**Experiment Number: 04**

**Problem Statement: Prolog- Tiny Expert System.**

**NAME: Priti Patil ROLLNO: 80**

# **CLASS: TY(IT) BATCH: 3**

**DATE OF PERFORMANCE: 15/ 10 / 2022**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Code-**

animal(dog) :- is\_true("has fur"), is\_true("says woof").

animal(cat) :- is\_true("has fur"), is\_true("says meow").

animal(duck) :- is\_true("has feathers"), is\_true("says quack").

is\_true(Q) :-

format("~s?\n", [Q]),

read(yes).

animals([animal(dog, [is\_true("has fur"), is\_true("says woof")]),

animal(cat, [is\_true("has fur"), is\_true("says meow")]),

animal(duck, [is\_true("has feathers"), is\_true("says quack")])]).

animal(A) :-

animals(As),

member(animal(A,Cs), As),

maplist(call, Cs).

animal(A) :-

animals(Animals),

Known0 = [],

phrase(any\_animal(Animals, A), [Known0], \_).

any\_animal([Animal|Animals], A) -->

any\_animal\_(Animal, Animals, A).

any\_animal\_(animal(A0, []), Animals, A) -->

( { A0 = A }

; any\_animal(Animals, A)

).

any\_animal\_(animal(A0, [C|Cs]), Animals, A) -->

state0\_state(Known0, Known),

{ condition\_truth(C, T, Known0, Known) },

next\_animal(T, animal(A0,Cs), Animals, A).

next\_animal(yes, Animal, Animals, A) --> any\_animal([Animal|Animals], A).

next\_animal(no, \_, Animals, A) --> any\_animal(Animals, A).

state0\_state(S0, S), [S] --> [S0].

condition\_truth(is\_true(Q), Answer, Known0, Known) :-

if\_(known\_(Q,Answer,Known0),

Known0 = Known,

( format("~s?\n", [Q]),

read(Answer),

Known = [known(Q,Answer)|Known0])).

known\_(What, Answer, Known, Truth) :-

if\_(memberd\_t(known(What,yes), Known),

( Answer = yes, Truth = true ),

if\_(memberd\_t(known(What,no), Known),

( Answer = no, Truth = true),

Truth = false)).

tree(if\_then\_else("has fur",

if\_then\_else("says woof",

animal(dog),

if\_then\_else("says meow",

animal(cat),

false)),

if\_then\_else("has feathers",

if\_then\_else("says quack",

animal(duck),

false),

false))).

animal(A) :-

tree(T),

tree\_animal(T, A).

tree\_animal(animal(A), A).

tree\_animal(if\_then\_else(Cond,Then,Else), A) :-

( is\_true(Cond) ->

tree\_animal(Then, A)

; tree\_animal(Else, A)

).

**Output-**

