DD1361 Programmeringsparadigm HT16

Logikprogrammering 4

Dilian Gurov, TCS

Idag

PROLOG-specifika konstruktioner

- Negation, snitt
- Aritmetik, I/O
- Kontrollpredikat, metapredikat

Generera och testa

Läsmaterial

- Boken: kap. 5, 9, 10
- PROLOG-fil: misc.pl
- Handouts: Föreläsningsanteckningar

Negation i PROLOG

Kom ihåg den "closed world assumption" som PROLOG utgår ifrån: allt som PROLOG inte lyckas bevisa betraktas som falskt!

Negation i PROLOG betraktas som bevis-teoretisk negation, och inte som logisk negation:

```
snygg(kia).
osnygg(X) :- \+ snygg(X).
```

Vad blir svaret på:

```
?- osnygg(nisse).
?- osnygg(kia).
?- osnygg(X).
```

Negation i Prolog

Vi implementerade medlemstestet för listor så här:

```
in(H, [H | _]).
in(X, [_ | T]) :- in(X, T).
```

Vad blir svaret till:

```
?- in(X, [1, 2]), \+ in(X, Y).
```

Beskriv kontrollflödet (uppgift från omtentan 2016-03-15).

Snitt

Snitt (eng: cut) skär bort backtrackingen för predikatet.

Exempel:

```
max(X, Y, X) :- Y<X.
max(X, Y, Y).
```

Vad blir första och andra svaret på frågan:

```
?- max(5, 3, X).
```

Snitt

Snitt (eng: cut) skär bort backtrackingen för predikatet.

Exempel:

```
max(X, Y, X) :- Y<X.
max(X, Y, Y).
```

Vad blir första och andra svaret på frågan:

```
?- max(5, 3, X).
```

För att eliminera flera än ett svar:

```
maxCut(X, Y, X) :- Y<X, !.
maxCut(X, Y, Y).
```

Exempel från *Learn Prolog Now!*

```
s(X, Y) :- q(X, Y).
s(0, 0).

q(X, Y) :- i(X), !, j(Y).
q(4, 4).

i(1).
i(2).

j(1).
j(2).
j(3).
```

Vilka blir svaren på frågan:

```
?- s(X, Y).
```

Grönt och rött snitt

Vi skiljer mellan två typer av snitt:

- **grönt snitt**: ändrar inte utdatan, bara skär bort onödiga sökningar;
- **rött snitt**: påverkar utdatan!

Exempel på grönt snitt: `lookup(+D, +T)`

Exempel på grönt snitt: `lookup(+D, +T)`

```
lookupCut(D, leaf(D)) :- !.
lookupCut(D, branch(D, _, _)) :- !.
lookupCut(D, branch(_, TL, _)) :- lookupCut(D, TL), !.
lookupCut(D, branch(_, _, TR)) :- lookupCut(D, TR).
```

Exempel på grönt snitt: `lookup(+D, +T)`

```
lookupCut(D, leaf(D)) :- !.
lookupCut(D, branch(D, _, _)) :- !.
lookupCut(D, branch(_, TL, _)) :- lookupCut(D, TL), !.
lookupCut(D, branch(_, _, TR)) :- lookupCut(D, TR).
```

OBS: Begreppet kan bero på hur predikatet ska användas!

Exempel på rött snitt

Betrakta programmet:

```
snygg(kia).
isnygg(X) :- snygg(X), !, fail.
isnygg(X).
```

Vad blir svaret på:

```
?- isnygg(nisse).
?- isnygg(kia).
```

Vad åstadkommer predikatet `isnygg`?

Aritmetik i Prolog

Vanlig likhet = betyder syntaktiskt likhet, inte “samma värde”!

Speciella predikatet `is` utvärderar andra termen och unifierar med första. Obs: andra termen måste vara tillräcklig instansierad!

Dubbelsidig utvärdering och jämförelse: `=:=`.

Aritmetiska operatörer: `+`, `-`, `*`, `/`, `//`, `mod`, etc.

Olikheter: `<`, `>`, `=<`, `>=`, etc.

I/O i Prolog

Två nödvändiga predikat:

- `read(t)`: läs en term fram till nästa punkt och unifiera den med termen `t`;
- `write(t)`: skriv ut termen `t` till terminalen.

Kontrollpredikat

Vi betraktade redan `fail`, ett predikat som alltid misslyckas.

Predikatet `call(X)` betraktar termen som `X` är unifierad med som ett predikat.

```
snygg(kia).
not(X) :- call(X), !, fail.
not(X).
```

Vad blir svaret på:

```
?- not(snygg(nisse)).
```

Metapredikat

Två predikat:

- `assert(t)`: lägg klausulen `t` till programmet;
- `retract(t)`: ta bort klausulen `t` från programmet.

Vilka blir svaren på:

```
?- assert(cool(anna)).
?- assert(hip(X) :- cool(X)).
?- hip(anna).
?- retract(cool(anna)).
?- hip(anna).
```

Båda predikat finns i flera varianter.