Programación Declarativa

Sudoku

Leandro González Montesino

Grupo 411

Facultad de Matemática y Computación

La Habana, Cuba

27 de mayo 2019

1- Resumen:

En el siguiente informe nos encontraremos con la descripción de la problemática a resolver. Métodos utilizados y breve explicación desde un enfoque más práctico.

2- Breve descripción del Problema:

Teniendo como entrada un conjunto de nueve nonominós, la ejecución del programa permitirá dar solución al sudoku que representan. Para ello debe verificarse que los nonominós se pueden "encajar" (o agrupar) para formar un cuadrado de 9x9, que representa el sudoku. Una vez obtenido el Sudoku se procede a encontrar su solución. Para resolver el sudoku debe desechar la estrategia que convierte un Sudoku en listas de listas. Como parte de la solución del problema tiene que ser posible observar tanto los nonominós como los sudokus en consola.

3- Definición objetiva:

La situación que se presentará posteriormente a modelar estará basa en los pasos siguiente, manteniendo todos los requisitos de la orientación.

1. Primeramente, dado los 9 nonominós conformar un tablero de sudoku.
2. Tratar de encontrar una solución para el sudoku.

La implementación del suceso fue concebida en el lenguaje de programación declarativa Haskell. Llevándose todo el proceso puramente en el mismo, con la posibilidad de correr en cualquier sistema operativo con WinGHCi u otro compilador de Haskell.

4- Ideas Principales:

Definición de nonominós: `nonominos :: Int -> [[[Int]]]`

Ejemplo:

```
nonominos 1 = [[[3],[5]],[[0,0,0,1,0],[-1,0,0,0,-1],[-1,-1,0,-1,-1]]]
nonominos 2 = [[[2],[5]],[[-1,2,0,3,0],[0,0,0,9,0]]]
nonominos 3 = [[[4],[4]],[[6,-1,-1,-1],[0,0,-1,0],[8,0,0,0],[-1,-1,4,-1]]]
nonominos 4 = [[[3],[5]],[[0,0,0,8,0],[-1,-1,-1,0,0],[-1,-1,-1,0,0]]]
nonominos 5 = [[[5],[2]],[[0,0],[0,0],[3,0],[4,0],[0,-1]]]
nonominos 6 = [[[6],[4]],[[-1,-1,-1,7],[-1,-1,5,0],[-1,2,0,-1],[-1,9,-1,-
1],[-1,1,-1,-1],[0,0,-1,-1]]]
nonominos 7 = [[[5],[3]],[[-1,4,0],[-1,6,0],[0,0,0],[-1,-1,0],[-1,-1,0]]]
nonominos 8 = [[[3],[3]],[[8,0,5],[7,0,9],[3,6,0]]]
nonominos 9 = [[[4],[3]],[[-1,0,0],[-1,0,0],[-1,0,0],[0,7,0]]]
```

Definición de tablero: `tablero :: [[Int]]`

Ejemplo

```
examplePuzzle = [[5, 3, 0,  0, 7, 0,  0, 0, 0],
                 [6, 0, 0,  1, 9, 5,  0, 0, 0],
                 [0, 9, 8,  0, 0, 0,  0, 6, 0],

                 [8, 0, 0,  0, 6, 0,  0, 0, 3],
                 [4, 0, 0,  8, 0, 3,  0, 0, 1],
                 [7, 0, 0,  0, 2, 0,  0, 0, 6],

                 [0, 6, 0,  0, 0, 0,  2, 8, 0],
                 [0, 0, 0,  4, 1, 9,  0, 0, 5],
                 [0, 0, 0,  0, 8, 0,  0, 7, 0]]
```

La idea presentada es generar todas las permutaciones de los nonominós y una vez con ellas intentar darle una solución al tablero generado de ser posible.

La solución del sudoku se busca de manera recursiva de esta manera:

```
solutions' :: [Location] -> Board -> [Board]
    solutions' []     b = [b]
    solutions' (x:xs) b = concatMap (solutions' xs) candidateBoards
```