Prolog Programming

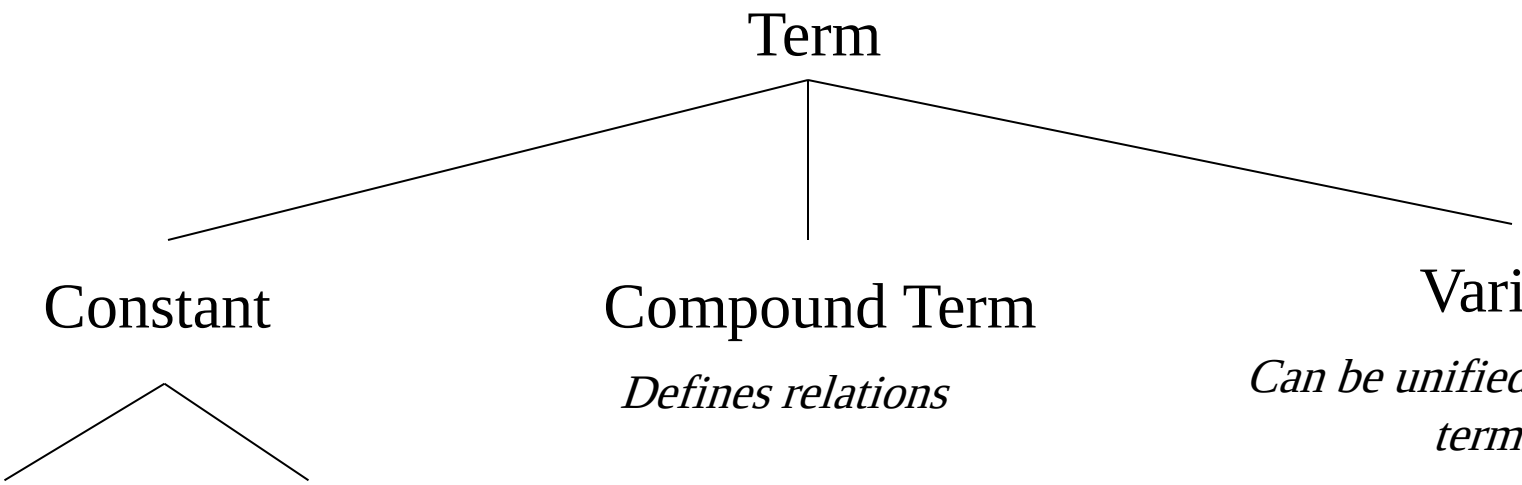
More on syntax

12-5-2024

The Plan

- Syntax of terms
- Some simple programs
- Terms as data structures, unification
- Writing programs

Complete Syntax of Terms



alpha17
gross_pay
ali_salem
car
+
=/=
’12Q&A’

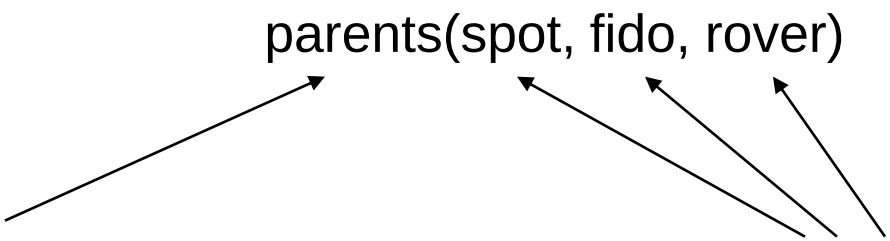
0
1
57
1.618
2.04e-27
-13.6

likes(john, apples)
give(‘Salem’, ‘Book’, to(ali))
f(x)
[1, 3, g(a), 7, 9]
-(+(15, 17), t)
15 + 17 - t

X
Gross_pay
Diagnosis
_257
_

Compound Terms

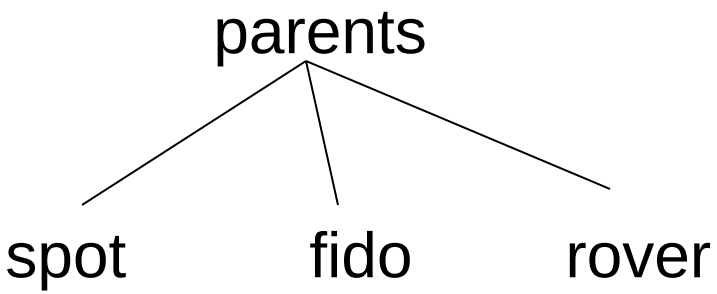
The parents of Spot are Fido and Rover.



Functor (an atom) of arity 3.
parents/3

components (any terms)

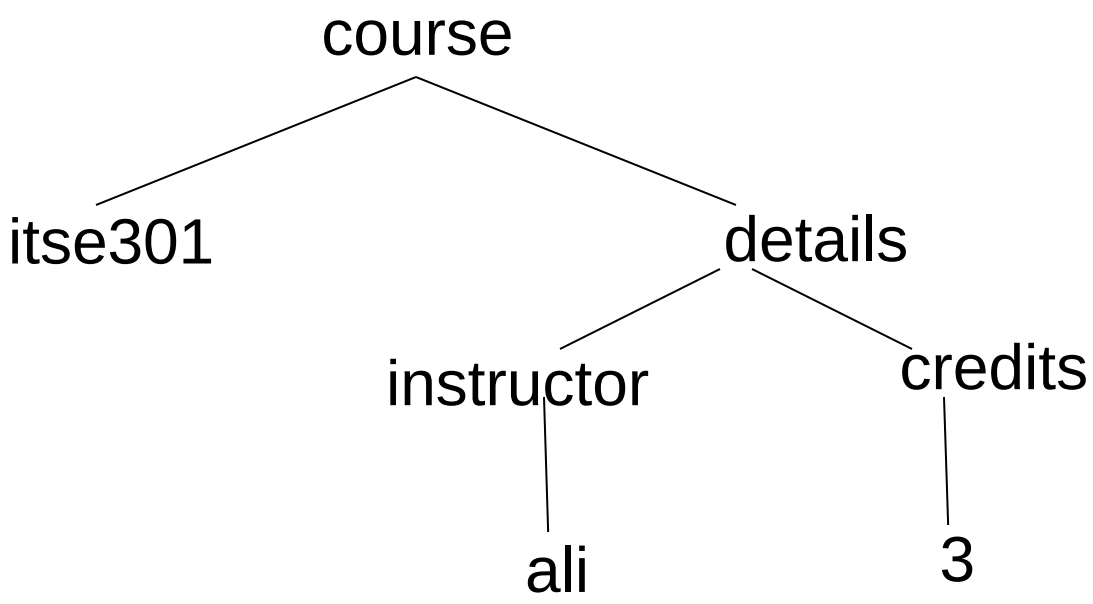
It is possible to depict the term as a tree:



Compound Terms

Any compound term can be represented as tree.

course(itse301,details(instructor(ali), credits(3))).



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:

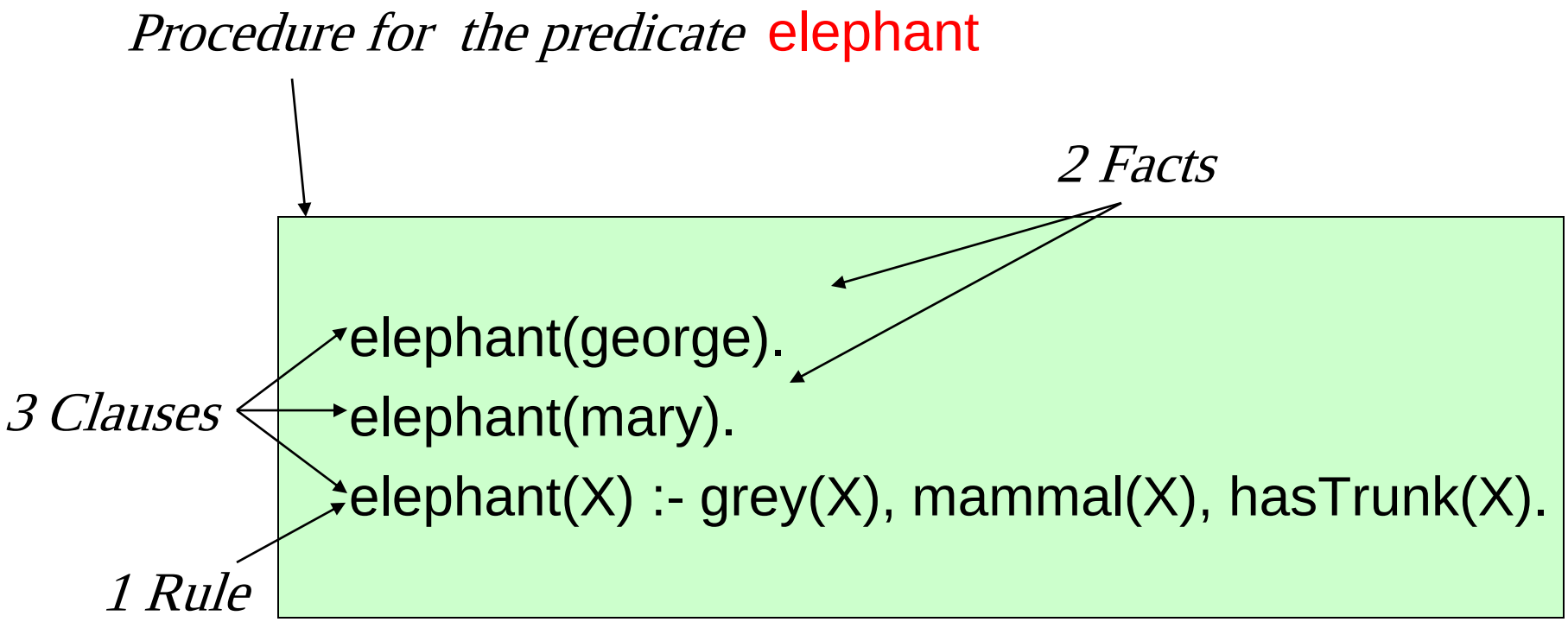
X+Y*Z is parsed as X+(Y*Z)

because multiplication has higher precedence.

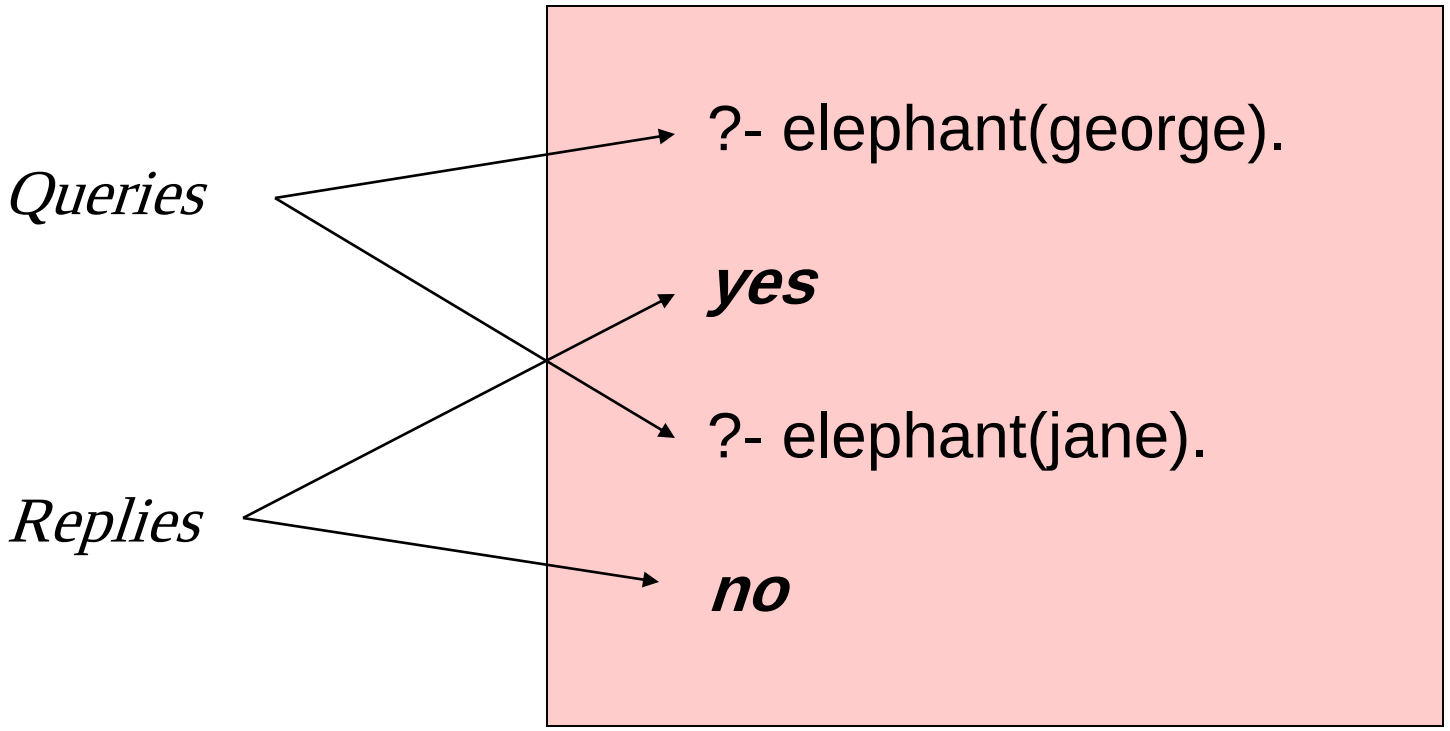
Structure of Programs

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

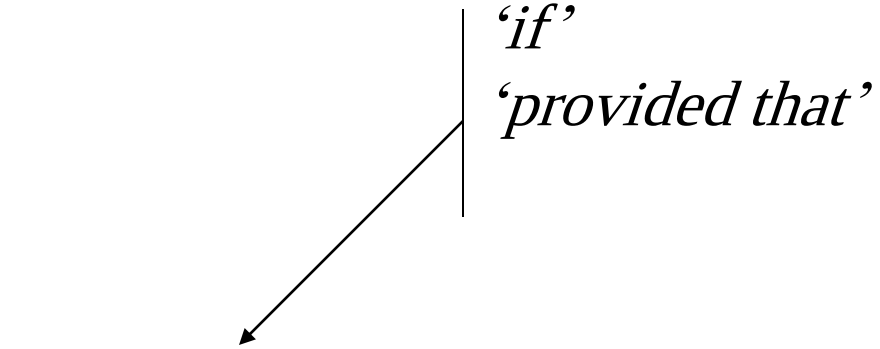
Example


Procedure for the predicate elephant
2 Facts
elephant(george).
elephant(mary).
elephant(X) :- grey(X), mammal(X), hasTrunk(X).
3 Clauses
1 Rule

Example



Clauses: Facts and Rules

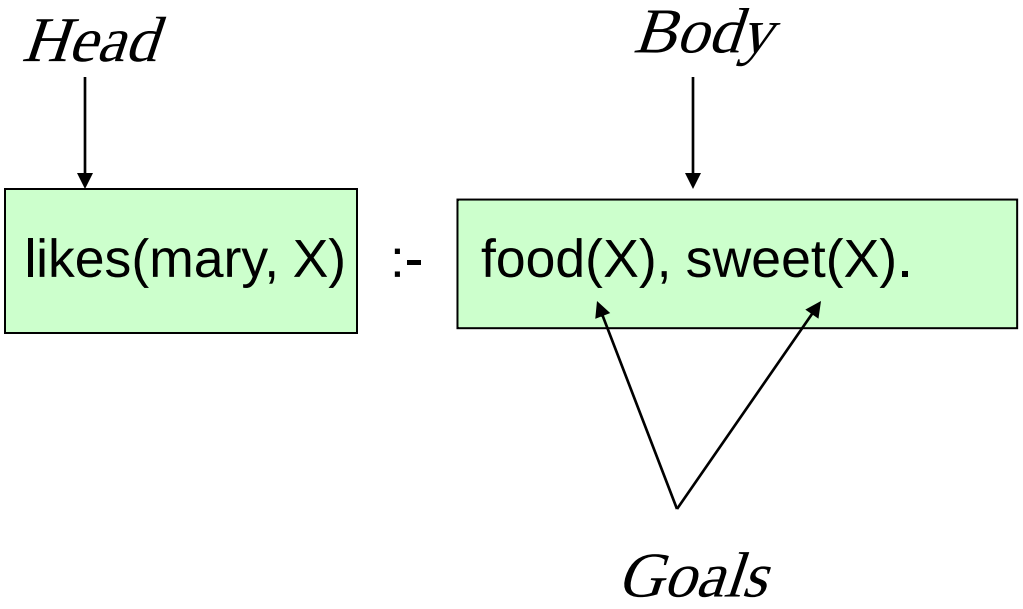


Head :- *Body*. ← This is a rule.

Head. ← This is a fact.

Clauses must end with Fullstop.

The body of a rule clause contains one or more goals that must hold for the head to hold.



Question: Talk about all parts of Prolog syntax you have seen so far.

```
% kb defining possible pairs of
% students who can work on a
% homework
male(ali).
male(salem).
male(khalid.
female(alya).
female(noora).
female(najwa).

pair(X, Y) :- male(X), male(Y).
pair(X, Y) :- female(X), female(Y).
```

```
?- pair(salem, X).
?- pair(khalid, X).
?- pair(noora, X).
?- pair(X, alya).
?- pair(X, X).
?- pair(ali, alya).
?- pair(X, khalid).
?- pair(X, Y).
```

Worksheet 2

```
likes(ali, java).
likes(khalid, ‘c++’).
likes(salem, java).
likes(noora, java).
likes(najwa, ‘c++’).

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

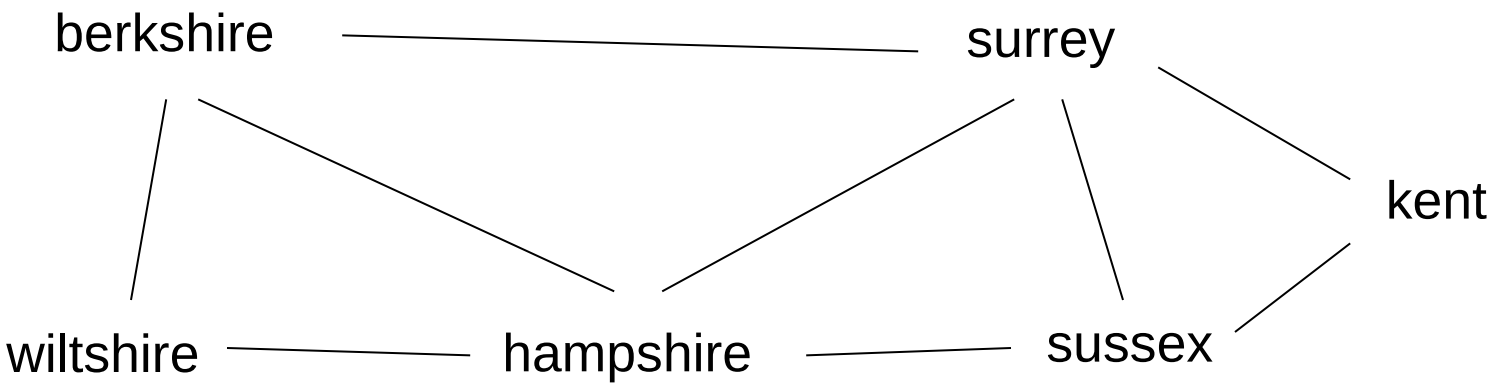
```
?- pair(X, ali).
?- pair(ali, X).
?- pair(salem, X).
?- pair(salem, khalid).
?- pair(X, Y).
?- pair(najwa, noora).
```

This definition forces X and Y to be distinct:

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

Worksheet 3

(a) Representing a symmetric relation.
(b) Implementing a strange ticket condition.



How to represent this relation?
Note that borders are symmetric.

WS3

This relation represents one ‘direction’ of border:

```
border(sussex, kent).
border(sussex, surrey).
border(surrey, kent).
border(hampshire, sussex).
border(hampshire, surrey).
border(hampshire, berkshire).
border(berkshire, surrey).
border(wiltshire, hampshire).
border(wiltshire, berkshire).
```

What about the other?

(a) Say

```
border(kent, sussex).
border(sussex, kent).
```

⋮

(b) Say

```
adjacent(X, Y) :- border(X, Y).
adjacent(X, Y) :- border(Y, X).
```

(c) Say

```
border(X, Y) :- border(Y, X).
```

WS3

Now a somewhat strange type of discount ticket. For the ticket to be valid, one must pass through an intermediate county.

A valid ticket between a start and end county obeys the following rule:

```
valid(X, Y) :- adjacent(X, Z), adjacent(Z, Y)
```

WS3

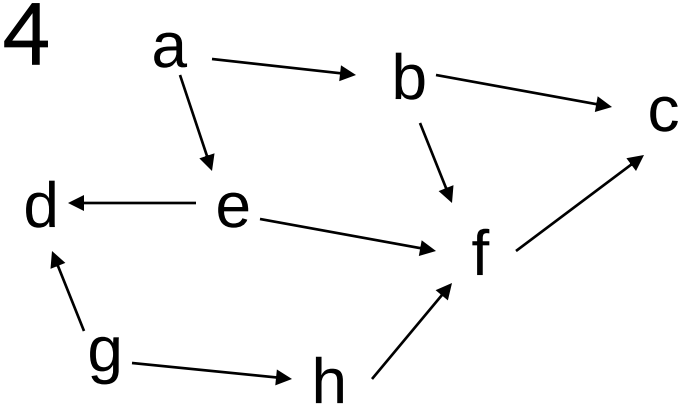
```
border(sussex, kent).
border(sussex, surrey).
border(surrey, kent).
border(hampshire, sussex).
border(hampshire, surrey).
border(hampshire, berkshire).
border(berkshire, surrey).
border(wiltshire, hampshire).
border(wiltshire, berkshire).

adjacent(X, Y) :- border(X, Y).
adjacent(X, Y) :- border(Y, X).
```

```
valid(X, Y) :-
        adjacent(X, Z),
        adjacent(Z, Y)
```

```
?- valid(wiltshire, sussex).
?- valid(wiltshire, kent).
?- valid(hampshire, hampshire).
?- valid(X, kent).
?- valid(sussex, X).
?- valid(X, Y).
```

Worksheet 4



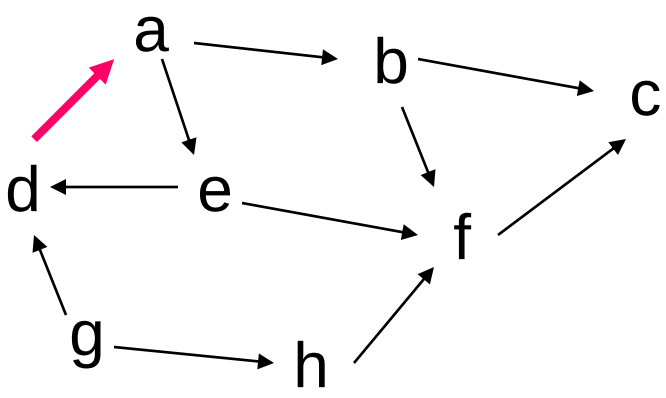

```
a(g, h).
a(g, d).
a(e, d).
a(h, f).
a(e, f).
a(a, e).
a(a, b).
a(b, f).
a(b, c).
a(f, c).
```

Note that Prolog can distinguish between the 0-ary constant a *(the name of a node) and the 2-ary functor* a *(the name of a relation).*

```
path(X, X).
path(X, Y) :- a(X, Z), path(Z, Y).
```

```
?- path(f, f).
?- path(a, c).
?- path(g, e).
?- path(g, X).
?- path(X, h).
```

But what happens if…




```
a(g, h).
a(g, d).
a(e, d).
a(h, f).
a(e, f).
a(a, e).
a(a, b).
a(b, f).
a(b, c).
a(f, c).
a(d, a).
```

```
path(X, X).
path(X, Y) :- a(X, Z), path(Z, Y).
```

This program works only for acyclic graphs. The program may infinitely loop given a cyclic graph. We need to leave a ‘trail’ of visited nodes. This is accomplished with a data structure (to be seen later).

Unification

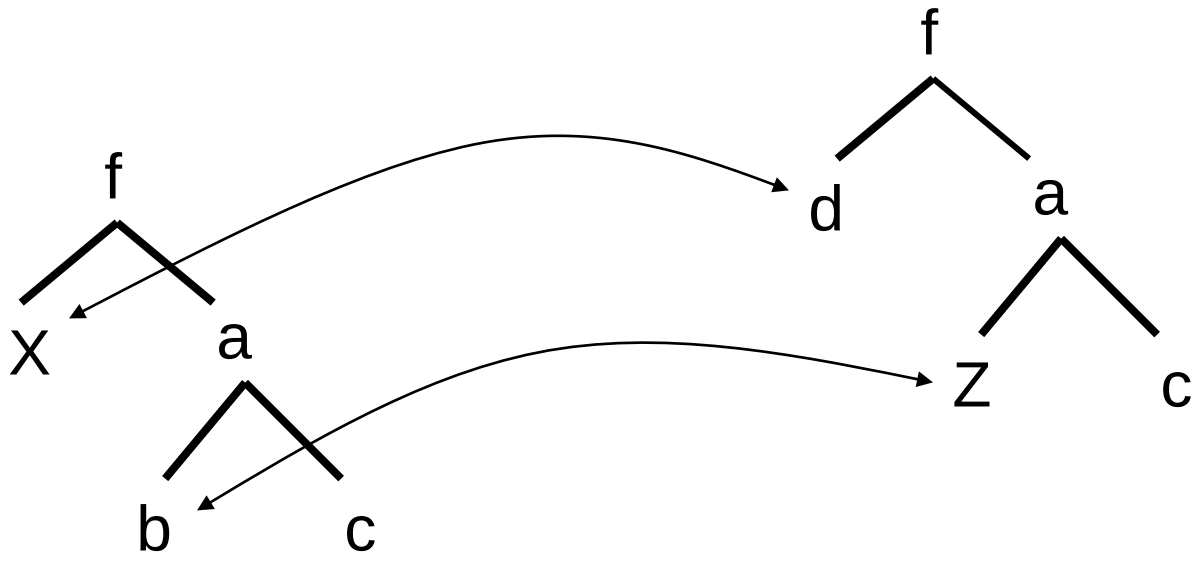
- Unification is a term-manipulation technique that passes parameters, returns results, selects and constructs data structures.
- Basic control flow model is backtracking.
- Program clauses and data have the same form.
- The relational form of procedures makes it possible to define ‘reversible’ procedures.

Unification

- Two terms unify if substitutions can be made for any variables in the terms so that the terms are made identical. If no such substitution exists, the terms do not unify.
- The Unification Algorithm proceeds by recursive descent of the two terms.
  - Constants unify if they are identical
  - Variables unify with any term, including other variables
  - Compound terms unify if their functors and components unify.

Examples

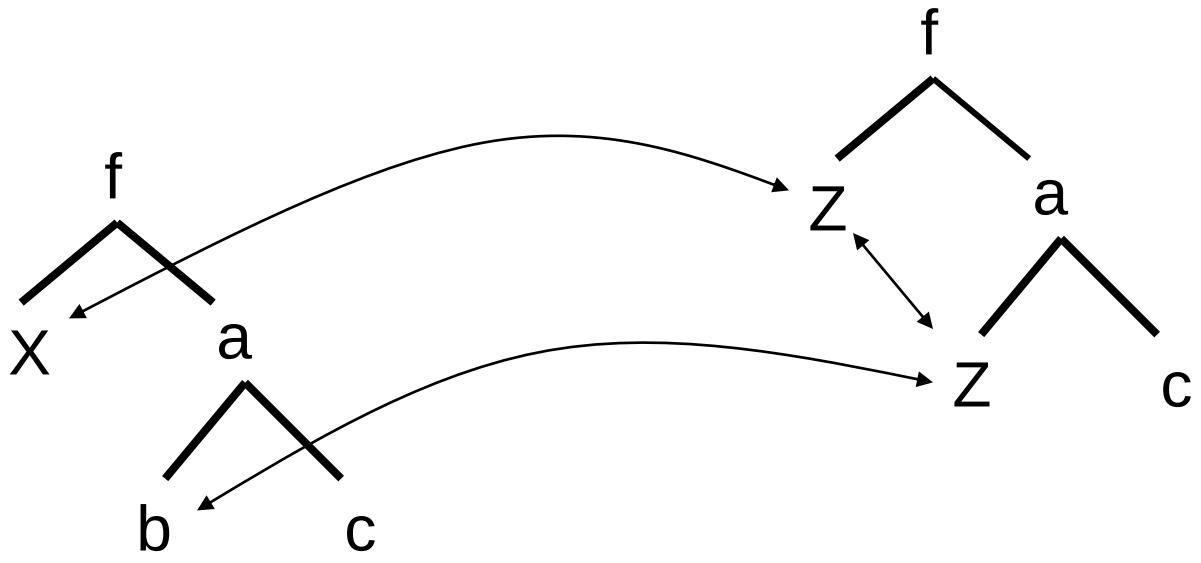
The terms f(X, a(b,c)) and f(d, a(Z, c)) unify.



The terms are made equal if d is substituted for X, and b is substituted for Z. We also say X is instantiated to d and Z is instantiated to b.

Examples

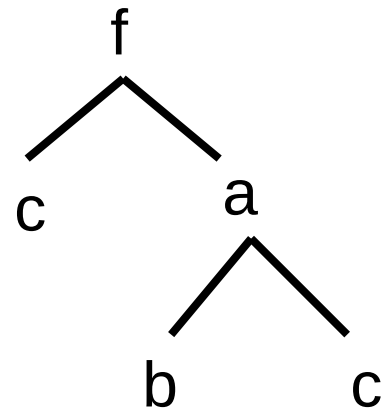
The terms f(X, a(b,c)) and f(Z, a(Z, c)) unify.

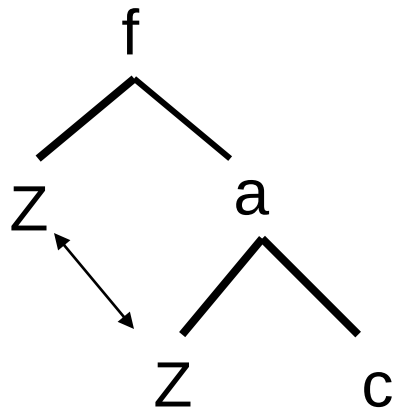


Note that Z co-refers within the term.

Examples

The terms f(c, a(b,c)) and f(Z, a(Z, c)) do not unify.





No matter how hard you try, these two terms cannot be made identical by substituting terms for variables.