Name : Tejas Redkar
PRN: 1032210937
Panel-C
Roll No : PC-44

AIES Lab Assignment - 4

* Aim - To implement unification algorithm

* Objective - To study and implement unification algorithm

* Theory -

1) Unification Algorithm :
   - It is a computational method used in symbolic reasoning & AI.
   - It is used for finding a common substitute for variables in logical expressions.
   - This algorithm plays a crucial role in various AI applications such as NLP, automated theorem proving, etc.
   - Conditions for Unification
     1) Predicate symbols must be same
     2) Number of arguments be same for both literals.
     3) Unification fails if two similar variables appear in same expression.

2) Resolution as Proof Procedure:

- A technique in automated theorem proving
- Assumes negation of the statement to be proved & attempt to derive a contradiction.
- Uses the ~~contradiction~~ resolution rule to combine clauses, aiming to prove the original statement true.

* Input - Two literals L1 & L2
* Output - A set of substitutions
* Algorithm - Unification algorithm
* FAQ's

Q1) Why resolution is required?

Ans
- Resolution is a fundamental technique is automated theorem proving, which is crucial in various fields of computer science, AI & formal logic. It enables the automatic derivation of new logical conclusions from a set of existing premises.

- Resolution is a complete inference rule, meaning that if there is a valid logical deduction to be made, resolution will eventually find it.

Q2) What are the prerequisites for applying unification algorithm?

Ans 1) Logical statements - You need a set of logical statements or predicates, typically represented in first-order logic.

2) Variables & constants - These statements should contain variables & constants, which are symbols representing variables

3) Unifiable tables - The literals should follow the conditions for unification.

Q3) What are the applications of unification algorithm

Ans Following are various applications of unification Algorithm -

- Automated theorem proving in AI
- Natural language processing for sentence parsing
- Type Inference in programming languages.
- Symbolic mathematics for simplifying expressions
- Knowledge representation in expert systems.

Anose
1/12/23

```python
def unify(e1, e2, theta={}):
    if theta is None:
        return None
    elif e1 == e2:
        return theta
    elif isinstance(e1, str):
        return unify_var(e1, e2, theta)
    elif isinstance(e2, str):
        return unify_var(e2, e1, theta)
    elif isinstance(e1, list) and isinstance(e2, list):
        if len(e1) != len(e2):
            return None
        else:
            for i in range(len(e1)):
                theta = unify(e1[i], e2[i], theta)
                if theta is None:
                    return None
            return theta
    else:
        return None

def unify_var(var, x, theta):
    if var in theta:
        return unify(theta[var], x, theta)
    elif x in theta:
        return unify(var, theta[x], theta)
    elif occurs_check(var, x, theta):
        return None
    else:
        theta[var] = x
        return theta

def occurs_check(var, x, theta):
    if var == x:
        return True
    elif isinstance(x, str) and x in theta:
        return occurs_check(var, theta[x], theta)
    elif isinstance(x, list):
        for e in x:
            if occurs_check(var, e, theta):
                return True
        return False

e1 = ["likes", "Parimal", "Kolhe","p"]
e2 = ["likes", "x", "y","q"]
theta = unify(e1, e2)
print(theta)
```

```
{'Parimal': 'x', 'Kolhe': 'y', 'p': 'q'}
```