIL und leere Liste -darg. ( )- sind äquivalent

- **Beispiel**

```
------>(EQ   NIL '( ))
  T                ; leere Liste bei Angabe stets durch NIL

------>(ATOM   '( ))
  T

                   ; Sei  a Ausdruck mit Wert als Liste
------>(ATOM a)
  T                ; g. d. w.   Liste leer
```

NIL ist der einzige Ausdruck der gleichzeitig Liste und Atom ist.

Prädikate und Strukturermittlung

Listenauskunftsfunktionen

- **(MEMBER Ausdruck Liste)**

  Wert: Liste, falls in der Liste (auf oberstem Niveau) ein Element mit dem Wert des Ausdrucks auftaucht; die Restliste ab erstem Auftauchen dieses Elements (incl.) NIL sonst

- **Beispiele:**

```
------>(SETQ   FUENF-X   '(X1 X2 X3 X4 X5))
(X1 X2 X3 X4 X5)

------>(MEMBER  'X6  FUENF-X)
NIL

------>(MEMBER  'X3  FUENF-X)
(X3 X4 X5)

------>(SETQ  A 'X4)
X4

------>(MEMBER A  FUENF-X)
(X4 X5)
```

Prädikate und Strukturermittlung

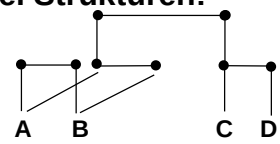
Weitere Beispiele - 1

```
-------> (MEMBER  A  '(( X1 X2)  (X3 X4 X5)))
NIL                                ; X4 nicht El. der zweielementigen Liste

-------> (MEMBER  A  (CDR  '(( X1 X2)  (X3 X4 X5))))
NIL                                ; X4 nicht El. von ((X3  X4 X5))

-------> (MEMBER  A  (CADR  '(( X1 X2)  (X3 X4 X5))))
(X4 X5)                            ; X4 ist El. von (X3 X4 X5)
```

Bemerkung 1:
MEMBER verwendet üblicherweise EQ bei Strukturen!
Problem, wenn erstes Argument eine Struktur ist.




```
-------> (MEMBER    '(A B)   '((A B) (C D)))
NIL
```

Erstes El. der zweiten Liste hat zwar dieselbe Struktur aber nicht dieselbe Speicherstruktur

Prädikate und Strukturermittlung

Weitere Beispiele - 2

- **Abhilfe**
  vorschreiben von EQUAL Strukturvergleich

  ```
  (MEMBER '(A B)   '((A B) (C D)) : Test  'EQUAL)
  ((A B) (C D))
  ```

  Systemprädikat kann auch selbstdefiniertes Prädikat sein

- **MEMBER ist kein Prädikat, liefert im positiven Fall Liste, kann aber als Prädikat verwendet werden, da für wahr irgendein Wert ≠ NIL genommen wird**

- **Probleme:**

```
---> (EQ   T   (MEMBER   'Y   '(X  Y  Z)))
NIL
---> (EQUAL  T   (MEMBER  'Y   '(X  Y  Z)))
NIL
```

Prädikate und Strukturermittlung

Längenbestimmung

- **(LENGTH Liste)**

  Wert: Anzahl der Elemente der Liste

- **Beispiel:**

```
------> (SETQ   L  '(A B))
(A B)
------> (LENGTH    L)
2
------> (LENGTH '((A B)  (C D)))
2
------> (LENGTH   (APPEND   L '(A B)))
4
------> (LENGTH   LIST   L   '((A B)))
2
------> (LENGTH   (MEMBER   'X3   '(X1 X2 X3 X4 X5)))
3
```


Prädikate und Strukturermittlung

Überprüfung/Vergleich von Atomen

- **Wertvergleiche**
  - Zahlen ganze Zahlen
    Gleitpunktzahlen verschiedener Genauigkeit
    komplexe Zahlen
  - Ergebnis von EQ nicht definiert, da Zahlen des gleichen Typs und mit gleichem Wert nicht notwendigerweise im gleichem Speicherbereich untergebracht sind.
  - Vergleich mit EQUAL liefert für Zahlen des gleichen Typs und Werts passendes Ergebnis
- **Beispiele:**

```
--->(EQ   2.0  2)              ; Interne Darstellung i.a. verschieden
NIL
--->(EQ   2.13  2.13)          ; nicht def.
NIL oder T                     ; systemabhängig
--->(EQUAL 2.0  2)             ; untersch.Typ
NIL
--->(EQUAL   4.5  (+ 2.3  2.2))
T
--->(EQUAL  4.5  '(+ 2.3  2.2)) )
NIL
```

Resumeé : EQ ungeeignet
EQUAL bedingt geeignet

Prädikate und Strukturermittlung

Spezielle Prädikate und Beispiele - 1

- **Spezielle Prädikate**

  (= Ausdr1 Ausdr2 ... AusdrN) alle Werte Zahlen sind gleich
  (< Ausdr1 Ausdr2 ... AusdrN) „ „ „ aufsteigend geordnet
  (> Ausdr1 Ausdr2 ... AusdrN) „ „ „ absteigend geordnet
  sonst NIL

- **Aufruf mit einem Argument, Ergebnis T**

  =,<, > haben keine Seiteneffekte, die Auswertung der Ausdrücke kann welche produzieren

- **Beispiele:**

```
---> (=  2.0  2.00  2)     ; Typen müssen nicht notwendigerweise übereinstimmen
 T
---> (< -1.8   0   2.3  4)
 T
---> (>  17)
 T
---> (= 17.0  (SETQ  X ( +  13  4)
 T
```


Prädikate und Strukturermittlung

Weitere Prädikate zu Vergleich

weitere Prädikate

(/= a1 a2 ... an)
(<= a1 a2 ... an)
(>= a1 a2 ... an)

} **Semantik analog zu oben**

Prädikate und Strukturermittlung

Artvergleiche

- **(NUMBERP Ausdr)**

  Wert T, falls Wert von Ausdruck numerisch ist, d.h. externe Zahlendarstellung besitzt, NIL sonst

```
-->(NUMBERP 'X)
NIL
-->(NUMBERP (+ 17 4))
T
```

- **(SYMBOLP Ausdr)**

  Wert T, falls Wert von Ausdruck ein Symbol ist
  NIL sonst

- **(STRINGP Ausdr)**

  Wert T, falls Wert von Ausdruck Zeichenkette als Wert hat
  NIL sonst

- **Beispiele:**

```
----> (SYMBOLP 3.0)        ----> (SYMBOLP 'AB)
NIL                        T
----> (STRINGP 'AB)        ----> (STRINGP "A0")
NIL                        T
```


Prädikate und Strukturermittlung

Komplexe Prädikate

- **Zusammengesetzt aus vordef. Prädikaten (Struktur- und Wertevergleich) und Junktoren AND, OR, NOT: spez. Formen**

- **(AND A1 A2 . . . An)**
  Wert: Anw. A_i
  NIL, falls ein A_i NIL, sonst Wert An
  n=0 Wert T

- **(OR A1 A2 . . . An)**
  Wert: Anw. A_i
  falls ein A_i Wert ≠ NIL, dieser Wert
  NIL sonst
  n=0 Wert NIL

- **(NOT A)**
  Wert: NIL, falls Wert von A ≠ NIL
  T sonst

n≥0

Auswertung von li nach re

Kurzschlußauswertung:
Gefahr Seiteneffekte, nicht durchgeführt

Prädikate und Strukturermittlung

Beispiele - 1

```
------► (AND (NUMBERP 3) (SETQ ACHTUNG 'SEITENEFF) (NULL 'T))
NIL

------► (AND (NUMBERP 3) (NULL 'T) (SETQ ACHTUNG 'K- SEITENEFF))
NIL

------► (AND)
T

------► (AND (NULL '() ) (MEMBER 'X3 '(X1 X2 X3 X4 )))
(X3 X4)

------► (OR (MEMBER 'X6 '(X1 X2 X3 )) (NULL '0))
NIL

------► (OR (MEMBER 'X3 '(X1 X2 X3 )) (NULL 'NIL))
(X3)

------► (OR (NULL 'NIL) (MEMBER 'X1 '(X1 X2 X3 )))
T

------► (OR)
NIL
```