Generalizing Sudoku Strategies

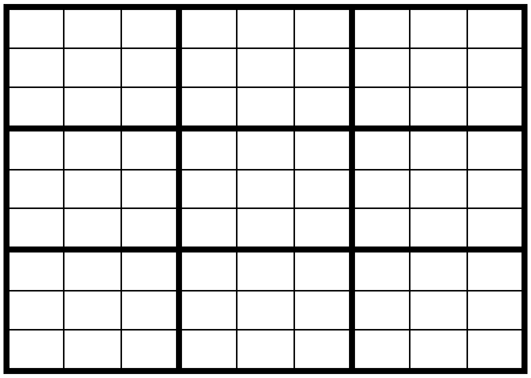
Kevin Gromley (March 2014)

Sudoku puzzles are quite popular, and numerous strategies exist for solving them. In the general literature these strategies are considered individually and given evocative names such as “X-wing” and “Swordfish”.

In this paper I will generalize some of these strategies, using ideas from set theory and combinatorics. These generalizations show the relationship of various individual strategies, and should help in coding algorithms for both solving and generating Sudoku puzzles.

About Sudoku

Sudoku is a form of Latin square puzzle, with added constraints. The standard Sudoku is a 9 X 9 grid. The objective is to place a number, 1 to 9, in each cell such that no number appears twice in any row or column. In addition, there are nine 3 X 3 regions, which I will term “blocks” herein; no number may appear twice in any block.



Standard Sudoku Grid

Other variants of the puzzle exist. In this paper I will only consider the standard 9 X 9 puzzle (although the generalizations here can be extended to other variants, in particular 16 X 16 and larger grids).

The user is given a partially-filled Sudoku grid and, using logic and following the basic row / column / block constraints, fills in the rest of the grid. A starting grid or puzzle is generally considered to be valid only if it has a unique solution.

There has been a good deal written on the mathematics of Sudoku; for a summary, see http://en.wikipedia.org/wiki/Sudoku . There are approximately 5.47 billion essentially different solutions (eliminating symmetries such as rotation and number substitution). The number of minimal puzzles (starting clues from which no cell value can be eliminated without losing uniqueness) is not precisely known, but has been shown to be of order 10^{25} (so we are not going to run out of Sudoku puzzles any time soon).

Approach and Nomenclature

A Sudoku puzzle can be considered to be a collection of constraint sets, i.e., a collection of rows, columns, and blocks. I use the term 'constraint set' to reflect the basic rule of the puzzle, i.e., that each number appears exactly once in each constraint set. Further, the layout of the Sudoku grid determines the relationship of these constraint sets, i.e, which sets intersect and where.

In this paper I will use the following nomenclature:

P	Sudoku puzzle
S_i	Particular constraint set (row, column, or block); this is a set of cells within P
S	Set of all constraint sets
S_R, S_C, S_B	Sets of all row, column, and block constraint sets, respectively
C_n	Cell set; this is a particular collection of cells within P (e.g., the intersection of two constraint sets S_i and S_j)
c	A particular cell
V	Value set; this is a particular collection of valid options (i.e., this is 'open' values; a cell that is filled in has no open values)
V(C)	Value set associated with cell set C (all the valid open values in set C)
v	A particular value

I will also use the normal set operators. These are summarized in Appendix 1.

With this in hand we can consider generalizations of various solution strategies. These strategies take the form of rules to eliminate possible values (v's) from unfilled cells in the puzzle.

Single-S Strategies

We consider two complementary strategies for individual row / column / block constraints. One addresses combinations of cells, the other combinations of values.

In words, the *cell combination* strategy is as follows:

> For each constraint set S_s, consider each proper subset of cells in S_s (i.e., each combination of 1 to 8 cells in S_s). Let n be the number of cells in a particular subset or combination. If there are exactly n valid values in all these cells combined, then none of these values is available for other cells in S_s.

In terms of our nomenclature:

```
(1)     For each $S_s$ in $P$ $(s = 1$ to $27)$
            For each $S_c \subset S_s$
                Let $n_c = |S_c|$ and $V_c = \bigcup_{i \in S_c} V(c_i)$.
                If $|V_c| = n_c$, then
                    for all $v \in V_c$, $v \notin V(S_c^c)$.
```

(Here, the expression "For each S_c" means taking all the combinations of cells within S_s; more on this below.)

In words, the *value combination* strategy is as follows:

> Consider each possible combination of n valid values, V_n. Consider each constraint set S_s. If there are exactly n cells in S in which all the values of V_n lie, then no other values are available to these n cells.

In terms of our nomenclature:

```
(2)     For each $S_s$ in $P$
            Consider each combination $V_n$ of $n$ valid values.
                Let $C_{V_n} = \bigcup_{S_s}($all $c_i$ in $S_s$ for which $\{V(c_i) \cap V_n\} \neq \emptyset)$.
                If $|C_{V_n}| = n$, then
                    for all $c_i \in C_{V_n}$, if $v \notin V_n$, then $v \notin V(c_i)$ .
```

Combinations of cells and values appear in the above strategies. Since there are 9 cells in any S and 9 permissible values, the number of combinations in both strategies is theoretically the same:

$$\binom{n}{k} \text{ where } n = 9 \text{ for Sudoku}$$

Elements (k)	Combinations
1	9
2	36
3	84
4	126
5	126
6	84

Elements (k)	Combinations
7	36
8	9
9	1

I noted above that these strategies are complementary. Testing for combinations of 8 cells is the same as testing for 'combinations' of single values. As a practical matter, it is unnecessary to test for more than four combinations of cells and four combinations of values.

These two generalizations subsume the "Naked Pairs / Triplets / Quads" and "Hidden Pairs / Triplets / Quads" strategies in the popular literature (see, for example, Andrew Stuart at http://www.sudokuwiki.org/sudoku.htm).

Two-S Strategies

This section considers strategies that involve pairs of constraint sets.

For our first generalization, the basic concept is that the values that lie within the interception of two constraint sets have implications for the other values outside the intersection. For example, consider the intersection of a row and block. Suppose the value '1' is valid for one or more of the three cells that lie in the intersection, but is not valid for any other cells in the block. This implies that '1' is not valid for any cells in the intersecting row that lie outside the block; if a '1' were placed in one of these cells, there would be no valid options left to place '1' in the block.

To generalize:

(3) $Consider\ any\ two\ constraint\ sets, S_i\ and\ S_j\ for\ which\ |S_i\ \cap\ S_j|\ > 1.$

(i.e., row / block or column / block intersection)

$If\ v \in V(\{S_i\ \cap\ S_j\}), then$

$$v\ \in\ V(\{S_i - \{S_i\ \cap\ S_j\}\})\ if\ and\ only\ if\ v\ \in\ V(\{S_j - \{S_i\ \cap\ S_j\}\}).$$

This generalization applies to row / block and column / block intersections. It subsumes the "Pointing Pairs", "Pointing Triplets", and "Box Line Reduction" strategies in the popular literature.

The next generalization also deals with the intersection of two constraint sets. It considers combinations of n cells that together contain n values. In certain circumstances this limits the values in the other cells of the two constraint sets.

(4) $Consider\ any\ two\ constraint\ sets, S_i\ and\ S_j\ for\ which\ \left|S_i\ \cap\ S_j\right|\ > 1.$

$Consider\ any\ n\ cells, C_{1..n}, for\ which$:

$C_{1..k}\ \in\ \{S_i\ \Delta\ S_k\}$ (C$_{1..k}$ lie in S$_i$ or S$_j$ outside the intersection)

$C_{k+1..n}\ \in\ \left\{S_i\ \cap\ S_j\right\}$ (the other cells are in the intersection; 0 < k < n → at least one cell in the intersection)

$Let\ V_n = \ \{\cup_{m=1}^{n} V(c_m)\}.$

$If\ |V_n| = n\ then$ (the n cells together have n valid values)

$consider\ each\ combination\ of\ two\ distinct\ cells\ c_x\ and\ c_y within\ C_{1..k}.$

(A) $If\ \left\{V(c_x)\ \cap V\left(c_y\right)\right\} = \ \emptyset\ for\ all\ c_x, c_y\ in\ \ C_{1..k}\ , then$

$$if\ v\ \ \in V(\{S_i\ \cap\ C_{1...k}\}), then\ v\ \ \notin\ V(\left\{S_i -\ \{S_i\ \cap\ C_{1..n}\}\right\})\ and$$

$$if\ v\ \ \in V\left(\left\{S_j\ \cap\ C_{1...k}\right\}\right), then\ v\ \ \notin\ V\left(\left\{S_j -\ \left\{S_j\ \cap\ C_{1..n}\right\}\right\}\right)$$

(if the C$_n$ selected cells in S$_i$ and S$_j$ outside the intersection share no values, then no value in one of these cells is valid in other cells in the same constraint set, S$_i$ or S$_{j,}$ outside the C$_n$ cells)

(B) $If\ \left\{V(c_x)\ \cap V\left(c_y\right)\right\} = \ \ \emptyset\ or\ \{v\}\ \ for\ all\ distinct\ c_x, c_y\ in\ \ C_{1..k}\ , then$

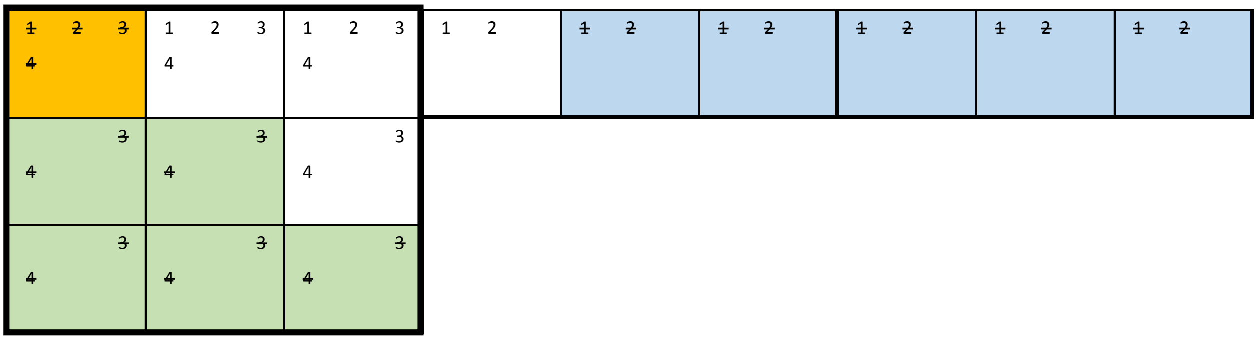
$$v\ \ \notin\ V(\left\{\left\{S_i\ \cap\ S_j\right\} -\ C_{k+1..n}\right\})$$

(if all pairs of cells in C$_{1..k}$ have at most one value v in common, then v is not valid in other cells in the intersection outside the C$_{k+1..n}$ cells)

This generalization subsumes the “XYZ / WXYZ Wing” strategies in the popular literature (plus some other rare cases).

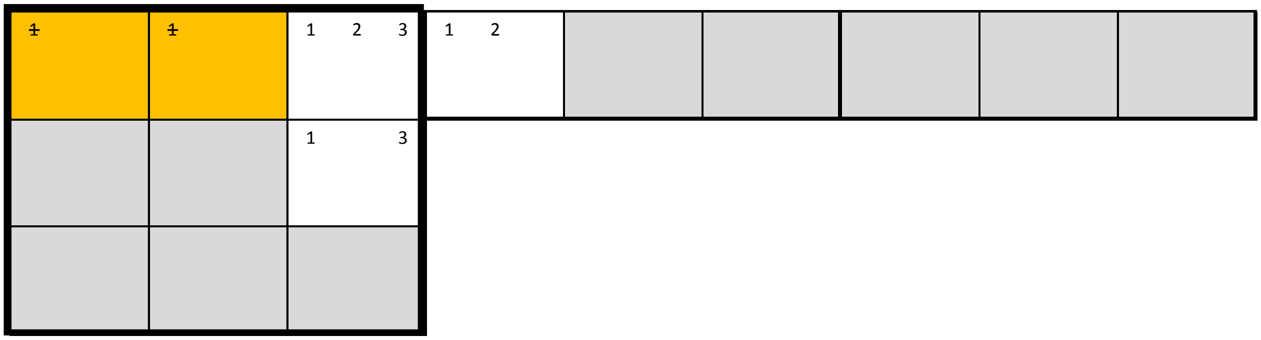
To give some examples, we will consider an intersecting row and block. Valid values for selected C$_{1..n}$ (white) cells are shown.

Case (A), n = 4



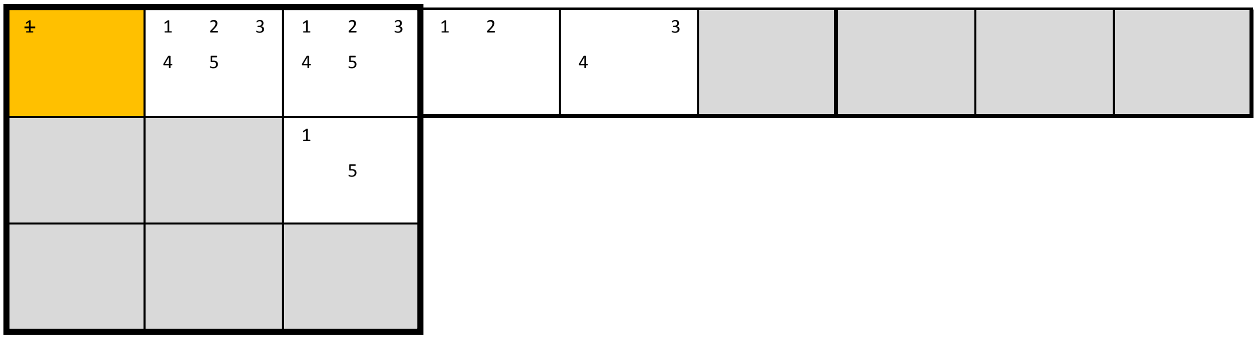
In this example, there are two cells in our $C_{1..4}$ (white cells) outside the intersection of the row and block. These two cells share no values. Consequently, the other cells in the row (blue) cannot have the same values as the cell in the row outside the intersection, i.e., neither 1 nor 2. Similarly, the other cells in the block (green) cannot have the same values as the cell in the block outside the intersection, i.e., neither 3 nor 4. Obviously, the remaining cell in the intersection (orange) cannot have 1, 2, 3, or 4.

Case (B), n = 3



In this example, there are two cells in our $C_{1..3}$ (white cells) outside the intersection of the row and block. These two cells share the value 1. Consequently, the other cells in the intersection (orange) cannot have 1 as a valid value. We cannot say anything about the grey cells.

Case (B), n = 5



In this example, there are three cells in our $C_{1..5}$ (white cells) outside the intersection of the row and block. One pair of these cells shares the value 1. Consequently, the other cell in the intersection (orange) cannot have 1 as a valid value. We cannot say anything about the grey cells.

Multi-S Strategies

This family of strategies considers the interactions of multiple constraint sets.

First we consider an even number of intersecting constraint sets.

(5) $Consider\ any\ 2N\ constraint\ sets,\ S_{2N},\ N > 1, such\ that,$

$$For\ all\ S_i, S_{i'}\ \in\ S_{1..N}\ and\ all\ S_j, S_{j'}\ \in\ S_{N+1..2N},$$

$$\{S_i\ \cap\ S_j\}\ \neq\ \emptyset\ and\ \{S_i\ \cap\ S_{i'\neq i}\} =\ \emptyset\ \ and\ \{S_j\ \cap\ S_{j'\neq j}\} =\ \emptyset$$

(e.g, $S_{1..N}$ are in S_R and $S_{N+1..2N}$ are in S_C, or vice versa; for the case where N = 2 we may also have either $S_{1..N}$ or $S_{N+1..2N}$ in S_B)

$$Let\ S_i\ \in\ S_{1..N}\ and\ S_j\ \in\ S_{N+1..2N}.$$

$$Let\ S_i'\ =\ \left\{\bigcup_{j=N+1}^{2N}\{S_i \cap S_j\}\right\}$$

(S_i' is all cells in S_i that intersect with $S_{N+1..2N}$; i = 1..N)

$$and\ S_j'\ =\ \left\{\bigcup_{i=1}^{N}\{S_i \cap S_j\}\right\}.$$

(analogous definition for S_j;
j = N+1..2N)

$$If, for\ all\ i = 1\ to\ N, v\ \in V(c_k)\ for\ all\ c_k\ \in\ S_i'\ and$$

$$v\ \notin\ V(\{S_i -\ S_i'\})\ , then$$

(v is an option for every intersection cell, but for no other cells in S_i, i = 1..N)

$$for\ all\ j = N + 1\ to\ 2N, v\ \notin\ V(\{S_j -\ S_j'\}).$$

(v is not an option for cells in S_j outside the intersection cells)

$$And\ conversely\ for\ all\ j = N + 1\ to\ 2N.$$

(Note: for N = 2, we may have blocks involved, e.g., the intersection of two rows and two blocks. In this case, we must add the stipulation that our *v* above appears in only one cell of each intersection. This is obviously not necessary to add in the case of row / column sets, since each intersection is one cell.)

This generalization covers the “X-Wing”, “Swordfish”, and “Jellyfish” strategies in the popular literature, as well as some other rare cases.

The last generalization addresses combinations of n cells that contain n values, but span multiple constraint sets. The general idea is as follows: Consider sets of n cells, C_n, that combined have n values, but which reside in different constraint sets. Consider a cell c_m outside C_n in one or more of the constraint sets in which at least one cell, but not all cells, of C_n reside. Then the c_m cell partitions C_n into two subsets, those cells that share a constraint set with c_m (C_o^c) and those that do not (C_o). If the open values in C_o are in some sense bounded by the open values in C_o^c, in some circumstances we may draw a conclusion about permissible values for c_m.

```
(6) $Consider\ all\ sets\ of\ n\ cells, C_n, for\ which\ |V(C_n)| = n.$

    $Consider\ all\ sets\ of\ N\ constraint\ sets, S_N, N \leq n+1,$
    $for\ which, for\ each\ S_i \in S_N, \{S_i \cap C_n\} \neq \emptyset, and\ \{S_i \cap \{S_N - S_i\}\} \neq \emptyset.$
                                        (pick up to n+1 constraint sets that contain the cells in $C_n$, taking care that each constraint set has at least one intersection with another in $S_N$)

    $Consider\ all\ c_m \in \{S_N - C_n\}.$        (pick a cell in $S_N$ outside $C_n$)

    $Let\ S_m = \bigcup_i \{S_i \in S_N\ for\ which\ S_i \cap \{c_m\} \neq \emptyset\}$
                                        (this is all constraint sets in $S_N$ that contain $c_m$)

    $Let\ C_o = \{C_n - \{C_n \cap S_m\}\}.$        (this is all cells in $C_n$ outside $S_m$)

    $If\ |C_o| > 0\ then$

      $For\ each\ c_i\ in\ C_o, let\ S_i = \bigcup_j \{S_j \in S_N\ for\ which\ S_j \cap \{c_i\} \neq \emptyset\}$
                                        (this is all the constraint sets in $S_N$ that contain $c_i$)

      $Let\ C_i' = \{C_o^c \cap S_i\}$        (this is all the cells of $C_n$, outside of $C_o$, (i.e., in $S_m$) that lie in constraint sets containing $c_i$)

      $If, for\ all\ c_j \in \{S_m \cap C_n\}, c_j \in of\ \{\bigcup_{i \in C_o} C_i'\}\ and$        (every cell of $C_n$ in $S_m$ shares at least one constraint set with a cell in $C_o$)

      $for\ all\ c_i in\ C_o, V(c_i) \subseteq V(C_i'), and$        (the set of values for each $c_i$ is a subset of the values of the $C_n$ cells in
```

S_m that share constraint set(s) with c_i)

$\mathit{for\ some}\ S_i, |V(\{C_n \cap S_i\})| \leq |\{C_n \cap S_i\}|, \mathit{then}$

(for at least one of the S_i constraint sets associated with the c_i in C_o, the number of values in the C_n cells in S_i is no greater than the number of C_n cells in S_i)

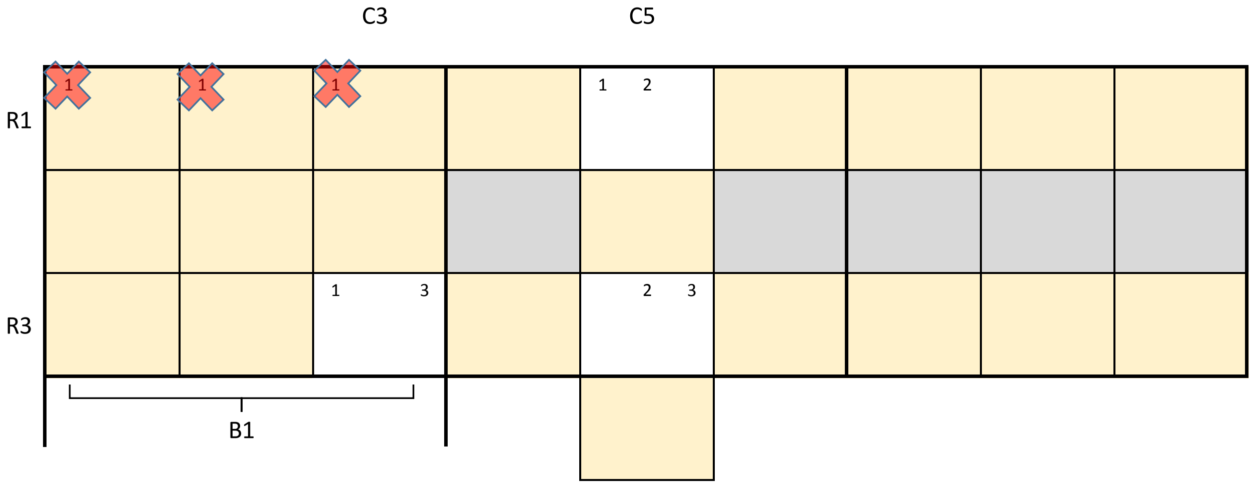
$\mathit{If\ there\ exists}\ v\ \mathit{such\ that}\ v \notin V(C_o), v \in V(C_i'), \mathit{and}$

$\mathit{for\ each\ distinct}\ c_x, c_y\ \mathit{in}\ C_i', \{V(c_x) \cap V(c_y)\} = \{v\}\ \mathit{or}\ \emptyset, \mathit{then}$

$v \notin V(c_m)$

(if v is not a value in C_o and v is the only common value among the cells in C_i', v cannot be a value for the chosen cell c_m)

This is a complex strategy, dealing with multiple constraint sets and cells. To illustrate an example for n = 3:



Our three cells with three values are (R1,C5), (R3,C3), and (R3,C5). We are considering four constraint sets, R1, R3, B1, and C5 [1]. If we look at c_m = any of the first three cells in R1, we see that:

S_m = R1 and B1, and hence includes (R1,C5) and (R3,C3) .

C_o is one cell in this case, so c_i = (R3,C5), S_i = {R3 and C5}, and C_i' = {(R1,C5), (R3,C3)}.

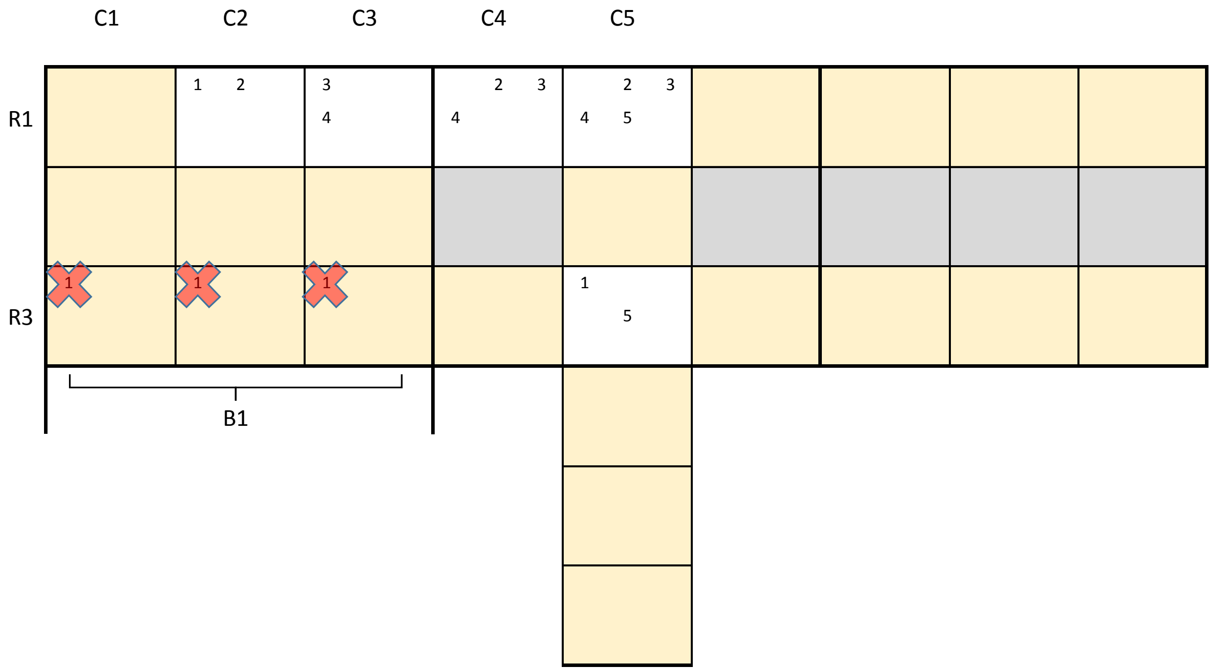
The values in c_i -- {2,3} -- are a subset of the values in C_i' -- {1,2,3}.

[1] Here, R indicates row, C column, and B block.

The number of C_n cells in S_i is three, equal to the number of possible values, so this test is also met.

The cells in C_i', (R1,C5) and (R3,C3) only share the value 1, and 1 is not present in c_i = (R3,C5). Consequently, none of the first three cells in R1 may have the value 1.

Another example, n = 5:



As before, the white cells show our $C_{n=5}$, with their permitted values due to values set elsewhere. Our S_N are again R1, R3, B1, and C5. Stepping through:

1. Consider c_m as cell (R3,C1). For this cell, S_m = {R3 and B1}, C_0 = {(R1,C4), (R1,C5)}.
2. The values in (R1,C4) are a subset of the values in {(R1,C2),(R1,C3)}, and the values in (R1,C5) are a subset of the values in {(R1,C2),(R1,C3),(R3,C5)}, so this test is met.
3. Consider cell (R1,C5), an element of C_0. Let this be c_i. For this cell, S_i = {R1 and C5} and C_i' = {(R1,C2),(R1,C3),(R3,C5)}.
4. The number of C_n cells in S_i is five, as is the number of possible values, so this test is met.
5. The value "1" is not an element of the C_0 cells but is the only common element among the C_i' cells. Consequently, 1 cannot be a value for c_m (R3,C1). The same logic applies to (R3,C2) and (R3,C3).

(Note: in step 3, we might have considered cell (R1,C4). However, this does not pass the test that the number of values in C_i' cannot be greater than the number of cells – four cells, five combined values. But the strategy also tests the other cell(s) in C_0, in this case (R1,C5).)

This generalization covers different variants of ‘Y-Wing’ and ‘XY-Chains’ in the popular literature.

Generalized Solution Algorithm

We have covered several general strategies for Sudoku. Many others exist. Solving a puzzle does not necessarily require all strategies. This leads to some interesting open questions. But first a bit more background.

Let us add the follow nomenclature:

R_0 the basic rule set for Sudoku; i.e., each value *v* appears exactly once in each S_i.

R_x additional solution strategies that we choose to apply; this may include some or all of the generalizations noted here, plus others.

For these purposes we will restrict R_x to strategies that are *state-deterministic*. That is, they can be applied based on the current state of the puzzle (filled in cells) and do not require making guesses of values or backtracking (although iterative applications of the rules are allowed). They are of the form “if the puzzle has this state (filled in cells), then we may eliminate these values as potential options from these cells”. The generalizations in this paper are all of this sort.

(As an aside, a state-deterministic rule set has this property: if a puzzle can be solved using only state-deterministic rules, than that solution is unique. The converse is not necessarily true, which gets to the issue of completeness, discussed in the next section.)

A general algorithm for solving a given Sudoku puzzle P (assuming it has a solution) is as follows.

```
1. Apply $R_0$ iteratively to remove options from non-filled-in cells. Iterate so long as values are removed.
2. Check for any cells that only have one valid option.
   A. If there are any such cells, fill them in.
      i. If all cells are filled in, stop, the puzzle is complete.
      ii. If some cells remain open, loop back to 1.
   B. If no open cell has a single option remaining, go to 3.
3. Apply $R_x$ iteratively to remove options from non-filled-in cells. Iterate so long as values are removed.
   A. If any values are removed, loop back to 1.
   B. Otherwise, go to 4.
4. Select a cell in which to make a guess.
5. Select an open value for the guess cell.
6. Apply 1 – 3 iteratively until one of these three possible outcomes:
   A. The puzzle is solved – stop.
```

B. The guess is infeasible (reach a point that violates R_0). In this case, undo the last guess and subsequent changes, then loop back to 5, selecting another open value for the current guess cell.

C. No more options can be removed but the puzzle is still unsolved. In this case, loop back to 4 and execute recursively (pick an additional cell in which to make a guess and iterate 4 – 6).

Our chosen set of strategies, Rx, appears in step 3. If this set of strategies, applied iteratively, does not solve the puzzle, then a guessing operation follows. (The guessing operation may be enhanced by applying some logic to step 4, the selection of the guess cell, based on the current open values and the rule set R_x being applied.)

Open Questions

The foregoing leads some interesting questions regarding completeness and efficiency.

Completeness: Is there a set of R_x for which no guesses are required for any valid (unique solution) Sudoku puzzle? I.e., is there a set of state-deterministic elimination strategies that, applied iteratively, solves any valid puzzle by step 3 in the above outline algorithm?

I suspect that the answer to this is 'no', but have no proof of this. It is known that solving Sudoku is a NP-complete problem. Since many of our solution strategies are combinatorial in nature, however, the existence of such an R_x would not necessarily mean a polynomial-time solution algorithm.

Efficiency: Several of the generalized solution strategies herein, as well as others in the literature, are combinatorial in nature – investigating possible combinations of cells and values. Some of these solution strategies may be computationally inefficient, i.e., they may take more steps or time than a simple recursive guessing approach (such as 'dancing links') based on the basic R_0 rules.

So two questions present themselves:

> For a given Sudoku puzzle, is there an a priori way to determine or estimate an efficient set of R_x to apply?
>
> Averaged over all valid, essentially different Sudoku puzzles, what R_x is most efficient?

APPENDIX 1: SET NOTATION

This is taken from http://www.rapidtables.com/math/symbols/Set_Symbols.htm

Symbol	Symbol Name	Meaning / definition	Example
$\{\ \}$	Set	a collection of elements	$A = \{3,7,9,14\}$, $B = \{9,14,28\}$
$A \cap B$	Intersection	objects that belong to set A and set B	$A \cap B = \{9,14\}$
$A \cup B$	Union	objects that belong to set A or set B	$A \cup B = \{3,7,9,14,28\}$
$A \subseteq B$	Subset	subset has fewer elements or equal to the set	$\{9,14,28\} \subseteq \{9,14,28\}$
$A \subset B$	proper subset / strict subset	subset has fewer elements than the set	$\{9,14\} \subset \{9,14,28\}$
$A \not\subset B$	not subset	left set not a subset of right set	$\{9,66\} \not\subset \{9,14,28\}$
$A = B$	Equality	both sets have the same members	$A=\{3,9,14\}$, $B=\{3,9,14\}$, $A=B$
A^c	Complement	all the objects that do not belong to set A	
$A - B$	relative complement	objects that belong to A and not to B	$A = \{3,9,14\}$, $B = \{1,2,3\}$, $A - B = \{9,14\}$
$A \Delta B$	symmetric difference	objects that belong to A or B but not to their intersection (similar to ‘XOR’ in logic)	$A = \{3,9,14\}$, $B = \{1,2,3\}$, $A \Delta B = \{1,2,9,14\}$
$a \in A$	element of	set membership	$A=\{3,9,14\}$, $3 \in A$
$x \notin A$	not element of	no set membership	$A=\{3,9,14\}$, $1 \notin A$
$\|A\|$	Cardinality	the number of elements of set A	$A=\{3,9,14\}$, $\|A\|=3$
Ø	empty set	Ø = { }	C = {Ø}