PROYECTO LÓGICA PARA CIENCIAS DE LA COMPUTACIÓN

Juan Sebastian Pedraza Guevara,
Tomas Felipe Patiño
Luceth Caterine Argote Radillo
Profesor: Daniel Felipe Bojaca
Fecha: Mayo/2023

I. Resumen

EL siguiente problema se puede definir como una cuadricula de tamaño n x n en la cual en cada columna y fila están indicados una secuencia de números o un numero solo. Es decir, existen dos casos el primero si es un numero solo indica el número exacto de casillas que deben ser rellenadas en esa fila y columna. El otro caso una secuencia de números en el que el primer elemento es la cantidad de cuadro seguidos posibles a rellenar hasta un espacio y así con todos los elementos de la secuencia.

Se usará una cuadricula de tamaño 5x5, con cualquiera de los posibles casos en la figura 1 .

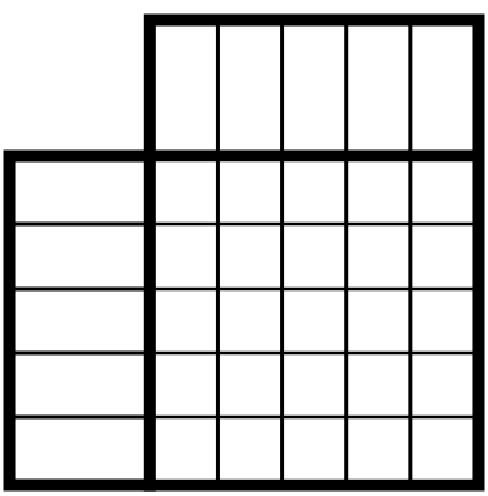

figura 1

II. Ejemplo

Para el ejemplo vamos a trabajarla figura dos en esta las primeras filas y la primera columna ya que se trabajan todas las posibles entradas (secuencia de numero o numero solo). Observe que para las filas y columnas se trabaja del mismo modo, pero cambiando la orientación en la que se toman las casillas (Horizontal a vertical).

Para la fila 1 tenemos indicado el numero 2 lo que significa que en esa fila vamos a rellenar solo dos casillas consecutivas. Una interpretación valida seria la siguiente:

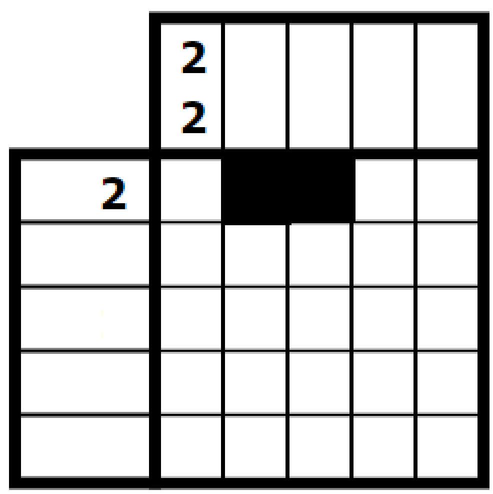

Figura 2

Para la fila dos tenemos la secuencia 2-2. Esto genera que vamos a rellenar solo una casilla y luego dejaremos mínimo un espacio hasta rellenar la otra casilla como lo indica el segundo numero de la secuencia. Una interpretación valida seria la siguiente:

El presente documento corresponde a un proyecto de lógica proposicional de ciencias de la computación presentado en la Universidad del Rosario 2023/I.

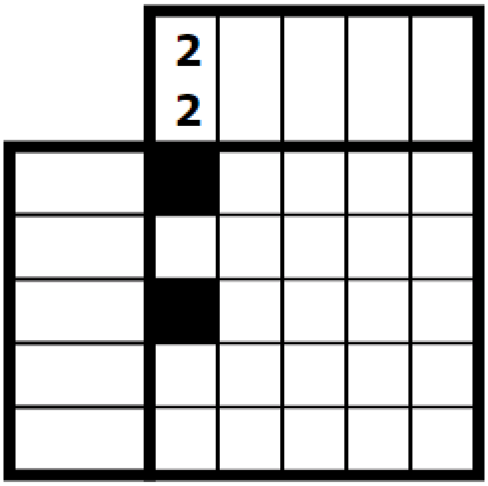

Figura 3

III. Introducción

Para la resolución de este problema orientada por la rama de lógica para ciencias de la computación vamos a definir ciertas letras proposicionales que nos van a ayudar a representar todas las posibilidades en la cuadricula de tamaño 5x5. Observe que el tablero al ser de tamaño 5x5 tiene dentro 25 para cada casilla representamos 2 letras una si esta rellena y otra si por el contrario no.

Por ejemplo
letra1: La casilla uno esta rellena
letra2: La casilla uno carece de relleno
letras3: La casilla dos está rellena
.....
Letras50: La casilla veinticinco no está rellena

En el caso de este problema podemos describir las letras proposicionales con descriptores de la forma:

$Ren_{(n,m)}$

De esta forma en el tablero 5x5 solo se tendrían la mitad de las letras dado que Ren indica si la casilla esta rellena y [n, m] la posición en el tablero. Cuando una casilla no este rellena en [n, m] su letra proposicional será:

$\neg Ren_{(n,m)}$

IV. Reglas en lenguaje no proposicional

Dado que vamos a hallar la solución para todos los nanogramas 5x5 con condiciones iniciales variables. las reglas no van a estar singularizadas y se van a implementar según el nanograma. Sin embargo se van a dar todas las posibles reglas para así según el caso implementar las necesarias.

Hay que tener en cuenta que en todos los casos la cantidad de reglas son 10 la variación se va a presentar según el numero indicado como condición inicial.

Todas las posibles reglas verbales para un nanograma 5x5 serian las siguientes:

1. La Fila n no tiene casillas rellenas.
2. La fila n tiene una única casilla rellena.
3. La fila n tiene dos casillas rellenas consecutivas.
4. La fila n tiene tres casillas rellenas consecutivas.
5. La fila n tiene cuatro casillas rellenas consecutivas.
6. La fila n tiene cinco casillas rellenas consecutivas.
7. La fila n tiene dos casillas rellenas con mínimo un espacio entre ambas.
8. La fila n tiene una casilla rellena hasta mínimo un espacio y luego dos rellenas consecutivas.
9. La fila n tiene una casilla rellena consecutiva hasta mínimo un espacio, luego tres rellenas consecutivas.
10. La fila n tiene dos casillas rellenas consecutivas hasta mínimo un espacio y luego una rellena.
11. La fila n tiene dos casillas rellenas consecutivas luego un espacio y 2 rellenas.
12. La fila n tiene tres casillas rellenas consecutivas luego un espacio y una rellena.
13. La fila n tiene una casilla rellena un espacio luego una rellena un espacio y finalmente una rellena.
14. La columna n no tiene casillas rellenas.
15. La columna n tiene una única casilla rellena.
16. La columna n tiene dos casillas rellenas consecutivas.
17. La columna n tiene tres casillas rellenas consecutivas.
18. La columna n tiene cuatro casillas rellenas consecutivas.
19. La columna n tiene cinco casillas rellenas consecutivas.
20. La columna n tiene dos casillas rellenas con mínimo un espacio entre ambas.
21. La columna n tiene una casilla rellena hasta mínimo un espacio y luego dos rellenas consecutivas.
22. La columna n tiene una casilla rellena consecutiva hasta mínimo un espacio, luego tres rellenas consecutivas.
23. La columna n tiene dos casillas rellenas consecutivas hasta mínimo un espacio y luego una rellena.
24. La columna n tiene dos casillas rellenas consecutivas luego un espacio y 2 rellenas.
25. La columna n tiene tres casillas rellenas consecutivas luego un espacio y una rellena.
26. La columna n tiene una casilla rellena un espacio luego una rellena un espacio y finalmente una rellena.

V. Reglas en lenguaje no proposicional para un Nanograma especifico

Dado que el objetivo del problema es lograr solucionar cualquier Nanograma de tamaño 5x5 se tienen en cuenta todos los posibles casos que se pueden dar. Pero para solucionar un Nanogama se tienen en cuenta las condiciones iniciales dadas que es una por cada fila y columna para un total de diez reglas estas van a estar contempladas en las veintiséis descritas antes. Notese que para las filas y columnas son las misma solo cambia la orientación.

Como se había aclarado en un nanograma de tamaño 5x5 se tiene 10 reglas estas siempre van a estar dentro de las 26 reglas predefinidas. Para hacerlo mas claro supongamos el siguiente nanograma en la figura4:

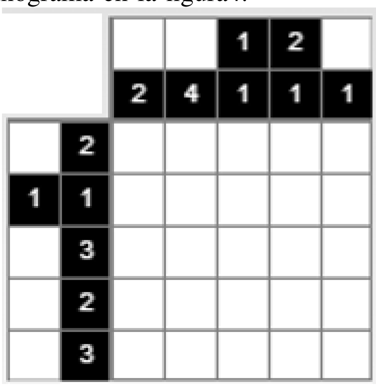

Figura 4

Para este caso las reglas serian las siguientes:

1. La fila cero debe tener únicamente dos casillas rellenas consecutivas.
2. La fila uno debe tener únicamente dos casillas rellenas NO consecutivas.
3. La fila dos debe tener únicamente tres casillas rellenas consecutivas.
4. La fila tres debe tener únicamente dos casillas rellenas consecutivas.
5. La fila cuatro uno debe tener únicamente tres casillas rellenas consecutivas.
6. La columna cero debe tener únicamente dos casillas rellenas consecutivamente.
7. La columna uno debe tener únicamente cuatro casillas rellenas consecutivamente.
8. La columna dos debe tener únicamente dos casillas rellenas NO consecutivamente.
9. La columna tres debe tener dos casillas rellenas hasta mínimo un espacio y luego una rellena .
10. La columna cuatro debe tener una única casilla rellenas consecutivamente.

Si se miran las veintiséis reglas del problema vemos que cada una de las diez que están en este caso tienes su representación.

VI. Letras / Descriptores Proposicionales

Se usarán descriptores proposicionales de esta manera:

$Ren_{(n,m)}$ = La casilla (n, m) esta rellena
$\neg Ren_{(n,m)}$ = La casilla (n, m) NO esta rellena
Con $n \wedge m \in [0, 1, 2, 3, 4]$

De este modo:
Ren (0, 0) = Casilla (0, 0) esta rellena
$\neg Ren(4, 2)$ = Casilla (4, 2) NO esta rellena

VII. Reglas en Formula

1. $\bigwedge_{i=0}^{4}\left(\neg Ren_{(n,i)}\right)$

2. $\bigvee_{i=0}^{4}\left[Ren_{(n,i)} \bigwedge_{j=o,j\neq i}^{4}\left(\neg Ren_{(n,j)}\right)\right]$

3. $\bigvee_{i=0}^{3}\left[Ren_{(n,i)} \wedge Ren_{(n,i+1)} \bigwedge_{j=o,j\neq i,i+1}^{4}\left(\neg Ren_{(n,j)}\right)\right]$

4. $\bigvee_{i=0}^{2}\left[Ren_{(n,i)} \wedge Ren_{(n,i+1)} \wedge Ren_{(n,i+2)} \bigwedge_{j=o,j\neq i,i+1,i+2}^{4}\left(\neg Ren_{(n,j)}\right)\right]$

5. $\bigvee_{i=0}^{1}\left[Ren_{(n,i)} \wedge Ren_{(n,i+1)} \wedge Ren_{(n,i+2)} \wedge Ren_{(n,i+3)} \bigwedge_{j=o,j\neq i,i+1,i+2,i+3}^{4}\left(\neg Ren_{(n,j)}\right)\right]$

6. $\bigwedge_{i=0}^{4}\left(Ren_{(n,i)}\right)$

7. $\left[\bigvee_{i=2}^{4}\left(Ren_{(n,0)} \wedge Ren_{(n,i)} \bigwedge_{j\neq 0,ij=0}^{4}\left(\neg Ren_{(n,j)}\right)\right)\right] \vee \left[\bigvee_{i=3}^{4}\left(Ren_{(n,1)} \wedge Ren_{(n,i)} \bigwedge_{j\neq 0,ij=0}^{4}\left(\neg Ren_{(n,j)}\right)\right)\right] \vee \left[Ren_{(n,2)} \wedge Ren_{(n,4)}\left(\bigwedge_{j\neq 2,4,ij=0}^{4}\left(\neg Ren_{(n,j)}\right)\right)\right]$

8. $\left[\bigvee_{i=2}^{3}\left(Ren_{(n,0)} \wedge Ren_{(n,i)} \wedge Ren_{(n,i+1)}\right) \bigwedge_{j\neq i,i+1 j=1}^{4}\left(\neg Ren_{(n,j)}\right)\right] \vee \left[Ren_{n,1} \wedge Ren_{n,3} \wedge Ren_{n,4} \bigwedge_{j\neq 1,3,4 j=0}^{4}\left(\neg Ren_{(n,j)}\right)\right]$

9. $\left(Ren_{(n,0)} \wedge Ren_{(n,2)} \wedge Ren_{(n,3)} \wedge Ren_{(n,4)} \wedge \neg Ren_{(n,1)}\right)$

10. $\Bigg[\bigvee_{i=3}^{4}\Bigg(Ren_{(n,0)} \wedge Ren_{(n,1)} \wedge Ren_{(n,i)} \bigwedge_{j=2,j\neq i}^{4}\left(\neg Ren_{(n,j)}\right)\Bigg] \vee \left(Ren_{(n,0)} \wedge Ren_{(n,2)} \wedge Ren_{(n,4)} \wedge \neg Ren_{(n,0)} \wedge \neg Ren_{(n,3)}\right)$

11.$\neg Ren_{(n,2)} \bigwedge_{j=0,j\neq 2}^{4} \left(Ren_{(n,j)}\right)$

12.$\neg Ren_{(n,3)} \bigwedge_{j=0,j\neq 3}^{4} \left(Ren_{(n,j)}\right)$

13.$Ren_{(n,1)} \wedge Ren_{(n,3)} \bigwedge_{j=0,j\neq 1,3}^{4} \left(\neg Ren_{(n,j)}\right)$

14. $\bigwedge_{i=0}^{4} \left(\neg Ren_{(i,n)}\right)$

15. $\bigvee_{i=0}^{4} \left[Ren_{(i,n)} \bigwedge_{j=o,j\neq i}^{4} \left(\neg Ren_{(j,n)}\right)\right]$

16. $\bigvee_{i=0}^{3} \left[Ren_{(i,n)} \wedge Ren_{(i+1,n)} \bigwedge_{j=o,j\neq i,i+1}^{4} \left(\neg Ren_{(j,n)}\right)\right]$

17. $\bigvee_{i=0}^{2} \left[Ren_{(i,n)} \wedge Ren_{(i+1,n)} \wedge Ren_{(i+2,n)} \bigwedge_{j=o,j\neq i,i+1,i+2}^{4} \left(\neg Ren_{(j,n)}\right)\right]$

18. $\bigvee_{i=0}^{1} \left[Ren_{(i,n)} \wedge Ren_{(i+1,n)} \wedge Ren_{(i+2,n)} \wedge Ren_{(i+3,n)} \bigwedge_{j=o,j\neq i,i+1,i+2,i+3}^{4} \left(\neg Ren_{(j,n)}\right)\right]$

19. $\bigwedge_{i=0}^{4} \left(Ren_{(i,n)}\right)$

20. $\left[\bigvee_{i=2}^{4} \left(Ren_{(0,n)} \wedge Ren_{(i,n)} \bigwedge_{j\neq 0,ij=0}^{4} \left(\neg Ren_{(j,n)}\right)\right)\right] \vee \left[\bigvee_{i=3}^{4} \left(Ren_{(1,n)} \wedge Ren_{(i,n)} \bigwedge_{j\neq 0,ij=0}^{4} \left(\neg Ren_{(j,n)}\right)\right)\right] \vee \left[Ren_{(2,n)} \wedge Ren_{(4,n)} \left(\bigwedge_{j\neq 2,4,ij=0}^{4} \left(\neg Ren_{(j,n)}\right)\right)\right]$

21. $\left[\bigvee_{i=2}^{3} \left(Ren_{(0,n)} \wedge Ren_{(i,n)} \wedge Ren_{(i+1,n)}\right) \bigwedge_{j\neq i,i+1j=1}^{4} \left(\neg Ren_{(j,n)}\right)\right] \vee \left[Ren_{(1,n)} \wedge Ren_{(3,n)} \wedge Ren_{(4,2)} \bigwedge_{j\neq 1,3,4j=0}^{4} \left(\neg Ren_{(n,j)}\right)\right]$

22. $\left(Ren_{(0,n)} \wedge Ren_{(2,n)} \wedge Ren_{(3,n)} \wedge Ren_{(4,n)} \wedge \neg Ren_{(1,n)}\right)$

23. $\left[\bigvee_{i=3}^{4} \left(Ren_{(0,n)} \wedge Ren_{(1,n)} \wedge Ren_{(i,n)} \bigwedge_{j=2,j\neq i}^{4} \left(\neg Ren_{(j,n)}\right)\right)\right] \vee \left(Ren_{(0,n)} \wedge Ren_{(2,n)} \wedge Ren_{(4,n)} \wedge \neg Ren_{(0,n)} \wedge \neg Ren_{(3,n)}\right)$

24.$\neg Ren_{(2,n)} \bigwedge_{j=0,j\neq 2}^{4} \left(Ren_{(j,n)}\right)$

25.$\neg Ren_{(3,n)} \bigwedge_{j=0,j\neq 3}^{4} \left(Ren_{(j,n)}\right)$

26.$Ren_{(1,n)} \wedge Ren_{(3,n)} \bigwedge_{j=0,j\neq 1,3}^{4} \left(\neg Ren_{(j,n)}\right)$

Notese que desde la regla trece las reglas solo varian el orden de los indices de la casilla.

VIII. Representación Gráfica

Se probaron cinco diferentes algoritmos, con dos posibles casos de Nanogramas cada uno con la unión de las diez reglas y sus respectivas restricciones iniciales, las cuales generan la formula, luego se obtuvieron estos datos:

Algoritmos	Caso 1	Caso2
SATtabla	78.489,77ms	4 reglas en 4.150,08 ms
Tableaux	2.170,55 ms	1.000,22 ms
DPLL	2,03 ms	4,94 ms
WalkSAT	ERROR	ERROR
Minsat22	ms	ms

Dados los datos obtenidos luego de la comparación de los algoritmos con los diferentes casos, obtenemos que SATtabla es un algoritmo algo lento pero puede obtenerse una respuesta con este, luego de una larga espera. Tableaux en este problema tiene un desempeño notorio en tiempos, si lo comparamos con DPLL no hay gran diferencia. WalkSAT al ser un algoritmo donde se desarrolla mediante una probabilidad a elegir que hace que el proceso cambie, en este caso no corrimos con suerte y luego de esperar mas de 20 minutos no arrojaba aun solución alguna. Minsat22 es el algoritmo mas rápido y útil dado los tiempos mostrados.

IX. Conclusión

En virtud de lo presentado hasta aquí, es claro que el desarrollo del problema trabajado tiene sus complicaciones, pero que con un buen análisis lógico es posible desarrollar una solución. Es por eso que resulta imprescindible que las reglas de generen de manera adecuada y se adapten según el Nanograma, de modo que cuando lleguemos al momento de encontrar la solución mediante los SATsolvers no haya mas condiciones de las mínimas necesarias para hallar solución al problema .