Prolog cheatsheet

Sintaksa

```
% Konjunkcija (A & B)
A, B

% Disjunkcija (A | B)
A; B

% Implikacija (A => B)
B :- A % (pozor, zamenjal se je vrstni red, B <= A)
```

Eksistenčnega in univerzalnega kvantifikatorja ne pišemo. **Kvantificirane spremenljivko pišemo z velikimi črkami**.

Konstante, predikate in funkcije pišemo z malimi črkami.

```
liho(succ(X)) :- sodo(X).
sodo(succ(Y)) :- liho(Y).
sodo(zero).
```

Seznami

V Prologu ni tipov, lahko pa uporabljamo poljubne konstante in konstruktorje, le z **malimi črkami jih je treba pisati**. Seznamov ni treba vnaprej definirati, ni treba razlagati kaj sta `nil` in `cons`.

```
% seznam [a; b; c]
cons (a, cons (b, cons (c, nil)))
```

Relacija `elem`

`elem(X, L)` je relacija, ki ugotovi, ali dani `X` pripada danemu seznamu `L`:

```
% če smo našli X, je element seznama
elem(X, cons(X, _)).
% sicer iščemo v repu seznama
elem(X, cons(_, L)) :- elem(X, L).
```

Primer:

```
elem(X, cons(X, _)).
elem(X, cons(_, L)) :- elem(X, L).

join(nil, Y, Y).
join(cons(A, X), Y, cons(A, Z)) :- join(X, Y, Z).

% v seznamu imamo a dvakrat, zato dvakrat dobimo true
?- elem(a, cons(b, cons(a, cons(c, cons(d, cons(a, nil)))))).
true ;
true ;
false.

% vprašamo, kateri so elementi danega seznama
?- elem(X, cons(a, cons(b, cons(a, cons(c, nil))))).
X = a ;
X = b ;
X = a ;
X = c ;
false.

% vprašamo lahko celo, kateri seznami vsebujejo dani element
?- elem(a, L).
L = cons(a, _3234) ;
L = cons(_3232, cons(a, _3240)) ;
L = cons(_3232, cons(_3238, cons(a, _3246))) ;
L = cons(_3232, cons(_3238, cons(_3244, cons(a, _3252)))) ;
L = cons(_3232, cons(_3238, cons(_3244, cons(_3250, cons(a, _3258))))) ;
L = cons(_3232, cons(_3238, cons(_3244, cons(_3250, cons(_3256, cons(a,
_3264))))))) .
```

Relacija `join`

`join(X, Y, Z)` je relacija, ki stakne seznama `X` in `Y` v seznam `Z`:

```
% stik praznega seznama in Y je seznam Y
join(nil, Y, Y).
% vzamemo prvi element prvega seznama, dodamo v Z in nadaljujemo, dokler nam
ne zmanjka elementov prvega seznama, nato se Y samo še pripne (glej zgornjo
vrstico)
join(cons(A, X), Y, cons(A, Z)) :- join(X, Y, Z).
```

`join(X, Y, Z)` pomeni, da je `Z` enak stiku seznamov `X` in `Y`.

Vgrajeni seznami

- `[e_1, e_2, ..., e_m]` je seznam elementov `e_1`, `e_2`, ..., `e_m`
- `[e | l]` je seznam z glavo `e` in repom `l`
- `[e_1, e_2, ..., e_m | l]` je seznam, ki se začne z elementi `e_1`, `e_2`, ..., `e_m` in ima rep `l`

Za delo s seznami potrebujemo knjižnico `lists`:

```
:- use_module(library(lists)).
```

Ta že vsebuje relaciji `member` (ki smo jo zgoraj imenovali `elem`) in `append` (ki smo jo zgoraj imenovali `join`).

```
?- append([a,b,c], [d,e,f], Z).
Z = [a, b, c, d, e, f].
```

Vprašamo, kako razbiti seznam `[a, b, c, d, e, f]` na dva podseznama:

```
?- append(X, Y, [a,b,c,d,e,f]).
X = [],
Y = [a, b, c, d, e, f] ;
X = [a],
Y = [b, c, d, e, f] ;
X = [a, b],
Y = [c, d, e, f] ;
X = [a, b, c],
Y = [d, e, f] ;
X = [a, b, c, d],
Y = [e, f] ;
X = [a, b, c, d, e],
Y = [f] ;
X = [a, b, c, d, e, f],
Y = [] ;
false.
```

Enakost in neenakost

Enakost pišemo `s = t`, neenakost pišemo `s \= t`.

Programiranje z omejitvami

Za programiranje z omejitvami moramo v Prolog naložiti ustrezno knjižnico:

```
% za delo s končnimi domenami (celimi števili)
:- use_module(library(clpfd)).

% za delo z realnimi števili
:- use_module(library(clpr)).
```

Operatorji

V knjižnici CLP(FD) so naslednji aritmetični in primerjalni operatorji:

```
+   -   *   ^   min   max   mod   rem   abs   //   div
#=   #\=   #>=   #=<   #>   #<
```

Določanje domen spremenljivk

Domeno za posamezno spremenljivko določimo z:

```
A in 0..42
B in inf..sup
```

Za seznam spremenljivk hkrati:

```
[A, B, C] ins 1..10
```

Omejitev `all_distinct([A, B, C])` zagotovi, da imajo spremenljivke `A`, `B` in `C` **različne vrednosti**. S predikatom `label([A, B, C])` naročimo Prologu, da s preiskovanjem **našteje konkretne vrednosti spremenljivk**, ki ustrezajo vsem podanim omejitvam. Preden uporabimo `label`, morajo biti domene vseh spremenljivk omejene.