# magicDATALOG

Užívateľská príručka

Autor:
Štefan Zajíček

# Obsah

**1 Inštalácia** **3**

**2 Datalóg** **5**
2.1 Definície . . . . . 5
2.2 Syntax . . . . . 5
2.3 Sémantika . . . . . 6

**3 Užívateľské rozhranie** **7**
3.1 Triedy implementujúce magické transformácie . . . . 7
3.1.1 Použitie . . . . . 7
3.2 Triedy pre vyhodnocovanie programov v datalógu . . 9
3.2.1 Spoločné prvky . . . . . 9
3.2.2 Trieda NaiveEvaluation . . . . . 9
3.2.3 Trieda DoubledProgramEvaluation . . . . . 11
3.2.4 Trieda WellFoundedMagicSetEvaluation . . . 11
3.3 Napojenie na SQL . . . . . 13
3.3.1 Konfigurácia . . . . . 13

# Kapitola 1

# Inštalácia

Pre beh jednotlivých programov je nutné mať nainštalované prostredie Java Runtime Environment. Je možné ho stiahnuť na adrese *http://www.java.com/en/download/*

Pre správne fungovanie napojenia na SQL je potrebné mať nainštalovaný JDBC connector od autora implementácie SQL servera. Napr. connector pre MySQL je možné stiahnuť na adrese *http://dev.mysql.com/downloads/connector/j/5.1.html*.

Súbor ovládača (napr. mysql-connector-java-5.1.6-bin.jar) treba umiestniť niekam do CLASSPATH.

# Kapitola 2

# Datalóg

## 2.1 Definície

**Definícia 1** (Term)**.** Konštanty a premenné sú termy.

**Definícia 2** (Atóm)**.** Ak $p$ je n-árny predikát a $t\_0, \ldots, t\_n - 1$ sú termy, tak $p(t\_0, \ldots, t\_n - 1)$ je *atóm*.

**Definícia 3** (Literál)**.** Ak $q$ je atóm, tak $q$ aj $\neg q$ sú *literály* - $q$ je pozitívny literál a $\neg q$ je negatívny literál.

**Definícia 4** (Pravidlo)**.** Ak $q$ je atóm, a $p\_0, \ldots, p\_n$ sú literály, tak

$q \leftarrow p\_0, \ldots, p\_n$

je pravidlo.

**Definícia 5.** Termy, literály a pravidlá sú *uzavreté*, ak neobsahujú žiadne premenné.

**Definícia 6.** *Program* v datalógu je množina pravidiel.

## 2.2 Syntax

- premenné začínajú veľkým písmenom, konštanty iným znakom, ako veľkým písmenom, okrem zátvoriek, čiarky a bodky

- mená predikátov začínajú malým písmenom
- biely priestor sa ignoruje
- pravidlá (prípadne dotaz) sú oddelené bodkou
- negácia sa značí $\sim$
- pri každom atóme môže byť najviac jedno znamienko negácie
- každé pravidlo musí byť v samostatnom riadku
- dotaz je tvaru $? - predikat(arg_1, arg_2, \ldots, arg_n)$
- atómy v pravidlách môžu byť bez argumentov - zátvorky sa vtedy nepíšu.
- dotaz sa môže nachádzať kdekoľvek v programe

## 2.3 Sémantika

Uvažujeme dva druhy sémantiky. Sémantiku najmenšieho pevného bodu a well-founded sémantiku.

- Sémantika najmenšieho pevného bodu je implementovaná v triede *NaiveEvaluation*.
- Well-founded sémantika je implementovaná v tiedach *DoubledProgramEvaluation* a *WellFoundedMagicSetEvaluation*.

# Kapitola 3

# Užívateľské rozhranie

## 3.1 Triedy implementujúce magické transformácie

Magické transformácie sú implementované v triedach *Magic* a *GeneralizedMagic*

### 3.1.1 Použitie

Triedy *Magic* a *GeneralizedMagic* sú spustiteľné. Ako vstup vezmú program a vrátia program transformovaný magickou transformáciou. Pri spustení v príkazovom riadku očakávajú aspoň jeden parameter a to meno súboru obsahujúceho datalógový program. Tento parameter musí byť zadaný ako prvý.

Voliteľný je parameter *-o* nasledovaný menom výstupného súboru.

**Príklad 1.** *V príkazovom riadku zadáme*

java Magic negacia.txt

*Výstup je na obrázku 3.1*

*Program:*

———

*p(a):-q(a),∼r(a).*

*q(a):-∼p(a).*

*r(a).*

*?-q(a)*

*Magic program:*

———————

*q(a):-magic_q(a),∼p(a).*

*p(a):-magic_p(a),q(a),∼r(a).*

*r(a):-magic_r(a).*

*magic_p(a):-magic_q(a).*

*magic_q(a):-magic_p(a).*

*magic_r(a):-magic_p(a),q(a).*

*magic_q(a).*

*?-q(a)*

Obr. 3.1: Príklad výstupu

## 3.2 Triedy pre vyhodnocovanie programov v datalógu

Triedy implementujúce vyhodnotenie datalógového programu sú nasledovné: *NaiveEvaluation*, *DoubledProgramEvaluation* a *WellFoundedMagicSetEvaluation*.

### 3.2.1 Spoločné prvky

Všetky triedy sú spustiteľné v príkazovom riadku príkazom
*java <MenoTriedy> <parametre>*
*<MenoTriedy>* je jedno z nasledujúcich: *NaiveEvaluation*, *DoubledProgramEvaluation*, *WellFoundedMagicSetEvaluation*
*<parametre>* môžu byť nasledovné:

- *<Meno súboru>* - povinný parameter, musí byť uvedený ako prvý. Označuje meno súboru obsahujúceho datalógovský program, ktorý sa má vyhodnotiť.
- *<Meno súboru>* - ak nejde o prvý parameter, určuje binárny súbor obsahujúci EDB reláciu, ktorá sa má načítať pred vyhodnotením.
- *-o <Meno súboru>* - určuje výstupný súbor. Výstupný súbor môže byť uvedený najviac jeden
- *-sql <Meno súboru>* - určuje súbor s nastaveniami pripojenia do databázy (pozri časť 3.3). Môže byť uvedený najviac jeden

### 3.2.2 Trieda NaiveEvaluation

Slúži na nájdenie najmenšieho pevného bodu programu metódou naívnej evaluácie.

**Príklad 2.** *V príkazovom riadku zadáme*

java NaiveEvaluation same_generation.txt person.rel par.rel -o sg.rel

*Výstup je na obrázku 3.2. Program načíta EDB relácie* preson.rel *a* par.rel*, vyhodnotí* program same_generation.txt *a reláciu pre odpoveď na dotaz uloží do výstupného súboru* sg.rel.

*Program:*

———

*sg(X,X):-person(X).*

*sg(X,Y):-par(X,Xp),sg(Xp,Yp),par(Y,Yp).*

*?-sg(j,Y)*

*Loaded EDB table: person.rel*

*Loaded EDB table: par.rel*

*Iteration 0:*

*sg[[a, a], [b, b], [k, k], [c, c], [j, j], [d, d], [e, e], [f, f], [g, g], [h, h], [i, i]]*

*Iteration 1:*

*sg[[b, b], [a, a], [c, d], [c, c], [d, d], [d, c], [e, e], [e, d], [d, e], [f, f], [f, g], [g, f], [g, g], [f, i], [h, h], [h, i], [i, h], [i, i], [k, k], [j, j], [i, f]]*

*Iteration 2:*

*sg[[a, a], [c, d], [c, c], [e, e], [e, d], [g, h], [g, i], [g, f], [g, g], [i, h], [i, i], [k, k], [k, j], [h, f], [b, b], [d, d], [d, c], [d, e], [f, f], [f, g], [f, h], [f, i], [h, g], [h, h], [h, i], [j, k], [i, g], [j, j], [i, f]]*

*Answer: sg(j,Y) = sg[[j, k], [j, j]]*

*Writting output file: sg.rel*

Obr. 3.2: Príklad výstupu

### 3.2.3 Trieda DoubledProgramEvaluation

Ide o implementáciu Van Gelderovej sémantiky alternujúceho pevného bodu. Zostrojí sa zvojený program a striedavo sa vyhodnocujú jeho dve polovice, pričom vždy sa použijú výsledky predchádzajúceho vyhodnotenia druhej polovice. Striedavo sa tak vypočítava čiastocný model obsahujúci pozitívne fakty (definitely true) alebo komplement negatívnych faktov(possibly true). Po dosiahnutí pevného bodu dostávame well-founded model. V našej implementácii pod tým rozumieme zjednotenie definitely true a possibly true faktov. Fakty, ktoré sú v possibly true, ale nie sú v definitely true majú hodnotu unknown, fakty, ktoré nie sú v possibly true majú hodnotu false.

Použitie triedy je rovnaké, ako pre všetky triedy na výpočet datalógových programov, popísané v tomto manuále. Navyše trieda obsahuje voliteľný parameter *-bottomup* nasledovaný menom triedy, ktorá sa má použiť pre základný výpočet zdola-nahor pri výpočte zdvojeného programu. Táto trieda musí byť podtriedou triedy *NaiveEvaluation*. Ak nie je tento parameter zadaný, použije sa trieda *NaiveEvaluation*.

**Príklad 3.** *V príkazovom riadku zadáme*

java DoubledProgramEvaluation negacia2.txt

*Výstup je na obrázku 3.3.*

### 3.2.4 Trieda WellFoundedMagicSetEvaluation

Je implementovaná ako podtrieda DoubledProgramEvaluation a teda jej použitie je úplne rovnaké. Rozdiel je v spôsobe výpočtu. Najskôr sa aplikuje magická transformácia. Nasledújci výpočet prebieha v dvoch fázach. V prvej fáze sa vyráta well-founded model pomocou techniky zdvojeneho programu. Následne sa overí, či sú všetky magické fakty dvojhodnotové. Ak áno, výpočet skončí. Inak prejde do druhej fázy. V nej sa k už vyrátaným faktom pridajú všetky possibly true magické fakty a s touto inicializáciou sa spustí znova výpočet well-founded modelu pomocou zdvojeneho programu.

Trieda obsahuje navyše voliteľný parameter *-magic* nasledovaný menom triedy, ktorá sa má použiť pre realizáciu Magickej transformácie. Táto trieda musí byť podtriedou triedy *Magic*. Ak nie je tento parameter zadaný, použije sa trieda *Magic*.

**Príklad 4.** *V príkazovom riadku zadáme*

```
Program:
———
p(a):-q(a),~r(a).
r(a).
q(a):-~q(a).
?-p(a)

Doubled program:
———————
p(a):-q(a),~r'(a).
r(a).
q(a):-~q'(a).

p'(a):-q'(a),~r(a).
r'(a).
q'(a):-~q(a).

Initialization: [q'[[a]], r'[[a]]]

Definitely true: [r[[a]]]

Possibly true: [q'[[a]], r'[[a]]]
Definitely true: [r[[a]]]

Possibly true: [q'[[a]], r'[[a]]]
Definitely true: [r[[a]]]
Fixpoint reached
Well-founded model:
[q'[[a]], r[[a]], r'[[a]]]
Answer:p(a) = p[]
```

Obr. 3.3: Príklad výstupu

java WellFoundedMagicSetEvaluation negacia2.txt

*Výstup je na obrázku 3.4.*

## 3.3 Napojenie na SQL

### 3.3.1 Konfigurácia

Pre pripojenie k externej SQL databáze je nutné pri vyhodnocovaní pomocou tried *NaiveEvaluation*, *DoubledProgramEvaluation* a *WellFoundedMagicSetEvaluation* načítať konfiguračný súbor pomocou parametra *-sql*.

Syntax konfiguračného súboru je nasledovná:

*<parameter> = <hodnota>*

Ak <hodnota> chýba, automaticky sa použije defaultna hodnota.

Priklad:

*port =*

Pravá strana nie je zadaná, tak sa použije defaultny port 3306.

Ak nejaký parameter nie je zadaný vôbec, tak si ho program vypýta počas behu. Riadky začínajúce bodkočiarkou sú komentáre a program ich ignoruje.

**Príklad 5.** *Priklad: mysql.ini*

driverclassname = com.mysql.jdbc.Driver

host = localhost

port =

;databasename = datalog_edb

subprotocol = mysql

;user = datalog

;password = heslo

Popis parametrov:

- *driverclassname* - meno triedy, ktorá implementuje jdbc konektor. Pri použití *mysql-connector-java-5.1.6-bin.jar*, ako jdbc ovládača , môžeme ponechať tento parameter nezmenený.
- *host* - doménový názov, alebo ip-adresa sql servera
- *port* - port sql servera

*Program:*

———

*p(a):-q(a),∼r(a).*

*r(a).*

*q(a):-∼q(a).*

*?-p(a)*

*Phase 1:*

———

*Magic program:*

*p(a):-magic_p(a),q(a),∼r(a).*

*q(a):-magic_q(a),∼q(a).*

*r(a):-magic_r(a).*

*magic_q(a):-magic_p(a).*

*magic_r(a):-magic_p(a),q(a).*

*magic_q(a):-magic_q(a).*

*magic_p(a).*

*?-p(a)*

*Well-founded model:*

*[q'[[a]], magic_r'[[a]], magic_p[[a]], p'[[a]], magic_q'[[a]], magic_p'[[a]], r'[[a]], magic_q[[a]]]*

*Phase 1 answer:p(a) = p[]*

*Magic predicates are not two-valued*

*Phase 2:*

———

*Well-founded model:*

*[magic_r'[[a]], magic_p[[a]], magic_q'[[a]], magic_r[[a]], magic_p'[[a]], magic_q[[a]]]*

*[q'[[a]], r[[a]], r'[[a]]]*

*Answer:p(a) = p[]*

Obr. 3.4: Príklad výstupu

- *databasename* - názov databázy, ktorej tabuľky sa majú použiť pri výpočte.
- *subprotocol* - kazdý ovládač má vlastný subprotocol (pri použití mysql, ponecháme nezmenené)
- *user* - meno užívateľa pre prístup do databázy
- *password* - heslo užívateľa pre prístup do databázy