**Lab Sheet # 2 Class: BSCS**

**OBJECT: TEST THE EVALUATION OF GOALS, USING UNIFICATION**

**AND BACKTRACKING**

**THEORY:**

*Unification* is the process of matching two predicates and assigning free variables to make the predicates identical. This mechanism is necessary so that Prolog can identify which clauses to call and bind values to variables. These are the major points about matching (*Unification*) presented in this session.

When a new call is made, a search for a match to that call also begins at the top of the program

When a call has found a successful match, the call is said to return, and the next sub goal in turn can be tried.

Once a variable has been bound in a clause, the only way to free that binding is through *backtracking*.

*Backtracking* is the mechanism that instructs Prolog where to go to look for solutions to the program. This process gives Prolog the ability to search through all known facts and rules for a solution. There are four basic principles of backtracking given in this session:

Sub goals must be satisfied in order, from top to bottom.

Predicate clauses are tested in the order they appear in the program, from top to bottom.

When a sub goal matches the head of a rule, the body of that rule must be satisfied next. The body of the rule then constitutes a new set of sub goals to be satisfied.

A goal has been satisfied when a matching fact is found for each of the extremities (leave) of the goal tree.

A call that can produce multiple solutions is *non-deterministic*,while a call that can produce one and only one solution *is deterministic*.

**Example 1:**

factorial(0,1).

factorial(X,Y) :-

X1 is X - 1,

factorial(X1,Z),

Y is Z\*X,!.

**Recursion Breakup for ‘factorial(4,What).’**

0

Factorial(0,1)

1

2

3

X1 is X – 1, (X1 is X – 1, (X1 is X – 1, (X1 is X – 1, (Z=1), Y is X \* Z), Y is X \* Z), Y is X \* Z), Y is X \* Z)

**Decomposition**:

X1 is 3, (X1 is 2, (X1 is 1, (X1 is 0, (Z=1), Y ==1 is X==1 \* Z==1), Y==2 is X==2 \* Z==1), Y==6 is X==3 \* Z==2), Y==24 is X==4 \* Z==6)

X is decreasing step by step

Each Y has the value of Z form previous (inner) step

**Example 2:**

talks\_about (A, B):- knows(A,B).

talks\_about(P,R):- knows(P,Q),

talks\_about(Q,R).

**Example 3:**

**Facts:**

parent(john,paul). /\* John is Paul's parent \*/

parent(paul,tom). /\* Paul is Tom's parent \*/

parent(tom,mary)./\* Tom is Mary's parent \*/

**Rules:**

ancestor(X,Y) :- parent(X,Y). /\* If X is a parent of Y, then X is an ancestor of Y \*/

ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y). /\* if X is parent of Z and Z is ancestor of Y, then X is ancestor of Y \*/

**Query:**

Execution scheme for the query “ancestor(john,tom).”

CALL ancestor(john,tom).

CALL parent(john,tom).

FAIL parent(john,tom).

CALL parent(john,Z).

TRY Z=paul (Unification)

CALL ancestor(paul,tom).

CALL parent(paul,tom). (Recursion)

SUCCEEDS parent(paul,tom).

SUCCEEDS ancestor(paul,tom).

SUCCEEDS with Z=paul

SUCCEEDS ancestor(john,tom).

**GOAL:**

**Name of few countries with population is given, define a general rule that Print the name of countries having population greater than 10 million.**

country(england, 3e7).

country(france, 2.3e7).

country(germany,1.6e7).

country(denmark,2.4e6).

country(canada,7.3e6).