Vamshi Jandhyala

Number Hooks

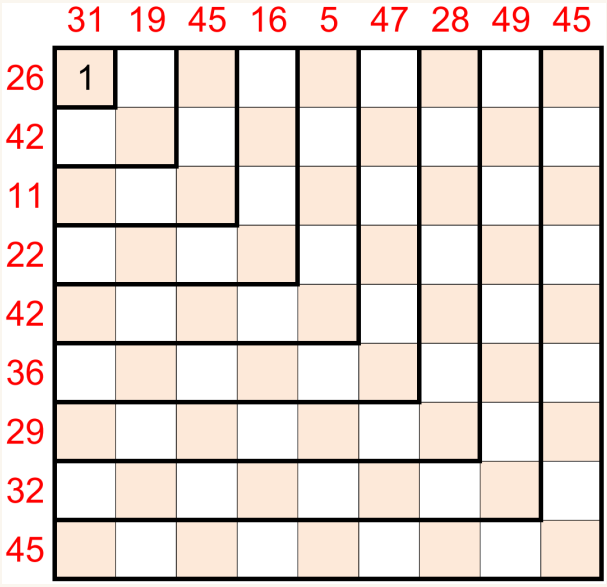
✠

November 2024

A nested-hook placement puzzle.

Problem

The grid below has a nested family of L-shaped *hooks*: the outermost hook spans the bottom row and the right-most column, the next hook is inset by one cell, and so on inward. In each hook of size k (for $k = 1, \ldots, 9$), place exactly k copies of the digit k, with the remaining cells of the hook left empty. The row and column sums printed along the borders must match the totals of the digits placed.



Solution

This is a constraint-satisfaction problem with three families of constraints (row sums, column sums, hook membership). Z3 solves it directly:

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

Python code

```
from z3 import *

def generate_hooks(size):
    hooks = []
    for n in range(1, size + 1):
        hook = []
        for i in range(n):
            for j in range(n):
                if i == n - 1 or j == n - 1:
                    hook.append((i, j))
        hooks.append(hook)
    return hooks

row_sums = [26, 42, 11, 22, 42, 36, 29, 32, 45]
col_sums = [31, 19, 45, 16, 5, 47, 28, 49, 45]

def solve_number_hook_puzzle():
    solver = Solver()
    grid = [[Int(f"cell_{i}_{j}") for j in range(9)] for i in range(9)]

    for i in range(9):
        solver.add(Sum(grid[i]) == row_sums[i])
        solver.add(Sum([grid[j][i] for j in range(9)]) == col_sums[i])

    hooks = generate_hooks(9)
    for i, hook in enumerate(hooks):
        value = i + 1
        cells_in_hook = [grid[r][c] for r, c in hook]
        solver.add(Sum([If(c == value, 1, 0) for c in cells_in_hook]) == value)
        for r, c in hook:
            solver.add(Or(grid[r][c] == value, grid[r][c] == 0))

    for row in grid:
        for cell in row:
            solver.add(And(cell >= 0, cell <= 9))

    if solver.check() == sat:
        m = solver.model()
        return [[m.evaluate(grid[i][j]).as_long()
                 for j in range(9)] for i in range(9)]
    return None

solution = solve_number_hook_puzzle()
```

```
if solution:
    for row in solution:
        print(" ".join(str(x) if x != 0 else '.' for x in row))
else:
    print("No solution found.")
```