candidate elimination

In [1]:

```
import random
import csv
```

In [2]:

```
def g_0(n):
    return ("?",)*n

def s_0(n):
    return ('0',)*n
```

In [3]:

```
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)

l1 = [1, 2, 3]
l2 = [3, 4, 5]

list(zip(l1, l2))
```

Out[3]:

```
[(1, 3), (2, 4), (3, 5)]
```

In [4]:

```
# min_generalizations
def fulfills(example, hypothesis):
    ### the implementation is the same as for hypotheses:
    return more_general(hypothesis, example)

def min_generalizations(h, x):
    h_new = list(h)
    for i in range(len(h)):
        if not fulfills(x[i:i+1], h[i:i+1]):
            h_new[i] = '?' if h[i] != '0' else x[i]
    return [tuple(h_new)]
```

In [5]:

```
min_generalizations(h=('0', '0'  , 'sunny'),
                    x=('rainy', 'windy', 'cloudy'))
```

Out[5]:

```
[('rainy', 'windy', '?')]
```

In [6]:

```
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
```

In [7]:

```
min_specializations(h=('?', 'x',),
                    domains=[['a', 'b', 'c'], ['x', 'y']],
                    x=('b', 'x'))
```

Out[7]:

```
[('a', 'x'), ('c', 'x'), ('?', '0')]
```

In [8]:

```
with open('C:\\Users\\Student\\Desktop\\data set\\finds.csv')  as csvFile:
        examples = [tuple(line) for line in csv.reader(csvFile)]

#examples = [('sunny', 'warm', 'normal', 'strong', 'warm', 'same',True),
# ('sunny', 'warm', 'high', 'strong', 'warm', 'same',True),
# ('rainy', 'cold', 'high', 'strong', 'warm', 'change',False),
# ('sunny', 'warm', 'high', 'strong', 'cool', 'change',True)]

examples
```

Out[8]:

```
[('Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same', 'Yes'),
 ('Sunny', 'Warm', 'High', 'Strong', 'Warm', 'Same', 'Yes'),
 ('Rainy', 'Cold', 'High', 'Strong', 'Warm', 'Change', 'No'),
 ('Sunny', 'Warm', 'High', 'Strong', 'Cool', 'Change', 'Yes')]
```

In [9]:

```
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]

get_domains(examples)
```

Out[9]:

```
[['Rainy', 'Sunny'],
 ['Cold', 'Warm'],
 ['High', 'Normal'],
 ['Strong'],
 ['Cool', 'Warm'],
 ['Change', 'Same'],
 ['No', 'Yes']]
```

In [10]:

```
def candidate_elimination(examples):
    domains = get_domains(examples)[:-1]

    G = set([g_0(len(domains))])
    S = set([s_0(len(domains))])
    i=0
    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]  # Splitting data into attributes and decisions
        if cx=='Y': # x is positive example
            G = {g for g in G if fulfills(x, g)}
            S = generalize_S(x, G, S)
        else: # x is negative example
            S = {s for s in S if not fulfills(x, s)}
            G = specialize_G(x, domains, G, S)
        print("\n G[{0}]:".format(i),G)
        print("\n S[{0}]:".format(i),S)
    return
```

In [11]:

```
def generalize_S(x, G, S):
    S_prev = list(S)
    for s in S_prev:
        if s not in S:
            continue
        if not fulfills(x, s):
            S.remove(s)
            Splus = min_generalizations(s, x)
            ## keep only generalizations that have a counterpart in G
            S.update([h for h in Splus if any([more_general(g,h)
                                               for g in G])])
            ## remove hypotheses less specific than any other in S
            S.difference_update([h for h in S if
                                 any([more_general(h, h1)
                                      for h1 in S if h != h1])])
    return S
```

In [12]:

```
def specialize_G(x, domains, G, S):
    G_prev = list(G)
    for g in G_prev:
        if g not in G:
            continue
        if fulfills(x, g):
            G.remove(g)
            Gminus = min_specializations(g, domains, x)
            ## keep only specializations that have a conuterpart in S
            G.update([h for h in Gminus if any([more_general(h, s)
                                                for s in S])])
            ## remove hypotheses less general than any other in G
            G.difference_update([h for h in G if
                                 any([more_general(g1, h)
                                      for g1 in G if h != g1])])
    return G
```

In [13]:

```
candidate_elimination(examples)
```

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

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

 G[1]: {('?', '?', '?', '?', 'Cool', '?'), ('?', 'Cold', '?', '?', '?',
'?'), ('Rainy', '?', '?', '?', '?', '?'), ('?', '?', 'High', '?', '?',
'?'), ('?', '?', '?', '?', '?', 'Change')}

 S[1]: {('0', '0', '0', '0', '0', '0')}

 G[2]: {('?', '?', '?', '?', 'Cool', '?'), ('?', 'Cold', '?', '?', '?',
'?'), ('Rainy', '?', '?', '?', '?', '?'), ('?', '?', '?', '?', '?', 'Chang
e')}

 S[2]: {('0', '0', '0', '0', '0', '0')}

 G[3]: {('?', 'Cold', 'Normal', '?', '?', '?'), ('Sunny', 'Cold', '?',
'?', '?', '?'), ('Rainy', 'Warm', '?', '?', '?', '?'), ('Sunny', '?', '?',
'?', '?', 'Change'), ('?', 'Cold', '?', '?', '?', 'Same'), ('?', '?', '?',
'?', 'Cool', '?'), ('Rainy', '?', '?', '?', '?', 'Same'), ('Rainy', '?',
'Normal', '?', '?', '?'), ('?', '?', 'Normal', '?', '?', 'Change'), ('?',
'Warm', '?', '?', '?', 'Change')}

 S[3]: {('0', '0', '0', '0', '0', '0')}

 G[4]: {('Sunny', 'Cold', '?', '?', '?', '?'), ('Rainy', '?', '?', '?',
'?', 'Same'), ('?', 'Cold', '?', '?', '?', 'Same'), ('Rainy', '?', 'Norma
l', '?', '?', '?'), ('?', 'Cold', '?', '?', 'Cool', '?'), ('?', '?', 'Norm
al', '?', 'Cool', '?'), ('?', 'Cold', 'Normal', '?', '?', '?'), ('?', 'War
m', '?', '?', 'Warm', 'Change'), ('Rainy', 'Warm', '?', '?', '?', '?'),
('Rainy', '?', '?', '?', 'Cool', '?'), ('?', '?', 'Normal', '?', '?', 'Cha
nge'), ('?', '?', '?', '?', 'Cool', 'Same'), ('Sunny', '?', '?', '?', 'War
m', 'Change')}

 S[4]: {('0', '0', '0', '0', '0', '0')}
```

In [43]:

```
list  = [1,2,3,4,5]
```

In [ ]:

```
list[:]
```

In [ ]:

```
list[1:-1]
```