Laborator 1

2017-2018

Programare Logică

De ce programare logică (PL)?

- Programarea logică este utilă în strategii de căutare, prototipuri, rezolvare de puzzle-uri etc.
- Idei "declarative" apar în multe domenii din informatică:
  - concepte din PL sunt utile în inteligență artificială și baze de date
  - demonstratoare automate, *model-checking*, *constraint programming*
  - devin importante în analiza programelor, semantica Web
- Învățând o metoda foarte diferită "de a gândi probleme", deveniți programatori mai buni :)

Prolog

- Prolog este cel mai cunoscut limbaj de programare logică.
  - bazat pe logica clasică de ordinul I (cu predicate)
  - funcționează pe bază de unificare și căutare
- Multe implementări îl transformă în limbaj de programare "matur"
  - I/O, operații implementate deja în limbaj etc.

Prolog

- Vom folosi implementarea SWI-Prolog
  - gratuit
  - folosit des pentru predare
  - conține multe librării
  - `http://www.swi-prolog.org/`
- Varianta online SWISH a SWI-Prolog
  - `http://swish.swi-prolog.org/`

Laboratorul 1

TODO

- Cum arată un program în Prolog?
- Cum punem întrebări în Prolog?

Mai multe detalii

- `Capitolul 1` din *Learn Prolog Now!*.

kb1: Un prim exemplu

Exemplu

```
stark(eddard).
stark(jon_snow).
stark(sansa).

lannister(tyrion).
lannister(cersei).

dislike(cersei,tyrion).
```

Sintaxă: atomi

Constante/atomi:

- secvențe de litere, numere și _, care încep cu o literă mică
- șiruri între ' '
- anumite simboluri speciale

Exemplu

- `sansa`
- `jon_snow`
- `aerys2`
- `'Ser Gregor Clegane'`
- `'(@ *+ '`
- `  +`

Sintaxă: variabile

Variabile:

- secvențe de litere, numere și _, care încep cu o literă mare sau cu _
- Variabilă specială: _ este o variabilă anonimă
  - două apariții ale simbolului _ sunt variabile diferite
  - este folosită când nu vrem detalii despre variabila respectivă

Exemplu

- `X`
- `Arya`
- `_cersei`

Sintaxă: predicate

Predicate:

- Predicatele sunt de forma `p(t1,...,tn)` unde
  - `p` este un atom,
  - `t1,...,tn` sunt termeni.
- Un termen este un atom, o variabilă sau un termen.

Exemplu

- `dislike(cersei,tyrion)`
- `dislike(cersei,X)`

- Un predicat are
  - un nume: `dislike` în `dislike(cersei,tyrion)`
  - o aritate (numărul de argumente): 2 în `dislike(cersei,tyrion)`

Sintaxă: predicate

- Predicate cu același nume, dar cu arități diferite, sunt predicate diferite.
- Scriem `foo/n` pentru a indica că un predicat `foo` are aritatea `n`.
- Predicatele pot avea aritatea 0 (nu au argumente); sunt predefinite în limbaj (`true`, `false`).

kb2: Un exemplu cu fapte și reguli

Exemplu

```
eating(joffrey).
deceased(robert).
dislike(cersei,tyrion).

happy(cersei) :- happy(joffrey).
happy(ser_jamie) :- happy(cersei), deceased(robert).
happy(joffrey) :- dislike(joffrey,sansa).
happy(joffrey) :- eating(joffrey).
```

Sintaxă: reguli

O **regulă** este o afirmație de forma `Head :- Body.` unde

- `Head` este de forma `p(ts)`
- `Body` este de forma `q1(ts1),...,qN(tsN)`
- `p(ts),q1(ts1),...,qN(tsN)` sunt predicate.
- Intuiția: `p(ts)` este adevărat dacă `q1(ts1)` și ... și `qN(tsN)` sunt adevărate.

Exemplu

```
happy(ser_jamie) :- happy(cersei), deceased(robert).
```

Sintaxă: reguli

Mai multe reguli care au același `Head` trebuie gândite că au sau între ele.

Exemplu

```
happy(joffrey) :- dislike(joffrey,sansa).
happy(joffrey) :- eating(joffrey).
```

Dacă `dislike(joffrey,sansa)` este adevărat sau `eating(joffrey)` este adevărat, atunci `happy(joffrey)` este adevărat.

Mai multe reguli cu aceeași parte stângă se pot uni folosind `;`.

Exemplu

```
happy(joffrey) :- dislike(joffrey,sansa) ; eating(joffrey).
```

Sintaxă: fapte

- Un **fapt** (*fact*) este o regulă fără `Body`.
- Un fapt specifică că o anumită instanță a unui predicat este adevărată.
- O colecție de fapte este numită și **bază de cunoștințe** (*knowledge base*).

Exemplu

```
stark(sansa).
lannister(tyrion).
dislike(cersei,tyrion).
```

Sintaxă: program

Un program în Prolog este o colecție de fapte și reguli.

Faptele și regulile trebuie grupate după atomii folosiți în `Head`.

Exemplu

Corect:

```
stark(eddard).
stark(jon_snow).
stark(sansa).

lannister(tyrion).
lannister(cersei).

dislike(cersei,tyrion).
```

Incorect:

```
stark(eddard).
dislike(cersei,tyrion).
stark(sansa).

lannister(tyrion).
stark(jon_snow).

lannister(cersei).
```

Sintaxă: ținte

- O **țintă** (*goal*) este o secvență de predicate, legate prin virgulă

```
p1(t1,...,tn),...,pn(t1',...,tn').
```

- Predicatele dintr-o țintă trebuie gândite că au și între ele.
- O țintă înseamnă că predicatul `p1` este adevărat pentru `t1,...,tn`, și similar pentru celelalte predicate.

Sintaxă: întrebări și răspunsuri

- O `întrebare` (*query*) este o secvență de forma `?- goal.`
- Fiind dată o întrebare (deci o țintă), Prolog caută răspunsuri.
- Prolog poate da două tipuri de răspunsuri:
  - `true`/valori pentru variabilele care apar în țintă
  - `false`
- Valorile găsite pentru variabile fac ținta adevărată.

Exemple de întrebări și răspunsuri

Exemplu

```
eating(joffrey).
deceased(robert).
dislike(cersei,tyrion).

happy(cersei) :- happy(joffrey).
happy(ser_jamie) :- happy(cersei),
                    deceased(robert).
happy(joffrey) :-
                    dislike(joffrey,sansa).
happy(joffrey) :- eating(joffrey).
```

```
?- happy(joffrey).
true

?- happy(cersei).
true

?- happy(ser_jamie).
true

?- happy(X).
X = cersei
X = joffrey
false
```

Întrebări în program

În SWI-Prolog puteți adăuga întrebări la finalul programului ca în exemplul de mai jos. În varianta online, întrebările vor apărea în lista din *Examples* (partea dreaptă).

Exemplu

```
eating(joffrey).
deceased(robert).
dislike(cersei,tyrion).

happy(cersei) :- happy(joffrey).
happy(ser_jamie) :- happy(cersei), deceased(robert).
happy(joffrey) :- dislike(joffrey,sansa).
happy(joffrey) :- eating(joffrey).

/** examples

?- happy(joffrey).
?- happy(cersei).
?- happy(ser_jamie).
?- happy(X).
*/
```

kb3: Un exemplu cu date și reguli ce conțin variabile

Exemplu

```
father(eddard,sansa).
father(eddard,jon_snow).

mother(catelyn,sansa).
mother(wylla,jon_snow).

stark(eddard).
stark(catelyn).

stark(X) :- father(Y,X),
            stark(Y).
```

Pentru orice `X`, `Y`, dacă `father(Y,X)` este adevărat și `stark(Y)` este adevărat, atunci `stark(X)` este adevărat.

Adică, pentru orice `X`, dacă tatăl lui `X` este `stark`, atunci și `X` este `stark`.

kb3: Un exemplu cu date și reguli ce conțin variabile

Exemplu

```
?- stark(jon_snow).
true

?- stark(X).
X = eddard
X = catelyn
X = sansa
X = jon_snow
false

?- stark(X), mother(Y,X), stark(Y).
X = sansa,
Y = catelyn
false
```

Diverse

- Un program în Prolog are extensia `.pl`
- Comentarii:

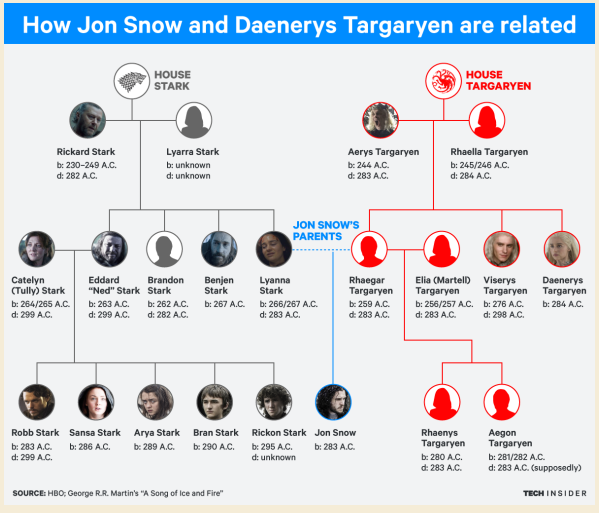
```
% comentează restul liniei

/* comentariu
pe mai multe linii */
```

- `not` - predicat predefinit în limbaj pentru negație.
  Exemplu: `not_parent(X,Y) :- not(parent(X,Y)).`
- `==` - pentru a verifica că doi termeni sunt egali.

Practică

Exercițiul 1: arbore genealogic

Folosiți predicatele `male/1`, `female/1` și `parent_of/2` pentru a reprezenta următorul arbore genealogic ca bază de cunostințe în Prolog.



`Aici` găsiți poza marită.

Practică

Exercițiul 1 (cont.)

Folosind predicatele `male/1`, `female/1` și `parent_of/2`, adăugați reguli pentru următoarele predicate:

- `father_of(Father, Child)`
- `mother_of(Mother, Child)`
- `grandfather_of(Grandfather, Child)`
- `grandmother_of(Grandmother, Child)`
- `sister_of(Sister,Person)`
- `brother_of(Brother,Person)`
- `aunt_of(Aunt,Person)`
- `uncle_of(Uncle,Person)`

Practică

Exercițiul 1 (cont.)

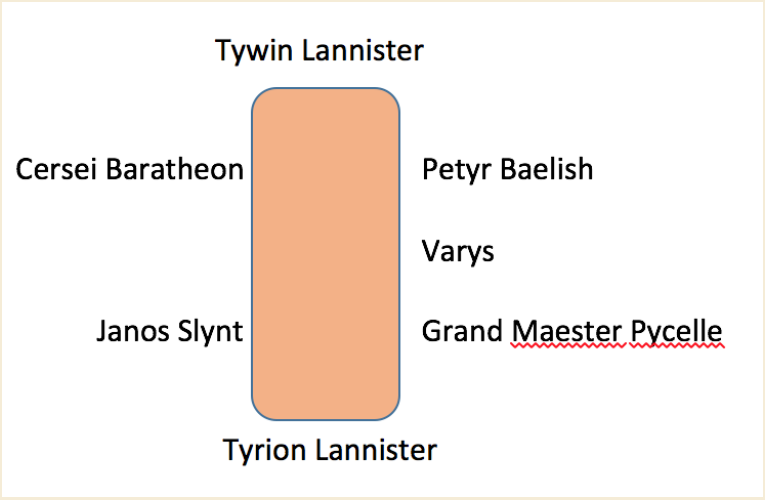
Verificați predicatele definite punând diverse întrebări.

Folosiți acest program pentru a afla dacă Daenerys Targaryen este matușa lui Jon Snow.

Practică

Exercițiul 2: consiliu

Imaginea de mai jos arată cum sunt asezați membrii consiliului lui Joffrey:



Definiți predicatul `sits_right_of/2` pentru a reprezenta cine lângă cine stă. `sits_right_of(X,Y)` trebuie să fie adevărat dacă `X` este la dreapta lui `Y`.

Practică

Exercițiul 2 (cont.)

Adăugați următoarele predicate:

- `sits_left_of/2`: `sits_left_of(X,Y)` trebuie să fie adevărat dacă `X` este la stânga lui `Y`.
- `are_neighbors_of/3`: `are_neighbors_of(X,Y,Z)` trebuie să fie adevărat dacă `X` este la stângă lui `Z` și `Y` este la dreapta lui `Z`.
- `next_to_each_other/2`: `next_to_each_other(X,Y)` trebuie să fie adevărat dacă `X` este lângă `Y`.

Practică

Exercițiul 2 (cont.)

Testați implementarea voastră punând următoarele întrebări:

- Este Petyr Baelish la dreapta lui Cersei Baratheon?
- Este Petyr Baelish la dreapta lui Varys?
- Cine este la dreapta lui Janos Slynt?
- Cine stă doua scaune la dreapta lui Cersei Baratheon?
- Cine stă între Petyr Baelish și Grand Master Pycelle?

Pe data viitoare!