Universidade do Minho
Escola de Engenharia
Departamento de Informática

Prolog

Unification and proof tree

Mestrado Integrado em Engenharia Informática
Licenciatura em Engenharia Informática
Inteligência Artificial

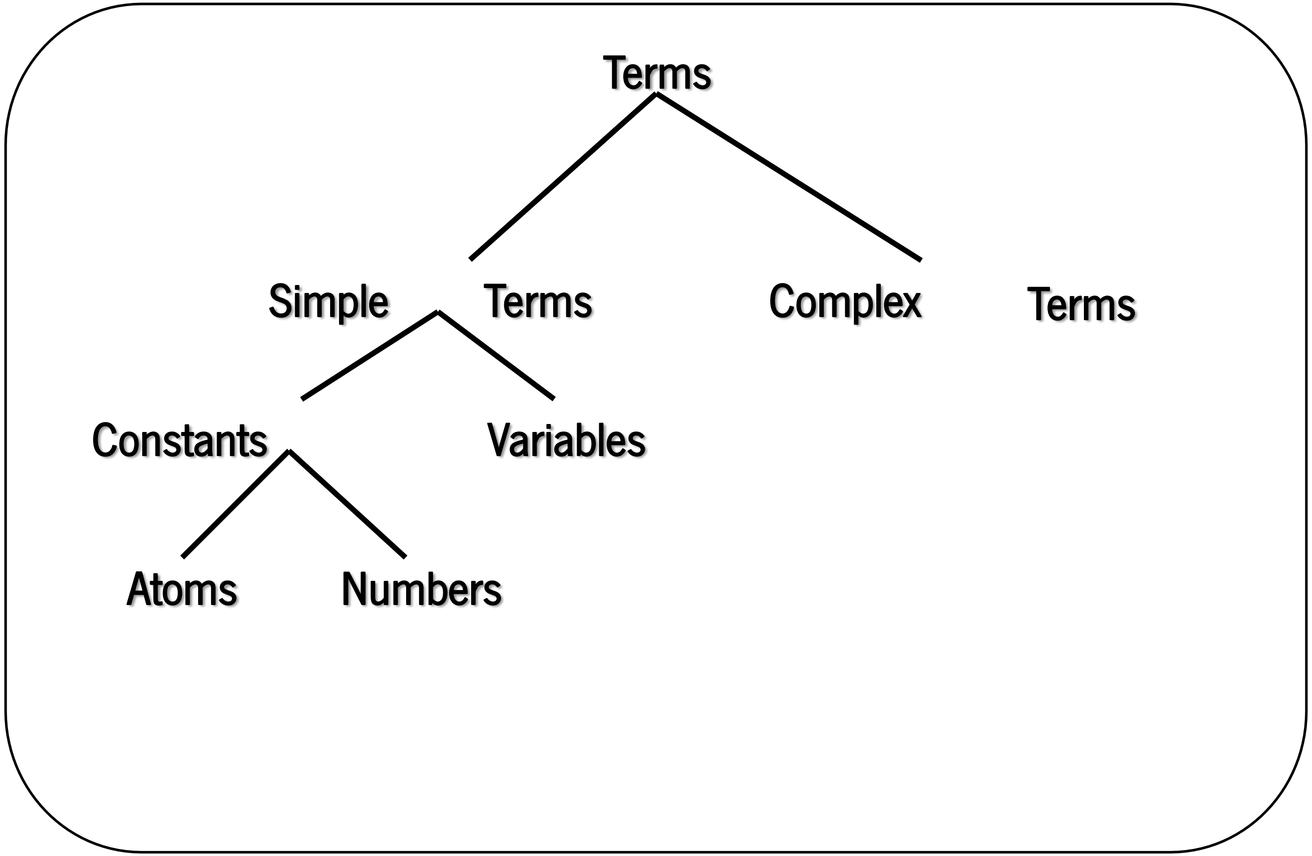
Summary

- Theory:
  - Unification;
  - Unification in Prolog;
  - Proof search.

Unification

- When Prolog unifies:
    - **mulher(X)**
- with
    - **mulher(ana)**
- It is instantiating the variable **X** with the atom **ana**.

Recall Prolog Terms



Unification

- **Two terms unify**:
  - if they are the same term,
  - or
  - if they contain variables that can be uniformly instantiated with terms in such a way that the resulting terms are equal.

Unification

- This means that:
  - **ana** and **ana** unify;
  - **42** and **42** unify;
  - **mulher(ana)** and **mulher(ana)** unify.

- This also means that:
  - **bruno** and **ana** do not unify;
  - **mulher(ana)** and **mulher(berta)** do not unify.

Unification

- What about the terms:
  - ana and X;
  - mulher(Z) and mulher(ana);
  - gosta(ana,X) and gosta(Y,miguel)

  - They unify!
  - X is instantiated to ana;
  - Z is instantiated to ana;
  - X is instantiated to Miguel, ana to Y.

Instantiations

- When Prolog unifies two terms, it performs all the necessary instantiations, so that the terms are equal afterwards;

- This makes unification a powerful programming mechanism.

ISLab
Synthetic Intelligence Lab

Unification

- If C_1 and C_2 are constants, then C_1 and C_2 unify if they are the same atom, or the same number;
- If C_1 is a variable and C_2 is any type of term, then C_1 and C_2 unify, and C_1 is instantiated to C_2 (and vice versa);

ISLab
Synthetic Intelligence Lab

Unification

- If C_1 and C_2 are complex terms then they unify if:
  - They have the same functor and arity, and all their corresponding arguments unify, and the variable instantiations are compatible.

How will Prolog respond?

```
?- X=ana, X=bruno.
 no
?-
```

Why? After working through the first goal, Prolog has instantiated X with **ana**, so that it cannot unify it with **bruno** anymore. Hence the second goal fails.

Example with complex terms

```
?- k(s(g),Y) = k(X,t(k)).
X=s(g)
Y=t(k)
 yes
?-
```

ISLab
Synthetic Intelligence Lab

Prolog and unification

- Prolog does not use a standard unification algorithm;
- Consider the following query:
    - ?- pai(X) = X.
- Do these terms unify or not?

Infinite terms

```
?- pai(X) = X.
X=pai(pai(pai(pai(pai(pai(pai(pai(pai(pai(...))))))))))? ;
no
```

Occurs Check

- A standard unification algorithm carries out an **occurs check:**
  - If it is asked to unify a variable with another term it checks whether the variable occurs in this term;
  - In Prolog (ISO standard):

```
?- unify_with_occurs_check(pai(X), X).
no
```

Programming with Unification

```
vertical(line(point(X,Y), point(X,Z))).

horizontal(line(point(X,Y), point(Z,Y))).
```

```
?- vertical(line(point(1,1),point(1,3))).
 yes
?- vertical(line(point(1,1),point(3,2))).
no
?-
```

Synthetic Intelligence Lab

Programming with Unification

```
vertical(line(point(X,Y), point(X,Z))).

horizontal(line(point(X,Y), point(Z,Y))).
```

```
?-horizontal(line(point(1,1),point(1,Y))).
Y = 1;
Yes
?-
```

Which of the following pairs unify?

1. pao = pao
2. 'Pao' = pao
3. 'pao' = pao
4. Pao = pao
5. pao = molho
6. comida(pao) = pao
7. comida(pao) = X
8. comida(X) = comida(pao)
9. comida(pao,X) = comida(Y,molho)
10. comida(pao,X,cerveja) = comida(Y,molho,X)
11. comida(pao,X,cerveja) = comida(Y,big_mac)
12. refeicao(comida(pao),bebida(cerveja)) = refeicao(X,Y)
13. refeicao(comida(pao),X) = refeicao(X,bebida(cerveja))

Which of the following pairs unify?

1. pao = pao yes
2. 'Pao' = pao No
3. 'pao' = pao Yes
4. Pao = pao Yes, Pao=pao
5. pao = molho No
6. comida(pao) = pao No
7. comida(pao) = X Yes, X=comida(pao)
8. comida(X) = comida(pao) Yes, X=pao
9. comida(pao,X) = comida(Y,molho) Yes, X=molho, Y=pao
10. comida(pao,X,cerveja) = comida(Y,molho,X) No
11. comida(pao,X,cerveja) = comida(Y,big_mac) No
12. refeicao(comida(pao),bebida(cerveja)) = refeicao(X,Y) Yes, X=comida(pao), Y=bebida(cerveja)
13. refeicao(comida(pao),X) = refeicao(X,bebida(cerveja)) No

Synthetic Intelligence Lab

Which queries are satisfied?

```
elfo(diogo).
bruxa(herminia).
bruxa('Maria').
bruxa(rita).
magico(X):- elfo(X).
magico(X):- feiticeiro(X).
magico(X):- bruxa(X).
```

```
?- magico(elfo).
?- feiticeiro(andre).
?- magico(feiticeiro).
?- magico('Maria').
?- magico(Herminia).
```

Which queries are satisfied?

```
elfo(diogo).
bruxa(herminia).
bruxa('Maria').
bruxa(rita).
magico(X):- elfo(X).
magico(X):- feiticeiro(X).
magico(X):- bruxa(X).
```

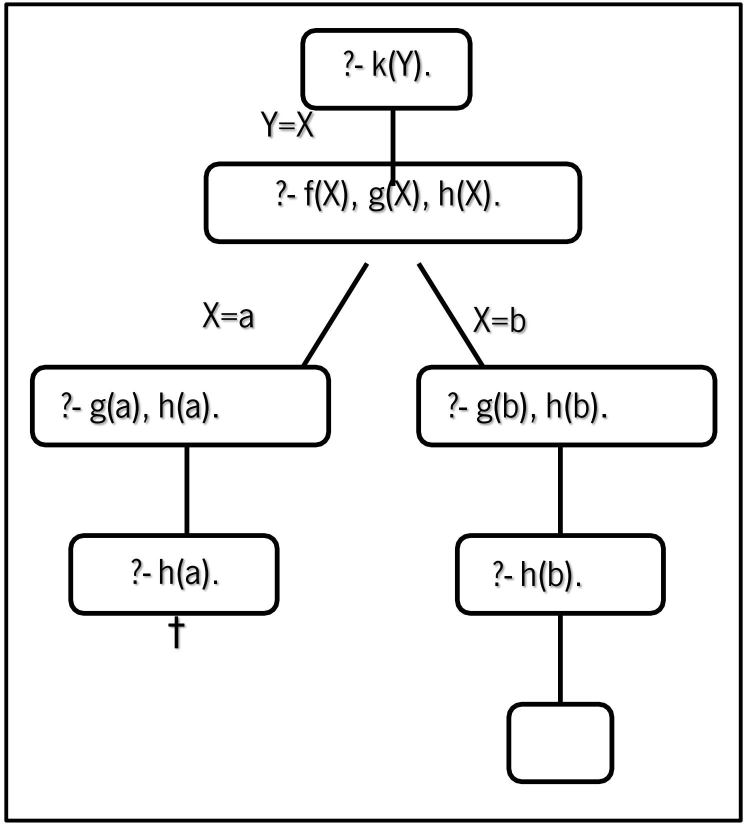
```
?- magico(elfo).            No
?- feiticeiro(harry).       No
?- magico(feiticeiro). No
?- magico('Maria'). Yes
?- magico(Herminia). Yes, Herminia = diogo ;
```

Synthetic Intelligence Lab

Example: search tree

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

```
?- k(Y).
Y=b;
true
?-
```

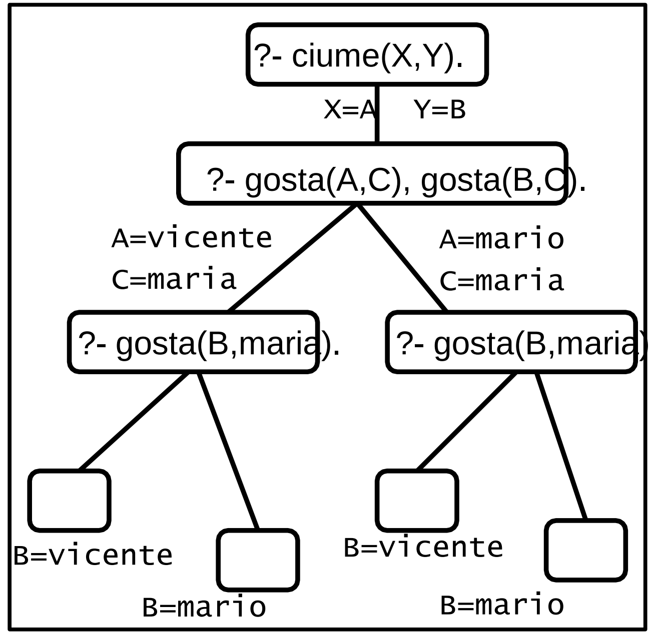


Another example

```
gosta(vicente,maria).
gosta(mario,maria).

ciume(A,B):-gosta(A,C),gosta(B,C).
```

```
?- ciume(X,Y).
X=vicente
Y=vicente;
X=vicente
Y=mario;
X=mario
Y=vicente;
X=mario
Y=mario;
no
```



Exercise

```
elfo(diogo).
bruxa(herminia).
bruxa('Maria').
bruxa(rita).
magico(X):- elfo(X).
magico(X):- feiticeiro(X).
magico(X):- feiticeiro(X).
```

```
?- magico(Herminia).
```

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Prolog

Unification and proof tree

Mestrado Integrado em Engenharia Informática
Licenciatura em Engenharia Informática
Inteligência Artificial