# Logische Sprachen

## Prolog:

In Fakten, Regeln: bisher Arbeiten mit Datenstrukturen fester Größe

```
datum(9,1,2013).

autor(werner,meier).
autor(X,Y) :- buch(_,_,autor(X,Y)).

buch(prolog, springer, autor(hugo,mueller)).

?- autor(X,Y).
X = werner
Y = meier

X = hugo
Y = mueller
```

# Logische Sprachen

## Prolog:

Datenstruktur, deren Größe im vorhinein unbekannt ist: Listen

- Leere Liste: []
- Struktur mit Funktor . und 2 Argumenten: .(a,[])
- 1. Argument: Element der Liste, 2.Argument: Liste
- Beispiele:
  .(x, .(y,[]))
  .(a,.(b, .(c,[])))

- Einfachere Schreibweise (Listennotaten):
  [] [1] [x,y] [a,b,c] [[1,2,3],[x,y,z]]

# Logische Sprachen

## Prolog:

Operationen auf Listen:

1. Aufspaltung in Listenkopf und Rest:
   [1|[]]  [x|[y]]  [a|[b|[c]]]

   dabei Mischformen erlaubt:
   [a,b,c]  entspr.  [a|[b,c]]  entspr. [a,b|[c]] entspr. [a,b,c|[]]
   entspr. [a,b|.(c,[])]

2. member(El,Liste)  beweisbar, wenn El Element der Liste
3. append(L1,L2,L3)  beweisbar, wenn L3 Konkatenation von L1 und L2
4. sort(L,Sorted)  beweisbar, wenn Sorted sortierte L-Liste

Über welche Regeln lassen sich member, append, sort definieren?

# Logische Sprachen

member(El, Liste):

```
%member(X,L): X ist Element der Liste L
member(X,[X|Rest]).
member(X,[Y|Rest]):- member(X,Rest).

?-member(anton,[gerd,eva,anton,bernd,susan]).
Yes

?-member(X,[a,b,c]).
X=a  ;
X=b  ;
X=c  ;
fail.
```

# Logische Sprachen

```
append(X,Y,Z):
/*Z ist die Liste, die entsteht, wenn man die Elemente der Liste X von
      hinten der Reihe nach vor die Liste Y anhaengt */

append([],L,L).
append([X|L1],L2,[X|L3]):- append(L1,L2,L3).

?- append([a,b], [c,d], L).
L =  [a,b,c,d]

?-append(X,[c,d],[a,b,c,d]).
X=[a,b] ;
fail.

?-append(X,Y,[a,b,c,d])
X=[], Y=[a,b,c,d]  ;
X=[a], Y=[b,c,d]  ;
…
```

# Logische Sprachen

Weitere Prädikate auf Listen:

| | | |
|---|---|---|
| 5. | is_list(term) | beweisbar, wenn term Liste ist |
| 6. | reverse(L1,L2) | beweisbar, falls L2 die Umkehrung von L1 ist |
| 7. | length(L,X) | beweisbar, falls X die Anzahl der Elemente von L ist |
| 8. | delete(L1,X,L2) | beweisbar, falls L2 L1 ohne X ist |

…