Proyecto de Haskell
Hidato

Loraine Monteagudo García - C411
l.monteagudo@estudiantes.matcom.uh.cu

Jorge Daniel Valle Díaz - C411
jorge.valle@estudiantes.matcom.uh.cu

1. Componentes básicas del juego

El Sudoku Hidato es un tablero con varios números ubicados en él y algunas casillas en blanco. El menor número siempre será 1 y el mayor coincidirá con la cantidad de casillas del tablero, ambos siempre estarán presentes en el tablero. El objetivo es rellenarlo con números consecutivos que se conectan horizontal, vertical o diagonalmente, formando así una secuencia de números que empieza con 1 que están inmediatamente conectados. Para cada número, su antecesor y su sucesor estará en alguna casilla adyacente.

Se representa el tablero como un tipo compuesto de 3 elementos: una lista compuesta por listas para representar las casillas, el menor número del tablero y el mayor. Las casillas están representadas con números enteros, usando la clase `Integer` de Haskell. La definición del tablero es:

```
data Board = Board {
  cells  ::  [[Integer]],
  minNum :: Integer,
  maxNum :: Integer
  } | Empty deriving (Show, Eq)
```

Donde `Empty` es un valor especial que se usa para representar cuando un tablero es inválido. A pesar de que se tiene el conocimiento de que existen otras estructuras en Haskell para representar la ausencia de valor, como `Maybe`, que tiene `Just a` para representar un valor del tipo genérico `a` y `Nothing` para la ausencia de este, se decidió modificar la definición `Board` con el objetivo de ganar en simplicidad y flexibilidad con los métodos definidos. Así, por ejemplo un tablero sabe como dibujarse incluso cuando toma el valor de `Empty`.

Asumiendo que los números usados en el juego son naturales mayores que 1 se usa el 0 para representar las casillas que están vacías. Como un tablero no es necesariamente rectangular se usa además -1 para representar las casillas en las que no se pueden poner ningún número. Por ejemplo, al tablero representado en la Figura 1 sería representado en Haskell con la siguiente definición:

```
sample = Board [
  [ 0, 33, 35,  0,  0],
  [ 0,  0, 24, 22,  0],
  [ 0,  0,  0, 21,  0,  0],
  [ 0, 26,  0, 13, 40, 11],
  [27,  0,  0,  0,  9,  0,  1],
  [-1, -1,  0,  0, 18,  0,  0],
```

[−1, −1, −1, −1, 0, 7, 0, 0],
[−1, −1, −1, −1, −1, −1, 5, 0]]
1 40

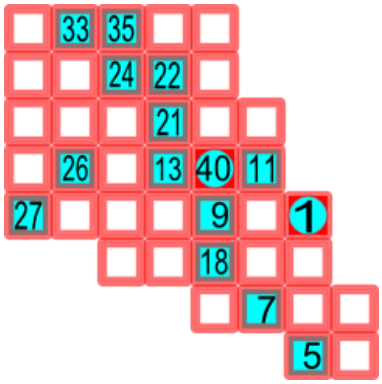

Figura 1: Un tablero de Hidato

2. Resolver Hidatos

Uno de los objetivos principales del proyecto es, teniendo como entrada un Hidato de cualquier forma, con algunas casillas anotadas y el menor y el mayor número señalados, ejecutar un programa que permitirá dar solución al tablero.

La función principal para dar solución a un Hidato se llama `solve` y recibe como parámetro un `Board` y como salida imprime el tablero solucionado.

La estrategia llevada a cabo para la generación de la solución del tablero fue un procedimiento recursivo, que a pesar de poder ser ineficiente garantiza la solución del mismo.

Se comienza buscando la posición del menor número, a partir de él se comenzará a generar la secuencia de números que se posicionarán en el tablero hasta llegar a poner el último. Se llega a la solución cuando todos los espacios vacíos son ubicados. Para comprobar esta condición se calcula inicialmente la cantidad de casillas vacías, y para cada nueva asignación esta cantidad se disminuye en 1. Cuando la cantidad de casillas vacías sea 0 entonces todas los números fueron puestos en el tablero y el método termina su ejecución.

Para cada número se analiza cada posición que puede tomar, se comprueban si se pueden poner arriba, abajo, a la derecha, a la izquierda o en alguna de las diagonales del número que fue su antecesor y se busca una solución en profundidad después de tomar una decisión con respecto a la ubicación del número. En el momento en que se encuentre una solución válida se dejan de comprobar otros caminos. A su vez, se intenta determinar lo más pronto posible cuando una solución deja de ser válida.

En el momento en que se descubre que una solución no es correcta se retorna el valor especial `Empty`. Las situaciones en las que se invalida un tablero son:

- Cuando se pone un número que no es el último en el tablero y todas las casillas alrededor están ocupadas. En este caso no es posible poner el siguiente número, lo que imposibilita la continuidad de la partida.
- Al ponerse un número se descubre que su sucesor está en el tablero y no está en ninguna de sus casillas adyacentes.

El método recursivo recibe además del tablero otros argumentos como un número y una posición, con el objetivo de que en ese llamado el número se ponga en la posición propuesta. Luego, el algoritmo general para resolver el Hidato es:

1. Si se pone el último número en el tablero, entonces ya se tiene una solución válida.

2. Si no se invalida la solución por uno de los casos expuestos anteriormente, entonces se continúa con la generación del tablero resulto.

3. Se pone el número actual en el tablero en la posición prefijada.

4. Se determina cuál es el siguiente número. Si el sucesor del último número no está en el tablero entonces este es el siguiente número a ubicar. En otro caso, se busca cuál es el siguiente número que no esté todavía ubicado.

5. Se determina la posición del antecesor del número que se decidió poner.

6. Se comprueban las casillas adyacentes a esa posición en las que se puede poner un número (aquellas que no estén ocupadas o estén invalidadas por estar fuera del rango del tablero) y se genera una solución poniendo el número elegido en esta posición.

7. Este procedimiento se repite para todos los números hasta tener el tablero completo.

3. Generación de Hidatos