Introduction to AI: The Prolog Language

François Jacquenet

Laboratoire Hubert Curien
Université de Saint-Etienne
Francois.Jacquenet@univ-st-etienne.fr

November, 2020

UNIVERSITÉ
JEAN MONNET
SAINT-ÉTIENNE

Outline

1. Introduction
2. Prog 1
3. Prog 2
4. Prog 3
5. Prog 4
6. Prog 5
7. Lexical and Syntactic conventions

Different programming paradigms

- Imperative Programming (C, Pascal, Fortran, ADA, ...)
  - define the state of the world by global and local variables
  - you have to give steps to change the state of the world (HOW)
- Object Oriented Programming (Java, C++, Smalltalk, ...)
  - define the state of the world using classes/objects
  - change the state of the world by sending messages (HOW)
- Functional Programming (LISP, ML, CAML, Haskell, ...)
  - define the state of the world using functions
  - change the state of the world by calling functions (HOW)
- Logic Programming (Prolog)

Logic Programming paradigm

A=L+C (Robert Kowalski, 1979)

ALGORITHM = LOGIC + CONTROL

Logic = You specify the world (the problem) using facts and rules (WHAT)

Control = The engine (Prolog interpreter) tries to solve the problem (HOW)

The user only focus on the WHAT and not on the how.

History

- The inventor of Prolog is Alain Colmerauer (Prof in Marseille, France)
- Prolog was born in 1972 while A. Colmerauer was working on NLP (Machine Translation)
- Many scientific discussions with Robert Kowalski (Edinburgh University)
- 1st interpreter in 1972 (Algol-W by Philippe Roussel)
- 2nd interpreter in 1973 (Fortran by G. Battani, H. Meloni and R. Bazzoli)
- Warren Abstract Machine (1977, David Warren)
- Then many researches and prototypes all around the world
- Several books and softwares (see the Claroline workspace)

Facts

Prog 1

```
woman(agatha).
woman(camilla).
woman(mary).
playsGuitar(mary).
november.
```

Here you can see:

- predicate symbols (woman/1, playsGuitar/1, november/0)
- constants (agatha, camilla, mary)
- facts (Horn clauses)

Submitting queries

```
?- woman(agatha).
true
```

Note that ?- is the command prompt of the interpreter
DON'T FORGET the dot at the end of each fact, rule and query

```
?- playsGuitar(mary).
true
```

```
?- playsGuitar(francois).
false
```

```
?- november.
true
```

```
?- playsGuitar(camilla).
false
```

```
?- playsPiano(agatha).
false
```

```
?- december.
false
```

Facts and rules

Prog2

```
happy(agatha).
listenMusic(mary).

listenMusic(agatha) :- happy(agatha).
playsGuitar(mary) :- listenMusic(mary).
playsGuitar(agatha) :- listenMusic(agatha).
```

Here you can see (Horn clauses):

- facts
- rules

lhs = head of the rule; rhs = body of the rule

Inference

If the KB contains a rule `Head :- Body`
and if `Body` is known to be true (given the information in the KB)
then Prolog may infer `Head` is true.

Example:

```
?- playsGuitar(mary).
true
```

There is no fact to prove `playsGuitar(mary)`
The interpreter tries to find the head of a rule that corresponds (unifies). Rule 4 is ok for that.
So, to prove `playsGuitar(mary)` Prolog has to prove `listenMusic(mary)` and given fact 2 it is true.

Inference

Other example:

```
?- playsGuitar(agatha).
true
```

Indeed, there is no fact `playsGuitar(agatha)` but

there are 2 rules:

```
listenMusic(agatha) :- happy(agatha).
playsGuitar(agatha) :- listenMusic(agatha).
```

and one fact:

```
happy(agatha).
```

Conjunctions

Prog3

```
happy(agatha).
listenMusic(camilla).

playsGuitar(agatha):-
    happy(agatha),
    listenMusic(agatha).
playsGuitar(camilla):-
    happy(camilla).
playsGuitar(camilla):-
    listenMusic(camilla).
```

Here you can see a conjunction of two literals

Conjunctions and disjunctions

Consider some queries:

```
?- playsGuitar(agatha).
false

?- playsGuitar(camilla).
true
```

explanations (on the blackboard)

Prog4

```
woman(mary).
woman(camilla).
woman(agatha).
loves(john,mary).
loves(mickael,mary).
loves(mickael,hector).
loves(franz,camilla).
loves(camilla,franz).
```

Unification

Suppose we write the query `woman(X)` let's look at the result:

```
?- woman(X).
X = mary (if the user enters ''return'', it stops with the first
result)

?- woman(X).
X = mary (if the user enters '';'' it looks for another result)
X = camilla (if the user enters '';'' it looks for another result)
X = agatha (if the user enters '';'' it looks for another result)
false
```

Unification

Consider the query `loves(mickael,X),woman(X)` let's look at the result:

```
?- loves(mickael,X),woman(X).
X = mary
```

explanations (on the blackboard)

Unification and Resolution

Prog5

```
loves(john,mary).
loves(mickael,mary).
loves(franz,camilla).
loves(camilla,franz).

jealous(X,Y):- loves(X,Z), loves(Y,Z).
```

Unification and Resolution

```
?- jealous(mickael,X).
X = john
X = mickael

?- jealous(john,X).
X = john
X = mickael

?- jealous(X,mickael).
X = john
X = mickael
```

explanations (on the blackboard - consider different orders of the clauses)

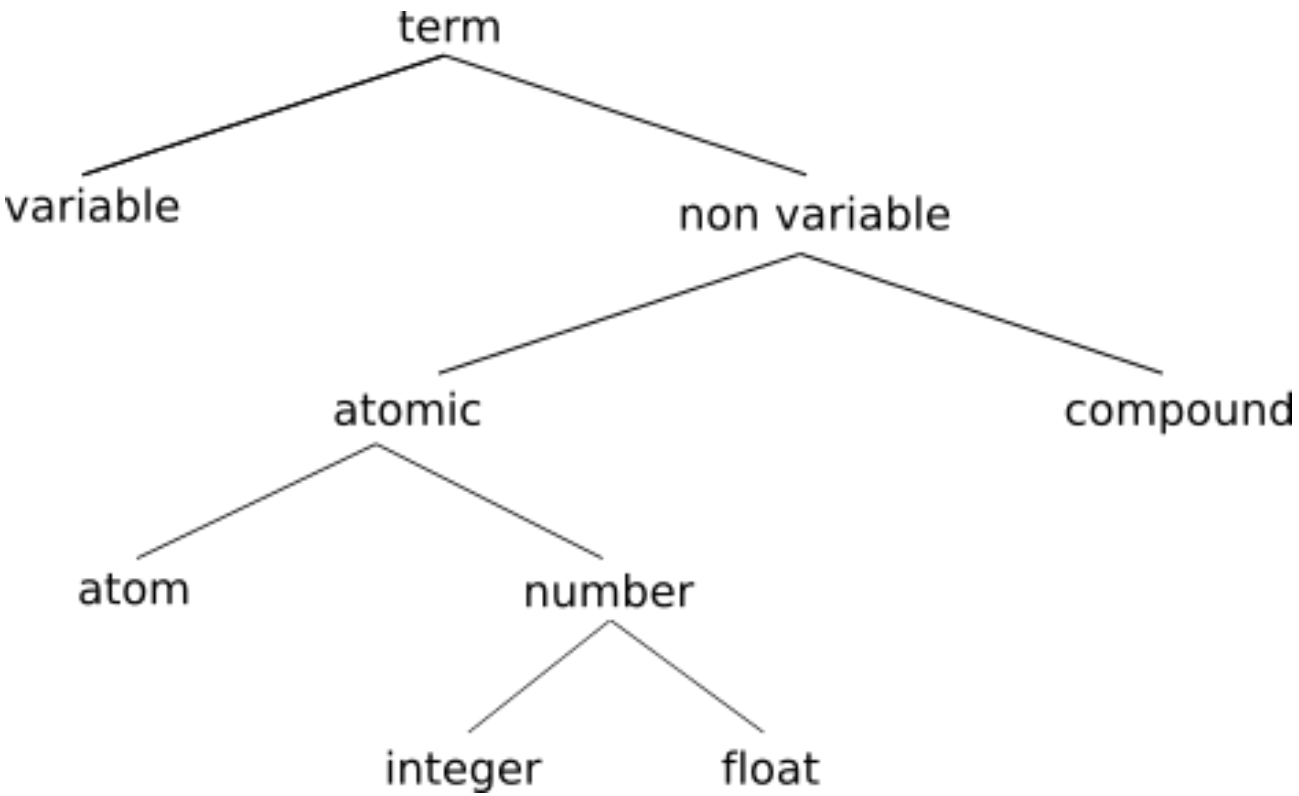
Unification and Resolution

We can even ask:

```
?- jealous(X,Y).
X = Y, Y = john
X = john, Y = mickael
X = mickael, Y = john
X = Y, Y = mickael
X = Y, Y = franz
X = Y, Y = camilla.
```

Terms

What is a term in Prolog?



Variables

Lexical definition:

```
[_A-Z][_a-zA-Z0-9]*
```

Examples:

- ThisIsAVariable
- X
- VARIABLE
- _ (this is called an anonymous variable)
- _192
- X42a

Atoms

Lexical definition:

```
[a-z][_a-zA-Z0-9]*
```

Examples:

- thisIsAnAtom
- x
- i_am_an_atom

But also the operators (+, -, *, /, <, ...)

and also the quoted atoms: 'X42a', '42', 'Mary'

Numbers

Integers

Floats

Compound terms

A compound term is made up of

- an atom
- a list of terms between parentheses

Example: date(6,november,2017)

The number of terms between parentheses is called the arity of the compound term

Other example:

```
person(
   information(
      'Smith',
      'John',
      american,
      male,
      birthdate(15,'February',1980)),
   adress('1337 Geeks street','San Francisco','CA')
   )
```

person/2 is called the functor of the compound term

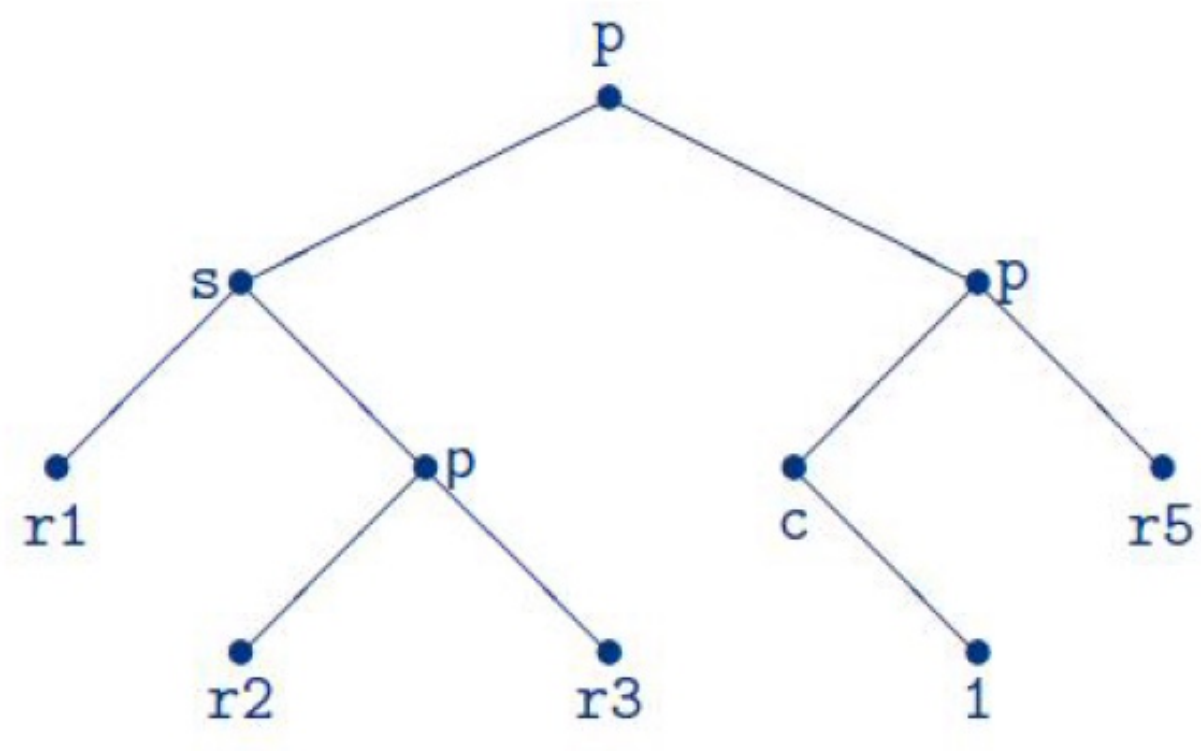
birthdate/3 is the functor of the compound term `birthdate(15,'February',1980)`

Compound terms and trees

Compound terms may be seen as trees

Example: the compound term p(s(r1,p(r2,r3)),p(c(1),r5))

may be seen as:



Exercise 1

For each of the string below, say whether it's a variable, an atom, or nothing:

- fRANZ
- Football
- variable42
- Constant2000
- big_mac_burger
- 'big mac burger'
- big mac burger
- 'William Shakespear'
- _William_Shakespear
- '_William_Shakespear'

Exercise 2

For each of the string below, say whether it's a variable, an atom, a compound term or nothing:

- loves(John,mia)
- 'loves(John,mia)'
- Butch(boxer)
- boxer(Butch)
- and(big(burger),kahuna(burger))
- and(big(X),kahuna(X))
- _and(big(X),kahuna(X))
- (Butch kills Vincent)
- kills(Butch Vincent)
- kills(Butch,Vincent

Exercise 3

For each sentence below, translate it to a Prolog fact

- The weather is good.
- Steve Jobs is dead.
- John loves Mary and Mary loves John.
- Henry IV is the father of Louis XIII.
- Mary of Medicis is the mother of Henry IV.
- Superman if stronger than everybody.
- $0! = 1$

Exercise 4

Convert to Prolog each sentence below

- Mary likes everybody who is a good dancer.
- John kills anyone who smiles at Mary.
- John eats everything that is nutritious or tastes good.
- If we are invincible, then we are stronger than everyone else.
- If you are the father of someone's father or mother then you are his grandfather.
- If you are a grandparent then you are grandfather or grandmother.

Exercise 5

Consider the following knowledge base:

```
wizard(ron).
hasMagicWand(harry).
playsQuidditch(harry).
wizard(X) :- hasBroom(X), hasMagicWand(X).
hasBroom(X) :- playsQuidditch(X).
```

What will be the answers to the following queries:

```
?- wizard(ron).
?- witch(ron).
?- wizard(hermione).
?- witch(hermione).
?- wizard(harry).
?- wizard(X).
?- witch(X).
```