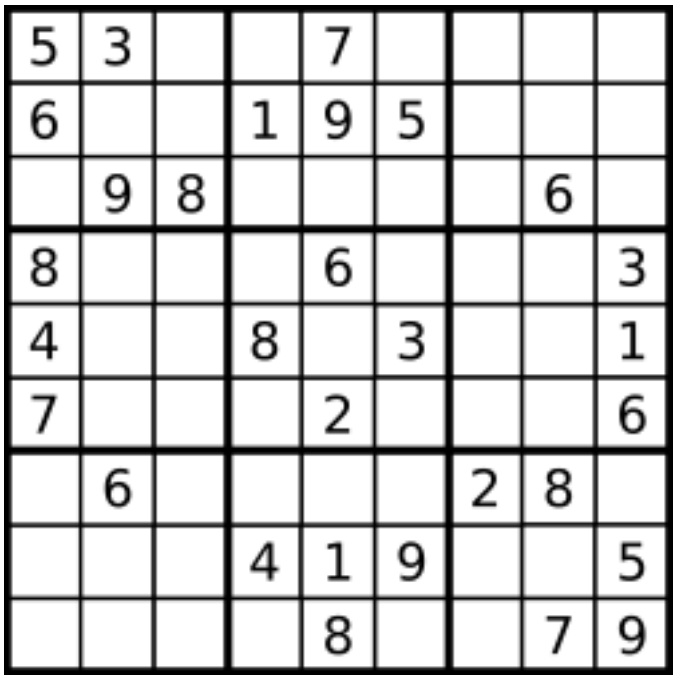

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

# Sudoku in Haskell

---

Диан Тодоров

61670

Функционално програмиране

1 семестър

доц. д-р. Тр. Трифонов

24.01.2016

# Въведение

Целта на този документ е да обясни реализацията на курсовия проект Судоку за избираемия предмет Функционално програмиране. Целият анализ се отнася за по програмния файл sudoku.hs. Судоку пъзелът представлява комбинаторна логическа игра, чиято цел е да се попълни таблица 9x9. Поставя се условие на всеки ред, колона и блок да има всички цифри от едно до девет. Под блок се има предвид участък от голямото судоку с размери 3x3. Когато се поставя условието за дъската тя е частично попълнена.

# Цели на проекта

- Правилно разрешаване на судоку пъзела.
- Генериране на решим судоку пъзел.

# Анализ на проблема

- Решение на пъзела - решава се чрез на така наречената техника backtracking. Интересно в имплементацията е, че се прави подобрение на наивния алгоритъм. Следващото попълване се прави като се избира от възможните свободни цифри, на от всички 1-9. Тоест спестяват се проверки.
- Генериране на пъзели

# Имплементация

1. ## Структура от данни

   За представянето на данните се използва списък от списъци от Int. Нарочно се използва Int, защото се работи с малки числа само от 1 до 9 .. Празните непопълнени полета се обозначават като нули. За четимост се въвежда следните типове:

   ```
   type Board = [[Int]] -> представлява самата судоку дъска
   type Row = [Int] -> списък от 9 елемента, представлява ред в судоку дъската
   type Column = [Int] -> списък от 9 елемента, представлява колона в судоку дъската
   type Block = [Int] -> списък от 9 елемента, представлява елементите от един 3x3 блок от дъската
   type Cell = (Int, Int) -> координати на поле от судоку дъската.
   ```

2. ## Функции за достъп

- Достъпване на колони

```
getColumns :: Board -> [Column] // транспонира дъската и така колоните стават редове
getColumn :: Int -> Board -> Column //  map на всеки ред с (!! j) за да върне j-тия стълб
```

- Достъпване на редици

```
getRows :: Board -> [Row] // връща самата дъска, която се дефинира като списък от редовето
си.
getRow ::Int -> Board -> Row // с помощта на оператора !! връща желания подред елемент в
списъка от редове.
```

- Достъпване на блокове

```
getBlocks :: Board -> [Block] // с помощта на редица операции връща съдържанието на
участък 3x3 от дъската
getBlockForCell :: Cell -> Board -> Block // връща списък с елементите на блок, към който
принадлежи съответното поле
```

- Достъпване на съдържанието на поле по координати

```
getCellContent ::  Cell -> Board -> Int // иозползва двойно прилагане на !! оператора
```

- Получаване на всички празни полета

```
getAllBlanks :: Board -> [Cell] // връща списък от координатите на всички празни полета
```

## 3. Валидатори

- isSolved

  Основна функция се пада containsAllNumbersInRange1. Тя проверява дали са налице всички числа в интервала 1-9. Проверката дали судоку пъзелът е решен се основава на прилагане именно на containsAllNumbersInRange1 върху списъците получени от извикването на функциите за достъп. Прави се конюнкция на валидирането дали има всички числа в отделно в колоните, редове и накрая блоковете. Тази конюнкция се опакова от метода isSolved, който се прилага над дъска и връща булева стойност.

- isInputBoardValid - > валидира съдържанието(дали елементите са между 1 и 9) и размерите на судоку пъзела

## 4. Генериране

## 5. Solver

- Методът `updateCell :: Cell -> Int -> Board -> Board` променя съдържанието на полето с координати, за дадени от аргумент a с тип Cell, като поставя нова стойност вътре

- Решението се прави със рекурентно извикване на solve. Това е функция, която изчерпатва всички възможности.
    i. има нарушения
    ii. решено е
    iii. в противен случай връща всички възможности
- Функцията updateBoards по подадена дъска с n празни полета връща списък от дъски с n - 1 на брой празни полета.
- избира се една от дъските

# Идеи за бъдещо подобрение

- **По умен избор за следващо попълване**
  Следващото попълнено поле да бъде това, което принадлежи на най-попълнялия 3x3 блок.
- **Моделиране на човешкия начин за решаване на судоку(propagation)**
  Използване на propagate функция, която сканира цялата судоку дъска и намира полета, които са с една единствена възможност за попълване. Избирателно първо се попълват тях, докато не останат свободни полета с повече от 1 възможности. След това се попълват и двете и рекурсията продължава.