Laborator 3

2018-2019

Programare Logică

Laboratorul 4

TODO

- Unificare (prezentare pe scurt)
- Exerciții.

Cum răspunde Prolog întrebărilor

Unificare

- În acest laborator prezentăm doar intuitiv ce înseamnă un unificator.
- Mai multe detalii și algoritmul de unificare care găsește un unificator pentru o mulțime de termeni, sunt prezentate în cadrul cursului.

Unificare

- Prolog are un operator (infixat) pentru egalitate:
  `t = u` (sau echivalent `=(t,u)`)
- Ecuația $t = u$ este o țintă de bază, cu o semnificație specială.
- Ce se întâmplă dacă punem următoarele întrebări:

```
?- X = c.
?- f(X,g(Y,Z)) = f(c,g(X,Y)).
?- f(X,g(Y,f(X))) = f(c,g(X,Y)).
```

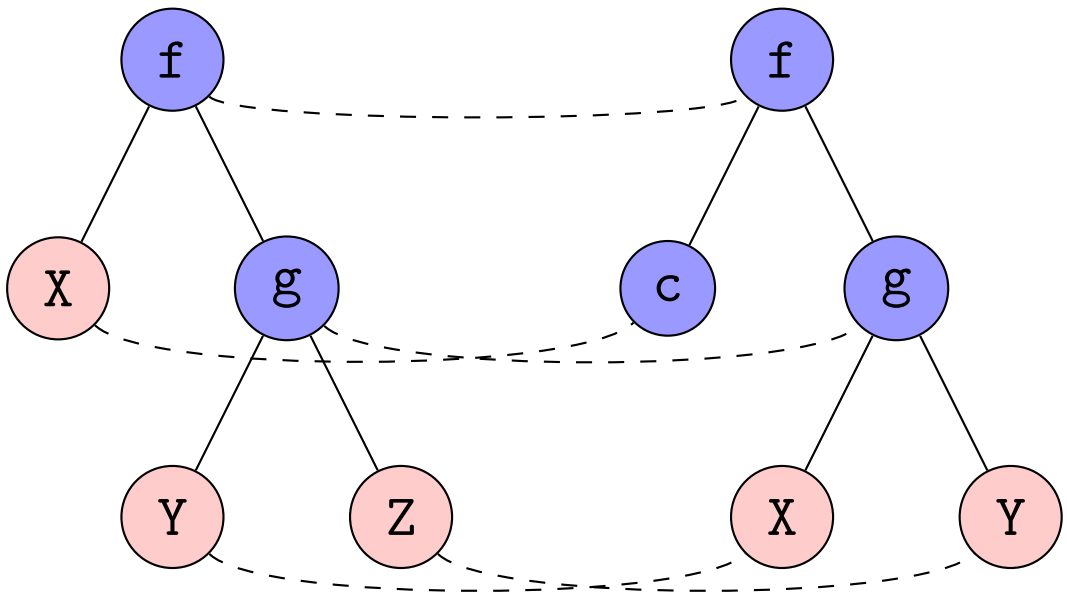
- Cum găsește aceste răspunsuri?

Unificare

- O substituție este o funcție (parțială) de la variabile la termeni.
  - $X_1 = t_1, \ldots, X_n = t_n$
- Pentru doi termeni t și u, cu variabilele $X_1, \ldots, X_n$, un unificator este o substituție care aplicată termenilor t și u îi face identici.

Exemplu

f(X,g(Y,Z)) = f(c,g(X,Y))



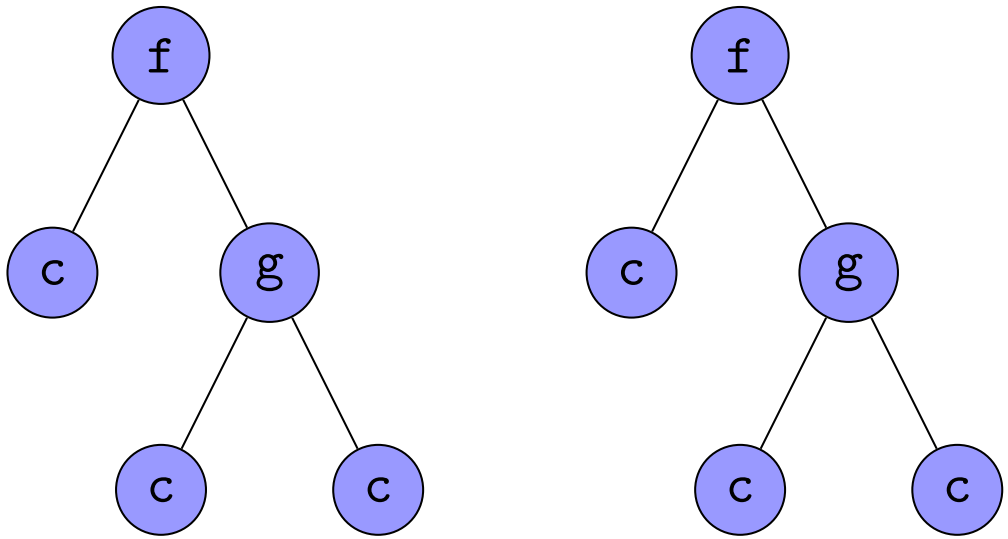
X=c

Y=X

Z=Y

Exemplu: aplicând substituția

`f(X,g(Y,Z)) = f(c,g(X,Y))`



`X=c`
`Y=c`
`Z=c`

Unificare

- Ce se întâmplă dacă încercăm să unificăm X cu ceva care conține X? Exemplu: `?- X = f(X).`
- Conform teoriei, acești termeni nu se pot unifica.
- Totuși, multe implementări ale Prolog-ului sar peste această verificare din motive de eficiență.
  - putem folosi `unify_with_occurs_check/2`

Ce se întamplă în Prolog când punem o întrebare?

- Pentru a găsi un răspuns, Prolog încearcă regulile în ordinea în care sunt scrise.
- Folosește unificarea pentru a potrivi țintele și clauzele (reguli și fapte).
- Prolog poate da 2 tipuri de răspunsuri:
  - `false` – în cazul în care întrebarea nu este o consecință a programului.
  - `true` sau valori pentru variabilele din întrebare în cazul în care întrebarea este o consecință a programului.
- Poate găsi zero, una sau mai multe soluții.
- Execuția se poate întoarce (*backtracking*).

Exerciții

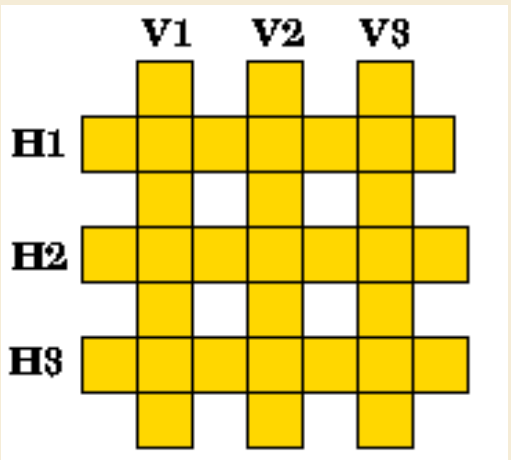
Practică

Exercițiul 1: cunvinte încrucișate

Șase cuvinte din engleză

abalone, abandon, anagram, connect, elegant, enhance

trebuie aranjate într-un puzzle de cuvinte încrucișate ca în figură.



Fișierul `words.pl` este o bază de cunoștințe ce conține aceste cuvinte.

Practică

Exercițiul 1 (cont.)

Definiți un predicat `crosswd/6` care calculează toate variantele în care puteți completa grila. Primele trei argumente trebuie să fie cuvintele pe verticală, de la stânga la dreapta, (V1,V2,V3), iar următoarele trei argumente trebuie să fie cuvintele pe orizontală, de sus în jos (H1,H2,H3).

Hint: Specificați că V1, V2, V3, H1, H2, H3 sunt cuvinte care au anumite litere comune. Unde este cazul, folosiți variabile anonime.

Practică

Exercițiul 2: baza de date

În acest exercițiu vom demonstra cum se poate implementa o bază de date simplă în Prolog.

Folosiți în programul vostru următoarea bază de cunoștințe:

```
born(jan, date(20,3,1977)).
born(jeroen, date(2,2,1992)).
born(joris, date(17,3,1995)).
born(jelle, date(1,1,2004)).
born(joan, date(24,12,0)).
born(joop, date(30,4,1989)).
born(jannecke, date(17,3,1993)).
born(jaap, date(16,11,1995)).
```

Reprezentăm datele calendaristice ca termeni de forma `date(Day,Month,Year)`.

U. Endriss, Lecture Notes. An Introduction to Prolog Programming, ILLC, Amsterdam, 2018.

Practică

Exercițiul 2 (cont.)

a) Scrieți un predicat `year/2` care găsește toate persoanele născute într-un anumit an.

Exemplu:

```
?- year(1995, Person).
Person = joris
Person = jaap
```

Hint: Folosiți variabile anonime.

Practică

Exercițiul 2 (cont.)

b) Scrieți un predicat `before/2` care primește două date calendaristice și care este adevărat dacă prima expresie reprezintă o dată calendaristică înaintea datei reprezentate de a doua expresie (puteți presupune că datele sunt corecte, e.g., nu puteți primi 31 Aprilie).

Exemplu:

```
?- before(date(31,1,1990), date(7,7,1990)).
true
```

Practică

Exercițiul 2 (cont.)

c) Scrieți un predicat `older/2` care este adevărat dacă persoana dată ca prim argument este mai în vârstă (strict) decât persoana dată ca al doilea argument.

Exemplu:

```
?- older(jannecke,X).
X = joris
X = jelle
X = jaap
```

Practică

Exercițiul 3: drumurile într-un labirint

Baza de cunoștințe din `maze.pl` descrie un labirint.

Faptele indică ce puncte sunt conectate (din ce punct se poate ajunge într-un alt punct într-un pas).

Drumurile sunt cu sens unic (se poate merge pe ele doar într-o direcție).

De exemplu, se poate ajunge într-un pas de la 1 la 2, dar nu și invers.

Adăugați un predicat `path/2` care indică dacă dintr-un punct puteți să ajungeți într-un alt punct (în mai mulți pași), legând conexiunile din baza de cunoștințe.

Practică

Exercițiul 3 (cont.)

Puneți următoarele întrebări:

- Puteți ajunge din punctul 5 în punctul 10?
- În ce puncte puteți să ajungeți plecând din 1?
- Din ce puncte puteți să ajungeți în punctul 13?

Practică

Exercițiul 3 (cont.)

Testați programul pentru următoarea bază de cunoștințe:

```
connected(1,2).
connected(2,1).
connected(1,3).
connected(3,4).

?- path(1,4).
```

Atenție!
Dacă graful conține cicluri, este posibil ca programul să nu se termine.

Scrieți un predicat care determină existența drumurilor, evitând ciclările.

Indicație: folosiți un predicat auxiliar care reține într-o listă punctele vizitate până la momentul curent.

Practică

Exercițiul 4: numere naturale

În acest exercițiu vom reprezenta numerele sub următoarea formă:

```
0   :   []
1   :   [x]
2   :   [x,x]
3   :   [x,x,x]
4   :   [x,x,x,x]
...
```

Practică

Exercițiul 4 (cont.)

a) Definiți un predicat `successor/2` care întoarce în al doilea argument succesorul numărului dat ca prim argument.

Exemplu:

```
?- successor([x,x,x],Result).
Result = [x,x,x,x]

?- successor([],Result).
Result = [x]
```

Practică

Exercițiul 4 (cont.)

b) Definiți un predicat `plus/3` care adună două numere.

Exemplu:

```
?- plus([x, x], [x, x, x, x], Result).
Result = [x, x, x, x, x, x]
```

Practică

Exercițiul 4 (cont.)

c) Definiți un predicat `times/3` care înmulțește două numere.

Exemplu:

```
?- times([x, x], [x, x, x, x], Result).
Result = [x, x, x, x, x, x, x, x]
```

Practică

Exercițiul 5

Definiți un predicat `element_at/3` care, primind o listă și un număr natural *n*, întoarce al n-ulea element din listă.

Exemplu:

```
?- element_at([tiger, dog, teddy_bear, horse, cow], 3, X).
X = teddy_bear


?- element_at([a, b, c, d], 27, X).
false
```

Practică

Exercițiul 6

În fișierul `animale.pl` găsiți o bază de cunoștinte care definește animalele cunoscute. Un "mutant" se obține prin combinarea a două animale cunoscute, cu proprietatea ca numele unuia are un sufix care este prefix pentru numele celuilalt. Numele animalului mutant se obține prin concatenarea celor două nume, în care partea comună apare o singură dată. De exemplu, din `alligator` și `tortue` se obține `aligatortue`. Scrieți un predicat `mutant/1` care generează pe rând toate animalele mutante.

H. Coelho, J.C.Cotta, Prolog by example, 1998

- Folosiți predicatul `name/2` care face conversia între un atom și lista caracterelor sale, reprezentate prin codurile ASCII. Verificați:

```
?- name(alligator,L).
?- name(A, [97, 108, 108, 105, 103, 97, 116, 111, 114]).
```

Pe data viitoare!