IMPLEMENTATION OF UNIFICATION AND RESOLUTION ALGORITHM

SOURCE CODE:

```
def unify(var1, var2, subst):
    if subst is None:
        return None
    elif var1 == var2:
        return subst
    elif isinstance(var1, str) and var1.islower():
        return unify_var(var1, var2, subst)
    elif isinstance(var2, str) and var2.islower():
        return unify_var(var2, var1, subst)
    elif isinstance(var1, list) and isinstance(var2, list) and len(var1) == len(var2):
        return unify(var1[1:], var2[1:], unify(var1[0], var2[0], subst))
    else:
        return None

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

```
def occurs_check(var, x, subst):
    if var == x:
        return True
    elif isinstance(x, list):
        return any(occurs_check(var, xi, subst) for xi in x)
    elif x in subst:
        return occurs_check(var, subst[x], subst)
    return False

def resolution(kb, query):
    clauses = kb + [negate(query)]
    while True:
        new_clauses = []
        for i, ci in enumerate(clauses):
            for j, cj in enumerate(clauses):
                if i >= j:
                    continue
                resolvents = resolve(ci, cj)
                if [] in resolvents:
                    return True  # Empty clause = contradiction found
                for res in resolvents:
                    if res not in new_clauses:
                        new_clauses.append(res)
        if all(new in clauses for new in new_clauses):
            return False  # No new clauses = cannot derive contradiction
        clauses.extend(new_clauses)

def negate(literal):
    if isinstance(literal, list):
        return [negate(lit) for lit in literal]
    if literal.startswith("~"):
```

```
        return literal[1:]
    else:
        return f"~{literal}"

def resolve(clause1, clause2):
    resolvents = []
    for lit1 in clause1:
        for lit2 in clause2:
            if lit1 == negate(lit2):
                new_clause = list(set(clause1 + clause2))
                new_clause.remove(lit1)
                new_clause.remove(lit2)
                if new_clause not in resolvents:
                    resolvents.append(new_clause)
    return resolvents

# --- Main Execution ---

if __name__ == "__main__":

    knowledge_base = [
        ["~P", "Q"],
        ["P"],
        ["~Q", "R"],
        ["~R"]
    ]
    query = ["R"]  # Representing the query as a list

    print("Knowledge Base:", knowledge_base)
    print("Query:", query)
    result = resolution(knowledge_base, query)
```

```
    if result:
        print("The query is satisfiable.")
    else:
        print("The query is not satisfiable.")
```

OUTPUT:

```
Knowledge Base: [['~P', 'Q'], ['P'], ['~Q', 'R'], ['~R']]
Query: ['R']
The query is not satisfiable.
```