Chapter 3

Advanced Techniques

The advanced techniques presented in this chapter are meant to solve *difficult* grids.

3.1. Pairs, triples and subsets

In this section, we focus on looking for sets of particular values inside a given region.

3.1.1. *The naked pair technique*

Consider a given region. Suppose that in this region there are two cells for which the two same values are the only candidates. As such, these values can be safely removed from the candidate lists of the other cells of the region.

Indeed, assigning any of these values to another cell will leave one of the two identified cells without a potential value, which is strictly forbidden for a sudoku grid.

Formally:

RULE 3.1 – naked pair –

– **Parameter**: a region R

– **Condition**: $\exists (i_1, j_1) \in R, (i_2, j_2) \in R,$
$\texttt{what}(\{(i_1, j_1), (i_2, j_2)\}) = \{v_1, v_2\}$

– **Deduction**: $\forall (i, j) \in R \setminus \{(i_1, j_1), (i_2, j_2)\}, (i, j) \neq v_1, (i, j) \neq v_2$

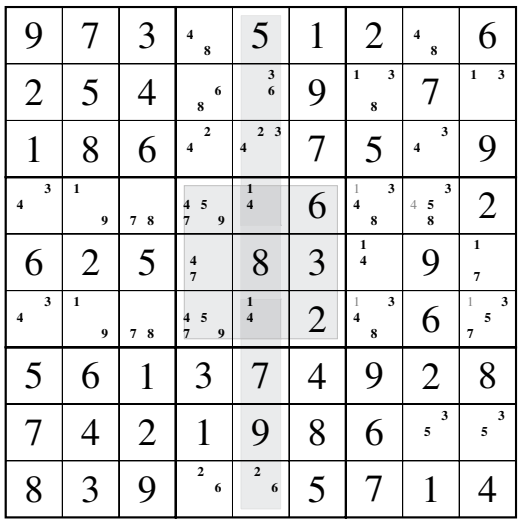

Figure 3.1. *Rule 3.1 at work*

EXAMPLE.– Consider the grid in Figure 3.1. In column C_5, cells $(4, 5)$ and $(6, 5)$ only accept values 1 and 4. Therefore, these values can be removed from the candidate lists of all the other cells in the column. Thus, the value 4 is removed from cell $(3, 5)$. The rule can also be applied on block B_5 leading to the removal of 4 from all cells in column C_4 intersecting with block B_5. It can then be deduced that cell $(5, 4)$ must be assigned with the value 7.

EXERCISE 18. – Solve the grid in Figure 3.1.

3.1.2. *The naked tuples technique*

Rule 3.1 can be rewritten to take into account k values. For example, for three values, the following formal rule can be written:

RULE 3.2 – naked triple –

– **Parameter**: a region R

– **Condition**: $\exists (i_1, j_1) \in R, (i_2, j_2) \in R, (i_3, j_3) \in R,$
$\texttt{what}(\{(i_1, j_1), (i_2, j_2), (i_3, j_3)\}) = \{v_1, v_2, v_3\}$

– **Deduction**: $\forall (i, j) \in R \setminus \{(i_1, j_1), (i_2, j_2), (i_3, j_3)\},$
$(i, j) \neq v_1, (i, j) \neq v_2, (i, j) \neq v_3$

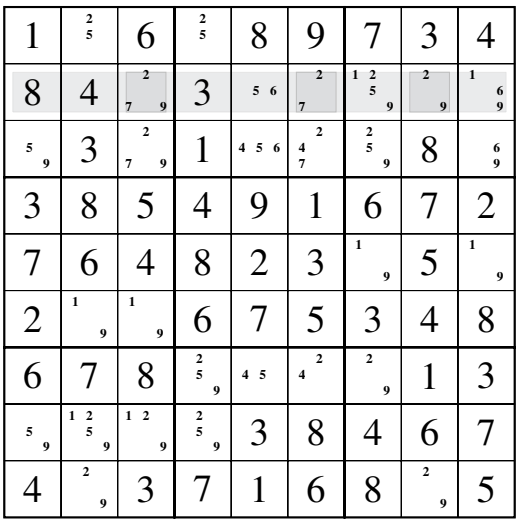

Figure 3.2. *Rule 3.2 at work*

A small issue should be considered here. Although this is not the case for the previous rule, the three values may not all be present in the three identified cells. This is the union of the candidates that form a triple over the three cells.

EXAMPLE.– On the grid of Figure 3.2, rule 3.2 can be applied on row R_2. Cells $(2,3)$, $(2,6)$, and $(2,8)$ share three candidates: 2, 7, and 9. Therefore, value 9 can be removed from cells $(2,7)$ and $(2,9)$ as well as value 2 from cell $(2,7)$.

EXERCISE 19. – Where is the triple on row R_6 in the grid of Figure 3.3?

More generally, for k values, the rule becomes:

RULE 3.3 – naked tuple –

– **Parameters**: a region R, an integer k

– **Condition**: $\exists\{(i_1,j_1),\ldots,(i_k,c_k)\} \in R^k$,
$\texttt{what}(\{(i_1,j_1),\ldots,(i_k,j_k)\}) = \{v_1,\ldots,v_k\}$

– **Deduction**: $\forall v \in \texttt{what}(\{(i_1,j_1),\ldots,(i_k,j_k)\})$,
$\forall (i,j) \in R \setminus \{(i_1,j_1),\ldots,(i_k,c_k)\}, (i,j) \neq v$

EXERCISE 20. – There is a quad in block B_9 in the grid of Figure 3.4. Can you see it?

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

Figure 3.3. *Looking for a triple (rule 3.2)*

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

Figure 3.4. *Looking for a quad (rule 3.3)*

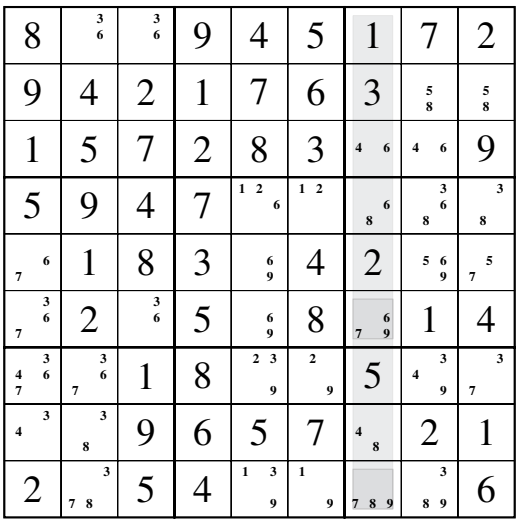

Figure 3.5. *Rule 3.4 at work*

3.2. Hidden subsets

There exist a dual set of rules (compared to the previous ones). They can be used when the subset of values that are being sought are *hidden*. Let us illustrate this duality first with hidden pairs.

3.2.1. *The hidden pair technique*

Consider a given region. If there are two values that have the same possible cell position of only two, then all other values are forbidden for these two cells.

Indeed, if any other value is assigned to one of these cells, one of the identified values will have no possible position in the region. This would not be acceptable.

Formally:

RULE 3.4 – hidden pair –

– **Parameters**: a region R, two values v_1 and v_2

– **Condition**: $\texttt{where}(\{v_1, v_2\}, R) = O_{v_1,v_2}$ and $|O_{v_1,v_2}| = 2$

– **Deduction**: $\forall v \notin \{v_1, v_2\}, \forall (i,j) \in O_{v_1,v_2}, (i,j) \neq v$

---

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

Figure 3.6. *Rule 3.5 at work*

EXAMPLE.– In the grid of Figure 3.5, values 7 and 9 on column C_7 are candidates in only two cells. All other values may be removed from those two cells.

NOTE.– *Inferring such information is useful because other rules may apply.*

EXERCISE 21. – Which other rule (from this chapter) can be applied here with exactly the same result?

3.2.2. *The naked tuple technique*

As for rule 3.1, the previous rule may be generalized to k values. Therefore:

RULE 3.5 – hidden tuple –

– **Parameter**: a region R, a set V of k values

– **Condition**: $\texttt{where}(V, R) = O_V$ and $|O_V| = k$

– **Deduction**: $\forall v \notin V, \forall (i,j) \in O_V, (i,j) \neq v$

NOTE.– *Be careful, because the same issue as above arises: values may not all be present in the considered cells.*

EXAMPLE.– On row R_5 in the grid in Figure 3.6, values 3, 6, and 7 are only possible in three cells: $(5,4)$, $(5,6)$, and $(5,9)$. Thus, the rule applies and value 9 can be removed from cell $(5,9)$.

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

Figure 3.7. *A difficult grid*

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

Figure 3.8. *A very difficult grid*

INFORMATION.– *The joint application of the rules from rule 2.1 up to rule 3.3 (when only considering $k \leq 3$) can solve any difficult grids.*

EXERCISE 22. – Solve the difficult grid in Figure 3.7.

EXERCISE 23. – Use the learnt techniques as much as possible on the grid in Figure 3.8. What is the resulting grid?

In the next chapter we will solve very difficult grids, but, before that, it is worth having a closer look at the rules presented in this chapter.

3.3. Intrinsic properties of subset based rules

The two sets of rules that we have presented are strongly related. We have seen this when answering exercise 21. We will now explicitly exhibit this relation, and we will also show that all these rules are subsumed by a more powerful and general rule.

3.3.1. *Subset-based rules duality*

Let R be a region with p non-assigned cells. It can easily be shown that rule 3.3 applied to region R for n values has the same result as rule 3.5 applied to region R and $p - n$ values.

Indeed, these two rules define a partitioning of the region into:

– a set E of $9 - p$ assigned cells;

– a set F of n cells for which globally only n candidates are available;

– a complement set G of $p - n$ cells which globally contain $p - n$ values that can be assigned on cells other than in G.

Rules 3.3 and 3.5 both forbid cells in G to receive a value shared by cells in F.

EXAMPLE.– In the grid in Figure 3.1 on page 38, consider column C_5. Using rule 3.5 on values 2, 3 and 6 (which have only three candidate rows R_2, R_3 and R_9) leads to the removal of value 4 from cell $(3, 5)$. This is exactly the same conclusion as considering rule 3.3 on rows R_4 and R_6 which only have two possible values: 1 and 4.

EXERCISE 24. –

What partitioning can be identified when considering row R_5 in the grid of Figure 3.6 on page 42?

3.3.2. *Some properties of region reasoning*

Reasoning on a given region of a sudoku grid leads to a common situation in graph theory: the *maximum matching problem*. On one side, the cells of the region are considered. On the other side, the digits from 1 to 9 are considered. What is needed is

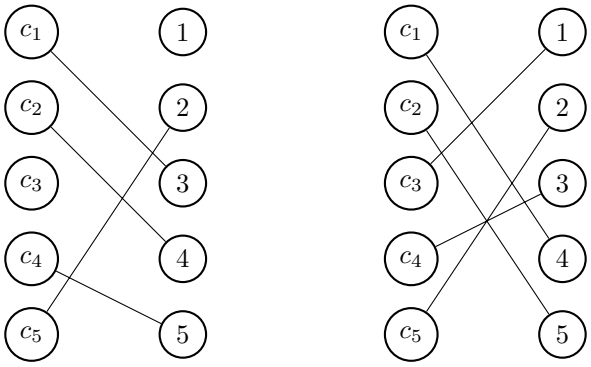

Figure 3.9. *Two example matchings. The matching on the left is not a maximal matching (cell c_3 has no match). However, the matching on the right is*

to assign to each cell a unique digit in such a way that a value is not assigned to two different cells. This is a *matching*. It is a *maximum* matching because all cells must have a value.

> EXAMPLE.– Consider five cells $(c_1, \ldots, c_5)$ and five candidates $(1, \ldots, 5)$. Figure 3.9 gives two example matchings. Such a matching can be represented as a graph in which left vertices are the cells and right vertices are the candidates. A link (an edge) between a cell and a candidate represents the fact that the value is assigned to the cell. In these examples, we can clearly that no two edges have a vertex in common.

When considering a sudoku grid, a value cannot be assigned to all the cells: some already have one (the givens), others are restricted to their candidate list, etc. This information has to be taken into account. Therefore, a specific graph is designed, in which a maximal matching is to be found: a cell is linked to a digit if the digit is a valid candidate for this cell.

> EXAMPLE.– Let us get back to our five cells and five values. Figure 3.10 gives an example graph. Here, we have: $\texttt{what}(\{c_1\}) = \{1, 2, 3\}$, $\texttt{what}(\{c_2\}) = \texttt{what}(\{c_3\}) = \{2, 3\}$, $\texttt{what}(\{c_4\}) = \{1, 2, 4, 5\}$, and $\texttt{what}(\{c_5\}) = \{3, 4, 5\}$. A maximal matching is sought using only the edges of this graph. Figure 3.11 gives two such matchings (the edges in the matching are in bold).

In the specific context of sudoku, what is looked for is not a solution to this problem. Indeed, such a solution may actually not fit with the other regions of the grid.

> EXAMPLE.– This can be clearly seen on the previous example (see Figure 3.11). The solution for a sudoku grid is unique: there is no way to tell which of the two matchings will lead to a solution.

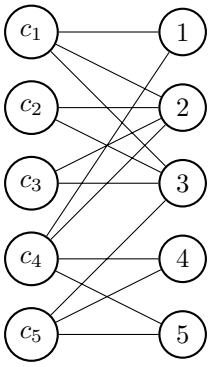

Figure 3.10. *A sudoku-like situation. A maximal matching is needed in this graph*

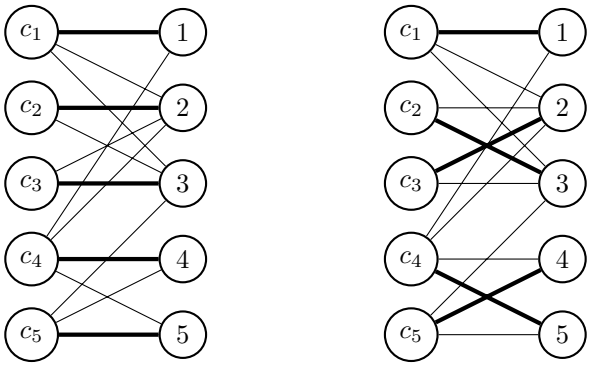

Figure 3.11. *A sudoku-like situation: two example matchings*

Indeed, what is needed are:

– the mandatory assignments (not counting givens): are there any cell/value couples that appear in all the solutions of this problem?

– the forbidden assignments: are there any cell/value couples that never appear in a solution of this problem?

EXAMPLE.– Consider Figure 3.10. The following points can be identified:

– one mandatory assignment: c_1 is bound to have value 1. Indeed, the other candidates are 2 and 3 but these values are the only possible ones for a pair of cells (c_2 and c_3). Therefore, rule 3.1 (the naked pair rule) can be applied and 1 becomes the only valid candidate for c_1 (rule 2.2);

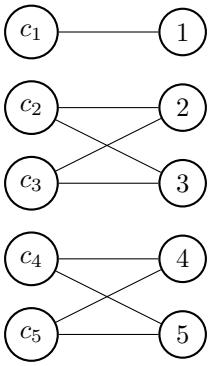

Figure 3.12. *Reasoning about maximum matchings*

– several forbidden assignments: c_4 cannot take value 1 (it has been assigned to c_1) and cannot take value 2 for the same reason as before. This is also the case for cell c_5 and value 3.

Thus, c_1 must be assigned to value 1, whereas 2 and 3 are shared by c_2 and c_3, and 4 and 5 are shared by c_4 and c_5. Indeed, making any other combination assignment will make it impossible to reach a maximal matching. The graph of Figure 3.10 becomes that in Figure 3.12: edges that correspond to values removed from the candidate lists were removed from the graph.

From an algorithmic point of view, identifying mandatory assignments is quite easy. This is exactly what rules 2.1 and 2.2 do. A simple (i.e., there exists an efficient algorithm to achieve the calculation) way to answer the second question (forbidden assignments) consists of identifying set of auto-sufficient and independent elements (cells and/or values). This is partly what rules 3.1 to 3.5 do with the partition that is provided. Any cell/value assignment *linking* any two of these sets will never appear in a solution.

EXAMPLE.– In Figure 3.10 (page 46), the 4 vertices c_2, c_3, 2, and 3 form an auto-sufficient set[1]. Trying to give another value to c_2 or c_3 will never lead to a maximal matching. The same applies if one tries to assign value 2 or 3 to another cell.

EXERCISE 25. – What are the other auto-sufficient sets in Figure 3.10?

1. The technical name is *strongly connected component* in an oriented graph where the edges *in* the matching are oriented from left to right and the edges *not in* the matching are oriented from right to left.

The rules presented here are all a particular case of this more general reasoning. There are very efficient algorithms that can be used to solve this problem. They can deduce several pieces of information at the same time. Unfortunately, using such a reasoning by hand is a quite difficult and tedious task. However, these algorithms are very useful when considering developing software to solve sudoku grids. This will be the topic of Chapter 5.

INFORMATION.– *The results presented here are part of a well-known tool in the constraint programming community: the* alldifferent *constraint. For more information, see the chapter* Global constraints and filtering algorithms *written by Jean-Charles* RÉGIN *in the book* Constraints and integer programming combined *edited by Michela* MILANO *and published by Kluwer in 2003, which gives more detail on involved algorithms.*