A Top-Down Logical Framework for Classifying Tree Structures and Their APIs

General Statement

We fix a language that speaks only of a rooted partial order and label maps into ordered domains. Observables are logical tests from a bounded fragment aligned with maintainable invariants. APIs are the closure of these observables under operations whose preconditions and postconditions can be preserved in logarithmic time. A tree type is a logical theory in this language. Concrete families are later instantiations by adding axioms and choosing a test family. Concrete names such as parent and child are derived from the cover relation, not primitive.

Obstruction Map

@l l l l l l@ **Witness Faultline Target Scope Remedy/Action TN**
Competing vocabularies Mixed abstraction levels Classify by observable power Finite rooted posets with labels Fix signature, tests, observable API and strength preorder Next: pin logical core, defer families
Over-strong interval equivalence Collapses to equality Observation-aligned equivalence Bounded logical fragment Choose a test family $\mathcal{F}$; define $x \approx_{\mathcal{F}} y$ Next: define $\mathcal{F}$, $\mathcal{O}$, $\preceq$

1 Language Arena

Motivation (M)

Classification starts from language that expresses order and label comparisons so that families differ by axioms rather than by implementation words.

Structure (S)

Definition 1.1 (Signature)**.** $\Sigma = \{\ \sqsubseteq\ ;\ (f_\alpha : X \to K_\alpha,\ \leq_\alpha)_{\alpha \in A}\ \}$ where $(X, \sqsubseteq)$ is a finite rooted poset whose Hasse diagram is a tree, and each f_α is a label map into an ordered set K_α.

Definition 1.2 (Structure)**.** A structure is $\mathcal{S} = (X, \sqsubseteq, (f_\alpha)_\alpha, \mathrm{Inv})$ where Inv are maintainability conditions that guarantee height bounds and local repair costs. The cover relation is

$$uv \iff u \sqsubset v \text{ and there is no } w \text{ with } u \sqsubset w \sqsubset v.$$

Parent and child are names derived from covers. They are not symbols of Σ.

Definition 1.3 (Generated order, optional provenance)**.** Optionally record a definable generating relation $G \subseteq X \times X$ and set $\sqsubseteq = (G)$, the least partial order containing G.

TN

From the bare language move to what can be seen and maintained: logical tests and their induced equivalence.

2 Logical Observables and Observation-Aligned Equivalence

Motivation (M)

APIs expose only what can be observed and maintained cheaply. Tests must be fixed before types are promised.

Structure (S)

Definition 2.1 (Logical fragment and test family)**.** $\mathcal{L}_\Sigma$ uses atomic $\sqsubseteq$ and $\leq_\alpha$, finite Boolean operations, and bounded quantifiers restricted to $\downarrow x$, $\uparrow x$, or a root path. A *test family* is $\mathcal{F} \subseteq \mathrm{Def}(\mathcal{L}_\Sigma)$.

Definition 2.2 (Observation map and equivalence)**.** For a structure $\mathcal{S}$ on carrier X, define $\chi_\mathcal{S} : X \to \{0,1\}^\mathcal{F}$ by

$$\chi_\mathcal{S}(x)(F) = 1 \iff x \models_\mathcal{S} F.$$

Define $x \approx_\mathcal{F} y \iff \chi_\mathcal{S}(x) = \chi_\mathcal{S}(y)$. The granularity of $\approx_\mathcal{F}$ is determined by $\mathcal{F}$.

Definition 2.3 (Positive tests and information preorder)**.** For $x \in X$ put $\mathrm{Pos}_{\mathcal{F}}(x) = \{\, F \in \mathcal{F} \mid x \models_{\mathcal{S}} F \,\}$. Define a preorder on X by

$$x \preceq_{\mathcal{F}} y \quad \Longleftrightarrow \quad \mathrm{Pos}_{\mathcal{F}}(x) \subseteq \mathrm{Pos}_{\mathcal{F}}(y).$$

Then $x \approx_{\mathcal{F}} y$ iff $x \preceq_{\mathcal{F}} y$ and $y \preceq_{\mathcal{F}} x$.

Definition 2.4 (Quotient partial order)**.** Let $X_{\mathcal{F}} = X/\approx_{\mathcal{F}}$ be the set of $\mathcal{F}$-classes and write $[x]_{\mathcal{F}}$ for the class of x. Define

$$[x]_{\mathcal{F}} \ \sqsubseteq_{\mathcal{F}} \ [y]_{\mathcal{F}} \quad \Longleftrightarrow \quad x \preceq_{\mathcal{F}} y.$$

This is well-defined and makes $(X_{\mathcal{F}}, \sqsubseteq_{\mathcal{F}})$ a partial order.

Proposition 2.5 (Monotonicity in tests)**.** *If $\mathcal{F} \subseteq \mathcal{G}$ then:*

1. *$\approx_{\mathcal{G}}$ refines $\approx_{\mathcal{F}}$, and the canonical surjection $\pi_{\mathcal{G}\to\mathcal{F}} : X_{\mathcal{G}} \to X_{\mathcal{F}}$, $[x]_{\mathcal{G}} \mapsto [x]_{\mathcal{F}}$, is order-preserving w.r.t. $\sqsubseteq_{\mathcal{G}}$ and $\sqsubseteq_{\mathcal{F}}$.*
2. *$\preceq_{\mathcal{G}}$ refines $\preceq_{\mathcal{F}}$ on X.*

Definition 2.6 (Refinement preorder on structures)**.** Fix X and $\mathcal{F}$. For two structures $\mathcal{S}, \mathcal{T}$ on X write

$$\mathcal{S} \succeq_{\mathcal{F}} \mathcal{T} \quad \Longleftrightarrow \quad (\forall x, y \in X)\ \big(x \approx^{\mathcal{S}}_{\mathcal{F}} y \ \Rightarrow \ x \approx^{\mathcal{T}}_{\mathcal{F}} y\big).$$

Equivalently, there exists an order-preserving surjection $\rho_{\mathcal{S}\to\mathcal{T}} : X^{\mathcal{S}}_{\mathcal{F}} \to X^{\mathcal{T}}_{\mathcal{F}}$ sending $[x]^{\mathcal{S}}_{\mathcal{F}}$ to $[x]^{\mathcal{T}}_{\mathcal{F}}$.

Definition 2.7 (Canonical representatives (optional))**.** Choose any linear extension $\leq^{\mathrm{lin}}_{\mathcal{F}}$ of the partial order $\sqsubseteq_{\mathcal{F}}$ on $X_{\mathcal{F}}$. Pick in each class $[x]_{\mathcal{F}}$ the $\leq^{\mathrm{lin}}_{\mathcal{F}}$-least representative as a canonical choice. This choice does not affect $\approx_{\mathcal{F}}$ or $\sqsubseteq_{\mathcal{F}}$.

Definition 2.8 (Action alphabet and output levels)**.** Let K be the key domain and $\mathrm{Ops} = \{\mathrm{search}, \mathrm{insert}\}$. Define the input alphabet $A = \mathrm{Ops} \times K$. Choose a level $L \in \{\mathrm{L0}, \mathrm{L1}, \mathrm{L2}\}$ with

$$\mathrm{L0}: \ \Omega = \{\mathrm{hit}, \mathrm{miss}\}, \qquad \mathrm{L1}: \ \Omega = \{\mathrm{hit}(d), \mathrm{miss}(d) \mid d \in \mathbb{N}\},$$

L2 : Ω extends L1 by policy-visible events returned by the API at layer L2.

For $\langle \mathrm{search}, k\rangle$ take δ to be identity on the tree and o the membership outcome at level L; for $\langle \mathrm{insert}, k\rangle$ take δ to insert k by the current policy and o the level-L observable outcome.

Definition 2.9 (Trace semantics and API-trace equivalence)**.** Extend δ, o to words $w \in A^*$ by

$$\hat{\delta}(x,\epsilon) = x, \quad \hat{\delta}(x,aw) = \hat{\delta}(\delta(x,a),w), \qquad \hat{o}(x,\epsilon) = \epsilon, \quad \hat{o}(x,aw) = o(x,a)\cdot\hat{o}(\delta(x,a),w).$$

For level L write

$$x \equiv_L y \iff \forall w \in A^*,\ \hat{o}(x,w) = \hat{o}(y,w).$$

Proposition 2.10 (Right congruence)**.** $\equiv_L$ *is an equivalence relation and a right congruence for* (X, A, δ, o)*.*

Definition 2.11 (Admissible test family relative to L)**.** A test family $\mathcal{F} \subseteq \mathrm{Def}(L_\Sigma)$ is admissible for an API layer L if (i) every atomic predicate in $\mathcal{F}$ is reconstructible from the observable output alphabet Ω_L (API alignment), and (ii) each formula in $\mathcal{F}$ uses only bounded locality (a fixed window/depth within $\downarrow x$, $\uparrow x$, or a root path). Equivalently, for each atomic $\theta \in \mathcal{F}$ there exist a finite experiment scheme E_θ and a map g_θ with $\theta(x) = g_\theta(\{\hat{o}(x,w) \mid w \in E_\theta\})$.

[Alignment] If $\mathcal{F}$ is admissible for L, then the observational equivalence $\approx_\mathcal{F}$ refines the API trace equivalence $\equiv_L$.

Definition 2.12 (Rank abstraction to a finite local alphabet)**.** For finite $x \in X$, list keys in-order by ranks $1, \ldots, n$ and add sentinels to form $n{+}1$ gaps. Define the local finite alphabet

$$A_x = \{\mathrm{hit}(i)\}_{i=1}^{n} \ \cup\ \{\mathrm{miss}(j)\}_{j=0}^{n} \ \cup\ \{\mathrm{ins\_gap}(j)\}_{j=0}^{n},$$

by mapping $\langle \mathrm{search}, k\rangle$ and $\langle \mathrm{insert}, k\rangle$ to the rank or gap of k in x. This erases absolute values and keeps relative position.

Principle 2.13 (Discriminability). Selection of $\mathcal{F}$ aligns with intended APIs. Larger $\mathcal{F}$ increases discriminability but raises maintenance cost; smaller $\mathcal{F}$ collapses indistinctions and weakens possible APIs.

TN. With tests fixed, collect the maintainable ones into an observable core and order structures by observable power. Rank abstraction provides a finite, key-agnostic interface for equivalence and minimal quotients; the quotient poset $(X_\mathcal{F}, \sqsubseteq_\mathcal{F})$ is the canonical skeleton on which these quotients are computed.

Examples

Example 2.14 (L0 collapses more than L1)**.** Let $K = \{2, 5, 9\}$. Consider T_{chain} built by inserts $2, 5, 9$ and T_{bal} by inserts $5, 2, 9$. At $L0$ both answer hit/miss identically for all traces, hence $T_{\text{chain}} \equiv_{L0} T_{\text{bal}}$. At $L1$, search(7) has depth 3 in T_{chain} versus 2 in T_{bal}, so they are distinguished and not $\equiv_{L1}$.

Example 2.15 (Rank abstraction in action)**.** Let T have in-order ranks $1, 2, 3, 4$. Then

$$A_T = \{\text{hit}(1..4),\ \text{miss}(0..4),\ \text{ins\_gap}(0..4)\}.$$

Any concrete $\langle \text{search}, k\rangle$ maps to hit(i) if k equals the i-th key, or to miss(j) if k falls into the j-th gap. This yields a finite observable interface independent of absolute key values.

3 Observable API and Strength Preorder

Motivation (M)

Classification by capability requires a canonical core of observables controlled by invariants that certify logarithmic maintenance.

Structure (S)

Definition 3.1 (Observable core)**.** $\mathcal{O}(\mathcal{S})$ is the closure under admissible composition of atomic observables maintainable in $O(\log n)$ under Inv:

$$\{\text{cmp}_\alpha,\ \text{in interval } \alpha,\ \min_\alpha,\ \max_\alpha,\ \text{pred},\ \text{succ},\ \text{split},\ \text{join}\}$$

plus carried aggregations such as subtree size and interval sums.

Definition 3.2 (Levels as observable choices)**.** Level $L0$ keeps only membership outcomes; $L1$ augments with depth-based costs; $L2$ admits policy-visible effects consistent with $\mathcal{F}$. Each level induces $\equiv_L$ and a quotient $X/{\equiv_L}$.

Definition 3.3 (API strength preorder)**.** For two structures on the same signature,

$$\mathcal{S} \preceq \mathcal{T} \iff \mathcal{O}(\mathcal{S}) \subseteq \mathcal{O}(\mathcal{T}).$$

Lemma 3.4 (Monotonicity under quotients)**.** *Let $\pi : X \to X/{\equiv_L}$ be the quotient by API-trace equivalence at level L. Every $f \in \mathcal{O}(\mathcal{S})$ factors uniquely through π. Hence $\mathcal{O}(\mathcal{S}) \subseteq \mathcal{O}(\mathcal{S}/{\equiv_L})$ canonically.*

Proposition 3.5 (Preorder and policy forgetfulness)**.** *$\preceq$ is a preorder. If a forgetful functor U erases policy-internal artifacts not exposed at level L, then $\mathcal{S} \preceq \mathcal{T}$ implies $U(\mathcal{S}) \preceq U(\mathcal{T})$. In particular, AVL and Red-Black trees coincide at $L0$ and often coarsen at $L1$.*

Rule 3.6 (Interface style)*.* Each operation is specified by preconditions and postconditions in $(\sqsubseteq, (f_\alpha)_\alpha, \mathcal{F}, \mathrm{Inv})$.

Principle 3.7 (Cost alignment)*.* Inv fixes height bounds and local repair cost, hence time bounds for $\mathcal{O}(\mathcal{S})$.

Examples

Example 3.8 (Strength comparison via $\mathcal{O}$)**.** Let $\mathcal{S}$ be a plain BST with $\mathrm{Inv} = \{\text{BST order}\}$ and $\mathcal{T}$ be an AVL tree with $\mathrm{Inv} = \{\text{BST order, balance factor}\}$. Then

$$\mathcal{O}(\mathcal{S}) \subseteq \{\mathrm{cmp}_\alpha, \mathrm{pred}, \mathrm{succ}, \mathrm{split}, \mathrm{join}\} \subseteq \mathcal{O}(\mathcal{T}),$$

since AVL maintains the same observables within $O(\log n)$ and additionally certifies height bounds enabling worst-case guarantees. Thus $\mathcal{S} \preceq \mathcal{T}$.

Example 3.9 (Level effect: AVL vs. RB at $L0$)**.** Let $\mathcal{A}$ be AVL and $\mathcal{R}$ be Red-Black on the same key set. At level $L0$, membership outcomes coincide for all traces, hence any two states with the same key set are $\equiv_{L0}$. Therefore $X_{\mathcal{A}}/{\equiv_{L0}} \cong X_{\mathcal{R}}/{\equiv_{L0}}$ as observable quotients, even though their repair sequences differ.

4 Types as Logical Theories

Motivation (M)

Families are theories rather than code sketches. Strengthening a family means adding axioms in the same language.

Structure (S)

Definition 4.1 (Type and type order)**.** A type is a set of axioms $\Theta \subseteq \mathrm{Sent}(\mathcal{L}_\Sigma)$ with $\mathcal{S} \models \Theta$. Horn or universal axioms suffice for typical invariants. Define

$$\Theta_1 \sqsubseteq \Theta_2 \iff \Theta_2 \vdash \Theta_1.$$

This yields a lattice of types.

Principle 4.2 (Provenance)*.* If $G_1 \subseteq G_2$ then $(G_1) \subseteq (G_2)$. Increasing generator strength increases comparability and can enlarge $\mathcal{O}$.

TN

Before instantiating families, fix how concrete roles are derived from covers and label predicates.

5 Derived Roles from Covers

Motivation (M)

Implementation words must be consequences so that the theory remains portable across representations.

Structure (S)

Proposition 5.1 (Covers as primitive carriers)**.** *If a theory partitions* $\downarrow x \setminus \{x\}$ *into convex connected classes by label predicates, then covers* ux *in distinct classes serve as entrance points of those classes. Naming them children is definitional. If a label is axiomatized to be monotone along* $\sqsubseteq$*, then every cover inherits the order constraint without new primitives.*

TN

With derivation rules in place, families can be presented minimally as theories plus chosen tests.

6 Instantiation Templates (Skeletons Only)

Motivation (M)

Expose capability without fixing balancing or rotation details. Keep only axioms and intended tests.

Structure (S)

[Monotone heap] Language: one label $f : X \to K$ into a poset $(K, \leq)$. Theory Θ_{heap}: $\forall u \sqsubseteq v,\ f(u) \leq f(v)$. Tests $\mathcal{F}$: ideals and filters along $\sqsubseteq$. Observable core: min, push, pop_min, optional decrease_key, meld. Predecessor, successor and full range are absent.

[Search over a total key] Language: key $f : X \to K$ with a total order and f injective. Theory Θ_{search}: for each x, the strict down-set splits by $f(\cdot) < f(x)$ versus $f(\cdot) > f(x)$ into at most two convex connected classes, each adjacent to x if nonempty. Tests $\mathcal{F}$: half-infinite and finite key intervals. Observable core: search, lower_bound, predecessor, successor, range iteration; with size labels, kth and rank.

[Combined order, treap-style] Language: key f with total order, priority $g : X \to P$ with a poset order. Theory Θ_{comb}: Θ_{search} plus g monotone along $\sqsubseteq$. Tests $\mathcal{F}$: those of Template 6 plus priority tests if needed. Observable core: Template 6 plus split and join in $O(\log n)$.

Principle 6.1 (Strength inclusion). With the same $\mathcal{F}$: $\mathcal{O}(\text{heap}) \subset \mathcal{O}(\text{search}) \subset \mathcal{O}(\text{comb})$.

TN

Templates expose capability. Next, specify APIs by pre/postconditions in the same language.

7 API Specification Style

Motivation (M)

APIs must state only what the theory and invariants can guarantee. This isolates semantics from layout.

Structure (S)

Rule 7.1 (Canonical operations).

- **search(k)**. Pre: $k \in K$. Post: return x with $f(x) = k$ if present; path respects tests in $\mathcal{F}$.
- **insert(k, v)**. Post: resulting state satisfies the same Θ and Inv; height respects the bound given by Inv.
- **delete(k)**. Post: Θ and Inv preserved.
- **range(a, b)**. Pre: $a \le b$. Post: $\{x \mid a \le f(x) \le b\}$. Completeness is with respect to $\mathcal{F}$.
- **split(pivot)**, **join(S_1, pivot, S_2)**. Available only in types whose Θ and $\mathcal{F}$ entail $O(\log n)$ maintenance.

TN

With the protocol fixed, extend or restrict Θ, $\mathcal{F}$, and Inv to move up or down in capability and cost.

8 Extension Principles and Category-Lift Notes

Motivation (M)

Keep the logical spine stable while allowing principled strengthening. Lift only when it compresses proofs into reusable laws.

Structure (S)

Principle 8.1 (Minimal-sufficient design). Do not add axioms unless they yield a strictly stronger $\mathcal{O}$ that is actually used.

Principle 8.2 (Non-forgetfulness). Every stronger type records its added tests or invariants explicitly with a clear cost note.

Principle 8.3 (Category-lift, optional). When two types' observables are related by a non-forgetful functor and the diagram of implementations admits a colimit, the colimit represents a cross-family equivalence class of behaviors. Use only when it compresses arguments.

Notation and Conventions

$\downarrow x = x, \uparrow x = x$. Convex means closed under intervals in $\sqsubseteq$. All complexity claims are conditional on Inv. All inclusions for $\mathcal{O}$ are relative to the chosen $\mathcal{F}$.

Meta-Reflection

The note follows the language-first pipeline. Pressure is exposed in natural language. Minimal mathematics captures that pressure. Templates bridge to executable models and pre/post APIs. A return path to language is given by the observable preorder $\preceq$. The framework is ready for the next step: attach motivations for specific families then insert balancing invariants and cost bounds without altering the logical spine.