Datalog

nejpoužívanější jazyk *pro výzkum* DDB

jen ta nejnutnější syntaxe, a tedy ne přátelský

vlastně *čistý* Prolog syntakticky i sémanticky

Syntaxe a sémantika

literál : jméno relace, proměnná, konstanta $p(X, a, Y)$

pravidlo : $p(X, Y)\ :-\ q(X, Z),\ r(Z, Y).$
konvence: $\forall X, Y \exists Z \mid p(X, Y)\ \Leftarrow\ q(X, Z),\ r(Z, Y).$

relace $p(X, Y)$ může být vyjádřena více pravidly: $=$ sjednocení

intuitivní sémantika: všechny dvojice (X, Y) získané výpočtem těl pravidel

bezpečná pravidla $=$ smysluplná pravidla : každá proměnná v hlavě pravidla se vyskytuje alespoň jednou v těle

fakt : literál jen s konstantami

Relace

konvence v Datalogu:

- vyjádřena buď jen fakty = **extensionální**
- nebo jen pravidly = **intensionální**

```
?- je to omezení
```

deduktivní databáze = extensionální + intensionální relace
tj. „konvenční" databáze rozšířená o pravidla

query : konjunkce literálů $p(X, Y), q(Y, a, Z), r(Z)$

Sémantika pevného bodu

pravidla se interpretují jako *funkce generující fakta* f_{IDB}

opakované použití f_{IDB} na EDB a dříve generovaná fakta
dokud není generován žádný fakt, tj. $f_{IDB}(EDB^*) = EDB^*$

Příklad

```
p(X,Y) :- q(X,Y).              (1)
p(X,Y) :- q(X,Z),p(Z,Y).       (2)

q(1,2). q(2,3). q(3,2).      = EDB0

q(1,2). q(2,3). q(3,2).
p(1,2). p(2,3). p(3,2).      appl. (1)  = EDB1

q(1,2). q(2,3). q(3,2).
p(1,2). p(2,3). p(3,2).
p(1,3). p(2,2). p(3,3).      appl. (2)  = EDB2 = EDB*
```

Způsoby výpočtu datalogovských programů

1. **Top-down** = Prolog spojený s databází

   výhoda: snadná a rychlá impementace

   výpočet = SLD-rezoluce

   nevýhoda: tuple-at-time => neefiktivní
   a možná nekonečný výpočet

2. **Top-down se zapamatováním** = řešení problému neukončení

   SLD-rezoluce
   + odpovědi na podcíle jsou ukládány do paměti
   + tyto odpovědi jsou použity nejdříve

   = mezi top-down a bottom-up

3. **Bottom-up**

   = forward chaining, fixpoint

   nevýhoda: odvození mnoha faktů, které nejsou užitečné pro vlastní řešení dotazu <= ignorují se hodnoty konstant v dotazu

   řešení: magické množiny

Magické množiny

= odvozují se jen fakta nutná pro výpočet dotazu *(Bancilhon 1986)*

Příklad: Program a dotaz

```
partof(X,Y) :- component(X,Y).
partof(X,Y) :- component(X,Z), partof(Z,Y).

?- partof(2,Y).
```

se transformují na

```
magic_partof(2).
magic_partof(Z) :- magic_partof(X), component(X,Z).

partof_m(X,Y) :- magic_partof(X), component(X,Y).
partof_m(X,Y) :- magic_partof(X), component(X,Z), partof_m(Z,Y).

?- partof_m(2,Y).
```