Strimko By Resolution

Strimko: The Puzzle

Strimko is a popular puzzle on an $n \times n$ board, where apart from the n rows and columns, there are n streams/chains each of length n, as shown in varying colors in Figure 1. Each of the n^2 cells must be filled with a natural number $\in [1, n]$. The constraint is that no number can appear more than once in any row, column, or stream. Some of the cells are initially filled (Figure 1(left)), and you would want to fill the remaining cells without violating the constraint. Each board has a unique solution (e.g., Figure 1(right)) that can be reached by logical inference alone.

Figure 1: Left: A Strimko puzzle, initial board. Right: Solved (final) board.

Knowledge Representation

The solver's knowledge of the puzzle's rules will be represented in propositional logic. We have two choices: either (a) encode the complete knowledgbase upfront, and then only make inferences about cell-content without further expanding the knowledgbase, or (b) start with fewer facts, but with each new inference expand the knowledgbase to enable further inferences. We will take the second (incremental) approach.

Consider an $n \times n$ Strimko board. So there are n rows, n columns, and n streams/chains of length n

each. Suppose c_{ij}^k represents the proposition that cell (i, j) contains the number k [1]. Then for each row i and for each number k we will have a disjunction of the form:

$$c_{i1}^k \vee c_{i2}^k \vee c_{i3}^k \vee \ldots \vee c_{in}^k$$

This disjunction says that the number k must be somewhere in the row i. For $k \in [1, n]$, this implies that no number can appear more than once in a row. Since $k \in [1, n]$, there will be n disjunctions of this form for each row, and n^2 disjunctions over all rows. There will be a similar disjunction for each column and each chain as well, of which there are n each. Therefore, there will be a total $3n^2$ disjunctions of this form.

Now for each cell (i, j) we will also have a disjunction of the form:

$$c_{ij}^1 \vee c_{ij}^2 \vee \ldots \vee c_{ij}^n$$

which says that each cell must be filled with some number in $[1, n]$. Again there will be n^2 disjunctions of this form, over all cells.

The above $4n^2$ disjunctions constitute the base set of facts, that will be encoded before considering the (accumulative) evidence. For each evidence $c_{ij}^k = True$, i.e., when we know (i, j) must contain k for sure, we can add to the knowledgbase the following conjunction for the row i:

$$\neg c_{i,1}^k \wedge \neg c_{i,2}^k \wedge \ldots \wedge \neg c_{i,j-1}^k \wedge \neg c_{i,j+1}^k \wedge \ldots \wedge \neg c_{i,n}^k.$$

This conjunction says that k can only occur in one cell in the entire row i. Such a conclusion can also be made for the same evidence for the column j as well as its chain. Moreover, the same evidence will lead to the following conjunction

$$\neg c_{i,j}^1 \wedge \neg c_{i,j}^2 \wedge \ldots \wedge \neg c_{i,j}^{k-1} \wedge \neg c_{i,j}^{k+1} \wedge \ldots \wedge \neg c_{i,j}^n,$$

which says that cell (i, j) may contain nothing other than k. Thus there will be 4 conjunctions of length $n-1$ each as shown above, for each evidence. These can be broken into $4(n-1)$ independent propositions that are all true. These new facts will expand the knowledgbase.

Solving Strimko

Problem 1: 13 points

Implement the algorithm given in Algorithm 1 (**10 points**). Your code goes in the marked spot in Boards.py in the basecode. The inputs to the algorithm – sets A and B – are computed as part of the basecode. To run the program, execute

```
$python GUI.py
```

Suppose we have p evidences initially (input B), i.e., $4p(n-1)$ initial facts as described in the previous section. Each of these will reduce the length of one (of the $4n^2$) disjunctions (input A) by 1, according to *disjunctive syllogism* [2] that takes the following form:

[1] If we reach the conclusion that (i, j) cannot contain k, it is represented as $\neg c_{ij}^k$.

[2] Disjunctive syllogism is a special form of *resolution*.

$$x_1 \vee x_2 \vee x_3 \vee \ldots \vee x_n$$

$$\neg x_1$$

$$\therefore x_2 \vee x_3 \vee \ldots \vee x_n$$

After each of the $4p(n-1)$ facts have made their reductions on the disjunctions, if there is a disjunction with length 0, then there must have been a contradiction in A and B – *resolution refutation*. Note that if B is simply the initial set of facts, then this cannot happen. If there is any disjunction with length 1 (i.e., a conclusion of the form $c_{ij}^k = True$) then this may be a newly inferred cell or may have been already inferred before (*Question 1:* **Give an example showing that it may have been inferred before (3 points)**). If new, then in its turn it may produce *up to* $4(n-1)$ new facts, and the inference procedure continues this way. But if neither the input has been refuted, nor is there any disjunction with length 1 (i.e., a newly inferred cell), then resolveStrimko cannot solve this puzzle, and returns status "Unsolved" (see Problem 2).

Problem 2: 17 points

The inference procedure in Algorithm 1 is *sound* because it is based on disjunctive syllogism which is sound. However, it is not complete. Figure 2 shows a puzzle where it can make no further inference, but you can. For instance, considering the red stream, you know that (C2, R5) must contain either 5 or 6. But if it did contain 5, then the green stream couldn't contain 5 anywhere – a contradiction. Therefore (C2, R5) must contain 6. This line of reasoning is *depth first (backtracking) search* (DFS).

Write a DFS function (say, solveStrimko) that will call resolveStrimko as a subroutine, and use its output (A, B and *status*) to make a *guess* and continue until a contradiction is reached (**10 points**). Now you should be able to solve all included puzzles, even the ones marked "Unsolved" in the GUI.

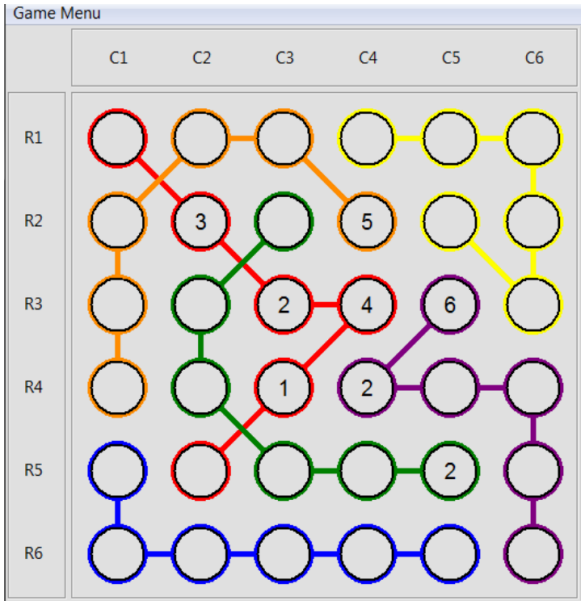

Figure 2: An unsolved but solvable board.

Question 2: **Prove that** SOLVESTRIMKO **is complete (5 points)**.

Question 3: **Prove that Sudoku (a more popular puzzle) is actually a special case of Strimko. That is,** SOLVESTRIMKO **can be applied to solve Sudoku as well (2 points)**. This question was contributed by Landon Kraemer.

Extra credits

FOL: (10 points) Since the knowledgebase grows as $O(n^3)$, it is useful to represent it more compactly, for large puzzles like Sudoku. Develop a *first order logic* representation, and rewrite the solver to work with this representation.

ASP: (5 points) Research the topic of *Answer set programming* and write a short note on how you could have written a complete Strimko solver with much less effort.

Submit

Submit answers to boxed questions 1–3 (+ extra credits ASP, if attempted) in print. Submit your code as a zip folder by email, and include a README to indicate whether the extra credits problem FOL was attempted, and any other remarks about the code.

Algorithm 1 resolveStrimko(A, B)

Input: Set A of current disjunctions, set B of p current facts.
Output: Potentially updated sets A and B, plus a flag indicating solve status
Local: Two sets *new* and *old*, latter initialized to B
while $|B| < n^2$ **do**
    $new \leftarrow \emptyset$
    for all $c_{ij}^k \in old$ **do**
        $C \leftarrow$ Generate a set of (up to $4(n-1)$) negative facts for empty cells in row i, column j and chain of (i, j)
        for all disjunction $d \in A$ **do**
            for all $\neg x \in C$ **do**
                if d contains x **then**
                    Remove x from d
                end if
            end for
            if $|d| = 0$ **then**
                Return (A, B, "Contradiction")
            end if
        end for
        for all disjunction $d \in A$ **do**
            if $|d| = 1$ **then**
                Remove d from A and add it to *new* (unless d is already in B)
            end if
        end for
    end for
    if *new* is empty **then**
        Return (A, B, "Unsolved")
    end if
    Add *new* to B
    $old \leftarrow new$
end while
Return (A, B, "Solved")