Computer Science and Engineering
UG SEM~5

Rutazeet Ritik Rout
1BM18CS151

1/11/20

Course :- Artificial Intelligence
Course code :- 20CS5PCAIP

```
def getAttributes(string):
    expr = '\([^)]+\)'
    matches = re.findall(expr, string)
    return [m for m in str(matches) if m.isalpha()]

def getPredicates(string):
    expr = '[a-z~]+\([A-Za-z,]+\)'
    return re.findall(expr, string)

def DeMorgan(sentence):
    string = ''.join(list(sentence).copy())
    string = string.replace('~~', '')
    flag = '[' in string
    string = string.replace('~[', '')
    string = string.strip(']')
    for predicate in getPredicates(string):
        string = string.strip(']')
    for predicate in getPredicates(string):
        string = string.replace(predicate, f'~{predicate}')
    s = list(string)
    for i, c in enumerate(string):
        if c == '|':
            s[i] = '&'
        elif c == '&':
            s[i] = '|'
    string = ''.join(s)
    string = string.replace('~~', '')
    return f'[{string}]' if flag else string
```

Rutazeet

Scanned by TapScanner

(2)

```
def Skolemization(sentence):
    SKOLEM_CONSTANTS = [f'{chr(c)}' for c in
                        range(ord('A'), ord('Z')+1)]
    statement = ''.join(list(sentence).copy())
    matches = re.findall('[∀∃.', statement)
    for match in matches[::-1]:
        Statement = Statement.replace(match,'')
        Statements = re.findall('\[\[[^]]+\]]',
                                statement)
        for s in statement
            statement = statement.replace(s, s[1:-1])
        for predicate in getPredicates(statement):
            attribute = getAttribute(predicate)
            if ''.join(attribute).islower():
                statement = statement.replace(match[1],
                            SKOLEM_CONSTANTS.pop(0))
        else:
            aV = [a for a in attributes if not
                  a.islower()][0]
            statement = statement.replace(aV,
                        f'{SKOLEM_CONSTATS.pop(0)}
                        ({match[1]})').
    return statement

import re
```

Scanned by TapScanner

(3)

```
def fol_to_cnf(fol):
    statement = fol.replace("<=>", "_")
    while '_' in statement:
        i = statement.index('_')
        new_statement = '[' + statement[:i] + '=>' +
            statement[i+1:] + ']&[' + statement[i+1:]
            + '=>' + statement[:i] + ']'
        Statement = new_statement
    statement = statement.replace("=>", "-")
    expr = '\[([^\]]+)\]'
    statement = re.findall(expr, statement)
    for i,s in enumerate(statement):
        if '[' in s and ']' not in s:
            statement[i] += ']'
    for s in statements:
        statement = statement.replace(s, fol_to_cnf(s))
    while '-' in statement:
        i = statement.index('-')
        br = statement.index('[') if '[' in
                statement else 0
        new_statement = '~' + statement[br:i] + '|' +
                        statement[i+1:]
        statement = statement[:br] + new_statement
            if br > 0 else new_statement
    while '~∀' in statement:
        i = statement.index('~∀')
        statement = list(statement)
        statement[i] = statement[i+1] + statement[i+2]
            = '∃', statement[i+2], '~'
        statement = ''.join(statements).
```

Retrieved.

Scanned by TapScanner

(4)

```
while '~∃' in statement:
        i = statement. index ('~∃')
        s = list( statement)
        s[i],s[i+1], s[i+2]='∀', s[i+2],'~'
        statement = ''.join(s).
statement = statement.replace('~[∀', '[~∀')
statement = statement.replace('~[∃', '[~∃')
expr ='(~[∀|∃].)'
statement = re. findall (expr, statement).
for s in statements:
        statement = statement.replace(s, fol_to_cnf(s))
expr = '~\[[^]]+\]'
statements = re. findall (expr, statement)
for s in statements:
    statements = statement. replace (s, DeMorgan(s))
return statement


def start()
    print ("Enter the queries"
     qs = input(). split ("next").
     print ("\n")
      for q in qs
          print (skEolemization (fol_to_cnf (q)))
          print ("\n").

start ( ).
```

Rutgers.

Scanned by TapScanner