Chess as an Executable Semantic System

An Algebraic Framework for Failure, Repair, and Adversarial Strategy

Adrian Diamond

AAD Systems

adrian@aadsystems.com

November 17, 2026

Abstract

We propose an executable mathematical framework for chess in which positions are represented as structured semantic states, legal moves as constrained state transformations, and strategic vulnerabilities as formally defined failure classes. Rather than treating move selection solely as numerical evaluation, the framework introduces an explicit ontology of chess-relevant conditions, including king safety, material exposure, piece mobility, coordination, pawn structure, and positional constraints.

Within this framework, repair operators act on positions exhibiting specified vulnerabilities, while adversarial operators characterize the opportunities available to opposing players. We propose a compiler that transforms chess notation into canonical board states, constructs semantic representations, evaluates legal successors, and produces candidate moves accompanied by checkable explanations. The framework distinguishes semantic justification from game-theoretic optimality.

The manuscript establishes formal properties of repair composition, adversarial strategy contracts, positional retention and competing objectives, chess-specific semantic footprints, and selective strategy transfer. It also specifies compiler correctness obligations, bounded strategy synthesis and refutation, checked human-facing explanations, and conditional minimax preservation under semantic search filtering.

A preregistrable experimental protocol specifies independent measurements of certificate soundness, transfer coverage, computational cost, move quality, playing strength, and human comprehension. A proposed executable implementation will evaluate these constructions. Chess provides a precise foundation for a broader theory of adversarial computation, with extensions to partial observation and stochastic transitions.

Keywords: Executable mathematics; formal semantics; chess; failure ontologies; repair algebras; adversarial search; state transitions; compiler architecture; explainable decision-making.

Contents

1 Introduction **1**

1.1 Chess as a Formal Computational System . . . 1

1.2 Motivation: Semantic Analysis Alongside Numerical Evaluation . . . 1

1.3 Research Questions and Proposed Contributions . . . 1

1.4 Organization . . . 1

2 Mathematical Preliminaries **1**

2.1 Chess States and Their Representations . . . 1

2.2 Legal Moves and State Transitions . . . 2

2.3 Terminal States and Game-Theoretic Outcomes . . . 2

2.4 Semantic Predicates and Structural Invariants . . . 2

3 An Ontology of Chess Failure Classes **2**

3.1 Failure Predicates . . . 2

3.2 Candidate Failure-Class Families . . . 3

3.3 A Concrete Five-Predicate Ontology . . . 3

3.4 Semantic State Signature . . . 4

3.5 Compound Failures and Interaction Effects . . . 4

4 Repair Algebras for Chess **4**

4.1 Local Repair Operators . . . 4

4.2 Adversarial Robustness . . . 4

4.3 Repair Profiles and Composition . . . 4

4.4 Repair Traces and a Compositional Algebra . . . 5

4.5 Secondary Failures and Adversarial Exploitation . . . 5

4.6 Interacting Repairs and Competing Candidates . . . 5

4.7 Adversarial Persistence and Interaction Effects . . . 6

4.8 Recovery and Strategic Evaluation . . . 7

5 Semantic Adversarial Strategy **7**

5.1 Candidate Moves and Evaluation . . . 7

5.2 Search Filtering and Opponent-Response Analysis . . . 7

5.3 Explanations and Evidence . . . 7

6 Algebraic Constructions of Strategies **7**

6.1 From Individual Moves to Adversarial Policies . . . 7
6.2 Strategy Constructors and Repair Composition . . . 8
6.3 Order-Sensitive Repair Protocols . . . 9
6.4 Counter-Repairs and Bounded Refutations . . . 9
6.5 Phase-Indexed Strategy Expressions . . . 10
6.6 Semantic Abstraction and a Compilation Obligation . . . 10
6.7 Executable Strategy Certificates . . . 10

7 Reusable Strategic Macros and Contractual Composition **11**

7.1 Semantic Macro Contracts . . . 11
7.2 Ordered Macro Composition . . . 11
7.3 Refinement and Safe Substitution . . . 12
7.4 Failure-Introduction Budgets . . . 13
7.5 Noncommutativity and Counter-Repairs . . . 13

8 Semantic Equivalence and Transfer of Certified Strategies **15**

8.1 The Transfer Problem . . . 15
8.2 Contract-Respecting State Relations . . . 15
8.3 Strategy Transfer with an Explicit Simulation Memory . . . 16
8.4 Finite-Horizon Equivalence . . . 17
8.5 Strategic Requirements Beyond Shared Signatures . . . 17

9 Foundational Results **18**

9.1 Lemmas . . . 18
9.2 Theorems . . . 19
9.3 Compositional Results and Certificate Soundness . . . 19
9.4 Interaction and Counterstrategy Results . . . 20
9.5 Strategy-Synthesis and Composition Results . . . 22
9.6 Order, Duality, and Counterstrategy Results . . . 23
9.7 Corollaries . . . 25

10 Selective Transfer of Certified Strategies **25**

10.1 Why Full Equivalence May Be Unnecessary . . . 25
10.2 Directional Contract Observations . . . 25

10.3 A Strategy-Relative Transfer Theorem . . . 26
10.4 Transfer Obligations as a Finite Proof Search . . . 27
10.5 Contract-Sufficient Semantic Footprints and Risk Transfer . . . 28
10.6 Research Consequences and Limitations . . . 29

11 Chess-Specific Semantic Footprints and Dependency Certificates 29
11.1 Candidate Footprints and Their Proof Obligations . . . 29
11.2 A Reachable Mate-in-One Family . . . 30
11.3 Local Tactical Certificates . . . 30
11.4 Finite Dependency Support and Opponent Completeness . . . 31
11.5 Footprint Extraction as a Research Problem . . . 32

12 Multi-Ply Chess Footprints and Adversarial Counter-Repair Closure 32
12.1 Branch-Indexed Footprints and Response Closure . . . 32
12.2 A Concrete Three-Ply Chess Strategy . . . 33
12.3 An Adversarial Counter-Repair: Underpromotion . . . 34
12.4 Footprint Refinement Against Counterexamples . . . 35
12.5 Next Mathematical Questions . . . 36

13 Persistent Positional Repairs and Adversarial Viability 36
13.1 Positional Objectives and Retention Contracts . . . 36
13.2 From Certified Repair to Certified Persistence . . . 37
13.3 A Concrete Positional Repair and Its Counter-Repair . . . 38
13.4 Counter-Repair Certificates and Retention Depth . . . 39
13.5 Finite-Arena Viability and Remaining Research Questions . . . 39

14 Competing Positional Repairs and Multiobjective Adversarial Strategy 40
14.1 Repair Objectives and the Single-Strategy Requirement . . . 40
14.2 Compatible Repair Choices and Delayed Conflicts . . . 42
14.3 A Concrete Chess Conflict: Support Versus Counter-Pawn Pressure . . . 42
14.4 Adversarial Retention Profiles and Pareto Choices . . . 43
14.5 Counter-Repair Refutations and the Next Descent . . . 44

15 Compiler Architecture 45
15.1 Source Semantics and History-Preserving Canonicalization . . . 45

15.2 The Typed Semantic Intermediate Representation 45
15.3 Semantic Lowering and a Commuting-Semantics Result 46
15.4 Independent Checking of Adversarial Portfolio Certificates 46
15.5 The Abstraction Boundary and Independent Verification 47
15.6 Compiler Phases, Failure Modes, and Experimental Boundaries 48

16 Executable Realization: Refinement Stages and Verification Obligations 48
16.1 A Minimal Executable Contract and Its Trust Boundary 49
16.2 Finite Portfolio Synthesis with Positive and Negative Certificates 50
16.3 Implementation Milestones and Acceptance Gates 51
16.4 Regression Corpus, Mutation Tests, and Reproducibility 52
16.5 Initial Chess Testbed and the Boundary to the Application 53

17 Experimental Evaluation and Research Hypotheses 53
17.1 Evaluation Objects, Claims, and Evidence Levels 53
17.2 Benchmark Families and Negative Controls 54
17.3 Falsifiable Hypotheses and Matching Baselines 55
17.4 Engines, Ablations, and Fair Comparison 56
17.5 Metrics, Uncertainty, and Interpretation 56
17.6 Pre-Registration, Stopping Gates, and Reproducibility 57

18 A Human-Executable Chess Reasoning System 58
18.1 Compact Observations and the Information They Omit 58
18.2 A Chess Witness: A Familiar Pattern and an Unseen Counter-Repair 59
18.3 Four-Pass Over-the-Board Reasoning Protocol 60
18.4 Compact Explanations with Explicit Evidence Levels 60
18.5 Learning, Validation, and Semantic Compression 61

19 Generalization Beyond Chess 62
19.1 Domain-Independent Repair Arenas 62
19.2 Alternating Refinement and the Transfer of Repair Guarantees 63
19.3 Information Structure and Strategic Guarantees 64
19.4 Probabilistic Transitions and Composable Risk Budgets 65
19.5 Domain-Specific Modeling Obligations 65
19.6 Generalization and Research Program 66

20 Discussion, Contributions, and Research Outlook **66**

20.1 Principal Mathematical Contributions 67

20.2 Contractual Precision and Chess Meaning 67

20.3 Semantic Footprints and the Reuse of Strategy 68

20.4 A Certifying Compiler Architecture 68

20.5 The Experimental Research Program 68

20.6 Human Chess Reasoning and Broader Adversarial Systems 69

20.7 Research Outlook and Publication Trajectory 69

21 Conclusion **69**

21.1 From Local Repairs to Composable Strategies 70

21.2 Mathematical Contributions and Experimental Realization 70

21.3 The Next Semantic Descent 70

A Notation and Proof Obligations **72**

References **73**

1. Introduction

1.1 Chess as a Formal Computational System

Chess is a finite-action, perfect-information, deterministic, alternating-move adversarial system. Its rules define an exact legal transition relation, while its strategic evaluation remains computationally demanding. This combination makes chess an attractive testbed for executable mathematics and semantic reasoning. The foundational program of computer chess was articulated by Shannon [1]; subsequent work in game AI has explored heuristic search, self-play learning, and general game descriptions [2, 3, 4].

1.2 Motivation: Semantic Analysis Alongside Numerical Evaluation

Modern engines such as Stockfish optimize move selection using efficient search and learned or specialized evaluation mechanisms [5]. This project proposes an additional axis of analysis: a move's explicitly specified effects on vulnerabilities, defensive constraints, and available counterstrategies. Such semantic claims should be checked independently of any explanatory prose.

1.3 Research Questions and Proposed Contributions

The central research question is whether formalized failure-class information can contribute to adversarial move selection without losing strategically critical continuations. The proposed contributions are:

1. A decidable ontology of precisely defined chess-position vulnerabilities.
2. Executable repair and exploitation relations over legal transitions.
3. Conditional correctness results for bounded adversarial analysis.
4. A semantic intermediate representation suitable for a compiler pipeline.
5. An engine-independent experimental framework for comparing search, explanation fidelity, and move quality.

The manuscript develops a connected calculus of chess-semantic repairs, adversarial strategies, strategy transfer, positional retention, and independently checkable certificates. Its experimental section specifies controlled tests of computational efficiency, explanation fidelity, and competitive playing strength.

1.4 Organization

Sections 2 to 6 introduce the formal objects. Section 7 builds reusable strategic contracts, while Section 8 studies their sound transfer to other states. Section 10 relaxes transfer to the alternatives actually required by a selected strategy. Section 9 collects foundational proofs; later sections develop the certificate-aware compiler, implementation plan, evaluation, and human-facing interface, and research program beyond chess.

2. Mathematical Preliminaries

2.1 Chess States and Their Representations

Definition 2.1 (History-aware chess state)**.** *Let $\mathcal{S}$ be the set of valid, legally reachable, history-aware chess states. A state $s \in \mathcal{S}$ contains the board configuration, the player to move, castling and en-passant rights, applicable move counters, and the history necessary to determine repetition-dependent game rules. Let $\pi(s) \in \{\mathrm{W}, \mathrm{B}\}$ be the player to move and $\mathcal{T} \subseteq \mathcal{S}$ the designated terminal states.*

The historical component matters: two identical piece arrangements can differ in legal options or draw eligibility. A canonical implementation must therefore encode or retain all relevant game-state information.

2.2 Legal Moves and State Transitions

Definition 2.2 (Admissible transition)**.** *For every nonterminal s, let $\mathcal{L}(s)$ be its finite set of legal moves. Define a partial function*

$$\delta : \mathcal{S} \times \mathcal{M} \rightharpoonup \mathcal{S}, \qquad \delta(s, m) = s' \quad \textit{iff } m \in \mathcal{L}(s) \tag{1}$$

and s' is obtained by applying m under the rules of chess. Illegal transitions have no defined output.

Terminology. A *ply* is one move made by one player (one "half-move"), corresponding to one transition of δ. For example, the opening sequence 1. e4 e5 consists of two plies: White's move and Black's reply. Hence an h-ply horizon comprises at most h individual state transitions.

2.3 Terminal States and Game-Theoretic Outcomes

The semantic model distinguishes terminal adjudication (checkmate, stalemate, automatic draws, and claimable draws) from heuristic evaluation. Complete state-dependent adjudication is an explicit obligation of the chess kernel in Sections 15 and 16; the mathematical strategy contracts use whichever terminal convention has been specified. Exact game-theoretic values and bounded evaluation estimates are separate objects.

2.4 Semantic Predicates and Structural Invariants

A structural predicate is a decidable statement about a position relative to a specified player. Examples include "a designated piece is attacked and undefended" or "a named square is occupied by a hostile piece." More interpretive concepts, such as "weak king safety," require additional operational definitions before they qualify as predicates in the ontology.

3. An Ontology of Chess Failure Classes

3.1 Failure Predicates

Definition 3.1 (Finite failure ontology)**.** *Let*

$$\mathcal{F} = \{f_1, \ldots, f_n\}, \qquad f_i : \mathcal{S} \times \{\mathrm{W}, \mathrm{B}\} \longrightarrow \{0, 1\},$$

be a finite collection of explicitly specified, computable predicates. For $p \in \{\mathrm{W}, \mathrm{B}\}$, define

$$C(s, p) = \{f_i \in \mathcal{F} \mid f_i(s, p) = 1\}. \tag{2}$$

Each failure label names a precisely testable condition in the selected ontology. The labels provide a common vocabulary for diagnosis, comparison, and subsequent adversarial analysis. Their strategic importance is assessed by the research program of Section 17.

3.2 Candidate Failure-Class Families

Family	Examples requiring precise operational definitions
King safety	Exposure to checks; inadequate legal escape squares; mating threats
Material	Unprotected pieces; threatened material; unfavorable exchanges
Mobility	Trapped or constrained pieces; limited legal moves
Coordination	Insufficient defense or disconnected tactical support
Pawn structure	Isolated, backward, doubled, or vulnerable pawns
Temporal and tactical	Threats requiring immediate responses; forcing sequences

These classes organize the chess ontology. The five executable predicates below provide its initial formal instantiation; further members are added through explicit algorithms and independently checkable contracts.

3.3 A Concrete Five-Predicate Ontology

We now refine five of the structural classes into an executable *prototype* ontology. These predicates describe facts about a position; their interpretation as strategically harmful is a separate, falsifiable hypothesis. All five are specified relative to a fixed player p, even if the opponent is to move.

For a square q, let $\mathrm{occ}_s(q)$ denote the occupying piece or the empty symbol. Let $\mathrm{att}_s(p, q)$ mean that at least one piece belonging to p attacks q according to the ordinary geometric attack relation, including the intervening-piece obstruction rules for sliding pieces and the diagonal attack rules for pawns. The attacking piece is allowed to be pinned: the relation concerns an *attacked square*, not whether a legal capture can actually be made. A separate legal-move kernel handles the latter issue. Let $\bar{p}$ denote the opponent of p.

Definition 3.2 (Concrete structural predicates)**.** *On a valid state s and fixed player p, define:*

$$f_{\mathrm{check}}(s,p) = \mathbf{1}[\mathrm{att}_s(\bar{p}, k_p(s))], \tag{3}$$

$$f_{\mathrm{center}}(s,p) = \mathbf{1}[\exists q \in \{d4, e4, d5, e5\} : \mathrm{occ}_s(q) = \mathrm{pawn}(p),\ \mathrm{att}_s(\bar{p}, q),\ \neg\,\mathrm{att}_s(p, q)], \tag{4}$$

$$f_{\mathrm{major}}(s,p) = \mathbf{1}[\exists q : \mathrm{occ}_s(q) \in \{\mathrm{rook}(p), \mathrm{queen}(p)\},\ \mathrm{att}_s(\bar{p}, q),\ \neg\,\mathrm{att}_s(p, q)], \tag{5}$$

$$f_{\mathrm{shield}}(s,p) = \mathbf{1}\big[\big(p = W \wedge k_p(s) = e1 \wedge \mathrm{occ}_s(f2) \neq \mathrm{pawn}(W)\big) \vee \big(p = B \wedge k_p(s) = e8 \wedge \mathrm{occ}_s(f7) \neq \mathrm{pawn}(B)\big)\big], \tag{6}$$

$$f_{\mathrm{isolated}}(s,p) = \mathbf{1}[\exists\ \textit{pawn of } p \textit{ on file } j : \textit{no pawn of } p \textit{ occurs on either adjacent file } j-1, j+1]. \tag{7}$$

Here $k_p(s)$ is the square of player p's king and $\mathbf{1}[\cdot]$ is the Boolean indicator. A nonexistent adjacent file is simply ignored in the isolated-pawn predicate.

Remark 3.3 (Operational meaning of the five predicates)**.** Each predicate records the exact geometric or structural condition specified in Definition 3.2. In particular, f_{shield} tracks displacement of the named king-cover pawn while its king remains on the home square. This explicit operational meaning makes the classification reproducible and supports subsequent adversarial and empirical analysis.

Example 3.4 (Two repairs of the same central-pawn predicate)**.** Consider the legal position after $1.e4\ e5\ 2.\mathrm{Nf3}$, with Black to move. The white knight attacks Black's pawn on $e5$, and no Black piece defends that pawn. Thus $f_{\mathrm{center}}(s, B) = 1$, while $f_{\mathrm{shield}}(s, B) = 0$.

Both legal responses $2\ldots\mathrm{Nc6}$ and $2\ldots f6$ repair the central-pawn predicate: the knight on $c6$, respectively the pawn on $f6$, geometrically defends $e5$. But the second move vacates $f7$ while Black's king remains on $e8$; therefore f_{shield} changes from 0 to 1 only after $2\ldots f6$. Thus the two legal repairs have distinct, explicitly computable *semantic effects*; the selected predicates distinguish their secondary structural consequences.

3.4 Semantic State Signature

Definition 3.5 (Semantic signature)**.** *For a fixed ordering of the failure predicates, define*

$$\Phi(s,p) = \big(f_1(s,p),\ldots,f_n(s,p)\big) \in \{0,1\}^n. \tag{8}$$

The signature is a projection of the complete chess state onto the selected ontology. It can group different positions under one description, while the legal-move kernel retains the full state required for precise adversarial analysis.

3.5 Compound Failures and Interaction Effects

Failure predicates can interact: repairing material exposure may alter king safety or mobility. The interaction energy of Definition 4.7 and its monotonicity theorem, Theorem 9.16, provide a formal method of expressing and analyzing such relationships.

4. Repair Algebras for Chess

4.1 Local Repair Operators

Definition 4.1 (Local repair)**.** *Fix a legal move $m \in \mathcal{L}(s)$, let $p = \pi(s)$, and put $s' = \delta(s,m)$. The move locally repairs $f \in \mathcal{F}$ when*

$$R_f(s,m) \iff f(s,p) = 1 \ \wedge\ f(s',p) = 0. \tag{9}$$

The local relation isolates an exact predicate change. The guarded operation in Definition 4.5 adds persistence conditions, and Sections 6 and 13 develops the adversarial guarantees required for strategic use.

4.2 Adversarial Robustness

Definition 4.2 (One-response robust repair)**.** *Let $s' = \delta(s,m)$, with $p = \pi(s)$. For a nonterminal s', define*

$$R_f^{(1)}(s,m) \iff R_f(s,m) \ \wedge\ \forall r \in \mathcal{L}(s') :\ f\big(\delta(s',r),p\big) = 0. \tag{10}$$

Terminal successors are evaluated separately using their game outcomes.

The quantifier represents adversarial response: the repaired predicate remains false against every legal immediate reply. Generalizations to alternating quantifiers over longer continuations are considered in Section 9.

4.3 Repair Profiles and Composition

Definition 4.3 (Semantic repair profile)**.** *For legal (s,m) and fixed player p, define*

$$\Delta(s,m,p) = \Phi(s,p) - \Phi\big(\delta(s,m),p\big) \in \{-1,0,1\}^n. \tag{11}$$

Positive components signal eliminated conditions; negative components signal newly introduced conditions. A sequence of legal moves composes as a sequence of transitions. A nontrivial *repair algebra* must additionally specify operations on repair objects, composability conditions, and properties not implied merely by function composition.

4.4 Repair Traces and a Compositional Algebra

A single net repair profile may conceal a failure that arises and disappears along a line of play. To retain this information, fix a player p, an ordered finite ontology, and a legal finite path $\gamma = (s_0, m_1, s_1, \ldots, m_k, s_k)$, with $s_i = \delta(s_{i-1}, m_i)$. Define the *gross repair count* and the *gross introduction count* for each predicate f_j by

$$G_j^-(\gamma, p) = \sum_{i=1}^{k} \mathbf{1}[f_j(s_{i-1}, p) = 1 \wedge f_j(s_i, p) = 0], \tag{12}$$

$$G_j^+(\gamma, p) = \sum_{i=1}^{k} \mathbf{1}[f_j(s_{i-1}, p) = 0 \wedge f_j(s_i, p) = 1]. \tag{13}$$

Let $G^-(\gamma, p), G^+(\gamma, p) \in \mathbb{N}^n$ denote the corresponding vectors, and define the *gross repair ledger*

$$\Lambda_p(\gamma) = \big(G^-(\gamma, p), G^+(\gamma, p)\big) \in \mathbb{N}^n \times \mathbb{N}^n. \tag{14}$$

Here the minus superscript denotes *removed* failures; the plus superscript denotes *introduced* failures. This convention aligns with the signed profile Δ.

The empty path 1_s has ledger $(0, 0)$. Two paths $\gamma : s \to t$ and $\eta : t \to u$ compose by concatenation, and their ledgers are combined by componentwise addition. This makes the ledgers a commutative monoid under addition; the actual *paths* are partial, ordered compositions and generally do not commute.

Remark 4.4 (Why preserve both counts?)**.** A path may first introduce and then repair the same failure. Its net semantic signature then shows no change, yet the gross ledger records both events. For verification and human-facing explanations, an optional *evidence trace* also records each state hash, legal move, predicate identifier, and observed before/after truth values.

4.5 Secondary Failures and Adversarial Exploitation

A move can repair one structural predicate while introducing another. The repair/introduction sets of Definition 4.6 record both effects; counterresponse witnesses in Definition 4.8 identify legal replies that defeat a guarded repair. These objects supply the formal basis for analyzing interacting repairs.

4.6 Interacting Repairs and Competing Candidates

The same failure may admit multiple repairs, and two repairs may disagree about which new vulnerabilities are acceptable. A move can also cause a predicate to become false by eliminating the very object that was at risk. For example, an attacked pawn that is captured no longer qualifies as an attacked pawn; this is not in itself a successful *repair of that pawn*. The following definitions separate factual predicate changes from the preservation requirements relevant to a particular repair.

Definition 4.5 (Guarded local repair)**.** *Let $p = \pi(s)$, $m \in \mathcal{L}(s)$, $s' = \delta(s, m)$, and let $g : \mathcal{S} \times \{\mathrm{W}, \mathrm{B}\} \to \{0, 1\}$ be a decidable* persistence guard*. A guarded repair of failure f is defined by*

$$R_{f;g}(s, m) \iff g(s, p) = g(s', p) = 1 \ \wedge \ f(s, p) = 1 \ \wedge \ f(s', p) = 0. \tag{15}$$

A guard might assert that a named pawn remains on its designated square, or that a named piece is still present. It prevents an unrelated disappearance of the target from being mistaken for a repair. A guard is an additional specification, not automatically supplied by a failure predicate.

Definition 4.6 (Repair and introduction sets)**.** *For $m \in \mathcal{L}(s)$ and fixed p, write $x = \Phi(s,p)$ and $y_m = \Phi(\delta(s,m),p)$. Define*

$$D^-(s,m,p) = \{i : x_i = 1,\ (y_m)_i = 0\}, \tag{16}$$

$$D^+(s,m,p) = \{i : x_i = 0,\ (y_m)_i = 1\}. \tag{17}$$

For two candidates $m, n \in \mathcal{L}(s)$ sharing an origin, we say that m weakly semantically dominates *n when*

$$D^-(s,m,p) \supseteq D^-(s,n,p), \qquad D^+(s,m,p) \subseteq D^+(s,n,p). \tag{18}$$

Any required guards must hold for both candidates. The comparison is strict *if at least one inclusion is strict.*

This yields an ontology-relative partial order of repair candidates. Incomparable moves can emphasize different objectives; their tradeoffs are represented by interaction energies and the multiobjective portfolios of Section 14.

Definition 4.7 (Quadratic failure-interaction energy)**.** *For $x \in \{0,1\}^n$, weights $w \in \mathbb{R}^n$, and a symmetric collection of pair coefficients $J_{ij} \in \mathbb{R}$ $(i < j)$, let*

$$H_{w,J}(x) = \sum_{i=1}^{n} w_i x_i + \sum_{1 \le i < j \le n} J_{ij} x_i x_j. \tag{19}$$

The pair terms express explicitly specified interactions among failures: positive coefficients penalize coexistence; negative coefficients permit specified offsets. Coefficients are modeling choices, not proven chess values. The semantic gain of candidate m is $H_{w,J}(\Phi(s,p)) - H_{w,J}(y_m)$.

Definition 4.8 (Adversarial counterresponse witness)**.** *Assume $R_{f;g}(s,m)$ and that $s' = \delta(s,m)$ is nonterminal. For each legal opponent reply $r \in \mathcal{L}(s')$ let $s_r = \delta(s',r)$. Define the* counterresponse set

$$\mathsf{K}_{f;g}(s,m) = \{r \in \mathcal{L}(s') : g(s_r,p) = 0 \ \vee \ f(s_r,p) = 1\}. \tag{20}$$

An element of this set is a concrete witness that a guarded repair does not persist through one legal opponent reply. A full executable checker must generate $\mathcal{L}(s')$ independently and distinguish terminal states.

Example 4.9 (Competing repairs after 1.$e4$ $e5$ 2.Nf3)**.** Fix Black's player perspective and restrict attention to the two predicates $(f_{\text{center}}, f_{\text{shield}})$ of Definition 3.2. The origin has signature $(1,0)$. The legal candidate $2\ldots\text{Nc6}$ produces $(0,0)$, whereas $2\ldots f6$ produces $(0,1)$: both geometrically protect the pawn on $e5$, but the latter also vacates the $f7$ square. Under the guard "Black's pawn remains on $e5$," both constitute guarded local repairs of the central-pawn predicate. On this *two-predicate projection*, Nc6 strictly semantically dominates $f6$. The resulting strict dominance is an exact, checkable relationship under the specified two-predicate ontology.

4.7 Adversarial Persistence and Interaction Effects

Even a guarded repair at depth one can be refuted on the next ply. We therefore distinguish local success, persistence against every reply, and longer-horizon evaluations. The counterresponse set in Definition 4.8 supplies checkable witnesses for failure of one-reply persistence. Quantitative interaction energies furnish an additional semantic comparison layer alongside exact legal move generation and state-dependent game outcomes.

4.8 Recovery and Strategic Evaluation

Local repair predicates, guarded repairs, and adversarial search furnish complementary levels of analysis. The latter distinguishes feasible recovery, retention, and the exact outcomes of specified bounded games. The framework therefore supports examination of defensive resources and critical opponent responses under explicit strategic contracts.

5. Semantic Adversarial Strategy

5.1 Candidate Moves and Evaluation

Definition 5.1 (Semantic move selection)**.** *Let $V(s, m)$ be a specified side-to-move evaluation of the legal move m in state s, potentially incorporating semantic repair profiles, tactical facts, opponent replies, and numerical search values. Define*

$$\mathrm{Select}(s) \in \arg\max_{m \in \mathcal{L}(s)} V(s, m) \tag{21}$$

for nonterminal states with nonempty move sets. Tie breaking is an implementation policy.

The main research task is to define a useful V, not merely to write an arg max expression. Evaluation and move ordering should distinguish provable properties from heuristics.

5.2 Search Filtering and Opponent-Response Analysis

A semantic filter $Q(s) \subseteq \mathcal{L}(s)$ may prioritize or remove moves using failure-class information. Prioritization is less risky than irreversible pruning: a move that does not immediately repair a diagnosed failure might still be game-theoretically optimal. The conditional result in Theorem 9.6 makes the preservation requirement explicit.

5.3 Explanations and Evidence

The output of a decision should distinguish a checkable factual claim (e.g., a piece is attacked), a bounded-search conclusion (e.g., no legal opponent reply at depth one restores a predicate), and a heuristic strategic recommendation. This distinction is relevant to research on the alignment between chess concepts and machine representations [6].

6. Algebraic Constructions of Strategies

6.1 From Individual Moves to Adversarial Policies

A local repair describes one transition. A *strategy* must also prescribe what happens after an opponent responds. Accordingly, this section treats a strategy as a policy over a tree of legal histories rather than as a fixed sequence of moves. All results in this section are statements about the specified alternating transition system; they do not assume that the prototype ontology is sufficient to decide chess strategy.

Definition 6.1 (Contingent strategy and outcome paths)**.** *Fix a player p and a finite horizon h. A (possibly history-dependent) p-*strategy *σ assigns one legal move to each nonterminal history of fewer than h plies ending at a state s with $\pi(s) = p$. The opponent is not constrained by σ. Write $\mathrm{Out}_h(s, \sigma; B)$ for all maximal legal continuations of length at most h consistent with these choices, stopping at the* first *visit to target B or at*

game termination. The policy need only specify choices prior to that stopping event.

Definition 6.2 (Guarded strategy objective)**.** *For a fixed player p, choose a decidable safety set $A \subseteq \mathcal{S}$ and a decidable target set $B \subseteq A$. An outcome path satisfies the h-ply objective $\mathsf{Goal}_h(A, B)$ when it remains in A and visits some state in B after at most h transitions. A p-strategy* guarantees *this objective from s when every path in $\mathrm{Out}_h(s, \sigma; B)$ satisfies it. For a guarded repair, one may take $B = \{s : g(s, p) = 1,\ f(s, p) = 0\} \cap A$, while A may prohibit specified secondary failures throughout the attempted repair.*

Definition 6.3 (Controllable predecessor and bounded reachability)**.** *For $X \subseteq \mathcal{S}$ define*

$$\mathsf{CPre}_p(X) = \left\{ s \in \mathcal{S} \setminus \mathcal{T} : \begin{cases} \exists m \in \mathcal{L}(s) : \delta(s, m) \in X, & \pi(s) = p, \\ \forall m \in \mathcal{L}(s) : \delta(s, m) \in X, & \pi(s) \neq p. \end{cases} \right\}. \tag{22}$$

All nonterminal states are assumed to have a nonempty legal move set. The h-ply guaranteed repair region *is defined recursively by*

$$\mathsf{W}_p^0(A, B) = A \cap B, \tag{23}$$

$$\mathsf{W}_p^{h+1}(A, B) = A \cap \big(B \cup \mathsf{CPre}_p(\mathsf{W}_p^h(A, B))\big). \tag{24}$$

At a terminal state outside B, the target is not attained; a terminal state in B succeeds immediately. Each region is exact for the stated bounded reachability *objective, not for perfect-play chess outcomes.*

Remark 6.4 (An opponent may undo a locally successful repair)**.** The goal B only requires that repair be achieved once. To demand persistence through a further opponent move, replace it by a stricter target, for example

$$B_{p,A}^{[1]} = \{t \in A \cap B :\ t \in \mathcal{T} \ \vee\ (\pi(t) = \bar{p} \ \wedge\ \forall r \in \mathcal{L}(t),\ \delta(t, r) \in A \cap B)\}. \tag{25}$$

Here $\bar{p}$ is the other player. The target therefore includes positions that survive every immediate opposing reply (or have already terminated). Longer persistence periods require additional conditions.

6.2 Strategy Constructors and Repair Composition

We distinguish three operations on strategy specifications: *choice* of a legal move at our nodes, *adversarial closure* over all legal replies at opponent nodes, and *guarded sequential composition* of strategies that share a compatible intermediate target. Unlike the gross repair ledger, which is an additive commutative summary, these constructors retain the branching and order of play.

Definition 6.5 (Guarded sequential composition)**.** *Let (A, B) and $(A \cap B, B \cap C)$ be two repair objectives. A first strategy reaches B while maintaining A; a second, started from an intermediate state in $A \cap B$, reaches $B \cap C$ while maintaining both A and B. Their* guarded sequential composition *executes the first policy until its first entry into B, then switches to the second. The switch is permitted only at states where the second policy is defined.*

Definition 6.6 (Guaranteed repair depth)**.** *For a fixed (p, A, B) define*

$$\rho_{p;A,B}(s) = \min\{h \in \mathbb{N} :\ s \in \mathsf{W}_p^h(A, B)\}, \tag{26}$$

with $\rho = \infty$ if no finite bound exists. This value measures the number of plies *needed to force a specified semantic target, not the number of moves needed to force a chess win.*

6.3 Order-Sensitive Repair Protocols

The net profile Δ and gross ledger Λ_p from Definition 4.3 and Equation (14) retain useful numerical information, but neither records the order in which structural conditions appeared. This is important whenever a strategy must satisfy an invariant *at every intermediate state* rather than only at the end.

Definition 6.7 (Ordered semantic trace and safety envelope)**.** *For a legal finite path $\gamma = (s_0, m_1, \ldots, m_k, s_k)$, with $s_i = \delta(s_{i-1}, m_i)$, and fixed player p, define*

$$\tau_p(\gamma) = \big(\Phi(s_0, p), \Phi(s_1, p), \ldots, \Phi(s_k, p)\big). \tag{27}$$

*For a decidable safety set $A \subseteq \mathcal{S}$, the path is A-*safe *if $s_i \in A$ for all i. An ordered* repair protocol *specifies an initial state, a legal path or contingent policy, an intermediate-state safety requirement, and a terminal goal. The full-state path, rather than its projected semantic trace, remains the authoritative record for legal-move checking.*

The ordered trace has a natural typed concatenation: if the last state of γ is the first state of η, their composition $\gamma \cdot \eta$ joins the paths and removes the duplicate boundary state. The ledger is additive under this composition by Theorem 9.8; however it is a *lossy* abstraction of the trace.

Example 6.8 (Identical repairs, different intermediate safety)**.** Consider a two-predicate *abstract semantic test system*, not a concrete chess position. From $(1, 1)$, operator a clears the first predicate and operator b clears the second. Both are idempotent and commute on final signatures. Yet their ordered traces are

$$\tau(a; b) = ((1, 1), (0, 1), (0, 0)), \tag{28}$$
$$\tau(b; a) = ((1, 1), (1, 0), (0, 0)). \tag{29}$$

Both produce the same net profile and gross ledger, namely two failures removed and none introduced. If the safety envelope excludes the intermediate signature $(0, 1)$ but permits the other three displayed signatures, then $b; a$ is safe and $a; b$ is not. This is an illustration of a *path constraint*; it does not imply that clearing a failure in chess automatically qualifies as a legal move.

Example 6.9 (A legal chess move-order prerequisite)**.** From the standard starting position, the White king's bishop on $f1$ cannot play Bc4 immediately because the pawn on $e2$ occupies its diagonal. After $1.e4\ e5$, the move 2.Bc4 is legal. Thus the same proposed developing action becomes admissible after the prerequisite pawn advance. A genuine chess-level strategy compiler must verify such enabling conditions against the complete legal-state representation; a failure signature alone is insufficient.

6.4 Counter-Repairs and Bounded Refutations

The counterresponse set in Definition 4.8 witnesses failure of *one-reply* persistence. Its strategy-level counterpart records a contingent opponent plan that can refute a claimed bounded guarantee.

Definition 6.10 (Bounded counterstrategy)**.** *Fix (p, A, B, h) as in Definitions 6.2 and 6.3, with $B \subseteq A$. A* bounded opponent counterstrategy *from s is a policy for $\bar{p}$ such that, against every choice made by player p, the resulting legal continuation either exits A before reaching B, or fails to reach B within h plies (including premature termination outside B). Counterstrategies and repair strategies are both quantified over* all *legal replies of the other player. The policy is not a claim about which opponent move is most likely in practical play.*

For the duality result below, let $N = \mathcal{S} \setminus \mathcal{T}$ denote the nonterminal states and let $\mathsf{CPre}_{\bar{p}}$ be the same controllable-predecessor operator with protagonist and antagonist swapped. The complement $\mathcal{S} \setminus X$ includes terminal states, although all predecessor operators exclude terminal origins.

6.5 Phase-Indexed Strategy Expressions

An ordered list of stage objectives can be viewed as a simple strategy *program*: a stage advances only when its current guarded target is first reached. The phase index is part of the program state; without it, two occurrences of the same board state can incorrectly be treated as interchangeable even though different obligations remain.

Definition 6.11 (Three-stage guarded strategy expression)**.** *Let $B, C, D \subseteq A$ and let $h, k, \ell \in \mathbb{N}$. A three-stage expression is an ordered triple of policies $\sigma_1; \sigma_2; \sigma_3$ with the successive obligations*

$$(A, B), \qquad (A \cap B, B \cap C), \qquad (A \cap B \cap C, B \cap C \cap D). \tag{30}$$

Each policy runs until the first *satisfaction of its designated target, then transfers control at the exact reached state to the next policy. A valid expression has defined policies at all reachable handoff states. Its semantics is the set of legal outcome traces with phase tags, not merely the untagged final state.*

The notation ";" denotes sequential strategy execution, not the commutative addition used for repair ledgers. Its meaning depends on which phase is active and on the opponent's intervening moves.

6.6 Semantic Abstraction and a Compilation Obligation

Definition 6.12 (Turn-respecting, transition-stable abstraction)**.** *Let $\alpha : \mathcal{S} \to Q$ be a finite semantic abstraction, such as $\alpha(s) = \Phi(s, p)$ enriched with turn and terminality information. It is* transition-stable *on a considered arena if $\alpha(s) = \alpha(t)$ implies that s, t have equal player-to-move and terminal status and, when nonterminal,*

$$\{\alpha(\delta(s, m)) : m \in \mathcal{L}(s)\} = \{\alpha(\delta(t, n)) : n \in \mathcal{L}(t)\}. \tag{31}$$

*The sets retain all distinct abstract successors; multiplicities are not needed for the existential/universal reachability objective. A subset $X \subseteq \mathcal{S}$ is α-*measurable *when membership is constant on every abstraction class.*

The basic five-predicate chess signature has *not* been shown to be transition-stable; chess legality and move choice must still be computed from the concrete state.

6.7 Executable Strategy Certificates

A bounded strategy certificate is an AND/OR proof tree. At a player-p node it records *one* selected legal transition (OR); at an opponent node it records *every* legal transition (AND). Its leaves are either states satisfying B or failures of the certificate check. The verifier must regenerate moves from a trusted transition kernel rather than trust an unverified move list supplied by the certificate. A dual refutation certificate uses all player moves and one opponent counter-move per node.

Example 6.13 (A repair that survives the opponent's answer)**.** Consider an *abstract turn-based test arena*, not a particular chess position. At our root state there are two options, named `fragile` and `durable`. After `fragile`, the opponent can choose a reply leading outside the safety region A. After `durable`, every opponent reply reaches one of two states in the target B while remaining in A. Thus the immediate semantic effects need not settle which strategy is guaranteed to work: $\mathsf{W}^2_p(A, B)$ includes the root by virtue of `durable`,

with the opponent's alternatives checked universally. The included Python prototype constructs and verifies this certificate.

7. Reusable Strategic Macros and Contractual Composition

The preceding sections define guarded repair, phase-indexed execution, and bounded adversarial strategy synthesis. The next question is how to treat an entire repair–counter-repair plan as a *reusable mathematical unit* rather than re-derive its internal moves whenever a comparable position is encountered. We introduce a contract calculus for such units. Its composition principles are related to traditional precondition–postcondition reasoning [7] and adversarial synthesis [8]; no claim of priority for the general idea of contract composition is intended.

7.1 Semantic Macro Contracts

Definition 7.1 (Certified strategic macro)**.** *Fix a player $p \in \{\mathrm{W}, \mathrm{B}\}$. A* certified strategic macro *is a tuple*

$$\mathfrak{M} = (p, P, A, G, I, h, \sigma) \tag{32}$$

with an entry set $P \subseteq A \subseteq \mathcal{S}$, a target set $I \subseteq G \subseteq A$, a nonnegative integer horizon h, and a history-dependent p-strategy σ such that, for every $s \in P$ and all legal opponent choices, the resulting play:

(i) stays inside A until the macro completes;

(ii) reaches G within at most h chess plies; and

(iii) reaches G for the first time *in a state belonging to I.*

The first target visit is the release event*. The set I is the* release interface*: a smaller certified subset of the target G. The policy must be defined at every player-p decision reached before release. A terminal state outside G is a failure, while a terminal state inside I can complete a final macro.*

Remark 7.2 (Why distinguish a target from an interface?)**.** A move sequence may be able to reach a class of repaired states G while only a subset I is appropriate for its next strategic stage. Requiring that the *first* target state belongs to I makes the handoff obligation explicit. A certified macro is thus stronger than merely asserting $P \subseteq \mathsf{W}_p^h(A, G)$; it also certifies where its first successful exit can occur. In chess, all sets are interpreted over complete legal states, not merely the Boolean signature Φ.

Definition 7.3 (Actual handoff set)**.** *The* actual handoff set *of $\mathfrak{M}$ is*

$$\mathsf{H}(\mathfrak{M}) = \{t \in \mathcal{S} : \exists s \in P \text{ and a legal outcome of } \sigma \text{ whose first visit to } G \text{ is } t\}. \tag{33}$$

Certification gives $\mathsf{H}(\mathfrak{M}) \subseteq I \subseteq G$. For a specified initial state s we also write $\mathsf{H}_s(\mathfrak{M})$ for the corresponding restricted set.

7.2 Ordered Macro Composition

Two macros need not be composable. The second may be unavailable at some state that the first can release, or it may violate a guard preserved by the first. Throughout the following construction the macros have the same protagonist p and a common safety envelope A; different-envelope contracts can be compared only after their safety obligations are reconciled.

Definition 7.4 (Compatible macro program)**.** *Let*

$$\mathfrak{M} = (p, P, A, G, I, h, \sigma),$$
$$\mathfrak{N} = (p, Q, A, H, J, k, \nu)$$

be certified macros. They are reachably composable *if*

$$\mathsf{H}(\mathfrak{M}) \subseteq Q. \tag{34}$$

The phase-indexed macro program $\mathfrak{M};\mathfrak{N}$ *runs σ until its release event, then runs ν from the concrete release state with a fresh phase-local memory. The phase switch consumes no chess ply. The program finishes when $\mathfrak{N}$ releases into J. The internal phase label distinguishes an intermediate visit to H from completion of the second stage. Hence the program is a strategy on an augmented state space $\mathcal{S} \times \{1, 2\}$, projected to legal chess transitions on $\mathcal{S}$.*

Theorem 7.5 (Reachable-handoff composition)**.** *If $\mathfrak{M}$ and $\mathfrak{N}$ are certified and satisfy Equation (34), then $\mathfrak{M};\mathfrak{N}$ is a certified phase-indexed strategy program from every $s \in P$: every complete outcome stays within A, completes the second phase in J, and uses no more than $h + k$ chess plies.*

Proof. Fix $s \in P$ and an arbitrary sequence of legal opponent choices. Certification of $\mathfrak{M}$ forces its first release into some $t \in \mathsf{H}_s(\mathfrak{M}) \subseteq Q$ in at most h plies, with all visited states in A. By certification of $\mathfrak{N}$, its policy can be started from that precise history-aware state t and, against every subsequent opponent choice, stays within A and releases into J in at most k further plies. Concatenating the two segments preserves A; the number of chess transitions is bounded by $h + k$. The zero-time phase switch does not alter the chess state or its move counters. Since the opponent choices were arbitrary, the guarantee holds for all complete outcomes. □

Corollary 7.6 (Composability by declared interface)**.** *The easily checked contract condition $I \subseteq Q$ is sufficient for $\mathfrak{M};\mathfrak{N}$ to be valid. It is not necessary: the weaker condition $\mathsf{H}(\mathfrak{M}) \subseteq Q$ suffices.*

Proof. By Definition 7.3, $\mathsf{H}(\mathfrak{M}) \subseteq I$. Thus $I \subseteq Q$ implies Equation (34), and Theorem 7.5 applies. □

7.3 Refinement and Safe Substitution

The release interface is especially useful when a better strategic unit is discovered. Rather than re-prove the complete surrounding plan, we may replace one certified unit by another satisfying a stronger contract.

Definition 7.7 (Macro contract refinement)**.** *For two certified macros for the same player, write*

$$\mathfrak{M} = (p, P, A, G, I, h, \sigma),$$
$$\mathfrak{M}' = (p, P', A', G', I', h', \sigma').$$

We say $\mathfrak{M}' \preceq \mathfrak{M}$ when

$$P \subseteq P', \qquad A' \subseteq A, \qquad I' \subseteq I, \qquad h' \le h. \tag{35}$$

The refined macro accepts at least the original entry states, respects an at least as restrictive safety envelope, releases into a subset of the original allowed interface, and uses no larger worst-case horizon. There is no requirement that the two policies use the same moves.

Theorem 7.8 (Contextual substitution of certified macros)**.** *Consider a finite compatible macro program whose adjacent stages obey the sufficient declared-interface rule in Corollary 7.6, and suppose all stages remain in a common global safety set A. Replacing any stage $\mathfrak{M}$ by a certified refinement $\mathfrak{M}' \preceq \mathfrak{M}$ preserves the existence of a valid phase-indexed program from the original entry set, its global safety guarantee, and its final release-interface obligation. Its worst-case horizon does not increase.*

Proof. The preceding stage, if present, releases into its declared interface, which was contained in P; by $P \subseteq P'$ this is also a legal entry for $\mathfrak{M}'$. If there is no preceding stage, the initial entry condition is likewise preserved. The refined stage stays in $A' \subseteq A$ and releases into $I' \subseteq I$. The original compatibility condition $I \subseteq Q$ for the next stage therefore implies $I' \subseteq Q$; if no following stage exists, the final release condition is strengthened. All unaffected stages retain their existing certificates. Composition using Theorem 7.5 is thus sound throughout, and replacing h by $h' \leq h$ cannot increase the total horizon bound. □

7.4 Failure-Introduction Budgets

A strategic macro may satisfy its target yet create transient weaknesses. We can account for these without assuming that weighted vulnerability counts equal the game-theoretic value of a position.

Definition 7.9 (Semantic risk cost and certified budget)**.** *For weights $w_j \geq 0$, fixed player p, and a legal transition $s \xrightarrow{m} t$, define the* failure-introduction cost

$$c_{p,w}(s, m) = \sum_{j=1}^{n} w_j \, \mathbf{1}[f_j(s, p) = 0 \ \wedge \ f_j(t, p) = 1]. \tag{36}$$

The cost of a finite path is the sum of its transition costs. A certified macro has a risk budget $b \geq 0$ *if every one of its legal outcomes, including all opponent replies, accumulates cost at most b before release. The cost also counts failures introduced by opposing moves.*

Proposition 7.10 (Additive worst-case budget under composition)**.** *If compatible certified macros $\mathfrak{M}, \mathfrak{N}$ have risk budgets $b_{\mathfrak{M}}, b_{\mathfrak{N}}$ under the same weights and player perspective, then their phase-indexed composition has the valid budget $b_{\mathfrak{M}} + b_{\mathfrak{N}}$.*

Proof. Every composed outcome splits at its first handoff into one path covered by the first certificate and one by the second. The join repeats the boundary *state* but no move. Transition costs are nonnegative and additive along concatenated paths, so the two separate upper bounds sum to an upper bound for the whole outcome. Since the opponent continuation was arbitrary, this holds for every outcome. □

Remark 7.11 (Meaning of a vulnerability budget)**.** The budget records the weighted number of *newly introduced* conditions along the path. It does not price removed vulnerabilities, prove a favorable outcome in chess, or substitute for adversarial search. The same failure may be introduced more than once and is counted each time, consistently with the gross ledger in Equation (14).

7.5 Noncommutativity and Counter-Repairs

Example 7.12 (Repair order changes the result)**.** Consider a two-bit *abstract semantic test system*, not a chess position. Let $a(x, y) = (0, y)$ clear the first failure, and let $b(x, y) = (x, x)$ synchronize the second predicate with the first. Starting from $(1, 0)$,

$$b\big(a(1, 0)\big) = (0, 0), \tag{37}$$

$$a\big(b(1, 0)\big) = (0, 1). \tag{38}$$

The operation b can therefore act as a counter-repair by reintroducing the second condition if applied before a. Both operators are total and idempotent, but their order changes the final failure signature. In actual chess, legal moves alternate by player and not every ordering is admissible; the concrete transition kernel must check any proposed instantiation. This example complements the *equal-endpoint* order-sensitive safety example in Example 6.8.

Proposition 7.13 (Bounded resistance to counter-repair)**.** *Let $\mathfrak{M}$ be a certified macro from an entry state $s \in P$. No opponent policy can force either an exit from A before release or failure to release into I within h plies while the prescribed σ is followed. Conversely, the existence of one such opponent policy proves that the claimed macro certificate is invalid at s.*

Proof. The first statement is exactly the universal guarantee in Definition 7.1. For the converse, the counterpolicy supplies a legal outcome inconsistent with at least one of the macro requirements. Thus the universal claim fails. This is a certificate-validity criterion, not a new chess-strength result; the bounded counterstrategy construction in Theorem 9.30 provides its game-theoretic background. □

Remark 7.14 (From move chunks to strategic contracts)**.** An opening variation is not automatically a certified macro: an opponent may choose a different legal branch. A reusable opening or defensive construction must therefore specify which opening states belong to its entry set and quantify over the opponent replies it purports to handle. The new macro calculus refines the phase-indexed strategy language of Section 6; it does not replace or amend the legal chess rules. Its potentially novel application lies in the selection of useful *chess-specific* failure predicates, interfaces, and low-cost contracts. Formal compilation and performance experiments remain later stages of the semantic descent.

8. Semantic Equivalence and Transfer of Certified Strategies

8.1 The Transfer Problem

Section 7 treats a complete adversarial plan as a certified macro, with an entry set, safety envelope, target, release interface, and bounded horizon. The present question is distinct from composition: *under which conditions can a macro established at one complete chess state be reused at a different state without losing its guarantee?*

The issue is not whether two diagrams look alike. The same pawn structure, material balance, or finite failure signature can conceal different legal moves, checking resources, castling rights, en-passant rights, or repetition histories. Furthermore, a macro does not prescribe one fixed sequence: its policy may branch in response to opposing moves. Successful transfer must preserve those adversarial alternatives, not merely the intended line.

We use a form of *bisimulation* for state-transition systems [9, 10]. We apply this established mathematical machinery to the strategic contracts of Definition 7.1, deriving concrete preservation conditions for chess-state abstractions and certified strategies.

8.2 Contract-Respecting State Relations

Definition 8.1 (Contract observation)**.** *Fix a player $p \in \{\mathrm{W}, \mathrm{B}\}$ and contract sets $P \subseteq A \subseteq \mathcal{S}$ and $I \subseteq G \subseteq A$. The* contract observation *of a state s is*

$$\omega_{A,G,I}(s) = \big(\pi(s),\ \mathbf{1}_{\mathcal{T}}(s),\ \mathbf{1}_A(s),\ \mathbf{1}_G(s),\ \mathbf{1}_I(s)\big). \tag{39}$$

Here $\pi(s)$ denotes the side to move; terminal states also carry an unambiguous turn convention. The observation records exactly the information necessary to test safety, first target arrival, and release. Additional failure predicates may be attached when the contract also requires particular semantic events or cost bounds.

Definition 8.2 (Contract-respecting transition bisimulation)**.** *Let $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$ be a binary relation. It is a* contract-respecting transition bisimulation *for (A, G, I) if every $(s, t) \in \mathcal{R}$ satisfies:*

(i) $\omega_{A,G,I}(s) = \omega_{A,G,I}(t)$*;*

(ii) if s, t are nonterminal, then for every $m \in \mathcal{L}(s)$ there is some $n \in \mathcal{L}(t)$ with $(\delta(s, m), \delta(t, n)) \in \mathcal{R}$;

(iii) conversely, for every $n \in \mathcal{L}(t)$ there is some $m \in \mathcal{L}(s)$ with $(\delta(s, m), \delta(t, n)) \in \mathcal{R}$.

Move labels *need not be equal. Only the legality of each move and the relationship of its successor matter. Write*

$$\mathcal{R}[X] = \{t \in \mathcal{S} : \exists s \in X\ (s, t) \in \mathcal{R}\} \tag{40}$$

for the forward image of a set of states.

The two successor clauses serve different roles. On the protagonist's turn, a move chosen by the original certified policy must have a legal counterpart in the new position. On the opponent's turn, *every* legal reply in the new position must have a matching reply in the original position. Requiring both clauses for every state is convenient, but stronger than necessary for transferring one fixed policy; relaxing this requirement is a later research problem.

Lemma 8.3 (Finite-path matching)**.** *Suppose $\mathcal{R}$ satisfies Definition 8.2 and $(s_0, t_0) \in \mathcal{R}$. Every finite legal path beginning at s_0 can be matched by a legal path of the same length beginning at t_0, with states related at each step. The converse also holds.*

Proof. Use induction on path length. Length zero is the hypothesis. Given related nonterminal states at step j, the forward successor clause produces a matching successor for the next legal move from the first path, and this pair again belongs to $\mathcal{R}$. Repeat. For the converse, use the reverse successor clause. Because terminality is preserved, a path cannot be spuriously extended after termination. □

8.3 Strategy Transfer with an Explicit Simulation Memory

Theorem 8.4 (Transfer of a certified strategic macro)**.** *Let $\mathfrak{M} = (p, P, A, G, I, h, \sigma)$ be certified as in Definition 7.1, and let $\mathcal{R}$ be a contract-respecting transition bisimulation for (A, G, I). There exists a history-dependent policy $\sigma^{\mathcal{R}}$ that is certified on $\mathcal{R}[P]$ with the* same *safety set A, target G, release interface I, and horizon h. Moreover, every first-release state reachable under the transferred policy is related to a first-release state reachable under the source policy:*

$$\mathsf{H}(\mathfrak{M}^{\mathcal{R}}) \subseteq \mathcal{R}[\mathsf{H}(\mathfrak{M})]. \tag{41}$$

Proof. Choose a target entry state $t_0 \in \mathcal{R}[P]$ and a witness $s_0 \in P$ with $(s_0, t_0) \in \mathcal{R}$. The transferred strategy maintains in memory a *simulated source history*, initially (s_0), paired with the actual target history.

At a protagonist decision, the source policy σ specifies a legal move from the current simulated state. By the forward successor clause, choose a legal target move leading to a related pair of successors, and extend both histories. At an opponent decision, observe the *actual* target move. The reverse successor clause supplies a legal source reply leading to a related pair; extend the simulated history with that reply. Such choices exist at every step and can be made from finite legal-move sets.

Inductively, every actual play consistent with the lifted policy has a paired source play consistent with σ; the two states remain related at every ply. The original certificate guarantees that the source play remains in A and first reaches G in I within h plies. Equality of contract observations ensures the same safety, first-arrival, and interface properties along the target play. The first-release states form a related pair, proving Equation (41). Since the actual opponent choices were arbitrary, the lifted policy is certified for every target entry state. □

Remark 8.5 (Transfer of contingent strategies)**.** The lifted policy may use different chess moves from the original policy, and it can retain the simulated source history as internal memory. Transfer therefore concerns a *strategic guarantee*, not a literal opening variation or a fixed sequence of square coordinates. The theorem establishes no preservation of an engine's numerical score or the game-theoretically best move.

Corollary 8.6 (Compositional transfer at certified handoffs)**.** *Suppose two macros $\mathfrak{M}; \mathfrak{N}$ are compatible as in Theorem 7.5, share a common safety set, and are both transported by the same contract-respecting bisimulation that preserves their respective target and interface memberships. Then the transferred phases form a valid phase-indexed program with the original combined horizon bound $h + k$.*

Proof. Apply Theorem 8.4 to each phase. The transferred first handoff set is included in $\mathcal{R}[\mathsf{H}(\mathfrak{M})]$. Since $\mathsf{H}(\mathfrak{M}) \subseteq Q$, where Q is the second macro's entry set, this image lies in $\mathcal{R}[Q]$, which is its transferred entry set. The compatible composition theorem Theorem 7.5 now applies, with its original horizon bound. □

8.4 Finite-Horizon Equivalence

A globally stable quotient of the complete chess state graph is a strong requirement. For a strategy that terminates within h plies, only a correspondingly bounded portion of the transition relation is relevant.

Definition 8.7 (Depth-indexed contract equivalence)**.** *Define relations $\mathcal{R}_j \subseteq \mathcal{S} \times \mathcal{S}$ recursively for $j \geq 0$. First,*

$$(s,t) \in \mathcal{R}_0 \iff \omega_{A,G,I}(s) = \omega_{A,G,I}(t). \tag{42}$$

For $j \geq 0$, put $(s,t) \in \mathcal{R}_{j+1}$ precisely when $(s,t) \in \mathcal{R}_0$ and either both are terminal or both are nonterminal and the following conditions hold:

$$\forall m \in \mathcal{L}(s)\ \exists n \in \mathcal{L}(t) : (\delta(s,m), \delta(t,n)) \in \mathcal{R}_j, \tag{43}$$

$$\forall n \in \mathcal{L}(t)\ \exists m \in \mathcal{L}(s) : (\delta(s,m), \delta(t,n)) \in \mathcal{R}_j. \tag{44}$$

Lemma 8.8 (Monotonicity of depth approximants)**.** *For every $j \geq 0$, $\mathcal{R}_{j+1} \subseteq \mathcal{R}_j$.*

Proof. For $j = 0$, membership in $\mathcal{R}_1$ explicitly requires membership in $\mathcal{R}_0$. Assume the claim for $j - 1$. Every successor pair required by $\mathcal{R}_{j+1}$ lies in $\mathcal{R}_j$ and hence in $\mathcal{R}_{j-1}$, so the defining conditions for $\mathcal{R}_j$ also hold. Terminal pairs remain terminal at both depths. Induction completes the proof. □

Theorem 8.9 (Horizon-bounded strategy transfer)**.** *Let $\mathfrak{M} = (p, P, A, G, I, h, \sigma)$ be certified. For a second state t_0, suppose there is a source entry $s_0 \in P$ such that $(s_0, t_0) \in \mathcal{R}_h$. Then there exists a policy for t_0 that maintains A, first reaches G in I, and releases within h legal plies against every opponent reply.*

Proof. Couple the source and target plays as in Theorem 8.4, but maintain the stronger invariant that states are related by $\mathcal{R}_{h-j}$ after j plies. At the protagonist's turn use Equation (43); at the opponent's turn use Equation (44). These clauses ensure a matching move until either the source strategy releases or the horizon is exhausted. At every reached depth, $\mathcal{R}_{h-j} \subseteq \mathcal{R}_0$ by Lemma 8.8, so the safety and target/release observations agree. Since the source always releases within h plies, the target releases at the same first target visit, within the same bound. □

Corollary 8.10 (Recovery of the stable-abstraction criterion)**.** *Let $\alpha : \mathcal{S} \to Q$ satisfy the turn- and terminality-preserving, transition-stability criterion in Definition 6.12. If A, G, and I are α-measurable, then any certified macro transfers to every state with the same α-value as one of its entry states.*

Proof. The relation $\mathcal{R}_\alpha = \{(s,t) : \alpha(s) = \alpha(t)\}$ preserves the contract observation by hypothesis. The abstract successor-set equality of Equation (31) provides the two successor-matching clauses of Definition 8.2. Apply Theorem 8.4. □

8.5 Strategic Requirements Beyond Shared Signatures

Example 8.11 (Same strategy contract, different concrete moves)**.** Consider two small abstract game trees, not chess positions. In the first, our player chooses action a from s to an opponent node x; in the second, the player chooses a differently named action b from t to an opponent node y. Each opponent node has two legal replies, leading to two terminal states in the same release interface $I = G$. Relate s to t, x to y, and the corresponding pairs of terminal goal states. With A equal to the entire arena, this relation satisfies Definition 8.2. The certified two-ply macro `play a, then accept either reply` therefore transfers to `play b, then accept either reply`. The transfer works not because the action labels agree (they do not), but because both opponent alternatives preserve the same contract.

Example 8.12 (Same signature, different strategic outcome)**.** Consider a finite *abstract turn-based arena*, not a chess position, with two nonterminal p-states s, t, each having only one legal successor. From s, the successor u belongs to a terminal goal set $G = I$; from t, its successor v is terminal but lies outside G. Let A be the entire arena, and let an abstraction assign the *same* signature to s and t while distinguishing u from v by goal membership. The one-ply macro starting at s is certified, but the corresponding objective is impossible from t. Thus signature agreement without a successor-matching condition does not justify strategy transfer. The two root states lie in $\mathcal{R}_0$ but not $\mathcal{R}_1$.

Proposition 8.13 (Preservation of a failure-introduction budget)**.** *Under Theorem 8.4, additionally suppose every related pair $(s, t) \in \mathcal{R}$ agrees on the values of all failure predicates $f_j(\cdot, p)$ appearing in the cost Equation (36), with the same nonnegative weights. Then a risk budget b certified for $\mathfrak{M}$ is also valid for its transferred policy $\mathfrak{M}^{\mathcal{R}}$.*

Proof. For every pair of corresponding transitions $s \to s'$ and $t \to t'$ in the coupled traces, both endpoint pairs agree on each relevant failure bit. Therefore the indicator of a $0 \to 1$ failure introduction is identical for each pair of transitions, and so are their weighted costs. Summing over the same number of plies gives equal total introduction cost. The source budget upper-bounds every source outcome, hence also every target outcome paired to it. □

Remark 8.14 (Checking the transfer conditions)**.** The theorems give explicit sufficient conditions for transferring *specified strategy contracts*. For concrete chess positions, these conditions are verified by checking a proposed transfer relation or constructing a matching certificate from the full legal-move graph. Discovering economical chess-specific abstractions with these properties is a central constructive problem for the semantic compiler.

9. Foundational Results

9.1 Lemmas

Lemma 9.1 (Decidability of finite failure classification)**.** *If $\mathcal{F}$ is finite and each predicate f_i is computable and terminating, then $C(s, p)$ is computable for every valid state s and player p.*

Proof. Evaluate each $f_i(s, p)$. There are finitely many predicates, and each evaluation terminates by assumption. Therefore the classifier $C(s, p)$ can be obtained after finitely many steps. □

Lemma 9.2 (Determinacy of local repair)**.** *For every legal state–move pair (s, m) and decidable failure predicate f, the truth value of $R_f(s, m)$ is uniquely determined.*

Proof. Legal state transition is deterministic, so $s' = \delta(s, m)$ is unique. The values $f(s, p)$ and $f(s', p)$ are determined by the predicate. Their conjunction in Equation (9) is therefore uniquely determined. □

Lemma 9.3 (Local repair is insufficient to rank moves)**.** *Without additional assumptions linking a local failure predicate to the move-evaluation function, the truth of $R_f(s, m)$ does not entail that m is preferable to every other legal move.*

Proof. The definition of R_f constrains only the two values of f before and after m. It imposes no ordering condition on V. Therefore an evaluation may assign a higher value to an alternative move m', including one

that leaves f unchanged. A ranking implication requires additional axioms or proved bounds connecting the failure predicate to V. □

9.2 Theorems

Theorem 9.4 (Legal-move soundness of semantic selection)**.** *If the selection procedure returns only an element of $\mathcal{L}(s)$, then every returned move is legal in state s.*

Proof. By its stated selection contract, $\mathrm{Select}(s) \in \mathcal{L}(s)$. By Definition 2.2, every element of $\mathcal{L}(s)$ is a legal chess move. □

Theorem 9.5 (Decidability of bounded adversarial repair)**.** *Assume legal move generation and the relevant failure predicates are computable and terminating. Every repair property specified through a fixed finite depth of legal adversarial continuations, with computable terminal conditions, is decidable.*

Proof. Every chess state has finitely many legal successors. A tree of continuations truncated at fixed finite depth has finitely many nodes. At every node, transition checks and relevant predicates terminate by hypothesis. The required existential and universal conditions can therefore be evaluated by finite recursion over the tree. □

Theorem 9.6 (Conditional preservation of bounded minimax value)**.** *Fix a finite search horizon and an identical terminal or horizon evaluation for full and filtered searches. Suppose the filter retains, at every maximizing node, at least one successor achieving the full-search maximum, and at every minimizing node, at least one successor achieving the full-search minimum, using the backed-up values of the corresponding full subtree. Then the filtered search has the same root minimax value as the full search.*

Proof. Proceed by induction on remaining search depth. At depth zero the two searches share the same evaluation. Assume the equality holds for retained successors at all nodes of depth at most $d - 1$. At a maximizing node of depth d, the filter retains a successor attaining the original maximum; by the induction hypothesis its value is unchanged. Removing other successors cannot raise the maximum, and retaining an optimal successor prevents it from decreasing. The minimizing case is dual. Hence equality holds at depth d, including the root. □

Remark 9.7 (Scope of the filtering theorem)**.** Theorem 9.6 is a conditional correctness result, not a new efficient pruning algorithm. The central open problem is to identify inexpensive semantic conditions that preserve the required optimal successors without first computing the full tree.

9.3 Compositional Results and Certificate Soundness

Theorem 9.8 (Additivity of the gross repair ledger)**.** *For any composable finite legal paths γ and η and fixed player p,*

$$\Lambda_p(\gamma \cdot \eta) = \Lambda_p(\gamma) + \Lambda_p(\eta). \tag{45}$$

Furthermore, the net semantic effect satisfies

$$\Phi(s_0, p) - \Phi(s_k, p) = G^-(\gamma, p) - G^+(\gamma, p). \tag{46}$$

Proof. The transitions of a concatenation are precisely the transitions of the first path followed by those of the second. Both count vectors are sums over transitions, so each vector is additive under concatenation. For a

binary coordinate f_j, each transition satisfies

$$f_j(s_{i-1}, p) - f_j(s_i, p) = \mathbf{1}[1 \to 0] - \mathbf{1}[0 \to 1].$$

Summing over $i = 1, \ldots, k$ telescopes on the left and equals the difference of the two gross counts on the right. Applying this to each coordinate proves the result. □

Corollary 9.9 (Net-zero paths can still contain failure events)**.** *If $\Phi(s_0, p) = \Phi(s_k, p)$, then $G^-(\gamma, p) = G^+(\gamma, p)$, but neither vector need be zero.*

Proof. Apply Equation (46). Equality of the two nonnegative count vectors follows from a zero difference; their common value can be positive. □

Example 9.10 (A transient central-pawn vulnerability)**.** Continue the opening of Example 3.4 after 1.*e*4 *e*5 2.Nf3 Nc6 with 3.Ng1 Nb8 4.Nf3 Nc6. The black central pawn is defended at the beginning and end of this continuation. Immediately after White's fourth move, however, the pawn on *e*5 is attacked by the knight on *f*3 and undefended because Black's knight has returned to *b*8. The final . . . Nc6 repairs it again. Thus the net change in $f_{\text{center}}(\cdot, B)$ is zero while both the introduction and repair counts for that predicate equal one.

Definition 9.11 (Local repair certificate)**.** *For a claimed local repair of predicate f, fix $p = \pi(s)$. A certificate consists of (s, m, p, f, b_0, b_1), where s identifies a complete state, m is a candidate move, and $b_0, b_1 \in \{0, 1\}$ are the claimed before/after values. A verifier independently checks that $m \in \mathcal{L}(s)$, sets $s' = \delta(s, m)$, recomputes $f(s, p)$ and $f(s', p)$, and accepts the repair claim exactly when $b_0 = 1$, $b_1 = 0$, and both recomputed values match.*

Theorem 9.12 (Soundness of independently checked local repair certificates)**.** *Assume the legal-transition kernel and the predicate evaluator are sound with respect to their mathematical specifications. Every accepted local repair certificate establishes $R_f(s, m)$ for the stated player.*

Proof. Acceptance implies that m is legal, that the successor $s' = \delta(s, m)$ has been computed, and that independent evaluation has verified $f(s, p) = 1$ and $f(s', p) = 0$. These are exactly the conditions in Definition 4.1, with $p = \pi(s)$. Therefore $R_f(s, m)$ holds. □

Remark 9.13 (From algebraic structure to chess computation)**.** Theorems 9.8 and 9.12 establish compositional accounting and conditional factual soundness. Together with guarded repairs and the strategy calculus, they support the paper's central construction: moving from checkable facts about individual positions to auditable claims about adversarial continuations. The chess-specific predicates, strategic interaction laws, and evaluation protocol connect this structure to the proposed executable system.

9.4 Interaction and Counterstrategy Results

Lemma 9.14 (Semantic dominance is coordinatewise dominance)**.** *Let $m, n \in \mathcal{L}(s)$ share origin s and player $p = \pi(s)$. Then m weakly semantically dominates n in Equation (18) if and only if*

$$\Phi(\delta(s, m), p) \le \Phi(\delta(s, n), p) \quad \textit{componentwise}. \tag{47}$$

Proof. Fix a coordinate with common origin bit x_i. If $x_i = 1$, $i \in D^-$ exactly when the destination bit is zero; there is no introduction at that coordinate. Therefore the removal inclusion is equivalent to $(y_m)_i \le (y_n)_i$. If $x_i = 0$, $i \in D^+$ exactly when the destination bit is one; there is no removal. Hence the introduction inclusion is equivalent to the same inequality. The coordinates exhaust the ontology. □

Lemma 9.15 (Exact interaction-gain decomposition)**.** *For $x, y \in \{0,1\}^n$ let $d = x - y$ over the integers. Then*

$$H_{w,J}(x) - H_{w,J}(y) = \sum_i w_i d_i + \sum_{i<j} J_{ij}(x_i d_j + x_j d_i - d_i d_j). \tag{48}$$

Proof. Since $y_i = x_i - d_i$, $x_i x_j - y_i y_j = x_i d_j + x_j d_i - d_i d_j$. Substitute into Equation (19) and collect terms. □

Theorem 9.16 (Exact criterion for monotone quadratic failure energy)**.** *The energy $H_{w,J}$ of Definition 4.7 is nondecreasing with respect to the componentwise order on $\{0,1\}^n$ if and only if, for every i,*

$$w_i + \sum_{j \neq i} \min(0, J_{\min(i,j),\max(i,j)}) \geq 0. \tag{49}$$

Under these conditions, if m weakly semantically dominates n, then $H_{w,J}(y_m) \leq H_{w,J}(y_n)$.

Proof. For fixed other coordinates, the increment from changing x_i from zero to one is

$$w_i + \sum_{j \neq i} J_{\min(i,j),\max(i,j)} x_j.$$

The minimum of this increment over all Boolean choices of the other coordinates is exactly the left-hand side of Equation (49): select $x_j = 1$ when its coupling is negative and $x_j = 0$ when positive. Hence the criterion is equivalent to a nonnegative energy increment for every single-coordinate transition $0 \to 1$. Any coordinatewise increase can be decomposed into such transitions, proving the equivalence. The final conclusion follows from Lemma 9.14. □

Remark 9.17 (Necessity of the monotonicity criterion)**.** If the coefficients violate Equation (49), a semantically weaker signature may receive a lower numerical energy. For example, $w_1 = w_2 = 0$, $J_{12} = -1$ gives $H(0,1) = 0 > H(1,1) = -1$. Thus negative interactions cannot be treated as harmless without checking monotonicity. Even when monotonicity holds, the ranking is about the selected semantic predicates, not chess strength.

Proposition 9.18 (Counterresponse witness criterion)**.** *For a nonterminal successor s' of a guarded local repair $R_{f;g}(s,m)$, the guard and repaired predicate persist against* every *immediate legal opponent reply if and only if $\mathsf{K}_{f;g}(s,m) = \varnothing$.*

Proof. By Definition 4.8, the set contains exactly the replies for which at least one of guard preservation and continued absence of the failure is violated. Therefore its emptiness is equivalent to the universal conjunction of those requirements. □

Corollary 9.19 (Preservation under monotone failure costs)**.** *When Equation (49) holds, a candidate that weakly semantically dominates another cannot have a higher quadratic failure energy. The same conclusion holds for any coordinatewise nondecreasing scalar objective on the Boolean ontology.*

Proof. Combine Lemma 9.14 and Theorem 9.16; the general case is the definition of coordinatewise monotonicity. □

Corollary 9.20 (Telescoping of interaction gain over legal paths)**.** *Let $\gamma = (s_0, m_1, s_1, \ldots, m_k, s_k)$ be a legal path and fix player p. Then*

$$\sum_{t=1}^{k} \big[H(\Phi(s_{t-1}, p)) - H(\Phi(s_t, p))\big] = H(\Phi(s_0, p)) - H(\Phi(s_k, p)).$$

The net energy change does not reveal intermediate failures; for that purpose the gross ledger of Equation (14) must be retained.

Proof. Adjacent terms cancel by telescoping. A path with a $0 \to 1 \to 0$ predicate change demonstrates why the net difference alone is insufficient. □

9.5 Strategy-Synthesis and Composition Results

Theorem 9.21 (Exactness of bounded adversarial repair synthesis)**.** *For every nonnegative horizon h, $s \in \mathsf{W}_p^h(A, B)$ if and only if some p-strategy guarantees $\mathsf{Goal}_h(A, B)$ from s. Furthermore, a witnessing strategy can be extracted constructively from Equations (23) and (24).*

Proof. Induct on h. At $h = 0$, the only admissible successful outcome is already in $A \cap B$, exactly $\mathsf{W}_p^0(A, B)$. For the step, a state outside A is unsuccessful, and a state already in $B \cap A$ succeeds immediately. At any remaining nonterminal player-p state, a successful strategy must select at least one legal successor with an $(h-1)$-ply guarantee. At an opponent state, every legal successor must possess that guarantee. The induction hypothesis converts these requirements precisely to membership in $\mathsf{CPre}_p(\mathsf{W}_p^{h-1}(A, B))$. The induction also selects one witness move at our nodes and recursively builds all necessary responses at opponent nodes. Terminal states outside B cannot succeed. □

Theorem 9.22 (Sound guarded sequential synthesis)**.** *Fix $h, k \geq 0$ and sets $B, C \subseteq A$. Suppose*

$$s \in \mathsf{W}_p^h(A, B) \quad \text{and} \quad A \cap B \subseteq \mathsf{W}_p^k(A \cap B, B \cap C). \tag{50}$$

Then

$$s \in \mathsf{W}_p^{h+k}(A, B \cap C). \tag{51}$$

Proof. By Theorem 9.21, a first strategy from s maintains A and reaches B by at most h plies, regardless of opponent actions. At the first such state t, we have $t \in A \cap B$. By the second premise, a strategy from t maintains $A \cap B$ and reaches $B \cap C$ within k additional plies. The combined history-dependent policy executes the first strategy until this handoff and then executes the second. Every complete outcome stays in A and reaches $B \cap C$ within $h + k$ plies; applying Theorem 9.21 proves the conclusion. □

Remark 9.23 (Why the handoff assumption matters)**.** Two locally successful repairs need not compose. The second may undo the first, may be illegal after the opponent's reply, or may require a state not actually produced by the first. The uniform inclusion in Equation (50) is sufficient but stronger than necessary; an implementation can check only the *reachable* handoff states in a particular certified first-stage strategy.

Proposition 9.24 (Controllable predecessor is monotone, not generally union- or intersection-preserving)**.** *For any $X \subseteq Y$, $\mathsf{CPre}_p(X) \subseteq \mathsf{CPre}_p(Y)$. However, there are finite alternating arenas in which*

$$\mathsf{CPre}_p(X \cup Y) \neq \mathsf{CPre}_p(X) \cup \mathsf{CPre}_p(Y),$$
$$\mathsf{CPre}_p(X \cap Y) \neq \mathsf{CPre}_p(X) \cap \mathsf{CPre}_p(Y).$$

Proof. Monotonicity holds because inclusion preserves both existential and universal successor conditions. For the union counterexample, take an opponent-to-move state with precisely two successors $x \in X \setminus Y$ and $y \in Y \setminus X$. All successors lie in $X \cup Y$, but not all lie in either X or Y. For the intersection counterexample, take a player-p state with exactly the same two successors. There is a successor in X and another in Y, but none in $X \cap Y$. □

Theorem 9.25 (Bounded strategy preservation under a stable semantic abstraction)**.** *Suppose α is turn-respecting and transition-stable as in Definition 6.12, and that A and B are α-measurable. Then $\mathsf{W}_p^h(A, B)$ is α-measurable for every finite h.*

Proof. At horizon zero the reachability region is $A \cap B$, which is constant on each abstraction class. Assume the claim at horizon h. Two states in the same abstraction class have identical player-to-move and terminality and identical sets of abstract successor classes. Since membership in $\mathsf{W}_p^h(A, B)$ is constant within each abstract successor class, the existential condition at player nodes and universal condition at opponent nodes give the same truth value for the two states. Intersecting with the abstraction-measurable A, and including the measurable B, preserves this equality. Hence the claim holds at $h + 1$. □

Corollary 9.26 (Construction of a bounded strategy certificate)**.** *Under a sound legal-transition kernel and decidable membership in A, B, every successful bounded strategy claim from s has a finite, checkable AND/OR certificate of the form described in Section 6. Conversely, the existence of an accepted certificate implies $s \in \mathsf{W}_p^h(A, B)$.*

Proof. The constructive induction in Theorem 9.21 supplies a chosen legal move at each player node and subcertificates for every legal opponent reply. Independently checking legality, exhaustive opponent branching, and the base conditions reproduces that induction, and hence acceptance implies membership. Finiteness follows from the finite horizon and finite branching of legal moves. □

Remark 9.27 (Mathematical foundations and research direction)**.** The algebra in this section combines alternating reachability with guarded policy composition to obtain explicit certificates and transfer obligations for the chess-semantic compiler. A central research challenge is to identify informative abstractions that accelerate search while preserving the strategic distinctions required by Theorem 9.25 (or by a suitable weaker refinement criterion).

9.6 Order, Duality, and Counterstrategy Results

Proposition 9.28 (The repair ledger cannot decide guarded safety)**.** *Even for total, commuting, idempotent operators on a finite Boolean signature space, there exist two differently ordered repair protocols with identical initial and final signatures and identical gross ledgers, but different satisfaction of an intermediate-state safety condition.*

Proof. Use the operators and safety envelope of Example 6.8. The two traces have the same endpoints. In either order, each predicate changes from 1 to 0 exactly once, so their gross ledger is $((1, 1), (0, 0))$, where the first vector counts removals. The first trace visits $(0, 1)$ and violates the envelope, whereas the second does not. Therefore neither the endpoints nor the gross ledger determine safety of the ordered execution. □

Theorem 9.29 (Duality of one-step controllable predecessors)**.** *For any subset $X \subseteq \mathcal{S}$, if every nonterminal state has a nonempty legal move set, then*

$$\mathsf{CPre}_p(X) = N \setminus \mathsf{CPre}_{\bar{p}}(\mathcal{S} \setminus X). \tag{52}$$

Proof. At a nonterminal player-p state, membership on the left means that *some* legal successor belongs to X. Membership in the opposite-player predecessor on the right means that *all* legal successors belong to $\mathcal{S} \setminus X$; negating this condition gives the same existential requirement. At an opponent-to-move state, the left condition is universal membership in X, while negating the opponent's existential membership in $\mathcal{S} \setminus X$ gives the same universal condition. Nonempty legal move sets make the usual quantifier duality unambiguous. Restriction to N handles terminal origins, which neither predecessor includes. □

Theorem 9.30 (Bounded strategy or bounded refutation)**.** *Fix decidable $B \subseteq A \subseteq \mathcal{S}$ and finite $h \geq 0$. From any state s, precisely one of the following obtains:*

1. *Player p has a strategy satisfying $\mathsf{Goal}_h(A, B)$.*
2. *Player $\bar{p}$ has a bounded counterstrategy in the sense of Definition 6.10.*

Each side admits a finite AND/OR certificate for its respective claim.

Proof. By Theorem 9.21, the first condition is equivalent to $s \in \mathsf{W}_p^h(A, B)$. We construct a counterstrategy for states outside this region by induction on h. At $h = 0$, failure means $s \notin B$ or $s \notin A$, so the empty policy refutes the objective. For $h + 1$, an unsafe state, terminal state outside B, or safe non-target state with no legal move is already refuted. Otherwise $s \notin \mathsf{W}_p^{h+1}(A, B)$ means $s \notin \mathsf{CPre}_p(\mathsf{W}_p^h(A, B))$. At a player-$p$ node, *every* legal move leads outside $\mathsf{W}_p^h(A, B)$, so the inductively constructed refutation covers all their choices. At an opponent node, there exists at least one legal move leading outside this region; choose one and continue by induction. Consequently the opponent can force failure within the stated horizon. Conversely, both strategies cannot coexist, because one guarantees success against *all* opponent responses while the other guarantees failure against *all* player-p responses. The inductive construction yields finite certificates: a successful repair certificate chooses one player-p move and branches over every opponent move; a refutation certificate has the dual shape. □

Theorem 9.31 (Associativity of compatible three-stage repairs)**.** *Let $B, C, D \subseteq A$ and $h, k, \ell \geq 0$. Suppose*

$$s \in \mathsf{W}_p^h(A, B), \tag{53}$$

$$A \cap B \subseteq \mathsf{W}_p^k(A \cap B, B \cap C), \tag{54}$$

$$A \cap B \cap C \subseteq \mathsf{W}_p^\ell(A \cap B \cap C, B \cap C \cap D). \tag{55}$$

Then

$$s \in \mathsf{W}_p^{h+k+\ell}(A, B \cap C \cap D). \tag{56}$$

For fixed component policies, the phase-indexed executions of $(\sigma_1; \sigma_2); \sigma_3$ and $\sigma_1; (\sigma_2; \sigma_3)$ coincide as ordered, phase-tagged legal traces under the first-handoff convention of Definition 6.11.

Proof. Apply Theorem 9.22 first to the initial two stages, yielding $s \in \mathsf{W}_p^{h+k}(A, B \cap C)$. Apply it again using the third hypothesis to obtain Equation (56). The constructive proof extracts a policy for each phase and switches only when the current phase first attains its target. Whether the policy list is parsed as $(\sigma_1; \sigma_2); \sigma_3$ or as $\sigma_1; (\sigma_2; \sigma_3)$, the same ordered three policies are run, and the same phase counter advances at the same first-handoff states. Hence the produced sets of phase-tagged legal traces are identical. This is associativity for *compatible, fixed* phase-indexed policy execution, not commutativity of repair operations. □

Corollary 9.32 (Checkable refutation evidence)**.** *Under an independently checked legal transition kernel, finite branching, and decidable membership in A, B, every failed bounded strategy claim admits a finite*

refutation certificate. A checker must enumerate every player-p move and verify at least one legal adversarial reply at opponent nodes.

Proof. Apply the construction in Theorem 9.30 and check each inductive node against independently generated legal moves. Finiteness follows from the finite horizon. □

Remark 9.33 (Scope and originality)**.** Theorem 9.30 is the standard bounded determinacy principle for finite-horizon perfect-information games, specialized to our guarded repair objective. Theorem 9.31 formalizes an associative guarded sequencing discipline. Neither theorem alone implies a new optimal chess search algorithm. The research challenge is to synthesize useful *chess-specific* guards and repair macros that permit computationally effective certificates and pruning.

9.7 Corollaries

Corollary 9.34 (Executability of finite-horizon analysis)**.** *Under the hypotheses of Lemma 9.1 and Theorem 9.5, an engine that composes legal transitions, finite failure classification, and fixed-depth repair evaluation can be implemented as a terminating procedure.*

Proof. Each constituent computation terminates, and fixed-depth composition uses finitely many such computations. □

Corollary 9.35 (Conditional correctness of search reduction)**.** *If a semantic filter satisfies the optimal-successor preservation conditions of Theorem 9.6 throughout the bounded search tree, then it does not alter that tree's root minimax value.*

Proof. Immediate from Theorem 9.6. □

10. Selective Transfer of Certified Strategies

10.1 Why Full Equivalence May Be Unnecessary

The contract-respecting bisimulation of Section 8 matches every legal successor of each source position with a successor of the target position, and conversely. This is sufficient for transferring *any* certified policy satisfying the contract, but it may be unnecessarily strong when transferring *one particular* policy. The policy chooses only one move at its own decision nodes, while the receiving position may offer fewer opponent continuations than the source position. We therefore seek an asymmetric, *strategy-relative* matching condition. This is a specialization of established alternating-simulation and game-refinement ideas [11, 10], not a claim that those methods originated here.

The new construction must respect history-aware chess states: equal failure signatures do not ensure the same legal moves, repetition rights, or adversarial outcomes. We use paired legal histories rather than squarewise piece correspondences. A strategy may transfer even when the concrete moves, board layouts, and unused alternatives differ.

10.2 Directional Contract Observations

Fix a protagonist p and two contracts (P_s, A_s, G_s, I_s, h) and (P_t, A_t, G_t, I_t, h), with $P_x \subseteq A_x$ and $I_x \subseteq G_x \subseteq A_x$ for $x \in \{s, t\}$. The subscripts identify the source and receiving *contracts*, not different chess rules. Both operate on the same history-aware state space $\mathcal{S}$.

Definition 10.1 (Contract-preserving history pair)**.** *Let $u = (s_0, \dots, s_j)$ and $v = (t_0, \dots, t_j)$ be legal histories of identical length. They are* directionally compatible *when their final states s_j, t_j satisfy:*

(i) $\pi(s_j) = \pi(t_j)$*, and both are terminal or both nonterminal, whenever neither state has reached its target;*

(ii) $s_j \in A_s \implies t_j \in A_t$*;*

(iii) $s_j \in G_s \iff t_j \in G_t$*; and*

(iv) $s_j \in I_s \implies t_j \in I_t$*.*

The equivalence in (iii) synchronizes the first *target visit: it rules out an early, potentially invalid release on the receiving side. Condition (iv) then preserves the release interface. Conditions (ii) and (iv) need not hold in the reverse direction.*

Definition 10.2 (Strategy-directed matching witness)**.** *Let $\mathfrak{M}_s = (p, P_s, A_s, G_s, I_s, h, \sigma)$ be a certified source macro. A* selective transfer witness *from $s_0 \in P_s$ to $t_0 \in P_t$ is a set $\mathcal{K}$ of pairs of equally long legal histories with $((s_0), (t_0)) \in \mathcal{K}$, such that:*

(i) Every pair in $\mathcal{K}$ is directionally compatible.

(ii) At a pre-release p-decision history pair (u, v) with fewer than h plies, let $m = \sigma(u)$ be the one *source move selected by the policy. There exists a legal receiving move n with $(u \cdot m, v \cdot n) \in \mathcal{K}$.*

(iii) At a pre-release opponent-decision history pair (u, v) with fewer than h plies, every *legal receiving reply n has at least one legal source reply m such that $(u \cdot m, v \cdot n) \in \mathcal{K}$.*

Here $u \cdot m$ denotes the unique history extension by the legal successor of m; similarly for $v \cdot n$. The relation need not include source histories inconsistent with σ or receiving histories inconsistent with the lifted policy. A source-to-target matching can retain the simulated source history as private memory.

The alternation of quantifiers is deliberate: at a protagonist node, *one* receiving move is sufficient; at an opponent node, *all* receiving replies must be covered. The converse conditions, required by full bisimulation, are not needed for this particular transfer.

10.3 A Strategy-Relative Transfer Theorem

Theorem 10.3 (Soundness of selective strategy transfer)**.** *Let $\mathfrak{M}_s$ be a certified source macro with horizon h. Suppose for every $t_0 \in P_t$ there exists $s_0 \in P_s$ and a selective transfer witness $\mathcal{K}_{s_0,t_0}$ as in Definition 10.2. Then there is a history-dependent receiving policy $\widehat{\sigma}$ such that*

$$\mathfrak{M}_t = (p, P_t, A_t, G_t, I_t, h, \widehat{\sigma})$$

is a certified macro. In particular, every legal receiving opponent continuation preserves A_t until the first visit to G_t, reaches I_t at that visit, and does so within h plies.

Proof. For each receiving entry state t_0, choose a source witness state s_0 and its matching relation. The receiving policy stores a paired source history, initially (s_0), alongside the actual receiving history. At a protagonist node, consult σ on the stored source history and use Definition 10.2(ii) to choose a matching legal receiving move. At an opponent node, observe the *actual* legal receiving reply and use clause (iii) to update the simulated source history with a corresponding legal source reply.

Induction on the number of plies shows that every receiving play under the lifted policy is paired at each step with a legal source play under σ. Because $\mathfrak{M}_s$ is certified, its paired source play stays in A_s and first enters

G_s at a state in I_s in at most h plies. Directional compatibility preserves receiving safety, makes the first target visit occur at the same ply, and maps source interface membership to receiving interface membership. The policy therefore achieves the receiving contract against every legal receiving reply. □

Corollary 10.4 (Policy-specific transfer without full bisimulation)**.** *Full contract-respecting bisimulation of Section 8 is sufficient for selective transfer of any certified policy, after restricting it to the paired histories that policy uses. The converse is false: some policies transfer selectively between states for which no full contract-respecting bisimulation can relate the two roots.*

Proof. Under a bisimulation, every source chosen move has a matching receiving move and every receiving reply has a matching source reply; equality of contract observations implies directional compatibility. Restriction to policy-consistent histories gives a witness.

For strictness, consider finite *abstract* one-ply arenas with protagonist root states s, t, a common safety set containing all states, and a goal/interface consisting of terminal good states. At s there are moves $a : s \to g_s$ and $b : s \to \ell_s$, where g_s is a goal and ℓ_s is not. At t the only move is $c : t \to g_t$, with g_t a goal. The policy selecting a is certified and selectively transfers by matching a to c. A full contract-respecting bisimulation cannot match the source move b to any receiving successor: the unique receiving successor is a goal, while ℓ_s is not. Hence selective transfer is strictly weaker. This counterexample concerns abstract arenas, not a claimed chess position. □

10.4 Transfer Obligations as a Finite Proof Search

A proposed witness can itself be constructed through bounded alternating proof obligations, without enumerating every source move at the protagonist's decision nodes.

Definition 10.5 (Bounded selective-matching test)**.** *Fix a certified source policy σ and two contracts as above. For paired legal histories (u, v) of the same length, define $\mathsf{Match}_j(u, v)$ by induction on $j \geq 0$. The test is false unless the pair satisfies Definition 10.1. If the source history has already reached G_s, the test is true exactly when the pair is compatible (hence the receiving position has reached G_t and lies in I_t along certified source plays). At a pre-release pair, $\mathsf{Match}_0(u, v)$ is false. For $j > 0$ and pre-release pairs:*

$$\mathsf{Match}_j(u, v) = \begin{cases} \bigvee_{n \in \mathcal{L}(t)} \mathsf{Match}_{j-1}(u \cdot \sigma(u), v \cdot n), & \pi(s) = p, \\ \bigwedge_{n \in \mathcal{L}(t)} \bigvee_{m \in \mathcal{L}(s)} \mathsf{Match}_{j-1}(u \cdot m, v \cdot n), & \pi(s) \neq p. \end{cases}$$

Here s, t denote the last states of u, v. The first disjunction ranges over receiving moves only; the selected source move is $\sigma(u)$. The opponent conjunct ranges over every *receiving reply. Preconditions require both states to be nonterminal before release; otherwise the test is false. Finite legal-move sets make the expressions well defined.*

Theorem 10.6 (Soundness of bounded selective matching)**.** *Suppose $\mathfrak{M}_s$ is certified with horizon h and that, for each $t_0 \in P_t$, some source $s_0 \in P_s$ satisfies $\mathsf{Match}_h((s_0), (t_0))$. Then the receiving contract admits a certified policy with horizon at most h.*

Proof. Unfold the finite recursion. Every protagonist disjunction supplies a specific legal receiving choice matching the source policy's selected move. Every opponent conjunction supplies a legal source continuation for every receiving reply. Together these choices form a finite history-pair matching witness through depth h. The base case rules out an unfinished path at the horizon, while paired release states satisfy the required

interface property by directional compatibility and source certification. Applying Theorem 10.3 proves the claim. □

Remark 10.7 (Meaning and complexity of the test)**.** The bounded test is a finite, potentially expensive proof search. It proves transfer only when its obligations succeed and does not establish that a receiving position lacking such a witness has no other successful strategy. Matching is an existential abstraction of one specific source policy, not a general decision procedure for whether a chess position is objectively winning or drawable.

10.5 Contract-Sufficient Semantic Footprints and Risk Transfer

The contract boundary conditions can be distinguished from the failure predicates used for explanatory bookkeeping. Let $D \subseteq \mathcal{F}$ be a selected subset of semantic predicates, and write $\Phi_D(s, p)$ for the corresponding restricted signature.

Definition 10.8 (Contract-sufficient semantic footprint)**.** *A predicate subset D is* contract sufficient *on a considered arena if each of A_x, G_x, I_x (for $x \in \{s,t\}$) can be recovered from the specified footprint observations for that contract, together with any explicitly retained turn, terminality, and history conditions. Thus the footprint must determine the* actual contractual tests*, not merely resemble the initial position's failure signature.*

Proposition 10.9 (Soundness of projected observations)**.** *Suppose a paired-history matching preserves all the footprint observations from Definition 10.8 in the directions required by Definition 10.1. Then those observations suffice for the contract-compatibility checks in the selective transfer theorem.*

Proof. Each contractual membership test is determined by its footprint observations. Preserving the observations in the indicated directions therefore implies the corresponding safety implication, synchronized goal membership, and release-interface implication. Turn and terminality conditions are retained explicitly. These are exactly the conditions of Definition 10.1; the successor-matching obligations remain independently necessary. □

Proposition 10.10 (Budget transfer under pointwise cost domination)**.** *In the setting of Theorem 10.3, suppose every pair of matched transitions $s \xrightarrow{m} s'$ and $t \xrightarrow{n} t'$ in the witness obeys*

$$c^t_{p,w}(t, n) \le c^s_{p,w}(s, m),$$

where the costs are computed according to Definition 7.9, possibly using different contract-specific predicate families and nonnegative weights. Then any source worst-case introduction budget b is also a valid receiving budget.

Proof. For any complete receiving outcome, construct its paired source outcome as in Theorem 10.3. Summing the edgewise cost inequality over the synchronized plies gives a receiving total no larger than the corresponding source total. The source total is at most b by hypothesis. Because the receiving opponent outcome was arbitrary, the same bound holds for every outcome. □

Corollary 10.11 (Composition of selectively transferred macros)**.** *Suppose two source macros are individually transferred to receiving contracts using Theorem 10.3. If the* actual *receiving handoff set of the first transferred macro lies in the entry set of the second, and both preserve a common receiving safety set, their receiving phase-indexed composition is certified with horizon bounded by the sum of the two component horizons.*

Proof. The transfer theorem certifies each receiving component. The stated handoff condition is precisely the reachable-handoff compatibility condition in Theorem 7.5, which then yields the desired composite certificate and horizon bound. □

10.6 Research Consequences and Limitations

The selective rule can permit strategy transfer where full contract-respecting bisimulation fails, because unused protagonist alternatives and source-only opponent choices need not be reproduced. This is a genuine logical relaxation of the Section 8 sufficient condition, *not* evidence of search acceleration or stronger chess play. It also introduces its own obligations: correct history matching, faithfully checked opponent branching, and sound recovery of contract predicates from any proposed footprint.

For future chess-level study, we seek nontrivial footprints that are small enough to recognize computationally but rich enough to sustain specific practical macro contracts. Any automated attempt to drop a failure predicate must *recheck* both contractual observations and the opponent-response obligations; matching on fewer bits alone is not a valid transfer certificate. This work remains a mathematical semantic-descent stage, prior to building the chess engine.

11. Chess-Specific Semantic Footprints and Dependency Certificates

The selective transfer theorem of Section 10 provides a sound *criterion* for reusing a certified strategy, but it does not by itself tell us which chess features should be retained. This section begins that semantic descent with concrete, legally reachable positions. Our aim is modest and testable: identify an informative local dependency pattern, demonstrate a successful and unsuccessful transfer, and separate *footprint discovery* from *certificate verification*. The examples establish tactical reuse for a one-ply contract, not a general chess-strength result.

11.1 Candidate Footprints and Their Proof Obligations

For a history-aware chess state s, a player p, and a proposed move m, consider the following observations:

(i) the pieces occupying a specified set of squares and the vacant squares required for the move's geometric route;

(ii) the player to move, legal-move status, and any relevant castling, en-passant, or game-history conditions;

(iii) after applying m, a selected attack/defense relation and the legal defensive continuations available to the opponent.

These observations need not describe the whole board. Nevertheless, *an attractive pattern of occupied squares is not automatically a sound strategy-transfer invariant*. Unobserved pieces can introduce checks, block routes, or supply defensive replies. Our first distinction is therefore between a cheap *candidate footprint* and an audited *dependency certificate*.

Definition 11.1 (Move-relative chess footprint)**.** *For a legal chess state s and a candidate geometric move descriptor m (e.g., `h5f7`), a* move-relative footprint *is a finite record*

$$\mathsf{FP}(s, m) = (p, \mathrm{Obs}_D(s), m, \Gamma),$$

where p is the intended player, D is a selected set of concrete square-occupancy and/or semantic observations, and Γ is a list of proof obligations to be checked on each receiving state. The obligations may include

legal-move membership, a guarded transition, contractual safety, and exhaustive opponent-response checks. Agreement on Obs_D *is* candidate similarity*; the footprint is* certified for a receiving state *only after every obligation in* Γ *has been discharged there.*

The distinction deliberately prevents a circular inference: retaining only a few useful board observations does not justify silently assuming that all other legal responses are unchanged. In a complete chess engine, the proof obligations must be evaluated against a trusted legal kernel on the full history-aware state.

11.2 A Reachable Mate-in-One Family

Example 11.2 (Two positions, one tactical construction)**.** Consider the common legal prefix

$$1.\ e4\ e5 \quad 2.\ Qh5\ Nc6 \quad 3.\ Bc4.$$

Black can now play `3...a6` or `3...Nf6`, producing the two distinct legal states s_a and s_n below, with White to move:

Black's reply	FEN of the resulting position
`3...a6`	`r1bqkbnr/1ppp1ppp/p1n5/4p2Q/2B1P3/8/PPPP1PPP/RNB1K1NR w KQkq - 0 4`
`3...Nf6`	`r1bqkb1r/pppp1ppp/2n2n2/4p2Q/2B1P3/8/PPPP1PPP/RNB1K1NR w KQkq - 4 4`

In both positions `4.Qxf7#` is a legal checkmating move. The source and receiving boards are not identical: the black a-pawn has moved in s_a, whereas the black king's knight has developed in s_n. Nevertheless, White's queen on h5 has an unobstructed diagonal through g6 to f7, the black f7-pawn can be captured, and White's bishop on c4 protects the queen on f7 from capture by the black king. After `Qxf7#`, Black is in check and has no legal reply.

A candidate footprint suggested by this pair contains the white queen on h5, white bishop on c4, black king on e8, black pawn on f7, and vacancy of g6, together with the intended move `h5f7`. These are *explanatory dependencies*, not an exhaustive condition for checkmate over all chess positions. The additional check that the resulting position actually has *zero legal opponent replies* is what closes the proof obligation for this example.

Example 11.3 (A one-square obstruction refutes the motif)**.** From the same prefix Black may instead play `3...g6`. The resulting legal state s_g has FEN

`r1bqkbnr/pppp1p1p/2n3p1/4p2Q/2B1P3/8/PPPP1PPP/RNB1K1NR w KQkq - 0 4`.

Here `Qxf7` is *not* a legal queen move: the pawn on g6 interrupts the h5–f7 diagonal. The more general motif "queen h5, bishop c4, king e8, pawn f7" therefore fails as a transferable strategy guarantee unless it also records or rechecks the route. The example does not imply that White has no other useful continuation.

11.3 Local Tactical Certificates

Definition 11.4 (One-ply mate certificate)**.** *Fix a legal history-aware state* s *with player* p *to move and a move descriptor* m*. A* one-ply mate certificate *consists of a legal transition* $s \xrightarrow{m} s'$ *together with verified facts*

$$m \in \mathcal{L}(s), \qquad \mathrm{Check}(s', -p), \qquad \mathcal{L}(s') = \varnothing.$$

The last equality requires exhaustive legal-move generation for the opponent, not a visual estimate of king mobility. For the contract of checkmating within one ply*, its safety set may be all legal states and its target and release interface the set of checkmated states with the opponent to move.*

Proposition 11.5 (Soundness of the mate footprint audit)**.** *If a receiving legal chess state t satisfies the same* candidate *footprint as s for descriptor m, and the mate-certificate obligations of Definition 11.4 independently hold at t, then the one-ply policy selecting m certifies the mate contract at t. No bisimulation of the full two chess game trees is necessary.*

Proof. Legality produces a unique legal successor t' after m. The verified check and absence of legal opponent moves make t' checkmate by the rules of chess. Thus the policy reaches the designated goal and interface in one ply. No opponent decision occurs before release, so the universal adversarial obligation is vacuous for this particular one-ply contract. The argument depends on the audited facts at t, not on the candidate footprint alone. □

Remark 11.6 (Scope of the chess witness)**.** The two positive FENs are a genuine transfer example for a *short* chess-specific macro. They do not establish a nontrivial multi-ply strategy transfer, a chess-specific complete abstraction, or an efficient search reduction: checking $\mathcal{L}(t') = \varnothing$ still requires a trusted legal-move calculation. This distinction matters when comparing the mathematics with familiar tactical-pattern recognition.

11.4 Finite Dependency Support and Opponent Completeness

For longer macros the relevant footprint is not confined to the starting board. It can change after each move, including each opponent counter-repair. The appropriate object is therefore a *branch-indexed dependency record*.

Definition 11.7 (Strategy-support certificate)**.** *Let σ be a certified source strategy with horizon h, and let t be a receiving position. A* strategy-support certificate *consists of a finite tree of paired legal histories, rooted at the source and receiving positions, subject to the following conditions:*

(i) At each protagonist node, the tree records the source action selected by σ and one corresponding legal receiving move.

(ii) At each opponent node, it enumerates every *receiving legal reply, assigning each a corresponding legal source reply.*

(iii) At every paired node it records sufficient evidence for the directed contractual observations of Definition 10.1.

(iv) Every branch reaches a matching release interface in at most h plies; no unfinished branch is accepted at depth h.

The support *of the certificate is the collection of concrete square facts, semantic predicates, legality facts, and historical conditions actually queried to verify its nodes. It may vary by branch.*

Theorem 11.8 (Soundness of a finite chess-support certificate)**.** *Suppose the source strategy is certified and a proposed receiving strategy-support certificate satisfies Definition 11.7, with all recorded legal transitions and contract observations independently checked against the complete receiving chess states. Then the receiving strategy satisfies its contract for all legal opponent responses through horizon h.*

Proof. The certificate supplies the paired-history matching witness required by Definition 10.2. At protagonist nodes it supplies a legal receiving choice for the source policy's selected action. At opponent nodes it covers every receiving reply, and therefore no opponent continuation can escape the tree. Condition (iii) gives directional compatibility at each paired node, while condition (iv) ensures correct, timely release. The

soundness claim follows from Theorem 10.3; equivalently, induction over the finite tree proves the claim directly. □

Corollary 11.9 (Safe omission of irrelevant observed facts)**.** *Removing a recorded feature from a proposed footprint is sound* for a particular receiving certificate *if the omitted feature is never used in establishing any legality, successor-matching, contract, or opponent-completeness obligation, and all those obligations are reverified after removal.*

Proof. Under the stated conditions the validated witness still satisfies every premise of Theorem 11.8. The size of the explanatory footprint has changed, but the verified transition tree and its contractual guarantees have not. □

11.5 Footprint Extraction as a Research Problem

The examples motivate an increasingly refined hierarchy:

(a) *Motif discovery*: propose a sparse, human-readable set of squares and failure predicates suggesting a tactic.

(b) *Local audit*: check the proposed move and its immediate outcome on the complete chess state, as in Example 11.2.

(c) *Adversarial audit*: for longer horizons, build the branch-indexed dependency certificate covering all opponent counter-repairs.

(d) *Footprint minimization*: drop observations only when the complete transfer proof obligations remain independently valid.

A small explanatory footprint may coexist with a larger verification certificate. This distinction is useful for the proposed chess application: the mathematical proof can remain exhaustive while an interface explains only the structural dependencies decisive to the chosen strategy. Finding *compact, broadly reusable* footprints with substantially cheaper verification than ordinary bounded search remains an open empirical and mathematical problem. In particular, no result in this section proves that the five-predicate ontology of Section 3.3 already contains enough information for nontrivial chess strategy transfer.

12. Multi-Ply Chess Footprints and Adversarial Counter-Repair Closure

Section 11 established a one-ply tactical footprint and introduced the abstract notion of a branch-indexed support certificate. We now make that construction concrete over *multiple chess plies*. The relevant mathematical object is not an attractive initial arrangement of pieces alone, but a *contingent plan*: a move chosen by one player, all replies available to the opponent, and a continuation for each reply. We distinguish a complete local proof of a particular strategy from a small set of observations that suggests where that strategy may transfer.

12.1 Branch-Indexed Footprints and Response Closure

Fix a protagonist p, a horizon h, a safety set A, a first-release target G and interface I as in Definition 7.1. We work with complete history-aware chess states and a legal transition operator δ. A *policy tree* for a proposed p-strategy σ from s is obtained by unfolding legal play until the first visit to G, a terminal state, or depth h. At a protagonist node the tree follows the one move selected by σ; at an opponent node it includes *all* legal

responses. Thus the tree has OR-nodes and AND-nodes in the usual sense, but each node remains a concrete chess position with its history.

Definition 12.1 (Branch-indexed chess footprint)**.** *A* branch-indexed footprint *for* (s, σ, h) *consists of a finite policy-tree shape T and, at each node v of T, a finite observation record D_v together with its intended local proof obligations. These may concern piece locations, free move corridors, checks, attack or defense relations, legal transitions, contract membership, or the exact set of legal opponent replies. A* candidate matching *between a source tree and a receiving tree preserves their rooted decision order and the selected protagonist continuations, but is* not *a transfer certificate until every legal receiving opponent response is included and all relevant contractual obligations are independently checked.*

An opponent can invalidate a proposed repair by creating a new legal reply not present in a remembered tactical line. We therefore make the universal condition explicit.

Definition 12.2 (Opponent-closed audit tree)**.** *An* opponent-closed audit tree *of depth at most h from s is a finite, concretely verified legal policy tree with the following properties:*

(i) Every pre-release state belongs to A and is nonterminal.

(ii) Each pre-release protagonist node specifies exactly one verified legal move selected by the audited policy.

(iii) Each pre-release opponent node has one outgoing edge for every distinct legal move*, including all possible promotion choices, even when different moves reach strategically similar states.*

(iv) Every maximal branch first enters G in a state in I by depth h. At a release leaf, no further moves are required.

The tree is complete for the specified policy and contract*, not for all protagonist strategies from s.*

Theorem 12.3 (Soundness of opponent-closed chess audits)**.** *If there exists an opponent-closed audit tree from a legal chess state s for contract $(p, \{s\}, A, G, I, h)$, then its protagonist choices induce a certified strategic macro at s with horizon h.*

Proof. At the root choose the audited protagonist move whenever the protagonist is to move. After an actual opponent move, follow the outgoing edge corresponding to that exact legal move; by Definition 12.2(iii), such an edge exists for every opponent choice, including underpromotions. Continue recursively. Every possible actual play under this policy therefore follows a branch of the audited tree. Pre-release safety and nonterminal status are preserved at each internal node by (i). Finite depth bounds the number of moves, while (iv) guarantees that every branch's first release belongs to $I \subseteq G$ by depth h. This is precisely certification under Definition 7.1. □

Remark 12.4 (What the theorem requires)**.** The theorem does not allow a verifier to accept a list of *anticipated* opponent replies. The legal-move set must be regenerated from the complete receiving chess state, including promotion options, castling and en-passant rights, check evasions, and any relevant game-history rules. A list omitting one legal counter-reply is not an opponent-closed tree.

12.2 A Concrete Three-Ply Chess Strategy

The following small endgame offers a directly auditable, nontrivial opponent branch. It deliberately uses a very simple material setting: our present objective is to establish *multi-ply transfer*, not yet to solve complicated positional strategy.

Example 12.5 (Two different queen positions, one branching plan)**.** Let s_e and s_f be the following legal configurations, both with White to move, no castling or en-passant rights, and zero halfmove count:

State	FEN (White: king a3, queen as indicated; Black: king b1)
s_e; queen e2	`8/8/8/8/8/K7/4Q3/1k6 w - - 0 1`
s_f; queen f2	`8/8/8/8/8/K7/5Q2/1k6 w - - 0 1`

In both positions White begins with `1.Kb3`. Black then has exactly two legal replies: `1...Ka1` and `1...Kc1`. The complete White policy is

$$\sigma(\text{root}) = \text{Kb3}, \qquad \sigma(\ldots\text{Ka1}) = \text{Qb2\#}, \qquad \sigma(\ldots\text{Kc1}) = \text{Qc2\#}. \tag{57}$$

At the source and receiving positions, the final queen moves have the same algebraic descriptions, though their *concrete source squares* are different: `e2b2`, `e2c2` at s_e, and `f2b2`, `f2c2` at s_f.

Proposition 12.6 (Certified three-ply transfer in the queen corridor)**.** *For the two-state entry class $P = \{s_e, s_f\}$ of Example 12.5, the policy in Equation (57), with concrete queen move selected according to the current board, certifies checkmate of Black within three plies against every legal Black reply. Consequently the same* semantic *macro applies to both states even though the first position and the second differ in queen location.*

Proof. For both initial positions `Kb3` is legal: no black piece attacks b3, and the move leaves the kings nonadjacent. Immediately thereafter Black's king occupies b1, White's king b3, and the White queen lies on e2 or f2. Black has no piece other than its king. The squares a1 and c1 are unoccupied and unattacked by White, so `Ka1` and `Kc1` are legal. Every other king neighbor is attacked by the White king or queen; therefore these are *all* Black's legal replies.

After `Ka1`, the queen has a clear rank-two route from e2 or f2 to b2; `Qb2` checks the black king on a1 and is protected by White's king on b3. The queen and king collectively cover every escape square, so there is no legal Black reply. After `Kc1`, the analogous move `Qc2` is legal and protected by the king b3; again all black king escapes are covered. In both cases the terminal state is checkmate. These two continuations exhaust Black's legal responses and reach the checkmate target at the third ply. Take A to be the full set of legal history-aware states, and $G = I$ to be the checkmate states with Black to move. The resulting three-ply tree satisfies Definition 12.2; apply Theorem 12.3. □

Remark 12.7 (What is actually transferred)**.** The king move and the *policy structure* are the same in the two positions; the final queen's origin square is allowed to vary. The reusable footprint records the kings on a3 and b1, the queen in a certified e2-or-f2 corridor, White to move, the required empty rank-two queen routes, and a checked set of opponent responses after `Kb3`. This is a proved two-member family, *not* a statement that any board sharing just those highlighted squares is safe to transfer to.

12.3 An Adversarial Counter-Repair: Underpromotion

Example 12.8 (One extra pawn defeats the proposed macro)**.** Compare the first position above with the configuration

`8/8/8/8/8/K7/p3Q3/1k6 w - - 0 1`.

It has the same white king a3, white queen e2, black king b1, and the same player to move, but *also* a black pawn a2. The move `Kb3` remains legal. Yet Black now has six legal responses, not two: `Ka1`, `Kc1`, and all four

promotions `a1=Q`, `a1=R`, `a1=B`, `a1=N`. In particular, `1...a1=N+` checks the White king on b3 and prevents any checkmate by White on the next ply. Thus the claimed three-ply mate certificate fails for this *specific* initial move and macro. We do not claim that White has no alternative winning strategy.

This counterexample makes the asymmetry in Definition 10.2 visible on a chessboard: extra receiving *protagonist* moves may be irrelevant to transferring one policy, but even one additional receiving *opponent* reply must be addressed. A footprint that records the three principal pieces and White's intended move but ignores Black's promotion resource is not sufficient for this three-ply contract.

Corollary 12.9 (No valid transfer from an incomplete reply list)**.** *A proposed audit of the pawn-augmented position that contains only Black's* `Ka1` *and* `Kc1` *replies cannot satisfy Definition 12.2; and the policy* `Kb3` *followed by a one-ply checkmate response cannot meet the specified horizon-three mate contract at that state.*

Proof. The omitted `a1=N+` is a distinct legal Black move, so opponent closure fails. After that move White's king is in check, and the audited legal-move enumeration contains no White mate in one; hence this particular policy does not force checkmate on ply three. No conclusion about every other possible first move is required. □

12.4 Footprint Refinement Against Counterexamples

The examples suggest a mathematically disciplined process: *propose* a sparse footprint, *audit* the resulting receiving policy tree, and *refine* the footprint when a counter-reply is found. The next definition makes clear which property any claimed compact footprint must actually have.

Definition 12.10 (Sound footprint relative to an admission class)**.** *Let $U \subseteq \mathcal{S}$ be an explicitly specified class of legal positions and fix a source certificate C with contract (p, A, G, I, h). A feature map $F : U \to \mathcal{V}$ is* sound for transferring *C* over *U if every receiving state $t \in U$ whose feature value matches the source's value admits a verified opponent-closed receiving tree satisfying the same contract, possibly with different concrete protagonist moves. The definition requires* existence *of a verified tree at each admitted state; matching features alone, without an independent soundness argument, never constitutes a certificate.*

A restricted domain is essential. The three-piece positions in Example 12.5 have a certified common macro; allowing arbitrary unobserved adversarial pieces enlarges the domain and can invalidate the same coarse footprint, as in Example 12.8.

Proposition 12.11 (Finite counterexample-guided separation)**.** *Let U be a finite admission class, let $s \in U$ be a certified source, and let $\mathcal{D}$ be a finite catalogue of Boolean features which separates s from every $t \in U$ for which the source macro* fails *its receiving audit. Start with a subset $D \subseteq \mathcal{D}$. Whenever a failing receiver t matches s on all features in D, add any feature from $\mathcal{D}$ on which s and t differ. If the audit and failure oracle are exact, this process terminates after at most $|\mathcal{D} \setminus D|$ feature additions, and the remaining admitted states all pass the receiving audit.*

Proof. By the separation hypothesis there is a distinguishing feature for each failing receiver. If a failing receiver matches s on all currently selected features, any distinguishing feature is absent from D; adding it strictly increases D. There are at most $|\mathcal{D} \setminus D|$ available additions. At termination, if a failing receiver still matched the source on all selected features, the exact audit oracle would return it as a counterexample, contradicting termination. Hence every remaining admitted receiver passes the audit. □

Remark 12.12 (Finite refinement properties)**.** The proposition proves neither that the resulting footprint is smallest nor that feature discovery or receiving audits are computationally cheaper than bounded chess search. Its premises may fail when U is unrestricted or $\mathcal{D}$ cannot distinguish states with different adversarial outcomes. The research task is to identify useful chess failure predicates and relational features which satisfy the separation requirement over meaningful admission classes, and to replace exhaustive re-auditing where possible by sound incremental proof obligations.

12.5 Next Mathematical Questions

The three-ply example is a genuine finite contingent strategy, yet it is a small endgame construction rather than a deep middlegame plan. The next semantic-descent steps are to (i) find multi-ply repair objectives beyond forced mate, (ii) identify semantic features stable under nontrivial opponent counter-repairs, and (iii) determine when a certified footprint can replace expensive state-by-state re-verification. No playing-strength or novelty claim follows from the present examples alone.

13. Persistent Positional Repairs and Adversarial Viability

The preceding section produced finite, opponent-complete certificates for a *forcing* chess objective: checkmate within a bounded number of plies. Many positional objectives differ in kind. Placing a rook behind a passed pawn, defending a vulnerable piece, maintaining a blockade, or preserving a king shelter may be desirable without ending the game. Such a condition can be established locally and then lost to the very next opposing move. The problem is therefore not only *reachability* of a repaired state but also *persistence* inside a suitably defined semantic region. We give an exact bounded adversarial interpretation of that distinction, followed by an explicit chess counter-repair.

All states below are the complete history-aware chess states of Section 2, so the next player, legal moves, and relevant terminal-history rules remain part of the semantics. The chess illustration is deliberately elementary; neither the choice of its positional goal nor the preservation of that goal implies a corresponding game-theoretic advantage.

13.1 Positional Objectives and Retention Contracts

Definition 13.1 (Semantic positional region)**.** *Fix a protagonist p. A* semantic positional region *is a decidable set $B \subseteq \mathcal{S}$ specified by chess-relevant predicates and, optionally, a risk threshold. For example, if f describes an exposure to be repaired, g a guard to be preserved, and $H_{w,J}$ is the interaction energy introduced in Section 4.6, one may choose*

$$B_{f,g,c} = \{s \in A : f(s,p) = 0, \quad g(s,p) = 1, \quad H_{w,J}(\Phi(s,p)) \le c\}, \tag{58}$$

where A is a decidable safety envelope and c is a chosen threshold. No general chess-value or monotonicity assertion is implicit in this definition.

Ordinary bounded reachability, defined in Definition 6.3, permits a strategy to stop as soon as it first enters its target B. That is insufficient when the objective is to *retain* a positional repair after release. We accordingly distinguish a specified *post-release retention horizon* from the horizon used to reach the target.

Definition 13.2 (Terminal-inclusive predecessor and retention kernel)**.** *For a fixed protagonist p and any $X \subseteq \mathcal{S}$ let*

$$\widehat{\mathsf{CPre}}_p(X) = (\mathcal{T} \cap X) \cup \mathsf{CPre}_p(X). \tag{59}$$

Here terminal states are treated as absorbing only for the purpose of finite-horizon monitoring*: a terminal state already satisfying the contract remains a successful, completed outcome; a terminal state outside the contract does not. Define*

$$\mathsf{K}_p^0(B) = B, \tag{60}$$

$$\mathsf{K}_p^{j+1}(B) = B \cap \widehat{\mathsf{CPre}}_p\big(\mathsf{K}_p^j(B)\big), \qquad j \geq 0. \tag{61}$$

We call $\mathsf{K}_p^j(B)$ *the* j-ply adversarial retention kernel *of* B *for player* p.

Theorem 13.3 (Exact bounded positional retention)**.** *For every finite* j *and state* s*, membership* $s \in \mathsf{K}_p^j(B)$ *is equivalent to existence of a possibly history-dependent* p*-strategy under which* every *legal opponent continuation stays inside* B *at each of the next* j *plies, including the initial state. A terminal state in* B *may finish the continuation early and is then counted as safely completed. In particular, the retention kernels are descending:*

$$\mathsf{K}_p^{j+1}(B) \subseteq \mathsf{K}_p^j(B) \subseteq B. \tag{62}$$

Proof. At $j = 0$, there is no future move to inspect, so the assertion means exactly $s \in B$. Suppose the assertion holds at j. At a nonterminal state with p to move, retention for $j + 1$ plies requires $s \in B$ and *some* legal move into a state from which retention for the remaining j plies is guaranteed. At a nonterminal opponent-to-move state, it requires $s \in B$ and that *every* legal response have the same property. These are precisely the two clauses of CPre_p. If s is terminal, safe completion is possible precisely when $s \in B$, which is the additional $\mathcal{T} \cap X$ clause in Equation (59). Induction proves the equivalence. The map $X \mapsto \widehat{\mathsf{CPre}}_p(X)$ is monotone. Since $\mathsf{K}_p^1(B) \subseteq B = \mathsf{K}_p^0(B)$, induction using this monotonicity establishes Equation (62). □

Theorem 13.4 (Composition of retention horizons)**.** *For all nonnegative integers* j, k*,*

$$\mathsf{K}_p^{j+k}(B) = \mathsf{K}_p^j\big(\mathsf{K}_p^k(B)\big). \tag{63}$$

Proof. Suppose $s \in \mathsf{K}_p^{j+k}(B)$. By Theorem 13.3, there is a strategy which keeps B true for $j + k$ plies. At every state reached during the first j plies, its remaining continuation maintains B for at least k further plies, because the original strategy still has at least k moves of guaranteed safety available. Thus all these states belong to $\mathsf{K}_p^k(B)$ and the original prefix is a strategy witnessing $s \in \mathsf{K}_p^j(\mathsf{K}_p^k(B))$.

Conversely, let a strategy maintain $\mathsf{K}_p^k(B)$ for j plies from s. This keeps B true during the first j plies since $\mathsf{K}_p^k(B) \subseteq B$. At each possible nonterminal state reached at the end of that prefix, membership in $\mathsf{K}_p^k(B)$ supplies a k-ply retention strategy. Select such a continuation contingent on the actual reached history. Concatenation retains B through $j + k$ plies against every opponent choice. Safe terminal states are handled by the absorbing convention. Hence $s \in \mathsf{K}_p^{j+k}(B)$. □

13.2 From Certified Repair to Certified Persistence

The macro contracts of Definition 7.1 specify a safety envelope, a target, and an admissible first-release interface. They do not, by themselves, certify how long the released positional condition can be maintained. The retention kernel supplies precisely this missing postcondition.

Definition 13.5 (Reach-and-retain positional macro)**.** *Let* $\mathfrak{M} = (p, P, A, G, I, h, \sigma)$ *be a certified macro, and let* $B \subseteq A$ *be a semantic positional region. The macro is* k-retentive for B *when*

$$I \subseteq G \cap \mathsf{K}_p^k(B). \tag{64}$$

The reach-and-retain contract *requires safe entry into G through I within h plies, followed by preservation of B over the next k plies, with arbitrary legal opponent responses throughout.*

Theorem 13.6 (Sound handoff from repair to positional maintenance)**.** *Suppose $\mathfrak{M} = (p, P, A, G, I, h, \sigma)$ is certified and satisfies Equation (64). Then there exists a phase-indexed strategy from every $s \in P$ which (i) reaches G for the first time in I within h plies while remaining in A and (ii) maintains B for the following k plies, including every legal opponent response. The total duration is at most $h + k$ plies, excluding harmless terminal padding.*

Proof. Execute σ to its first release. Certification ensures that the release occurs within h plies, remains inside A, and belongs to I. By Equation (64) this reached state lies in $\mathsf{K}_p^k(B)$. Apply Theorem 13.3 to select a k-ply maintenance strategy for that particular release state, and switch to it without consuming a chess ply. The policy is contingent on the observed release history. All preceding opponent replies are covered by macro certification; all following replies are covered by the maintenance strategy. Since $B \subseteq A$, safety is retained throughout both phases. □

Corollary 13.7 (Monotonicity under relaxed positional requirements)**.** *If $B_1 \subseteq B_2$, then $\mathsf{K}_p^k(B_1) \subseteq \mathsf{K}_p^k(B_2)$ for every finite k. Consequently, a certificate for the stricter positional region B_1 is also valid for the relaxed region B_2 over the same horizon.*

Proof. The base case follows from inclusion of regions, and the inductive step follows by monotonicity of $\widetilde{\mathsf{CPre}}_p$ and intersection. The certificate claim is Theorem 13.3 applied to the inclusion. □

13.3 A Concrete Positional Repair and Its Counter-Repair

We now use actual chess positions rather than a generic adversarial arena. Let p be White and let B require all of the following: (i) a White rook on `a1`; (ii) a White passed pawn on `a4`; (iii) the rook on `a1` is not attacked by Black; and (iv) White is not in check. We use the standard geometric meaning of a passed pawn: no opposing pawn on its file or adjacent files ahead of it toward promotion. The predicate is decidable from a complete board state. It is *not* a claim that White has a forced win or that `Ra1` is objectively the best move.

Example 13.8 (Rook-behind-passed-pawn repair)**.** Consider these legal positions, each with White to move and no castling or en-passant rights:

Case	FEN
Control s_0	`6k1/7p/8/8/P7/8/4K3/7R w - - 0 1`
Counter-repair s_1	`6k1/7p/8/8/P7/1p6/4K3/7R w - - 0 1`

Both have White king `e2`, rook `h1`, and pawn `a4`; Black has king `g8` and pawn `h7`. The second position additionally has a Black pawn on `b3`, *behind* White's passed pawn. In both positions `1.Ra1` is legal and enters B: White's rook is now behind the passed pawn and is not attacked.

In the control, Black has exactly six legal replies: `...Kf8`, `...Kh8`, `...Kf7`, `...Kg7`, `...h6`, and `...h5`. All six preserve B. Writing $t_0 = \delta(s_0, \mathrm{Ra1})$, we have

$$t_0 \in \mathsf{K}_{\mathrm{W}}^1(B). \tag{65}$$

In the second position Black has an additional, seventh legal reply, `1...b2`. A Black pawn on `b2` attacks `a1`; hence this move attacks the White rook. Writing $t_1 = \delta(s_1, \mathrm{Ra1})$, we obtain

$$t_1 \in B \setminus \mathsf{K}_{\mathrm{W}}^1(B). \tag{66}$$

The White pawn remains passed, and the rook remains behind it, but the *unattacked-rook guard* fails after the opponent's counter-repair.

Proposition 13.9 (A concrete one-reply separation)**.** *The two legal chess states of Example 13.8 agree on the immediate positional repair achieved by* `Ra1`*, but only the control state certifies retention of B through every immediate Black response. Consequently, a footprint recording only a passed pawn and a rook behind it is insufficient for* 1*-ply positional retention over a class admitting both states.*

Proof. The rook's path `h1-a1` is clear because White's king occupies `e2`. Both boards have a White pawn `a4`; no Black pawn lies ahead of it on the `a`- or `b`-file. The move `Ra1` is legal and establishes all the defining conditions of B in both states. In the control, the legal Black replies are exactly the six listed in Example 13.8; checking the resulting boards shows that none attacks the rook on `a1`, removes the pawn, or checks White. Hence $t_0 \in \mathsf{K}^1_{\mathrm{W}}(B)$ by Theorem 13.3. In the second board, the extra Black pawn can move `b3-b2` legally. From `b2` it attacks `a1`, violating B. Since Black is to move at t_1, that single legal reply shows $t_1 \notin \mathsf{K}^1_{\mathrm{W}}(B)$, despite $t_1 \in B$. Thus the coarser footprint cannot certify retention in both states. □

Remark 13.10 (Scope of the counter-repair)**.** The witness `...b2` refutes this *specified retention contract*, not every possible White plan. It neither proves that White is lost nor excludes a later rook retreat or a different first move. Moreover, the positional region B is a deliberately narrow formal objective; a longer-horizon engine would need further conditions and controlled comparisons before treating its satisfaction as a useful measure of chess strength.

13.4 Counter-Repair Certificates and Retention Depth

Proposition 13.11 (One-step adversarial refutation)**.** *Let $t \in B$ be nonterminal, with the opponent of p to move. Then*

$$t \notin \mathsf{K}^1_p(B) \quad \Longleftrightarrow \quad \exists r \in \mathcal{L}(t) : \delta(t, r) \notin B. \tag{67}$$

Thus a legal opposing reply together with a verified failure of B is a complete one-ply counter-repair certificate.

Proof. Since $t \in B$ and is nonterminal, Equations (22) and (61) reduce membership in $\mathsf{K}^1_p(B)$ to the assertion that every legal opponent reply leads into B. Negating that universal statement gives exactly the right-hand side. □

Definition 13.12 (Bounded positional retention depth)**.** *For a specified bound $H \in \mathbb{N}$ and positional region B, define the* bounded retention depth

$$d^H_{p,B}(s) = \begin{cases} -1, & s \notin B, \\ \max\{0 \le j \le H : s \in \mathsf{K}^j_p(B)\}, & s \in B. \end{cases} \tag{68}$$

This number compares guaranteed duration of a particular semantic condition *within the bound H, not minimax value. A candidate that attains B with depth zero repairs a feature locally but supplies no even-one-ply persistence guarantee. In Example 13.8, for $H \ge 1$, $d^H_{\mathrm{W},B}(t_0) \ge 1$ but $d^H_{\mathrm{W},B}(t_1) = 0$.*

13.5 Finite-Arena Viability and Remaining Research Questions

On a finite, explicitly explored state arena, the descending sequence in Equation (62) eventually stabilizes. Write $\mathsf{K}^\infty_p(B)$ for its stabilized value *on that finite arena*; transitions must be closed within the considered arena or interpreted with a stated boundary condition.

Theorem 13.13 (Finite-arena positional viability kernel)**.** *On a finite closed turn-based arena with n states, the sequence* $\mathsf{K}_p^j(B)$ *stabilizes after at most n strict decreases. Its limit is the greatest fixed point of*

$$F_B(X) = B \cap \widehat{\mathsf{CPre}}_p(X), \tag{69}$$

and consists exactly of the states from which p can keep B true indefinitely against every opposing choice, subject to the same safe-terminal convention.

Proof. The kernels form a descending chain of subsets of a finite n-element arena. Each strict decrease removes at least one state, so no more than n strict decreases are possible. The stabilized set K^* satisfies $F_B(K^*) = K^*$. For any fixed point $Y = F_B(Y)$, we have $Y \subseteq B$, and induction gives $Y \subseteq \mathsf{K}_p^j(B)$ for every j; hence $Y \subseteq K^*$ and K^* is the greatest fixed point. At a protagonist state in K^*, select a legal successor in K^*, which exists by the fixed-point equation; at an opponent state, all legal successors already lie in K^*. Iterating these choices keeps the play in B forever or ends safely at a terminal state in B. Conversely, any strategy guaranteeing such indefinite retention also guarantees every finite horizon, so its starting state lies in every $\mathsf{K}_p^j(B)$ and therefore in K^*. □

The viability-kernel construction is related to standard safety-game and invariant-set reasoning [10]. The present contribution is a chess-semantic *specification* that connects this construction to the failure ontology, repaired conditions, and certified strategy handoffs developed earlier in this manuscript; it is not a claim of priority for the fixed-point mathematics.

Three problems remain central to the semantic descent. First, which chess positional predicates admit useful retention certificates without degenerating into elementary tautologies? Second, can a *small* footprint determine the needed adversarial predecessor facts over a meaningful class of middlegame positions? Third, how should multiple incompatible retention objectives be compared without silently identifying a semantic heuristic with chess minimax value? These are mathematical and empirical questions, not results established by the present example.

14. Competing Positional Repairs and Multiobjective Adversarial Strategy

The preceding section defined a retention kernel for one positional region B. Yet chess positions exhibit interacting objectives: protecting a piece may impede its activity, occupying a defensive square may prevent support for a passed pawn, and repairing pawn structure may weaken king shelter. The problem is not to assign arbitrary scalar weights to these desiderata, but first to determine *which combinations are simultaneously achievable by one strategy under the same legal opponent replies*. This section introduces a multiobjective retention calculus and distinguishes simultaneous strategic compatibility from the existence of separate certificates. The resulting order is a *partial* order over declared strategic objectives and admits explicit tradeoff analysis.

The general setting of multiobjective adversarial synthesis and Pareto reasoning is established in the literature [10, 12]. Our formulation specializes these principles to chess failure predicates, the single-policy requirement, and checkable counter-repair obligations.

14.1 Repair Objectives and the Single-Strategy Requirement

Definition 14.1 (Portfolio of positional repair obligations)**.** *Fix* $p \in \{\mathrm{W}, \mathrm{B}\}$*, a finite collection of decidable positional regions* $\mathbf{B} = (B_1, \ldots, B_r)$ *with* $r \geq 2$*, and finite retention requirements* $\mathbf{d} = (d_1, \ldots, d_r)$*, where* $d_i \in \{-1, 0, 1, \ldots, H\}$*. The value* $d_i = -1$ *denotes an inactive obligation;* $d_i \geq 0$ *requires* B_i *at the present*

state and through d_i further plies. The portfolio contract $(\mathbf{B}, \mathbf{d})$ *is satisfied from a state s when there exists* one *possibly history-dependent p-strategy σ such that, against every legal opponent continuation, B_i holds at every elapsed ply $t \le d_i$ for each active i. Safe terminal outcomes are monitored by the absorbing convention of Definition 13.2. Let $\mathsf{J}_p^{\mathbf{d}}(\mathbf{B})$ be the set of states satisfying this portfolio contract.*

The *same* σ must satisfy all active obligations. The weaker assertion that each B_i admits its *own* strategy permits incompatible choices and therefore cannot justify an intersection of certificates.

For $\mathbf{d}$ as above, define the currently active region and the decrement operation by

$$U_{\mathbf{B}}(\mathbf{d}) = \bigcap_{i : d_i \ge 0} B_i, \qquad \text{(empty intersection interpreted as } \mathcal{S}\text{)}, \tag{70}$$

$$(\partial \mathbf{d})_i = \begin{cases} d_i - 1, & d_i \ge 0, \\ -1, & d_i = -1. \end{cases} \tag{71}$$

and put $m(\mathbf{d}) = \max\{d_i : d_i \ge 0\}$, with $m = -1$ if all components are inactive.

Theorem 14.2 (Exact multiobjective adversarial retention)**.** *For every finite* $\mathbf{d}$*, the portfolio winning region is characterized recursively by*

$$\mathsf{J}_p^{\mathbf{d}}(\mathbf{B}) = \begin{cases} U_{\mathbf{B}}(\mathbf{d}), & m(\mathbf{d}) \le 0, \\ U_{\mathbf{B}}(\mathbf{d}) \cap \widehat{\mathsf{CPre}}_p\left(\mathsf{J}_p^{\partial \mathbf{d}}(\mathbf{B})\right), & m(\mathbf{d}) > 0. \end{cases} \tag{72}$$

Consequently, for all nonnegative d_i,

$$\mathsf{J}_p^{\mathbf{d}}(\mathbf{B}) \subseteq \bigcap_{i=1}^{r} \mathsf{K}_p^{d_i}(B_i). \tag{73}$$

The inclusion may be strict, even for two objectives and a one-ply horizon.

Proof. Proceed by induction on $m(\mathbf{d})$. If all active requirements expire at the current state, satisfaction is exactly membership in their intersection $U_{\mathbf{B}}(\mathbf{d})$. Otherwise the present state must lie in $U_{\mathbf{B}}(\mathbf{d})$. At a nonterminal protagonist position there must exist one legal move to a state satisfying every remaining obligation $\partial \mathbf{d}$; at an opponent position *all* legal replies must do so. This is precisely the controlled predecessor of the recursively defined portfolio region. A terminal position already in the current region remains there under the specified monitoring convention, and every future active requirement is a subset of those already required, so the terminal-inclusive predecessor is exact. This proves Equation (72). A portfolio-winning strategy witnesses each individual requirement by restriction, giving Equation (73) via Theorem 13.3. Strictness is witnessed by Example 14.3 below. □

Example 14.3 (Separate winning strategies with no joint winner)**.** Consider a finite abstract turn-based arena with a nonterminal protagonist state s and exactly two legal successors x, y, both safely terminal. Set $B_1 = \{s, x\}$ and $B_2 = \{s, y\}$. The choice $s \to x$ witnesses $s \in \mathsf{K}_p^1(B_1)$; the different choice $s \to y$ witnesses $s \in \mathsf{K}_p^1(B_2)$. Nevertheless, $B_1 \cap B_2 = \{s\}$ and neither successor belongs to it, so

$$s \in \mathsf{K}_p^1(B_1) \cap \mathsf{K}_p^1(B_2) \quad \text{but} \quad s \notin \mathsf{K}_p^1(B_1 \cap B_2). \tag{74}$$

This is an abstract countermodel to a general algebraic equality, not an assertion about a particular chess position.

Corollary 14.4 (Equal-horizon conjunction and strictness)**.** *For each $j \geq 0$,*

$$\mathsf{J}_p^{(j,\dots,j)}(B_1,\dots,B_r) = \mathsf{K}_p^j\left(\bigcap_{i=1}^{r} B_i\right) \subseteq \bigcap_{i=1}^{r} \mathsf{K}_p^j(B_i), \tag{75}$$

and equality with the rightmost intersection is not valid in general, already for $r = 2$, $j = 1$.

Proof. Requiring each B_i at each monitored step is exactly requiring its intersection at each such step; apply Theorems 14.2 and 13.3. Strictness follows from Example 14.3. □

14.2 Compatible Repair Choices and Delayed Conflicts

Definition 14.5 (One-ply admissible repair choices)**.** *At a nonterminal protagonist state $s \in B$ define*

$$\mathsf{A}_p(s;B) = \{m \in \mathcal{L}(s) : \delta(s,m) \in B\}. \tag{76}$$

A pair of one-ply repair contracts B_1, B_2 is immediately compatible at *s if $\mathsf{A}_p(s;B_1) \cap \mathsf{A}_p(s;B_2) \neq \varnothing$. At a nonterminal opponent state, compatibility means that* every *legal reply remains in both regions; there is no protagonist move to select at that state.*

Proposition 14.6 (Exact one-ply compatibility test)**.** *Suppose $s \in B_1 \cap B_2$ is nonterminal. If $\pi(s) = p$, then*

$$s \in \mathsf{K}_p^1(B_1 \cap B_2) \iff \mathsf{A}_p(s;B_1) \cap \mathsf{A}_p(s;B_2) \neq \varnothing. \tag{77}$$

If $\pi(s) \neq p$, the corresponding condition is $\forall m \in \mathcal{L}(s)$, $\delta(s,m) \in B_1 \cap B_2$. For horizons longer than one ply, a nonempty intersection of immediate choices is not *in general a sufficient joint-retention certificate.*

Proof. At a protagonist state, membership in the one-ply retention kernel means that some legal successor satisfies both predicates. Distribute conjunction over the predicate describing each fixed successor to obtain Equation (77). At an opponent state, the controlled predecessor uses universal quantification, yielding the stated condition. At larger horizons, membership of an immediate successor in each separate kernel may be witnessed by different future protagonist choices; the single-strategy obstruction of Example 14.3 can reappear one ply later. □

A useful distinction follows. *Local incompatibility* means no move meets both immediate repair targets. *Adversarial incompatibility* means that, although a common move may exist, no single continuation withstands all later opponent replies while maintaining the portfolio. The latter is detected by the recursive region J, not by an immediate signature comparison or by naively intersecting independently computed strategies.

14.3 A Concrete Chess Conflict: Support Versus Counter-Pawn Pressure

Example 14.7 (Two legal but competing rook repairs)**.** Return to the legal chess position s_1 of Example 13.8, with White to move:

$$\texttt{6k1/7p/8/8/P7/1p6/4K3/7R w - - 0 1}. \tag{78}$$

White has king `e2`, rook `h1`, and a passed pawn `a4`; Black has king `g8`, pawns `h7` and `b3`. Define the *immediate* post-move objectives

$$B_{\text{sup}} = \{t : \text{White's rook is unattacked on } \texttt{a1} \text{ behind the passed } \texttt{a4} \text{ pawn}\}, \tag{79}$$

$$B_{\text{press}} = \{t : \text{White's rook directly attacks the Black pawn on } \texttt{b3}\}. \tag{80}$$

The legal move `1.Ra1` reaches B_{sup} but not B_{press}; the legal move `1.Rb1` reaches B_{press} but not B_{sup}. With the single White rook fixed on `a1`, it cannot attack a pawn on `b3` along a rank or file, and so no legal *one-move* repair from this state attains both objectives.

This local incompatibility also has an adversarial sequel: after `1.Ra1`, the legal reply `1...b2` attacks the rook on `a1` and defeats B_{sup}, just as in Example 13.8. The two objectives describe distinct immediate forms of positional maintenance; neither is asserted to be better in game-theoretic value. Nor does the example rule out a longer strategy that trades off these objectives over time.

Proposition 14.8 (Chess witness of incompatible immediate repair targets)**.** *From the complete chess state in Example 14.7, both sets of legal immediate repairing moves*

$$T_{\text{sup}}(s_1) = \{m \in \mathcal{L}(s_1) : \delta(s_1, m) \in B_{\text{sup}}\},$$
$$T_{\text{press}}(s_1) = \{m \in \mathcal{L}(s_1) : \delta(s_1, m) \in B_{\text{press}}\}$$

are nonempty, whereas $T_{\text{sup}}(s_1) \cap T_{\text{press}}(s_1) = \varnothing$.

Proof. The squares from `h1` through `b1` are clear, so both rook moves named in the example are legal. On `a1`, the rook sits behind the passed `a4` pawn but does not share either a rank or a file with the Black pawn `b3`. On `b1`, the rook attacks `b3` along the unobstructed `b`-file via `b2`, but is not behind the `a4` pawn on `a1`. Since White has only one rook, any immediate move satisfying B_{sup} leaves that rook on `a1`, which precludes B_{press} while the Black pawn remains on `b3`. A White rook move cannot simultaneously relocate that Black pawn, so no one-ply White move satisfies both conditions. □

Remark 14.9 (Scope of the concrete conflict witness)**.** The rook example establishes an immediate trade-off between *specified predicates*, not a strategic evaluation of `Ra1` versus `Rb1`. Unlike Example 14.3, its starting state is outside both objective regions, so it is a one-step *reachability conflict*, not a counterexample to an intersection identity for retention kernels. The examples therefore establish complementary and deliberately distinct claims.

14.4 Adversarial Retention Profiles and Pareto Choices

Definition 14.10 (Guaranteed retention profile)**.** *Fix* $s \in \mathcal{S}$*, a horizon* $H \geq 0$*, and a finite portfolio* $\mathbf{B} = (B_1, \ldots, B_r)$*. For a protagonist policy* σ *defined at all its decision histories through* H *plies, let* $\text{Runs}_H(s, \sigma)$ *denote the complete length-*H *outcome histories against all opponent replies, with terminal plays padded by their last state. Define*

$$q_i^H(s, \sigma) = \max\big(\{-1\} \cup \big\{\, j \in \{0, \ldots, H\} : \forall \gamma \in \text{Runs}_H(s, \sigma), \quad \forall\, 0 \leq t \leq j, \gamma_t \in B_i \big\}\big). \tag{81}$$

This convention agrees with terminal-inclusive monitoring in Definition 13.2. The guaranteed profile *is the integer vector* $q^H(s, \sigma) = (q_1^H, \ldots, q_r^H)$*, ordered componentwise. A value* -1 *signifies failure at the starting state; a value* 0 *means the condition is initially satisfied with no further safety guarantee. These are guarantee depths, not expected payoffs or material evaluations.*

Definition 14.11 (Certified Pareto frontier)**.** *Let* $Q_p^H(s; \mathbf{B})$ *consist of profiles* $q^H(s, \sigma)$ *from all legal* H*-ply protagonist policies at* s*. A profile* u strictly dominates v *if* $u_i \geq v_i$ *for every* i *and* $u_i > v_i$ *for at least one. Define the* certified Pareto frontier

$$\text{Pareto}_p^H(s; \mathbf{B}) = \{q \in Q_p^H(s; \mathbf{B}) : \nexists q' \in Q_p^H(s; \mathbf{B}) \text{ that strictly dominates } q\}. \tag{82}$$

Theorem 14.12 (Existence and completeness of the finite Pareto frontier)**.** *For each fixed state s, finite H and nonempty finite portfolio, the set* $\mathrm{Pareto}_p^H(s;\mathbf{B})$ *is nonempty. Every achievable profile is either on the frontier or is strictly dominated by some frontier profile. Moreover, for any threshold vector $\mathbf{d} \in \{0,\dots,H\}^r$,*

$$s \in \mathsf{J}_p^{\mathbf{d}}(\mathbf{B}) \iff \exists q \in Q_p^H(s;\mathbf{B})\ \forall i:\ q_i \geq d_i. \tag{83}$$

Proof. At a fixed finite depth, every chess state has finitely many legal successors. Consequently the legal depth-H game tree from s is finite and has at least one complete protagonist policy (possibly vacuous when s is terminal). Each profile lies in the finite set $\{-1,0,\dots,H\}^r$; hence the nonempty set of achievable profiles has maximal elements under componentwise order. Repeatedly moving from a dominated profile to a strictly dominating one terminates at such a maximal element, establishing the first two assertions. By Definitions 14.1 and 14.10, a single policy guarantees every threshold d_i exactly when its profile meets all the indicated componentwise bounds. This is Equation (83). □

Corollary 14.13 (Priority-constrained strategic selection)**.** *If $s \in \mathsf{J}_p^{(h_1,0,\dots,0)}(\mathbf{B})$, then at least one certified profile preserves B_1 for h_1 plies. Among all policies satisfying this hard constraint, there exists one whose retention profile is Pareto-maximal under that constraint. Imposing a priority therefore selects among certified trade-offs* without *identifying a scalar semantic score with chess minimax value.*

Proof. By Theorem 14.12, the hard-constraint policy set is nonempty and its achievable profiles form a finite nonempty partially ordered set. It has a maximal element. □

14.5 Counter-Repair Refutations and the Next Descent

Proposition 14.14 (One-step portfolio refutation)**.** *Let $m(\mathbf{d}) > 0$, let $s \notin \mathcal{T}$ satisfy $s \in U_{\mathbf{B}}(\mathbf{d})$, and let $\mathbf{d}' = \partial\mathbf{d}$. Then:*

(i) If $\pi(s) \neq p$, failure $s \notin \mathsf{J}_p^{\mathbf{d}}(\mathbf{B})$ is equivalent to existence of a legal reply m with $\delta(s,m) \notin \mathsf{J}_p^{\mathbf{d}'}(\mathbf{B})$.

(ii) If $\pi(s) = p$, the same failure is equivalent to every legal move m satisfying $\delta(s,m) \notin \mathsf{J}_p^{\mathbf{d}'}(\mathbf{B})$.

These assertions yield finite AND/OR refutation obligations; a counter-repair witness need not violate a positional region immediately if it instead moves the play outside the appropriate remaining-horizon portfolio region.

Proof. Under the hypotheses, the present-state conjunct in Equation (72) already holds. At an opponent node the controlled predecessor is universal over all replies, so its failure is witnessed by at least one successor outside the recursively required set. At a protagonist node the predecessor requires the existence of a legal successor in that set; its failure means none exists. Each condition is exactly as stated. □

The next research problem is to find compact *conflict footprints*: minimal chess observations sufficient to certify that a finite portfolio is incompatible or to identify the Pareto-efficient strategic alternatives. A repair-algebra representation could expose which predicates a repair preserves, which it consumes, and which counter-repairs become available. Such footprints are candidates for later semantic compilation, not an established substitute for legal adversarial search.

Our point is narrowly mathematical: choosing between two repairs requires a *joint adversarial certificate*, not the conjunction of two independent claims. Whether these contracts predict chess strength is an empirical question left for subsequent development.

15. Compiler Architecture

The mathematical development now provides decidable chess failure predicates, guarded local repairs, finite adversarial certificates, strategy-transfer conditions, positional retention kernels, and multiobjective repair portfolios. A semantic compiler should preserve these objects as *checkable assertions about legal chess states*, not merely translate a move into an explanatory sentence. This section specifies an architecture and proves conditional properties of its outputs. It is a paper-level specification, not a report of a completed chess engine. The distinction between an untrusted producer and an independent certificate checker is related to established proof-carrying computation [13, 7].

15.1 Source Semantics and History-Preserving Canonicalization

The input language admits a validated initial history-aware state, a possibly empty legal move sequence, and an explicit adjudication policy. A move sequence may be supplied in standard chess notation, but its parsing is not itself evidence that the resulting moves are legal. In particular, a board diagram or a bare FEN record need not supply the repetition history necessary to decide some draw claims. The chess laws distinguish claimable draws from automatically terminated games [14]; an implementation must choose and record the relevant adjudication convention. A concrete move library such as `python-chess` provides useful parsing and legal-move facilities but does not eliminate the need to retain that context [15].

Definition 15.1 (Faithful state encoding)**.** *Let* $\mathsf{Enc} : \mathcal{S} \to \mathsf{CState}$ *and* $\mathsf{Dec} : \mathsf{CState} \rightharpoonup \mathcal{S}$ *be a canonical encoding and a partial decoding function. An encoding is* faithful on a considered arena $D \subseteq \mathcal{S}$ *if*

$$\mathsf{Dec}(\mathsf{Enc}(s)) = s \qquad (s \in D), \tag{84}$$

where equality means equality of every component relevant to π*,* $\mathcal{T}$*,* $\mathcal{L}$*,* δ*, and the declared semantic predicates. An identifier for an encoded state is an indexing aid; its equality must not be accepted as proof of state equality unless collisions have been excluded or the underlying states are compared.*

Remark 15.2 (Full chess-state representation)**.** A legality-only tool may operate on board, side to move, castling rights, and en-passant information. A compiler that also certifies repetition-based adjudication must retain sufficient move history. Different histories that render the same pieces on the same squares need not be interchangeable for that stronger contract. The semantic scope of a certificate must therefore be declared before any state abstraction is applied.

15.2 The Typed Semantic Intermediate Representation

For a fixed protagonist p, write $\mathsf{Obs}_p(s)$ for the finite vector of declared predicate values and any derived facts whose algorithms have been specified. At a minimum it contains $\Phi(s, p)$; richer instantiations may record guard values, active portfolio obligations, terminal status, and the identities of the predicates on which a conclusion depends. The observation vector is not, by itself, a complete chess state.

Definition 15.3 (Semantic edge record)**.** *For* $s \in \mathcal{S} \setminus \mathcal{T}$ *and* $m \in \mathcal{L}(s)$*, let* $s' = \delta(s, m)$*. A* semantic edge record *is*

$$\mathsf{E}_p(s, m) = \big(\mathsf{Enc}(s),\ m,\ \mathsf{Enc}(s'),\ \mathsf{Obs}_p(s),\ \mathsf{Obs}_p(s'),\ \Delta(s, m, p),\ \mathsf{Claim},\ \mathsf{Evidence}\big). \tag{85}$$

Here Δ *is the semantic change vector of Section 4. The claim field indicates a* specific *assertion—for example, a guarded local repair, a bounded reachability guarantee, a retention contract, or a vector of portfolio guarantees. The evidence field provides the material needed by a verifier to reconstruct that assertion. An empty claim field is allowed: an edge may be legal without being certified as strategically useful.*

The representation has three distinct layers, preventing a common category error between facts, certificates, and preferences:

Layer	Contents	Validation obligation
Concrete	full state, turn, legal move, successor	independent rules-based replay
Semantic	Boolean failures, guards, repair introductions	independent predicate evaluation
Strategic	finite policy tree, retention or portfolio contract, preference score	exhaustive bounded checking for a guarantee; scores remain heuristic

15.3 Semantic Lowering and a Commuting-Semantics Result

A compiler pass *lowers* a validated concrete edge to the intermediate representation. The word *validated* is important: an untrusted generator may propose the move and even propose its evidence, but it does not thereby establish either.

Definition 15.4 (Correct edge checker)**.** *A checker* $\mathsf{CheckEdge}_p$ *accepts a proposed record only if it independently decodes its source and target, verifies that the source is nonterminal, regenerates its full legal move set, checks* $m \in \mathcal{L}(s)$*, recomputes* $\delta(s, m)$ *and both observation vectors, and compares the resulting values with the record. It also recomputes the stated semantic change vector. No heuristic score can override a failed comparison. Acceptance of an edge record certifies only its concrete and predicate fields: a strategic claim in its claim field requires its own contract-specific checker.*

Theorem 15.5 (Faithful bounded semantic compilation)**.** *Fix a considered arena* D*, a finite legal chess path* $\gamma = (s_0, m_1, s_1, \ldots, m_h, s_h)$ *within* D*, and a finite family of total, correct predicate evaluators. Assume Equation (84), exact legal-move and transition computation, and that every emitted edge record is accepted by* $\mathsf{CheckEdge}_p$*. Then the emitted records decode to the original legal path, their reported semantic observations are exact, and*

$$\sum_{i=1}^{h} \Delta(s_{i-1}, m_i, p) = \Phi(s_0, p) - \Phi(s_h, p). \tag{86}$$

Their separate repair/introduction counts are also equal to the ledger counts of Theorem 9.8.

Proof. For the first record, faithful decoding gives s_0. Checker acceptance independently establishes both legal membership and that the decoded target equals $\delta(s_0, m_1) = s_1$. Applying the same argument to each subsequent record, inductively, recovers the entire concrete path. By correctness of the predicate evaluators, every observation is $\mathsf{Obs}_p(s_i)$ and every change vector is $\Phi(s_{i-1}, p) - \Phi(s_i, p)$. Summation cancels the intermediate vectors, proving Equation (86). Each step's gross repair and introduction indicators is checked from the same before/after predicates; summing them recovers the ledger by definition. □

This theorem is a *representation-preservation* result. It does not state that the emitted move is best, that the ontology captures every strategically meaningful feature, or that emitting the records is computationally inexpensive.

15.4 Independent Checking of Adversarial Portfolio Certificates

The preceding theorem establishes factual correctness along a *single path*. A guaranteed chess strategy is a branching object: the protagonist makes selected choices, but *every* legal opponent answer must be considered.

Accordingly, an adversarial certificate is an AND/OR tree whose nodes contain complete chess states and remaining repair obligations.

Definition 15.6 (Portfolio certificate checker)**.** *Fix a protagonist p, decidable positional regions $\mathbf{B} = (B_1, \dots, B_r)$ and a finite deadline vector $\mathbf{d}$ as in Definition 14.1. A checker* $\mathsf{CheckPort}(s, \mathbf{d}, C)$ *verifies, recursively:*

(i) the present state belongs to every currently active region $U_{\mathbf{B}}(\mathbf{d})$;

(ii) if all active obligations expire now, or if the present state is terminal, it accepts without demanding fictitious additional moves;

(iii) otherwise, if $\pi(s) = p$, the certificate identifies one legal move and contains a recursively accepted child at $\delta(s, m)$ with deadlines $\partial\mathbf{d}$;

(iv) otherwise, the checker regenerates every *legal opponent move and requires an accepted child at each corresponding successor, again with deadlines $\partial\mathbf{d}$.*

The states, legal move sets, predicate values, and terminal status must be recomputed using the trusted concrete semantics. The certificate is rejected for an omitted opponent edge, an illegal move, an incorrect state, or an unproved active condition.

Theorem 15.7 (Soundness of checked portfolio compilation)**.** *Assume the concrete chess kernel and the semantic region evaluators are correct. If* $\mathsf{CheckPort}(s, \mathbf{d}, C)$ *accepts, then*

$$s \in \mathsf{J}_p^{\mathbf{d}}(\mathbf{B}). \tag{87}$$

The moves selected at protagonist nodes define a single contingent strategy satisfying all active positional obligations against every legal opponent continuation over the declared finite horizons.

Proof. Induct on the largest active deadline $m(\mathbf{d})$. At a nonpositive deadline the checker has independently verified $U_{\mathbf{B}}(\mathbf{d})$, which equals the required region by Equation (72). For a larger deadline, the checker first verifies membership in $U_{\mathbf{B}}(\mathbf{d})$. At a terminal state the terminal-inclusive convention in Definition 13.2 satisfies the remaining obligations. Otherwise, at a protagonist node acceptance certifies one legal successor satisfying the decremented portfolio by induction; at an opponent node it certifies *all* legal successors. This is precisely the predecessor clause of Equation (72). The selected protagonist edges form the required history-dependent policy. □

Corollary 15.8 (Heuristic independence of an accepted guarantee)**.** *Under the assumptions of Theorem 15.7, changing or replacing the candidate generator, evaluation function, or search-order heuristic cannot invalidate an already accepted portfolio certificate, provided the checker, its specification, and the underlying chess-state semantics are unchanged.*

Proof. Acceptance and Theorem 15.7 depend on independently replayed transitions, exact predicates, and the certificate tree, not on the mechanism that proposed it. Hence changing an untrusted producer leaves the accepted implication unchanged. □

15.5 The Abstraction Boundary and Independent Verification

A small semantic footprint can suggest a promising strategy from Sections 10 and 11, but it is not automatically a sound replacement for chess legality or the opponent-completeness requirements of Definition 15.6. A useful formal limitation clarifies the compiler's responsibilities.

Proposition 15.9 (Insufficiency of a nonseparating footprint)**.** *Let $\alpha : \mathcal{S} \to Q$ be any proposed semantic footprint, and $B \subseteq \mathcal{S}$ a specified positional contract. If two states s, t satisfy $\alpha(s) = \alpha(t)$ but $s \in B$ and $t \notin B$, then no Boolean decision procedure of the form $\widehat{B} \circ \alpha$ can decide membership in B correctly on both s and t.*

Proof. The decision procedure receives the same abstract input on s and t, hence must return the same answer for both. Correct membership answers differ. This is impossible. □

The obstruction can also arise when two states agree on all present predicates but differ in available legal replies. Thus compiling a strategy from a footprint requires either verified selective-transfer obligations or a return to complete concrete state checking. A missing footprint observation is not proof that the omitted chess feature is irrelevant.

15.6 Compiler Phases, Failure Modes, and Experimental Boundaries

The proposed pipeline is deliberately split into independently testable contracts:

Position / move history + adjudication policy
↓
Canonical state → legal transition kernel
↓
Semantic facts → edge records and repair dependencies
↓
Candidate strategies / portfolios / semantic footprints
↓
Independent certificate checking → accepted claims
↓
Optional move ranking and human-facing explanation

Legal parsing, state equality, adjudication policy, predicate implementation, search exhaustion, and certificate verification have different failure modes. A failure at any trust-critical stage must cause the affected mathematical claim to be withheld, not silently replaced by a plausible natural-language explanation. In particular, a bounded strategic guarantee is valid only for its *specified* horizon and objective.

The examples already developed in this manuscript offer conformance tests without yet constituting a general chess application: the mate-in-one family of Section 11, the reply-complete three-ply construction of Section 12, the positional persistence example of Section 13, and the conflicting rook objectives of Section 14. Each must be replayed from its complete concrete input, with exactly the opponent alternatives dictated by the legal kernel.

This architecture permits a later executable semantic system to use a conventional engine as an optional candidate generator [5] or to replace that generator entirely, while retaining identical certificate-checking obligations. The central *open* compiler question is whether sufficiently compact chess-specific semantic footprints can provide useful search reduction or explanation without sacrificing required certified branches. The conditional minimax-preservation result of Theorem 9.6 does not, on its own, solve that problem.

16. Executable Realization: Refinement Stages and Verification Obligations

The preceding section gives a *compiler specification*: legal chess transitions are represented as typed semantic edges, while strategic claims must be checked independently of the procedure that proposes them. The present

section turns that specification into a staged *implementation research plan*. It does not report a completed chess engine. Its immediate purpose is to identify the smallest executable interfaces, prove what a bounded certificate-producing procedure would establish, and state the tests required before any future application may advertise a mathematical guarantee.

There are three logically separate achievements: (i) realizing the concrete chess semantics faithfully, (ii) computing and checking claims relative to an explicit semantic ontology, and (iii) discovering useful strategies or moves quickly. Success at (i) or (ii) does not entail success at (iii). This separation also permits a later conventional engine, heuristic selector, or learned model to supply candidate moves without enlarging the trusted core.

16.1 A Minimal Executable Contract and Its Trust Boundary

Fix a rules and adjudication policy ρ, a finite ontology $\mathcal{F}$, and a declared set of computable positional regions. An implementation must expose the following mathematical operations, regardless of programming language or data structures:

Operation	Required semantics and failure behavior
$\mathsf{Load}_\rho(x)$	Parse and validate a complete game input: starting state, move history, and adjudication convention; reject unsupported or inconsistent inputs.
$\mathsf{Moves}_\rho(s)$	Enumerate *exactly* $\mathcal{L}(s)$, including special moves, promotions, and check restrictions.
$\mathsf{Step}_\rho(s, m)$	Return $\delta(s, m)$ only for legal m, retaining all history-dependent information; reject illegal moves.
$\mathsf{Observe}_\mathcal{F}(s, p)$	Return $\Phi(s, p)$ and declared auxiliary predicates by their total, versioned definitions.
$\mathsf{CheckEdge}(E)$	Independently replay an edge, recompute observations, and reject mismatches as in Definition 15.4.
$\mathsf{CheckPort}(s, \mathbf{d}, C)$	Check all active obligations and *all* opponent replies according to Definition 15.6.

The trusted computational base consists of the specified rules kernel, faithful state decoding, the predicate implementations, and the certificate checkers. A candidate generator, move ordering heuristic, positional score, semantic-footprint index, user interface, or external chess engine is *outside* that boundary. A convenient third-party move library may be used initially, but calling such a library does not itself establish that its behavior agrees with the selected mathematical rules convention; that agreement remains a conformance obligation [14, 15].

Definition 16.1 (Executable refinement relation)**.** *Let* CState *be the set of encoded implementation states and $D \subseteq \mathcal{S}$ the considered arena. A relation $\mathcal{R} \subseteq D \times$ CState is a* rules refinement *if for every $(s, c) \in \mathcal{R}$:*

(i) the implementation reproduces the declared turn and terminal status of s, including the policy-dependent adjudication distinctions;

(ii) $\mathsf{Moves}_\rho(c) = \mathcal{L}(s)$ under a specified, unambiguous move-identifier correspondence;

(iii) for each $m \in \mathcal{L}(s)$, $\mathsf{Step}_\rho(c, m)$ is defined and $(\delta(s, m), \mathsf{Step}_\rho(c, m)) \in \mathcal{R}$;

(iv) illegal moves are rejected, and $\mathsf{Observe}_\mathcal{F}(c, p)$ returns exactly the declared observations of s for both players.

The relation is initially faithful *when validated inputs produce related initial states. This is a specification to be tested or formally verified, not a property established merely by choosing an implementation library.*

Proposition 16.2 (Trace conformance from one-step refinement)**.** *If $\mathcal{R}$ is an initially faithful rules refinement, then every finite legal abstract path from an admitted input has a unique corresponding implementation replay with equal legal moves, turns, terminal status, and semantic observations at every step.*

Proof. At the initial state the claim holds by initial faithfulness. Suppose it holds at the ith state. Clause (ii) identifies precisely the same legal moves, and clause (iii) provides the related successor for the selected move. Clauses (i) and (iv) preserve the claimed observations at that successor. Induction over the path length establishes the assertion; uniqueness is relative to the deterministic Step_ρ operation and the fixed move correspondence. □

This proposition is deliberately a *conditional refinement result*, not a proof that any particular software library satisfies the premises. It also does not show that the finite ontology is sufficient for evaluating chess strength.

16.2 Finite Portfolio Synthesis with Positive and Negative Certificates

The portfolio recurrence of Theorem 14.2 is not only a set identity: under the stated decidability assumptions it specifies a terminating, certificate-producing procedure on any finite search horizon. A successful branch records one protagonist choice and all opponent answers. An unsuccessful branch has the dual shape: it records all protagonist choices and one defeating opponent answer. The latter is essential when reporting that a bounded positional guarantee *cannot* be established from a specified state.

Definition 16.3 (Dual bounded portfolio evidence)**.** *For a state s, regions $\mathbf{B}$, and deadline vector $\mathbf{d}$, write $\mathsf{Win}(s, \mathbf{d}, C)$ for a positive certificate accepted by $\mathsf{CheckPort}$, and $\mathsf{Lose}(s, \mathbf{d}, D)$ for a dual refutation tree. A refutation tree contains:*

(i) an explicitly failed active predicate at a leaf, or

(ii) when $m(\mathbf{d}) > 0$ at a nonterminal protagonist node, a refutation for every *legal protagonist move, or*

(iii) when $m(\mathbf{d}) > 0$ at a nonterminal opponent node, one legal opponent move with a recursively checked refutation.

No refutation is permitted for a terminal state already satisfying all present obligations, nor when all obligations have expired and the present conditions hold. Both forms of evidence carry the complete state, policies, regions, and remaining deadlines needed for independent verification.

The following recursive specification, denoted $\mathsf{Solve}_p(s, \mathbf{d})$, chooses one of those two evidence forms:

(1) Recompute $U_{\mathbf{B}}(\mathbf{d})$. If $s \notin U_{\mathbf{B}}(\mathbf{d})$, return a losing leaf identifying an active condition violated at s.

(2) If $m(\mathbf{d}) \leq 0$ or $s \in \mathcal{T}$, return a winning leaf.

(3) Generate the complete set $\mathcal{L}(s)$ and solve every relevant child with deadlines $\partial\mathbf{d}$. If $\pi(s) = p$, return a winning certificate upon finding a winning child; otherwise assemble a losing certificate for *all* legal children.

(4) If $\pi(s) \neq p$, return a losing certificate upon finding a losing child; otherwise assemble a winning certificate containing *every* legal child.

A producer may stop early after finding a *positive* protagonist witness or a *negative* opponent witness, but it must not omit the universally quantified children in the opposite cases. A certificate checker regenerates the full legal move set rather than trusting an unverified child list.

Theorem 16.4 (Totality and exactness of bounded certificate synthesis)**.** *Assume a correct finite legal-move kernel, decidable regions* $\mathbf{B}$*, the terminal-inclusive monitoring convention of Definition 13.2, and finite deadlines. Then* $\mathsf{Solve}_p(s, \mathbf{d})$ *terminates and returns exactly one of the following forms:*

$$\mathsf{Win}(s, \mathbf{d}, C) \iff s \in \mathsf{J}_p^{\mathbf{d}}(\mathbf{B}), \tag{88}$$

$$\mathsf{Lose}(s, \mathbf{d}, D) \iff s \notin \mathsf{J}_p^{\mathbf{d}}(\mathbf{B}). \tag{89}$$

Every returned evidence tree can be independently validated by applying the corresponding existential/universal legal-successor rules.

Proof. Induct on $m(\mathbf{d})$. For $m \le 0$, the algorithm directly tests membership in $U_{\mathbf{B}}(\mathbf{d})$, exactly the base case of Equation (72). At a larger deadline, failure of the present-state test yields a valid negative leaf. For a terminal state passing that test, the absorbing convention yields a positive leaf. Otherwise the legal move set is finite and every recursive call has a strictly smaller largest deadline, so all calls terminate by induction. At protagonist nodes the algorithm reports a win exactly when some legal successor is winning; if none is, its complete collection of losing children refutes all protagonist choices. At opponent nodes it reports a win exactly when every legal successor wins; otherwise one losing child refutes the universal claim. These are precisely the clauses of Equation (72). The tree-verification conditions mirror the same induction. Hence the algorithm is total and the two outcomes are sound and complementary. □

Corollary 16.5 (Safe separation of candidate search and certification)**.** *A heuristic that proposes a promising positive certificate may be arbitrarily incomplete without compromising the soundness of any certificate subsequently accepted by the independent checker. In contrast, a report that* no *bounded winning strategy exists requires a validated negative certificate or an exhaustive, correct decision procedure; failure of a heuristic search is not such evidence.*

Proof. The positive implication is Theorem 15.7. The negative implication follows from Equation (89) and the dual evidence requirements of Definition 16.3. Absence of an output from an incomplete producer proves neither condition. □

The worst-case recursion is exponential in the horizon. For a fixed initial state and a finite explored tree with maximum branching factor $b \ge 1$ and horizon H, an unshared exhaustive expansion visits at most $\sum_{i=0}^{H} b^i$ tree nodes. This is an upper bound on that *finite unfolding*, not a practical performance claim. In particular, a correct bounded synthesizer does not automatically provide an effective full-game chess engine.

16.3 Implementation Milestones and Acceptance Gates

The engineering order should follow the proof dependencies, not the appeal of a polished interface. We propose six *gates*; passage through one gate does not certify the claims belonging to later gates.

Gate A — chess-state fidelity.
Implement the validated history-aware state, selected draw policy, legal-move enumeration, and exact transition replay. Compare a candidate kernel with an independent rules implementation on fixed fixtures and systematically generated legal games. Any mismatch blocks all downstream claims about correctness.

Gate B — semantic predicate fidelity.
Implement each selected ontology predicate as a pure, total function of a complete state and player. Retain a versioned definition, input assumptions, hand-checked fixtures, and counterexamples. Prove or test its claimed equivalence to the mathematical definition before using it as a guard or a risk measure.

Gate C — local repair certificates.
Emit and independently check semantic edge records, guarded repairs, and repair/introduction ledgers. Test exact replay and the telescoping identity Equation (86) on both fixed and generated sequences. Do not infer chess-value superiority from a reduced failure count.

Gate D — bounded adversarial certification.
Realize a small positive/negative portfolio solver for short horizons. The checker must independently enumerate every required opponent reply and reject altered or incomplete certificates. Test the exact chess examples of Sections 11 to 14.

Gate E — compact strategic abstractions.
Study whether semantic footprints and reusable macros reduce computation or support concise explanations. Every proposed transfer must discharge the selective-matching obligations of Section 10 or fall back to full-state checking; sharing a footprint alone does not certify equivalence.

Gate F — move choice and application.
Only after the preceding gates, combine candidate generation, optional engine baselines, bounded strategy analysis, and a human-readable interface. Label proposed moves, locally checked facts, bounded guarantees, and heuristic preferences differently.

The gates are *acceptance criteria*, not a report of completed milestones. At the time of writing, there are small Python prototypes for ledger accounting, guarded repair and adversarial certificate reasoning over abstract finite states, together with chess-example audit scripts and regression tests. They do not instantiate a trusted, complete chess semantic compiler or a playable chess engine.

16.4 Regression Corpus, Mutation Tests, and Reproducibility

A valid program demonstration must survive cases chosen to falsify its own advertised guarantees. A minimal corpus must cover two different kinds of evidence:

(i) *Concrete legality and history*: castling rights and castling through check, en-passant legality, all four promotion choices, stalemate, checkmate, the distinction between threefold claim eligibility and automatic repetition adjudication, and histories with identical boards but different permitted claims.

(ii) *Semantic and strategic obligations*: a purported repair that disappears because its target piece is lost, opponent-branch omission, invalid predicate values, wrong horizon or active deadline, incorrectly transported footprints, competing rook objectives, and the additional underpromotion counter-reply in the three-ply example of Section 12.

Mutation tests should deliberately change a legal move into an illegal one, flip a predicate bit, delete an opponent child, change a terminal marker, or alter a declaration of the draw policy. A checker that still accepts any such change without independent revalidation cannot support the proposed claim type. Differential testing against a second chess kernel can expose errors but cannot prove the absence of shared defects; theorem-level soundness remains conditional on the trusted semantics.

Definition 16.6 (Reproducible semantic claim manifest)**.** *A* claim manifest *contains the full or faithfully recoverable initial history-aware state, rules/adjudication policy, ontology revision, positional predicates and horizons, the proposed certificate, its verifier version, and the verifier's decision. Any reported search statistics or heuristic score are separate, explicitly marked experimental fields. A bare position diagram or numerical engine score is not a substitute for the manifest.*

For an experimental run, the manifest should also identify the hardware, runtime, limits on examined nodes and elapsed time, and candidate-generator version. These quantities are needed to evaluate future hypotheses in Section 17; they are not part of the mathematical truth conditions of a successfully checked certificate.

16.5 Initial Chess Testbed and the Boundary to the Application

The existing chess fixtures provide a progressive testbed. The *mate-in-one family* asks whether the semantic feature set records a relevant blocked attack line. The *three-ply king-and-queen family* tests adversarial branch completeness and an unexpected underpromotion counter-repair. The *passed-pawn and rook example* tests retaining a positional condition over a short horizon. The *competing rook repairs* ask whether one strategy can jointly satisfy two different positional contracts. These examples isolate distinct failure modes; none demonstrates that the current ontology predicts high-quality chess play in general.

The opening comparison 1.$e4$ $e5$ 2.Nf3 Nc6 versus 1.$e4$ $e5$ 2.Nf3 $f6$ is a useful initial pedagogical example of local defenses with different side effects. It is not sufficient to calibrate a general evaluation function. Future playing-strength experiments must compare equal computational budgets and must not confuse a valid explanation of a move's *declared* effects with a proof that it is game-theoretically optimal.

We therefore retain the research priority established earlier in the manuscript: finish the mathematical and proof audit before constructing the full application. When executable development begins, the first useful artifact should be a small *checker-first chess analysis demonstrator* that displays state, legal options, exact predicate changes, and the status of bounded certificates. A playable engine and a polished demonstration may follow, but are not prerequisites for the paper's present conditional results.

17. Experimental Evaluation and Research Hypotheses

This section specifies a *prospective, preregistrable evaluation protocol* for the mathematical and executable system described above. The protocol identifies the measurements and comparisons needed to evaluate its explanatory, computational, and chess-playing contributions. It distinguishes *validity of a contract*, *coverage of strategic situations*, *computational cost*, and *quality of chess play*.

17.1 Evaluation Objects, Claims, and Evidence Levels

A registered evaluation instance is a tuple

$$e = (s, p, \mathcal{O}, A, \mathbf{B}, \mathbf{h}, \mathcal{R}, b, \nu), \tag{90}$$

where $s \in \mathcal{S}$ is a history-aware legal chess state, $p \in \{\mathrm{W}, \mathrm{B}\}$ is the designated player, $\mathcal{O}$ is a versioned failure ontology, $A \subseteq \mathcal{S}$ is the safety region, $\mathbf{B} = (B_1, \ldots, B_k)$ and $\mathbf{h} = (h_1, \ldots, h_k)$ are declared positional objectives and horizons, $\mathcal{R}$ is any proposed transfer relation, b is a computational budget, and ν records software and experimental versions. The tuple may specify an empty portfolio when a test is concerned only with move selection. The evaluator must preserve all state fields relevant to castling, en passant, repetition, and terminal adjudication.

The system may emit four distinct forms of evidence:

1. a *legal-move claim*, checked directly against $\mathcal{L}(s)$;
2. a *local semantic claim*, checked from exact declared predicates and the faithful transition δ;
3. a *bounded strategic certificate* or a bounded adversarial refutation, independently checked against all required branches;

4. a *heuristic prediction* about move quality, playing strength, or strategic usefulness, which carries no formal guarantee unless accompanied by one of the foregoing certificates.

The fourth category must not be described as proved merely because a separately certified local predicate appears in its explanation. In particular, passing the checker of Section 15 does not imply that the move is minimax-optimal or even useful outside its declared contract.

Definition 17.1 (Certificate acceptance and empirical coverage)**.** *Let D be a finite, versioned collection of proposed certificates with independently adjudicated contract truth values. Write $\chi(c) \in \{0,1\}$ for the checker decision and $T(c) \in \{0,1\}$ for the truth of the claimed contract under the registered rules kernel. Define*

$$\mathrm{Acc}(D) = \frac{\sum_{c\in D} \chi(c)}{|D|}, \qquad \mathrm{FAR}(D) = \frac{\sum_{c\in D} \chi(c)(1-T(c))}{\sum_{c\in D} \chi(c)}, \tag{91}$$

where FAR *is undefined if no certificate is accepted. Acceptance rate measures* coverage of submitted claims*, not correctness. Report the number of accepted claims and the independently adjudicated sample size alongside any error-rate estimate.*

Proposition 17.2 (Generator-independent safety of accepted claims)**.** *Assume the checker is sound for the declared chess rules and ontology: $\chi(c) = 1 \Rightarrow T(c) = 1$ for every well-formed submitted claim c. Then any candidate generator, including a heuristic engine, language model, or transferred strategic macro, has zero false acceptances on every finite collection of claims checked without bypassing this verifier.*

Proof. For each accepted claim soundness gives $T(c) = 1$; hence every term $\chi(c)(1-T(c))$ in the numerator of Equation (91) vanishes. The conclusion does not bound the rejection rate or the cost of verification. □

Remark 17.3 (The trust boundary)**.** The proposition is conditional on *actual* checker soundness and correct contract interpretation, not a measured theorem about our future software. Independent test oracles, mutation tests, and differential chess-rules audits are required to detect implementation errors. A zero observed false-acceptance rate on a small sample is not proof of checker soundness.

17.2 Benchmark Families and Negative Controls

The proposed public and held-out evaluation corpus will be versioned and stratified by *the problem being tested*, not merely by chess opening:

1. **Legality and history:** castling, en passant, underpromotion, check evasions, repetition-sensitive states, move counters, and disputed or malformed state representations.
2. **Local repairs and conflicts:** defended pieces, exposed material, king-cover changes, passed-pawn support, competing rook objectives, and situations where a repaired threat is replaced by a new one.
3. **Bounded adversarial plans:** mating trees, required opponent replies, refuting replies, and examples where incomplete branch exploration incorrectly suggests a guarantee.
4. **Footprint transfer:** paired positions with a valid transferred contract; near-neighbor pairs with the same coarse Φ but a different legal counter-reply; and deliberately corrupted footprint assumptions.
5. **Positional prediction and play:** tactical, defensive, maneuvering, endgame, and unbalanced positions from sources held separate from ontology selection, weight fitting, or macro discovery.

Each family contains both positive and negative instances. The examples of Sections 11 and 12 and the positional counter-repairs in the subsequent sections serve as *development regressions*; they must not be

counted as an untouched validation set. Where feasible, add adversarially generated mutations of each example, including newly legal defensive replies and altered history fields.

Use at least three explicit partitions: development, validation, and locked test. Freeze the exact test positions and acquisition rules before the final comparative experiment; screen for duplicate or transposed near-duplicates across partitions. For certified correctness, a trusted rule implementation or an independent exhaustive enumerator is the reference; for small piece-count endgames, exact tablebases may additionally supply objective win/draw/loss labels *only within their stated coverage and rule conventions*. Deep search output is an *approximate reference for move quality*, never an oracle for the truth of an arbitrary strategic contract.

17.3 Falsifiable Hypotheses and Matching Baselines

The following hypotheses replace a generic assertion that the semantic system will be "better" than conventional chess engines. An observed improvement in one dimension must not be substituted for another:

H1: Sound explanation.
A checker-gated system will have lower adjudicated *unsupported-claim rate* than an otherwise identical ungated explanation generator, while possibly accepting fewer claims. Report acceptance, false acceptance, abstention, and checker latency.

H2: Transfer utility.
A footprint-guided macro system will certify valid multi-position strategy transfers more often per unit time than a no-transfer certificate-search baseline, without increasing accepted false claims. Record transfer attempts, certificates, refutations, and search/certificate work.

H3: Search efficiency.
At a fixed move-quality threshold, an ontology-informed move-ordering system will reduce wall-clock time or total computation relative to an *identical* search with the semantic features disabled. Compare quality–cost curves, not only nodes searched.

H4: Defensive robustness.
For a specified, preregistered suite of defensive and multiobjective positions, a repair-aware system will attain more certified positional contracts or higher independent move-quality scores than the corresponding ablated system at the same resource budget.

H5: Playing strength.
A complete UCI-compatible semantic engine or hybrid will achieve a measured match score against a pinned conventional baseline under paired openings, both colors, and equal stated resource constraints. *No direction of the result is assumed.*

H6: Human interpretability.
Under blinded annotation, readers will identify the correct repaired predicate, relevant counter-reply, and scope of the guarantee more accurately with certified semantic explanations than with matched control explanations. Measure comprehension and response time independently of perceived persuasiveness.

H1–H4 and H6 can be meaningfully studied without constructing an engine stronger than existing programs; H5 requires an actual playable engine. Generalization to non-chess adversarial domains remains a separate research program, not evidence for H1–H6.

17.4 Engines, Ablations, and Fair Comparison

A first comparative study will freeze the binary, network weights, build flags, settings, and version of Stockfish [5, 16] and optionally a second established engine. The *within-family* ablations are especially important because they isolate proposed semantic contributions:

Variant	Purpose / controlled difference
C_0	Rule-correct reference search; semantic layer disabled.
C_1	Same search plus semantic *move ordering* only; no selective pruning, so move quality can be compared without excluding branches.
C_2	Same search plus bounded contracts, checker, and footprint reuse; measure verification overhead and certificate yield.
C_3	The proposed complete semantic or hybrid engine, if implemented; compare actual move quality and match performance.
E_0	A pinned external engine, e.g. Stockfish, with an explicitly recorded UCI configuration and equalized match conditions.

If C_0–C_2 differ in learned evaluation, search algorithm, or hardware, the experiment no longer isolates the semantic component. An optional Leela Chess Zero comparison needs its own hardware and inference budget accounting; CPU-node counts are not comparable to neural accelerator work. A safe heuristic move ordering is not equivalent to dropping branches. Any pruning experiment must either prove the relevant successor-retention obligation from earlier sections or label the pruning as *empirical and potentially unsound*.

A UCI match runner (for example `cutechess-cli` [17]) can use a locked opening set, reverse the colors for each opening, and retain every game in PGN. Before collecting test results, register the wall-clock controls, processor/core allocation, threads, transposition-table size, pondering, tablebases, opening sampling policy, time and increment settings, deterministic seeds where supported, and adjudication rules. Run both *equal wall-clock* comparisons (primary practical metric) and separately labeled *fixed-node* comparisons (diagnostic; different node costs make these noninterchangeable). Record actual CPU time, node counts, memory, and certificate-checking work.

17.5 Metrics, Uncertainty, and Interpretation

For a fixed collection S of analysis positions, let $\widehat{Q}(s,m)$ be a frozen *reference-engine estimate*, evaluated from a consistent player perspective, and let $m^*(s)$ be the reference's selected move. A provisional nonnegative surrogate regret is

$$\widehat{r}(s,m) = \max\bigl\{0, \widehat{Q}(s,m^*(s)) - \widehat{Q}(s,m)\bigr\}. \tag{92}$$

Evaluate candidate moves and the reference-selected move using the same independently fixed analysis procedure and budget. This is *reference regret*, not true minimax regret. Mate scores must be handled as a distinct outcome category, and centipawn, mate-distance, and win/draw/loss estimates must not be silently combined into a single linear numerical scale. When UCI information is used, normalize viewpoint explicitly; available score and WDL interfaces are documented in `python-chess` [15] and the Stockfish documentation [16].

For a completed match containing W wins, D draws, and L losses from one engine's perspective, report the observed score

$$\widehat{s} = \frac{W + \frac{1}{2}D}{W + D + L}, \qquad W + D + L > 0. \tag{93}$$

Record the number of *paired openings*, not only games. Estimate uncertainty by resampling opening *pairs* or by another clearly justified clustered procedure, and publish the exact game scores; results from paired games are not independent Bernoulli observations. For larger engine-strength studies, a preregistered sequential testing protocol may be appropriate, but stopping rules and effect sizes must be specified *before* viewing match outcomes (cf. Stockfish's test practices [18]).

For each experimental variant, report a vector rather than a single ambiguous success score:

$$\mathsf{Perf} = (\text{verified validity},\ \text{coverage},\ \text{transfer yield},\ \text{cost},\ \text{reference regret},\ \text{match score}). \tag{94}$$

The terms are not generally commensurable. Pareto comparisons and quality–cost curves are preferable to an arbitrary weighted sum unless weights are separately justified. Publish exact numerator/denominator counts and confidence intervals; avoid labeling a non-significant result as equivalence without an equivalence margin and adequate power.

17.6 Pre-Registration, Stopping Gates, and Reproducibility

Before running the final comparisons, record the primary outcome and smallest practically relevant effect for each hypothesis, the planned sample size or power criterion, inclusion and exclusion rules, treatment of forfeits/timeouts, analysis horizon, multiple-testing strategy, and decision threshold. Small smoke tests can test the harness but are *not* evidence of engine strength. Human explanation studies require independent annotations, blinded labels, agreement measures, and a matched-text control; a fluent explanation must not be scored as true merely because it sounds convincing.

A reproducibility bundle should contain the source revision, compiled binary hashes, ontology identifier, predicate definitions, legal-rules version, certificate format, all random seeds, benchmark FENs together with required histories or histories encoded in PGN, engines and configuration files, opening book, hardware specification, raw search traces, all produced certificates and refutations, independent verifier logs, match PGNs, and analysis scripts. *History-sensitive positions must not be treated as if bare FEN alone determines repetition rights.*

The experimental sequence has explicit readiness gates:

1. **G0:** rules-kernel conformance, semantic-predicate tests, and hostile certificate mutation tests pass;
2. **G1:** checker correctness is independently audited on finite adversarial trees with both positive and negative cases;
3. **G2:** hidden transfer and repair benchmarks report validity, coverage, and computational overhead;
4. **G3:** fixed-budget move-quality ablations isolate the value of semantic ordering, caching, and contract reuse;
5. **G4:** a complete, legal, clock-aware engine exists before any head-to-head engine match or public claim of chess strength;
6. **G5:** only then does the explanatory chess application undergo usability testing with chess players.

Failure at any gate requires revising the implementation or narrowing the claim, rather than reinterpreting a failed test as support. A negative result on H3 or H5 need not invalidate a positive result on independently verified certificates under H1 or H2. The value of the proposed research may lie in *auditable strategic reasoning* even if an eventual engine proves weaker or slower than the strongest conventional baseline.

18. A Human-Executable Chess Reasoning System

The semantic program has so far treated diagnosis, guarded repair, and adversarial response as mathematical objects over complete chess states. There is a separate, deliberately weaker question: can a person use a *small* vocabulary of failure classes to reason more reliably at the board? A human-facing procedure must make strategically useful observations without misrepresenting an unexamined reply as a theorem. This section specifies such a procedure and gives formal limits on what its compressed observations can establish. It is a proposed cognitive interface, not a claim that the procedures below have already improved human play.

The distinction between recognition and deliberate search is consistent with classical investigations of chess expertise [19, 20, 21]. Those studies motivate an empirical comparison; they do not establish that our particular failure ontology is psychologically natural or effective.

18.1 Compact Observations and the Information They Omit

Fix an admissible collection $D \subseteq \mathcal{S}$ of legal, history-aware chess states, a player p, and a finite family of human-observable queries $Q = (q_1, \dots, q_r)$. Each q_i is a specified function from D to a finite alphabet. A query may ask whether a particular piece is under attack, whether an identified pawn is passed, or which player is to move. Its value must be defined objectively before the query can appear in a certificate. Interpretive judgments such as "Black is dangerous" are permitted as *heuristic prompts*, but not as exact predicate witnesses.

Definition 18.1 (Human semantic view and compatibility class)**.** *The compact observation of $s \in D$ under Q and its compatible states are, respectively,*

$$V_Q(s) = (q_1(s), \dots, q_r(s)), \qquad [s]_{Q,D} = \{t \in D : V_Q(t) = V_Q(s)\}. \tag{95}$$

The compatibility class represents all positions that the chosen observations fail to distinguish*; it does not assert that the player literally cannot see the remaining pieces. For transfer of fixed chess instructions, D and Q must also fix whose turn it is and the intended move notation. Full legality continues to depend on the complete state.*

The view is not assumed to be transition-stable in the sense of Sections 6 and 8. In particular, identical query answers do not imply identical legal moves, equal opponent choices, or equal retention depth. The distinction reproduces, at the human interface, the proof obligations developed for strategic footprints in Sections 10 to 12.

Definition 18.2 (Contract-admissible move and view-robust advice)**.** *For a declared move-level contract $\kappa(s, m)$, define*

$$\mathcal{A}_\kappa(s) = \{m \in \mathcal{L}(s) : \kappa(s, m) \text{ holds}\}. \tag{96}$$

The set of uniformly justified *moves for a view v is*

$$\mathrm{Advice}_\kappa(v; D) = \bigcap_{\substack{t \in D \\ V_Q(t) = v}} \mathcal{A}_\kappa(t). \tag{97}$$

Only nonempty view classes are used. A fixed move in this intersection is legal and satisfies the declared contract in every *position compatible with the retained observations; a move outside it may still be excellent in an individual, fully inspected position.*

Theorem 18.3 (Observation refinement and sound uniform advice)**.** *Let Q' refine Q on D in the sense that equality of $V_{Q'}$ implies equality of V_Q. For every $s \in D$,*

$$\mathrm{Advice}_\kappa(V_Q(s); D) \subseteq \mathrm{Advice}_\kappa(V_{Q'}(s); D). \tag{98}$$

Furthermore, a rule that selects the same *move solely from the Q-view and guarantees κ throughout its compatibility class exists precisely when the left-hand advice set is nonempty.*

Proof. Every state with the refined view has the original view, so $[s]_{Q',D} \subseteq [s]_{Q,D}$. Intersecting the same move-admissibility sets over fewer states can only enlarge the intersection, which proves Equation (98). If a uniform rule chooses a move m, its guarantee at every compatible state requires $m \in \mathcal{A}_\kappa(t)$ for every such t; hence m belongs to Eq. (97). Conversely, any member of this intersection is a uniform choice satisfying the contract throughout that view class. □

Corollary 18.4 (The abstraction-collision barrier)**.** *If $s, t \in D$ have the same human view, but a fixed move m satisfies $\kappa(s, m)$ and fails $\kappa(t, m)$ (including the case $m \notin \mathcal{L}(t)$), then no rule depending on that view alone can correctly certify m as uniformly contract-admissible for both positions.*

Proof. The move is absent from $\mathcal{A}_\kappa(t)$ and therefore from the intersection defining uniformly justified advice. □

This theorem is elementary set-theoretic mathematics, not a claim of a new psychological mechanism. It isolates an important design constraint: selective observation can help prioritize analysis, but deleting a relevant counter-reply can invalidate a supposedly transferable human rule.

18.2 A Chess Witness: A Familiar Pattern and an Unseen Counter-Repair

We reuse the two checked chess positions of Example 13.8; they are especially instructive because the proposed positional move is *legal and locally repairing in both*. With White to move, let

$$s_0 = \texttt{6k1/7p/8/8/P7/8/4K3/7R w - - 0 1}, \tag{99}$$

$$s_1 = \texttt{6k1/7p/8/8/P7/1p6/4K3/7R w - - 0 1}. \tag{100}$$

Both contain White's king `e2`, rook `h1`, and passed `a4`-pawn, and Black's king `g8` and `h7`-pawn. Only s_1 contains Black's additional pawn on `b3`. Define a compact human view that records the player to move, whether the `a4`-pawn is passed, the king and rook locations just listed, and whether the rank-one rook path `h1–a1` is unobstructed; it does *not* record the `b3`-square. Both states receive the same view.

Let B be the explicitly defined passed-pawn support condition in Example 13.8: rook on `a1` behind White's passed `a4`-pawn, rook unattacked, and White not in check. For the *fixed* proposed move m = Ra1 define

$$\kappa_B(s, m) \iff m \in \mathcal{L}(s), \quad \delta(s, m) \in B \cap \mathsf{K}^1_{\mathrm{W}}(B). \tag{101}$$

As established in Proposition 13.9, $\kappa_B(s_0, m)$ holds: all six legal Black replies preserve B. In s_1, however, `1...b2` is a seventh legal reply; from `b2`, Black's pawn attacks `a1`, so $\kappa_B(s_1, m)$ is false.

Proposition 18.5 (A chess-specific human-footprint obstruction)**.** *The proposed coarse view cannot justify the instruction "play `Ra1` to preserve B against all immediate replies" uniformly across a family containing both s_0 and s_1. Adding the query "Is a Black pawn on `b3`?" separates these* two examples*, but by itself does not prove that the refined view suffices over all legal chess positions.*

Proof. The states agree on every query in the coarse view. The checked legal moves and their reply sets give $\kappa_B(s_0, m)$ and $\neg\kappa_B(s_1, m)$ as above. The first claim follows from Corollary 18.4. The added occupancy query distinguishes the two specified states because it has opposite Boolean values there. No claim about additional states follows from this two-state separation. □

This example identifies the difference between a *useful cue* ("put the rook behind the passed pawn") and a *certified instruction* ("that placement survives every legal opponent reply"). The cue can remain helpful even when its stronger contractual reading is false.

18.3 Four-Pass Over-the-Board Reasoning Protocol

The proposed short procedure is *Scan–Repair–Reply–Recheck*. It is not an executable substitute for the chess rules kernel. Its purpose is to allocate limited human attention while preserving the distinction between observed facts, candidate moves, and proved guarantees.

1. **Scan the position.** Determine the side to move and immediate legal constraints. Ask: "Am I in check? Are any pieces hanging? Does either player have a forcing check, capture, or promotion?" Record only predicates actually inspected. Checks must be answered by legal moves; a suggested repair that leaves one's king in check is discarded.
2. **Repair with an explicit objective.** Name one or two vulnerabilities and a proposed target condition B. Consider a small candidate set $M_H \subseteq \mathcal{L}(s)$ if legality can be confirmed. Annotate each move with the failures it may remove, secondary failures it may introduce, and any positional guard it is meant to preserve. A proposed repair is not a claim of optimality.
3. **Reply adversarially.** For each leading candidate ask: "What is the opponent's strongest forcing response, and what response would make my stated repair fail?" Seek a concrete refuting legal move, using the ordered strategy and counter-repair concepts of Sections 6 and 14. Examining only a few replies may help reject candidates, but *cannot* certify a universal condition unless all required branches have been checked.
4. **Recheck and state the evidence level.** Verify legality, inspect newly exposed weaknesses, and distinguish *observed local repair*, *refuted specific plan*, *independently certified bounded guarantee*, and *unresolved conjecture*. If the time budget expires without a complete check, the human interface must be willing to say "not established."

Remark 18.6 (Time and attention as strategic budgets)**.** A player may choose, for example, two candidate moves and one initial counter-reply per move under a short clock. Such a policy is a *search heuristic* and may miss decisive resources. The human protocol makes no claim of complete tactical coverage or superiority to ordinary chess calculation. Human response time, cognitive effort, and chess strength are empirical quantities to be measured, not assumed.

18.4 Compact Explanations with Explicit Evidence Levels

A mathematical explanation need not display an entire adversarial search tree to a human. It can instead provide a concise *claim card* with an explicit link to a full certificate when one exists.

Definition 18.7 (Human-facing claim card)**.** *A human-facing claim card is*

$$e = (\iota, s, \kappa, m, h, \mathsf{status}, w, \mathsf{link}), \tag{102}$$

where ι identifies the claim and ontology versions, s is the complete source-state identifier, κ is the exact contract, m is the proposed legal move or contingent macro, h is its horizon, the status field has one of the three values

CERTIFIED, REFUTED, UNRESOLVED,

w is a readable witness or counterexample summary, and link *identifies the corresponding full certificate or refutation when present. The displayed summary w is a* projection *of the underlying argument, not a replacement for its complete branches.*

Proposition 18.8 (Sound display of checked human-facing claims)**.** *Assume the independent verifier of Section 15 is sound under the registered chess rules and predicate interpretation. Suppose the claim-card renderer marks an assertion* CERTIFIED *only if it verifies a certificate against the exact state s, contract κ, horizon h, and version ι, without allowing the natural-language summary to alter any of those fields. Then every displayed* CERTIFIED *assertion satisfies its declared bounded contract, irrespective of the candidate generator or the brevity of w.*

Proof. The renderer's promotion rule requires acceptance by the sound verifier for the same state, contract, horizon, and version. By checker soundness, that accepted assertion satisfies the contract. The readable summary cannot change the checked claim by hypothesis. This is an application of the generator-independence principle of Proposition 17.2, not a claim that unverified prose is itself a mathematical proof. □

A legitimate REFUTED card for a *fixed move or fixed contingent strategy* must include a legal, verified counter-continuation that violates the declared contract. Such a counterexample does not necessarily prove that *no alternative strategy* could succeed. That stronger negative assertion requires the dual adversarial certificate of Sections 6 and 16. If neither kind of certificate is available, the accurate status is UNRESOLVED.

Example 18.9 (Readable counter-repair card)**.** For s_1 in Equation (100), a compact card can truthfully report:

Field	Human-readable statement
Candidate	`Ra1`: rook behind the passed `a4`-pawn.
Local result	The move reaches the declared support condition B.
Opponent reply	`...b2` attacks the rook on `a1`.
Scope	One-reply persistence of B after `Ra1`.
Status	**REFUTED** for this fixed move and this contract.
Verified claim	This particular `Ra1` retention contract is refuted.

The accompanying full-state witness is the mathematical evidence underlying the card; an executable interface can expose that witness for independent verification.

18.5 Learning, Validation, and Semantic Compression

The proposed teaching method is to start with concrete chess states and have a reader predict (i) a failure predicate, (ii) a candidate repairing move, (iii) an adversarial refutation if one exists, and (iv) the precise scope of any remaining claim. The same tasks can be presented with unannotated chess explanations, purely numerical engine evaluations, or certified semantic cards as controls. Section 17 gives the prospective H6 evaluation framework; the human section adds three specific outcomes: accurate separation of *local* from *persistent* repair, accurate reporting of the *quantifier over replies*, and appropriate *abstention* when available observations do not suffice.

A graded series of exercises would begin with the one-move tactical footprints of Section 11, proceed through the multi-ply counter-repair examples of Section 12, and then use the competing-rook objectives in Section 14. Results should be stratified by chess experience and measured with held-out examples. No improvement in recall, time management, accuracy, or rating has yet been measured. These observations must also respect the rules of the particular play setting: assistance from a device or an external engine is not permitted during many rated over-the-board games, whereas a memorized human reasoning method can be practiced independently.

The mathematical upshot is modest but precise: semantic compression can organize a player's attention, while the observational equivalence barrier explains why compression alone cannot supply universal strategic guarantees. The paper's full-state mathematics supplies the proof boundary; the human interface supplies candidate questions and a vocabulary for interpreting verified results. Whether this separation is useful in practice is a falsifiable part of the research program.

19. Generalization Beyond Chess

The chess construction is a testbed rather than an assertion that every adversarial domain behaves like chess. Its exact legal moves, public state, deterministic transitions, and alternating turns allow us to separate the mathematics of *failure and repair* from the difficulties of observing and controlling a system. We now identify what survives when the board is replaced by another semantic state space, and state explicitly what does not. The underlying repair-algebra motivation comes from the AAD Systems work on state, viability, semantic observation, damage, and composable repair [22, 23]. The adversarial-transfer results also draw on established program-refinement and alternating-simulation ideas [11, 10]; the purpose here is to expose their precise role in our repair contracts, not to claim priority for those tools.

19.1 Domain-Independent Repair Arenas

Definition 19.1 (Turn-based adversarial repair arena)**.** *A* turn-based adversarial repair arena *is a tuple*

$$\mathcal{A} = (S, S_p, S_e, E, T, \mathcal{F}, \mathcal{O}), \tag{103}$$

where $S = S_p \dot{\cup} S_e$ is a set of complete model states, $E \subseteq S \times S$ is a legal transition relation, $T \subseteq S$ consists of terminal states, $\mathcal{F}$ is a chosen family of state predicates, and $\mathcal{O}$ is a family of semantic observations. At states in S_p the protagonist chooses a successor; at states in S_e an environment or adversary chooses a successor. Nonterminal states have at least one successor; terminal states have no ordinary successor. Any termination convention used for monitoring is stated separately. The relation is effectively finitely branching *when all successor sets can be computed, are finite, and membership in relevant predicates is decidable.*

In this abstract arena a failure class $f \in \mathcal{F}$ is still a predicate on states, a *local repair* is a legal transition from $f(s) = 1$ to $f(t) = 0$, and a guarded repair additionally preserves a specified invariant. The repair/introduction ledger of Section 4 remains an additive account of a finite path:

$$\Lambda(\gamma \cdot \eta) = \Lambda(\gamma) + \Lambda(\eta), \tag{104}$$

whenever the two legal paths concatenate. This identity does not depend on chess pieces or moves. It also retains the same limitation: two paths with equal net effects may have different intermediate safety outcomes.

A *repair contract* specifies an entry region P, a safety region A, a goal region $B \subseteq A$, and a horizon h. It is satisfied if one protagonist policy ensures that every resulting path stays in A until it first enters B within h transitions. A terminal state outside B fails this reachability contract. This is the same existential-versus-universal distinction used in Sections 6 and 7; only the underlying transition system has changed. Multiple

positional conditions can instead be expressed as an intersection $B = \bigcap_i B_i$ or as the different horizon obligations of Section 14.

Proposition 19.2 (Transport of pathwise repair bookkeeping)**.** *Let every predicate in a finite $\mathcal{F}$ be decidable along a legal finite path in a repair arena. The ordered sequence of its failure-removal and failure-introduction vectors determines an additive repair ledger satisfying Equation (104). Neither this identity nor its computability requires an adversarial winning strategy.*

Proof. For every adjacent pair of states, evaluate all predicates and record componentwise removals and introductions. The ledger of a path is the sum of these edge records. Concatenation partitions that finite sum into two consecutive sums; associativity of vector addition gives the identity. Termination of the predicate procedures and finiteness of the path give computability. No choices by a second player occur in the bookkeeping argument. □

19.2 Alternating Refinement and the Transfer of Repair Guarantees

A different domain can instantiate the repair arena and still admit the same proof principles. More importantly, an *abstract* arena may be used to certify a strategy for a *concrete* arena only if all relevant concrete adverse responses are accounted for. An abstraction that merely agrees on its failure signature does not suffice, as our chess counterexamples in Sections 11 and 12 illustrate.

Definition 19.3 (Policy-relative adversarial refinement)**.** *Let $\mathcal{A}_0$ and $\mathcal{A}_1$ be turn-based repair arenas, with their respective regions (P_i, A_i, B_i) and the same protagonist role. Let σ_0 be a history-dependent policy in $\mathcal{A}_0$. A relation $R \subseteq S_0 \times S_1$ is a σ_0-*relative adversarial refinement *on the policy tree if:*

(i) for every $t \in P_1$ there is $s \in P_0$ with sRt;

(ii) every related pair reached in the construction has the same controller (protagonist or environment), the same terminal status, and agrees on membership in the two contract predicates: $s \in A_0 \iff t \in A_1$ and $s \in B_0 \iff t \in B_1$;

(iii) at a related protagonist history, the successor selected by σ_0 has some *legal matching successor at the concrete state, again related by R;*

(iv) at a related environment history, every *legal successor of the concrete state has a related legal successor of the abstract state.*

The relation is checked recursively only for histories reachable while following σ_0 and matching concrete responses, up to the specified horizon. Concrete moves need not share their names with abstract moves. The lifted policy may remember the abstract history used to witness the relation.

Theorem 19.4 (Sound transport of bounded repair contracts)**.** *Suppose a policy σ_0 certifies (P_0, A_0, B_0, h) in $\mathcal{A}_0$, and the hypotheses of Definition 19.3 hold through depth h. Then there is a history-dependent policy σ_1 in $\mathcal{A}_1$ that certifies (P_1, A_1, B_1, h).*

Proof. For each concrete starting state in P_1, choose a related abstract start in P_0 and store it as simulation memory. Proceed by induction on the number of transitions taken. At a protagonist node, use clause (iii) to choose a concrete successor related to the successor prescribed by σ_0, and extend both histories. At an environment node, wait for the concrete response and use clause (iv) to extend the stored abstract history with a matching response. In either case the next pair is related, so the induction continues. Clause (ii) preserves safety and goal membership at every matched step, as well as the termination status; the concrete

play therefore cannot encounter an unmatched terminal failure. Every resulting abstract history follows σ_0 and reaches B_0 in at most h steps while remaining in A_0 beforehand. The concrete history reaches B_1 on the corresponding step while remaining in A_1. The constructed policy thus supplies the required universal guarantee. □

Corollary 19.5 (Multiobjective transfer)**.** *The conclusion of Theorem 19.4 also preserves a bounded multiobjective repair portfolio whenever the refinement preserves membership in each required region at every corresponding phase, and the same lifted policy satisfies all of the phase-indexed obligations.*

Proof. Apply the path-matching construction in the theorem to the single lifted policy. At each monitored phase all required regional memberships match. Consequently every component of the portfolio contract holds along every resulting concrete play. □

Example 19.6 (Why an unmodeled adversarial response matters)**.** An abstract protagonist choice leads to an environment state with one legal successor, labeled *safe goal*. Its concrete counterpart has that successor *and* a second successor labeled *failure*. The chosen protagonist action and all initial failure predicates may agree. Nevertheless the abstract policy wins and its concrete counterpart does not. Clause (iv) of Definition 19.3 correctly rejects this attempted transfer. The example is an abstract transition system, not an asserted real-world security or intelligence scenario.

The refinement theorem is intentionally conditional. It does not establish that a useful abstraction exists, that the relation R can be computed efficiently, or that the specified goals are good strategic objectives. Those are separate obligations for each instantiation.

19.3 Information Structure and Strategic Guarantees

Chess players observe the complete board and may base a strategy on the complete legal history. In intelligence analysis, cyber-defense monitoring, and information-flow systems the decision-maker may see only a projection of the underlying state. Distinct states can therefore require different repairs without being distinguishable to the agent. Formal games of incomplete information and partially observable planning supply well-developed starting points for this change of semantics [24, 25].

Definition 19.7 (Observation-uniform repair policy)**.** *Let $o_p : S \to O_p$ be the protagonist's observation map. For a history $\gamma = s_0 \cdots s_k$, let $\mathrm{obs}_p(\gamma)$ be the visible observation history, including the protagonist's own choices. A policy σ is* observation-uniform *when two legal protagonist histories with identical observation histories receive the same action, and that action is legal at both histories. A contract for an initial information set $I \subseteq S$ is* uniformly certified *if one such policy satisfies the contract from every $s \in I$.*

Proposition 19.8 (Perfect-information certification does not imply uniform certification)**.** *There is a finite deterministic repair arena with an initial information set I such that each $s \in I$ individually admits a one-step repair policy, while no observation-uniform policy repairs all states in I within one step.*

Proof. Take two protagonist states u and v with the same observation and with available actions a, b. In u action a reaches a goal and b reaches a failure; in v action b reaches a goal and a reaches a failure. Thus a perfect-information player can repair both u and v, selecting a different action in each state. An observation-uniform one-step policy must select the same action for u and v, so it fails in one of them. The arena is finite and all transitions are deterministic, establishing the claim. □

Accordingly, Theorem 19.4 yields a policy for the concrete *full-state* arena. It does not by itself yield an implementable policy for a partially observing agent. Such a claim requires an additional proof that the lifted decisions are uniform over all histories sharing observations (or an appropriate belief-state construction), and that the observation and action interfaces are faithfully implemented. The need for these extra premises is a mathematical boundary, not a minor engineering detail.

19.4 Probabilistic Transitions and Composable Risk Budgets

Another departure from chess is uncertainty about what follows an action. A stochastic repair model can equip each state and action with a transition probability kernel. The adversarial guarantee "all legal replies" must then be distinguished from a probability threshold or a worst-case guarantee over the kernel's support. In particular, a high-probability repair is *not* a certified universal repair.

Theorem 19.9 (Sequential risk-budget bound)**.** *Consider h stages of an adaptive repair policy in a stochastic system, and let F_i denote a contract violation at stage i. Suppose, for each i, and conditional on* any positive-probability *history in which no earlier violation has occurred, the probability of F_i at the next stage is at most $\varepsilon_i \in [0, 1]$. Then*

$$\Pr\left(\bigcup_{i=1}^{h} F_i\right) \leq \min\left\{1, \sum_{i=1}^{h} \varepsilon_i\right\}. \tag{105}$$

This holds without independence assumptions and permits later actions to depend on earlier observations.

Proof. Partition failure into the disjoint events that the *first* violation occurs at stage i. By conditioning on the preceding violation-free history and applying the stated bound, the probability of first violation at i is at most ε_i. Sum over all stages, and use the trivial upper bound of one. No step assumes that failures at different stages are independent. □

The union bound provides a conservative, contract-relative way to compose risk budgets. It says nothing about optimal probabilities, uncertainty calibration, or how the ε_i are to be established. Those questions require a separate probabilistic model and empirical justification. Qualitative adversarial counter-repair and probabilistic degradation should therefore be reported as *different* certificate types.

19.5 Domain-Specific Modeling Obligations

The following are mathematical *research interpretations*, not operationally validated deployments. Each requires new ontology engineering and domain-specific semantics; none inherits chess's legal-move oracle automatically.

Domain	Candidate state/repair interpretation	Additional proof obligation
Intelligence analysis	Hypothesis or belief states; revision of an inconsistent or poorly supported assessment	Evidence provenance, uncertain observations, and calibrated probabilistic claims
War-game models	Abstract resource and viability conditions; restoration of a constrained strategic objective	Multi-agent timing, nondeterministic outcomes, and resource-invariant soundness
Cybersecurity	Security-configuration graphs; repairs of explicitly modeled defensive invariant violations	Concurrency, partial observability, faithful access-control semantics, and adversarial completeness
Espionage modeling	Information sets and permitted information flows; repairs of specified epistemic or confidentiality constraints	Observation-uniform strategies and knowledge-sensitive equivalence

A biological regenerative runtime, considered in the originating AAD Systems repair-algebra work [22], introduces yet another kind of obligation: continuous or stochastic state evolution and an independently justified viability region. In each domain, the *schema* of failure, guarded repair, adversarial interference, strategic transfer, and checkable contracts may be useful. But logical soundness transfers only after the state model and its observations have been shown to represent the intended real-world phenomena to an appropriate degree of fidelity.

19.6 Generalization and Research Program

Three distinctions guide the proposed extension:

(i) *Syntactic portability* means that definitions and proof rules can be written over a new state space. It does not imply that the predicates capture useful domain-specific failure modes.

(ii) *Semantic soundness* means that the model's transitions, observations, and strategy quantifiers are faithfully implemented. It does not establish that the model adequately describes an actual organization, biological process, or adversarial environment.

(iii) *Practical usefulness* concerns whether certified repairs and explanatory footprints improve real decisions under resource, uncertainty, or human-comprehension constraints. It is an empirical question to be tested separately in each domain.

The immediate mathematical agenda is to characterize *minimal* contract observations under which policy-relative refinement remains sound, and then to determine how those observations must be extended when state is hidden, transitions are probabilistic, or multiple agents act asynchronously. The chess-specific proofs provide a controlled foundation; theorems such as Theorems 19.4 and 19.9 and Proposition 19.8 identify both what may be reused and the assumptions that must be re-established. These are conditional results and design criteria, not evidence of validated intelligence, security, or other non-chess applications.

20. Discussion, Contributions, and Research Outlook

The present work develops an integrated mathematical account of chess as an adversarial semantic system. Its organizing objects are explicit failure predicates, guarded repairs, composable strategic operations, adversarial certificates, and transferable semantic footprints. The central contribution is a progression from *local state*

diagnosis to *contractually specified strategy*: moves are analyzed by the properties they change, strategies by the properties they maintain against legal opposition, and transfers by the relationships required to preserve their guarantees.

The framework builds on the Repair Algebras program of AAD Systems [22, 23]. In chess, the viability and restoration perspective acquires an adversarial dimension: the opponent can introduce new failures, obstruct a proposed repair, and force tradeoffs between simultaneously desirable conditions. The mathematics developed here makes these interactions explicit and gives them a common language.

20.1 Principal Mathematical Contributions

The paper contributes a connected sequence of constructions. First, finite, computable failure predicates and guarded repairs provide a transparent description of local semantic change. A compositional repair ledger retains the intermediate effects of a move sequence, including transient vulnerabilities that its final state would otherwise conceal. Second, bounded adversarial strategies and counterstrategies are represented by proof objects whose obligations follow the alternation of legal play. Third, strategic macros support ordered composition, contractual interfaces, and safe substitution. Fourth, semantic transfer and chess-specific footprints express precisely which relationships must survive when a plan is reused in another position. Finally, positional retention kernels and multiobjective portfolios capture maintenance, competition, and conflict between strategic conditions.

Mathematical layer	**Construction**	**Role in the theory**
Local semantic change	Failure predicates, guarded repairs, compositional ledgers	Identifies and records the exact effects of legal moves
Adversarial strategy	Alternating certificates, refutations, strategy macros	Represents plans that account for legal opposition
Strategic reuse	Contract interfaces, selective matching, chess-specific footprints	Transfers certified constructions among related positions
Positional persistence	Retention kernels, competing objectives, repair portfolios	Formalizes sustained advantages and strategic tradeoffs
Executable semantics	Faithful state refinement, independent checking, evaluation protocol	Connects mathematical claims to verifiable computational artifacts

These constructions are related to established contract reasoning, alternating simulation, model checking, and multiobjective optimization [7, 11, 10]. The synthesis here is their organization around the failure-and-repair ontology and the concrete adversarial structure of chess. Individual priority questions belong to scholarly comparison with those literatures; the present framework states its results by explicit mathematical assumptions and proofs.

20.2 Contractual Precision and Chess Meaning

The theory deliberately separates the meaning of a *declared contract* from the broader task of selecting a strong chess move. A contract names its predicates, safety conditions, horizon, and quantification over possible replies. When the hypotheses of the corresponding theorem hold, its conclusion is precise: the designated property is achieved or maintained within that formal model. This precision is one of the framework's strengths, because it makes an explanation checkable rather than leaving it as an informal assessment.

Strategic evaluation adds a second layer: the choice of which contracts matter in a particular position. Sacrifices, initiative, king safety, passed pawns, and positional compensation illustrate why several competing

objectives can be relevant at once. The multiobjective results of Section 14 provide a formal setting for representing these choices without arbitrarily compressing them into one score. The research program thereby separates *correct execution of a strategic specification* from *selection of a strategically valuable specification*, making each a well-posed problem in its own right.

20.3 Semantic Footprints and the Reuse of Strategy

Semantic footprints connect the abstract contracts to recognizable chess structure. A footprint records selected conditions on a position, while its dependency certificate accounts for the legal transitions and opponent replies required by a specific claim. The concrete examples in Sections 11 and 12 show both successful reuse and decisive counter-repairs. In this way, chess examples serve as mathematical witnesses to the strategy-transfer calculus.

A particularly productive notion is the *contract-sufficient footprint*: a collection of observations and branch dependencies that supports the transfer of a specified strategy over a specified class of positions. This formulation suggests three directions for further mathematical work: constructive synthesis of sufficient footprints; refinement driven by counterexamples; and characterization of minimal or economical representations relative to a chosen observation language. The most useful representations may be compact precisely because they retain the relevant relationships rather than every detail of the board.

20.4 A Certifying Compiler Architecture

The proposed compiler architecture translates the preceding mathematics into a modular verification design. It distinguishes the component that *proposes* a move or strategic construction from the checker that *validates* its declared contract. The latter reconstructs relevant legal successors from complete chess states, evaluates semantic predicates, and checks the required opponent branches. In this architecture, a candidate generator can be heuristic, algebraic, search-based, or hybrid, while acceptance of a formal claim follows the same checking discipline.

A faithful implementation must represent the full game state, including piece placement, side to move, castling rights, en-passant eligibility, repetition-relevant history, move counters, and terminal rules. The implementation plan of Section 16 specifies corresponding refinement and validation obligations. These are constructive engineering requirements: they identify exactly which interfaces and invariants the executable system must realize in order to inherit the mathematical soundness results.

20.5 The Experimental Research Program

The experimental protocol of Section 17 provides a direct path from mathematical specification to reproducible evidence. It distinguishes four complementary forms of progress: soundness of accepted certificates, coverage and reuse of strategic footprints, computational efficiency, and quality of play. These yield separate opportunities for contribution. For example, a semantic checker may add value through independently inspectable explanations; a compact footprint may add value through reuse; and a repair-guided search policy may add value through stronger or more efficient play.

The proposed comparisons use fixed conventional-engine baselines, including Stockfish, with controlled hardware, time limits, opening selection, color assignment, and reproducible configurations [5, 18, 17]. This design makes it possible to attribute observed effects to individual semantic components through ablation studies, while reporting uncertainty and adversarially chosen counterexamples alongside successful cases.

The bounded-search specification also identifies a concrete complexity parameter. With branching bounded

by $b \geq 1$ and horizon h, a naive complete expansion visits at most

$$N_h \leq \sum_{j=0}^{h} b^j \tag{106}$$

nodes. Sharing transpositions, retrieving verified macros, and selectively matching strategy dependencies are natural mechanisms to investigate for reducing repeated work. The separate cost of constructing and checking certificates is part of that evaluation.

20.6 Human Chess Reasoning and Broader Adversarial Systems

The four-pass human protocol of Section 18—scan, repair, reply, recheck—is a compact interpretation of the formal calculus. It makes failure identification and counter-repair analysis available as an organized method of over-the-board thought. The explanation labels *certified*, *refuted*, and *unresolved* connect the human-facing presentation to the evidence structure of the compiler. Controlled studies can evaluate their effects on comprehension, decision quality, and the recognition of tactical exceptions.

The abstract theory also supports a general research program in adversarial computation. Fully observed deterministic arenas admit versions of the chess constructions under the transfer hypotheses of Section 19. Domains with hidden information, probabilistic transitions, or asynchronous interaction motivate corresponding extensions to belief states, risk budgets, and richer game semantics. Chess thereby provides a rigorous first arena from which the mathematical program can expand without conflating distinct domain assumptions.

20.7 Research Outlook and Publication Trajectory

The next descent has a clear mathematical and computational sequence:

(i) *Proof consolidation.* Audit the quantifiers, terminal conventions, theorem dependencies, and chess-state assumptions across the complete manuscript; mechanize representative results where useful.

(ii) *Chess-specific footprint discovery.* Construct and refine contract-sufficient footprints for increasingly varied tactical and positional families, with reproducible certificates.

(iii) *Executable verification.* Implement the complete state kernel, semantic predicates, and independent certificate checker using the refinement principles already specified.

(iv) *Controlled evaluation.* Measure certification fidelity, strategy reuse, search cost, playing strength, and human explanatory value against registered baselines.

(v) *Application and generalization.* Build the interactive chess demonstration and extend the mathematical methods to other adversarial domains with appropriately defined semantics.

The mathematical paper, the verified executable system, and the chess application are complementary expressions of one research program. The first supplies the formal language and compositional results; the second turns their proof obligations into computational checks; the third makes the resulting strategic reasoning accessible to chess players and researchers. Together they define a pathway from repair algebra to adversarial semantic computation.

21. Conclusion

The aim of this work has been to develop a mathematically explicit account of chess as an *executable semantic system*. Its starting point is an ontology of states and failure conditions rather than a numerical evaluation

alone. This orientation draws on the repair-algebra program of AAD Systems [22, 23]: identify what a state fails to satisfy, specify transformations that could repair that failure, and make the conditions for accepting those transformations precise. Chess supplies an unusually exact adversarial setting in which to test how far this approach can be developed.

21.1 From Local Repairs to Composable Strategies

The central mathematical progression is from diagnosis to adversarial control. Local repair relations and compositional ledgers record changes in explicitly named predicates; guarded repair distinguishes an improvement from a change that violates an accompanying safety condition. The next level is strategic: alternating legal moves require a plan that remains valid under *all* opponent responses covered by its contract. We accordingly construct bounded strategy certificates, counterstrategy refutations, reusable macros, and composition conditions. The positional-retention results extend repair beyond a single successful move to continued maintenance, while the multiobjective results make precise why individually achievable repairs need not admit a common strategy (Sections 4, 6, 7, 13 and 14).

A second progression is from constructing a strategy in one state to justifying its use in another. Contract-respecting relations and strategy-relative matching specify sufficient conditions for transfer, but shared failure labels alone do not establish those conditions. The chess witnesses developed in Sections 11 and 12 show both sides of this distinction: a compact pattern can support the same verified continuation in different legal positions, and an additional legal counter-reply can invalidate the proposed transfer. The resulting *semantic footprint* is therefore best understood as a candidate representation of a strategy's dependencies, not as an automatic proof that the strategy succeeds.

21.2 Mathematical Contributions and Experimental Realization

The manuscript provides a coherent formal progression: local repair predicates, compositional ledgers, bounded adversarial certificates, strategic macros, selective strategy transfer, positional retention kernels, and multiobjective repair portfolios. The theorems establish stated properties of these constructions under explicit hypotheses, and the finite chess witnesses illustrate how strategy dependencies and counter-repairs arise in legal positions. The resulting framework supplies an algebraic language for both the construction and the independent checking of strategic claims (Sections 17 and 20).

The proposed compiler turns this language into an executable architecture. Candidate moves and strategic certificates are generated by a semantic or hybrid search process, while an independent checker validates their legal transitions, failure predicates, and adversarial branches. This separation makes correctness obligations explicit and reusable across search methods. The implementation stages of Sections 15 and 16 identify how a faithful chess-state kernel and certificate checker can realize the mathematical specifications.

The evaluation protocol complements the theory by measuring several distinct kinds of success: formal certification, coverage of transferable strategic patterns, computational efficiency, move quality, playing strength, and human comprehension. Fixed conventional-engine baselines and controlled ablations will allow the contribution of the semantic architecture to be assessed independently of the strength of any particular search engine. Together, these measurements create a reproducible route from formal structure to an operational chess system.

21.3 The Next Semantic Descent

The most concentrated open question is whether there exist *small, contract-sufficient chess footprints* that are both reusable and economical to verify. Such footprints would have to retain the precise dependencies on

which a chosen strategy rests, including every opponent response relevant to its guarantee, while omitting observations that do not affect that guarantee. Establishing useful families of such representations, rather than merely defining them, is the bridge between the present formal calculus and an operational semantic chess system.

Beyond chess, the abstract constructions suggest a more general study of adversarial repair and persistence. Their transfer to intelligence analysis, cybersecurity, or other domains is conditional on faithful domain models; incomplete information and stochastic transitions introduce obligations absent from ordinary chess (Section 19). Chess is thus the first precisely specified arena for the calculus, with additional adversarial domains obtained through explicit modeling and refinement conditions.

The contribution of this paper is a unified, inspectable route from *failure ontology* through *repair* and *counter-repair* to *certified strategy*, *semantic transfer*, and an *executable verification architecture*. It places the next research question in a form that can be proved or refuted by a combination of formal mathematics, chess-specific certificates, and reproducible experiments: *can semantic structure make adversarial reasoning more reusable, more auditable, and ultimately more useful in actual play?*

A. Notation and Proof Obligations

Symbol	Meaning
$\mathcal{S}$	Valid history-aware chess states
$\mathcal{L}(s)$	Legal moves available in state s
$\delta(s, m)$	Partial legal state transition
$\mathcal{F}$	Finite family of computable failure predicates
$C(s, p)$	Failure classes satisfied by player p in state s
$\Phi(s, p)$	Binary semantic signature
$R_f(s, m)$	Local repair of failure predicate f
$R_f^{(1)}(s, m)$	Repair robust against every immediate legal reply
$\Delta(s, m, p)$	Componentwise change of semantic signature
$\Lambda_p(\gamma)$	Gross repair/introduction ledger of a legal path
$R_{f;g}(s, m)$	Guarded repair of predicate f preserving guard g
D^-, D^+	Failure-removal and failure-introduction sets
$H_{w,J}$	Quadratic semantic interaction energy
$\mathsf{K}_{f;g}(s, m)$	Counterresponse witness set for guarded repair
$\mathsf{CPre}_p(X)$	Adversarially controllable predecessor of set X
$\mathsf{W}_p^h(A, B)$	States guaranteeing target B within h plies while remaining in A
$\rho_{p;A,B}(s)$	Least guaranteed repair horizon, or infinity
$\tau_p(\gamma)$	Ordered trace of semantic signatures along a legal path
N	Set of nonterminal states
$\sigma_1; \sigma_2; \sigma_3$	Phase-indexed guarded strategy composition
α	Semantic abstraction subject to transition-stability criterion
$\mathfrak{M}$	Certified strategic macro with entry, safety, goal, interface and horizon
$\mathcal{R}$	Contract-respecting state relation
$\mathcal{R}_h$	Depth-h contract equivalence relation
$\omega_{A,G,I}$	Contract observation recording turn, terminality, safety and release
$\mathcal{K}$	Strategy-directed relation between paired legal histories
Match_j	Bounded selective strategy-transfer proof obligation
$\mathsf{K}_p^j(B)$	j-ply adversarial positional retention kernel
$d_{p,B}^H(s)$	Bounded duration of guaranteed positional retention
$V_Q(s)$	Compact human semantic observation of state s
$\mathrm{Advice}_\kappa(v; D)$	Uniformly contract-admissible moves across a human-view class
$\mathsf{Enc}, \mathsf{Dec}$	Faithful chess-state encoding and decoding
$\mathsf{E}_p(s, m)$	Checked semantic intermediate-representation edge
$\mathsf{CheckPort}$	Independent verifier for adversarial portfolio certificates
$V(s, m)$	Proposed side-to-move move-evaluation function
$\mathcal{A}$	General turn-based adversarial repair arena of Definition 19.1
$R \subseteq S_0 \times S_1)$	Policy-relative adversarial refinement relation
o_p	Protagonist observation map for uniform strategies
ε_i	Stage-wise conditional failure-risk upper bound

References

[1] Claude E. Shannon. *Programming a Computer for Playing Chess*. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, vol. 41, no. 314, pp. 256–275, 1950. doi:10.1080/14786445008521796.

[2] Arthur L. Samuel. *Some Studies in Machine Learning Using the Game of Checkers*. IBM Journal of Research and Development, vol. 3, no. 3, pp. 210–229, 1959. doi:10.1147/rd.33.0210.

[3] Michael Genesereth, Nathaniel Love. *General Game Playing: Overview of the AAAI Competition*. Stanford University, 2005. https://citeseerx.ist.psu.edu/document?doi=6656451fb5fbcb556cefffad144bd5c1fbfd7870&repid=rep1&type=pdf.

[4] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, Demis Hassabis. *A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-Play*. Science, vol. 362, no. 6419, pp. 1140–1144, 2018. doi:10.1126/science.aar6404.

[5] Stockfish Developers. *Stockfish: Strong Open-Source Chess Engine and Documentation*. Project documentation, 2026. Accessed November 16, 2026. https://stockfishchess.org/documentation.

[6] Semyon Lomasov, Judah Goldfeder, Mehmet Hamza Erol, Matthew So, Yao Yan, Addison Howard, Nathan Kutz, Ravid Shwartz-Ziv. *Exploring Human-AI Conceptual Alignment through the Prism of Chess*. Advances in Neural Information Processing Systems (NeurIPS 2025), Creative AI Track, 2025. https://arxiv.org/abs/2510.26025.

[7] C. A. R. Hoare. *An Axiomatic Basis for Computer Programming*. Communications of the ACM, vol. 12, no. 10, pp. 576–580, 1969. doi:10.1145/363235.363259.

[8] Amir Pnueli, Roni Rosner. *On the Synthesis of a Reactive Module*. Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 179–190, 1989. doi:10.1145/75277.75293.

[9] Robin Milner. *Communication and Concurrency*. Prentice Hall, ISBN 0131150073, 1989.

[10] Christel Baier, Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, ISBN 9780262026499, 2008. https://mitpress.mit.edu/9780262026499/principles-of-model-checking/.

[11] Rajeev Alur, Thomas A. Henzinger, Orna Kupferman, Moshe Y. Vardi. *Alternating Refinement Relations*. CONCUR '98: Concurrency Theory, Springer, Lecture Notes in Computer Science, vol. 1466, pp. 163–178, 1998. doi:10.1007/BFb0055622.

[12] Marta Kwiatkowska, David Parker, Clemens Wiltsche. *PRISM-games: Verification and Strategy Synthesis for Stochastic Multi-player Games with Multiple Objectives*. International Journal on Software Tools for Technology Transfer, vol. 20, pp. 195–210, 2018. doi:10.1007/s10009-017-0476-z.

[13] George C. Necula. *Proof-Carrying Code*. Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, ACM, pp. 106–119, 1997. doi:10.1145/263699.263712.

[14] FIDE. *FIDE Laws of Chess Taking Effect from 1 January 2023*. FIDE Handbook, E.01, 2023. Accessed November 17, 2026. https://handbook.fide.com/chapter/E012023.

[15] python-chess Contributors. *python-chess: A Chess Library for Python*. Software documentation, n.d.. Accessed November 16, 2026. https://python-chess.readthedocs.io/en/latest/.

[16] Stockfish Developers. *Stockfish Documentation: Frequently Asked Questions and Engine Evaluation*. Official Stockfish documentation, 2026. Accessed November 17, 2026. https://official-stockfish.github.io/docs/stockfish-wiki/Stockfish-FAQ.html.

[17] Cute Chess Contributors. *Cute Chess: Command-Line Engine Tournament Runner*. 2026. Project repository; accessed November 17, 2026. https://github.com/cutechess/cutechess.

[18] Stockfish Fishtest Contributors. *Fishtest Testing Guide and Sequential Probability Ratio Tests*. Stockfish project documentation, 2026. Accessed November 17, 2026. https://official-stockfish.github.io/docs/fishtest-wiki/Creating-my-first-test.html.

[19] Adriaan D. de Groot. *Thought and Choice in Chess*. Mouton, 2 ed., 1978. doi:10.1515/9783110800647.

[20] William G. Chase, Herbert A. Simon. *Perception in Chess*. Cognitive Psychology, vol. 4, no. 1, pp. 55–81, 1973. doi:10.1016/0010-0285(73)90004-2.

[21] Fernand Gobet, Herbert A. Simon. *Templates in Chess Memory: A Mechanism for Recalling Several Boards*. Cognitive Psychology, vol. 31, no. 1, pp. 1–40, 1996. doi:10.1006/cogp.1996.0011.

[22] Adrian Diamond. *Repair Algebras and Regenerative Runtimes: A Runtime Semantics for Biological Systems*. AAD Systems, 2026. https://aadsystems.com/papers/Repair_Algebras_and_Regenerative_Runtimes__A_Runtime_Semantics_for_Biological_Systems.pdf.

[23] Adrian Diamond. *Executable Σ-Types: A Runtime for Proof-Carrying State Transitions via Gödel Encoding*. AAD Systems, 2026. https://aadsystems.com/papers/Executable-sigma-types.pdf.

[24] John H. Reif. *The Complexity of Two-Player Games of Incomplete Information*. Journal of Computer and System Sciences, vol. 29, no. 2, pp. 274–301, 1984. doi:10.1016/0022-0000(84)90034-5.

[25] Leslie Pack Kaelbling, Michael L. Littman, Anthony R. Cassandra. *Planning and Acting in Partially Observable Stochastic Domains*. Artificial Intelligence, vol. 101, no. 1–2, pp. 99–134, 1998. doi:10.1016/S0004-3702(98)00023-X.