01-01-21

AI lab Test 2

Sanjeev Kumar Singh
1BM18CS093
B3
5B

Prgm 4

Convert English Sentence into FOL, FOL to CNF and using a input

```
import re

def remove_brackets(Source, id):
    reg = '\(([^\(]*?\))'
    m = re.search(reg, Source)
    if m is None:
        return None, None
    new_Source = re.sub(reg, str(id), Source, count=1)
    return new_Source, m.group(1)

class logic_base:
    def __init__(self, input):
        self.my_stack = []
        self.Source = input
        final = input
    while 1:
        input, tmp = remove_brackets(input, len(self.my_stack))
        if input is None:
            break
        final = input
        self.my_stack.append(tmp)
    self.my_stack.append(final)

def get_result(self):
    root = self.my_stack[-1]
    m = re.match('\s*([0-9]+)\s*$', root)
    if m is not None:
        root = self.my_stack[int(m.group(1))]
    reg = '(\d+)'
while 1:
```


Sanjeev

1-1-2021

AI lab Test 2

Sangeev Kumar Singh
IBM 18CS093
B3
SB

```
m = re.search (reg, root)
if m is None:
    break
new = '(' + self.my_stack[int(m.group(1))] + ')'
root = re.sub (reg, new, root, count=1)
return root

def merge_items (self, logic):
    reg0 = '(\d+)'
    reg1 = 'no is + (\d+)'
    flag = false
for i in range (len (self.my_stack)):
    target = self.my_stack[i]
    if logic not in target:
        continue
    m = re.search (reg1, target)
    if m is not None:
        continue
    ~~m = re.search (reg1,~~
    m = re.search (reg0, target)
    if m is none:
        continue
    for j in re.findall (reg0, target)
        child = self.my_stack[int(j)]
        if logic not in child:
            continue
        new_reg = "(^|\s)" + j + '
        self.myStack[i] = re.sub (new_reg, ' ' + child + ' ')
        self.myStack[i] = self.my_stack[i].strip()
        flag = true
```


Sangeev

Sanjeev Kumar Singh
IBM18CSa93

```
if flag:
sef. merge-item (logic)

def Demorgan (Sentence)
  String = ''. join (list (Sentence). copy ())
  String = String. replace ('--', '')
  flg = '[' in String
  String = String. replace ('~[', '')
  String = String. strip (']')
  for predicate in getPredicate (String):
     String = String. replace (predicate, f'{~predicate}')
  S = list (String)
  for i, c in enumerate (String):
     if c == 'V'
        S[i] = '^'
     elif c == '^'
        S[i] = 'V'
  String = ''. join (S)
  String = String. replace ('~~', '')
  return f'[{String}]' if flg else String

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 ('[A]', Statement)
  for match in matches [::-1]:
     Statement = Statement. replace (match, '')
     Statement = re. findall ('\(\[^]]+\)', Statement)
```


Sanjeev

Sanjeev Kumar Singh
IBM18CS093

```
for S in statement:
    Statement = statement.replace(S, S[1:-1])
for predicate in get_predicates(statement):
    if ''.join(attributes).islower():
        statement = statement.replace(
            match(1), skolem_constant.pop(0))
    else:
        aL = [a for a in attribute if a.islower()]
        aU = [a for a in attribute if not in a.islower()][0]
        statement = statement.replace(aU, f'{skolem.pop(0)}{aL[0]}')
return statement

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 = new_statement
    statement = statement.replace("=>", "-")
    expr = '\[([^\]]+)\]'
    statement = re.findall(expr, statement)
    for i, s in enumerate(statements):
        if '[' in s & ']' not in s:
            statement[i] += ']'
    for s in statements:
        statement = statement.replace(s, fol_to_cnf(s))
```


Sanjeev

Sengeev Kumar Singh
1BM18CS093

```
while '-' in Statement:
    i = Statement.index('-')
    br = Statement.index('(') if '[' in statement else 0
    new Statement = '~' + Statement[br:i] + 'v' +
                    Statement[i+1:]

while '-∀' in Statement:
    i = Statement.index('~∀')
    Statement = list(Statement)
    Statement = ''.join(Statement)
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('~[∃', '[~∃')
    capr = '([~[∀|∃])'

    for s in Statement:
Statement = Statement.replace(s, fol_to_cnf(s))
capr = r'~|[[^]]+|]'

    for s in Statements:
    Statement = Statement.replace(s, Demorgan(s))
    return Statement
fol = input("Enter FOL Statement\n")
    print("\nCNF form is :")
    print(Skolemization(fol_to_cnf(fol)))
```


Sengeev