Programación Declarativa
Sudoku Nonomino

Alberto González Rosales
Grupo C-411

1. Problema

El objetivo de este trabajo es resolver un *sudoku nonominó* utilizando el lenguaje de programación *Haskell*. Un *sudoku nonominó* es una versión más general del *sudoku clásico*. En este tipo de *sudokus* las reglas son las mismas que en el *sudoku clásico* : se debe rellenar el tablero con números entre 1 y 9, no debe existir ningún número repetido en ninguna fila ni columna y tampoco en ninguna de las 9 regiones definidas en el tablero. La diferencia radica en que las regiones de un *sudoku clásico* son cuadrados de $3 * 3$, mientras que las regiones del *sudoku nonominó* son regiones conexas de 9 cuadrados de $1 * 1$.

Nuestro problema específico parte del hecho de que se nos da como entrada un conjunto de 9 *nonominós* y tenemos primeramente que verificar que exista una forma de ubicar las piezas *nonominós* sobre el tablero del *sudoku*(un cuadrado de dimensiones $9 * 9$). Una vez determinada una posible ubicación de las piezas sobre el tablero procedemos entonces a resolver el *sudoku nonominó* resultante.

2. Definiciones e implementación

La forma de representar los *nonominós* que se consideró más conveniente fue mediante una clase que tuviese dos campos : un identificador numérico y una lista de tuplas de la forma $((x, y), val)$, o sea, una posición y un valor asociado a esa posición. El valor 0 significa que en esa posición aún no hay ningún número asignado. El identificador numérico sirve para asociar un índice a un *nonominó* para que sea más fácil identificarlo, lo cual es útil a la hora de mostrarlos por consola. Los valores (x, y) son las posiciones relativas del *nonominó* asumiendo que el cuadrado más arriba y más a la izquierda es el $(0, 0)$.

La forma de representar un *sudoku* es una lista de tuplas de la forma $((x, y), val)$ donde (x, y) es una posición en la matriz de $9 * 9$ y val es el valor asociado a esa posición.

La forma de resolver el problema planteado consiste en dada una lista de *nonominós* pasarla como parámetro a un método que es el encargado de comprobar si existe una distribución válida de estos y, en caso de encontrarla, pasa esta distribución a otro método que se encarga de resolver el *sudoku*.

3. Ideas principales para resolver el problema

El proceso de solución del *sudoku nonominó* consiste en dos pasos:

3.1. Ordenar los *nonominós*

Primeramente hay que comprobar que exista una forma de ubicar los *nonominós* de forma tal que cubran todo el tablero y no exista en esa distribución ningún número repetido en ninguna fila o columna. Estamos asumiendo que en un mismo *nonominó* no existen números repetidos.

Para esto vamos a seguir el criterio de que dado un orden de los *nonominós*, el primero en la lista debe ser el que cubra la posición $(0, 0)$ en la matriz. El siguiente en la lista debe cubrir la posición más arriba y más a la izquierda que aún no esté cubierta y así sucesivamente. Sabremos si el orden que estamos comprobando es válido si cuando estén ubicados todos los *nonominós* no quedó ninguna posición vacía.

La idea es comprobar si se puede ubicar alguna de las permutaciones de la lista de *nonominós*, si es posible ubicar los *nonominós* en este orden entonces se pasa a resolver el *sudoku* que quedó conformado. En caso de que no exista ninguna distribución válida para conformar un *sudoku* determinamos que no hay solución.

3.2. Resolver el *sudoku*

Para resolver el *sudoku* utilizaremos un método recursivo que va a comprobar si existe solución de *nonominó* en *nonominó*. El método recibe, a grandes rasgos, el *nonominó* actual, la posición actual dentro de este *nonominó*, el número que se quiere poner en esta posición y todas las posiciones donde se ha puesto algún número.

Los casos base son: si revisamos todos los *nonominós* entonces pudimos resolverlo, si el número que queremos probar en esta posición es mayor o igual a 10 entonces no hay solución. De lo contrario comprobamos si el número que queremos ubicar en la posición actual no genere contradicciones en su fila, columna y *nonominó*. Si no hay contradicción lo ubicamos y resolvemos recursivamente el *sudoku* en la siguiente posición del *nonominó*; si hay contradicción entonces se prueba con el número actual incrementado en uno.

4. Función Main

La función que da inicio al programa lee un fichero "*input.in*" que contiene una lista de *nonominós* en el formato explicado anteriormente. Esta lista de "*Strings*" la convierte a *nonominós* y la pasa como parámetro al método principal llamado "*solution*". Este método devuelve una tupla donde el primer elemento es la distribución de los *nonominós* en el tablero y el segundo es la solución del *sudoku*. En caso de no existir solución devuelve dos listas vacías. Posteriormente se muestran en la consola los resultados obtenidos en forma de matrices de $9 * 9$.

Para ejecutar el programa basta con configurar en el archivo "*input.in*" adjunto, la lista de *nonominós* siguiendo el formato especificado anteriormente y luego ejecutar el comando "*main*" en la consola una vez cargado el módulo "*nonominos.hs*".

4.1. Código para iniciar el programa

```
Prelude> :l nonominos.hs
[1 of 1] Compiling Main             ( nonominos.hs, interpreted )
Ok, one module loaded.
*Main> main
```

4.2. Código de la función Main

```
main = do
    inp <- readFile "input.in"
    let x = map readNonomino (lines inp)

    let sol = solution x

    let setting = fst sol
    let filled = snd sol

    print "sudoku␣distribution:"
    printList (convertToIds (setting) )

    print "solution:"
    printList (convertToVals (filled) )
```

4.3. Salida ejemplo

```
"sudoku␣distribution:"
[1,1,1,1,1,2,2,2,2]
[3,1,1,1,2,2,2,2,2]
[3,3,1,3,4,4,4,4,4]
[3,3,3,3,6,7,7,4,4]
[5,5,3,6,6,7,7,4,4]
[5,5,6,6,7,7,7,9,9]
[5,5,6,8,8,8,7,9,9]
[5,5,6,8,8,8,7,9,9]
[5,6,6,8,8,8,9,9,9]
"solution:"
[9,5,7,1,8,2,6,3,4]
[6,2,3,4,1,8,7,9,5]
[7,1,6,2,5,3,4,8,9]
[8,3,5,9,7,4,1,2,6]
[2,9,4,5,3,6,8,1,7]
[1,8,2,6,9,7,5,4,3]
[3,7,9,8,4,5,2,6,1]
[4,6,1,7,2,9,3,5,8]
[5,4,8,3,6,1,9,7,2]
```