True or False questions:
e.g. Is Art the parent of Bob?

Fill-in-the-blanks questions:
e.g. Art is the parent of ____?
e.g. ____ is the parent of Bob?
e.g. ____ is the parent of ____?

Compound questions:
e.g. Is Art the parent of Bob and the parent of Bud?
e.g. ____ has sons and no daughters?
Syntax
A **constant** is a string of lower case letters, digits, underscores, and periods *or* a string of ascii characters within double quotes.

Examples:

- joe, bill, cs151, 3.14159
- person, worksfor, office
- the_house_that_jack_built,
- “Mind your p’s & q’s!”

Non-examples:

- Art, p&q, the-house-that-jack-built
Symbols / object constants represent objects.
   joe, bill, harry, a23, 3.14159
   the_house_that_jack_built
   “Mind your p’s & q’s!”

Constructors / function constants represent functions.
   pair, triple, set

Predicates / relation constants represent relations.
   person, parent, prefers
The **arity** of a constructor or a predicate is the number of arguments that can be associated with the constructor or predicate in writing complex expressions in the language.

**Unary** predicate (1 argument): \(\text{person}(joe)\)

**Binary** predicate (2 arguments): \(\text{parent}(art,bob)\)

**Ternary** predicate (3 arguments): \(\text{prefers}(art,bob,bea)\)

In defining vocabulary, we sometimes notate the arity of a constructor or predicate by annotating with a slash and the arity, e.g. `male/1`, `parent/2`, and `prefers/3`. 
A **variable** is either a lone underscore or a string of letters, digits, underscores, and periods beginning with an upper case letter.

\[
\begin{array}{ccc}
  X & Y23 & \text{Somebody} & _ \\
\end{array}
\]
Symbols
  art
  bob

Variables
  X
  Y23

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

\[ p(a,b) \]
\[ p(a,X) \]
\[ p(Y,c) \]

Negations

\[ \neg p(a,b) \]

Literals (atoms or negations of atoms)

\[ p(a,Y) \]
\[ \neg p(a,Y) \]

An atom is a *positive literal*.
A negation is a *negative literal*. 
goal(a,b) :- p(a,b) & ~q(b)
goal(X,Y) :- p(X,Y) & ~q(Y)
A **query** is a non-empty, finite set of query rules.

\[
\text{goal}(X,Y) :- p(X,Y) \land q(X)
\]

\[
\text{goal}(X,Y) :- p(X,Y) \land \neg q(Y)
\]

**NB:** The IDEs for most Logic Programming systems (including Sierra) do not support queries with multiple rules.
Semantics
An **instance of a rule** is a rule in which all variables have been consistently replaced by ground terms.

**Rule**

\[
goal(X,Y) :- p(X,Y) \land \neg q(Y)
\]

**Herbrand Universe**

\{a, b\}

**Instances**

\[
\begin{align*}
goal(a,a) & :- p(a,a) \land \neg q(a) \\
goal(a,b) & :- p(a,b) \land \neg q(b) \\
goal(b,a) & :- p(b,a) \land \neg q(a) \\
goal(b,b) & :- p(b,b) \land \neg q(b)
\end{align*}
\]
The **result of applying a rule** $r$ **to a dataset** $\Delta$ (written $\nu(r,\Delta)$) is the set of all $\psi$ such that (1) $\psi$ is the head of an arbitrary instance of $r$, (2) every positive subgoal in the instance is a member of $\Delta$, and (3) no negative subgoal in the instance is a member of $\Delta$. 
Example

**Dataset**

- \( p(a,b) \)
- \( p(b,c) \)
- \( p(c,d) \)
- \( p(d,c) \)

**Result**

- \( \text{goal}(a,b) \)
- \( \text{goal}(b,c) \)

**Rule**

\[
\text{goal}(X,Y) :- p(X,Y) \land \neg p(Y,X)
\]

**Instances**

- \( \text{goal}(a,a) :- p(a,a) \land \neg p(a,a) \)
- \( \text{goal}(a,b) :- p(a,b) \land \neg p(b,a) \)
- \( \text{goal}(a,c) :- p(a,c) \land \neg p(c,a) \)
- \( \text{goal}(a,d) :- p(a,d) \land \neg p(d,a) \)
- \[ \ldots \]
- \( \text{goal}(b,c) :- p(b,c) \land \neg p(c,b) \)
- \[ \ldots \]
- \( \text{goal}(c,d) :- p(c,d) \land \neg p(d,c) \)
- \[ \ldots \]
Dataset

\begin{align*}
& p(a, b) \\
& p(b, c) \\
& p(c, d) \\
& p(d, c)
\end{align*}

Rule

\begin{align*}
goal(X, Y) & \leftarrow p(X, Y) \land \neg p(Y, X)
\end{align*}

Not Results

\begin{align*}
goal(a, b) & \quad goal(a, b) \nonumber \\
& \quad goal(b, c) \nonumber \\
& \quad goal(c, d) \nonumber \\
& \quad goal(d, c) \nonumber
\end{align*}
The **extension** of a query (written $E(\Omega,\Delta)$), is the set of all facts that can be "deduced" from a dataset $\Delta$ on the basis of the rules in the query, i.e. it is the union of $v(r_i,\Delta)$ for each $r_i$ in our query $\Omega$.

Dataset: \{p(a,b), p(b,c)\}

\begin{align*}
  \text{goal}(X) & : - p(X,Y) \\
  \text{goal}(Y) & : - p(X,Y)
\end{align*}

Result: \{goal(a), goal(b), goal(c)\}

**NB:** For our language, there one and only one extension for any dataset and and ruleset, i.e. the extension is unique.
Safety
A rule is *safe* if and only if every variable in the head appears in some positive subgoal in the body and every variable in a negative subgoal appears in a prior positive subgoal.

Safe Rule:
\[
goal(X, Z) :- \text{p}(X, Y) \& \text{q}(Y, Z) \& \neg\text{r}(X, Y)
\]

Unsafe Rule:
\[
goal(X, Z) :- \text{p}(X, Y) \& \neg\text{q}(Y, Z)
\]
Ruleset

\[
\text{goal}(X,Y,Z) :- \ p(X,Y)
\]

Dataset

\[
p(a,b)
\]

Output for s

\[
\begin{align*}
goal(a,b,a) \\
goal(a,b,b) \\
goal(a,b,c) \\
goal(a,b,d) \\
goal(a,b,e) \\
\cdots
\end{align*}
\]
Ruleset

\[
goal(X,Y) :- p(X,Y) & \neg q(Y,Z)
\]

Dataset

- \(p(a,b)\)
- \(p(a,c)\)
- \(q(c,d)\)

Output

- \(goal(a,b)\)
- \(goal(a,c)\)

English

Find all \(X\) and \(Y\) such that \(p(X,Y)\) is true and there is some \(Z\) for which \(q(Y,Z)\) is false.

versus

Find all \(X\) and \(Y\) such that \(p(X,Y)\) is true and there is no \(Z\) for which \(q(Y,Z)\) is true.
Predefined Concepts
Functions
Arithmetic Functions (e.g. plus, times, min, max, etc.)
String functions (e.g. concatenate, string matching, etc.)
Other (e.g. converting between formulas and strings, etc.)
Aggregates (e.g. sets of objects with given properties)

Relations
Equality and Inequality
Evaluable term - constant, variable, f(t₁,…tn)
f is a predefined function or user-defined function
t₁,…,tn are evaluable terms

Examples

plus(2,3)
stringappend(“abc”,”def”)
stringify(vinay)
symbolize(“vinay”)

min(plus(2,3),times(2,3))

NB: Many predefined functions are variadic, e.g. plus.
evaluate(x,v)
  x is a term
  v is the value of x

Examples
  evaluate(times(2,3),6)
  evaluate(plus(times(2,3),4),10)

goal(X,Z) :-
  height(X,H) & width(X,W) &
  evaluate(times(H,W),Z)

NB: unbound variables allowed in second argument only
Predefined Aggregates

- setofall
- countofall

Example

goal(X,N) :-
  person(X) &
  evaluate(countofall(Y,parent(X,Y)),N)
Identity
same(t₁, t₂) is true iff t₁ and t₂ are identical

Difference
distinct(t₁, t₂) is true iff t₁ and t₂ are different

NB: This is not ordinary equality (e.g. 2+2 = 4)
same(plus(2,2),4) is false
distinct(plus(2,2),4) is true

evaluate(plus(2,2),V) & same(V,4) is true
evaluate(plus(2,2),4) is true
Kinship
Kinship

Diagram:

- Art
  - Bob
    - Cal
  - Bea
    - Cam
    - Cat
    - Coe
parent(art,bob)
parent(art,bea)
parent(bob,cal)
parent(bob,cam)
parent(bea,cat)
parent(bea,coe)
Query

\[
\text{goal}(X, Z) :- \text{parent}(X, Y) \land \text{parent}(Y, Z)
\]

Dataset:
- parent(art, bob)
- parent(art, bea)
- parent(bob, cal)
- parent(bob, cam)
- parent(bea, cat)
- parent(bea, coe)

Result:
- goal(art, cal)
- goal(art, cam)
- goal(art, cat)
- goal(art, coe)
Query:

\[
\text{goal}(X) :- \text{parent}(X,Y) \\
\text{goal}(X) :- \text{parent}(Y,X)
\]

Dataset:  

- parent(art,bob)
- parent(art,bea)
- parent(bob,cal)
- parent(bob,cam)
- parent(bea,cam)
- parent(bea,coe)

Result:

- goal(art)
- goal(bob)
- goal(bea)
- goal(cal)
- goal(cam)
- goal(coe)
Query

\[ \text{goal}(Y,Z) \leftarrow \text{parent}(X,Y) \land \text{parent}(X,Z) \land \text{distinct}(Y,Z) \]

Dataset:                                     Result:

\[ \text{parent}(\text{art}, \text{bob}) \]
\[ \text{parent}(\text{art}, \text{bea}) \]
\[ \text{parent}(\text{bob}, \text{cal}) \]
\[ \text{parent}(\text{bob}, \text{cam}) \]
\[ \text{parent}(\text{bea}, \text{cat}) \]
\[ \text{parent}(\text{bea}, \text{coe}) \]

\[ \text{goal}(\text{bob}, \text{bea}) \]
\[ \text{goal}(\text{bea}, \text{bob}) \]
\[ \text{goal}(\text{cal}, \text{cam}) \]
\[ \text{goal}(\text{cam}, \text{cal}) \]
\[ \text{goal}(\text{cat}, \text{coe}) \]
\[ \text{goal}(\text{coe}, \text{cat}) \]
Dataset:

parent(art,bob)
parent(art,bea)
parent(art,ben)
parent(bob,eli)

Query: find every person with at least one child

goal(X) :- parent(X,Y)
Dataset:

parent(art,bob)
parent(art,bea)
parent(art,ben)
parent(bob,eli)

Query: find every person with at least two children

\[ \text{goal}(X) \; :\; - \\
\text{parent}(X,Y) \; \& \; \text{parent}(X,Z) \; \& \; \text{distinct}(Y,Z) \]
Dataset:

\[
\begin{align*}
\text{parent(art,bob)} \\
\text{parent(art,bea)} \\
\text{parent(art,ben)} \\
\text{parent(bob,eli)}
\end{align*}
\]

Query: find every person with at least three children

\[
\text{goal}(X) :- \\
\text{parent}(X,Y) \& \text{parent}(X,Z) \& \text{parent}(X,W) \\
\text{mutex}(Y,Z,W)
\]
Dataset:

parent(art,bob)
parent(art,bea)
parent(art,ben)
parent(bob,eli)

Query: find every person with exactly three children

goal(X) :-
    parent(X,Y) &
    evaluate(countofall(Z,parent(X,Z)),3)
Map Coloring
Dataset

hue(red)
hue(green)
hue(blue)
hue(purple)
goal(C1,C2,C3,C4,C5,C6) :-
    hue(C1) & hue(C2) & hue(C3) & hue(C4) & hue(C5) & hue(C6) &
    distinct(C1,C2) & distinct(C1,C3) & distinct(C1,C5) &
    distinct(C1,C6) & distinct(C2,C3) & distinct(C2,C4) &
    distinct(C2,C5) & distinct(C2,C6) & distinct(C3,C4) &
    distinct(C3,C6) & distinct(C5,C6)