Prolog Tutorial-I

Dr A Sahu
Dept of Computer Science & Engineering
IIT Guwahati

Outline

- What is Prolog?
- How to install prolog?
- An example program
- Syntax of terms
- Some simple programs
- Terms as data structures
- **Unification and The Cut : next tutorial**

What is Prolog?

- Prolog is the most widely used language to have been inspired by logic programming research.
- Logic program: **consist of facts and rules**
- Computation **: is deduction**
- Some features:
  - Prolog uses **logical variables**. These are **not the same as variables in other languages**.
  - Programmers can use **logical variable** as '**holes**' in data structures that are **gradually filled in as computation proceeds**.

What is Prolog?

- **Unification** is a built-in term-manipulation method
  - that passes parameters, returns results, selects and constructs data structures.
- Basic control flow model: **Backtracking**
- Clauses and data have : Same form
- Relation treat arguments and results uniformly
- The **relational form** of procedures makes it possible to define **'reversible' procedures**.

What is Prolog?

- Clauses provide a convenient way to express
  - Case analysis
  - Nondeterminism.
- *Sometimes it is necessary to use control features that are not part of 'logic'*.
- A Prolog program can also be seen as a relational database containing rules as well as facts.

How to install prolog?

- Linux Ubuntu
  - $sudo apt-get install swi-prolog-nox
  - $sudo apt-get install gprolog
- Window
  - https://www.swi-prolog.org/download/stable
- Webshell:
  - https://swish.swi-prolog.org/

How to install prolog?

- Programming in Prolog, bu clocksin and melish
- Learn Prolog Now [online], LPN Home
  - By blackburn,..

Prolog: Hello World Program

```
| ?- write('hellow World').
hellow World
yes
| ?- write("hellow World").
[104,101,108,108,111,119,32,87,111,11
  4,108,100]
yes
```

Compare Prolog: Relational Database and Queries

Relation

A concrete view of relation is a table with $n \geq 0$ columns ad a possible infinite set of rows

A tuple $(a_1, a_2, \ldots, a_n)$ is n a relation of a_i appears in column i, $1 \leq i \leq n$, of some row in the table

Compare Prolog: Relational Database and Queries

Logic programming deal with relation rather than functions

- Based on premise the programming with relation is more flexible then programming function
- Because relation treat arguments and result uniformly
- Informally
  - Relation have no sense of direction
  - No prejudice about who is computed from whom

Prolog Relational Database: Example

[a, b, c]= [a|[b,c]] = [Head is symbol | Tail is list]

Relation **append** is a set of tuples of the form (X,Y,Z) where Z consist if X followed by the element of Y.

X	Y	Z
[]	[]	[]
[a]	[]	[a]
…	…	….
[a,b]	[c,d]	[a,b,c,d]
…..	…….	………

Relation are also called ***predicates.***

Query : Is a given tuple in relation append?

```
?-append([a],[b],[a,b]).
yes
```

```
?-append([a],[b],[]).
no
```

Writing append relation in prolog

Rules

```
append([],Y,Y).
append([H|X],Y,[H|Z]):- append(X,Y,Z).
```

Queries

```
?-append ([a,b],[c,d],[a,b,c,d]).
 yes
?-append([a,b],[c,d],Z).
Z=[a,b,c,d]
?-append([a,b],Y,[a,b,c,d]).
Y=[c,d]
?-append(X,[c,d],[a,b,c,d]).
X=[a,b]
?-append(X,[d,c],[a,b,c,d]).
no
```

Prolog is a 'Declarative' language

- Clauses are statements about **what is true about a problem**, instead of instructions how to accomplish the solution.
- The Prolog system uses the clauses to work out how to accomplish the **solution by searching through the space of possible solutions**.
- *Not all problems have pure declarative specifications. Sometimes extralogical statements are needed.*

What a program looks like

```
/* At the Zoo */
elephant(gaj).
elephant(aswasthama).

panda(chi_chi).
panda(ming_ming).
```

Facts

```
dangerous(X)  :- big_teeth(X).
dangerous(X)  :- venomous(X).
guess(X, tiger):- stripey(X), big_teeth(X),
                  isaCat(X).
guess(X, zebra):- stripey(X), isaHorse(X).
```

Rules

Example: Concatenate lists a and b

Imperative language

```
node *concat(node*list1,node *list2){
      node *p;     p=list1;
      while(p->next->next!=NULL)
                   p=p->next;
      p->next=list2;
      return(list1);
}
```

functional language

```
cat(a,b) ≡
 if b = nil then a
else cons(head(a), cat(tail(a),b))
```

Declarative language

```
cat([], Z, Z).
cat([H|T], L, [H|Z]) :- cat(T, L, Z).
```

Factorial Program

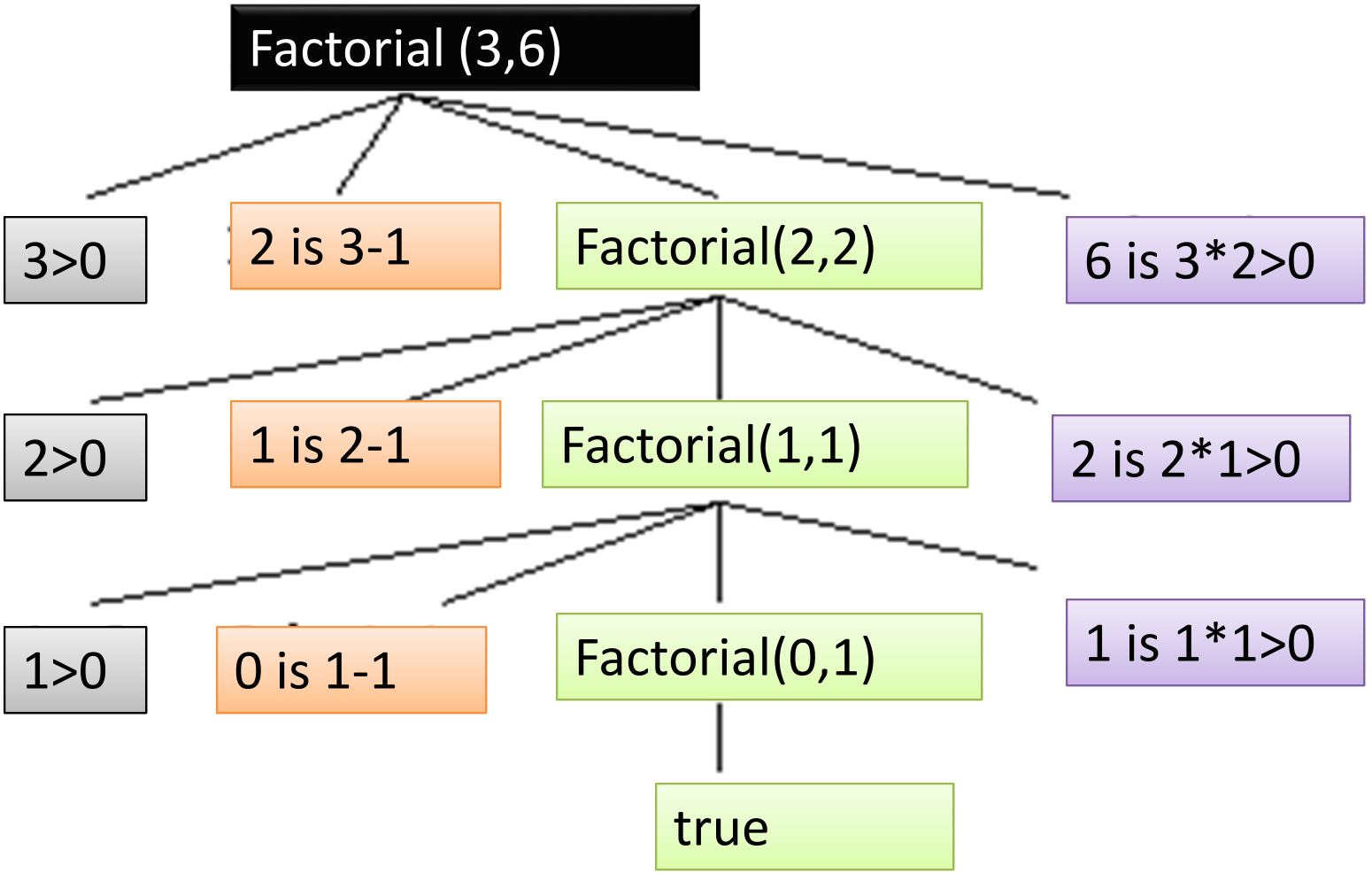
```
factorial(0,1).
factorial(N,F):- N>0, N1 is N-1,
    factorial(N1,F1),F is N * F1.
```

The Prolog goal to calculate the factorial of the number 3 responds with a value for W, the goal variable:

```
?- factorial(3,W).
   W=6
```

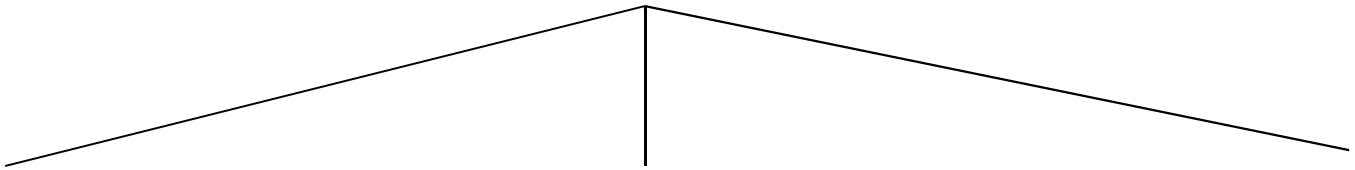
Factorial Program Evaluation

```
factorial(0,1).
factorial(N,F):- N>0, N1 is N-1, factorial(N1,F1),F is N*F1.
```



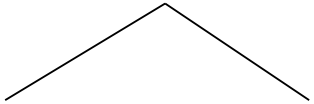
Complete Syntax of Terms

Term



Constant

Names an individual



Atom

alpha17
gross_pay
john_smith
dyspepsia
+
=/=
'12Q&A'

Number

0
1
57
1.618
2.04e-27
-13.6

Compound Term

Names an individual that has parts

likes(john, mary)
book(dickens, Z, cricket)
f(x)
[1, 3, g(a), 7, 9]
-(+(15, 17), t)
15 + 17 - t

Variable

Stands for an individual unable to be named when program is written

X
Gross_pay
Diagnosis
_257
_

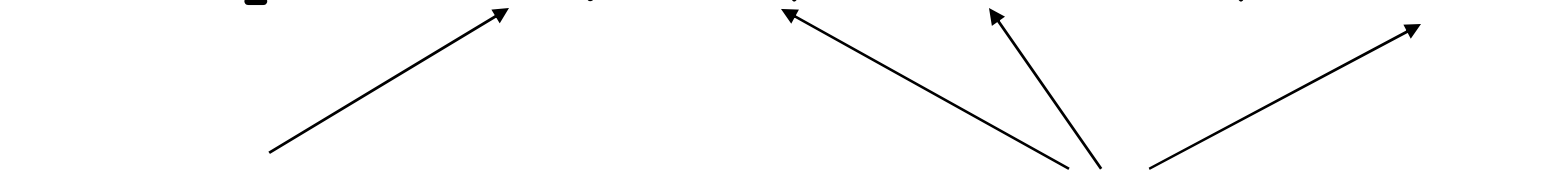
General Rules

- variable start with
  - Capital letter or underscore
  - Mostly we use Capital X,Y,Z,L,M for variable
- atom start with
  - Mostly word written in small letters
  - likes, john, mary in likes (john, mary).
  - elephant gaj in elephant(gaj).

Compound Terms

The parents of Rama are Dasarath and Kousalya.

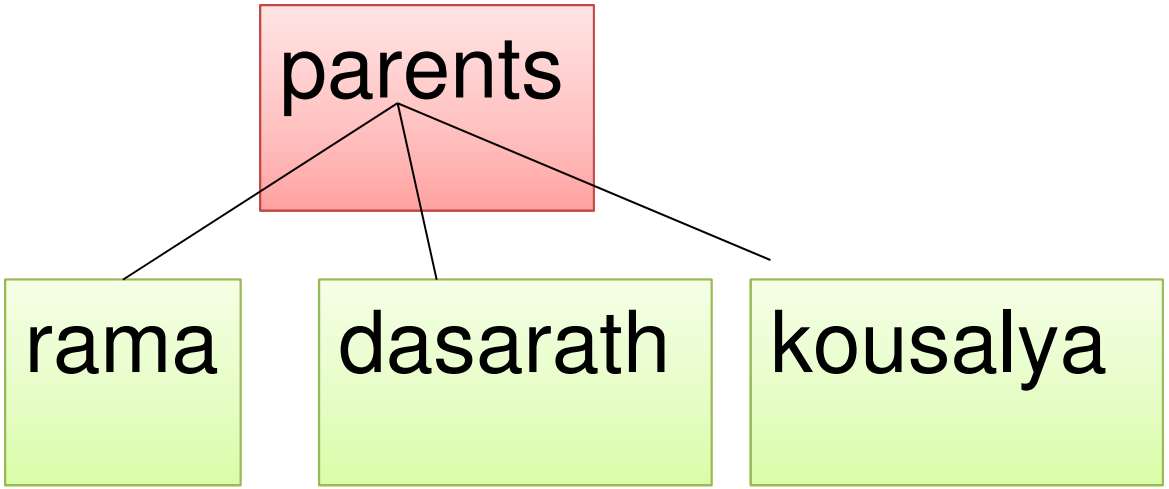
```
parents(rama, dasarath, kousalya)
```



Functor (an atom) of arity 3.

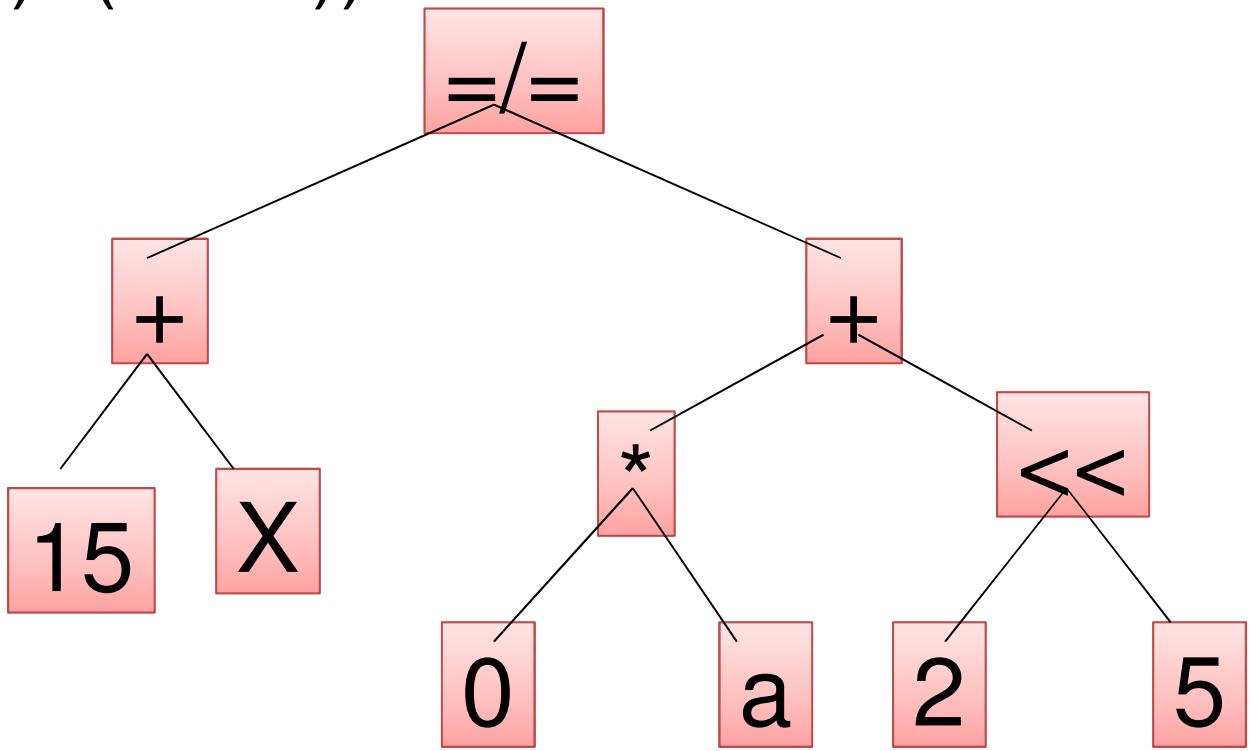
components (any terms)

It is possible to depict the term as a tree:



Compound Terms: Example

=/=(15+X, (0*a)+(2<<5))



X =\= Y **means X and Y stands for different numbers**

More about operators

- Any atom may be designated an operator. The only purpose is for convenience; the only effect is how the term containing the atom is parsed.
- Operators are '**syntactic sugar**'.
  - ***Easy to write in our own way***
- Operators have three properties: position, precedence and associativity.

Examples of operator properties

Position	Operator Syntax	Normal Syntax
Prefix:	-2	-(2)
Infix:	5+17	+(17,5)
Postfix:	N!	!(N)

Associativity: left, right, none.

X+Y+Z is parsed as (X+Y)+Z

because addition is left-associative.

These are all the same as the normal rules of arithmetic.

Precedence: an integer.

X+Y*Z is parsed as X+(Y*Z)

because multiplication has higher precedence.

Logical Operation on Numbers

X =:= Y	X and Y stands for the same number
X =\= Y	X and Y stands for different numbers
X < Y	X is less then Y
X > Y	
X =< Y	**Not same as in C (>=, <=)**
X >= Y	

The last point about Compound Terms…

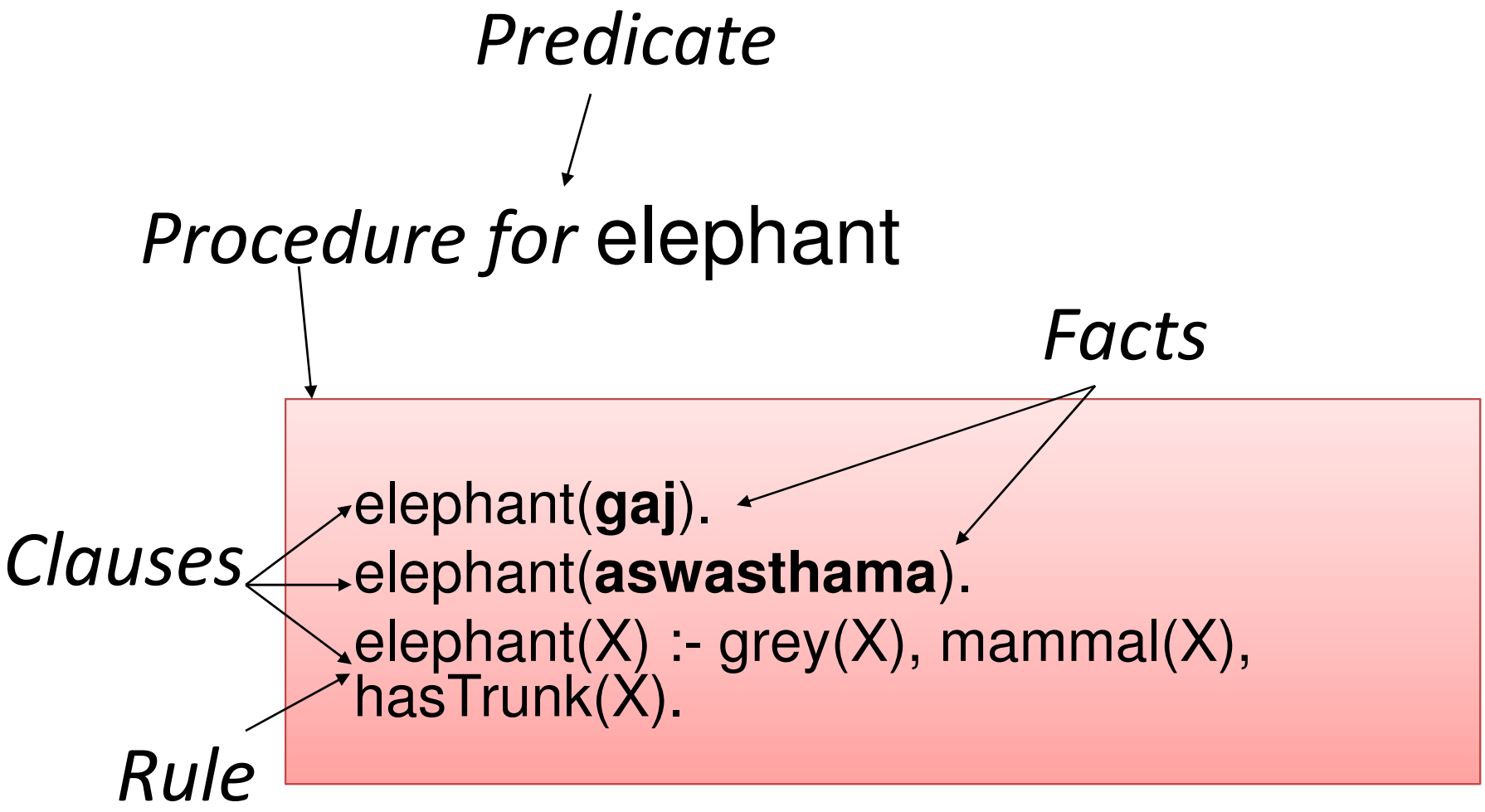
Constants are simply compound terms of arity 0.

badger means the same as
badger()

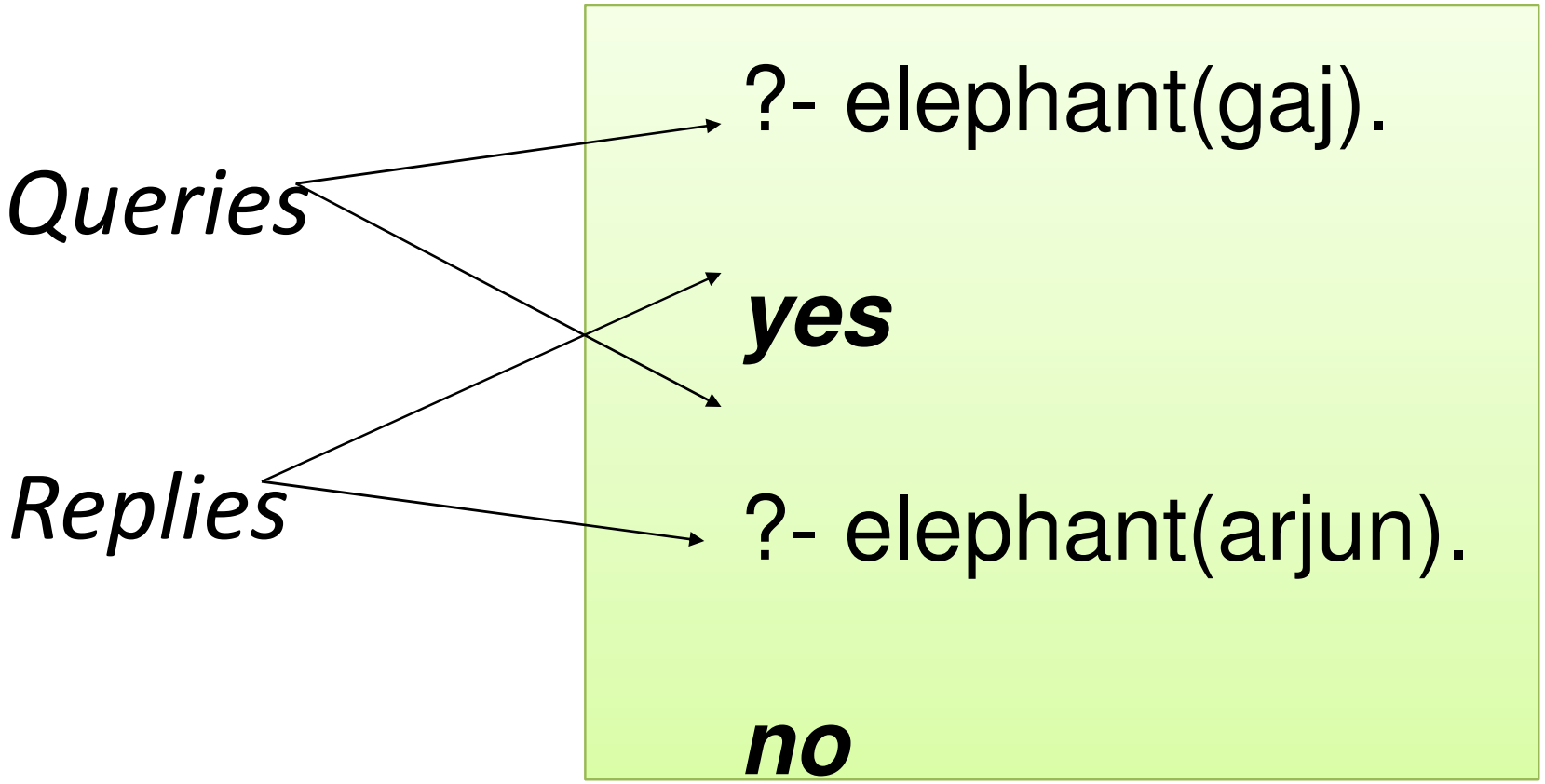
Structure of Prolog Programs

- Programs consist of procedures.
- Procedures consist of clauses.
- Each clause is a fact or a rule.
- Programs are executed by posing queries.

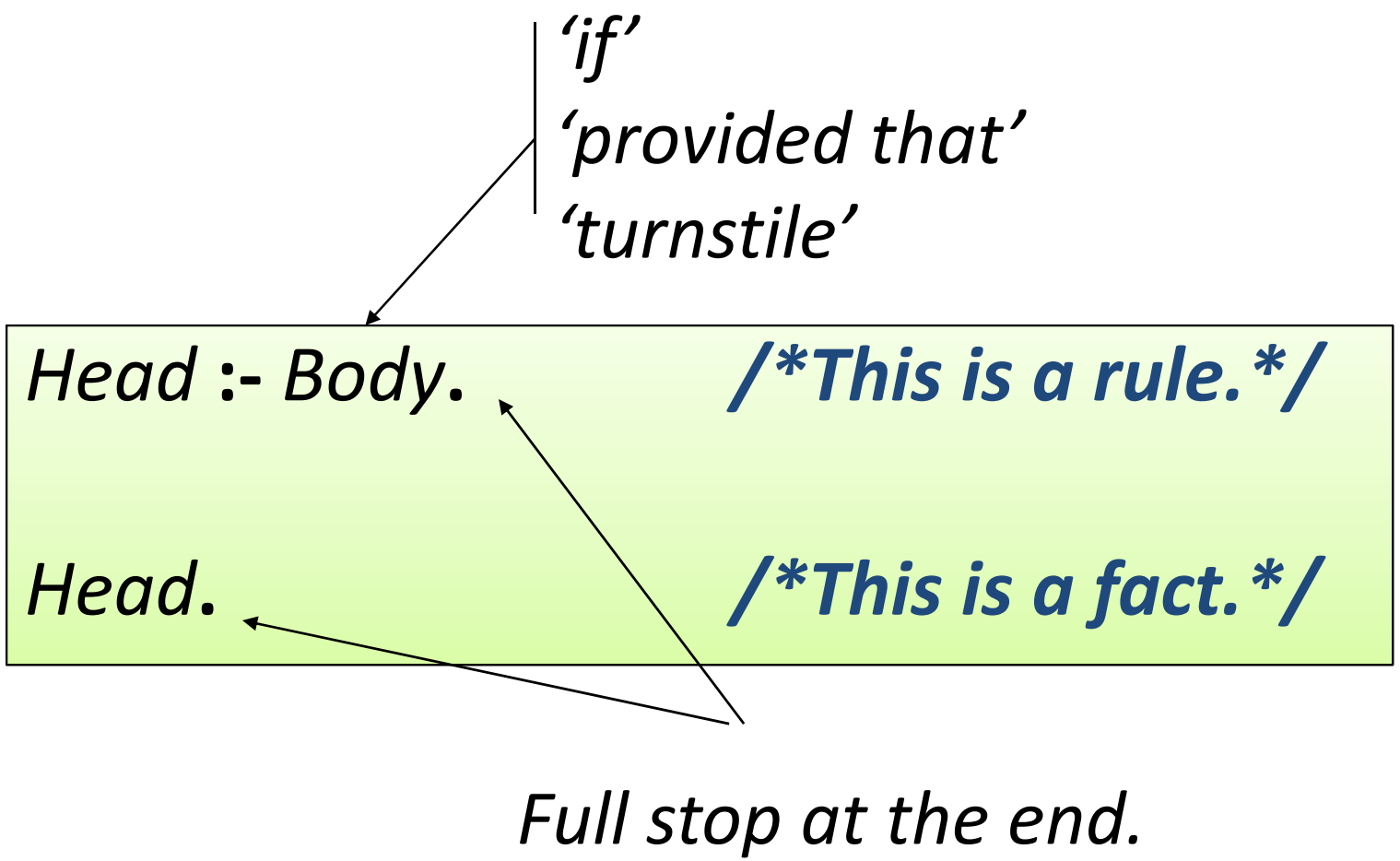
An Example


Predicate
Procedure for elephant
Facts
Clauses
Rule
elephant(gaj).
elephant(aswasthama).
elephant(X) :- grey(X), mammal(X), hasTrunk(X).

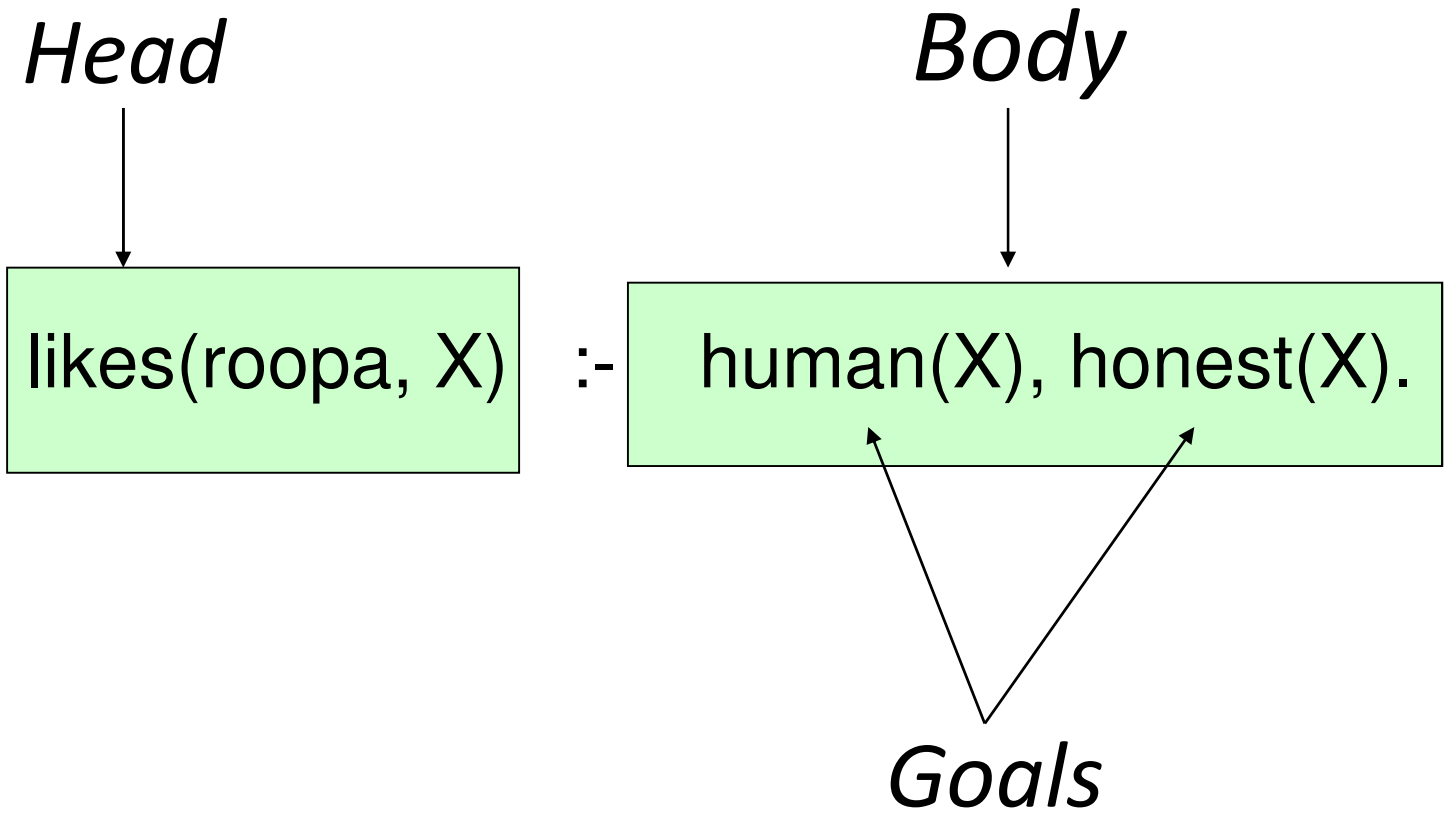
Example



Clauses: Facts and Rules



Body of a (rule) clause contains goals.



Interpretation of Clauses

Clauses can be given a declarative reading or a procedural reading.

Form of clause:

HORN Clause

$H \;:-\; G_1,\; G_2,\; \ldots,\; G_n.$

Declarative reading:

"That H is provable follows from goals G_1, G_2, …, G_n being provable."

Procedural reading:

"To execute procedure H, the procedures called by goals G_1, G_2, …, G_n are executed first."

Another Example

Program

```
male(rohan).
male(vivek).
female(purbi).
female(neha).
pair(X,Y):-male(X),
          female(Y).
```

Queries

```
?- pair(vivek, X).
?- pair(ram, sita).
?- pair(neha, X).
?- pair(X, purbi).
?- pair(X, X).
?- pair (rohan,purbi).
?- pair(X, dhanush).
?- pair(X, Y).
```

?- consult('test1.pl').
?-pair(vivek, X). //use ; get next option

Example 2

```
drinks(raj, tea).
drinks(mann, coffee).
drinks(sagar, juice).
drinks(raj, coffee).
drinks(kush, coffee).

pair(X, Y, Z) :-
      drinks(X, Z),
      drinks(Y, Z).
```

```
?- pair(X,raj,tea).
?- pair(mann,sagar,coffee).
?- pair(raj,mann,coffee).
?- pair(raj,raj,coffee).
?- pair(X,Y,coffee).
?- pair(sita,gouri,juice).
?- pair(X, Y, Z).
```

This definition forces X and Y to be

```
pair(X,Y,Z):- drinks(X,Z), drinks(Y,Z), X \== Y.
```

Another Examples: Density Calculation

```
%popultaion.pl  Population in Million
pop(usa,280).     pop(india,1000).
pop(china,1200). pop(brazil,130).
area(usa,3). /* millions of sq miles */
area(india,1). area(china,4).
area(brazil,3).

density(X,Y)  :- pop(X,P),
                 area(X,A),
                 Y is P/A.
```

The population density of country X is Y, if:

The population of X is P, **and**

The area of X is A, **and**

Y is calculated by dividing P by A.

Examples: Density Calculation

```
?- consult(population.pl).
% population compiled 0.00 sec,
  1,548 bytes
Yes
?- density(usa,D).
D = 93.3333
Yes
?- density(china,D).
D = 300
Yes
```

Thanks