

Knowledge Representation

Prolog

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA
Sistemas de Representação de Conhecimento e Raciocínio

- Introduction to Prolog;
- Facts, Rules and Queries;
- Prolog Syntax.

- "Programming with Logic";
- Different from other programming languages;
- Declarative;
- Recursion;
- Relations;
- Unification;

- Describe the problem in hands;
- Ask a Question.
- Prolog:
 - logically deduces new facts about the problem in hands;
 - returns its deductions as answers.

- Think declaratively, not procedurally;
- Challenging;
- Requires a different approach;
- High-level language;
- Relatively low efficiency ;
- Rapid prototyping;
- Useful in many AI applications (knowledge representation, inference).

- Programming in Prolog is :
 - Providing axioms that indicate some facts about the world;
 - Providing rules that allow to infer other facts about the world.
- What should be computed rather than how it should be computed...

- Facts, rules and queries are built out off Prolog terms;
- A term is either:
 - A constant, which can be either an atom or a number;
 - A variable;
 - A complex term.

- A sequence of characters of upper-case letters, lower-case letters, digits, or underscore, starting with a lowercase letter
 - Examples: mae, somaNumeros, jogar
- An arbitrary sequence of characters enclosed in single quotes
 - Examples: 'Manuel', 'Abraço', '@\$%'
- A sequence of special characters
 - Examples: : , ; . :-

- Integers:
 - 12, -34, 22342
- Floats:
 - 3473.32, 0.4567

- A sequence of characters of upper-case letters, lower-case letters, digits, or underscore, starting with either an uppercase letter or an underscore
- **Examples:**
 - **X, Y, Variable, Ana, _tag**
 - underscore (**_**) represents an unknown variable

Complex Terms

- Atoms, numbers and variables are building blocks for complex terms;
- Complex terms are built out of a functor directly followed by a sequence of arguments;
- Arguments are put in round brackets, separated by comas;
- The functor must be an atom.

- Examples:
 - toca(joana).
 - gosta(mark, ana).
 - inveja(mike, mark).
- Complex terms inside complex terms:
 - relacao(X,pai(pai(pai(rui))))

- The number of arguments a complex term has is called its arity
- Examples:
- `mulher(sara)` is a term with arity 1

`gosta(vicente,sara)`

arity 2

`pai(pai(rui))`

arity 1

Arity is important

- Predicates with the same functor but with different arity are not the same!
- Arity of predicate usually indicated with the suffix "/" followed by a number to indicate the arity
- Example: filho/2

Example of Arity

```
feliz(ana).  
ouvemusica(carlos).  
ouvemusica(ana):- feliz(ana).  
tocapiano(ana):- ouvemusica(ana).  
tocapiano(joana):- ouvemusica(joana).
```

- This knowledge base defines
 - feliz/1
 - ouvemusica/1
 - tocapiano/1

Knowledge Base

example 1

mulher(ana).
mulher(joana).
tocaGuitarra(joana).
festa.

?- mulher(ana).
yes
?- tocaGuitarra(joana).
yes
?- tocaGuitarra(ana).
no

mulher(ana).
mulher(joana).
mulher(paula).
tocaguitarra(joana).
festa.

?- tatuada(joana).
! Existence error in user: tatuada/1
?-

?-concertoRock.

(answer?)

```
feliz(paula).  
ouveMusica(ana).  
ouveMusica(paula):- feliz(paula).  
tocaguitarra(ana):- ouveMusica(ana).  
tocaGuitarra(paula):- ouveMusica(paula).
```

feliz(paula).

fact

ouveMusica(ana).

fact

ouveMusica(paula):- feliz(paula).

rule


tocaGuitarra(ana):- ouveMusica(ana).

rule

tocaGuitarra(paula):- ouvemusica(paula).

rule

feliz(paula).
ouveMusica(ana).
ouveMusica(paula):- feliz(paula).
tocaGuitarra(ana):- ouveMusica(ana).
tocaGuitarra(paula):- ouveMusica(paula).



head

body

feliz(paula).
ouveMusica(marco).
ouveMusica(paula):- feliz(paula).
tocaGuitarra(marco):- ouveMusica(marco).
tocaGuitarra(paula):- ouveMusica(paula).

?- tocaguitarra(marco).
yes
?- tocaGuitarra(paula).
yes

feliz(paula).
ouveMusica(marco).
ouveMusica(paula):- feliz(paula).
tocaGuitarra(marco):- ouveMusica(marco).
tocaGuitarra(paula):- ouveMusica(paula).

There are five clauses in this knowledge base: two facts, and three rules.

The end of a clause is marked with a full stop.

feliz(paula).
ouveMusica(marco).
ouveMusica(paula):- feliz(paula).
tocaGuitarra(marco):- ouveMusica(marco).
tocaGuitarra(paula):- ouveMusica(paula).

There are three predicates in this knowledge base:

feliz, ouveMusica, and tocaGuitarra

```
feliz(bruno).  
ouveMusica(miguel).  
tocaGuitarra(bruno):- ouveMusica(bruno), feliz(bruno).  
tocaGuitarra(miguel):- feliz(miguel).  
tocaGuitarra(miguel):- ouveMusica(miguel).
```

The comma “,” expresses conjunction in Prolog


```
feliz(bruno).  
ouveMusica(miguel).  
tocaGuitarra(bruno):- ouveMusica(bruno), feliz(bruno).  
tocaGuitarra(miguel):- feliz(miguel).  
tocaGuitarra(miguel):- ouveMusica(miguel).
```

```
?- tocaGuitarra(bruno).  
no  
?-
```

```
feliz(bruno).  
ouveMusica(miguel).  
tocaGuitarra(bruno):- ouveMusica(bruno), feliz(bruno).  
tocaGuitarra(miguel):- feliz(miguel).  
tocaGuitarra(miguel):- ouveMusica(miguel).
```

```
?- tocaGuitarra(miguel).  
yes  
?-
```

Expressing Disjunction

```
feliz(bruno).  
ouveMusica(miguel).  
tocaguitarra(bruno):- ouveMusica(bruno), feliz(bruno).  
tocaGuitarra(miguel):- feliz(miguel).  
tocaGuitarra(miguel):- ouveMusica(miguel).
```

```
feliz(bruno).  
ouveMusica(miguel).  
tocaguitarra(bruno):- ouveMusica(bruno), feliz(bruno).  
tocaGuitarra(miguel):- feliz(miguel); ouveMusica(miguel).
```

mulher(ana).
mulher(berta).
mulher(paula).

gosta(mario, ana).
gosta(miguel, ana).
gosta(pedro, helena).
gosta(helena, pedro).

```
mulher(ana).  
mulher(berta).  
mulher(paula).
```

```
gosta(mario, ana).  
gosta(miguel, ana).  
gosta(pedro, helena).  
gosta(helena, pedro).
```

```
?- mulher(X).
```

```
X=ana;
```

```
X=berta;
```

```
X=paula;
```

```
no
```

mulher(ana).
mulher(berta).
mulher(paula).

gosta(mario, ana).
gosta(miguel, ana).
gosta(pedro, helena).
gosta(helena, pedro).

?- loves(mario,X), woman(X).

X=ana

yes

?-

mulher(ana).
mulher(berta).
mulher(paula).

gosta(mario, ana).
gosta(miguel, ana).
gosta(pedro, helena).
gosta(helena, pedro).

?- gosta(pedro,X), woman(X)

No

?-

Knowledge Base

example 5

```
gosta(miguel,ana).  
gosta(bruno,ana).  
gosta(pedro, helena).  
gosta(helena, pedro).
```

```
ciume(X,Y):- gosta(X,Z), gosta(Y,Z).
```



```
gosta(miguel,ana).  
gosta(bruno,ana).  
gosta(pedro, helena).  
gosta(helena, pedro).
```

```
ciume(X,Y):- gosta(X,Z), gosta(Y,Z).
```

```
?- ciume(bruno,W).
```

```
W=mark
```

```
?-
```

- Artificial Intelligence: A Modern Approach, Stuart Russell and Peter Norvig, (3rd Edition), ISBN 978-9332543515, 2015.
- Prolog Programming for Artificial Intelligence (4th Edition), Ivan Bratko, ISBN-13: 978-0321417466, 2011.
- Inteligência Artificial-Fundamentos e Aplicações, E.Costa, A.Simões; FCA, ISBN: 978-972-722-340-4, 2008.
- Artificial Intelligence: Foundations of Computational Agents, Poole and Mackworth, 2nd ed., ISBN 978-1107195394, 2017.
- Learn prolog Now!, <http://www.learnprolognow.org/index.php>

Some Prolog Implementations

SWI-Prolog - A Free Software Prolog environment, licensed under the Lesser GNU public license. This popular interpreter was developed by Jan Wielemaker. This is the interpreter we used while developing this book.

<http://www.swi-prolog.org/>

SICStus Prolog - Industrial strength Prolog environment from the Swedish Institute of Computer Science. <http://www.sics.se/sicstus/>

GNU Prolog - Another more widely used free Prolog compiler developed by Daniel Diaz. <http://www.gprolog.org>

YAP Prolog - A Prolog compiler developed at the Universidade do Porto and Universidade Federal do Rio de Janeiro. Free for use in academic environments.
<http://www.ncc.up.pt/~vsc/Yap/>

Knowledge Representation

Prolog

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA
Sistemas de Representação de Conhecimento e Raciocínio