



Scudoku – Sudoku in Scala

Martin Lehmann, Kristine Schaal – Scala Meetup 20151103

<http://www.meetup.com/Rhein-Main-Scala-Enthusiasts/events/225920553/>

Problem

Magic Square is nothing new...



Albrecht Dürer



Melencolia I
(1514)

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Sudoku is ... what, again?

9x9 grid (9 lines,
9 rows, 9 blocks)

Each cell is filled
with 1, 2, ... , 9

The problem is
NP complete.
There is a whole theory
of mathematics behind
this. For the brave:
Read [1]

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

No two cells in the
same line have the
same value

The number of valid
Sudoku solution grids for
the standard 9×9 grid was
calculated (...) in 2005 to be
6670903752021072936960

i.e. $> 6 \cdot 10^{21}$

No two cells in the
same row have the
same value

No two cells in the
same block have the
same value

Java solution

<https://github.com/MartinLehmann1971/scudoku/tree/master/workspace/judoku>

Overall structure

Package judoku:
Overall stuff

Package ...data:
Data structures

Package ...test:
JUnit Tests

```
judoku [Scudoku master]
├── src/main/java
│   ├── judoku
│   │   ├── Debug.java
│   │   ├── SudokuReverser.java
│   │   └── SudokuSolver.java
│   ├── judoku.checkers
│   │   ├── CheckCellsForDuplicateFixedValuesInBlockOrLineOrColumn.java
│   │   ├── CheckCellsForNoMorePotentialValuesAvailable.java
│   │   └── IConsistencyCheck.java
│   ├── judoku.data
│   │   ├── AbstractCellSet.java
│   │   ├── Block.java
│   │   ├── Cell.java
│   │   ├── Column.java
│   │   ├── Grid.java
│   │   ├── Line.java
│   │   └── OneToNine.java
│   ├── judoku.solvers
│   │   ├── IfValueFixedToLineOrColumnInsideABlockThenDeleteFromRest.java
│   │   ├── ISolvingAlgorithm.java
│   │   ├── LastPotentialValueInBlockOrLineOrColumn.java
│   │   └── OnlyOnePotentialValueLeftInCell.java
│   └── src/test/java
│       └── judoku.test
│           └── TestSudokuSolver.java
```

Main class (reads
grid from String)

Package ...checkers:
Consistency
checkers

Package ...solvers:
Solving algorithms

Data structures in judoku.data

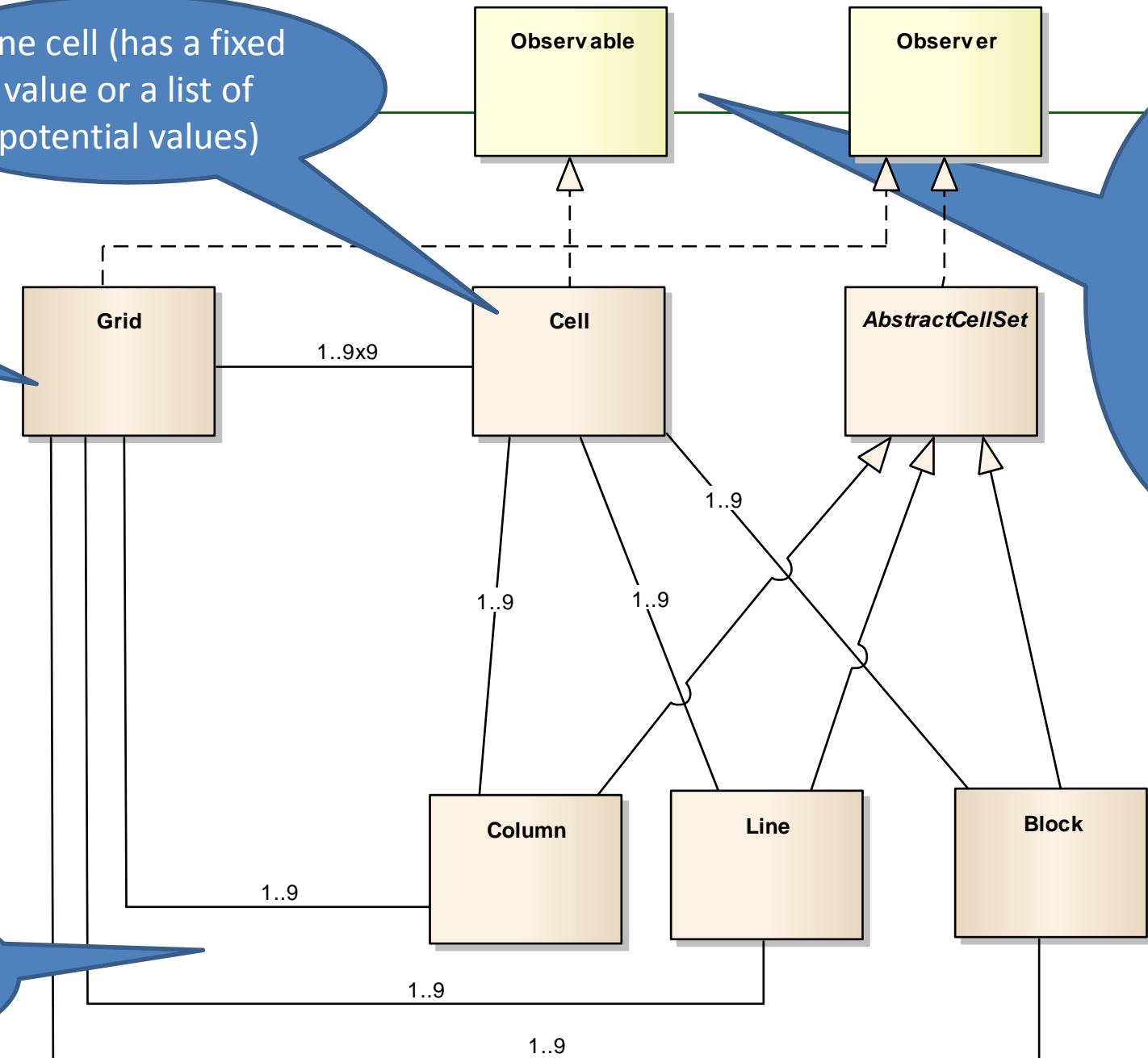
One cell (has a fixed value or a list of potential values)

Overall grid, used mainly for initial setup

Enumeration of values 1..9

«Enum»
OneToNine

One grid has 9 lines, 9 columns (=rows), 9 blocks



Changes in each cell can be observed ... in its line, column, block and in the grid

Algorithm

```
// Constraint propagation
```

```
if (cell.value is fixed) => eliminate the value  
    from all potential values in all cells
```

```
//MAIN
```

```
while (!solved) {  
    in its line, column and block
```

```
    apply each solver algorithm
```

```
    apply each checker
```

```
}
```

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

Algorithm

```
// Constraint propagation
```

```
if (cell.value is fixed) => eliminate the value  
    from all potential values in all cells  
    in its line, column and block
```

```
//MAIN
```

```
while (!solved) {  
    apply each solver algorithm  
    apply each checker  
}
```

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

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

5 is fixed
here

Algorithm

```
// Constraint propagation
```

```
if (cell.value is fixed) => eliminate the value  
from all potential values in all cells  
in its line, column and block
```

```
//MAIN  
while (!solved) {  
  apply each solver algorithm  
  apply each checker  
}
```

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

5 is **fixed**
here

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

5 can be
deleted in
line, column,
block

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

5 is
here

Algorithm

```
// Constraint propagation
if (cell.value is fixed) => eliminate the value
    ... in all cells ... its line, column and block

//MAIN
while (!solved) {
    apply each solver algorithm
    apply each checker
}

// Solver Algorithm: OnlyOnePotentialValueLeftInCell
if (cell has only one potential value left)
    => fix the cell.value

// Solver Algorithm: LastPotentialValueInBlockOrLineOrColumn
if (only one cell in a block/line/column where a value fits)
    => Fix the cell.value

// Solver Algorithm: IfValueFixedToLineOrColumnInside...
// ...ABlockThenDeleteFromRest
if (potential values are only possible in one line/column
    of a block - as others are fixed, for example)
    => delete this potential value from the rest of the line/column
```

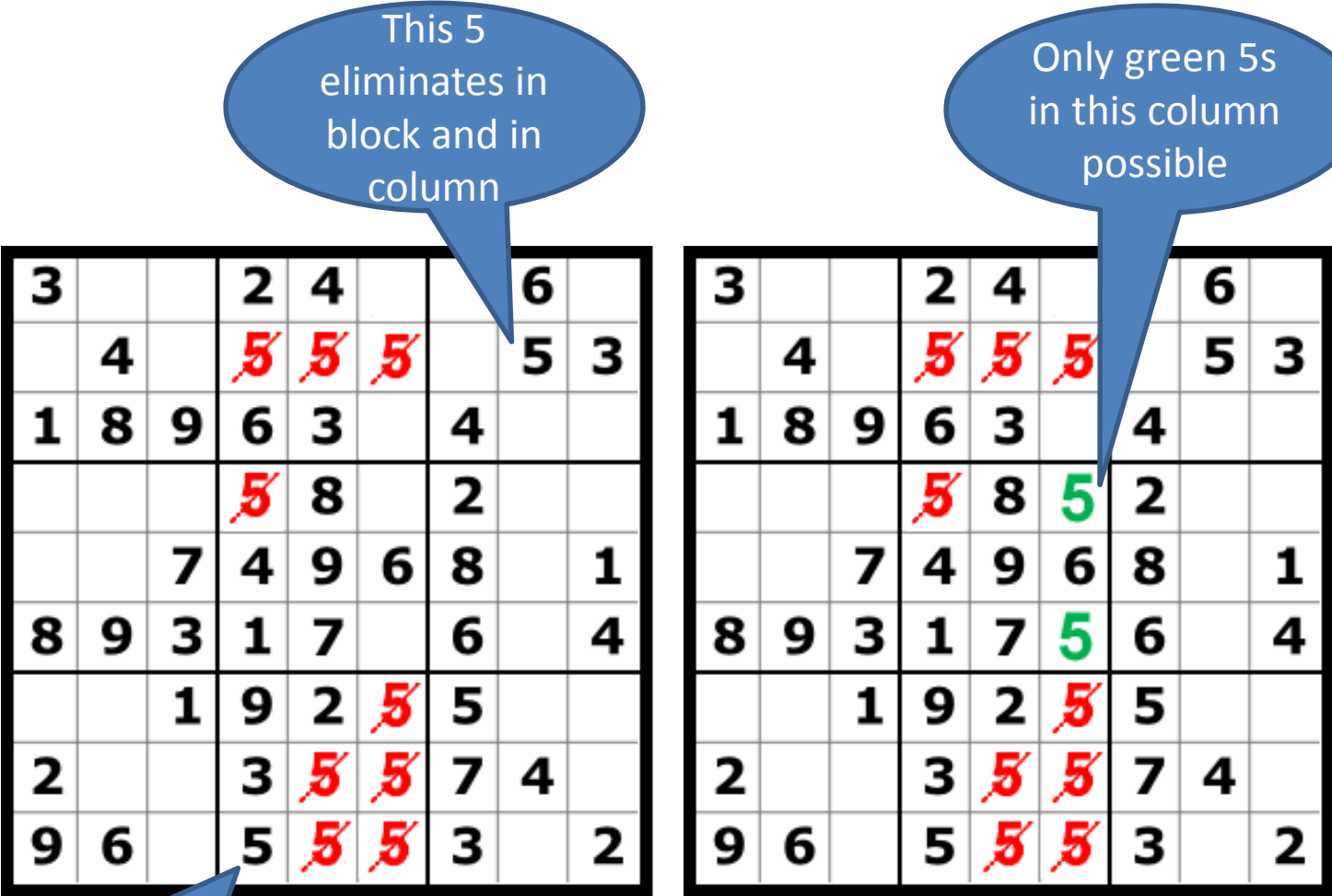
Example for IfValueFixedToLineOrColumnInsideABlockThenDeleteFromRest



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

This 5
eliminates in
block and in
column

Example for IfValueFixedToLineOrColumnInsideABlockThenDeleteFromRest



Example for IfValueFixedToLineOrColumnInsideABlockThenDeleteFromRest

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

This 5
eliminates in
block and in
column

This 5
eliminates in
block and in
column

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

Only green 5s
in this column
possible

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

So green 5's
eliminate the
rest of the column

Algorithm

```
// Constraint propagation
if (cell.value is fixed) => eliminate the value
    ... in all cells ... in its line, column and block

//MAIN
while (!solved) {
    apply each solver algorithm
    apply each checker
}

// Checker Algorithm:
// CheckCellsForNoMorePotentialValuesAvailable
fails, if any cell found which is not fixed but has no
    more potential values left

// Checker Algorithm:
// CheckCellsForDuplicateFixedValuesInBlockOrLineOrColumn
fails, if there are two cells in a line/column/block
    with the same fixed value
```

Scala solution

<https://github.com/MartinLehmann1971/scudoku/tree/master/workspace/scudoku>



**Begeisterung für die
anspruchsvollen Aufgaben unserer Kunden**