Laborator 1 - Prolog

2016-2017

Programare Logică

De ce programare logica (PL)?

- Programarea logică este de multe ori foarte utilă în strategii de căutare, prototipuri, rezolvare de puzzle-uri etc.
- Idei "declarative" apar în multe domenii din informatică:
  - concepte din PL sunt foarte utile în inteligență artificială și baze de date
  - demonstratoare automate, *model-chekcing*, *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/`
- Vom folosi 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

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

lannister(tyrion).
lannister(cersei).

dislike(cersei,tyrion).
```

Sintaxă: atomi și variabile

Atomi:

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

Exemple: `sansa, jon_snow, khalDrogo, aerys2,`
`'Ser Gregor Clegane', '(@ *+ ', +`

Variabile:

- secvențe de litere, numere și _, care încep cu o literă mare sau cu _
- _ este o variabilă anonimă
  - două apariții ale simbolului _ sunt variabile diferite

Exemple: `X, Arya, _cersei`

Sintaxă: predicate

Predicate:

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

Exemple: `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)`
- 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

```
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: fapte

- Un **fapt** (*fact*) este o afirmație că o instanță a unui predicat este adevărată.
- Faptele trebuie urmate de punct!!
- O colecție de fapte este numită și **bază de cunoștințe** (*knowledge base*).

Exemple:

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

Sintaxă: reguli

- O **regulă** este o afirmație de forma

  ```
  p(ts) :- q1(ts1),...,qN(tsN).
  ```

  unde `p(ts)`,`q1(ts1)`,...,`qN(tsN)` sunt predicate.

- Intuiția: `p(ts)` este adevărat dacă `q1(ts1)` și ... și `qN(tsN)` sunt adevărate.

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

- Mai multe reguli trebuie gândite că au sau între ele.

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

Sintaxă: program

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

Sintaxă: ținte

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

```
p(t1,...,tn),...,q(t1',...,tn').
```

- O țintă înseamnă că predicatul `p` este adevărat pentru `t1,...,tn`, și similar pentru celelalte predicate.

Sintaxă: întrebări și răspunsuri

- O `întrebare` (*query*) este o secvență de forma `?- tinta.`
- Fiind dată o întrebare (deci o țintă), Prolog caută răspunsuri.
- Cele două răspunsuri posibile sunt:
  - `true` (posibil cu o substituție)
  - `false`
- Substituțiile dau valori variabilelor pentru a face ținta adevărată.

Exemple de întrebări și răspunsuri

```
?- happy(joffrey).
true

?- happy(cersei).
true

?- happy(ser_jamie).
false

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

kb3: Un exemplu cu date și reguli ce conțin variabile

```
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`.

Putem pune întrebări

```
?- 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
```

Sintaxă

- Comentarii:

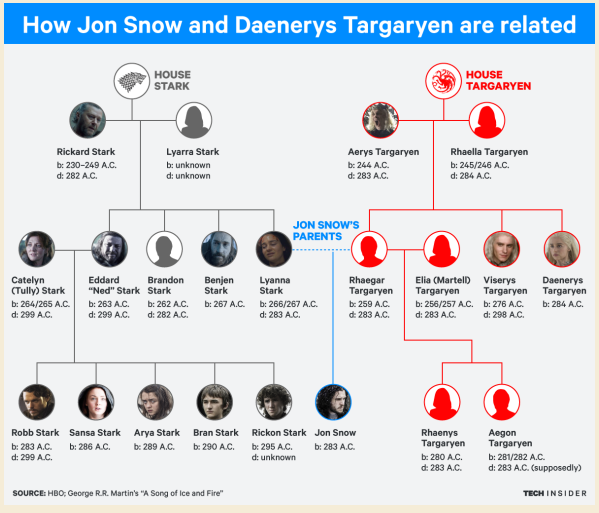
```
% comentează restul liniei

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

- `not` - predicat predefinit în limbaj pentru negație.
  Exemplu: `not_parent(X,Y) :- not(parent(X,Y)).`

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.

Practica

Exercițiul 1 (cont.)

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)`

Folosiți programul pentru a afla gradul de rudenie dintre Jon Snow și Daenerys Targaryen.

Pe săptămâna viitoare!