Ejercicio sobre Factores de Certeza

Ejemplo 1

La base de conocimientos de un sistema inteligente contiene las siguientes reglas de producción con sus correspondientes factores de certeza asociados:

> R1: Si H1 y H3 Entonces C1, 0.6

> R2: Si C1 Entonces C, -1

> R3: Si H2 y H4 Entonces C2, 0.4

> R4: Si C2 Entonces C, 0,5

> R5: Si H3 ó H5 ó H6 Entonces C2, 0.9

El sistema experto que usa esta base de conocimientos observa los hechos H1 y H3 con factores de certeza respectivos 0.4 y -0.3

¿Cuál sería la certeza de C a partir de las reglas enunciadas y las observaciones disponibles?

Indicar detalladamente los cálculos realizados.

¿qué es lo que tenemos/sabemos?

- R1: Si (H1, 0.4) y (H3,-0.3) Entonces (C1, 0.6)
- R2: Si C1 Entonces (C, -1)
- R3: Si H2 y H4 Entonces (C2, 0.4)
- R4: Si C2 Entonces (C, 0,5)
- R5: Si (H3,-0.3) ó H5 ó H6 Entonces (C2, 0.9)

In [13]:

```
def reglas(r,tups):
    if r == 1:
        if tups[0] == ('H1',0.4) and tups[1] == ('H3',-0.3):
            return ('C1',0.6)
    if r == 2:
        # print('reglas',r,tups)
        if tups[0][0] == 'C1':
            # print("tups[0] == ('C1') OK =>",('C', -1))
            return ('C', -1)
    if r == 3:
        if tups[0][0] == 'H2' and tups[1][0] == 'H4':
            return ('C2',0.4)
    if r == 4:
        if tups[0][0] == 'C2':
            return ('C', 0.5)
    if r == 5:
        if tups[0] == ('H3',-0.3) or tups[1][0] == 'H5' or tups[2][0] == 'H6':
            return ('C2',0.9)
    else:
        return None
```

Consideremos la regla 1: Si (H1, 0.4) y (H3,-0.3) Entonces (C1, 0.6)

Si una regla tiene varias condiciones enlazadas con una conjunción (&) y cada una tiene asociado un factor de certeza:

R_1: Si $e_1, CF(e_1) \& \cdots \& e_n, CF(e_n)$ Entonces $h, CF_R(h, e_1 \wedge \cdots \wedge e_n)$

donde

> $CF_R(h, e_1 \wedge \cdots \wedge e_n)$ es el factor de certeza de la regla R_1

> $CF(e_1)$ es el factor de certeza de e_1 y

> $CF(e_n)$ es el factor de certeza de e_n

¿cuál es la certeza de h con las n evidencias presentes?

$$CF(h, e_1 \wedge \cdots \wedge e_n) = CF_R(h, e_1 \wedge \cdots \wedge e_n) * min[CF(e_1), \cdots, CF(e_n)]$$

In [14]:

```
import factores_certeza as fc

evidencias = [('H1', 0.4), ('H3',-0.3)]

# Para combinar dos factores de certeza en uno, tenemos
# una función para calcular el factor de certeza combinado de
# dos reglas que comparten la misma hipótesis

def factor_certeza_and(r, h,levid):
    r1 = reglas(r,levid)
    # print('factor_certeza_and', r1)
    if tuple(r1) and h == r1[0]: # ¿h=='C'?
        cfs = [cf for (_,cf) in levid]
        return (h,r1[1]*min(cfs)) # r1[1] es CFR(h,e1 and e2)
    else:
        return None

# aplicamos la regla 1
cfR1 = factor_certeza_and(1,'C1',[('H1', 0.4), ('H3',-0.3)])
cfR1
```

Out[14]:

```
('C1', -0.18)
```

In [15]:

```
# Teniendo C1, podemos aplicar la regla 2: Si C1 Entonces (C, -1)
cfR2 = factor_certeza_and(2,'C',[('C1', -0.18)])
cfR2
```

Out[15]:

```
('C', 0.18)
```

Consideremos la regla 5: Si (H3,-0.3) ó H5 ó H6 Entonces (C2, 0.9)

Si una regla tiene varias condiciones enlazadas con una disyunción (|) y cada una tiene asociado un factor de certeza:

R_1: Si $e_1, CF(e_1)| \cdots |e_n, CF(e_n)$ Entonces $h, CF_R(h, e_1 \vee \cdots \vee e_n)$

> $CF_R(h, e_1 \vee \cdots \vee e_n)$ es el factor de certeza de la regla R_1

> $CF(e_1)$ es el factor de certeza de e_1 y

> $CF(e_n)$ es el factor de certeza de e_n

¿cuál es la certeza de h con las n evidencias presentes?

$$CF(h, e_1 \vee \cdots \vee e_n) = CF_R(h, e_1 \vee \cdots \vee e_n) * max[CF(e_1), \cdots, CF(e_n)]$$

In [16]:

```
def factor_certeza_or(r,h,levid):
    r2 = reglas(r, levid)
    if tuple(r2) and h == r2[0]: # ¿h=='C'?
        cfs = [cf for (_,cf) in levid]
        return (h,r2[1]*max(cfs)) # r1[1] es CFR(h,e1 and e2)
    else:
        return None

# aplicamos la regla 5
cfR5 = factor_certeza_and(5,'C2',[('H3',-0.3)])
cfR5
```

Out[16]:

```
('C2', -0.27)
```

Apliquemos ahora la regla R4: Si C2 Entonces (C, 0,5)

In [17]:

```
cfR4 = factor_certeza_and(4,'C',[('C2', -0.27)])
cfR4
```

Out[17]:

```
('C', -0.135)
```

In [18]:

```
cfR4
```

Out[18]:

```
('C', -0.135)
```

In [19]:

```
# definimos una función para calcular el factor de certeza combinado de
# dos reglas que comparten la misma hipótesis

def factor_certeza(h, x, y):
    if x>0 and y>0:
        return (h, x+y-x*y)
    elif x<0 and y<0:
        return (h, x+y+x*y)
    else:
        return (h, (x+y)/(1-min(abs(x),abs(y))))
```

In [20]:

```
fcC = factor_certeza('C', 0.18, -0.135)
fcC
```

Out[20]:

```
('C', 0.05202312138728322)
```

In [ ]:

```

```