

# Function Symbols in Prolog: From Deductive Databases to Logic Programs

In logic, there are two kinds of objects: predicates and functions.

- Predicates represent statements about the world:

John hates Mary: `hates(john,mary)`.

John is short: `short(john)`

(`hates` is a predicate symbol, `short(john)` is an atomic formula)

- Function terms represent objects in the world
    - the mother of Mary: `mother-of(mary)`
    - a rectangle of length 3 and width 4:  
`rectangle(3,4)`
- (`mother-of(mary)` is a function term, `rectangle` is a function symbol)

Function terms do not have values. In Prolog, they act as data structures:

```
let p2(X,Y) denote a point in 2-dim space
let p3(X,Y,Z) denote a point in 3-dim space.
```

Write a Prolog program, SQDIST(Point1,Point2,D), that returns the square of the distance between two points. The program should work for 2- and 3-dim points.

Want:

```
SQDIST(p2(1,2) , p2(3,5) , D)
  returns D = (3-1)**2 + (5-2)**2
           = 4+9 = 13
and
SQDIST(p3(1,1,0) , p3(2,2,3) , D)
  returns D = (1-2)**2 + (1-2)**2 + (0-3)**2
           = 1+1+9 = 11
and
SQDIST(p2(0,0) , p3(1,1,1) , D)
  is undefined
```

Prolog Program:

```
(1) SQDIST(p2(X1, Y1), p2(X2, Y2), D)
    :- XD is X1-X2,
       YD is Y1-Y2,
       D is XD*XD + YD*YD.
```

```
(2) SQDIST(p3(X1, Y1, Z1), p3(X2, Y2, Z2), D)
    :- XD is X1-X2,
       YD is Y1-Y2,
       ZD is Z1-Z2,
       D is XD*XD + YD*YD + ZD*ZD.
```

Query: SQDIST(p2(1,2), p2(3,5), D)

This query unifies with the head of rule (1) with {X1\1, Y1\2, X2\3, Y2\5}

SO, XD is X1-X2 = 1-3 = -2

YD is Y1-Y2 = 2-5 = -3

D is  $(-2)^2 + (-3)^2 = 13$

So, D=13 is returned

Note: the query does not unify with the head of rule (2), so only rule (1) is used.

## Prolog Program:

```
(1) SQDIST(p2(X1, Y1), p2(X2, Y2), D)
    :- XD is X1-X2,
       YD is Y1-Y2,
       D is XD*XD + YD*YD.
```

```
(2) SQDIST(p3(X1, Y1, Z1), p3(X2, Y2, Z2), D)
    :- XD is X1-X2,
       YD is Y1-Y2,
       ZD is Z1-Z2,
       D is XD*XD + YD*YD + ZD*ZD.
```

Query: SQDIST(p3(1,1,0),p3(2,2,3),D).

This query unifies with the head of rule (2),  
with {X1\1, Y1\1, Z1\0, X2\2, Y2\2, Z2\3}  
so, XD is 1-2 = -1  
YD is 1-2 = -1  
ZD is 0-3 = -3  
D is 1+1+9 = 11  
So, D=11 is returned

Note: the query does not unify with the head  
of rule (1), so only rule (2) is used.

Prolog Program:

```
(1) SQDIST(p2(X1, Y1), p2(X2, Y2), D)
    :- XD is X1-X2,
       YD is Y1-Y2,
       D is XD*XD + YD*YD.
```

```
(2) SQDIST(p3(X1, Y1, Z1), p3(X2, Y2, Z2), D)
    :- XD is X1-X2,
       YD is Y1-Y2,
       ZD is Z1-Z2,
       D is XD*XD + YD*YD + ZD*ZD.
```

Query: sqdist(p2(0,0), p3(1,1,1), D).

Note: this query does not unify with any rule,  
so Prolog simply returns no, i.e., no answers  
for D.

## Returning Function Terms as Answers

*e.g.*, given a point, `p2(X,Y)`, return a new point with double the coordinates. *e.g.*,

Query: `double(p2(3,4),P)`

Answer:`P = p2(6,8)`.

Prolog Program:

```
double(p2(X1,Y1), p2(X2,Y2))
:- X2 is 2*X1,
   Y2 is 2*Y1.
```

In Plain English: if `X2 = 2*X1` and `Y2 = 2*Y1`, then the double of `p2(X1,Y1)` is `p2(X2,Y2)`.

An equivalent program using "=":

```
double(p2(X1,Y1), P)
:- X2 is 2*X1, Y2 is 2*Y1,
   P = p2(X2,Y2).
```

Here, "=" is being used to assign a value to variable `P`. Try to avoid this!!!! It reflects procedural thinking.

## Sample Execution

Prolog Program:

```
double(p2(X1, Y1), p2(X2, Y2))  
:- X2 is 2*X1,  
   Y2 is 2*Y1.
```

Query: double(p2(3,4),P)

The query unifies with the head of the rule,  
where the mgu is

$$\{X1\backslash 3, Y1\backslash 4, P\backslash p2(X2, Y2)\}$$

The body of the rule then evaluates:

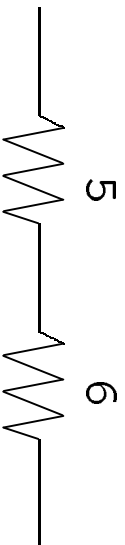
```
X2 is 2*X1,   i.e., 6  
Y2 is 2*Y1,   i.e., 8
```

The mgu becomes  $\{X1\backslash 3, Y1\backslash 4, P\backslash p2(6,8)\}$ .

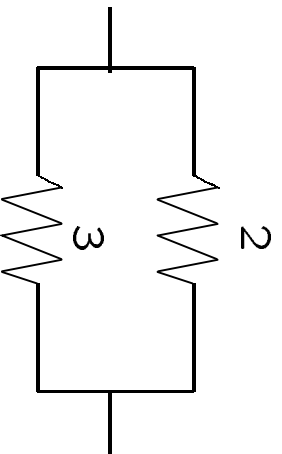
So, the answer is  $P = p2(6,8)$ .

# Recursion with Function Symbols

Example: Electrical circuits

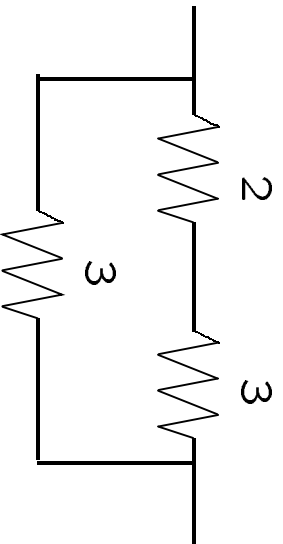


- Two resistors in series, with resistances  $R_1$  and  $R_2$ , respectively.
- Total resistance of the circuit is  $5 + 6 = 11$ .
- Can represent the circuit as a function term: series(5,6).

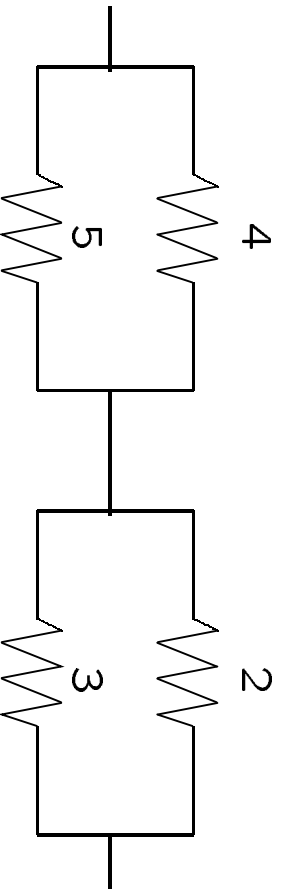


- Two resistors in parallel.
- Total resistance of the circuit is  $\frac{2 \times 3}{2+3} = 1.2$
- Represent the circuit as a function term: par(2,3).

## More Complex Circuits



`par(3, series(2,3))`



`series(par(4,5), par(2,3))`

## **Problem:**

Write a Prolog program that computes the total resistance of any circuit.

For example,

Query: resistance(series(1,2), R)

Answer: R = 1+2 = 3

Query: resistance(par(2,3), R)

Answer: R = (2\*3)/(2+3) = 6/5 = 1.2

Query: resistance(series(3,par(2,3))), R)

Answer: R = 3 + 1.2 = 4.2

Query: resistance(3, R)

Answer: R = 3

## Solution

- (1) `resistance(R,R) :- number(R).`
- (2) `resistance(series(C1,C2), R)`  
`:- resistance(C1, R1),`  
`resistance(C2, R2),`  
`R is R1+R2.`
- (3) `resistance(par(C1,C2), R)`  
`:- resistance(C1,R1),`  
`resistance(C2,R2),`  
`R is (R1*R2)/(R1+R2).`

Sample Query:

```
resistance(series(3,par(6,3)), TR)
```

*i.e.*, compute the total resistance,  $T_R$ , of the following circuit:

