The Logic of Exclusion

A Comprehensive Analysis of Sudoku Locked Candidates and Intersection Removal Techniques

Research Report

December 16, 2025

Contents

1 Introduction: Sudoku as a Constraint Satisfaction System

The game of Sudoku, while popularized as a logic puzzle accessible to the layperson, represents a sophisticated exercise in finite domain constraint satisfaction. At its core, a standard 9×9 Sudoku grid is a system governed by a precise set of exclusionary rules. The puzzle demands that every cell $C_{r,c}$ (where r is the row index and c is the column index) be assigned a value $v \in \{1, \ldots, 9\}$ such that no value is repeated within any of the three fundamental "units" or "houses": the Row, the Column, and the Block (the 3×3 sub-grid).

Solving Sudoku is not merely a process of placing digits; it is a subtractive process of candidate elimination. The solver begins with a full universe of possibilities—potentially 729 candidates in an empty grid—and systematically whittles this domain down until only one valid configuration remains. While elementary techniques rely on explicit "singles" (where a cell has only one remaining possibility), intermediate and advanced solving depends on analyzing the relationships between candidates.

Among the hierarchy of solving strategies, the technique known as **Locked Candidates** (often referred to within technical communities as **Intersection Removal**) serves as a critical bridge. It moves the solver beyond the localized logic of a single cell or unit and forces an examination of the interactions *between* units. Specifically, Locked Candidates addresses the logical consequences that arise when a candidate digit is restricted to the intersection of a Block and a Line (Row or Column). This restriction creates a "lock" that acts as a forceful constraint, propagating eliminations to other parts of the grid.[1, 2]

This report provides an exhaustive, expert-level analysis of the Locked Candidates technique. It will explore the set-theoretic foundations of the strategy, differentiate between its two primary modalities (Pointing and Claiming), analyze the cognitive and algorithmic processes involved in its detection, and examine its relationship to broader Sudoku theory, including "Fish" patterns and Almost Locked Sets.

2 Theoretical Foundations and Set Theory

To understand Locked Candidates at a professional level, one must move beyond the intuitive notion of "scanning" and engage with the underlying set theory that validates the logic. The technique is fundamentally an application of the Principle of Intersection.

2.1 The Geometry of Intersections

A standard Sudoku grid is composed of 27 distinct units (or houses):

- 9 Rows (R_1 to R_9)
- 9 Columns (C_1 to C_9)
- 9 Blocks (B_1 to B_9)

Every individual cell is a member of exactly three units: one row, one column, and one block. Consequently, any given Block intersects with three specific Rows and three specific Columns. These intersections are not merely geometric features; they are shared subsets of cells that link the constraints of the linear units to the constraints of the box units.

Let U_L represent a Line Unit (a specific Row or Column). Let U_B represent a Block Unit. Let $I = U_L \cap U_B$ be the intersection of these two units.

In a standard Sudoku grid, this intersection I always contains exactly three cells (a "min-row" or "min-col").

2.2 The Logic of Mandatory Placement

The validity of the Locked Candidates technique rests on the **Pigeonhole Principle** applied to constraints. The rule of Sudoku states that a specific digit k must appear exactly once in U_L and exactly once in U_B.

The General Theorem of Intersection Removal: If the set of all possible locations for digit k in unit A is contained entirely within the intersection $A \cap B$, then the digit k *must* exist in the intersection. Since the digit must exist in the intersection, and the intersection is a subset of unit B, the digit k cannot exist anywhere in unit B outside of that intersection.

Formally: Let $Candidates_A(k)$ be the set of cells in Unit A that can contain digit k. If $Candidates_A(k) \subseteq (A \cap B)$, then k is "locked" in $A \cap B$. Therefore, $\forall c \in (B \setminus A)$, $k \notin c$.

This theorem gives rise to the two distinct "directions" of the technique, distinguished by which unit acts as the restrictor and which acts as the target for elimination.[2]

2.3 Taxonomy and Nomenclature

In the literature surrounding Sudoku algorithms and heuristic solving, terminology can vary. The distinction is crucial for clarity in technical discussions.

Technical Term	Common Name	Direction	Description

Type 1	**Pointing**	Block → Line	Candidates in a Block are restricted to a single Line. Eliminations occur in the Line outside the Block.
Type 2	**Claiming**	Line → Block	Candidates in a Line are restricted to a single Block. Eliminations occur in the Block outside the Line.
Intersection Removal	Locked Candidates	Bidirectional	The umbrella term covering both Type 1 and Type 2.
Box-Line Reduction	BLR / Claiming	Line → Block	A specific synonym for Type 2, emphasizing the reduction of the box by the line.

3 Locked Candidates Type 1: Pointing

Type 1, widely known as "Pointing," is generally the first intersection technique encountered by solvers. It exploits the human visual tendency to process information within the compact 3×3 boxes before scanning the longer rows and columns.

3.1 Mechanism of Action

The mechanism of Pointing relies on "local" information (within a block) to force a "global" truth (across a row or column).

1. **Observation:** The solver examines a specific Block B.
2. **Constraint Identification:** The solver isolates a specific digit k and identifies all cells in B where k is still a valid candidate.
3. **Pattern Recognition:** The solver notices that all such cells are collinear—they all fall within a single Row R (or Column C).
4. **Inference:** Since B *must* contain k, and the only places for k in B are inside R, the actual solution for k in Row R is forced to be inside Block B.
5. **Elimination:** Consequently, k cannot appear in Row R outside of Block B.

3.2 Detailed Visual Analysis: Pointing Pairs

To illustrate, consider the following grid segment focusing on the top-left area (Blocks 1 and 2, Row 2). We are analyzing the digit **4**.

Visual Legend:

- **{...}**: Pencil marks (candidates).
- *****: Candidates to be eliminated.

	Col 1	**Col 2**	**Col 3**	**Col 4**	**Col 5**	**Col 6**	**Notes**
Row 1	.	.	.	.	.	.	(No 4s in Block 1)
Row 2	**{2,4}**	**{4,5}**	.	{1,4}*	{4,8}*	.	**The Interaction**
Row 3	.	.	.	.	.	.	(No 4s in Block 1)
Block	**Block 1 (Source)**			**Block 2 (Target)**			

Step-by-Step Analysis:

1. **Scan Block 1:** We look for the candidate **4**. We find it in R2C1 **{2,4}** and R2C2 **{4,5}**.
2. **Check Exclusivity:** We scan the rest of Block 1 (Rows 1 and 3). There are no 4s.
3. **The Lock:** The 4s in Block 1 are "locked" into Row 2. They form a **Pointing Pair**.
4. **The Implication:** One of these two cells *must* be the 4 for Block 1. Because they are both in Row 2, one of them must also be the 4 for Row 2.
5. **The Elimination:** We look along Row 2 into Block 2. We see 4s in {1,4} and {4,8} (marked with *). These 4s are "seen" by the lock.
6. **Result:** The 4s in Block 2/Row 2 are removed.

3.3 Detailed Visual Analysis: Pointing Triples

Pointing is not limited to pairs; it often involves three cells.

Scenario B: Vertical Pointing Triple (Digit 5)

Row	Col 4	Col 5 (Line)	Col 6	Status
R1-3	.	**{5,9}***	.	**Block 2 (Target)** - Eliminate 5
R4	.	**{2,5}**	.	**Block 5 (Source)**
R5	.	**{5,8}**	.	**Block 5 (Source)**
R6	.	**{3,5}**	.	**Block 5 (Source)**
R7-9	.	**{1,5}***	.	**Block 8 (Target)** - Eliminate 5

Analysis: The digit 5 is restricted to Column 5 within Block 5. This "Pointing Triple" fires up and down the column, eliminating any 5s in Blocks 2 and 8 along that column.

4 Locked Candidates Type 2: Claiming (Box-Line Reduction)

While Pointing (Type 1) is intuitive, Claiming (Type 2) is often a stumbling block for intermediate solvers. It requires a shift in perspective: instead of looking for what a box tells a line, one must look for what a line tells a box. This technique is frequently called **Box-Line Reduction** (BLR).

4.1 Mechanism of Action

Claiming is the logical inverse of Pointing.

1. **Observation:** The solver examines a specific Line (Row or Column).
2. **Constraint Identification:** The solver isolates a digit k and finds all instances of k within that Line.
3. **Pattern Recognition:** The solver notices that all these instances fall within a single Block B.
4. **Inference:** Since the Line *must* contain k, and the only valid spots are inside Block B, the solution for k in Block B is forced to be on that Line.
5. **Elimination:** Consequently, k cannot appear in Block B outside of that Line.

4.2 Detailed Visual Analysis: Claiming (Row)

Let us examine **Row 5** across Blocks 4, 5, and 6. The digit of interest is **7**.

Row	Block 4	Block 5 (Center)	Block 6	Notes
Row 4	...	{1,7}* {2,7}* {3}	...	**Eliminate 7** (Target)
Row 5	No 7s	**{4,7} {7,8} {7,9}**	No 7s	**The Claimant** (Source)
Row 6	...	{5,7}* {6} {8}	...	**Eliminate 7** (Target)

Step-by-Step Analysis:

1. **Scan Row 5:** We look across the entire row for the candidate **7**.
2. **Observation:** In Block 4 (left) and Block 6 (right), there are no 7s in Row 5. Therefore, *all* 7s for Row 5 are clustered in the middle (Block 5).
3. **The Claim:** Row 5 "claims" the 7 for Block 5.
4. **The Elimination:** Since the 7 for Block 5 is locked to Row 5, it cannot exist in Row 4 or Row 6 of that same block.
5. **Result:** The 7s marked with * in Rows 4 and 6 of Block 5 are eliminated.

4.3 Detailed Visual Analysis: Claiming (Column)

The logic applies vertically. Consider **Column 8** and **Block 6**. Digit **2**.

Row	Col 8 Content	Block	Logic
R1-3	No 2s	Block 3	
R4	**{1,2}**	**Block 6**	The only 2s in Col 8 are here
R5	**{2,5}**	**Block 6**	The only 2s in Col 8 are here
R6	**{2,9}**	**Block 6**	The only 2s in Col 8 are here
R7-9	No 2s	Block 9	

Side View of Block 6 (Target):

	Col 7	Col 8 (Line)	Col 9
Row 4	{2,3}*	**{1,2}**	{4}
Row 5	{5}	**{2,5}**	{2,6}*
Row 6	{8}	**{2,9}**	{7}

Result: Column 8 locks the 2 into Block 6. We eliminate 2s from Col 7 and Col 9 within Block 6 (marked with *).

5 The "Fish" Taxonomy: Advanced Theoretical Categorization

For professional solvers, Locked Candidates are the simplest members of the "Fish" family.

- **Size 1 Fish (Cyclops Fish):** Locked Candidates.
- **Base Set:** 1 Line (e.g., Row 2).
- **Cover Set:** 1 Box (e.g., Block 3).
- **Logic:** All candidates in the Base (Row) are contained in the Cover (Box). Therefore, eliminate candidates in Cover \ Base.

This places Locked Candidates at the foundation of a complexity pyramid:

1. **Size 1:** Locked Candidates (Cyclops Fish).
2. **Size 2:** X-Wing (2 Rows covered by 2 Columns).
3. **Size 3:** Swordfish (3 Rows covered by 3 Columns).
4. **Size 4:** Jellyfish.

6 Interactions with Other Solving Techniques

Relation to X-Wings: Consider a configuration that looks like an X-Wing (a rectangle of candidates) but where the two corners on one side reside in the *same* block. While geometrically an X-Wing, the restriction in the shared block acts as a Pointing Pair. Logic hierarchies prioritize Locked Candidates over X-Wings because they are computationally cheaper.

Almost Locked Sets (ALS): An Almost Locked Set is a set of N cells containing $N+1$ candidates. Locked Candidates are frequently the triggers that reduce an ALS to a Locked Set. For example, a distant Pointing Pair might eliminate a candidate from an ALS, transforming it into a Naked Pair, which then solves the block.

7 Human Factors: Strategies for Detection

- **Cross-Hatching:** Integrate detection into the basic cross-hatching routine. If the open spots for a digit narrow down to a single line, immediate recognition is required.

- **Snyder Notation:** Marking candidates only if there are exactly two possible locations in a block. This makes Pointing Pairs visually "pop" out. However, Snyder notation is notoriously bad at revealing Claiming (Box-Line Reduction) because it is block-centric.
- **Visual Filtering:** Modern apps allow highlighting specific digits. To find Claiming, look at the highlighted cells in each row/column. If they are clustered in one block, check the rest of the block.

8 Algorithmic Implementation

Efficient solvers represent the grid using bitmasks.

- **Complexity:** Intersection Removal is a polynomial-time operation, specifically $O(n^3)$.
- **Bitwise Logic:** Let $Mask_{Row1}$ be the candidates in the intersection of Block B and Row $R1$. Let $Mask_{Other}$ be the candidates in B excluding $R1$. If $(Candidate \in Mask_{Row1}) \wedge (Candidate \notin Mask_{Other})$, then the candidate is locked.

9 Common Pitfalls

The "Reverse Logic" Error: Eliminating the candidate from the wrong unit.

- If the **Block** restricts the candidates, eliminate from the **Line**.
- If the **Line** restricts the candidates, eliminate from the **Block**.

The "Phantom" Lock: Seeing candidates in a line within a block but missing an outlier in that block (for Pointing) or in that line (for Claiming) that invalidates the lock.

10 Conclusion

The Locked Candidates technique is more than a mere trick; it is a fundamental expression of the logical constraints that define Sudoku. Whether viewed through the lens of set theory (Intersection Removal), taxonomy (Cyclops Fish), or cognitive strategy, the core principle remains the same: **Restriction in one unit compels exclusion in another.**

Appendix: Summary Table of Intersection Logic

Feature	**Pointing (Type 1)**	**Claiming (Type 2)**
Primary Unit	Block	Line (Row/Col)
Secondary Unit	Line (Row/Col)	Block
Restriction	Candidates in this **Block** are limited to this **Line**.	Candidates in this **Line** are limited to this **Block**.
Elimination	Remove from **Line** (outside Block).	Remove from **Block** (outside Line).
Visual Cue	A line of candidates inside a box.	A cluster of candidates in a row/col confined to one box area.
Logic Direction	Local $\rightarrow$ Global	Global $\rightarrow$ Local