From Observable Answers to Executable Queries for CM/LM Programs

Brian Theory

8 November 2026 — Technical report, version 1.1

Abstract

An answer can be determined by an abstract observation without being obtainable from an available sensor library or executable measurement policy. This note separates these three requirements for finite Boolean programs. It connects static acquisition to weighted conflict covering, gives exact adaptive cost recurrences, and proves that paid addressing requires at least $b + 1$ queries on every input, attaining the same optimum under every prior. A CM/LM example shows why cancelled internal frames need not be observed; an outcome gate shows why knowing a prerequisite differs from executing it. A small independent verifier checks policy legality, outputs and costs. The contribution is an expository synthesis with executable examples of established methods.

1 Three questions about an observable answer

For a program's required output, ask: what information determines it, which available queries supply that information, and which query sequences are legal? These questions become distinct when inputs must be acquired at a cost. Tensor Logic's tensor-equation perspective motivates specifying the observable interface of a computation [2]. Correspondence and logical matrices (CM/LM) make Boolean frames and pairing explicit [1]. Neither notation alone supplies a measurement policy.

The underlying mathematics is established: functional dependency concerns whether an answer is a function of given observations [3]; relative discernibility and test-cost reduction underlie static selection [4, 5]; decision-tree certificates bound what a computation must learn [6]. This report connects these ideas through a common finite model, worked counterexamples and reproducible code. Its addressing calculation is an elementary certificate argument, not a claim of historical firstness. No CM-exclusive algorithmic advantage or integrated Tensor Logic backend is claimed.

2 Model and costs

Fix a nonempty finite feasible set Ω, required output $F : \Omega \to Y$, free initial information $\alpha : \Omega \to A$, and binary coordinate or sensor queries $s_j : \Omega \to \{0, 1\}$. Write m for the number of queries. Execution prices c_j and installation prices a_j are nonnegative rationals; nonnegative prior weights $p(x)$ sum to one. Exactness is required on every feasible state, including those with zero prior weight. Queries are deterministic, do not change the state, and occur at most once per execution path. Policies use only α and acquired readings. A correct leaf outputs $F(x)$ for every state reaching it. No extra action is required after the answer is known. Unless specified otherwise, all queries are available.

An abstract representation β suffices when $F = h \circ \beta$. In static acquisition, $\beta = (\alpha, (s_j)_{j \in J})$ for one globally fixed subset J chosen before α is revealed; every selected query is paid. Adaptive policies start with α available and choose subsequent queries from their readings. If $C_T(x)$ is path execution cost, the three objectives are

$$\max_x C_T(x), \qquad \sum_x p(x) C_T(x), \qquad \sum_{j \in U(T)} a_j,$$

where $U(T)$ is the union of queries used across all feasible paths and initial observations. Conditional static selection after learning α is a different comparator. Storage, decoding, calibration and communication are not priced.

The coarsest abstract repair is (α, F): its classes are the nonempty intersections of the original observation and target classes. Every refinement determining F must refine these intersections. This does not create a cheap sensor for F. Likewise, replacing F by an equal output leaves acquisition optima unchanged when states, sensors, costs, prior and legal actions are fixed.

3 Costed conflict covering

Let

$$E = \{\{x, y\} \subseteq \Omega : \alpha(x) = \alpha(y),\ F(x) \neq F(y)\}, \qquad D_{xy} = \{j : s_j(x) \neq s_j(y)\}.$$

Proposition 1 (Exact static acquisition)**.** *A subset J determines F given α iff $J \cap D_{xy} \neq \varnothing$ for every $\{x, y\} \in E$. Minimum static execution cost is therefore*

$$\min_{z \in \{0,1\}^m} \sum_j c_j z_j \quad \textit{subject to} \quad \sum_{j \in D_{xy}} z_j \geq 1 \quad (\{x, y\} \in E).$$

Without extra compatibility restrictions, a solution exists iff every D_{xy} is nonempty. Empty E permits $J = \varnothing$.

Proof. If a conflict pair is missed, the two states yield identical acquired readings and different answers, so no decoder works. Conversely, if every conflict is hit, no acquired-observation fiber contains unequal targets. Assign each attained reading its fiber's target value; values on unattained readings are arbitrary. This constructs a decoder. Taking the indicator of J gives the optimization. Selecting the complete library hits every nonempty D_{xy}; an empty one is hit by no subset. □

This is weighted set cover on conflicting pairs, or hitting set on the D_{xy}. Relative discernibility clauses are direct antecedents [4]; additive test-cost reduction is also established [5]. Unlike preservation of an inconsistent table's existing discrimination, total recovery must retain an empty conflict clause as infeasibility. An acquisition budget adds a price bound; compatibility restrictions add admissibility constraints. Full-library separation alone no longer suffices if the whole library cannot be jointly selected.

4 Exact adaptive acquisition

A history $h \in \{*, 0, 1\}^m$ records untested sensors and acquired values. Set

$$B(h) = \{x \in \Omega : s_j(x) = h_j \text{ whenever } h_j \neq *\}, \qquad q(B) = \sum_{x \in B} p(x).$$

For nonconstant α, restrict Ω to the initial observed fiber. Let $L(h)$ contain the legal untested sensors, and write $h[j = v]$ for extension. This history representation is sufficient for our outcome gate below. An order-sensitive protocol requires an ordered trace or sufficient automaton state.

Proposition 2 (Exact recurrences)**.** *Let $W(h)$ be minimum worst future execution cost and $E(h)$ minimum* unnormalized *expected future cost using the original prior. Both are zero if $B(h)$ is empty or F is constant on it. Otherwise define*

$$A(h) = \{j \in L(h) : W(h[j = v]) < \infty \text{ for every } v \text{ with } B(h[j = v]) \neq \varnothing\}.$$

If $A(h)$ is empty, both values are infinity. Otherwise

$$W(h) = \min_{j \in A(h)} \left(c_j + \max_{v : B(h[j=v]) \neq \varnothing} W(h[j = v]) \right),$$

$$E(h) = \min_{j \in A(h)} \left(q(B(h))c_j + \sum_{v : B(h[j=v]) \neq \varnothing} E(h[j = v]) \right).$$

Proof. Induct on the number of untested sensors. A correct leaf has constant target; unreachable branches impose no obligations. Every nonterminal legal policy first selects an allowed test and must attach a feasible policy to every nonempty outcome branch. Thus its first choice belongs to $A(h)$. Execution pays c_j on every remaining feasible path; worst costs maximize child costs, whereas expectation contributes $q(B)c_j$ and sums the children's unnormalized expectations. Induction bounds every child's cost by its optimum. Conversely, choose a minimizing action and attach separately minimizing child policies for the objective in question. Their composition attains the displayed expression. There are finitely many tests and actions. If no feasible first action exists, no correct nonterminal tree exists. Each extension removes a test, so this induction also covers zero prices without a cyclic optimality equation. □

The two minima may require different trees. Store separate argmins when extracting policies. At positive mass, $E(h)/q(B(h))$ is the conditional optimum; at zero mass no division is needed. A nonempty impossible child is disallowed even when its mass is zero: exact recovery is required on all feasible states. We never use $0 \cdot \infty$. A feasible zero-mass subtree has zero expected contribution. For free initial observations, aggregate fiber values by maximum for worst cost and sum for unnormalized expectation. Installed unions couple the fiber policies.

Proposition 3 (Installation under unrestricted availability)**.** *When tests are always available, minimum installed price of an exact policy equals minimum static sufficient-subset price with coefficients a_j.*

Proof. Let $U(T)$ be the sensors occurring anywhere in an exact policy. Two states with the same α and identical readings on $U(T)$ follow the same tree path, since each next action depends only on readings already matched. Hence their targets agree and $U(T)$ is a sufficient subset. Its installed cost is exactly the price of that subset. Conversely, test any sufficient subset in a fixed order and decode the attained signature; all tests are available and this policy uses only that subset. The two inequalities prove equality. □

5 A paid-address separation

Proposition 4. *For each $b \geq 0$, let $k = 2^b$ and let states range over all assignments to b address bits a and k data bits $d_0, \ldots, d_{k-1}$. Sensors read individual bits, all execution and installation prices*

are one, α is constant, and $F(a, d) = d_{\mathrm{index}(a)}$. The static and installed optima are $b + k$; every correct policy uses at least $b + 1$ queries on every input. Reading the address and then its selected data bit uses exactly $b + 1$ queries on every input. Thus both the adaptive worst optimum and the expected optimum under every fixed prior are exactly $b + 1$, including priors with zero-mass feasible states.

Proof. Each data coordinate is essential by setting the address to its index and changing that bit alone. Each address coordinate is essential by choosing two addresses differing at that coordinate and assigning their selected data bits different values. Omitting any sensor therefore leaves a conflict pair. All sensors together suffice, proving the static claim; Proposition 3 proves the installed claim.

Fix any terminating feasible transcript ending at a correct leaf with answer v, and consider the coordinate values acquired along that particular path. Every full input agreeing with those values follows the same transcript: inductively the deterministic policy sees the same earlier readings and chooses the same next test. Thus the consistent inputs form the entire subcube specified by the distinct coordinates read on that path.

Suppose r address coordinates remain unread. Exactly 2^r addresses are compatible with the transcript. For each such address i, coordinate d_i must have been read and must equal v. Otherwise, if it is unread, complete the address to i and set $d_i = 1 - v$, retaining every acquired value. If it was read with a different value, simply complete the address to i. In either case the full feasible cube supplies an input following this leaf but having target $1 - v$, a contradiction. Conversely, if every compatible addressed coordinate has been read with the same value, the target is constant on that subcube.

Consequently the path has read at least

$$(b - r) + 2^r \geq b + 1$$

distinct coordinates, since $2^r \geq r + 1$ for every integer $r \geq 0$ (by induction). For $b = 0$ this requires the sole data coordinate, so the same proof applies. Additional irrelevant tests cannot reduce this count. Although our model uses each test at most once, allowing repeated queries at unit price cannot weaken the lower bound: total queries are at least distinct queried coordinates.

Reading all address bits and then the selected data bit attains $b + 1$ on every input. Therefore it is simultaneously optimal at every input. For any fixed prior p, summing the pointwise lower bound gives $\sum_x p(x)C_T(x) \geq (b+1)\sum_x p(x) = b+1$, attained by the same policy. No full-support assumption on p is needed. □

This is the ordinary decision-tree certificate argument: a leaf's observations certify its answer, and here every input has certificate complexity $b + 1$. The standard pointwise certificate and strict instance-optimality definitions are given in [6, Definitions 2.2 and 2.5]. The derivation above specializes those established notions; no historical-firstness claim is made.

The full feasible cube is essential. A prior changes weights, not which inputs require exact answers. A promise changes feasibility: for $b = 1$, restricting to $d_0 = d_1$ permits reading only d_0, at cost 1, even with both addresses possible. Concentrating a prior on those same inputs while keeping the entire cube feasible still requires cost at least 2 on every path of an exact policy. Restricting feasibility to a single known input even permits zero tests.

The static-to-worst and static-to-optimal-expected ratios are both exactly $(k + b)/(b + 1)$ for every prior, and diverge with b. The static subset here is globally fixed and all its readings are paid; it must be sufficient on the full cube. Supplying the address free changes that comparator: conditional static selection can then choose its single relevant data coordinate. Composite sensors, bounded-error answers and nonunit prices are outside this proposition.

6 A worked CM/LM interface and its Boolean baseline

Formula-valued matrices here multiply over the Boolean ring: addition is XOR, multiplication is AND. This is not existential OR–AND relational composition or ordinary real matrix multiplication. Define the true-first state vector $|u\rangle = (u, \neg u)^T$ and frame

$$S_g = \begin{pmatrix} g & \neg g \\ \neg g & g \end{pmatrix}, \qquad C_\wedge = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

Disjointness $g \wedge \neg g = 0$ and $g \oplus \neg g = 1$ give $S_g^2 = I$. The logical-matrix normal form for conjunction is

$$M_{p\wedge q} = S_p C_\wedge S_q = \begin{pmatrix} pq & p\neg q \\ \neg p\, q & \neg p\, \neg q \end{pmatrix}.$$

Here pq abbreviates $p \wedge q$, not a probabilistic independence approximation. By the one-hot disjoint selector calculation,

$$\langle u | M_{p\wedge q} | v \rangle = (u \leftrightarrow p) \wedge (v \leftrightarrow q).$$

For example, $u = p, v = q$ selects truth value 1, whereas $u = v = 1$ selects $p \wedge q$. The upper-left entry and reference-frame pairing are different readouts.

Now take states $(p, g) \in \{0, 1\}^2$, sensors p, g with prices 1,100, and export

$$F(p, g) = (p,\ \langle 1 | S_g I S_g | 1 \rangle).$$

Using associativity in this ring, $S_g I S_g = I$ and the pairing equals 1, so $F(p, g) = (p, 1)$. This is the matched-frame cancellation law [1] in a fully specified interface. The ordinary Boolean formulation exports exactly $(p, 1)$; it requires sensor p at cost 1 and never needs g. The two states with $p = 0$ and $p = 1$ force acquiring p; hence 1 is optimal for both descriptions. If the intermediate matrix S_g is also exported, its entries determine g, so both sensors are needed at price 101. That is a stronger contract, not a frame-derived speedup. If a fixed workflow mandates reading g regardless of exports, eliminating it requires permission to change that workflow.

CM/LM adds an explicit account of polarity, pairing and allowed cancellation. It does not add information to the sensors, alter the true optimum for unchanged exports, or certify a learned numerical embedding. The selector's disjoint weights permit XOR and OR to agree for this pairing; arbitrary multiwitness aggregation does not share that property. Nor may an internal frame be canceled through an arbitrary threshold or pointwise nonlinearity without another proof.

7 Sufficiency can fail to be executable

Consider sensors g, t with execution prices $0, 1$, where t is available only after g has actually been tested and returned 1. On the full four-state cube $(g, t) \in \{0, 1\}^2$, target t is statically determined by the single sensor t. Nevertheless no exact legal policy exists: the only possible first test is g, and its zero branch leaves both target values possible with no legal remaining query. Giving that branch zero prior probability does not remove its correctness obligation.

Now restrict feasibility to $(1, 0), (1, 1)$. Querying g then t is legal and costs 1. Every correct policy must acquire t, so this cost is optimal. Although g conveys no new information, its execution unlocks t. The belief set before and after g is the same; legal actions differ. Thus discarding all non-splitting tests or recording only the belief set can destroy feasibility.

This outcome gate is stronger than a fixed precedence constraint, which requires earlier tests irrespective of their answers [7]. No approximation guarantee for that different model is imported. More general chronological protocols require sufficient ordered history or an explicit protocol state.

8 Checking a proposed policy

The supplied verifier takes trusted explicit sensor/target tables, prices, prior and prerequisite rules. Its certificate gives one root per attained initial observation, test nodes with outcome branches, declared leaf answers, and exact claims for worst, expected and installed costs. Prerequisites are conjunctions of previously executed query outcomes. This restricted format does not implement arbitrary history-dependent legal-action predicates.

For each feasible state, including every zero-prior state, the verifier follows the proposed policy and checks prerequisites, branches and leaf answers. It rejects repeated tests or cycles along a feasible execution path. Shared nodes are checked under each reaching history; global graph acyclicity is not required. The canonical serialization contains exactly the nodes and edges used by feasible states. Unreachable branches must be omitted. A digest binds canonical model JSON, not the authenticity or physical adequacy of those tables.

Acceptance certifies legal finite execution, exact answers and the three claimed costs within this format. To see this, enumerate all feasible paths: their individual price sums determine maximum and prior-weighted costs, and their union of query labels determines installed cost. Every operational node and edge is covered by the canonical check. Conversely, a legal canonical policy with truthful answers and costs passes these checks. No comparison with other policies occurs, so optimality is not certified. A retained fixture deliberately accepts a correct three-query multiplexer policy although two queries suffice.

9 Reproducibility and scope

The accompanying Python 3.10+ package uses only the standard library. From the package root, run:

```
python -B code/run_checks.py
```

This reproduces static selection, adaptive acquisition, matrix identities, policy verification and the addressing certificate checks in temporary directories. Machine-readable expected results are included. The README explains model formats, commands, provenance and the distinction between validity and optimality.

The retained policy suite has five accepted fixtures and nineteen rejected corruptions, including forged costs, missing branches and zero-prior failures. The addressing audit exhausts 177,906 partial assignments for $b = 0, 1, 2, 3$ and also checks sampled adaptive paths with repeated queries. These finite checks support the proofs; they do not prove the all-b statement. Full counts and source hashes are supplied with the code, outside the report's narrative.

Explicit conflict construction costs $O(|\Omega|^2 m)$ comparisons. Exhaustive subset optimization is exponential in m; observed-outcome dynamic programming has at most 3^m records per initial observation fiber, plus legal-action evaluation, protocol-state and exact-arithmetic costs. The verifier traverses at most $|\Omega|m$ test steps, plus schema, prerequisite and arithmetic work. Succinct encodings can have large intermediates; none of these bounds proves a scalable tensor algorithm.

The examples distinguish semantic sufficiency, affordable acquisition and legal execution. The CM/LM frame and pairing structure describes the output contract; identical Boolean outputs with identical access assumptions have identical acquisition optima. The report's value is this usable synthesis, not a new learning result or exclusive computational advantage. No noisy sensing, numerical-embedding guarantee, physical deployment or performance benchmark is established.

Preparation and review. AI tools assisted with drafting, code and internal mathematical review. The proofs and exact tests are available for inspection; internal AI reviews are distinct from human peer review or proof-assistant verification. Compilation and page-inspection details accompany this edition.

References

[1] Brian Theory. *Correspondence and Logical Matrices: A Typed Boolean Operator Calculus.* September 27, 2026 revision, public reading copy 1. Sections 3.2, 4 and 5.1; Theorems 3.1, 4.1 and 5.4. `https://www.b-theory.com/manuscripts-html/correspondence-matrices/`. Accessed 4 November 2026.

[2] P. Domingos. *Tensor Logic: The Language of AI.* arXiv:2510.12269v3. `https://arxiv.org/html/2510.12269v3`.

[3] J.-H. R. Jiang, C.-C. Lee, A. Mishchenko and C.-Y. Huang. To SAT or Not to SAT: Scalable Exploration of Functional Dependency. *IEEE Transactions on Computers* 59(4), 457–467, 2010. `https://people.eecs.berkeley.edu/~alanmi/publications/2010/tc10_sat.pdf`.

[4] P. Hońko. Attribute reduction: a horizontal data decomposition approach. *Soft Computing* 20, 951–966, 2016 (online 2014). `https://doi.org/10.1007/s00500-014-1554-8`.

[5] F. Min, Q. Hu and W. Zhu. *Feature selection with test cost constraint.* arXiv:1209.5601, 2012. Section 2.2. `https://arxiv.org/pdf/1209.5601`.

[6] T. Grossman, I. Komargodski and M. Naor. Instance Complexity and Unlabeled Certificates in the Decision Tree Model. *LIPIcs, ITCS 2020*, 151:56:1–56:38, 2020. `https://doi.org/10.4230/LIPIcs.ITCS.2020.56`.

[7] M. Szyfelbein and D. Dereniowski. *Precedence-Constrained Decision Trees and Coverings.* arXiv:2602.21312v4, 12 July 2026. `https://arxiv.org/html/2602.21312v4`.