Logic - CM0845
Introduction to Prolog

Diego Alejandro Montoya-Zapata

EAFIT University

Semester 2016-1

What is Prolog?

Prolog is a language for **logic programming** and symbolic computation.

What is Prolog?

Prolog is a language for **logic programming** and symbolic computation.

A Prolog program consists of:

- Declaration of the facts of the relations involved.

What is Prolog?

Prolog is a language for **logic programming** and symbolic computation.

A Prolog program consists of:

- Declaration of the facts of the relations involved.
- Declaration of rules concerning relations.

What is Prolog?

Prolog is a language for **logic programming** and symbolic computation.

A Prolog program consists of:

- Declaration of the facts of the relations involved.
- Declaration of rules concerning relations.
- Formulation of questions to be answered (query).

What is Prolog?

Prolog is a language for **logic programming** and symbolic computation.

A Prolog program consists of:

- Declaration of the facts of the relations involved.
- Declaration of rules concerning relations.
- Formulation of questions to be answered (query).

What is Prolog?

Example 1

```
pet(dog).
pet(carrot).
pet(cat).
sing(carrot).
nothing_special.
```

What is Prolog?

Example 1

```
pet(dog).
pet(carrot).
pet(cat).
sing(carrot).
nothing_special.
```

Example 2

```
sad(jason).
sad(maria).
cry(jason) :- sad(jason).
plays_soccer(jason) :- cry(jason).
```

What is Prolog?

Example 3

```
sad(jason).
sad(maria).
loves(jason, maria).
cry(X) :- sad(X).
plays_soccer(X) :- cry(X).
```

What is Prolog?

Example 3

```
sad(jason).
sad(maria).
loves(jason, maria).
cry(X) :- sad(X).
plays_soccer(X) :- cry(X).
```

Example 4

```
loves(a,maria).
loves(b,maria).
loves(richard,margaret).
loves(margaret,richard).
jealous(X,Y):- loves(X,Z), loves(Y,Z).
```

Prolog Syntax

Constant: A constant is either a string of characters made up of upper-case letters, lower-case letters, digits, and the underscore character, that begins with a lower-case letter, or an arbitrary sequence of characters enclosed in single quotes.

a, a_a, q66,'A b'

Prolog Syntax

Constant: A constant is either a string of characters made up of upper-case letters, lower-case letters, digits, and the underscore character, that begins with a lower-case letter, or an arbitrary sequence of characters enclosed in single quotes.

a, a_a, q66,'A b'

Variable: A variable is a string of upper-case letters, lower-case letters, digits and underscore characters that starts either with an upper-case letter or with an underscore.

X, IM4_u, _k, _

Prolog Syntax

Complex term or Structure: Complex terms are build out of a functor followed by a sequence of arguments. The arguments are put in ordinary parentheses, separated by commas, and placed after the functor.

loves(maria, pet(X))

Prolog Syntax

Complex term or Structure: Complex terms are build out of a functor followed by a sequence of arguments. The arguments are put in ordinary parentheses, separated by commas, and placed after the functor.

loves(maria, pet(X))

The number of arguments of a complex term is called its arity.

loves/2, maria/0

Lists

Inductive definition

Prolog has a built-in syntax for lists, where a list is either:

- the empty list, written **[ ]**, or
- an elemenn **x** and a list **xs**, written [**x** | **xs**].

Lists

Inductive definition

Prolog has a built-in syntax for lists, where a list is either:

- the empty list, written **[ ]**, or
- an elemenn **x** and a list **xs**, written [**x** | **xs**].

[[], loves(x, Y), [2, [b, c]], [], Z, [2, [b, c]]]

Lists

Example

```
-- Length of a list.
length( [], 0 ).
length( [ _ | Y ], N ) :- length( Y, M ) ,
                           N is M + 1 .

-- Given an object X and a list Y, tests if
-- X belongs to Y.
member( X, [ X | _ ]).
member( X, [ _ | Y ] ) :- member( X, Y ).

-- Our own definition for lists.
list([]).
list([_|X]) :- list(X).
```

Recursion - Arithmetic

Example

```
-- Our own definition for natural numbers.
nat(0).
nat(succ(X)) :- nat(X).

-- Sum for natural numbers.
add(0,X,X).
add(succ(X), Y, succ(Z)) :- add(X, Y, Z).
```

Unification

The aim of the unification of two terms A and B is to find a substitution for its variables, such that the two terms become identical.

Example

A	***B***	***Substitution***
pet(cat)	*pet(cat)*	$\{\ \}$
X	*Y*	$\{X/Y\}$
X	*a*	$\{X/a\}$
f(X, g(t))	*f(m(h), g(M))*	$\{X/m(h), M/t\}$
f(X, X)	*f(a, b)*	*Impossible*

Execution Tree

Example

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

Execution Tree

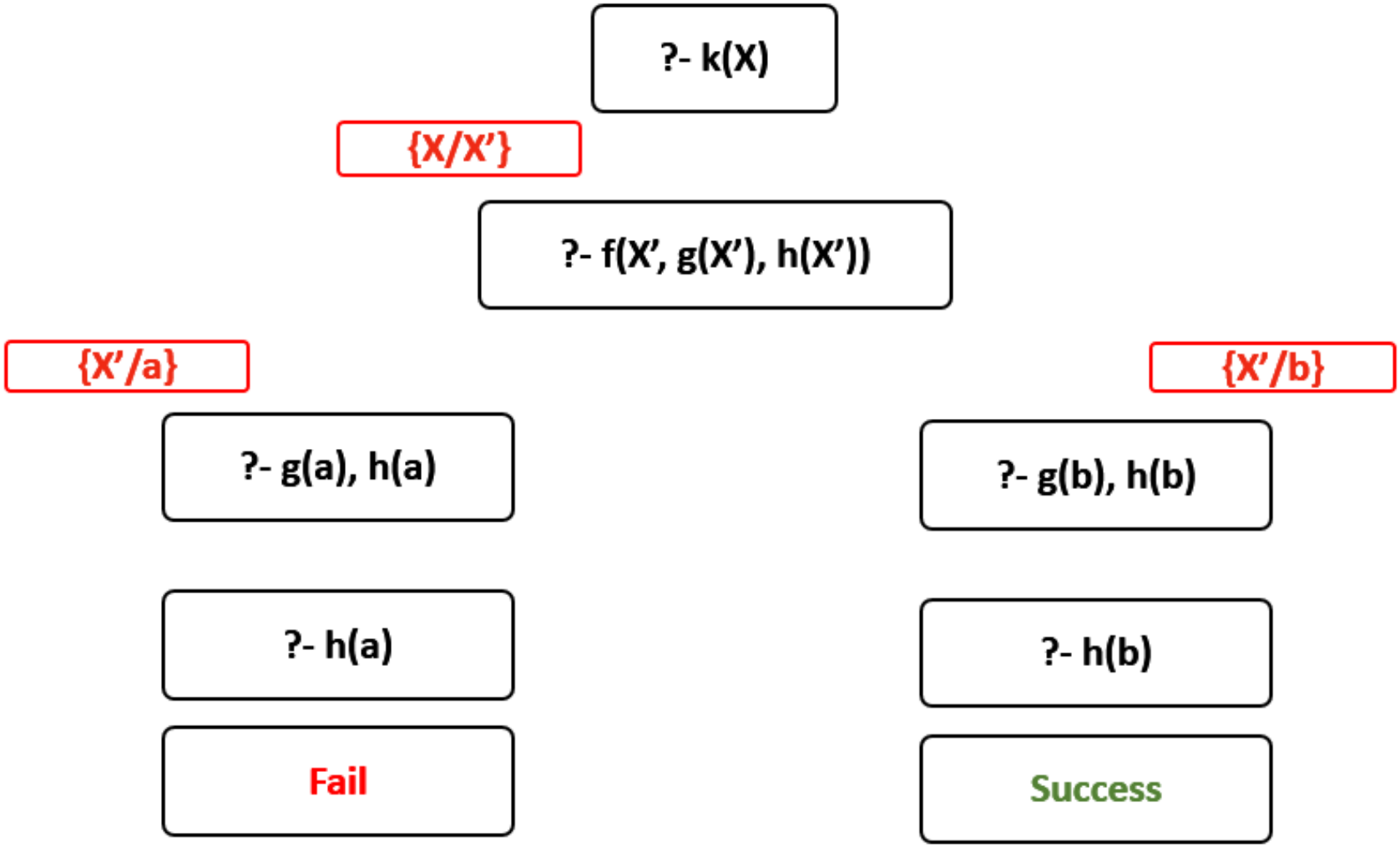
Example

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

How does Prolog work?

How does Prolog process this query: `k(X)`?

Execution Tree



Some Links

- **Nice Tutorial**

  See http://lpn.swi-prolog.org/lpnpage.php?pageid=top.

- **Documentation**

  See http://www.swi-prolog.org/.