```
add(X, n0, X).
add(X, Y, Z) :- inc(X, X1), dec(Y, Y1), add(X1, Y1, Z).

?- add(n1, n2, X).
```

This matches head of rule with $X_1 = n1$, $Y_1 = n2$, $Z_1 = X$

The RHS is then inc(n1, X1), dec(n2, Y1), add(X1, Y1, X)

These are taken L-R as new queries. The first produces X1 = n2. dec(n2, Y1) produces (eventually) Y1 = n1. The last query is then add(n2, n1, X)

This again eventually becomes add(n3, n0, X) which now matches the fact with $X_2 = n3$, $X_2 = X$

The result is X = n3.

A query in Prolog is called a <u>goal</u>.

A better way to draw the succession of calls is in a goal tree.

$?-\ add(n1, n2, X).$

$\left| \begin{array}{l} X_1 = n1 \\ Y_1 = n2 \\ Z_1 = X \end{array} \right.$

$inc(n1, X1_1), dec(n2, Y1_1), add(X1_1, Y1_1, X)$

$| X1_1 = n2$

$dec(n2, Y1_1), add(n2, Y1_1, X)$

$\left| \begin{array}{l} X_2 = n2 \\ Y_2 = Y1_1 \end{array} \right.$

$inc(Y1_1, n2), add(n2, Y1_1, X)$

$| Y1_1 = n1$

$add(n2, n1, X)$

$\left| \begin{array}{l} X_3 = n2 \\ Y_3 = n1 \\ Z_3 = X \end{array} \right.$

$inc(n2, X1_2), dec(n1, Y1_2), add(X1_2, Y1_2, X)$

$|$

$$\left| X1_2 = n3 \right.$$

$dec(n1, Y1_2), add(n3, Y1_2, X)$

$$\left| \begin{array}{l} X_3 = n1 \\ Y_3 = Y1_2 \end{array} \right.$$

$inc(Y1_2, n1), add(n3, Y1_2, X)$

$$\left| Y1_2 = n\emptyset \right.$$

$add(n3, n\emptyset, X)$

$$\left| \begin{array}{l} X_4 = n3 \\ X_4 = X \end{array} \right.$$

$X = n3$

This is a program trace. Each goal is either satisfied (or not) and if so, it is replaced by the goal sequence on the RHS of the matching rule.

?- add(n3, S, n5).

$X_1 = n3$
$Y_1 = S$
$Z_1 = n5$

inc(n3, $X1_1$), dec(S, $Y1_1$), add($X1_1$, $Y1_1$, n5)

$X1_1 = n4$

dec(S, $Y1_1$), add(n4, $Y1_1$, n5)

$X_2 = S$
$Y_2 = Y1_1$

choice point → inc($Y1_1$, S), add(n4, $Y1_1$, n5)

$Y1_1 = n0$
$S = n1$

add(n4, n0, n5)

fail

$Y1_1 = n1$
$S = n2$ ← answer

add(n4, n1, n5)

yes, S = n2

We can force Prolog to backtrack.

?- inc(X,Y).

X = n0, Y = n1 ; 'or?'

X = n1, Y = n2 ;

;

;

X = n9, Y = n10 ;

no

This is called resatisfied

?- add(n4, n0, n5).

$\left|\begin{array}{l} X_1 = n4 \\ Y_1 = n0 \\ Z_1 = n5 \end{array}\right.$

inc(n4, $X1_1$), dec(n0, $Y1_1$), add($X1_1$, $Y1_1$, n5)

$\left|\ X1_1 = n5\right.$

dec(n0, $Y1_1$), add(n5, $Y1_1$, n5)

$\left|\begin{array}{l} X_2 = n0 \\ Y_2 = Y1_1 \end{array}\right.$

inc($Y1_1$, n0), add(n5, $Y1_1$, n5)

|

fail, no

```
?- add(X,Y,n3).
   X=n3, Y=nø;
   X=nø, Y=n3;
   X=n1, Y=n2;
   X=n2, Y=n1;
   no

?- add(X,n3,Y).
   X=nø, Y=n3;
   X=n1, Y=n4;
   X=n2, Y=n5;
   X=n3, Y=n6;
   X=n4, Y=n7;
   X=n5, Y=n8;
   X=n6, Y=n9;
   X=n7, Y=n1ø;
   no <—— closed world assumption
```

Procedural semantics of Prolog.

- facts/rules
- unification of terms
- failure and backtracking to a choice point
- and/or [and is , or is ;]
- replacement of a goal with goals from the body of a rule.

Now to multiplication:

```
mul(X,n1,X).
mul(X,Y,Z) :- dec(Y,Y1), mul(X,Y1,M1),
              add(X,M1,Z).
```

?- mul(n3, n2, X).

$X_1 = n3$
$Y_1 = n2$
$Z_1 = X$

dec(n2, Y1₁), mul(n3, Y1₁, M1₁), add(n3, M1₁, X)

$X_2 = n2$
$Y_2 = Y1_1$

inc(Y1₁, n2), mul(n3, Y1₁, M1₁), add(n3, M1₁, X)

$Y1_1 = n1$

mul(n3, n1, M1₁), add(n3, M1₁, X)

$X_3 = n3$
$X_3 = M1_1$

add(n3, n3, X)

⋮

X = n6

Handling zero case:

mul(X, n0, n0).
mul(n0, X, n0).

?- mul(n2, X, n6).

X = n3

?- mul(X, n3, n6).

X = n2

?- mul(X, Y, n6).

X = n1, Y = n6 ;

X = n2, Y = n3 ;

X = n3, Y = n2 ;

X = n6, Y = n1 ;

no

Original problem

$$2y = x + 5$$

$$y + 2 = 2x$$

?- mul(Y, n2, Y2), add(X, n5, Y2),
mul(X, n2, X2), add(Y, n2, X2).

X = n3, Y = n4, Y2 = n8, X2 = n6