Datalog
Atomic names only
  e.g. art, 234, “Hello!”

Prolog
Atomic names and compound names
  e.g. art, 234, “Hello!”
  e.g. list(a, b, c), set(2, 4, 6), tree(tree(a,b),tree(b,c))
Syntax
Constants are strings of lower case letters, digits, underscores, and periods or strings of arbitrary ascii characters within double quotes. Main difference is that we include constructors as well as symbols and predicates.

art  cs151  3.14159
    pair    triple
    person  hasoffice

Variables are strings of letters, digits, underscores beginning with an upper case letter.

   X    Y23    Somebody
A compound term is an expression formed from an $n$-ary constructor and $n$ terms enclosed in parentheses and separated by commas.

\[
\begin{align*}
\text{pair}(a,b) \\
\text{triple}(a,b,c)
\end{align*}
\]

Compound terms can be nested.

\[
\text{pair}(\text{pair}(a,b),\text{pair}(b,c))
\]
Terms

Symbols
  art
  bob

Variables
  X
  Y23

Compound Terms
  pair(art,bob)
  pair(X,Y23)
  pair(pair(art,bob),pair(X,Y23))
Atoms

\[ p(a, \text{pair}(b, c)) \]
\[ p(a, \text{pair}(b, X)) \]
\[ p(\text{pair}(Y, Z), c) \]

Negations

\[ \neg p(a, \text{pair}(b, c)) \]

Literals (atoms or negations of atoms)

\[ p(a, \text{pair}(b, X)) \]
\[ \neg p(\text{pair}(Y, Z), c) \]

An atom is a \textit{positive literal}.
A negations is a \textit{negative literal}.
Rules
   art
   bob

Datasets
   p(a,pair(b,c))
   p(a,pair(b,X))
   p(pair(Y,Z),c)

Rulesets
   g(pair(Y,X),Z) :- p(X,pair(Y,Z))
Semantics
The **Herbrand universe** for a basic logic program is the set of all *ground terms* that can be formed from the symbols and constructors in the vocabulary.

The **Herbrand base** for a basic logic program is the set of all *ground atoms* that can be formed from the vocabulary.

An **interpretation** of a basic logic program is an arbitrary *subset* of the program’s Herbrand base.

A **model** of a basic logic program is an extension that satisfies the program (definition to follow).
An extension $D$ satisfies a rule instance $p : - p_1 \& \ldots \& p_n$ if and only if $p \in D$ whenever $D$ satisfies $p_1$ and … and $p_n$. $D$ satisfies a ground atom $p$ if and only if $p \in D$. $D$ satisfies a ground negation $\neg p$ if and only if $p \notin D$.

An extension $D$ satisfies a basic logic program if and only if (1) all of the elements of the constituent dataset are included in $D$ and (2) $D$ satisfies all instances of all of the rules in the constituent ruleset.
Basic logic programs can have multiple models.

A factoid is **logically entailed** by a basic logic program if and only if it is true in *every* model of the program, i.e. the set of conclusions is the intersection of all models of the program.

Solution 1 - the *intersection* of all models.

Solution 2 - consider all *minimal* models.

When basic logic program is stratified, there is a unique minimal model and it is the intersection of all models.
Without Constructors:
Object Constants: a, b
Unary Predicate: \( p \)
Binary Predicate: \( q \)

\[ \{p(a), p(b), q(a,a), q(a,b), q(b,a), q(b,b)\} \]

With Constructors:
Object Constants: a, b
Unary Constructor: \( f \)
Unary Predicate: \( p \)

\[ \{p(a), p(f(a)), p(f(f(a))), \ldots\} \]
**Difference:** An important difference between Datalog and Prolog is that, in Prolog, we have interpretations that are infinitely large and there are infinitely many of them.

**Bad News:** Straightforward bottom-up execution is not possible in all cases.

**Good News:** Top-down execution works to compute finitely many answers, and that is enough for many applications.
Examples
Facts and Rules

\[
\begin{align*}
  p(a,1) \\
p(f(a),2) \\
p(f(f(a)),3) \\
q(X,g(Y)) & : \ p(f(X),Y)
\end{align*}
\]

Trace

```
Call: q(a,Z)                      Z=g(Y) \\
| Call: p(f(a),Y)     X=a \\
| Exit: p(f(a),2)     X=a, Y=2 \\
Exit: q(a,g(2))        Z=g(2)
```
Decreasing Recursion

Facts and Rules

\[ p(a) \]
\[ p(b) \]
\[ p(f(f(X))) :- p(X) \]

Trace

Call: \( p(a) \)
Exit: \( p(a) \)

Call: \( p(f(a)) \)
Fail: \( p(f(a)) \)

Call: \( p(f(f(a))) \)
Call: \( p(a) \)
Exit: \( p(a) \)
Exit: \( p(f(f(a))) \)

Fail: \( p(f(f(a))) \)

Exit: \( p(f(f(a))) \)

Infinitely many answers!
Facts and Rules

\[ p(a) \]
\[ p(b) \]
\[ p(X) :- p(f(f(X))) \]  \hspace{1cm} \textit{One answer; infinite recursion.} \\

Trace

Call: \( p(a) \)
Exit: \( p(a) \)

Call: \( p(b) \)
  | Call: \( p(f(f(b))) \)
  |   Call: \( p(f(f(f(f(b))))) \)
  |     Call: \( p(f(f(f(f(f(f(b))))))) \)
  |       | \( ... \)
Example - Arithmetic
In *Natural Arithmetic*, we are concerned with all of the natural numbers, not just a finite subset, and functions do not wrap around as in Modular Arithmetic.

\[
\begin{array}{cccc}
0+0=0 & 1+0=1 & 2+0=2 & 3+0=3 \\
0+1=1 & 1+1=2 & 2+1=3 & 3+1=5 \\
0+2=2 & 1+2=3 & 2+2=4 & 3+2=6 \\
\vdots & \vdots & \vdots & \vdots \\
0\times0=0 & 1\times0=0 & 2\times0=0 & 3\times0=0 \\
0\times1=0 & 1\times1=1 & 2\times1=2 & 3\times1=3 \\
0\times2=0 & 1\times2=2 & 2\times2=4 & 3\times2=6 \\
\vdots & \vdots & \vdots & \vdots \\
\end{array}
\]
Possible Representation

Symbol: 0, 1, 2, …

Unary Predicate:
    number - holds of natural numbers and nothing else

Ternary Predicate:
    plus - the third argument is the sum of the first two
    times - third argument is the product of the first two
Enumerating all ground data is impossible

number(0)
number(1)
number(2)
...

plus(0,0,0)
plus(0,1,1)
plus(1,0,1)
...

times(0,0,0)
times(0,1,0)
times(1,0,0)
...

Axiomatization
Symbol: 0
Unary Constructor: s
Herbrand Universe: 0, s(0), s(s(0)), …

Unary Predicate:
    number - holds of numbers and only numbers

Ternary Predicate:
    plus - the third argument is the sum of the first two
    times - third argument is the product of the first two
number(0)
number(s(X)) :- number(X)
number(0)
number(s(X)) :- number(X)

plus(0,Y,Y) :- number(Y)
plus(s(X),Y,s(Z)) :- plus(X,Y,Z)
number(0)
number(s(X)) :- number(X)

plus(0,Y,Y) :- number(Y)
plus(s(X),Y,s(Z)) :- plus(X,Y,Z)

times(0,Y,0) :- number(Y)
times(s(X),Y,Z) :- times(X,Y,W)&plus(W,Y,Z)
A *polynomial equation* is an algebraic expression composed using only addition and multiplication and exponentiation with fixed exponents.

\[ 3x^2 + 2y = 2z^3 \]

A *natural Diophantine equation* is a polynomial in which the values of variables are restricted to natural numbers.

\[ x = 2 \]
\[ y = 2 \]
\[ z = 2 \]
Diophantine equation:

\[ 3x^2 = 1 \]

Diophantine equation in Peano Arithmetic:

\[ \text{goal}(X) :- \text{times}(X,X,Y) \& \text{times}(3,Y,1) \]

We can evaluate goal to solve the equation. However, this is a brute force method. Also, there is no general way of terminating the computation for equations that do not have solutions.