ABDUCTIO MVP

A Permutation-Invariant, Credit-Bounded Framework for Symmetric Hypothesis Evaluation

David Joseph (adaptable)

December 19, 2025

Abstract

ABDUCTIO MVP is a lightweight methodology for evaluating a mutually exclusive, collectively exhaustive (MECE) set of hypotheses under strict resource constraints. It is designed for domains in which one hypothesis may appear "far-fetched" yet plausibly correct, and where common reasoning failures arise from asymmetric scrutiny: evaluators decompose the focal hypothesis into many requirements while leaving rivals vague and atomic, allowing rivals to win by default.

ABDUCTIO MVP eliminates focal privilege and enforces **permutation invariance**: the output assigned to any hypothesis is independent of which hypothesis is chosen as a seed or listed first. The approach combines (i) stand-alone, well-defined hypotheses (no complement bundles), (ii) a fixed obligation template so each hypothesis must "pay the same kind of explanatory rent," (iii) a deterministic, seed-invariant credit allocation policy, and (iv) an auditable ledger update rule with an explicit open-world "Other" absorber.

The result is a publication-ready, implementation-ready framework that a software engineer can implement directly without EVSI calculations, Bayesian machinery, or complex statistical assumptions.

1. Motivation and Problem

Many controversial evaluations fail for a structural reason:

- Hypothesis H^* (often "far-fetched") is decomposed into multiple subclaims.
- Rival hypotheses R_i remain broad or underspecified.
- Evidence undermining one subclaim of H^* shifts weight to rivals.
- Rivals gain weight not because they are supported, but because they were not required to articulate necessary commitments.

This is not merely a cognitive bias ("argument from incredulity")—it is also a **systems design** failure. If the procedure taxes some hypotheses with specificity and not others, it bakes in unfairness.

ABDUCTIO MVP addresses this by requiring:

1. every hypothesis to be defined as a stand-alone mechanism, and
2. every hypothesis to be evaluated under the same obligation template and the same credit schedule, independent of ordering.

2. Core Requirement: Permutation Invariance

2.1 Informal statement

Given the same hypothesis set, the same evidence, and the same credit budget, the final (p, k) assigned to any hypothesis must not depend on:

- which hypothesis was chosen as "focal,"
- the order hypotheses are listed,
- the order evaluation steps are printed.

2.2 Formal statement

Let $H = \{h_1, \dots, h_n, h_{\text{other}}\}$ be a MECE set (named hypotheses plus a catch-all Other). Let an engine F map:

$$F(H, E, B, \theta) \mapsto \{(p(h_i), k(h_i))\}_{i=1}^{n} \cup (p(h_{\text{other}}), k(h_{\text{other}}))$$

where E is evidence, B a credit budget, and θ configuration parameters.

Permutation invariance requires that for any permutation π of the **named** hypotheses,

$$F(H, E, B, \theta) = F(\pi(H), E, B, \theta)$$

up to the same renaming/reordering of outputs.

2.3 Design implications

Permutation invariance forces three conditions:

1. **Semantic independence**: each hypothesis must be meaningful without reference to a "seed."
2. **Procedural symmetry**: credit allocation and stopping rules must not privilege any hypothesis.
3. **Determinism**: tie-breaking must not depend on presentation order.

ABDUCTIO MVP implements all three.

3. Design Principles

P1. Stand-alone hypotheses

Each named hypothesis must be describable without mentioning any other hypothesis. Prohibited: "NOT H1," "some mundane explanation," "any other cause," or umbrella OR-bundles as roots.

P2. MECE + explicit Other

The set of named hypotheses is intended to be mutually exclusive (ME) and collectively exhaustive (CE). Collective exhaustiveness is implemented pragmatically by always including:

- H_{other}: "Unknown/unmodeled explanation."

P3. No-free-probability

Listing more subcases must not increase a hypothesis's probability. Decomposition clarifies structure; it does not create credence.

P4. Same burdens for all

Each hypothesis is evaluated through a fixed **obligation template** (Section 6). This prevents one hypothesis from being saddled with "cosmic feasibility" while rivals face only local plausibility checks.

P5. Credit-bounded termination

Only two operations exist (Evaluate, Decompose), each costing 1 credit. The process halts by budget or by meeting confidence thresholds.

P6. Fully auditable

Every update must be reproducible from logged arithmetic and rubric scoring. No “implicit” ledger shifts are allowed.

4. Data Model

4.1 Hypothesis roots and nodes

A hypothesis is represented as a root node with an obligation template and optional internal decomposition trees.

```
from dataclasses import dataclass, field
from typing import Optional, Literal, Dict, List, Tuple

Role = Literal["NEC", "EVID"]
DecompType = Literal["AND", "OR"]
OrMode = Literal["EXCLUSIVE", "INCLUSIVE"]
Scope = Literal["LOCAL_ONLY", "GLOBAL_TO_TEMPLATE_SLOT", "GLOBAL_TO_ROOT"]

@dataclass
class Node:
    id: str
    statement: str

    # Local scores for this node (not necessarily ledger probability)
    p: float = 1.0          # default neutral for NEC nodes (see §7)
    k: float = 0.15

    # Audit
    k_rubric: Optional[Dict[str, int]] = None  # {"A":0..2,"B":0..2,"C":0..2,"D":0..2}
    factors: List[str] = field(default_factory=list)
    mind_change: Optional[str] = None
    evidence_refs: List[str] = field(default_factory=list)

    # Decomposition
    role: Optional[Role] = None
    children: Dict[str, "Node"] = field(default_factory=dict)
    decomp_type: Optional[DecompType] = None
    or_mode: Optional[OrMode] = None

    # AND coupling for NEC children (pragmatic dependence weight)
    coupling: Optional[float] = None  # one of {0.20, 0.50, 0.80, 0.95}

    # Accounting
    credits_spent: int = 0
    status: Optional[str] = None      # "SCOPED", "UNSCOPED"

@dataclass
class RootHypothesis:
```

```python
    id: str
    statement: str
    exclusion_clause: str  # one line: what makes this not any other root

    # Ledger probability (MECE bookkeeping)
    p_ledger: float
    k_root: float = 0.15

    # Obligation slots (fixed template; §6)
    obligations: Dict[str, Node] = field(default_factory=dict)

    # Audit
    credits_spent: int = 0

@dataclass
class HypothesisSet:
    roots: Dict[str, RootHypothesis]  # includes "H_other"
```

4.2 Ledger invariants

Let named roots be $H_1..H_n$ and H_{other}. Maintain:

- $p_{\text{ledger}}(h) \in [0, 1]$
- $\sum_{i=1}^{n} p_{\text{ledger}}(H_i) + p_{\text{ledger}}(H_{\text{other}}) = 1$

5. Cost Model

Only two operations exist.

- `DECOMPOSE(target)` : 1 credit
- `EVALUATE(target)` : 1 credit

Everything else (aggregation, ledger enforcement, scheduling) is "free" but must be logged.

6. Obligation Template (Permutation-Invariance Backbone)

Every named root hypothesis must be evaluated through the same template of obligation slots. This guarantees that each hypothesis faces comparable explanatory burdens.

6.1 Required slots (default MVP)

Each root H_i must define four slots:

1. **Feasibility (general)** [NEC]
   - The mechanism is possible in principle.
2. **Availability (context)** [NEC]
   - The mechanism is present/available in the specific time/place/context.
3. **Fit to key features** [NEC]
   - The mechanism explains the core reported observations better than at least one competitor.
4. **Defeater resistance** [NEC]
   - The strongest competitor-specific defeater does not apply.

These are expressed as NEC nodes. Additional EVID nodes are allowed but may not be used to inflate probability.

6.2 Template customization

Implementations may add slots, but must:

- apply the same slots to all named roots, and
- keep total slots small (4–7 recommended).

6.3 Why this matters

Without a template, decomposition can be weaponized: one hypothesis can be loaded with "universal feasibility" while rivals get only vague local stories. Template parity removes this asymmetry.

7. Semantics of p within trees ("No-free-probability")

ABDUCTIO MVP distinguishes **ledger probability** from **internal node p**:

- $p_{\text{ledger}}(H_i)$: MECE bookkeeping probability over roots.
- $p(\text{NEC node})$: a **requirement-satisfaction score** for the obligation slot, interpreted as: "How likely is it that this necessary condition is satisfied, given current evidence and assumptions?"

7.1 Neutral defaults

To prevent "conjunction crushing by listing," unassessed NEC nodes are neutral:

- NEC nodes initialize at $p = 1.0$ (neutral multiplier)
- with low confidence $k = 0.15$

EVID nodes may initialize at $p = 0.5$ (uninformative) and $k = 0.15$.

7.2 Consequence

Decomposition cannot lower a hypothesis merely by adding structure. Only evaluated requirements can reduce the multiplier.

8. Confidence k: Rubric and Mapping

Confidence k is the stability/robustness of a credence estimate under reasonable re-checking.

8.1 Rubric (0–2 each)

A: Evidence Traceability B: Cross-Validation C: Sensitivity to Assumptions D: Adversarial Resilience

Total $T = A + B + C + D$ maps to:

- 0–1 $\rightarrow$ 0.15
- 2–3 $\rightarrow$ 0.35
- 4–5 $\rightarrow$ 0.55
- 6–7 $\rightarrow$ 0.75
- 8 $\rightarrow$ 0.90

Guardrail: if any check = 0, cap $k \leq 0.55$.

8.2 Root confidence

Root confidence k_{root} is the minimum k over assessed NEC slots (conservative), optionally capped if UNSCOPED (Section 10).

9. Decomposition Rules

9.1 Root scoping is mandatory

All named roots must be decomposed into the obligation template before any root can be accepted as "well-scrutinized."

9.2 Additional decomposition within slots (optional)

Each slot node may be decomposed further (2–5 children) when its confidence is below threshold and credits remain.

9.3 Coupling for AND nodes (within a slot)

When decomposing a slot into an AND of NEC children, choose coupling $c \in \{0.20, 0.50, 0.80, 0.95\}$. Interpretation: pragmatic weight toward bottlenecking (min) vs independence (product), not a statistical coefficient.

Soft-AND for assessed children:

$$m = c \cdot p_{\min} + (1 - c) \cdot p_{\prod}$$

where $p_{\min}$ and $p_{\prod}$ are computed over assessed NEC children (unassessed treated as 1.0).

10. Anti-Vagueness (UNSCOPED rule)

A mechanism-like hypothesis must be able to state concrete necessary commitments.

Rule (root level)

If a named root cannot instantiate the obligation template with meaningful NEC statements, it is marked UNSCOPED and:

- cap $k_{\text{root}} \leq 0.40$,
- it remains in the evaluation schedule until it becomes SCOPED or credits exhaust.

Rule (slot level)

If a slot cannot be decomposed into at least 1 meaningful NEC statement, cap that slot's $k \leq 0.40$.

This prevents "winning by labels."

11. Aggregation: From obligations to root proposal

Let a root H_i have base ledger probability $p_{\text{base}} = p_{\text{ledger}}(H_i)$ at the time it is scoped (template instantiated).

For each NEC slot s, let its current satisfaction score be $p_s \in [0, 1]$. Compute a **root multiplier**:

$$m_i = \prod_{s \in \text{slots}} p_s$$

but crucially, because unassessed slots start at $p_s = 1.0$, this multiplier only decreases when a slot is actually evaluated and found wanting.

Then propose a new root probability:

$$p_{\text{prop}}(H_i) = \text{clip}(p_{\text{base}} \cdot m_i, 0, 1)$$

Notes

- This is intentionally conservative: penalties arise from discovered weaknesses, not from the mere existence of multiple requirements.
- Alternative within-slot AND aggregation can be used to compute each p_s; the above treats the template as a product across slots (because these are distinct necessary obligations).

12. Ledger Update with Other Absorber

Ledger updates must be stable and auditable.

12.1 Damping

After computing $p_{\text{prop}}(H_i)$ for any root, update:

$$p'_{\text{ledger}}(H_i) = (1-\alpha)\, p_{\text{ledger}}(H_i) + \alpha\, p_{\text{prop}}(H_i)$$

with $\alpha \in (0, 1]$ (default 0.4).

12.2 Other absorber invariant

Let $S = \sum_{i=1}^{n} p'_{\text{ledger}}(H_i)$ over named roots excluding Other.

- If $S \leq 1$: set $p_{\text{ledger}}(H_{\text{other}}) = 1 - S$.
- If $S > 1$: renormalize named roots only: $p_{\text{ledger}}(H_i) = p'_{\text{ledger}}(H_i)/S$, set $p_{\text{ledger}}(H_{\text{other}}) = 0$.

Log S and which branch was taken each time.

13. Scheduling: Deterministic, Seed-Invariant Credit Allocation

Permutation invariance requires that, given identical inputs, the same multiset of operations be performed regardless of input ordering.

ABDUCTIO MVP uses **cycle scheduling** over a **frontier** defined purely from the ledger state.

13.1 Frontier definition (no focal injection)

Let leader be $H_L = \arg\max p_{\text{ledger}}(H)$ over named roots. Frontier:

$$F = \{H_i : p_{\text{ledger}}(H_i) \geq p_{\text{ledger}}(H_L) - \varepsilon\}$$

with ε default 0.05.

No "seed" or user-focal term may be unioned into frontier. (The UI may highlight a focal, but the engine must not change scheduling.)

13.2 Round-robin credit slices

Within each cycle:

- iterate over hypotheses in frontier ordered by a canonical ID (e.g., SHA256 of root.statement),
- for each hypothesis, perform exactly one operation chosen by the deterministic rule below,
- decrement credits, update ledger, continue.

This round-robin is permutation-invariant because ordering is canonical, not input order.

13.3 Deterministic operation choice per hypothesis

For a root H_i in frontier, choose:

1. If H_i is UNSCOPED: DECOMPOSE (attempt to scope template or slot).
2. Else if any required slot is uninstantiated: DECOMPOSE to create missing slot node(s).
3. Else pick the slot s with lowest k (tie-break canonically by node ID):
   - If slot can be decomposed and $k < \tau$: DECOMPOSE(slot)
   - Else: EVALUATE(slot) or EVALUATE(slot's most critical child)

This ensures each frontier hypothesis is advanced comparably.

13.4 Tie-breaking (mandatory)

All ties are broken by canonical ID derived from the statement text (hash), never by input ordering.

14. Stopping Conditions

Stop when any holds:

A) credits exhausted. B) For all hypotheses in the current frontier:

- root is SCOPED,
- all template NEC slots have $k \geq \tau$ (or credit exhaustion prevents further improvement).

C) No legal next operation exists (e.g., maximum decomposition depth reached and no evaluable nodes remain).

15. Evaluator and Decomposer Interfaces (Implementation Contracts)

15.1 Evaluator contract (human or agent)

`evaluate(node, evidence)` returns:

- p in [0,1] (requirement-satisfaction for NEC; support score for EVID)
- rubric A–D scores and derived k
- 1–3 short factors
- $\text{mind}_{\text{change}}$ sentence
- explicit assumption list if evidence is weak, each tagged with fragility (low/med/high)
- $\text{evidence}_{\text{refs}}$ used (may be empty)

Constraints:

- If no $\text{evidence}_{\text{refs}}$: enforce conservative p movement (implementation default: $|\Delta\text{p}| <= 0.05$ from prior node.p)
- Evaluator must not reference "seed" or "focal" status.

15.2 Decomposer contract

`decompose(node)` returns:

- 2–5 children nodes, each labeled NEC or EVID
- $\text{decomp}_{\text{type}}$ AND/OR; for OR, mode EXCLUSIVE/INCLUSIVE
- for AND with NEC, coupling bucket c

Constraints:

- For root hypotheses, decomposer must instantiate the obligation template.

- If cannot, mark UNSCOPED.

16. MVP Algorithm (Pseudocode)

```
inputs:
  claim text
  (optional) rivals list
  credits B
  tau, epsilon, gamma, alpha
  canonical_id = hash(statement)

initialize:
  build named roots H1..Hn (claim + 3-5 single-mechanism rivals)
  add H_other
  set uniform priors: p_i=(1-gamma)/n ; p_other=gamma
  set all k=0.15
  set all status=UNSCOPED initially (until template instantiated)

for cycle = 1.. while credits > 0:
  leader = argmax named roots by p_ledger (tie-break by canonical_id)
  frontier F = {Hi : p_ledger(Hi) >= p_ledger(leader)-epsilon}

  order frontier by canonical_id
  for Hi in F:
    if credits == 0: break

    choose operation deterministically:
      if Hi is UNSCOPED or missing template slots:
         DECOMPOSE(Hi)  # instantiate template if possible
      else:
         pick slot s with lowest k (tie-break by canonical_id)
         if can_decompose(s) and k(s) < tau:
             DECOMPOSE(s)
         else:
             EVALUATE(s)

    spend 1 credit
    update credits_spent on Hi and node
    recompute slot p_s and k_s via aggregation if needed
    compute p_prop(Hi) = p_base(Hi) * Π_s p_s
    apply damping to ledger p_ledger(Hi)
    enforce Other absorber invariant
    log every arithmetic step and invariant checks

stop when stopping conditions met
output full audit trace
```

17. Why ABDUCTIO MVP is Permutation-Invariant

ABDUCTIO MVP achieves permutation invariance by construction:

1. No focal injection: frontier depends only on ledger state.
2. Canonical ordering: iteration order is defined by hash(statement), not by input list order.

3. Round-robin slicing: each frontier hypothesis receives equal opportunity per cycle.
4. No-free-probability semantics: decomposition cannot change ledger p; OR cannot inflate.
5. Template parity: each hypothesis is evaluated under the same obligation slots.
6. Deterministic tie-breaks everywhere.

Given identical inputs, the engine performs the same operations in the same canonical order and produces identical outputs, independent of the seed.

18. Practical Defaults

- tau = 0.70
- epsilon = 0.05
- gamma = 0.20
- alpha = 0.40
- $\text{max}_{\text{children}} = 5$
- $\text{coupling}_{\text{default}} = 0.80$
- conservative delta p when no evidence: $|\Delta\text{p}|$ <= 0.05 per evaluation

19. Limitations and Extensions

Limitations

- This is not full Bayesian inference; it is a structured, auditable scoring-and-budgeting framework.
- Results depend on the evaluator's discipline and evidence quality.
- MECE is approximated; H_{other} absorbs residual uncertainty.

Extensions

- Multi-assessor panels with aggregation rules for p and k.
- Evidence objects with explicit likelihood impacts.
- Specialized decomposers per domain (medicine, security, historical events).

Appendix A: Minimal "Well-Defined Hypothesis" Checklist

A root hypothesis must include:

- A mechanism statement (stand-alone).
- An exclusion clause distinguishing it from other roots.
- A template instantiation with NEC slots: feasibility, availability, fit, defeater resistance.

If it cannot, mark UNSCOPED and cap k.

Appendix B: Canonical ID

Canonical ordering uses:

$$\text{canonical\_id}(h) = \text{SHA256}(\text{normalized\_statement\_text})$$

Normalization removes extra whitespace and lowercases text.

This ensures permutation invariance even if input ordering changes.

Appendix C: Coupling buckets (within-slot AND)

Choose c as max of:

- Evidence overlap
- Mechanism overlap
- Failure-mode overlap

Buckets: 0.20, 0.50, 0.80, 0.95 Default 0.80 if unsure.