**Formulation du Sudoku comme un CSP**

Un Sudoku classique est une grille 9×9 à compléter avec les chiffres de 1 à 9, de manière à ce que :

* Chaque ligne contienne les chiffres de 1 à 9 sans répétition.
* Chaque colonne contienne les chiffres de 1 à 9 sans répétition.
* Chaque sous-grille 3×3 contienne les chiffres de 1 à 9 sans répétition.

**1. Variables**

Chaque case de la grille est une variable. On peut nommer les variables de la manière suivante :

* A1 à A9 pour la première ligne,
* B1 à B9 pour la deuxième ligne,
* ...
* I1 à I9 pour la neuvième ligne.

Total : **81 variables**

**2. Domaines**

* Si une case est vide : domaine = {1, 2, ..., 9}
* Si une case est préremplie avec un chiffre : domaine = {ce chiffre}

**3. Contraintes**

On a 27 **contraintes globales de type Alldiff** :

* 9 pour les lignes
* 9 pour les colonnes
* 9 pour les blocs 3×3

Mais pour les algorithmes comme **AC-3**, on a besoin de **contraintes binaires**, donc on transforme chaque Alldiff en plusieurs contraintes binaires :

Par exemple :

* Pour la ligne A (A1 à A9) :  
  A1 ≠ A2, A1 ≠ A3, ..., A8 ≠ A9 → 36 contraintes binaires
* Idem pour les colonnes et les blocs

**4. Résumé du CSP Sudoku**

* **Variables**: 81 (une par case)
* **Domaines**: {1..9} ou singleton si préremplie
* **Contraintes**:
  + 27 Alldiff ⇒ transformées en **~1000 contraintes binaires** (chaque Alldiff sur 9 cases donne 36 contraintes binaires)

**5. Algorithmes à appliquer**

Tu peux ensuite résoudre le CSP avec :

* **Backtracking + AC-3** pour la propagation des contraintes
* **Heuristiques** :
  + **MRV (Minimum Remaining Values)** : choisir la case avec le moins de valeurs possibles
  + **Degree Heuristic** : préférer les cases qui imposent le plus de contraintes
  + **LCV (Least Constraining Value)** : choisir la valeur qui limite le moins les autres