**AML Algorithm #6 : Candidate Elimination Algorithm using EnjoySport dataset**

import csv

with open('EnjoySport2.csv') as csvFile:

examples = [tuple(line) for line in csv.reader(csvFile)]

def get\_domains(examples):

d = [set() for i in examples[0]]

for x in examples:

for i, xi in enumerate(x):

d[i].add(xi)

return [list(sorted(x)) for x in d]

def g\_0(n):

return ('?',)\*n

def s\_0(n):

return ('0',)\*n

def more\_general(h1, h2):

more\_general\_parts = []

for x, y in zip(h1, h2):

mg = x == '?' or (x != '0' and (x == y or y == '0'))

more\_general\_parts.append(mg)

return all(more\_general\_parts)

def consistent(hypothesis,example):

return more\_general(hypothesis, example)

def min\_generalizations(h, x):

h\_new = list(h)

for i in range(len(h)):

if not consistent(h[i:i+1],x[i:i+1]):

if h[i] != '0':

h\_new[i] = '?'

else:

h\_new[i] = x[i]

return [tuple(h\_new)]

def generalize\_S(x, G, S):

S\_prev = list(S)

for s in S\_prev:

if s not in S:

continue

if not consistent(s,x):

S.remove(s)

Splus = min\_generalizations(s, x)

S.update([h for h in Splus if any([more\_general(g,h) for g in G])])

S.difference\_update([h for h in S if any([more\_general(h, h1) for h1 in S if h != h1])])

return S

def min\_specializations(h, domains, x):

results = []

for i in range(len(h)):

if h[i] == '?':

for val in domains[i]:

if x[i] != val:

h\_new = h[:i] + (val,) + h[i+1:]

results.append(h\_new)

elif h[i] != '0':

h\_new = h[:i] + ('0',) + h[i+1:]

results.append(h\_new)

return results

def specialize\_G(x, domains, G, S):

G\_prev = list(G)

for g in G\_prev:

if g not in G:

continue

if consistent(g,x):

G.remove(g)

Gminus = min\_specializations(g, domains, x)

G.update([h for h in Gminus if any([more\_general(h, s) for s in S])])

G.difference\_update([h for h in G if any([more\_general(g1, h) for g1 in G if h != g1])])

return G

def candidate\_elimination(examples):

domains = get\_domains(examples)[:-1]

G = set([g\_0(len(domains))])

S = set([s\_0(len(domains))])

i=0

print('All the hypotheses in General and Specific boundary are:\n')

print('\n G[{0}]:'.format(i),G)

print('\n S[{0}]:'.format(i),S)

for xcx in examples:

i=i+1

x, cx = xcx[:-1], xcx[-1]

if cx=='Yes': # x is positive example

G = {g for g in G if consistent(g,x)}

S = generalize\_S(x, G, S)

else: # x is negative example

S = {s for s in S if not consistent(s,x)}

G = specialize\_G(x, domains, G, S)

print('\n G[{0}]:'.format(i),G)

print('\n S[{0}]:'.format(i),S)

return

candidate\_elimination(examples)

**OUTPUT : -**

All the hypotheses in General and Specific boundary are:

G[0]: {('?', '?', '?', '?', '?', '?')}

S[0]: {('0', '0', '0', '0', '0', '0')}

G[1]: {('?', '?', '?', '?', '?', '?')}

S[1]: {('Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same')}

G[2]: {('?', '?', '?', '?', '?', '?')}

S[2]: {('Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same')}

G[3]: {('?', 'Warm', '?', '?', '?', '?'), ('?', '?', '?', '?', '?', 'Same'), ('Sunny', '?', '?', '?', '?', '?')}

S[3]: {('Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same')}

G[4]: {('Sunny', '?', '?', '?', '?', '?'), ('?', 'Warm', '?', '?', '?', '?')}

S[4]: {('Sunny', 'Warm', '?', 'Strong', '?', '?')}