

# Procedurell Generering: spelprototyp

Patrik Lindfors

d21patli

# Översikt

## AI-Teknologi:

- Procedurell Generering: metod för att generera data algoritmiskt istället för manuellt (Procedural Generation, 2023)
- Wave Function Collapse (WFC)
- Graph Grammar (GG)

## Spelprototyp:

- Sudoku generator (WFC – skapar brädet)
- Sudoku-lösare (GG – säkerställer att brädet är tillfredsställande att lösa)

## Spelupplevelse

- Välja svårighetsgrad, få ett slumpmässigt sudoku
- Inbyggda verktyg för att spelet ska kännas bra

# Översikt + demonstration

The image shows a screenshot of a Sudoku game interface. On the left is a 9x9 grid with some numbers pre-filled and others being entered or highlighted. On the right is a numeric keypad with various control buttons.

**Sudoku Grid:**

	5						9	
8	7							4
3					2		8	
			4	<sup>2</sup>	8			
7	2		6			8		
	8	5	1	<sup>2</sup>	9			
		8			5			1
				8	1			
2	1			3	<sub>46</sub>	7		8

**Keypad:**

- Numbers 1-9 and 0 in blue buttons.
- A button with a left arrow and an 'X' (delete).
- A button with a color wheel icon.
- Four buttons at the bottom: undo (left arrow), redo (right arrow), help/question mark, and refresh/reset (circular arrow).
- Four buttons on the right side of the keypad: a button with '1' in a box, a button with '1 2' and '3' in a box, a button with '12' in a box, and a button with a color wheel icon.

# Procedurell Generering (1 av 8)

## Wave Function Collapse (WFC)

Uppstår när en vågfunktion, ursprungligen mer flera superpositioner (**kandidater**) reduceras till en enda position p.g.a. interaktion med den externa världen (Wave Function Collapse, 2023).

En kollapsad vågfunktion påverkar andra vågfunktioner: informationen **propagerar**

**Entropi** = Antal kandidater

Villkor för att reducera vågfunktion:

- I regel: komplexa
- I Sudoku: Trivialt

# Wave Function Collapse (WFC)

Förklaring: se bilder

## Min algoritm för att fylla ett bräde:

```
While (grid.Full == false){
    var tile = FindLowestEntropyTile(grid)
    int candidate = tile.CollapseToLowsetCandidate(grid)
    Collapse(grid, tile, candidate)
```

```
if (grid.contradiction)
    Backtrack
```

}

[illegible][illegible]

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

Bildkälla: Donald (2020)

# Procedurell Generering (3 av 8)

## Wave Function Collapse (WFC)

Nu är brädet fyllt, så några siffror måste tas bort

Algoritm för att skapa pussel från färdigt bräde:

```
bool [,] visited = new bool[9,9]
```

```
while (AllTilesAreVisited(visited) == false){
```

```
    var tile = FindLowestEntropyTile  
    visited(tile.row, tile.col) = true;  
    grid.Remove(tile);
```

```
    if (grid.NumberOfSolutions != 1)  
        grid.Add(tile)
```

```
    Else if (grid.HumanlySolvable == false)  
        grid.Add(tile)  
}
```

**NumberOfSolutions** måste kollas eftersom ett giltigt sudokubräde bara får ha en lösning

- Kollas med en modifierad version av backtracking-algoritmen.

**HumanlySolvable** måste kollas för att se till att en människa faktiskt kan lösa pusslet med rimliga strategier, annars blir spelet inte kul att spela.

- Hur tusan vet man det??

# Procedurell Generering (4 av 8)

## Vad är en mänsklig strategi?

### Digit Method:

- Direkt utifrån brädet kan man se vars nästa siffra måste vara

### Candidate Method:

- Man kan logiskt resonera vars en kandidat kan tas bort (men inte direkt placera någon siffra i brädet)

# Procedurell Generering (5 av 8)

## Vad är en mänsklig strategi? – Digit Method

	1	2	3	4	5	6	7	8	9
A	2 3 4 7 8	1 3 4 7	5 5	1 3 1 4 6 8	1 3 1 4 6 8	1 3 1 4 6 8	1 3 1 4 6 8	9 4 6 8	1 3 1 4 6 8
B	3 1 3 4 7 8	1 3 1 4 7 8	1 3 1 4 7 8	9 4 7 8	5 6 8	1 3 1 4 6 8	1 3 1 4 5 6	3 6 8	2 4 5 6
C	3 1 3 4 8 9	6 4 8 9	1 3 4 8	2 4 8	1 5 8	1 3 1 4 8	1 3 1 4 5	7 4 5 8	1 3 1 4 5 8
D	4 5 6 8 9	1 4 5 4 5 9	1 4 6 4 8	1 2 6 4 8	1 2 6 4 8 9	1 2 6 4 8 9	3 8	2 6 8	7 8 9
E	3 1 3 4 7 8 9	1 3 1 4 7 9	2 4 7 9	1 3 1 4 8 9	1 3 1 4 8 9	1 3 1 4 8 9	1 3 1 6 9	5 8 9	1 3 1 6 8 9
F	3 1 3 6 8 9	1 3 1 6 9	1 3 1 6 8	7 5 8	1 2 6 6 9	1 2 6 6 9	4 6 9	1 3 1 6 8 9	1 3 1 6 8 9
G	1 4 7	4 3 4 7	5 4 6	2 6 8 9	2 6 7 8 9	2 6 4 7 9	2 3 4 6 9	2 3 4 6 9	3 4 6 9
H	3 5 6 7	8 9 7	9 7	6 4 7	2 6 7	2 6 5 6	1 5 6 7	1 5 6 7	3 5 6
I	4 5 6 7	2 4 6 7	3 1 6 7	3 1 6 7	8 6 9 7	8 6 9 7	8 6 9 7	6 4 5 6 9	6 4 5 6 9

	1	2	3	4	5	6	7	8	9
A	2 3 4 7 8	1 3 4 7	5 5	1 3 1 4 6 8	1 3 1 4 6 8	1 3 1 4 6 8	1 3 1 4 6 8	9 4 6 8	1 3 1 4 6 8
B	3 1 3 4 7 8	1 3 1 4 7 8	1 3 1 4 7 8	9 4 7 8	5 6 8	1 3 1 4 6 8	1 3 1 4 5 6	3 6 8	2 4 5 6
C	3 1 3 4 8 9	6 4 8 9	1 3 4 8	2 4 8	1 5 8	1 3 1 4 8	1 3 1 4 5	7 4 5 8	1 3 1 4 5 8
D	4 5 6 8 9	1 4 5 4 5 9	1 4 6 4 8	1 2 6 4 8	1 2 6 4 8 9	1 2 6 4 8 9	3 8	2 6 8	7 8 9
E	3 1 3 4 7 8 9	1 3 1 4 7 9	2 4 7 9	1 3 1 4 8 9	1 3 1 4 8 9	1 3 1 4 8 9	1 3 1 6 9	5 8 9	1 3 1 6 8 9
F	3 1 3 6 8 9	1 3 1 6 9	1 3 1 6 8	7 5 8	1 2 6 6 9	1 2 6 6 9	4 6 9	1 3 1 6 8 9	1 3 1 6 8 9
G	1 4 7	4 3 4 7	5 4 6	2 6 8 9	2 6 7 8 9	2 6 4 7 9	2 3 4 6 9	2 3 4 6 9	3 4 6 9
H	3 5 6 7	8 9 7	9 7	6 4 7	2 6 7	2 6 5 6	1 5 6 7	1 5 6 7	3 5 6
I	4 5 6 7	2 4 6 7	3 1 6 7	3 1 6 7	8 6 9 7	8 6 9 7	8 6 9 7	6 4 5 6 9	6 4 5 6 9

	1	2	3	4	5	6	7	8	9
A	2 3 4 7 8	1 3 4 7	5 5	1 3 1 4 6 8	1 3 1 4 6 8	1 3 1 4 6 8	1 3 1 4 6 8	9 4 6 8	1 3 1 4 6 8
B	3 1 3 4 7 8	1 3 1 4 7 8	1 3 1 4 7 8	9 4 7 8	5 6 8	1 3 1 4 6 8	1 3 1 4 5 6	3 6 8	2 4 5 6
C	3 1 3 4 8 9	6 4 8 9	1 3 4 8	2 4 8	1 5 8	1 3 1 4 8	1 3 1 4 5	7 4 5 8	1 3 1 4 5 8
D	4 5 6 8 9	1 4 5 4 5 9	1 4 6 4 8	1 2 6 4 8	1 2 6 4 8 9	1 2 6 4 8 9	3 8	2 6 8	7 8 9
E	3 1 3 4 7 8 9	1 3 1 4 7 9	2 4 7 9	1 3 1 4 8 9	1 3 1 4 8 9	1 3 1 4 8 9	1 3 1 6 9	5 8 9	1 3 1 6 8 9
F	3 1 3 6 8 9	1 3 1 6 9	1 3 1 6 8	7 5 8	1 2 6 6 9	1 2 6 6 9	4 6 9	1 3 1 6 8 9	1 3 1 6 8 9
G	1 4 7	4 3 4 7	5 4 6	2 6 8 9	2 6 7 8 9	2 6 4 7 9	2 3 4 6 9	2 3 4 6 9	3 4 6 9
H	3 5 6 7	8 9 7	9 7	6 4 7	2 6 7	2 6 5 6	1 5 6 7	1 5 6 7	3 5 6
I	4 5 6 7	2 4 6 7	3 1 6 7	3 1 6 7	8 6 9 7	8 6 9 7	8 6 9 7	6 4 5 6 9	6 4 5 6 9



# Procedurell Generering (6 av 8)

**Vad är en mänsklig strategi?**  
– **Candidate Method**

	1	2	3	4	5	6	7	8	9
A	2	<div>1 4</div> <div>3</div>	5	<div>3 8</div> <div>1 6</div>	7	<div>1 4</div> <div>6</div>	9	<div>4 8</div>	
B	<div>4 8</div>	<div>1 4</div>	7	9	<div>1 5 8</div> <div>6 4</div>	<div>6 4</div>	<div>1 4 5 6</div>	3	2
C	9	6	<div>3 8</div>	2	<div>1 5 8</div>	<div>3 4</div>	<div>1 4 5</div>	7	<div>4 5 8</div>
D	6	5	1	4	2	9	3	8	7
E	<div>4 7 8</div> <div>3</div>	<div>4 7</div> <div>3</div>	2	<div>3 8</div>	<div>6 8</div>	<div>3 6</div>	9	5	1
F	<div>3</div> <div>8</div>	9	<div>3</div> <div>8</div>	1	7	5	2	4	6
G	1	<div>3 7</div>	6	5	9	8	<div>4 7</div>	2	<div>4 3</div>
H	<div>3 7</div>	8	9	6	4	2	<div>5 7</div>	1	<div>3 5</div>
I	5	2	4	7	3	1	8	6	9

# Procedurell Generering (7 av 8)

## Graph Grammar (GG)

Teknik för att generera en **ny graf** utifrån en **befintlig graf**  
Utifrån bestämda **regler** (Graph Rewriting, 2022)

**Befintlig Graf:** **Brädet innan** candidate method

**Regel:** **Candidate Method – Pointing Pair**

**Ny graf:** **Brädet efter** candidate method

	1	2	3	4	5	6	7	8	9
A	2	<sup>1</sup> <sub>4</sub> 3	5	<sup>3</sup> <sub>8</sub> 1	<sup>6</sup> <sub>8</sub>	7	<sup>1</sup> <sub>4</sub> 6	9	<sup>4</sup> <sub>8</sub>
B	<sup>4</sup> <sub>8</sub>	<sup>1</sup> <sub>4</sub>	7	9	<sup>1</sup> <sub>5</sub> <sup>6</sup> <sub>8</sub> 4	<sup>6</sup> <sub>6</sub>	<sup>1</sup> <sub>4</sub> 5 6	3	2
C	9	6	<sup>3</sup> <sub>8</sub>	2	<sup>1</sup> <sub>5</sub> <sup>8</sup>	<sup>4</sup> <sub>3</sub>	<sup>1</sup> <sub>4</sub> 5	7	<sup>4</sup> <sub>5</sub> <sup>8</sup>
D	6	5	1	4	2	9	3	8	7
E	<sup>4</sup> <sub>7</sub> <sup>8</sup>	<sup>4</sup> <sub>7</sub>	<sup>2</sup> <sub>8</sub>	<sup>3</sup> <sub>8</sub>	<sup>6</sup> <sub>8</sub>	<sup>3</sup> <sub>6</sub>	9	5	1
F	<sup>3</sup> <sub>8</sub>	9	<sup>3</sup> <sub>8</sub>	1	7	5	2	4	6
G	1	<sup>3</sup> <sub>7</sub>	6	5	9	8	<sup>4</sup> <sub>7</sub>	2	<sup>4</sup> <sub>3</sub>
H	<sup>3</sup> <sub>7</sub>	8	9	6	4	2	<sup>5</sup> <sub>7</sub>	1	<sup>5</sup> <sub>3</sub>
I	5	2	4	7	3	1	8	6	9

# Procedurell Generering (8 av 8)

## Graph Grammar (GG)

```
bool HumanlySolvable (grid) {  
  
    while (grid.Solved == false){  
  
        if (DigitProgress(grid))  
            continue;  
  
        if (CandidateProgress(grid) == false)  
            return false;  
    }  
  
    return true;  
}
```

```
private static readonly List<DigitMethod> DigitMethods = new ()  
{  
    // Naked Single First because its fastest  
    new NakedSingle(),  
  
    // Hidden box single second because it's much easier for humans  
    // good for giving hints  
    new HiddenSingleBox(),  
  
    new HiddenSingleColumn(),  
    new HiddenSingleInRow(),  
};
```

```
private static readonly List<CandidateMethod> CandidateMethods = new ()  
{  
    // Pointing Pairs  
    new PointingPairRowToBox(),  
    new PointingPairColToBox(),  
    new PointingPairBoxToRow(),  
    new PointingPairBoxToCol(),  
  
    // Pointing Triples  
    new PointingTripleRowToBox(),  
    new PointingTripleColToBox(),  
    new PointingTripleBoxToRow(),  
    new PointingTripleBoxToCol(),  
  
    // Naked Pairs  
    new NakedPairInCol(),  
    new NakedPairInRow(),  
    new NakedPairInBox(),  
  
    // Hidden pairs  
    new HiddenPairInBox(),  
    new HiddenPairInRow(),  
    new HiddenPairInCol(),  
  
    // Naked Triples  
    new NakedTripleInRow(),  
    new NakedTripleInCol(),  
    new NakedTripleInBox(),  
  
    // Hidden triples  
    new HiddenTripleInBox(),  
    new HiddenTripleInRow(),  
    new HiddenTripleInCol(),  
  
    // Naked Quad (includes row, col and box)  
    new NakedQuad(),  
  
    // Hidden Quad (includes row, col and box)  
    new HiddenQuad(),  
  
    // XWings  
    new XWingRow(),  
    new XWingCol(),  
  
    // Swordfish  
    new SwordFishRow(),  
    new SwordFishCol(),  
  
    // Jellyfish  
    new JellyFishRow(),  
    new JellyFishCol(),  
  
    // Extended Wings  
    new XYWing(),  
    new XYZWing(),  
};
```