
Exercise 7 – Inference from Propositional Logic

Date: 03/11/2022

Aim:

To write functions for the connectives of propositional logic and validate expressions for tautology and equivalence.
Connectives: AND, OR, NOT, IMPLICATION, BI-CONDITIONAL

Code:

```
#Inference from Propositional Logic

T = True
F = False
value = {'T':1, 'F':0}

# 1:and 2:or 3:not 4:implies 5:biconditional

def precedence(op):
  return op

def notFun(a):
  return not a

def andFun(a,b):
  return a and b

def orFun(a,b):
  return a or b

def impliesFun(a,b):
  if a==T and b==F:
    return False
  return True

def biconFun(a,b):
  if impliesFun(a,b) and impliesFun(b,a):
    return True
```

```
  return False


track = 1
print("\n1. p -> (p or q)\n")
check = True
print('p', '  ', 'q', '  ', 'expr')
print("--------------")
for a in range(2):
  for b in range(2):
    check = impliesFun(a, orFun(a,b))
    print(a, '  ', b, '  ', check)
    if (not check):
      track = 0
if (track):
  print("Tautology")
else:
  print("Not a tautology")

track = 1
print("\n2. ~p -> p\n")
check = True
print('p', '  ', '~p', '  ', 'expr')
print("--------------")
for a in range(2):
  check = impliesFun(notFun(a),a)
  print(a, '  ', int(notFun(a)), '  ', check)
  if (not check):
    track = 0
if (track):
  print("Tautology")
else:
  print("Not a tautology")

track = 1
print("\n3. (~a or b), (a->b)\n")
check1, check2 = True, True
print('a', '  ', 'b', '  ', 'expr1', ' \t', 'expr2')
print("------------------------")
for a in range(2):
  for b in range(2):
    check1 = orFun(notFun(a), b)
    check2 = impliesFun(a,b)
```

```
    print(a, '  ', b, '  ', bool(check1), ' \t', bool(check2))
    if (check1 != check2):
      track = 0
if (track):
  print("Equivalent")
else:
  print("Not equivalent")

track = 1
print("\n4. (p or ~p), (p and ~p)\n")
check1, check2 = True, True
print('p', '  ', '~p', '  ', 'expr1', ' \t', 'expr2')
print("-------------------------")
for a in range(2):
  check1 = orFun(a, notFun(a))
  check2 = andFun(a, notFun(a))
  print(a, '  ', int(notFun(a)), '  ', bool(check1), ' \t', bool(check2))
  if (check1 != check2):
    track = 0
if (track):
  print("Equivalent")
else:
  print("Not equivalent")
```

Output:

Checking for Tautology:

```
~/AIwork$ python ex7logic.py

1. p -> (p or q)

p    q    expr
-------------
0    0    True
0    1    True
1    0    True
1    1    True
Tautology

2. ~p -> p

p    ~p    expr
--------------
0    1    False
1    0    True
Not a tautology
```

Checking for Equivalence:

```
3. (~a or b), (a->b)

a    b    expr1      expr2
-----------------------
0    0    True       True
0    1    True       True
1    0    False      False
1    1    True       True
Equivalent

4. (p or ~p), (p and ~p)

p    ~p    expr1     expr2
------------------------
0    1    True       False
1    0    True       False
Not equivalent
```