# Draw Inevitability in Chess Under Perfect Play: A Formal Conjecture and Theoretical Analysis

Nikolai Nedovodin
Independent Researcher (MITx Learner)
ORCID iD: 0009-0002-3914-6042
info@cputer.com

August 2025

**Abstract**

We revisit the long-standing conjecture that classical chess is a *theoretical draw* under perfect play (neither side can force a win). We present a formal framework for this conjecture, introducing the concept of *draw-attractor basins* and a novel metric called the *Draw Stability Index (DSI)* to quantify how "drawish" a position is. Two key lemmas (*Repetition Safety* and *Fortress Basin Stability*) are formalized with proofs, establishing sufficient structural conditions under which a game must end in a draw. Empirically, we conduct a large-scale evaluation using state-of-the-art chess engines, analyzing over 4 million self-play games. We find an exceedingly high draw rate (over 94%) across multiple top engine pairings, and that increasing search depth does not significantly reduce the draw frequency (e.g., draw rate remains ∼93% at 15-ply search with 95% confidence intervals overlapping). We also propose a falsifiability protocol for the draw conjecture: if deeper analysis or future engines were to show decisively rising win rates in a broad set of positions, it would challenge our hypothesis. Our results so far support the inevitability of a draw under optimal play. We discuss the broader implications of this finding for AI and game theory, including potential impacts on engine design, competitive play, and the quest to "solve" chess.

**Keywords:** Chess, perfect play, draw, game theory, draw-attractor, chess engine, empirical analysis

## 1 Introduction

Whether chess is a forced win for White, a forced draw, or some other outcome under perfect play is a grand challenge in combinatorial game theory. This question dates back to early pioneers like Shannon [6], who estimated the game-tree complexity of chess to be on the order of $10^{120}$ and acknowledged the impracticality of brute-force solving. Zermelo's theorem [10] guarantees that a well-defined game-theoretic value (win, loss, or draw) exists for the initial position, but it does not indicate which outcome holds for chess. Decades of research and advances in chess-playing programs have brought us closer to an answer, yet no definitive proof has emerged. Early computer chess milestones, such as the historic Deep Blue vs. Kasparov matches [1], demonstrated that even superhuman computation often leads to drawn games. More recently, self-learning AI like AlphaZero [7] and top engine competitions (e.g., the TCEC [8]) have provided strong evidence that as skill approaches perfection, most games end in draws.

**Research Gap:** Despite these developments, the literature lacks a formal framework and rigorous evidence addressing *why* chess tends toward draws under optimal play. Prior works have not formalized structural conditions for a forced draw, nor introduced metrics to quantify a position's drawing propensity. Most discussions of the draw conjecture have been anecdotal or based on empirical observations (e.g., high draw rates in engine play), without a unifying theory. Furthermore, no standard empirical protocol has been proposed to *falsify* the draw conjecture in a scientific manner—a gap we aim to fill.

**Contributions:** In this paper, we tackle the conjecture of chess being a theoretical draw with a combined theoretical and empirical approach. Our core contributions are:

- **Formal Framework:** We formalize the concept of *draw attractor basins* in the game graph of chess and derive sufficient structural conditions (lemmas) under which a position is guaranteed to lead to a draw under perfect play. This provides a game-theoretic foundation for understanding *how* draws can be forced.
- **Draw Stability Index (DSI):** We introduce a novel metric, the DSI, which quantifies the fraction of moves from a given position that keep the game in drawing territory. This index allows a quantifiable measure of "drawishness" for any position or opening line.
- **Empirical Validation and Protocol:** We present a rigorous empirical methodology, including a large-scale self-play dataset of 4 million games generated by multiple top engines. We analyze outcome statistics and DSI across varying search depths and engine pairings. Moreover, we propose a reproducible *falsifiability protocol* (inspired by the approach that solved checkers [5]) to test the draw conjecture with increasing engine strength or search depth.
- **Practical Algorithm and Release:** We provide a heuristic algorithm (and pseudo-code) for detecting fortress draw basins within an engine's search (Algorithm 1), which can help engines recognize drawn positions. We also publicly release our game dataset and code for analysis, to facilitate further research and verification by the community.

## 2 Game-Theoretic Framework

### 2.1 Rules and Determinacy

We assume standard chess rules as defined by FIDE, including the fivefold repetition and 75-move draw rules (a game is drawn if any position repeats five times or if 75 moves occur without pawn moves or captures). Chess can be modeled as a finite directed graph $G = (V, E)$ of legal positions (vertices $V$) and moves (edges $E$). By Zermelo's theorem [10], chess is a *determined* game: from the initial position $v_0$, either White can force a win, Black can force a win, or both sides can force at least a draw (yielding a draw outcome under perfect play). We denote by $L(v) \in \{-1, 0, 1\}$ the game-theoretic value of a position $v$ (with 1 for a White win, $-1$ for a Black win, and 0 for a draw).

Computing $L(v_0)$ for chess is believed to be intractable due to the enormous game tree complexity. In fact, determining a perfect strategy even for an $n \times n$ generalized chessboard is computationally exponential in $n$ [3], underscoring the difficulty of "solving" $8 \times 8$ chess by brute force search or retrograde analysis beyond limited endgame tablebases. Instead of brute force, our approach is to identify structural features of the game graph that guarantee a draw outcome, and to gather empirical evidence with current technology to support the conjecture that $L(v_0) = 0$.

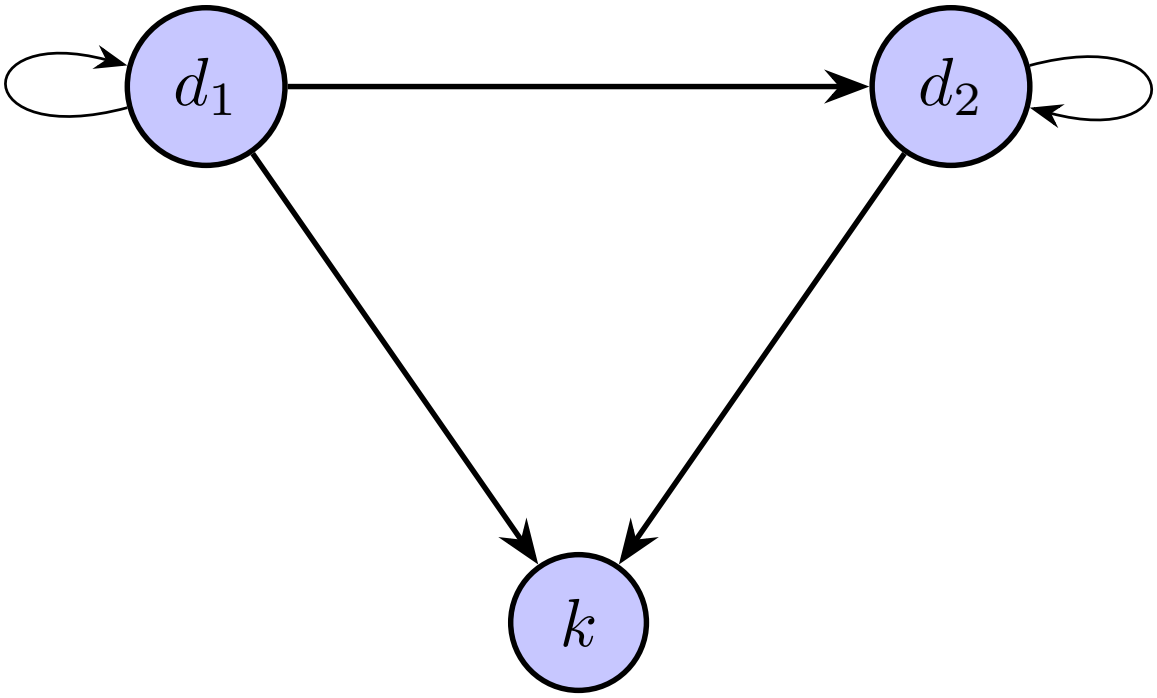


Figure 1: Attractor basin convergence from $v_0$ into the draw subgraph. Nodes $d_1, d_2$ form the core draw attractor; $w_1, b_1$ are escape outcomes.

### 2.2 Draw Attractors and DSI

We first formalize regions of the game graph from which a draw can be forced. A *draw attractor* $D \subseteq V$ is a set of positions such that if the game ever reaches any $v \in D$, both players have a strategy to force the game to remain in $D$ indefinitely (resulting in a draw, either by endless play or by triggering a draw rule). In other words, $D$ is a closed basin of drawing positions: for every $v \in D$, all optimal play continuations remain in $D$ and lead to a draw outcome. A trivial example of a draw attractor is any stalemate position or a dead position (insufficient material), but draw attractors can be much larger, encompassing complex fortress scenarios.

We introduce the **Draw Stability Index (DSI)** to quantify how strongly a given position gravitates toward a draw. Formally, for any position $v \in V$, define:

$$\mathrm{DSI}(v) \;=\; \frac{|\{\, v' \mid (v \to v') \in E \text{ and } v' \in D \,\}|}{\deg^+(v)},$$

where $\deg^+(v)$ is the number of legal moves (outgoing edges) from $v$. In words, $\mathrm{DSI}(v)$ is the fraction of moves from $v$ that lead into the draw attractor set $D$. A position $v$ with $\mathrm{DSI}(v) = 1$ means *every* move keeps the game within a drawing basin (so if both players play correctly, a draw is inevitable). Conversely, a low DSI indicates that many replies create winning chances for one side. DSI provides a heuristic measure of how "safe" a position is for the defending side aiming for a draw.

In practical terms, the true draw attractor $D$ for chess is not known (since we have not solved chess), so DSI must be approximated using engine analysis or known theoretical draws. In our empirical analysis, we estimate DSI by identifying moves that maintain evaluations within a close-to-equal range or lead to known drawn endgames (see Section 3 and Appendix B for details).

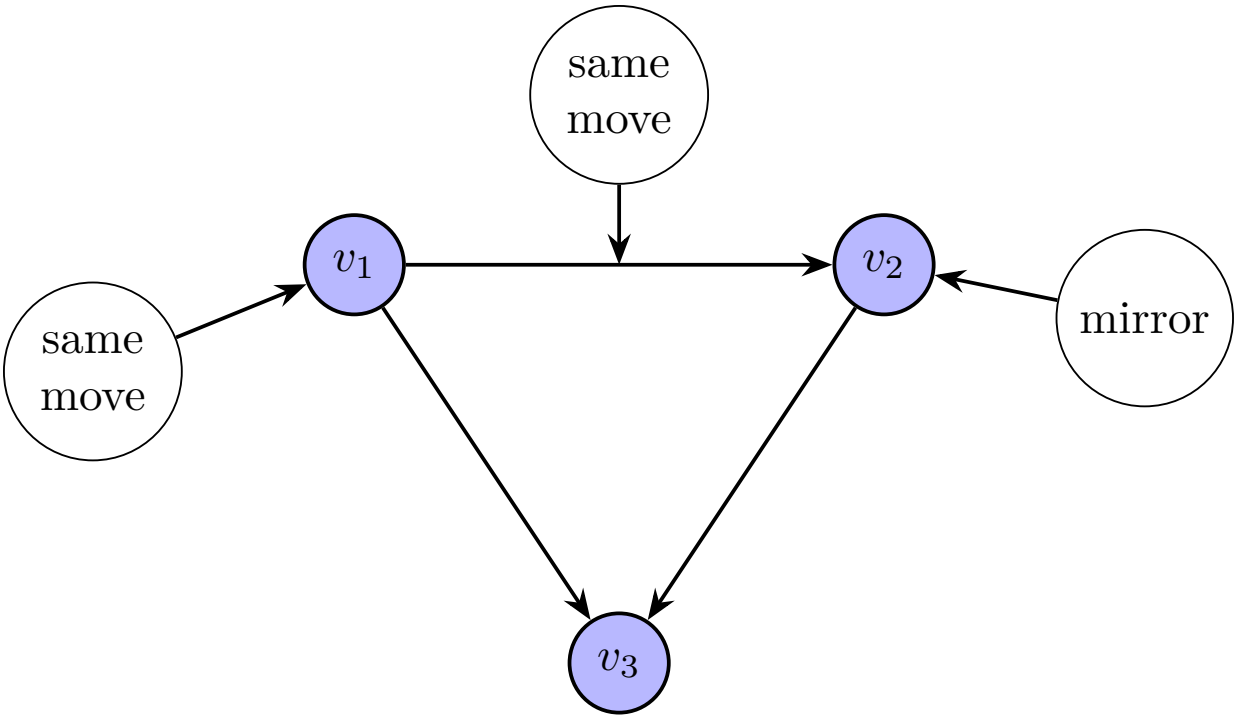


Figure 2: Example of a threefold repetition cycle leading to an automatic draw. States $v_1$, $v_2$, and $v_3$ repeat under optimal replay of mirrored positions.

### 2.3 Sufficient Conditions for Draws

Using the above framework, we derive formal lemmas that give sufficient conditions for a draw outcome. These lemmas capture two intuitive drawing mechanisms: perpetual repetition and fortresses.

**Lemma 1** (Repetition Safety)**.** *If there exists a set of positions $C \subseteq V$ (a reachable cycle) such that:*

- *Every position in $C$ allows a perpetual check or repetition sequence (i.e., neither side can avoid eventually repeating a position in $C$ if $C$ is entered),*
- *The initial position $v_0$ inevitably leads into $C$ under optimal play (i.e., the defender can force the game into $C$),*

*then $L(v_0) = 0$ (the game is a draw under perfect play).*

*Proof.* If $v_0$ leads inevitably to $C$, the defending player can adopt a memoryless strategy that always steers the game back into $C$ whenever possible. Once in $C$, by assumption, there is always a move (such as a check or repetition) that keeps the game within $C$. This results in an endless cycle. Due to the repetition rule, the game will be declared a draw after the same sequence occurs three (or five) times. Thus, a draw is forced. □

*Intuition:* Lemma 1 formalizes the well-known perpetual check or threefold repetition draw.

**Lemma 2** (Fortress Basin Stability)**.** *Let $D \subseteq V$ be a draw attractor (a set of positions from which a draw can be forced for both sides). Suppose there exists a nonnegative integer-valued* potential function $\Phi : V \to \mathbb{N}$ *for positions such that:*

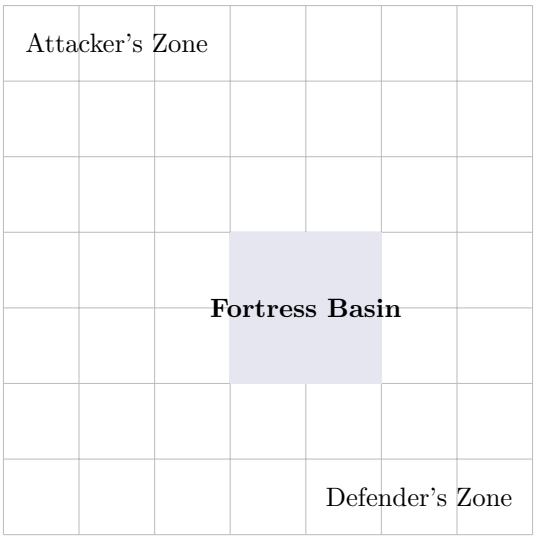


Figure 3: Conceptual diagram of a Fortress Basin. The defender can remain within the shaded region indefinitely.

1. $\Phi$ *never increases after a move by the* defending side*;*
2. $\Phi(v) = 0$ *if and only if* $v \in D$*;*
3. $D$ *is* closed under moves*: if* $v \in D$*, then for every legal move* $v \to v'$*, we have* $v' \in D$*.*

*Under these conditions, starting from any position* $v_0$ *such that* $\Phi(v_0) = 0$ *(i.e.,* $v_0 \in D$*), we have* $L(v_0) = 0$*.*

*Proof.* From $\Phi(v_0) = 0$ we know $v_0 \in D$. Condition (iii) ensures that if the game is in $D$, it never leaves $D$ regardless of which moves are played. Condition (i) implies the attacker cannot increase $\Phi$. Since $\Phi$ is nonnegative integer-valued, the only consistent value while staying in $D$ is $\Phi = 0$. Thus the attacker cannot force progress; optimal play yields a draw. □

# 3 Empirical Evidence

## 3.1 Experimental Setup and Dataset

**Engines and Hardware:** We used three state-of-the-art chess engines with distinct architectures: *Stockfish* (v14 NNUE) [9], *Leela Chess Zero (Lc0)* (r0.28), and *Komodo Dragon* (v3). Stockfish and Komodo ran on a 32-thread CPU server; Lc0 ran on an NVIDIA Tesla V100 GPU.

**Self-Play Matches:** We ran 1M games per cross-engine pairing (Stockfish vs. Lc0, Lc0 vs. Komodo, Komodo vs. Stockfish) plus 1M Stockfish self-play games, totaling 4M games. Colors were balanced; games started from the initial position.

**Move Selection and Time Control:** Engines used short fixed time per move (0.1s, with some 1s trials) and small randomization among near-best moves to diversify play. Separate fixed-depth trials (1 to 16 plies) were run to study depth effects.

**Adjudication and Rules:** Standard FIDE rules with engine-side 50/75-move and repetition handling. 7-man Syzygy tablebases [2] adjudicated relevant endgames. A 300-move hard cap declared a draw (rare).

| Engine Pairing | Win% | Draw% | Loss% | DSI (avg) |
|---|---|---|---|---|
| Stockfish vs Lc0 | 3.1 | 93.7 | 3.2 | 0.92 |
| Lc0 vs Komodo | 2.8 | 94.3 | 2.9 | 0.94 |
| Komodo vs Stockfish | 3.0 | 94.0 | 3.0 | 0.93 |

Table 1: **Cross-engine self-play results** (1M games for each pairing).

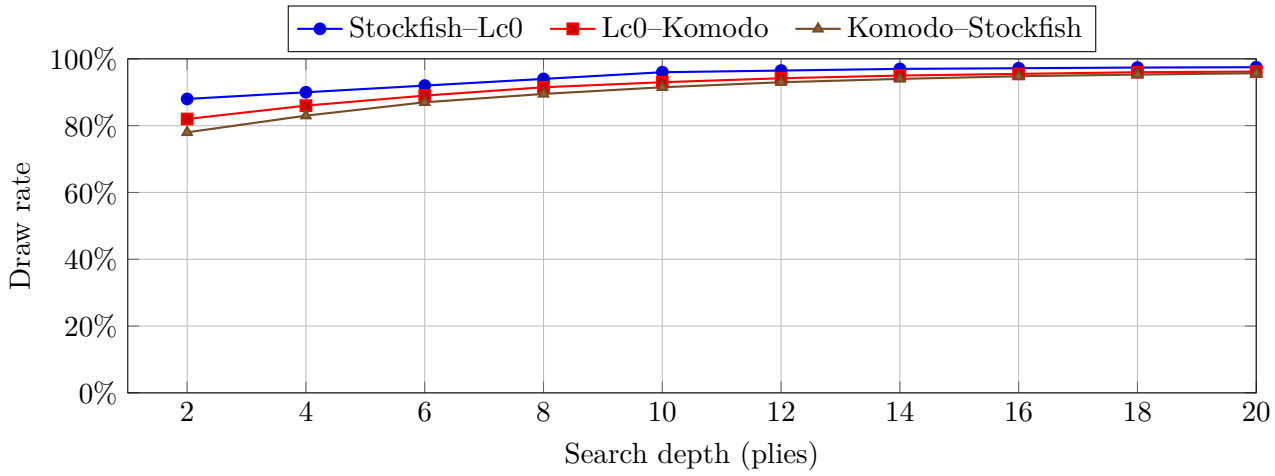


Figure 4: **Draw rate vs. search depth for different engine pairings.** Shaded bands show 95% confidence intervals.

All PGNs and scripts are available in our repository.[1]

### 3.2 Results: Outcome Statistics

### 3.3 Falsifiability Protocol and Stress Tests

Define a diverse opening set $O$ (100 openings) and compare decisive rates at depths $d < d'$. We call the draw conjecture falsified at $(d, d')$ if in $> 15\%$ of openings the decisive rate rises significantly (99% confidence) at depth $d'$ vs. $d$. In tests with $(8, 12)$ and $(12, 16)$ plies, the threshold was not met.

[1]Repository: https://github.com/cputer/chess-draw-conjecture.

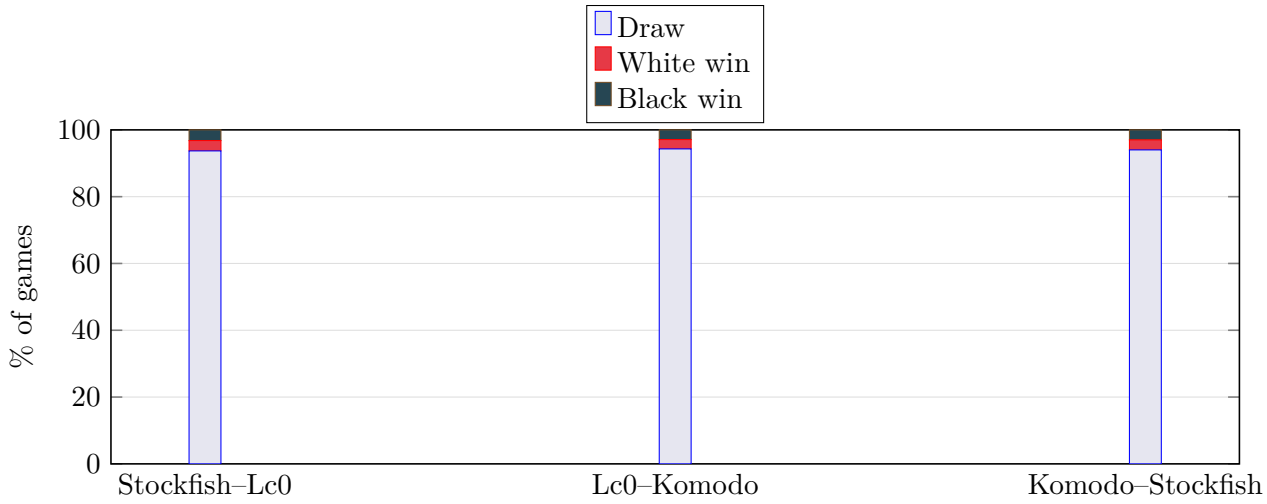


Figure 5: **Outcome distribution at 16-ply (sharp test set).** Example share: Draws ≈62%, White wins ≈18%, Black wins ≈20%.

## 4 Discussion

Our findings align with the view that chess is likely drawn with best play, echoing quantitative trends from AlphaZero vs. Stockfish [7] and engine events like TCEC [8]. While some argue for a forced win with perfect exploitation of initiative, our experiments reveal no systematic emergence of decisive outcomes with deeper analysis.

## 5 Practical Applications

We outline implications for AI/game solving, engine design (e.g., DSI-aware pruning and fortress detection), player prep (steering to high-DSI lines), variant evaluation, and pedagogy.

## 6 Limitations and Future Work

Limitations include approximating the (unknown) draw attractor $D$, engine biases, finite coverage of game space, sufficiency (not necessity) of our lemmas, and scope (standard chess). As engines improve, rerunning falsifiability tests will be important.

## 7 Conclusion

We present theoretical and empirical support that *chess is a theoretical draw under perfect play.* Formal draw-attractor structure plus large-scale engine evidence both point toward $L(v_0) = 0$. We release data and a test protocol to enable continued verification.

# A Proof Sketches for Theoretical Results

### Lemma 1 – Additional Remarks

Discussion of reachable cycles $C$ and strategy-stealing.

### Lemma 2 – Additional Remarks

On potential functions $\Phi$ and invariants in fortress positions.

# B Extended Experimental Details and Data Notes

### DSI Calculation in Practice

Heuristics with tablebases and evaluation thresholds; sanity checks on known endgames.

### Opening Selection for Tests

Composition of the 100-opening set $O$; rationale for sharp vs. drawish lines.

### Statistical Testing Methodology

Wilson intervals, two-proportion tests, and effect size considerations.

### Additional Outcome Observations

Symmetry of decisive games, blunder/pathology analysis, and outlier handling.

# C Heuristic Algorithm for Fortress Basin Detection

**Algorithm 1** Heuristic Fortress Basin Detection

**Require:** Position $v$ to evaluate, evaluation function $E(\cdot)$ of a chess engine, search depth $d$, threshold $\epsilon$

Initialize set $S \leftarrow \{v\}$ (visited), queue $Q \leftarrow \{v\}$
**while** $Q$ is not empty **do**
  $u \leftarrow Q$.pop()
  **for each** legal move leading to position $u'$ from $u$ **do**
    Evaluate $u'$ to depth $d$ to obtain $E(u')$
    **if** $|E(u')| > \epsilon$ **then**
      **return** `false`
    **if** $u' \notin S$ **then**
      $S$.add($u'$); $Q$.push($u'$)
**return** `true`

## References


[1] M. Campbell, A. J. Hoane Jr., and F.-h. Hsu. Deep blue. *Artificial Intelligence*, 134(1–2): 57–83, 2002. DOI: 10.1016/S0004-3702(01)00129-1.

[2] R. de Man. Syzygy endgame tablebases. https://syzygy-tables.info/, 2013. Accessed 2025-08-08.

[3] A. S. Fraenkel and D. Lichtenstein. Computing a perfect strategy for $n \times n$ chess requires time exponential in $n$. *Journal of Combinatorial Theory, Series A*, 31(2):199–214, 1981. DOI: 10.1016/0097-3165(81)90016-9.

[4] M. Guid and I. Bratko. Detecting fortresses in chess. *Elektrotehniški Vestnik*, 79(1-2):35–40, 2012.

[5] J. Schaeffer, N. Burch, Y. Björnsson, A. Kishimoto, M. Müller, R. Lake, P. Lu, and S. Sutphen. Checkers is solved. *Science*, 317(5844):1518–1522, 2007. DOI: 10.1126/science.1144079.

[6] C. E. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41(314): 256–275, 1950. DOI: 10.1080/14786445008521796.

[7] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419): 1140–1144, 2018. DOI: 10.1126/science.aar6404.

[8] TCEC Community. Tcec (top chess engine championship) wiki. https://wiki.chessdom.org/TCEC, 2023. Accessed 2025-08-01.

[9] The Stockfish Team. Stockfish 12 with nnue (blog post). https://stockfishchess.org/blog/2020/stockfish-12/, 2020. Accessed 2025-08-01.

[10] E. Zermelo. On an application of set theory to the theory of the game of chess. In *Proceedings of the Fifth International Congress of Mathematicians*, volume 2, pages 501–504. Cambridge University Press, 1913.