



PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC1253 - MATEMÁTICAS DISCRETAS

Tarea 1

22 de agosto de 2025

2º semestre 2025 - Profesores M. Arenas - A. Kozachinskiy - M. Romero
Álvaro Panozo - 24664037

Respuestas

Pregunta 1

Pregunta 1.1

Sea $\varphi \equiv p \rightarrow (q \rightarrow (r \rightarrow s))$
 $\equiv p \rightarrow (\neg q \vee r \vee s)$
 $\equiv \neg p \vee q \vee r \vee s$

Consideramos entonces $\neg p \vee q \vee r \vee s$ como clausula unica para una formula CNF, y entonces φ es una formula CNF. 

Pregunta 1.2

Sabemos que $\{\neg, \rightarrow\}$ es FC, entonces $\{\oplus, \rightarrow\}$ es FC si podemos expresar \neg con \oplus . Tomemos $p \in \mathcal{L}(p)$, luego

p	$\neg p$
0	1
1	0

Notemos que $\sigma(p \oplus p) = 0$ cuando $\sigma(\varphi) = 1$, entonces necesitamos un ϕ tal que $\sigma(\phi) = 1$ cuando $\sigma'(p) = 0$, por lo que nos aprovecharemos de la implicancia, que es 0 cuando $V \rightarrow F$ únicamente. Entonces:

Vemos entonces que $\neg p \equiv p \rightarrow (p \oplus p)$, y así $\{\oplus, \rightarrow\}$ es FC.

p	$\neg p$	$p \oplus p$	$p \rightarrow (p \oplus p)$
0	1	0	1
1	0	0	0



Pregunta 1.3

El conectivo \oplus es verdadero si solo si los valores de verdad que se relacionan son distintos, por lo que $p \oplus p$ es una contradicción, entonces es imposible expresar $\neg p$ solo usando \oplus , y así \oplus no puede expresar todas las formulas proposicionales. 

Pregunta 2

Pregunta 2.1

Sea $x_{i,j,k}$ donde i es la fila, j la columna, k el numero. Tenemos entonces 4 restricciones para cumplir con el cuadrado latino $n \times n$:

- Cada casilla tiene al menos 1 numero

$$\bigwedge_{i=1}^n \bigwedge_{j=1}^n \left(\bigvee_{k=1}^n x_{i,j,k} \right)$$

- Cada casilla tiene exactamente un numero

$$\bigwedge_{i=1}^n \bigwedge_{j=1}^n \bigwedge_{k=1}^{n-1} \bigwedge_{m=k+1}^n x_{i,j,k} \rightarrow \neg x_{i,j,m}$$



- Cada elemento de la columna debe ser distinto

$$\bigwedge_{i=1}^n \left(\bigwedge_{k=1}^n \left(\bigwedge_{j=1}^{n-1} \bigwedge_{m=j+1}^n x_{i,j,k} \rightarrow \neg x_{i,m,k} \right) \right)$$



- Analogamente, cada fila debe tener elementos distintos

$$\bigwedge_{j=1}^n \left(\bigwedge_{k=1}^n \left(\bigwedge_{i=1}^{n-1} \bigwedge_{m=i+1}^n x_{i,j,k} \rightarrow \neg x_{m,j,k} \right) \right)$$



Luego, la conjunción de todas las restricciones nos da

$$\begin{aligned} \varphi \equiv & \left(\bigwedge_{i=1}^n \bigwedge_{j=1}^n \bigvee_{k=1}^n x_{i,j,k} \right) \wedge \left(\bigwedge_{i=1}^n \bigwedge_{j=1}^n \bigwedge_{k=1}^{n-1} \bigwedge_{m=k+1}^n x_{i,j,k} \rightarrow \neg x_{i,j,m} \right) \wedge \left(\bigwedge_{i=1}^n \left(\bigwedge_{k=1}^n \left(\bigwedge_{j=1}^{n-1} \bigwedge_{m=j+1}^n x_{i,j,k} \rightarrow \neg x_{i,m,k} \right) \right) \right) \wedge \\ & \left(\bigwedge_{j=1}^n \left(\bigwedge_{k=1}^n \left(\bigwedge_{i=1}^{n-1} \bigwedge_{m=i+1}^n x_{i,j,k} \rightarrow \neg x_{m,j,k} \right) \right) \right) \end{aligned}$$

Pregunta 2.2

El código usado para comprobar la fórmula anterior es:

```

from z3 import *
def SAT_Solver(n, celdas = False):
    solver = Solver()
    # Definimos  $x[i][j][k]$  como una variable booleana
    x = [[[Bool(f"x_{i}_{j}_{k}")]
           for i in range(n)]
          for j in range(n)]
             for k in range(n)]

    #Restriccion 1: Cada casilla tiene al menos un numero
    for i in range(n):
        for j in range(n):
            solver.add(Or([x[i][j][k] for k in range(n)]))

    #Restriccion 2: Cada casilla tiene exactamente un numero
    for i in range(n):
        for j in range(n):
            for k in range(n-1):
                for m in range(k+1,n):
                    solver.add(Implies(x[i][j][k], Not(x[i][j][m])))

    #Restriccion 3: Cada elemento de la columna es distinto
    for i in range(n):
        for k in range(n):
            for j in range(n-1):
                for m in range(j+1,n):
                    solver.add(Implies(x[i][j][k], Not(x[i][m][k])))

    #Restriccion 4: Analogamente, cada fila debe tener numeros distintos
    for j in range(n):
        for k in range(n):
            for i in range(n-1):
                for m in range(i+1,n):
                    solver.add(Implies(x[i][j][k], Not(x[m][j][k])))

    if celdas:
        for celda in celdas:
            i = int(celda[0]-1)
            j = int(celda[1]-1)
            k = int(celda[2]-1)
            solver.add(x[i][j][k] == True)

```

```
return solver , x

n = 4
celdas = [[1 ,1 ,2] , [4 ,1 ,3] , [3 ,2 ,4] ,[2 ,3 ,1] , [3 ,4 ,3]]

solver , variables = SAT_Solver(n, celdas)
if solver.check() == sat:
    for i in range(n):
        for j in range(n):
            for k in range(n):
                if solver.model().evaluate(variables [ i ][ j ][ k ]):
                    print(f"({i+1},{j+1}) - tiene - el - numero - {k+1}")
    print("Satisfacible")
else:
    print("No-satisfacible")
```

Luego, el output del código anterior:

```
(1, 1) tiene el numero 2
(1, 2) tiene el numero 1
(1, 3) tiene el numero 3
(1, 4) tiene el numero 4
(2, 1) tiene el numero 4
(2, 2) tiene el numero 3
(2, 3) tiene el numero 1
(2, 4) tiene el numero 2
(3, 1) tiene el numero 1
(3, 2) tiene el numero 4
(3, 3) tiene el numero 2
(3, 4) tiene el numero 3
(4, 1) tiene el numero 3
(4, 2) tiene el numero 2
(4, 3) tiene el numero 4
(4, 4) tiene el numero 1
Satisfacible
```