Überblick

12 Logische Programmierung

- Prolog
- Syntax und Semantik

Syntax von Prolog

Atome sind

1. String aus Buchstaben, Ziffern und _, angefangen mit Kleinbuchstaben.
   Beispiele: *petik*, *xB7*, *a_HOPP*
2. String aus Sonderzeichen.
   Beispiele: `===>` , `.--.`, `:=:`
3. String in einfachen Anführungszeichen.
   Beispiele: `'Petik'`, `'Hello, World'`

Syntax von Prolog

Variablen sind Strings aus Buchstaben, Ziffern und _, angefangen mit einem Großbuchstaben oder _

Beispiele: *X*, *X7*, *X_7*, *_anton*

Es gibt außerdem *Zahlen*.

Strukturen

Strukturen sind Objekte, die aus mehreren Komponenten bestehen.

Beispiele:

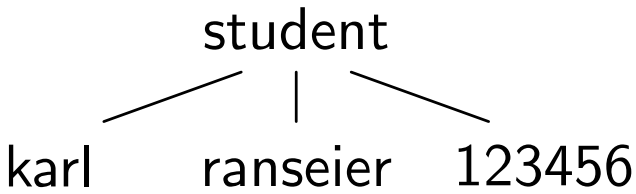
$student(karl, ranseier, 123456)$

$strecke(punkt(0,0), punkt(5,3))$

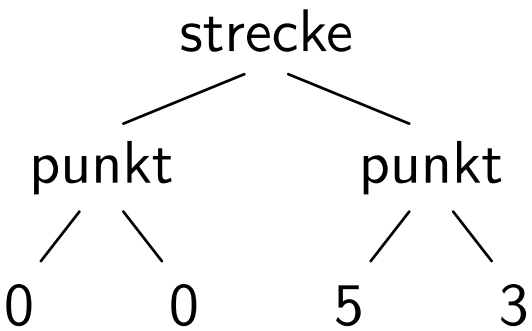
Der *Funktor* einer Struktur (z.B. *student*) ist ein Atom, die Komponenten sind Atome, Strukturen, oder Variablen.

Wir können Strukturen als Bäume darstellen:

$student(karl, ranseier, 123456)$



$strecke(punkt(0,0), punkt(5,3))$



Unifikation oder Matching

Sind S und T zwei Terme, so ist $S = T$ eine Anfrage, sie zu *unifizieren*:

- Sind S und T Konstanten (Atome oder Zahlen), dann unifizieren sie genau dann, wenn sie identisch sind.
- Ist S eine Variable, dann unifizieren S und T und S wird zu T *instantiiert*: Ab jetzt wird S stets durch T ersetzt:
- Ist T eine Variable dann ebenso.
- Sind S und T Strukturen, dann unifizieren sie, falls die Funktionen identisch sind und alle Komponenten unifizieren.

Unifikation – Beispiele

- $X = adam$ $\surd$
- $punkt(X, 7) = punkt(5, Y)$ $\surd$
- $punkt(X, 7) = punkt(7, X)$ $\surd$
- $punkt(X, 7) = punkt(5, X)$ –
- $X = punkt(Y, 3)$ $\surd$
- $strecke(X, Y) = strecke(Z, punkt(3, 5))$ $\surd$
- $strecke(X, Y) = X$?

Welche Instantiierungen finden statt?

Backtracking

Prolog versucht mit Backtracking, Anfragen mit seinen gespeicherten Deklarationen zu unifizieren. Scheitert dies an einer Stelle, dann werden die letzten Instantiierungen rückgängig gemacht und bei der nächsten Deklaration weitergemacht.

$punkt(3, 5).$
$punkt(1, 2).$
$punkt(2, 5).$
$nebeneinander(punkt(X1, Y), punkt(X2, Y)).$

Anfrage $punkt(X, Y), punkt(Y, Z)$

Zuerst wird $punkt(X, Y)$ mit der ersten Zeile unifiziert. Dann wird versucht $punkt(Y, Z)$ ebenfalls mit der ersten Zeile zu unifizieren, was scheitert, gefolgt von den anderen Zeilen. Schließlich scheitert $punkt(Y, Z)$ endgültig. Jetzt wird die Instantiierung von X, Y zurückgenommen und $punkt(X, Y)$ mit der zweiten Zeile unifiziert. Schließlich unifiziert $punkt(Y, Z)$ mit der letzten Zeile.

Backtracking

$punkt(3, 5).$
$punkt(1, 2).$
$punkt(2, 5).$
$nebeneinander(punkt(X1, Y), punkt(X2, Y)).$

Bei der Anfrage $nebeneinander(A, B)$ sind die ersten drei Deklarationen *belanglos*.

$nebeneinander(A, B)$ wird mit der vierten Zeile erfolgreich unifiziert.

Dabei wird $A = punkt(X1, Y)$ und $B = punkt(X2, Y)$ instantiiert.

Backtracking

$punkt(3,5).$
$punkt(1,2).$
$punkt(2,5).$
$nebeneinander(punkt(X1,Y),punkt(X2,Y)).$

Die Anfrage
$nebeneinander(punkt(A,B),punkt(C,D)),punkt(A,B),punkt(C,D)$
liefert dagegen $(A,B)=(3,5)$ und $(C,D)=(3,5)$ als erste Lösung.

$nebeneinander(punkt(A,B),punkt(C,D))$ wird mit der letzten Zeile unifiziert, denn die Funktoren stimmen dort überein.

Was für Instantiierungen werden gemacht? Nur $B=D$.

Jetzt wird $punkt(A,B)$ mit der ersten Zeile unifiziert und $A=3,B=5$ instantiiert. Dann wird versucht $punkt(C,5)$ mit einer Deklaration zu unifizieren.

Listen

In Prolog wird $[X|Y]$ für eine Liste mit Kopf X und Schwanz Y geschrieben.
Die leere Liste ist $[\,]$.
Wir können die Abkürzung $[1,2,3,4,5]$ benutzen. Beispiel:

```
removed(X,[X|Y],Y).
removed(X,[X1|Y],[X1|Y_ohne_X]) :-
    X\=X1,
    removed(X,Y,Y_ohne_X).
perm([],[]).
perm([X|Y],Z) :- member(X,Z),removed(X,Z,Z1),perm(Y,Z1).
aufsteigend([]).
aufsteigend([_]).
aufsteigend([X|[Y|Z]]) :- X =< Y,aufsteigend([Y|Z]).
sortiert(X,Y) :- perm(X,Y),aufsteigend(X).
```

$\backslash =$ ist ungleich, $=<$ und $>=$ sind in Prolog $\leq$ und $\geq$.