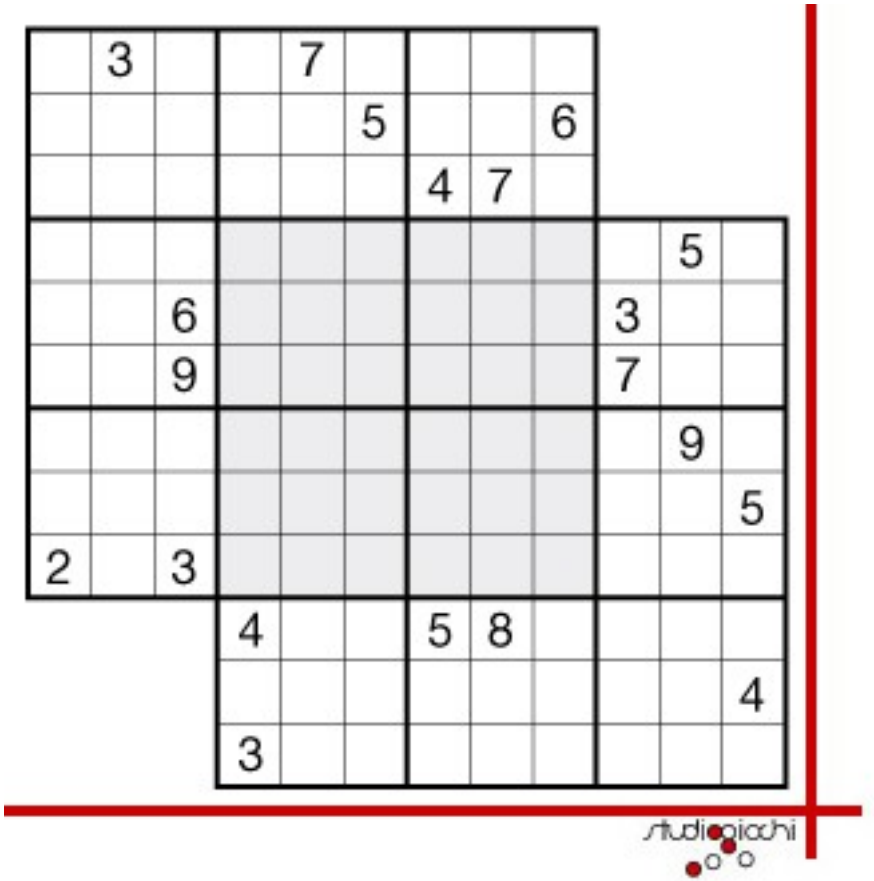
Advanced Sudoku with FunLog

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

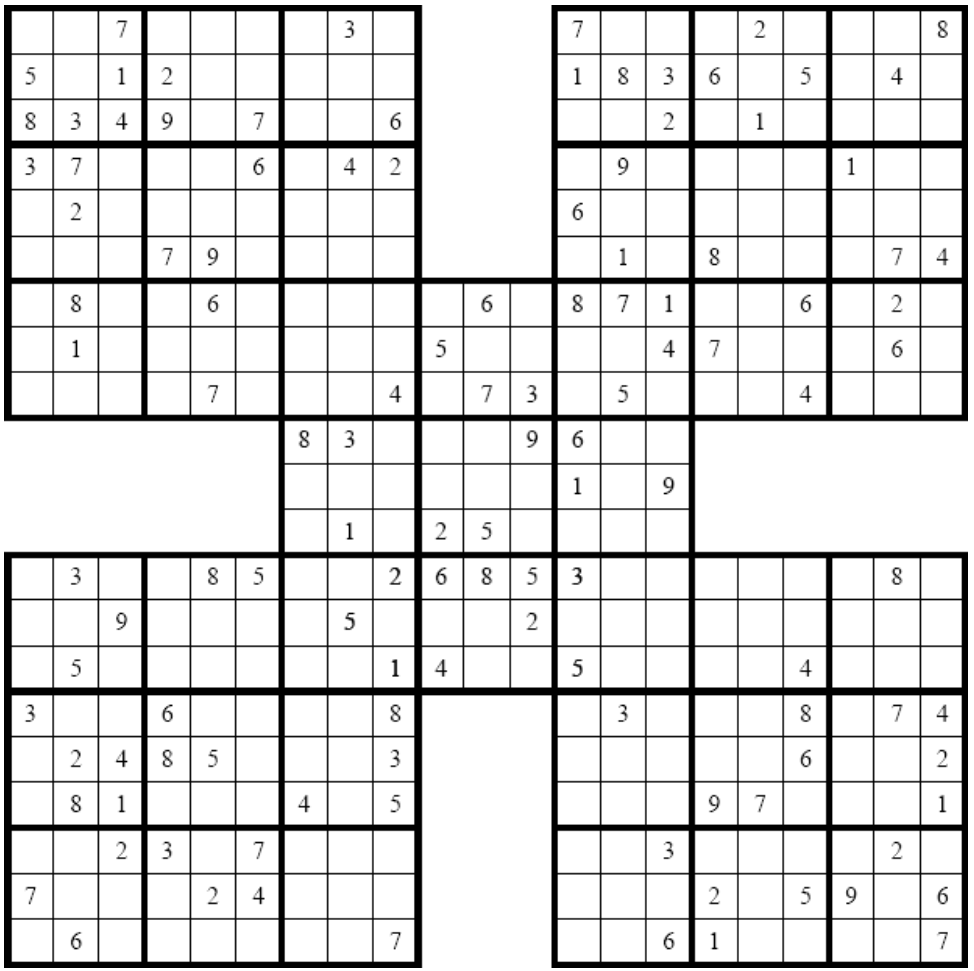
http://0.tqn.com/d/np/memory-booster-puzzles/037-1.jpg

Multi Sudoku



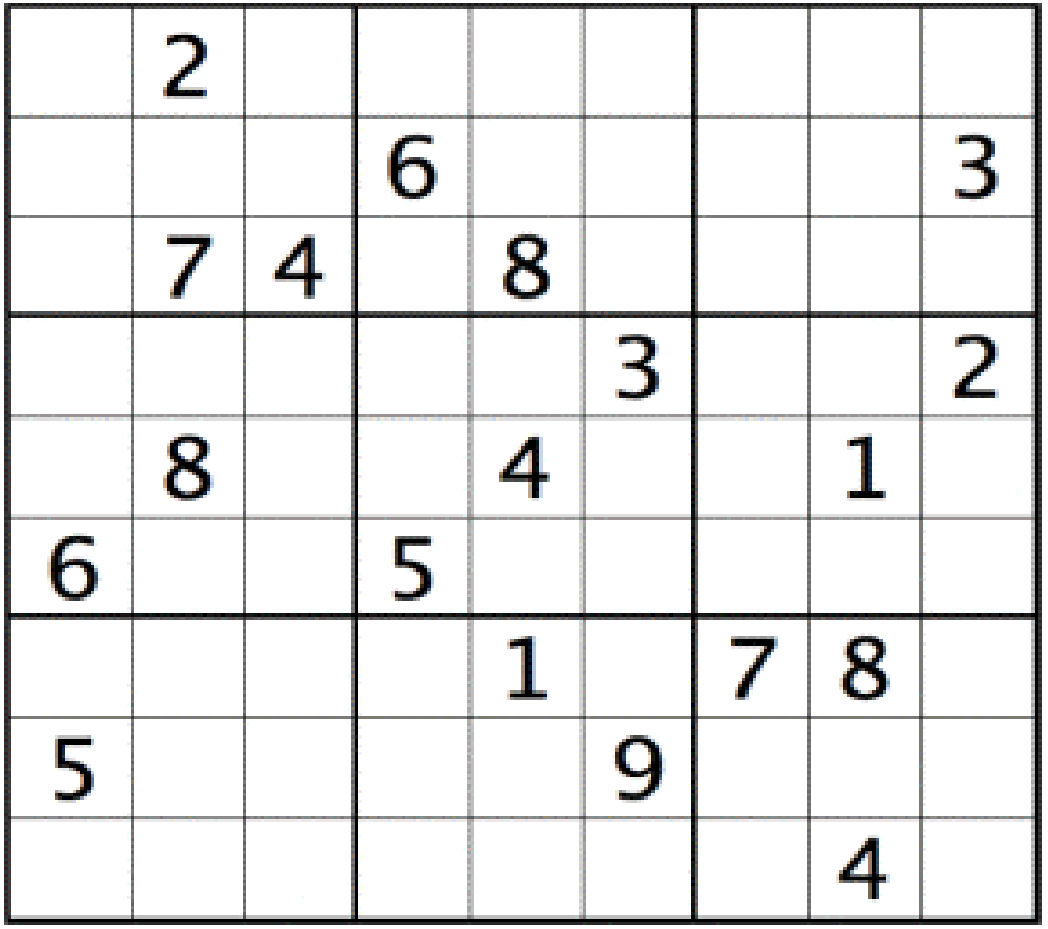
http://www.studiogiochi.com/admin/gcg/81/MultiSudokuNNconsecutivo.jpg

Samurai Sudoku



http://upload.wikimedia.org/wikipedia/commons/b/bb/Samurai-sudoku.png

“Not Fun” Basic Sudoku



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

http://dingo.sbs.arizona.edu/~sandiway/sudoku/examples.html

Generalized Solution for Overlapping Sudoku Puzzles

- Basic settings the control the game ranges:

```
rank = 3
range = rank*rank

dim value#Int = [1 .. range]
dim coord#Int = [1 .. rank]
dim array#Int = [0 .. range-1]
dim standardLocation#Int = [ 1 .. range ]
dim standardRowSet#Int = [1 .. rank]
dim standardColSet#Int = [1 .. rank]
```

Generalized Solution for Overlapping Sudoku Puzzles

- this controls checking over groups of boxes in the search:

```
standardRowSets = set #(standardRowSet,standardLocation) [(1, 1),
(1, 2), (1, 3), (2, 4), (2, 5), (2, 6), (3, 7), (3, 8), (3, 9)]
standardColSets = set #(standardColSet,standardLocation) [(1, 1),
(1, 4), (1, 7), (2, 2), (2, 5), (2, 8), (3, 3), (3, 6), (3, 9)]
```

- this controls the display of each box in the output:

```
baseColMap = set #(standardColSet,coord,array) [(1, 1, 0), (1, 2,
1), (1, 3, 2), (2, 1, 3), (2, 2, 4), (2, 3, 5), (3, 1, 6), (3, 2,
7), (3, 3, 8)]
baseRowMap = set #(standardRowSet,coord,array) [(1, 1, 0), (1, 2,
1), (1, 3, 2), (2, 1, 3), (2, 2, 4), (2, 3, 5), (3, 1, 6), (3, 2,
7), (3, 3, 8)]
```

Generalized Solution for Overlapping Sudoku Puzzles

- For normal Sudoku:

```
-- how many overlapping Sudoku games
games = 1
dim gameSet#Int = [1 .. games]

-- rank*rank*games-overlappers
boxes = 9
dim location#Int = [1 .. boxes]

-- this maps each box onto standard Sudoku games
gameMap = set #(gameSet, location, standardLocation) [(1, 1, 1),
(1, 2, 2), (1, 3, 3), (1, 4, 4), (1, 5, 5), (1, 6, 6), (1, 7, 7),
(1, 8, 8), (1, 9, 9)]
```

Generalized Solution for Overlapping Sudoku Puzzles

- For Multi Sudoku:

```
-- how many overlapping Sudoku games
games = 2
dim gameSet#Int = [1 .. games]

-- rank*rank*games-overlappers
boxes = 14
dim location#Int = [1 .. boxes]

-- this maps each box onto standard Sudoku games
gameMap = set #(gameSet, location, standardLocation) [(1, 1, 1),
(1, 2, 2), (1, 3, 3), (1, 4, 4), (1, 5, 5), (1, 6, 6), (1, 7, 7),
(1, 8, 8), (1, 9, 9), (2, 5, 1), (2, 6, 2), (2, 10, 3), (2, 8,
4), (2, 9, 5), (2, 11, 6), (2, 12, 7), (2, 13, 8), (2, 14, 9)]
```

Generalized Solution for Overlapping Sudoku Puzzles

- For Samarai sudoku:

```
-- how many overlapping Sudoku games
games = 5
dim gameSet#Int = [1 .. games]

-- rank*rank*games-overlappers
boxes = 41
dim location#Int = [1 .. boxes]

-- this maps each box onto standard Sudoku games
gameMap = set #(gameSet, location, standardLocation) [(1, 1, 1),
(1, 2, 2), (1, 3, 3), (1, 4, 4), (1, 5, 5), (1, 6, 6), (1, 7, 7),
(1, 8, 8), (1, 9, 9), (2, 10, 1), (2, 11, 2), (2, 12, 3), (2, 13,
4), (2, 14, 5), (2, 15, 6), (2, 16, 7), (2, 17, 8), (2, 1, 9), (3,
18, 1), (3, 19, 2), (3, 20, 3), (3, 21, 4), (3, 22, 5), (3, 23,
6), (3, 3, 7), (3, 24, 8), (3, 25, 9), (4, 26, 1), (4, 27, 2),
(4, 7, 3), (4, 28, 4), (4, 29, 5), (4, 30, 6), (4, 31, 7), (4,
32, 8), (4, 33, 9), (5, 9, 1), (5, 34, 2), (5, 35, 3), (5, 36,
4), (5, 37, 5), (5, 38, 6), (5, 39, 7), (5, 40, 8), (5, 41, 9)]
```

Generalized Solution for Overlapping Sudoku Puzzles

- this provides the actual puzzle inputs for each box:

```
puzzle = set #(location,coord,coord,value)
          [(1, 1, 2, 2), (1, 3, 2, 7), (1,
3, 3, 4), (2, 2, 1, 6), (2, 3, 2, 8), (3,
2, 3, 3), (4, 3, 1, 6), (4, 2, 2, 8), (5,
3, 1, 5), (5, 2, 2, 4), (5, 1, 3, 3), (6,
2, 2, 1), (6, 1, 3, 2), (7, 2, 1, 5), (8,
1, 2, 1), (8, 2, 3, 9), (9, 1, 1, 7), (9,
1, 2, 8), (9, 3, 2, 4)]
```

Generalized Solution for Overlapping Sudoku Puzzles

```
-- this computes some internally useful sets
rowColSets = games*rank
dim rowSet#Int = [1 .. rowColSets]
dim colSet#Int = [1 .. rowColSets]

-- [(1, 1, 1), (1, 2, 2), (1, 3, 3)]
n = [ (g, r+(g-1)*rank, r) | g <- [1 .. games], r <- [1 .. rank]
]
gameSetRowSets = set #(gameSet, rowSet, standardRowSet) n
gameSetColSets = set #(gameSet, colSet, standardColSet) n

-- set #(rowSet,location) [(1, 1), (1, 2), (1, 3), (2, 4), (2,
5), (2, 6), ...]
rowSets = $( { (r, l) <- gameSetRowSets(g, r,
q),standardRowSets(q, s),gameMap(g, l, s) } )
-- set #(colSet,location) [(1, 1), (1, 4), (1, 7), (2, 2), (2,
5), (2, 8), ...]
colSets = $( { (c, l) <- gameSetColSets(g, c,
q),standardColSets(q, s),gameMap(g, l, s) } )
```

Generalized Solution for Overlapping Sudoku Puzzles

- functions used in finding an answer for it:

```
getBox q r = $( {(k) <- r(n,i,j,k), n =#location $q } )
getRow q qs r = $( {(k) <- r(n,q,j,k),rowSets(i,n), i
=#rowSet $qs, q =#coord $q} )
getCol q qs r = $( {(k) <- r(n,j,q,k),colSets(i,n), i
=#colSet $qs, q =#coord $q} )

and [] = True
and (x:xs) = x && and xs
```

Generalized Solution for Overlapping Sudoku Puzzles

- the actual search for finding an answer for it:

```
exists grid : set #(location,coord,coord,value) puzzle .. fullRel
#(location,coord,coord,value)
 where          and [ full (getBox q grid) | q <- location ]
                && and [ full (getRow q list grid) | q <- coord,
list <- rowSet ]
                && and [  full (getCol q list grid) | q <- coord,
list <- colSet ]
                && $(grid(i,j,k,n),grid(i,j,k,m) -> n =#value m)
  find First
    by SAT
```

Generalized Solution for Overlapping Sudoku Puzzles

- functions for processing the display output:

```
-- set #(standardLocation, coord, array) [(1, 1, 0), ...]
leftCoordMap = $( { (l, c, a) <- standardRowSets(o,
l),baseRowMap(o, c, a) } )

-- set #(standardLocation, coord, array) [(1, 1, 0), ...]
rightCoordMap = $( { (l, c, a) <- standardColSets(o,
l),baseColMap(o, c, a) } )

-- set #(standardLocation, coord, coord, array, array) [(1, 1, 1,
0, 0)]
baseGameMaps = $( { (l, c, d, a, b) <- leftCoordMap(l, c,
a),rightCoordMap(l, d, b) } )

-- set #(gameSet,array,array,location,coord,coord) [(1, 0, 0, 1,
1, 1)]
gameSets = $( { (g,a,b,l,c,d) <- gameMap(g, l, s),
baseGameMaps(s, c, d, a, b)  } )
```

Generalized Solution for Overlapping Sudoku Puzzles

- functions for displaying a single game within:

```
getGame r n = $( {(i,j,k) <-
r(n,h,l,k),gameSets(q,i,j,n,h,l), q =#gameSet $n} )
display r n = setToArray (getGame r n)
```

Generalized Solution for Overlapping Sudoku Puzzles

- To view the solution in FunLog:

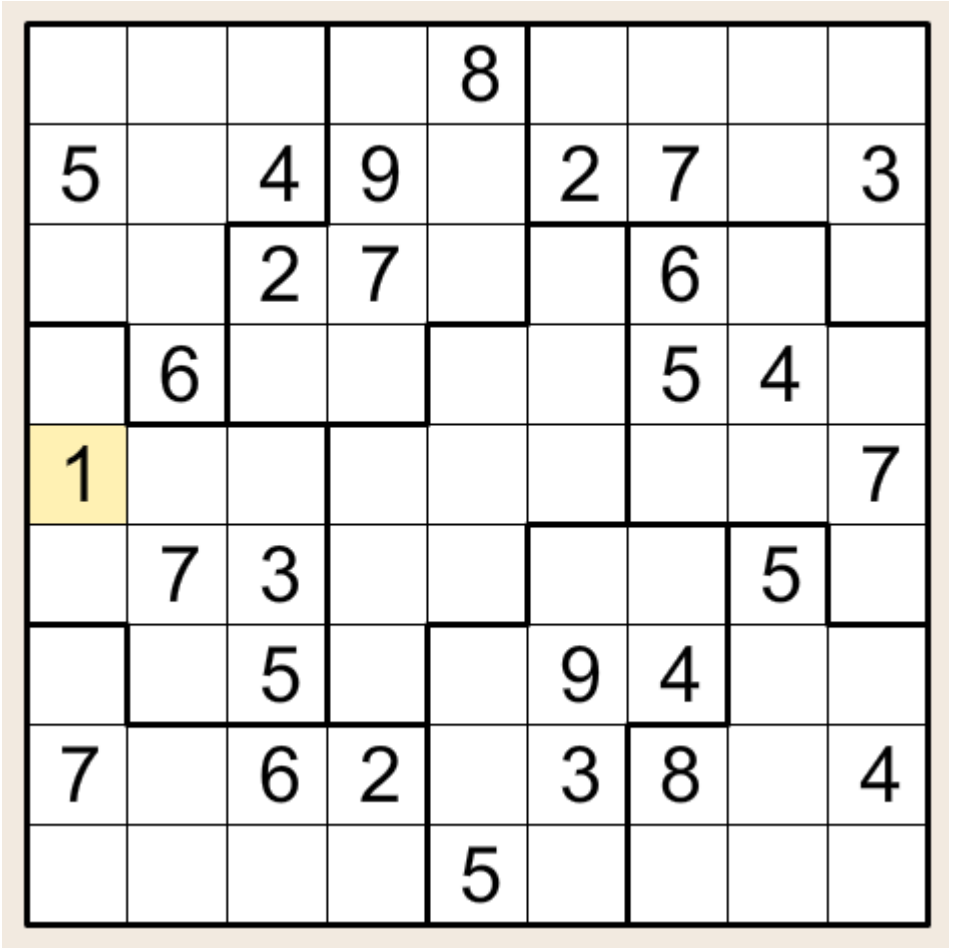
```
SATISFIABLE
exp> display grid 1

  0 1 2 3 4 5 6 7 8
 +-+-+-+-+-+-+-+-+-+
0|1|2|6|4|3|7|9|5|8|
 +-+-+-+-+-+-+-+-+-+
1|8|9|5|6|2|1|4|7|3|
 +-+-+-+-+-+-+-+-+-+
2|3|7|4|9|8|5|1|2|6|
 +-+-+-+-+-+-+-+-+-+
3|4|5|7|1|9|3|8|6|2|
 +-+-+-+-+-+-+-+-+-+
4|9|8|3|2|4|6|5|1|7|
 +-+-+-+-+-+-+-+-+-+
5|6|1|2|5|7|8|3|9|4|
 +-+-+-+-+-+-+-+-+-+
6|2|6|9|3|1|4|7|8|5|
 +-+-+-+-+-+-+-+-+-+
7|5|4|8|7|6|9|2|3|1|
 +-+-+-+-+-+-+-+-+-+
8|7|3|1|8|5|2|6|4|9|
 +-+-+-+-+-+-+-+-+-+
:: Array (Int,Int) Int
```

FunLog solved it quick!

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

http://dingo.sbs.arizona.edu/~sandiway/sudoku/examples.html

Irregular Sudoku Puzzle



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

http://www.oregonlive.com/puzzles-kingdom/?content_name=sud_irregular

Solution for Irregular Sudoku

- basic settings the control the game ranges:

```
rank = 3
range = rank*rank

dim value#Int = [1 .. range]
dim coord#Int = [0 .. range-1]
```

Solution for Irregular Sudoku

- this provides the actual puzzle inputs for each point:

```
puzzle = set #(coord,coord,value)
          [(0, 4, 8), (1, 0, 5), (1, 2, 4), (1, 3, 9), (1, 5,
2), (1, 6, 7), (1, 8, 3), (2, 2, 2), (2, 3, 7), (2, 6, 6),
(3, 1, 6), (3, 6, 5), (3, 7, 4), (4, 0, 1), (4, 8, 7), (5,
1, 7), (5, 2, 3), (5, 7, 5), (6, 2, 5), (6, 5, 9), (6, 6,
4), (7, 0, 7), (7, 2, 6), (7, 3, 2), (7, 5, 3), (7, 6, 8),
(7, 8, 4), (8, 4, 5)]
```

Solution for Irregular Sudoku

- this specifies the cells in each irregular box:

```
gameMap = set #(value, coord, coord) [(1, 0, 1), (1, 1, 0),
(1, 0, 0), (1, 1, 1), (1, 2, 0), (1, 0, 2), (1, 1, 2), (1,
2, 1), (1, 3, 1), (2, 0, 3), (2, 0, 4), (2, 1, 3), (2, 1,
4), (2, 2, 3), (2, 2, 4), (2, 3, 3), (2, 2, 2), (2, 3, 2),
(3, 0, 5), (3, 0, 6), (3, 0, 7), (3, 0, 8), (3, 1, 5), (3,
1, 6), (3, 1, 7), (3, 1, 8), (3, 2, 8), (4, 2, 5), (4, 3,
5), (4, 4, 5), (4, 3, 4), (4, 4, 4), (4, 5, 4), (4, 4, 3),
(4, 5, 3), (4, 6, 3), (5, 3, 0), (5, 4, 0), (5, 4, 1), (5,
4, 2), (5, 5, 0), (5, 5, 1), (5, 5, 2), (5, 6, 1), (5, 6,
2), (6, 2, 6), (6, 2, 7), (6, 3, 6), (6, 3, 7), (6, 3, 8),
(6, 4, 6), (6, 4, 7), (6, 4, 8), (6, 5, 8), (7, 6, 0), (7,
7, 0), (7, 8, 0), (7, 7, 1), (7, 8, 1), (7, 7, 2), (7, 8, 2),
(7, 7, 3), (7, 8, 3), (8, 6, 4), (8, 7, 4), (8, 8, 4), (8,
5, 5), (8, 6, 5), (8, 7, 5), (8, 8, 5), (8, 5, 6), (8, 6,
6), (9, 5, 7), (9, 6, 7), (9, 6, 8), (9, 7, 6), (9, 7, 7),
(9, 7, 8), (9, 8, 6), (9, 8, 7), (9, 8, 8)]
```

Solution for Irregular Sudoku

- functions for displaying a the game:

```
display r = setToArray r
```

- functions used in finding an answer for it

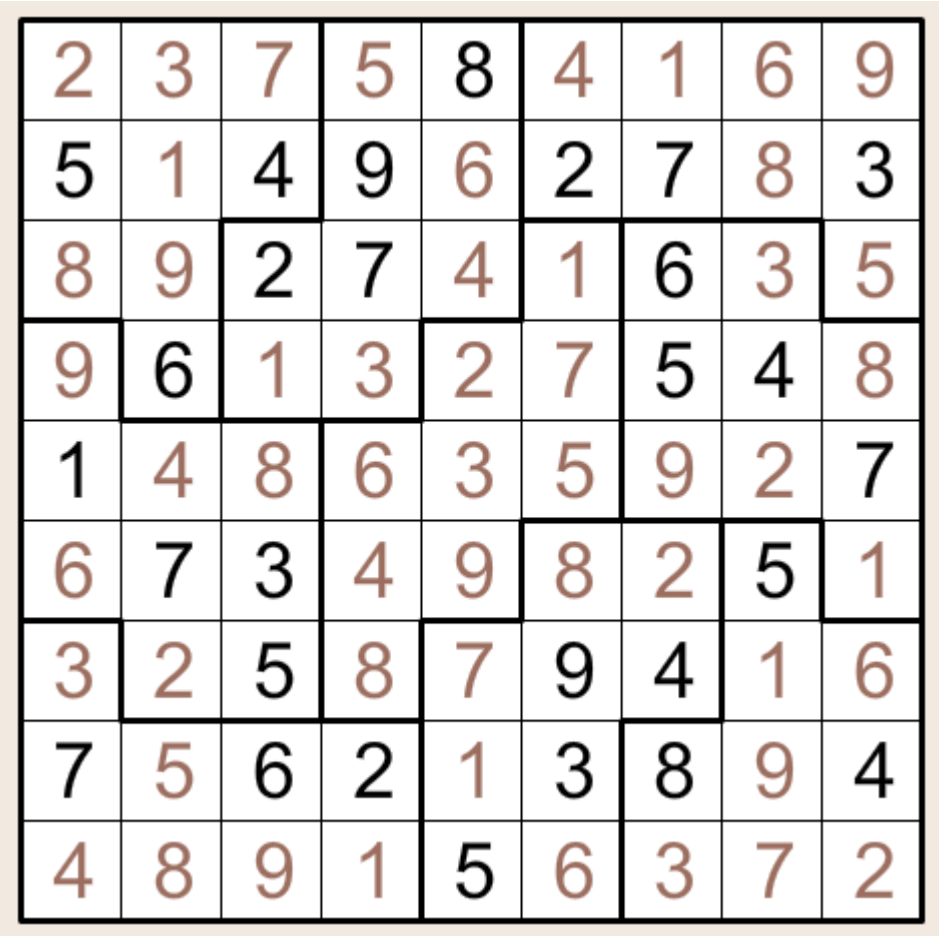
```
getBox q r = $( {(k) <- r(i,j,k),gameMap(q,i,j), q =#value
$q } )
getRow q r = $({(k) <- r(i,j,k), i =#coord $q})
getCol q r = $({(k) <- r(i,j,k), j =#coord $q})

and [] = True
and (x:xs) = x && and xs
```

Solution for Irregular Sudoku

- the actual search for finding an answer for it

```
exists grid : set #(coord,coord,value) puzzle .. fullRel
#(coord,coord,value)
 where          and [ full (getBox q grid) | q <- value ]
                && and [ full (getRow q grid) | q <- coord ]
                && and [ full (getCol q grid) | q <- coord ]
                && $(grid(i,j,n),grid(i,j,m) -> n =#value m)
  find First
    by SAT
```

FunLog solved this quick too!



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

http://www.oregonlive.com/puzzles-kingdom/?content_name=sud_irregular