PROBLEM STATEMENT :

Implementation of Unification algorithm

OBJECTIVE :

- To develop problem solving abilities for gamifications.
- To apply algorithmic strategies while solving problems
- To develop time and space efficient algorithms

THEORY :

Lified inference rules require finding substitutions that make different logical expressions look identical. This process is called unification and is a key component of all first-order inference algorithms. The UNIFY algorithm takes two sentences and returns a unifier for them if one exists.

A **unification algorithm** should compute for a given problem a complete, and minimal substitution set, that is, a set covering all its solutions, and containing no redundant members.

UNIFY (p q)= 0 where SunsT(0,p)= SuBsT(0, q) .

Let us look at some examples of how UNIFY should behave. Suppose we have a query Askilars(Knows (John , x)): whom does John know? Answers to this query can be found by finding all sentences in the knowledge base that unify with Knows(John, x). Here are the results of unification with four different sentences that might be in the knowledge base:

UNIFY (Knows (John x). Knows (John, Jane)) = {x / Jane}
UNIFY (Knows (John , x). Knows (y, Bill)) = {x / Bill, y / John}
UNIFY (Knows(John , x), Knows (y , Mother (y))) = {y/John ,x / mother(John)}
UNIFY (Knows(John, x). Knows (x , Elizabeth)) = fail.

ALGORITHM :

```
Procedure Unify(t1,t2)
    Inputs
        t1,t2: atoms Output
        most general unifier of t1 and t2 if it exists or ⊥ otherwise
    Local
        E: a set of equality statements
        S: substitution
    E ←{t1=t2}
    S={}
    while (E≠{})
        select and remove x=y from E
        if (y is not identical to x) then
        f (x is a variable) then
        replace x with y everywhere in E and S
            S←{x/y}∪S
        else if (y is a variable) then
            replace y with x everywhere in E and S
            S←{y/x}∪S
        else if (x is f(x1,...,xn) and y is f(y1,...,yn)) then
```

```
        E←E∪{x1=y1,...,xn=yn}
    else
        return ⊥
    return S
```

INPUT :

Values of Expression E1 and E2 to be unified .

EXPECTED OUTPUT :

Uninfication output or Cannot apply unification.

MATHEMATICAL MODEL :

Let S be the solution perspective .

S={s, e, i, o, f, DD, NDD, success, failure}

s = { Initial Expressions E1 and E2 }

I = Input of the system → { I1, I2 }
where I1 = { Element 1 }
I2 = { Element 2 }

o = Output of the system → { O1 , O2 }
where O1 = { Unification output }
O2 = { Cannot apply unification }

f = Functions used → { f1}
where f1 = { Unification algorithm }

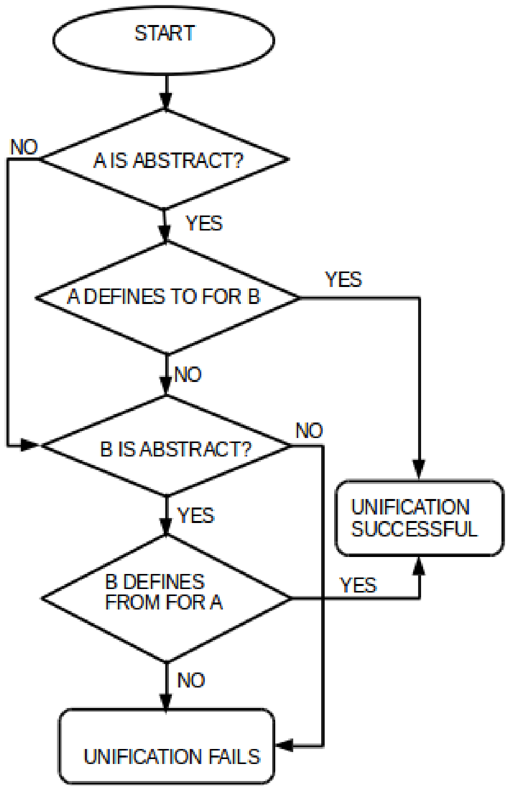
DD - Deterministic data → { }

NDD - Non deterministic data → { Element E1, E2 and Output }

Success - Desired outcome generated → {Unification result }

Failure - Desired outcome not generated or forced exit due to system error.

FLOWCHART :



TEST CASES :

TEST CASE	**INPUT**	**EXPECTED OUTPUT**	**OUTPUT ACHIEVED**	**REMARKS**
1.	E1 f(x,y) E2 g(x,y)	CANNOT APPLY UNIFICATION	PREDICATES NOT MATCHING	Correct
2.	E1 knows(x,y) E2 knows(knows(x), y)	CANNOT APPLY UNIFICATION	CANNOT APPLY UNIFICATION	Correct
3.	E1 knows(john,x) E2 knows(john,jane)	X / jane	jane / x	Correct
4.	E1 knows(john,x) E2 knows(y,bill)	y / john bill / x	y / john bill / x	Correct
5.	E1 knows(john,x) knows(y,mother(y))	y / john mother(john) / x	y / john mother(john) / x	Correc

SPACE AND TIME COMPLEXITIES :

The worst-time complexity of the unification algorithm is EXPTIME. Let N = N(D). There are at most O ((2^ N)^2) cases. Hence complexity of unification is O (2^N) ∈ EXPTIME.

CONCLUSION :

Hence we have successfully implemented the unification algorithm.

OUTCOMES ACHIEVED :

COURSE OUTCOME	**ACHIEVED(√)**
Problem solving abilities for smart devices.	
Problem solving abilities for gamifications.	√
Problem solving abilities of pervasiveness,embedded security and NLP.	
To solve problems for multicore or distributed,concurrent/Parallel environments	