Artificial Intelligence LAB-7

Unification and Resolution

Date:8-2-22

-<u>Source Code:</u>

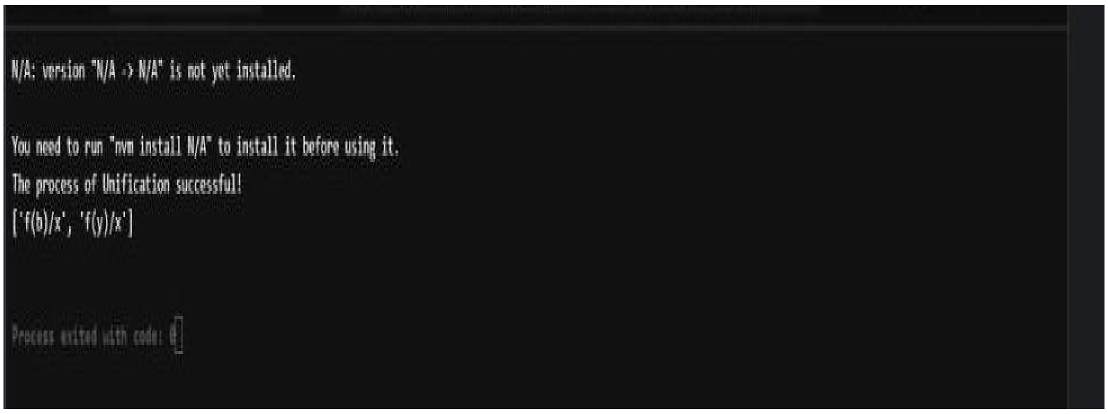
```
def get_index_comma(string): index_list =  list() par_count = 0

for i in range(len(string)):
if string[i] == ',' and par_count == 0: index_list.append(i)
elif string[i] == '(': par_count += 1 elif string[i] == ')':
par_count -= 1


return index_list




def is_variable(expr): for i in expr:
if i == '(' or i == ')': return False

return True




def process_expression(expr): expr = expr.replace(' ', '') index = None
for i in range(len(expr)): if expr[i] == '(':
index = i break
predicate_symbol = expr[:index]
expr = expr.replace(predicate_symbol, '') expr = expr[1:len(expr) - 1]
arg_list = list()
indices = get_index_comma(expr)


if len(indices) == 0: arg_list.append(expr)
else:
```

```
arg_list.append(expr[:indices[0]]) for i, j in zip(indices, indices[1:]):
arg_list.append(expr[i + 1:j])
arg_list.append(expr[indices[len(indices) - 1] + 1:])


return predicate_symbol, arg_list




def get_arg_list(expr):
_, arg_list = process_expression(expr)

flag = True while flag:
flag = False


for i in arg_list:
if not is_variable(i): flag = True
_, tmp = process_expression(i) for j in tmp:
if j not in arg_list: arg_list.append(j)
arg_list.remove(i)


return arg_list




def check_occurs(var, expr): arg_list = get_arg_list(expr) if var in arg_list:
return True


return False
```

```
def unify(expr1, expr2):


if is_variable(expr1) and is_variable(expr2): if expr1 == expr2:
return 'Null' else:

return False
elif is_variable(expr1) and not is_variable(expr2): if check_occurs(expr1,
expr2):
return False else:
tmp = str(expr2) + '/' + str(expr1) return tmp
elif not is_variable(expr1) and is_variable(expr2): if check_occurs(expr2,
expr1):
return False else:
tmp = str(expr1) + '/' + str(expr2) return tmp
else:
predicate_symbol_1, arg_list_1 = process_expression(expr1)
predicate_symbol_2, arg_list_2 = process_expression(expr2)

# Step 2
if predicate_symbol_1 != predicate_symbol_2: return False
# Step 3
elif len(arg_list_1) != len(arg_list_2): return False
else:
# Step 4: Create substitution list sub_list = list()

# Step 5:
for i in range(len(arg_list_1)):
tmp = unify(arg_list_1[i], arg_list_2[i])


if not tmp: return False
elif tmp == 'Null': pass
else:
if type(tmp) == list: for j in tmp:
```

```
sub_list.append(j)
else:
sub_list.append(tmp)


# Step 6 return sub_list



if     name     == ' main ':


f1 = 'Q(a, g(x, a), f(y))'
f2 = 'Q(a, g(f(b), a), x)' # f1 = input('f1 : ')
# f2 = input('f2 : ')


result = unify(f1, f2) if not result:
print('The process of Unification failed!') else:
print('The process of Unification successful!') print(result)
```

Output



```
N/A: version "N/A -> N/A" is not yet installed.

You need to run "nvm install N/A" to install it before using it.
The process of Unification successful!
['f(b)/x', 'f(y)/x']



Process exited with code: 0
```