Computer Science and Engineering VG SEM~ 5

Rutagent Ritik Rout 113MISCSISI 1/1/20

Course: - Artificial gradligence Course code: - 20CS5 PCAIP

def get Attributes (strings):

expr = '\((\(\L^1) \right) + \)'

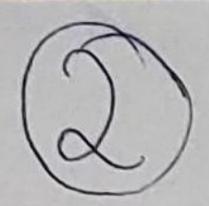
mothers = re. findall (expr, string)

return [m for m in str (matched) if m. is alpha ()]

def get Predicates (string): expr = '[a-z~]+1(EA-za-z,]+1)'
return re. findall (expr, string)

def DeMargan (sentence): String = ''. join (list (sentence). copy (s) String = string. replace ('nn', '1) flag = '[in string String = String replace (~[', '') (iti) dints. Lounts = buyes for predicate in get Predicates (string): string = (tring. strip (']')
for predicate in get Predicates (string): String = String. replace (predicated) (ist Cetting) "¿ C in en une rate (string). elif c== 1 &1. :(5) violi: = Louges string = string replacel no, return f'[{& hing }] if flag else string

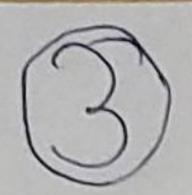
D'XOAR



Skolemization (sentence): SKOLEM_ CONSTANTS = [f'{ chr(c)}' for cin soongs rong Cord ('A'), ord ('Z') + 1)] statement = 1. join (list (sentence). copy()) matches = re. findall ("[+], statement) too match in matches [::-1]: Statement = Statement-replace (mottch, ") Statements = re. findall ('I[I[1]]+\]]' Statement) for sin statement Statement = Statement replace (s, S[1:-1]) for predicate in getPridicates (statement): attribute = get Attribute (predicate) if "! join lattribute) is lowers. Statement = stament. replace (match [1], SKOLEM_CONSTATUTS. pop(0)) all ta for a in attributes it not · a, is lower () I LOJ Statement = statement. replace (aU) & 1 & SKO LFML CONSTATS, POPLODY (& match [1] 3)'). return statement

import re

byrogz

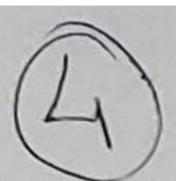


dif tol-to crof (tol): Statement = fol. replace ("<=>","-") while '-' in statement:

i = statement.index('-') new_statement = "['+ statement [: i]+ '>'+ etatement [i+1:]+ 'J&['+ statement[i+1:] + '=>'+ Statement [:i]+ 1 J1 Statement = new - statement statement = statement . replace (=5", "-") expr = 1. [([]] +) \]' Statement = re. tindall (expr, statement). if '['I' in s and 'J' not is s: Statement [i] += 1] for a in statements! Statement = statement - replace (S, fol-to cnof(s)) ushile !-! in statement: i = statement.index('-') br = statement. indux ("[") if "[" in - Statement else O new_statement = 1 ~ 1 statement [br: i]+11+ Statement [i+1:] Statement = Statement [: brid + new - Statement if by o else new - statement while 't' in stalement!

('Y ~') rubni, trumstrot? = ; Statement = list (Statement) Statement [i] statement [i+1]+ Statement [i+2]
= 'F', Statement [i+2], ~ Statement = 11. join (statements).

Scanned by TapScanner



while '~3' in Statement:

i = Statement · indu (~3').

S = list (Statement)

S[i] & [i + 1], & [i + 2] = '+', & [i + 2], ~'

Statement = '.' join (S).

Statement = Statement · replace (~[+]', [-+4'])

expr = 'C [++ 1 =].)

expr = 'C [++ 1 =].)

statement = re. fin doll (expr, statement).

bor s in Statements:

Statement = statement · replace (e, tol to - cnf(s))

expr = 'n | [[^]] + |]

statement = re. fin doll (expr, statement)

for c in statement:

Statement = re. fin doll (expr, Statement)

tor c in statement:

Statement = statement · replace (s, DeMargen (s))

return statement

def Stort (" Enter the queries"

print ("Enter the queries"

queries")

print ("\n")

for q in qs

print (skeolemization (fol-to-onf (q)))

print ("\n")

Start ().