PRÁCTICO 1

INF428 SB-Sistemas Expertos. Gestión 1-2024.

1. Convertir la siguiente regla en reglas esenciales **válidas**:

```
IF (P=b ∨ y=0) → Q=a
   THEN
      P=a ∨ Q=b
```

P,Q ∈ {a, b}, y∈R

También, **saque** de la solución aquellas reglas cuyas premisas son siempre False.

2. Convertir la siguiente regla en reglas esenciales **válidas**:

```
IF x<0 ∧ (x>0 ∨ Q=b)
   THEN                    P,Q∈{a, b}, x∈R
     ¬(P≠a → Q=a)
```

También, **saque** de la solución aquellas reglas cuyas premisas son siempre False.

3. Convertir la siguiente regla en reglas **FND válidas**:

```
IF (P=a ∨ Q=a ∨ S=a)→x=0
   THEN
     x=0 ∨ S=b
```

P,Q, S∈{a, b}, x∈R

No sobre convierta las reglas hasta que éstas sean esenciales. El proceso de conversión debe parar cuando las reglas ya estén en su FND.

4. Convertir la siguiente regla en reglas esenciales **válidas**:

```
IF (Q=b ∧ x=0) ∨ Z=b
   THEN                 Z,Q∈{a, b}, x∈R
     x≠0 ∨ Z=a
```

También, **saque** de la solución aquellas reglas cuyas premisas son siempre False.

5. Muestre que en la siguiente BC, es **inconsistente**.

```
R1) IF x=0                R4) IF S > 4
       THEN                      THEN
         P=a                       C=b

R2) IF x=0                R5) IF S < 4
       THEN                      THEN
         C=b                       P=a

R3) IF P=a ∨ Q=b          R6) IF S=10
       THEN                      THEN
         C=a                       Q=a
```

6. Con la siguiente BC={R1, R2} y la BH= [Z=c, P=50], realice un FWC manual, sabiendo que: Con R1 se hace un Modus Ponens y con R2 se hace un Modus Tollens

```
R1) IF Q≠0 ∨ P=0          R2) IF Q≠10 ∧ P>10
       THEN                      THEN
         M=a                       Z=a
```

Muestre los cálculos realizados

7. *(FWC)* Se tiene una BC donde cada una de sus reglas tienen **premisas atómicas**. Escribir un FWC (sin metas) para esta BC

```
void FWC(Regla[1..N], BH)
  //Este FWC empieza con la BH vacía.
```

el cual antes de la Pasada a la BC, obtiene la primera regla no-marcada R y consulta al cliente por el valor de la variable de la premisa de R.

Recuerde que al cliente no se le puede consultar dos veces por el valor de una misma variable.

8. Escribir un FWC (raro) sin metas

```
void FWC(Regla[1..N], BH, Pila HN)
  //La BH no necesariamente está vacía.
```

Este encadenamiento hacia adelante, tiene un comportamiento normal. Sin embargo, **antes** de realizar la pasada a la BC, saca un hecho nativo H de la Pila HN (H=HN.pop()) y consulta al cliente por el valor de H.

Todos los hechos nativos están guardados en la Pila HN. La Pila HN no tiene duplicados.

9. Escriba el siguiente FWC

```
void FWC(Regla[1..N], BH, Meta)
```

Este encadenamiento hacia adelante divide lógicamente la BC en dos: BC1=Regla[1..N/2] y BC2=Regla[N/2+1..N]. En cada uno de sus ciclos, realiza una pasada a una de las BC's alternativamente.

Note que en cada ciclo del while (true) se hace una pasada a BC1 o a BC2, pero no a ambas.

10. Se tiene una BC donde cada una de sus reglas tienen **premisas atómicas** y empieza con una **BH vacía**. Escribir el algoritmo

```
void FWC(Regla[1..N], BH, Lista HN)
  //La lista HN tiene a TODOS los hechos nativos.
```

el cual (en la pasada) trabaja normalmente con c/u de las reglas de la BC, excepto con aquellas cuya variable de su premisa es un hecho nativo NO unificado. Como el hecho nativo (variable) no está unificado, se realiza una consulta (pregunta) al cliente por el valor de esta variable.

Recuerde que al cliente, no se le puede consultar más de una vez, por el valor de una variable.

11. Escriba el siguiente FWC para una Meta

```
void FWC(Regla[1..N], BH, Meta, Pila HN)
```
*/*La BH empieza vacía. En la Pila HN, que no tiene elementos duplicados, están **todos** los hechos nativos*/*

Este encadenamiento hacia adelante divide lógicamente la BC en dos: BC1=Regla[1..N/2] y BC2=Regla[N/2+1..N].

En cada uno de sus ciclos, realiza una pasada a una de las BC's alternativamente.

Sin embargo, luego de realizar la pasada a BC1, saca un hecho nativo H de la Pila (H=HN.pop()) y consulta al cliente por el valor de H.

Note que en cada ciclo del while se hace una pasada a BC1 o a BC2, pero no a ambas.