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)

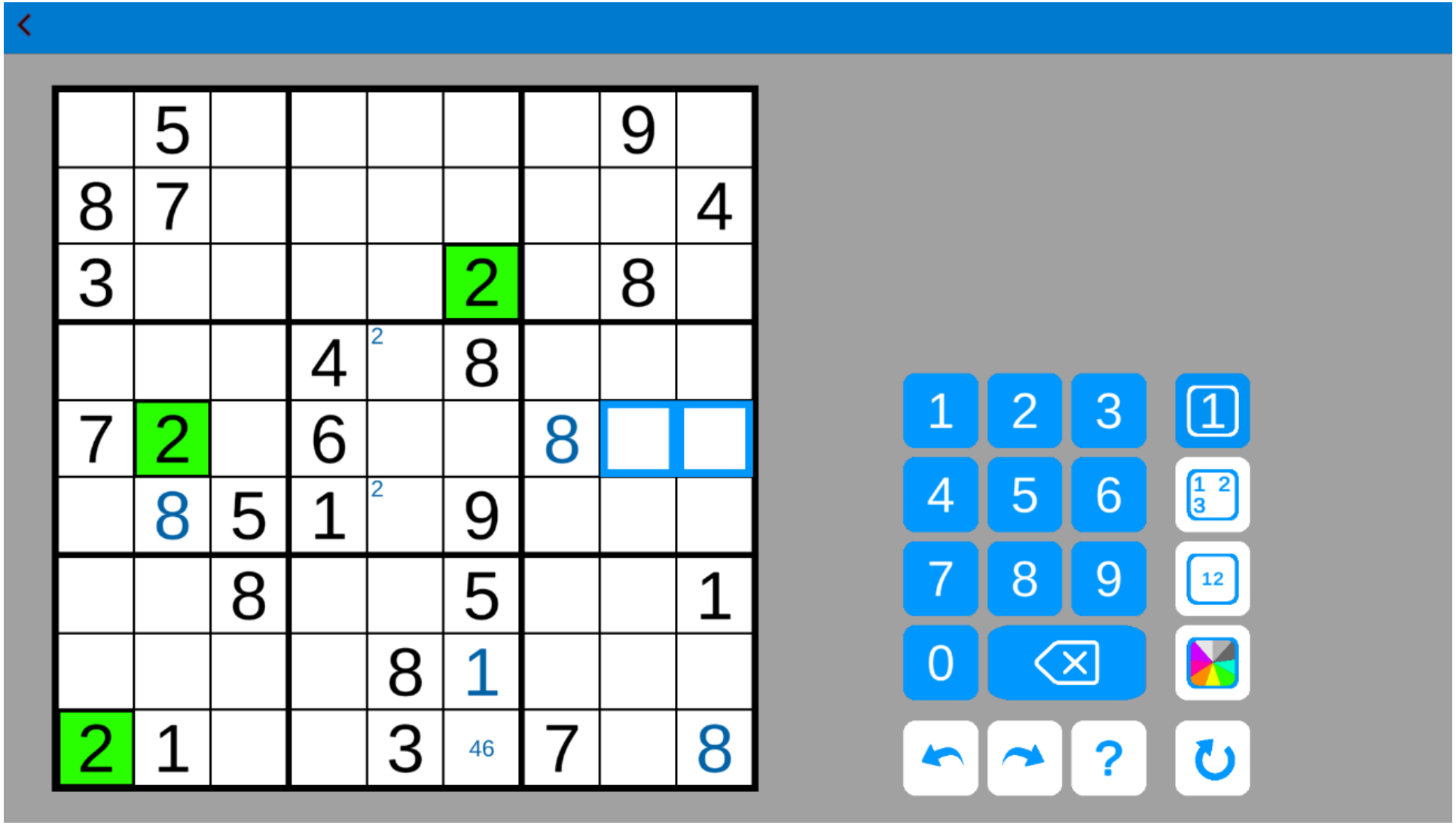
Spelprototyp:

- 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



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

Procedurell Generering (2 av 8)

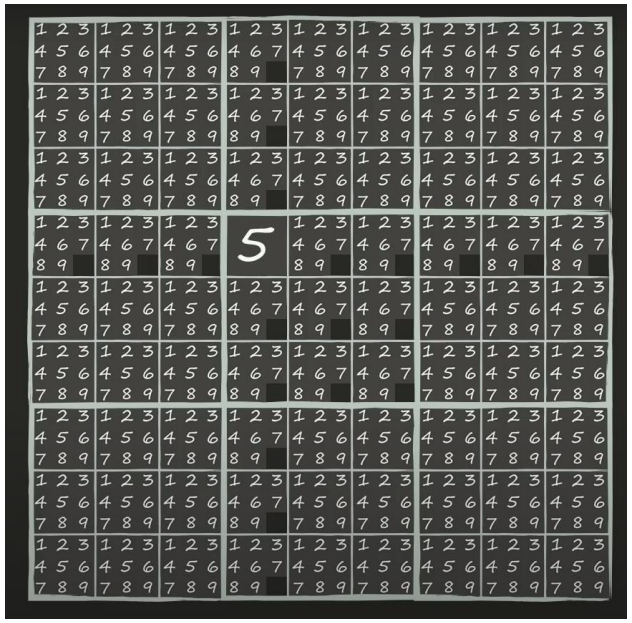
Wave Function Collapse (WFC)

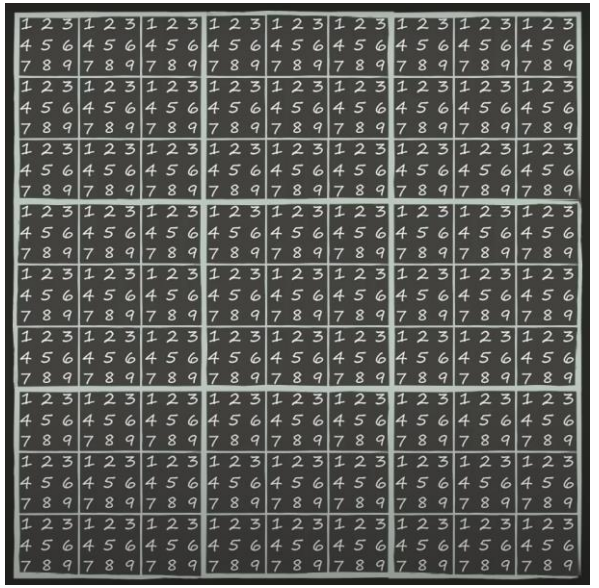
Förklaring: se bilder

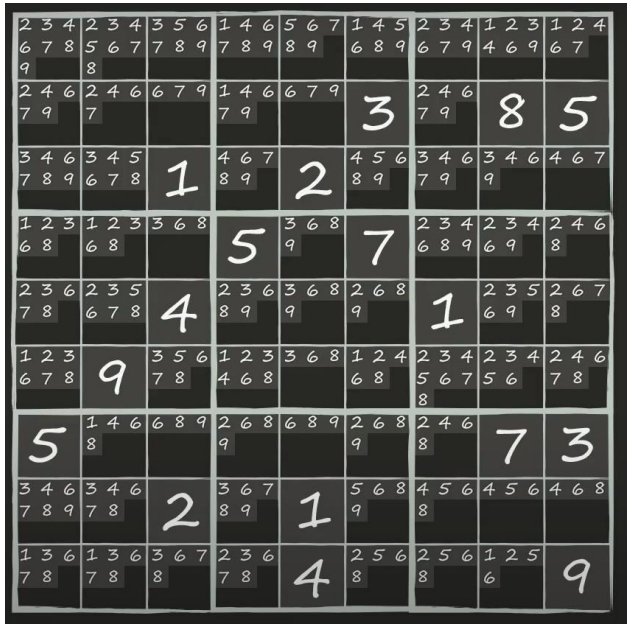
Min algoritm för att fylla ett bräde:

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

   if (grid.contradiction)
      Backtrack
}
```







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 gilitgt 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:

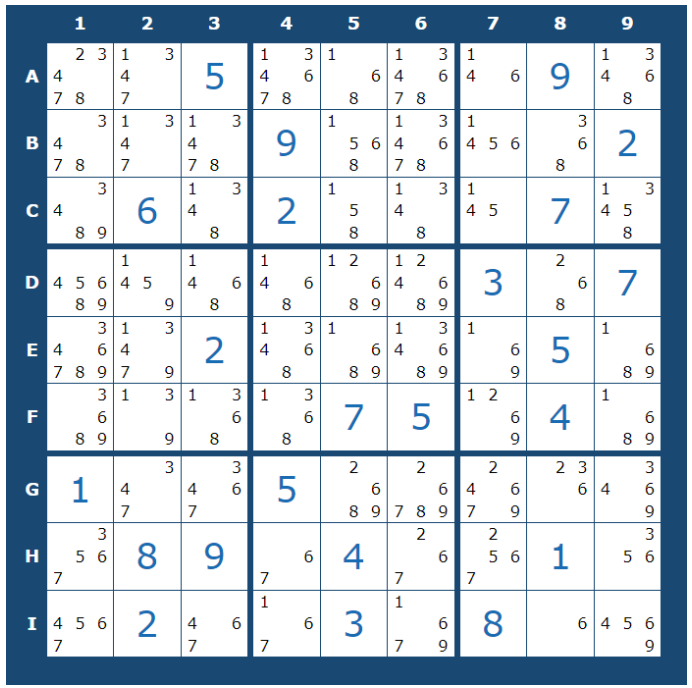
- Direkt utifrån brädet kan man se vars nästa siffra måste vara

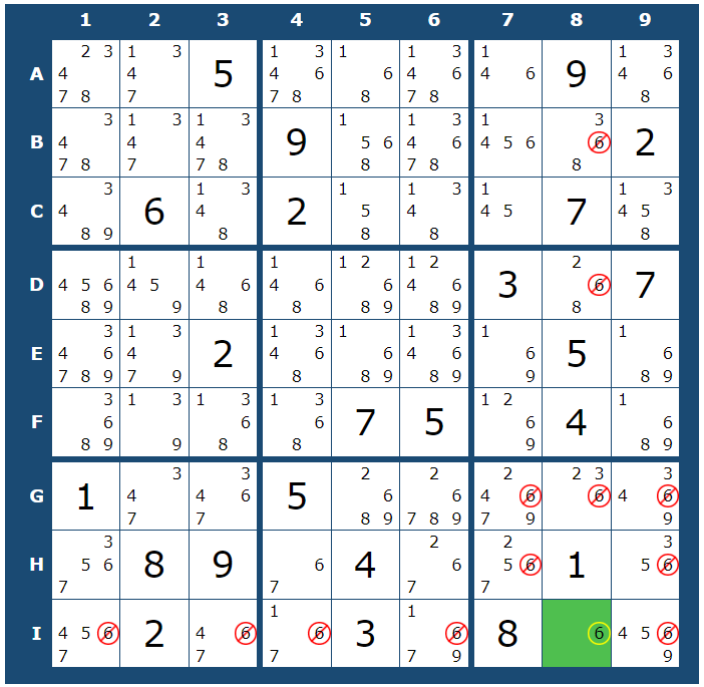
Candidate Method:

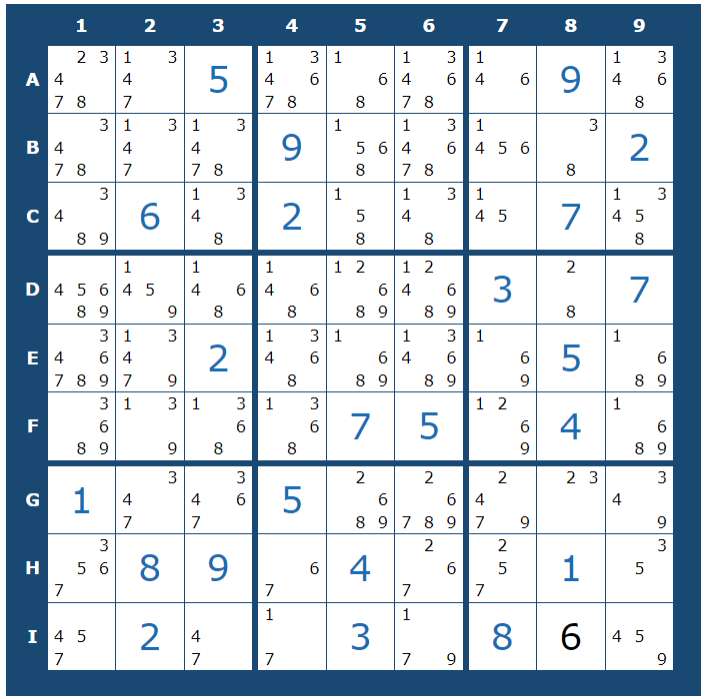
- Man kan logiskt resonera vars en kandidat kan tas bort (men inte direkt placera någon siffra i brädet)

Procedurell Generering (5 av 8)

Vad är en mänsklig strategi? – Digit Method







Bildkälla: Sudoku Solutions (u.å.).

Procedurell Generering (6 av 8)

Vad är en mänsklig strategi? – Candidate Method

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

Bildkälla: Sudoku Solutions (u.å.).

Procedurell Generering (7 av 8)

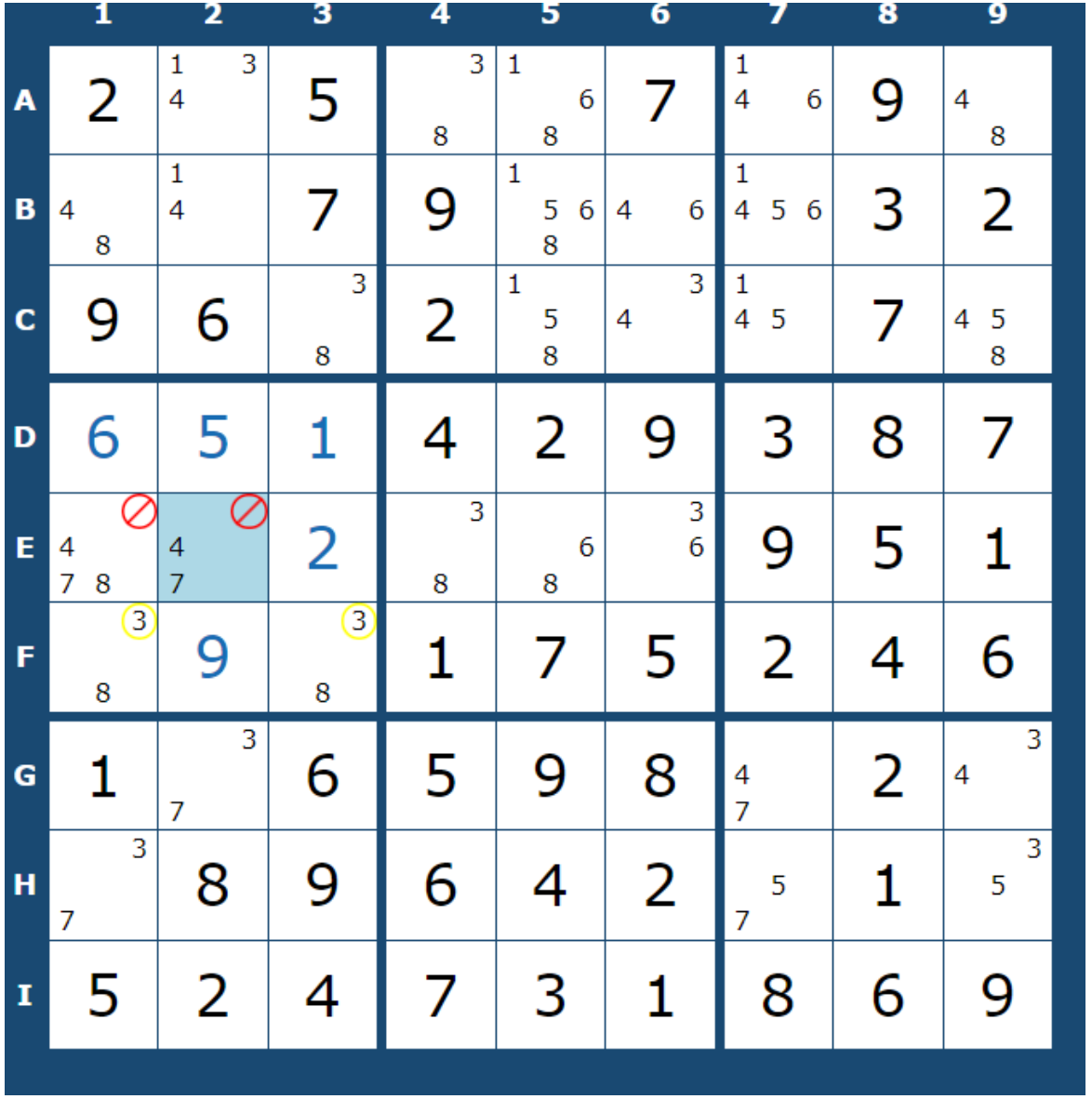
Graph Grammar (GG)

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



Bildkälla: Sudoku Solutions (u.å.).

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(),
};
```