# Consequence Closure

## Semantic Assurance at the Machine Action Boundary

Moon Lee · RISU Institute · August 2026

DOI: 10.5281/zenodo.22095709

**ABSTRACT**

Machine systems can correctly record authority, evidence, or control state while the consequence of a concrete action still depends on unresolved system state. Consequence Closure makes that residual dependence an assurance target. At a declared action cut, it asks whether all admissible realizations consistent with current evidence yield the same specified consequence. When they do not, relational counterexamples expose distinctions that remain material. Over a finite candidate vocabulary, the resulting distinguishing sets form a materiality hypergraph whose inclusion-minimal transversals are exactly the sufficient semantic obligation sets. A finite CEGIS procedure either returns one such set or certifies that no sufficient set exists within the declared vocabulary. Obligations remain separate from realizable Establishments (observations or interventions) and from Routes that use them to reach closure. To prevent source abstractions from preserving final decisions while corrupting partial-state assurance, the note defines P0/P1/P2 preservation levels for closure, obligations, and Routes. A preregistered differential against Cedar 4.12.0 confirmed exact P1 over the frozen bounded surface with zero post-outcome semantic repair; separate Linux, SQLite, filesystem, and OAuth commissioning tests probe operative and evidence semantics at real action cuts. An inspectability contract carries results, witnesses, replay identity, and qualifications without strengthening the underlying claim.

**Keywords:** formal methods; runtime assurance; authorization; revocation; partial information; semantic obligations; runtime enforcement; autonomous software

# 1. Introduction: When Control State and Action Consequence Diverge

A system can correctly record a control change while an already established action path remains effective.

In the Linux commissioning reported in Section 5, a parent process changed a memfd's mode to 0444 and verified that mode with fstat before a child process attempted pwrite through an inherited writable descriptor. The write succeeded in all 200 trials. In a second condition, the parent established F_SEAL_WRITE before the child attempted the write; all 200 attempts failed with EPERM. Linux specifies that file seals are enforced immediately when successfully added and that F_SEAL_WRITE prevents modification of file contents [1]. The two conditions expose a concrete boundary between recorded control state and the effect still available through an established action path.

The same structure appears in OAuth 2.0. RFC 7009 specifies immediate token invalidation while acknowledging propagation delay in which some servers may know about the invalidation before others [2]. RFC 7662 permits protected resources to cache introspection responses and warns that a revoked token can remain usable during the resulting stale-information window [3]. At the moment a protected request is evaluated, authority-side evidence and resource-side behavior can therefore diverge in a way that changes whether the request succeeds.

These cases share a semantic condition. Current evidence can be compatible with multiple realizations of the relevant system, and those realizations can produce different consequences for the action under consideration. Long-lived capabilities, cached authority state, asynchronous propagation, distributed enforcement, external tools, and partial observation create recurring ways for that condition to arise. When it does, the action consequence cannot be determined from the current evidence alone.

This note studies that condition at a declared action cut:

> *Given the evidence available at an action cut, what can still change the specified consequence, and what can the system establish to eliminate that dependence before acting?*

**Consequence Closure** holds when, under a qualified semantic boundary, every admissible realization consistent with current evidence yields the same specified consequence for the action. If compatible realizations still yield different consequences, the state is OPEN. A pair of such realizations identifies a material distinction that remains unresolved. The reference verifier preserves the pair as a witness, and the synthesis layer uses accumulated witnesses to derive the semantic obligations required for closure.

The note makes four contributions. First, it defines a small carrier-neutral semantic target for consequence determinacy at a declared action cut. Second, it gives a reference method that converts relational counterexamples into inclusion-minimal semantic obligations while keeping those obligations separate from the realizable Establishments available to close the gap. Third, it gives source front ends a claim-indexed P0/P1/P2 preservation hierarchy and subjects P0/P1 to a frozen external-oracle differential against the official Cedar runtime. Fourth, it connects the model-relative analysis to real-system commissioning and an inspectability contract that preserves witnesses, qualifications, and replay identity without amplifying the underlying assurance claim.

The target is intentionally narrow. Authorization languages, planners, enforcement architectures, and runtime monitors continue to supply their own semantics and mechanisms. Consequence Closure asks whether the evidence and declared Establishments available at one concrete action boundary are sufficient to make one specified consequence invariant. The formal claims are qualified to the declared boundary, and the empirical claims are qualified to the commissioned paths reported later in the note.


**Figure 1.** The action cut. The Core evaluates one specified action consequence within a declared boundary using the evidence and establishments available at the cut.

# 2. Foundations and Positioning

Consequence Closure sits at the intersection of several established lines of work. Access control and usage control separate decisions from enforcement and treat authority as continuing state. Planning under partial information distinguishes the world from what an agent knows and provides methods for acting across multiple possible states. Runtime enforcement changes behavior at execution time. Relational verification and abstraction provide machinery for comparing executions and preserving semantics across representations. Within that setting, the Core fixes a specific target: consequence determinacy at a declared action cut.

## 2.1 Authority, enforcement, and continuing control

Saltzer and Schroeder's complete mediation principle requires every access to be checked for authority and cautions against remembered authorization results unless they are updated when authority changes [4]. XACML makes a related architectural separation explicit: a Policy Decision Point evaluates policy and renders an authorization decision, while a Policy Enforcement Point makes decision requests and enforces the resulting decision [5]. Park and Sandhu's UCONABC model extends access control with authorizations, obligations, conditions, decision continuity, and attribute mutability [6]. Capability systems have also treated revocation of previously issued authority as a concrete systems problem; Aguilera and colleagues designed a capability group mechanism for revoking previously issued capabilities in network attached storage [7]. Dean and Hu analyze a different temporal hazard in the classic time of check to time of use race between an access check and a later use [19].

These results cover several mechanisms by which control state and action behavior can separate. The semantic pattern extends beyond interleaving races: previously issued authority can remain effective after an administrative update, an authority change can propagate unevenly, or a protected-resource can act on cached state. At the action cut, Consequence Closure evaluates the joint result of decision state, evidence, enforcement state, and the concrete action path by asking whether any currently admissible realizations still disagree on the specified consequence.

Terminology is important here. In usage-control and policy systems, an obligation commonly denotes an action or duty associated with a policy decision [5], [6]. This note's semantic obligation is different: it is a set of declared semantic discriminators sufficient to determine the specified consequence. Realizable actions that acquire information or change the operative state are represented separately as Establishments.

## 2.2 Partial information and operative control

Possible-world epistemic models already give the basic information-set reading behind closure: a fact is known at a point when it holds throughout the points indistinguishable under the available information [34]. When E is treated as an information state, CLOSED(c) has the analogous form that the multi-valued consequence $K_a$ is constant over every realization compatible with E. Knowing-value logics go closer still: Baltag formalizes knowing the value of a term

conditional on the values of other terms [35]. Consequence Closure therefore does not claim novelty for conditional value determinacy or for a new epistemic modality at this kernel.

Son and Baral distinguish the state of the world from an agent's state of knowledge and formalize sensing actions that update knowledge under incomplete information [8]. Bertoli and colleagues study strong planning under partial observability, seeking conditional plans that succeed across multiple initial states, nondeterministic action effects, and incomplete observations [9]. Jiang, Kumar, and Garcia study optimal sensor selection for partially observed discrete-event systems, asking for sufficient yet minimal observation information that preserves a desired formal property [31]. Bulychev and colleagues formulate the cost-minimal observation problem directly: compute a sufficient subset of observable predicates whose total cost is minimal [10]. Together these results ground evidence refinement, contingent sensing, and the search for sufficient observation surfaces.

Runtime enforcement supplies the complementary operational basis. Edit automata monitor and transform action streams through termination, suppression, or insertion [11]. A closure procedure can therefore resolve uncertainty by acquiring information, or alter reachable consequences through an operative mechanism. The Core keeps those roles separate: semantic obligations identify what still matters, while Establishments identify available operations that can observe, change, guard, transact, or suppress. Section 3 formalizes that separation.

### 2.3 Determinacy, reducts, and sufficient information

Several mature literatures already contain the mathematical kernel of Section 3's sufficiency relation. In relational database theory, a functional dependency $X \rightarrow Y$ holds when any two tuples that agree on X also agree on Y [27]. If a finite analysis set S is written as a relation whose columns are the declared Propositions P together with the consequence $K_a$, then Q is sufficient over S exactly when that relation satisfies $Q \rightarrow K_a$. Database view determinacy generalizes the same information-theoretic form: a set of views determines a target query when any two database instances indistinguishable by the views have the same target-query answer [28]. Consequence Closure therefore does not claim novelty for the bare implication from equal selected information to equal output, or for determinacy as an information-theoretic notion.

Rough-set attribute reduction makes the minimality connection still closer. Pawlak's reducts remove dispensable attributes while preserving a declared classification property [29]. Skowron and Rauszer's discernibility matrix and discernibility function associate pairs of objects with the attributes that distinguish them and derive reducts from minimal combinations of those distinctions [30]. On a finite realization table whose decision attribute is $K_a$, the Section 3.6 sets $D(r_1,r_2)$ for consequence-divergent pairs are decision-relative discernibility sets. The resulting exact hitting-set characterization and inclusion-minimal transversals are thus not presented here as new combinatorics; the note reuses that reduction pattern for a different semantic and operational purpose.

Formal explanation work provides another nearby notion of sufficiency. Sufficient reasons for Boolean classifier decisions and abduction-based explanations seek compact feature sets whose fixed values force a prediction [32], [33], and domain constraints can change which such reasons are sufficient [15]. These are close precedents for compact sufficiency, but their usual object is a local explanation of a classifier decision. Here the selected Q is a semantic discriminator vocabulary over a declared set of source-realizable states or histories; the current-evidence query may restrict that set, but Q is not itself a post-hoc assignment chosen to explain one realized output.

The contribution lies in how these foundations are composed around a machine-action assurance boundary. Consequence Closure fixes an action cut and qualified Boundary, retains divergent pairs as replayable Witnesses, separates semantic obligations from realizable Establishments, permits contingent Routes that can observe or change the world, and makes exported claims depend on a claim-indexed P0/P1/P2 source-preservation contract. Section 5 then asks the separate empirical question of whether selected operative mechanisms behave as declared at real action cuts.

### 2.4 Relational verification and source preservation

Dijkstra's predicate transformer calculus provides an earlier formal lineage for deriving conditions sufficient to establish a postcondition [20]. Here the sufficiency target is relational: a selected semantic surface is adequate when any two admissible realizations that agree on it also agree on the specified consequence. Hyperproperties characterize properties over sets of traces [12], and Terauchi and Aiken use self-composition to reduce secure information flow questions to safety verification over paired executions [13]. The reference closure verifier follows this broad pattern

by pairing bounded admissible histories, imposing equality on selected information, and searching for a difference in the specified consequence.

Source compilation raises a second correctness obligation. Ranzato and Tapparo study strong preservation, where abstract and concrete models agree on the formulas of a specification language [14]. Gorji and Rubin show that domain constraints can change the structure of sufficient reasons when impossible feature combinations would otherwise be admitted [15]. These results motivate the front end contract in Section 4. Complete-world decision agreement can coexist with different closure, minimal-obligation, and route structure at partial evidence states, so source correlations and establishment semantics must remain available where those outputs are computed.

**Table 1.** Established foundations used by the Consequence Closure argument.

| Research line | Established result | Use in this note |
|---|---|---|
| Complete mediation and PDP/PEP architectures | Authority checks and decision enforcement are distinct control responsibilities | Locates the closure question at the concrete action boundary |
| UCON, capability revocation, and TOCTOU | Continuing control, revocation, mutability, and check-use races are established system concerns | Grounds the temporal and operative authority problem |
| Epistemic, partial-observation, and minimal-observation models | Knowledge can be modeled over indistinguishable worlds; control can be synthesized across multiple possible states; sufficient observation sets can be minimized | Grounds the information-set reading of closure, evidence refinement, and observation minimization without treating those foundations as new |
| Database dependencies, view determinacy, reducts, and sufficient reasons | Selected information can determine a target; inclusion-minimal attribute sets can preserve a decision; constraints can change sufficiency | Treats determinacy and minimal reduction as foundations, then adds action-cut semantics, Establishments and Routes, claim-indexed preservation, and commissioning |
| Runtime enforcement | Execution can be terminated, suppressed, or modified at runtime | Grounds operative Establishments |
| Predicate transformers and relational verification | Sufficiency can be stated relative to a postcondition and relational properties can compare executions | Grounds consequence-relative sufficiency and the paired closure check |
| Strong preservation and constrained reasoning | Abstractions and domain constraints affect preserved semantics and sufficient reasons | Grounds partial-state, joint-realizability, and source-preservation requirements |

Taken together, these literatures supply the mathematical and operational ingredients used here. The remaining design problem is to bind them to one declared action consequence without losing source-realizable correlations, conflating semantic obligations with mechanisms, or exporting a stronger claim than the front end preserves. When the state remains OPEN, the verifier also retains the divergent pair that shows what remains material. Section 3 defines that composite target.

# 3. Consequence Closure

## 3.1 Closure at an action cut

For one closure problem, let a Boundary B define a finite qualified set of admissible realizations $R_B$. A realization may be a completed state or a bounded history, according to the external system. Let E denote the evidence available at the current cut. The compatible realization set is

$$\mathrm{Compat}_B(\mathrm{E}) = \{\ \mathrm{r} \in R_B \mid \mathrm{r} \text{ is consistent with } \mathrm{E}\ \}$$

For a declared action a, let the specified consequence function be

$$K_a : R_B \to \mathrm{C}$$

where C is the declared finite consequence codomain for the decision being analyzed. The consequence is action-specific. Carrier-specific detail remains in the external model and its front end; the Core retains the distinctions that can change $K_a$.

Write $K_a(\mathrm{S}) = \{\ K_a(\mathrm{r}) \mid \mathrm{r} \in \mathrm{S}\ \}$ for the consequence values reachable from a realization set S. The closure status at evidence state E is:

$$\text{IMPOSSIBLE if } \mathrm{Compat}_B(E) = \varnothing.$$
$$\text{OPEN if } |K_a(\mathrm{Compat}_B(E))| > 1.$$
$$\text{CLOSED(c) if } K_a(\mathrm{Compat}_B(E)) = \{c\}.$$

Closure is therefore relative to the declared Boundary, current evidence, and specified consequence. CLOSED(c) records determinacy even when c is adverse, denied, error, or another declared value; downstream execution policy interprets the value. OPEN means that some pair $r_1$ and $r_2$ in $\mathrm{Compat}_B(E)$ still yields different consequence values. Such a pair is the basic materiality witness used below.

Cut adequacy is part of the Boundary. An action-level closure claim is warranted only when every source transition between the evidence cut and the realization of the specified consequence that can change $K_a$ is represented in $R_B$ (for example as part of a bounded history) or is excluded by a source-grounded assumption or operative Establishment. If a consequence-relevant post-cut transition is omitted, CLOSED(c) is only snapshot-relative and must not be exported as a guarantee about the later action effect. The verifier cannot infer this adequacy from the compiled model itself.

## 3.2 Reference semantic objects

Consequence Closure Core v0.1.0 uses six top-level semantic objects as its compilation target.

**Table 2.** Six object reference vocabulary of Consequence Closure Core v0.1.0.

| Object | Role |
|---|---|
| **Boundary** | Declares admissible realizations, joint-realizability constraints and assumptions, the evaluation cut, current information state, and the available Establishment surface |
| **Consequence** | Maps each admissible realization to the action-relevant outcome |
| **Proposition** | Represents a carrier-neutral semantic discriminator whose value can affect the specified consequence |
| **Establishment** | Represents a realizable observation or intervention with declared information or transition semantics |
| **Route** | Represents a realizable finite strategy over Establishments for reaching closure |
| **Witness** | Preserves replayable evidence supporting materiality, sufficiency, derivation, front-end preservation, or commissioning |

Front ends can retain carrier syntax, source-specific typing, identity rules, and provenance while compiling consequence-relevant structure into these objects. The stable Core interface is then shared by closure verification, obligation synthesis, route analysis, and witness replay.

## 3.3 Declaration integrity

Closure claims are meaningful only relative to declarations fixed independently of the result they justify. Before verification or route synthesis, an analysis instance fixes (i) Boundary B, including admissible realizations, joint-realizability constraints, the evaluation cut, and current evidence; (ii) the specified Consequence $K_a$ and codomain C; (iii) the candidate Proposition surface P with source derivation; and (iv) the Establishment surface with preconditions and outcome or transition semantics. Every output is scoped to that declaration version.

Result-guided refinement creates a new analysis instance. A returned witness may motivate a new source-grounded candidate, but that candidate cannot retroactively become a premise of the result that exposed it. A proposition may determine $K_a$ when independently grounded in source semantics or evidence; circular construction from $K_a$ or a closure result merely to secure a preferred claim is excluded. Establishment semantics likewise cannot be revised after commissioning and treated as premises of the original selection. Changing any declaration requires rerunning the relevant analysis. The Core therefore synthesizes within a declared problem, not its target or mechanism semantics.

## 3.4 Materiality and sufficiency

Let P be the declared candidate proposition surface. Each $q \in P$ has a declared value in each admissible realization. For $Q \subseteq P$, define observational equivalence on Q by

$$\mathrm{Eq}_Q(r_1, r_2) \Leftrightarrow q(r_1) = q(r_2) \text{ for every } q \in Q.$$

An obligation set Q is sufficient over a realization set S when

$$\forall\, r_1, r_2 \in S:\ \mathrm{Eq}_Q(r_1, r_2) \Rightarrow K_a(r_1) = K_a(r_2).$$

For finite S, this predicate is exactly the functional dependency $Q \rightarrow K_a$ on the relation induced by S [27]; the database view-determinacy literature gives the same information-theoretic implication for more general views [28]. Its role here is therefore not to introduce a new notion of determinacy, but to make determinacy the obligation criterion inside the action-cut semantics.

The frozen global obligation profile uses $S = R_B$. A current cut residual query uses $S = \mathrm{Compat}_B(E)$, which asks which additional distinctions remain after the evidence already available at that cut.

For a fixed Q, the reference verifier searches for a counterexample to sufficiency with the paired formula

$$r_1 \in S \wedge r_2 \in S \wedge \mathrm{Eq}_Q(r_1, r_2) \wedge K_a(r_1) \neq K_a(r_2).$$

This paired form follows the relational verification pattern summarized in Section 2.4 [12], [13]. The formula is exactly the negation of the universal sufficiency condition above. SAT returns a pair $(r_1, r_2)$ that remains indistinguishable under Q while producing different consequences. UNSAT establishes sufficiency for Q within the declared analysis set S.

The UNSAT result establishes model-relative sufficiency for Q. Its external force depends on the qualified Boundary and the front end that produced the model; Section 4 states the preservation conditions required at that interface.

| **REALIZATION A** | same selected information | **REALIZATION B** |
|---|---|---|
| $K_a(r_1) = c_1$ | $\mathrm{Eq}_Q(r_1, r_2)$<br>$c_1 \neq c_2$ | $K_a(r_2) = c_2$ |

**Figure 2.** Materiality witness. A satisfiable fixed-Q query returns two admissible realizations that agree on Q and disagree on the specified consequence.

### 3.5 Structural laws of closure and sufficiency

The definitions above imply several structural laws that are useful for interpreting closure claims and for auditing changes to the declared problem. They apply to any fixed analysis set S and do not depend on how that set was represented or obtained. Their external force remains conditional on the source-preservation contract in Section 4.

**Proposition 1 (restriction laws).** Let $S' \subseteq S$. If $K_a(S) = \{c\}$, then $S'$ is either empty or $K_a(S') = \{c\}$. Consequently, restricting the compatible realization set cannot turn CLOSED(c) into OPEN: it can only preserve CLOSED(c) or yield IMPOSSIBLE. If Q is sufficient over S, then Q is sufficient over $S'$. If Q is sufficient over S and $Q \subseteq Q' \subseteq P$, then $Q'$ is also sufficient over S.

**Proof.** The image of a subset under $K_a$ is a subset of the original image. For sufficiency under $S'$, every pair in $S'$ is already a pair in S. For sufficiency under a proposition superset, $\mathrm{Eq}_{Q'}(r_1, r_2)$ implies $\mathrm{Eq}_Q(r_1, r_2)$.

For evidence refinement this yields a precise monotonicity statement. Whenever stronger evidence $E'$ only removes compatible realizations, $\mathrm{Compat}_B(E') \subseteq \mathrm{Compat}_B(E)$. A previously closed consequence cannot be reopened by that refinement, and every previously sufficient Q remains sufficient. The family of sufficient sets can expand as evidence removes material pairs, so the family of inclusion-minimal obligations may change; no stronger monotonicity claim about the particular minimal basis is assumed.

**Proposition 2 (realization-set extension and consequence granularity).** Let $S \subseteq T$ with the same specified $K_a$. If S is OPEN, then T is OPEN. If S is CLOSED(c), T may remain CLOSED(c) or become OPEN depending on the added realizations. Thus enlarging the admissible realization set can preserve or weaken closure assurance, but cannot erase an already present divergent pair.

For consequence granularity, suppose $K_{\mathrm{fine}}$ and $K_{\mathrm{coarse}}$ share the same realization domain and a declared map g satisfies

$$K_{\mathrm{coarse}}(r) = g(K_{\mathrm{fine}}(r)) \quad \text{for every } r.$$

Then CLOSED under $K_{\mathrm{fine}}$ implies CLOSED under $K_{\mathrm{coarse}}$, OPEN under $K_{\mathrm{coarse}}$ implies OPEN under $K_{\mathrm{fine}}$, and every Q sufficient for $K_{\mathrm{fine}}$ is sufficient for $K_{\mathrm{coarse}}$.

Coarsening may merge outcomes that the finer consequence distinguishes, so closure under a coarse codomain does not establish closure under a finer one. Boundary comparison is meaningful only when the realization relation and consequence-refinement map are stated rather than inferred from labels such as broader or narrower.

**Proposition 3 (Establishment-surface invariance at the current cut).** Suppose the current realization set S, consequence $K_a$, candidate surface P, and evidence are fixed, and the declaration is changed only by adding Establishments without changing the semantics of existing ones. Then the current-cut closure status and the sufficiency predicate for every $Q \subseteq P$ are unchanged. The set of realizable Routes can only stay the same or expand, and an existing successful Route remains successful. A newly available Route can nevertheless change an optimization result.

These laws separate three directions that are easy to conflate: refining evidence restricts possible realizations; refining the consequence distinguishes more outcomes; and enlarging the Establishment surface changes available means rather than current semantic determinacy. None of the propositions certifies that a carrier-specific front end represents the source correctly; P0-P2 provide that separate claim in Section 4.

### 3.6 Materiality hypergraph and CEGIS synthesis

The synthesis objective is an inclusion-minimal obligation set Q within the declared candidate surface P. The exact combinatorial structure can be stated before giving the reference loop.

Inclusion-minimal means minimal under set inclusion, not minimum cardinality, minimum cost, or uniqueness. The materiality hypergraph can have several incomparable inclusion-minimal transversals; the reference CEGIS loop certifies one such obligation set unless a separate enumeration or optimization objective is requested.

$$H_S = (P, E_S), \quad E_S = \{\, D(r_1, r_2) \mid r_1, r_2 \in S \wedge K_a(r_1) \neq K_a(r_2) \,\}$$

where $D(r_1, r_2) = \{\, q \in P \mid q(r_1) \neq q(r_2) \,\}$. $H_S$ is the materiality hypergraph of the declared analysis set: propositions are vertices, and each consequence-divergent pair contributes the set of candidate distinctions capable of separating that pair.

For a finite realization table with $K_a$ as the decision attribute, this edge family is isomorphic to the decision-relative discernibility construction used in rough-set reduct theory [29], [30]. The term materiality hypergraph names the role of that structure in this framework; it is not a claim to a new hypergraph or hitting-set primitive.

If $S' \subseteq S$, then $E_{S'} \subseteq E_S$ because every consequence-divergent pair available after restriction was already available before it. Proposition 1 therefore has an equivalent hypergraph reading: evidence refinement can delete materiality edges but cannot create a new edge from an excluded pair.

**Proposition 4 (exact hitting-set characterization).** A set $Q \subseteq P$ is sufficient over S if and only if Q intersects every hyperedge in $E_S$. Therefore the inclusion-minimal sufficient obligation sets are exactly the inclusion-minimal transversals of $H_S$. If $E_S$ contains the empty set, no $Q \subseteq P$ is sufficient.

$$Q \text{ is sufficient over } S \;\Leftrightarrow\; \forall D \in E_S\colon Q \cap D \neq \varnothing.$$

**Proof.** If Q misses an edge $D(r_1,r_2)$, the corresponding consequence-divergent pair agrees on every proposition in Q, so Q is not sufficient. Conversely, if Q is not sufficient, its fixed-Q counterexample $(r_1,r_2)$ yields an edge $D(r_1,r_2)$ disjoint from Q. The empty-edge case is a pair that agrees on all of P while producing different consequences, so no subset of P can separate it.

**Corollary 1 (existence criterion).** A sufficient obligation set exists within the declared candidate surface if and only if P itself is sufficient, equivalently if and only if $E_S$ contains no empty hyperedge. This makes UNRESOLVABLE UNDER DECLARED EVIDENCE SURFACE an exact vocabulary-boundary result rather than a search failure.

The reference implementation does not enumerate $E_S$. It discovers only the materiality hyperedges needed to certify one answer. The existential candidate followed by a universal pair check gives the procedure its counterexample-guided inductive synthesis, or CEGIS, structure [16]. The reference loop is:

1. Start with an empty accumulated hyperedge family A and witness set W.

2. Choose an inclusion-minimal hitting set Q of A. The initial candidate for A empty is the empty set.

3. Run the sound-and-complete fixed-Q relational verifier over the frozen analysis set S.

4. If the verifier returns UNSAT, return Q as an inclusion-minimal sufficient obligation set.

5. If the verifier returns a witness w, compute $D_w$. If $D_w$ is empty, return UNRESOLVABLE UNDER DECLARED EVIDENCE SURFACE. Otherwise add $D_w$ to A, preserve w in W, and repeat.

Theorem 1 (finite CEGIS correctness and relative completeness). Assume P is finite; S, $K_a$, P, and their declaration remain fixed during the run; the fixed-Q query is decidable and the verifier is sound and complete for it; and the candidate solver returns an inclusion-minimal hitting set of the accumulated nonempty hyperedges. Then the loop terminates after at most $2^{|P|}$ fixed-Q verifier calls. If it returns Q after UNSAT, Q is sufficient and inclusion-minimal among sufficient subsets of P. If it returns an empty $D_w$, no sufficient $Q \subseteq P$ exists. Thus, relative to the declared candidate surface and verifier, the procedure returns an inclusion-minimal sufficient obligation set exactly when one exists, and otherwise returns the explicit unresolvable result.

Proof. A SAT witness returned for the current candidate Q satisfies $\mathrm{Eq}_Q$, so its nonempty distinguishing set $D_w$ is disjoint from Q. Once $D_w$ is added, every later hitting set must intersect it, and the same Q cannot recur. As long as all accumulated edges are nonempty, P itself is a hitting set, so the candidate step remains defined. Because finite P has only $2^{|P|}$ subsets, nonrepetition forces a terminal verifier call within that many candidates: either UNSAT or a witness with $D_w$ empty.

If the loop returns UNSAT, the fixed-Q query has no consequence-divergent pair and Q satisfies the universal sufficiency condition. To prove inclusion minimality, let q be any element of the returned Q. Because Q is an inclusion-minimal hitting set of the accumulated family, Q without q fails to hit some accumulated $D_w$; the preserved witness for that edge is a concrete counterexample to the sufficiency of Q without q.

If $D_w$ is empty, a consequence-divergent pair agrees on every proposition in P and therefore on every $Q \subseteq P$, so no sufficient obligation set exists. Conversely, if no sufficient Q exists, UNSAT is impossible, and finite nonrepetition forces termination at the empty-edge result. These arguments are model-relative; source-level validity additionally requires the preservation conditions in Section 4.

### 3.7 From obligations to establishments

An obligation answers an informational sufficiency question: which semantic distinctions would determine the consequence if their values were available. Establishments describe the realizable operations available to acquire that information or transform the system so that the unresolved distinction ceases to affect the consequence.

The current reference kinds are:

1. epistemic observation
2. state-making act
3. operative guard
4. operative transaction
5. action suppression

The distinction can be stated directly over a compatible realization set S. For an epistemic observation e with outcome function $O_e$, observing result o produces the branch

$$S_o = \{ \mathrm{r} \in \mathrm{S} \mid O_e(\mathrm{r}) = \mathrm{o} \}.$$

The underlying realization remains the same while the compatible set is refined. For exposition, an interventional Establishment u can be represented by a declared transition relation $T_u$ over realizations. Its reachable successor set is

$$\mathrm{Post}_u(\mathrm{S}) = \{ \mathrm{r}' \mid \exists\, \mathrm{r} \in \mathrm{S}: (\mathrm{r}, \mathrm{r}') \in T_u \}.$$

The transition can change state, authority, execution semantics, or action availability as declared by the Boundary. Action suppression can terminate in a declared suppressed consequence when that value belongs to the consequence codomain.

Consider a state that remains open because a legacy capability may still be operative. A probe can partition the current realizations by observing whether the capability succeeds. An operative guard can instead transform the reachable set so that the legacy capability can no longer produce the relevant effect. The first route resolves the distinction epistemically; the second removes its consequence relevance through an intervention. Obligation synthesis and route synthesis therefore remain separate layers. A front end used for route synthesis must also preserve the source-dependent observation outcomes and declared transition semantics that define those branches; Section 4 makes that boundary explicit.


**Figure 3.** Reference compilation flow from a materiality witness to an obligation and then to a realizable closure route.

### 3.8 Closure routes

A Route is a finite policy over declared Establishments and the analysis states reachable from them. It may be a fixed sequence or a contingent policy whose branches depend on observation outcomes. Every step must be drawn from the Establishment surface declared by the Boundary, and its declared preconditions must hold for the states that can reach that step.

At an observation node, each possible outcome produces the corresponding compatible subset $S_o$. At an intervention node, successor analysis states follow the declared transition semantics. Each reachable leaf receives the ordinary closure status from the consequence values still possible at that leaf.

A successful closure route has a closed status at every reachable leaf: each leaf l has CLOSED($c_l$) for some declared consequence $c_l$. Different observation branches may close to different consequence values because the observed branch identifies which closed case applies. An explicit UNRESOLVED leaf records incomplete analysis and remains outside successful closure.

Semantic validity precedes optimization. Cost, probability, latency, or another declared objective can rank routes after every reachable branch satisfies the closure requirement. This ordering keeps optimization from selecting a cheaper policy that leaves some consequence-divergent branch unresolved.

This branch-complete success criterion matches the strong-planning structure used under partial observability [9]. The role of Route here is not to introduce a new planning formalism, but to bind that contingent-plan structure to the Consequence Closure leaf predicate, the declared Establishment semantics, and the source-preservation level required for exporting the route claim.

Section 3 therefore separates four computations: closure status at a cut, semantic sufficiency of an obligation set, operational realizability through declared Establishments, and optimization among successful closure routes. The structural laws identify how those claims move under evidence, realization, consequence, and Establishment changes; the materiality hypergraph makes obligation synthesis an exact minimal-transversal problem discovered lazily by CEGIS. The witness set remains available throughout as replayable evidence for the distinctions that drove synthesis.

## 4. Source Semantics and Closure Preservation

Section 3 assumes that the Boundary, candidate propositions, and declared Establishments preserve the consequence-relevant semantics of the source system. A carrier-specific front end can violate that assumption even while reproducing every final decision on the concrete source worlds. Closure also depends on the partial evidence states at which the compiled model may stop, request more evidence, or select an Establishment. This section first shows how source correlations can falsify a seemingly decision-preserving abstraction, then states the claim-indexed preservation contract, and finally subjects P0 and P1 to a frozen differential against the official Cedar 4.12.0 runtime.

### 4.1 Joint realizability and spurious completions

Cedar was used first as an external pressure case because its authorization model combines typed entities, identity, hierarchy, and relation-based policy conditions [21]. In this initial stage, the bounded front end mechanically parsed four policies from the Cedar integration-test corpus and matched all 14 expected request decisions in those cases. Its grammar was narrow by construction. Unsupported forms were rejected under the harness's fail-closed boundary rather than counted as evidence of semantic equivalence. This was a pressure test of the abstraction, not yet an official-runtime conformance claim. The 14-of-14 result here concerns the initial extraction and decision pressure case; it is distinct from the later C1 runtime-sanity gate in Section 4.3.

A separate source-derived policy fragment produced four semantic predicates: whether the principal is in the interns group, whether the principal is in the resource editors, whether the resource owner equals the principal, and whether the resource owner is in the interns group. Treating the predicates as independent Booleans admits 16 complete valuations. The source identity and hierarchy semantics admit 12. The difference changes closure under partial evidence.

**Table 3.** A source correlation changes closure under partial evidence in the bounded Cedar pressure case.

| Partial evidence | Source-constrained model | Independent Boolean abstraction |
|---|---|---|
| principal in interns = true<br>principal in editors = false<br>owner in interns = false | CLOSED(DENY)<br>owner = principal conflicts with the known membership facts | OPEN<br>admits a spurious completion with owner = principal |
| Bounded comparison | 12 reachable complete patterns<br>3 inclusion-minimal DENY reasons | 16 complete patterns<br>2 inclusion-minimal DENY reasons |

The comparison enumerated all 81 partial assignments obtained by giving each predicate the value unknown, true, or false. Seven assignments differed in closure status between the constrained and independent models; one of those differences occurred on the declared externally reachable observation surface. At the critical state shown in Table 3, the source-constrained model is CLOSED(DENY) while the independent abstraction remains OPEN. Complete-world decision agreement can therefore coexist with different stop conditions and different minimal-obligation structure. Relative to the bounded source-constrained model, the independent abstraction thus violates the status-preservation level defined below on those seven states and also changes the stronger obligation-preservation result. Constrained-reason analysis identifies the same underlying risk when impossible combinations are admitted [15]. The purpose of this stage was falsification: complete-world agreement did not rescue a partial-state abstraction that admitted source-impossible completions.

## 4.2 Preservation hierarchy over reachable analysis states

Let M denote the external source model and F the front end. An analysis state E denotes the source-realizable completions still possible at a cut, together with any source state needed to interpret subsequent Establishments. Let $\text{Reach}_M$ be the least set containing the declared externally reachable evidence states and closed under the declared Establishment steps. F(E) is the corresponding Core analysis state. Let $\text{Status}_M(E)$ be the closure status induced by the source-realizable completions at E, and let $\text{Status}_{\text{Core}}(F(E))$ be the status induced by the compiled Core realizations. Each q in the candidate surface P and each Establishment has the source interpretation recorded by its derivation certificate. Preservation is claim-indexed: reproducing closure status does not by itself justify a minimal-obligation or route claim. The reference contract therefore distinguishes three levels.

$$\text{P0:}\ \ \text{Status}_M(E) = \text{Status}_{\text{Core}}(F(E)) \quad \forall\, E \in \text{Reach}_M.$$

P0 - closure preservation. The equality includes the consequence value when the status is CLOSED(c) and distinguishes IMPOSSIBLE from nonempty closed states. P0 is sufficient for a consumer whose source-level claim is only whether closure analysis may terminate with the current status. Strong preservation provides a formal precedent for requiring agreement between concrete and abstract models over a chosen observable language [14]; here the preserved observable is the consequence-closure class at every reachable analysis state. P0 does not by itself preserve why a state is OPEN, which candidate distinctions are sufficient, or which Establishments form a realizable route.

**P1 - obligation preservation.** For E in $\text{Reach}_M$ and $Q \subseteq P$, let $\text{CE}_M(E,Q)$ mean that the source admits two completions consistent with E that agree on the source interpretations of every q in Q and disagree on the specified consequence. Let $\text{CE}_{\text{Core}}(F(E),Q)$ denote the analogous paired counterexample in the compiled model. Exact obligation preservation requires, in addition to P0:

$$\text{CE}_M(E,Q) \Leftrightarrow \text{CE}_{\text{Core}}(F(E),Q) \quad \forall\, E \in \text{Reach}_M,\ \forall\, Q \subseteq P.$$

Because sufficiency is exactly the absence of such a counterexample, P1 preserves the sufficiency predicate for every subset Q of the declared candidate surface. The source and Core therefore have the same family of sufficient obligation sets and, consequently, the same inclusion-minimal obligation families. P1 preserves the existence of a materiality witness for each failed Q, but existence preservation alone does not certify the source identity of a particular Core witness. If a concrete witness is exported as source-level evidence, its derivation record must reify it as a source-realizable pair with matching declared candidate values and consequence values; otherwise it remains a model witness. Multiple source witnesses may still collapse to one Core-equivalent witness. For the frozen global obligation profile, the same condition is evaluated at the declaration's unconditioned analysis state whose source completion set is $R_B$.

**P2 - route preservation.** For a declared Establishment e and reachable source analysis state E, let $\text{Enabled}_M(e,E)$ state whether its source precondition holds, and let $\text{Step}_M(e,E)$ be the set of labeled successor analysis states generated by

the source semantics. A label is an observation outcome, a declared intervention branch, or a declared terminal branch such as suppression. Define $\text{Enabled}_{\text{Core}}$ and $\text{Step}_{\text{Core}}$ analogously. Exact route preservation requires P0 and P1 and, for every declared e and E:

$$\text{Enabled}_M(e,E) \Leftrightarrow \text{Enabled}_{\text{Core}}(e,F(E))$$

$$\text{Step}_{\text{Core}}(e,F(E)) = \{\ (b, F(E')) \mid (b, E') \in \text{Step}_M(e,E)\ \}$$

where duplicate successor images merged by F under the declared source compression are identified. The condition preserves both enabledness and the complete branch structure modulo that compression. For observations, it forbids inventing or dropping a source-reachable outcome. For interventions, it forbids a compiled transition from omitting a source-reachable successor or introducing an unsupported operative effect.

**Preservation consequence.** Under P0 and P1, a set Q is sufficient in the source exactly when it is sufficient in the Core, so inclusion-minimal obligations correspond. Under P0-P2, finite route success is preserved: induction on route depth gives corresponding reachable branches, and P0 gives the same closure status at every corresponding leaf. A successful Core closure route is therefore successful in the source over the declared Establishment surface, and exact P2 also reflects source routes back into the Core. Source-level claims of minimum cost, latency, probability, or another route objective additionally require the objective annotations and aggregation rule used for ranking to be preserved; route success alone does not establish optimality.

The exact levels above are stronger than the one-sided contracts useful for conservative front ends. An over-approximating front end can support a sound but incomplete claim if, at every corresponding reachable state, Core CLOSED(c) implies source CLOSED(c) with the same c and Core IMPOSSIBLE does not hide a nonempty source state; every source counterexample relevant to Q is represented by a Core counterexample; every Core-enabled Establishment is enabled in the source; and every source-reachable successor of a selected Establishment appears among the Core successors under the same declared branch label.

Under such a contract, Core closure or route success can imply source closure or route success even though the Core may remain OPEN, disable an available source Establishment, add spurious successor branches, or fail to find a route that exists in the source. The contract therefore does not justify equality of closure status, inclusion-minimality, route completeness, or source-level optimality. A conformance record must name the exact or one-sided level it supports.

The hierarchy is output-relative rather than representation-prescriptive. A front end may compress carrier syntax, identities, or complete worlds whenever the resulting quotient satisfies the claimed level. P0 can admit compressions excluded by P1, and P1 can admit compressions excluded by P2, because stronger outputs expose progressively more of the source distinction and transition structure. This is why complete-world decision agreement alone is an insufficient front-end test.

### 4.3 Official-runtime differential after semantic freeze

The preservation hierarchy was then converted into a preregistered external-oracle gate rather than inferred from the source-constrained pressure model. Before primary execution, the action cut, 64-world bounded source domain, four source-derived Proposition meanings, acquisition surface, source-oracle and Core paths, and independent comparator were locked. Official cedar-policy-cli 4.12.0, pinned to release commit fdcbaed32bdb8c8d13e4eaf2b58db5555e9fb8c5, served as the source oracle [22]. For each complete world it evaluated the frozen target-policy consequence and, through one-policy wrappers containing the frozen Cedar expressions, the four candidate Proposition values. Source and Core result bundles were produced separately and sealed before the comparator read both. The primary reachable surface contained 183 acquisition-reachable analysis states, and P1 checked every one of the 16 subsets Q of P at every such state. The 81 ternary states from Section 4.1 were retained as a secondary exhaustive stress surface, not promoted into $\text{Reach}_M$.

The change-control path was exercised before any primary semantic outcome. RUN-001 passed the 14-of-14 sanity gate, then official whole-policy validation stopped the run before the 64-world source-oracle loop because the reduced schema declared only CreateTask although the locked policy also referenced UpdateList, UpdateTask, and DeleteTask. No primary source-world result or comparator classification had been observed. RUN-001 was retained rather than overwritten. AMENDMENT-001 was classified SCHEMA_COMPLETENESS_ONLY, and RUN-002 added only the missing action declarations. The target action remained Action::"CreateTask"; the primary policy, request, 64 worlds, candidate meanings, acquisition declarations, source-oracle semantic code, Core, comparator, and

Cedar runtime remained unchanged. The correction was therefore selected from a static validation failure, not from the primary semantic result.

**Table 4.** Official Cedar 4.12.0 differential over the frozen bounded source boundary.

| Check | Qualified surface | RUN-002 result |
|---|---|---|
| Runtime sanity | 14 selected request decisions | 14 / 14 |
| Complete-world consequence | 64 frozen worlds | 0 mismatches; 52 ALLOW / 12 DENY matched per world |
| Proposition interpretation | 4 candidates × 64 worlds | 0 mismatches |
| P0 closure preservation | 183 reachable analysis states | 0 mismatches |
| P1 obligation preservation | 183 states × all 16 Q subsets (2,928 checks) | 0 mismatches |
| Witness reification | exported materiality witnesses | 427 checked; 0 failed |
| Secondary A81 stress | 81 ternary analysis states | 0 source/Core mismatches; not a reachability claim |
| Declaration integrity | post-outcome semantic repair | 0 |

**Primary classification: EXACT_P1_CONFIRMED_ON_FROZEN_BOUNDARY**

All C0-C6 gates passed. The recorded post-outcome semantic repair count was zero. The result is stronger than complete-world decision agreement: on this frozen boundary and under the locked candidate mapping, source and Core agreed on closure status at every reachable analysis state and on the paired-counterexample predicate for every candidate subset Q. They therefore induced the same sufficient-set family and the same inclusion-minimal obligation families over the declared surface. All 427 exported materiality witnesses checked by the gate were reified to source-realizable pairs without failure.

The claim remains bounded to this experiment. It does not establish equivalence for Cedar generally, arbitrary policies, or other carriers. P2 was not tested, route transfer is not inferred, and A81 remains a secondary stress audit. The reported P1 result is grounded in official concrete execution over the frozen source realizations and exhaustive restriction to the declared reachable information states; it does not claim equivalence to Cedar's own partial-evaluation machinery. Within that boundary, the earlier failure mode in Section 4.1 is no longer supported only by a hand-built reference model: the frozen front end is checked against the official runtime on the exact P0/P1 outputs it claims to preserve.

### 4.4 Derivation, conformance, and trust boundary

A source-preservation claim needs replayable lineage and an explicit conformance level. The reference derivation certificate records the external source artifact or machine evidence, the relevant expression, field, event, or protocol result, available type and identity information, constraints that preserve joint realizability, the transformation that produced the Core object, and any bounded approximation introduced by the front end. The conformance record names the declaration version, the reachable analysis surface tested, the highest exact or one-sided preservation level claimed, and the oracle or derivation evidence supporting that level. A source-level concrete Witness additionally identifies the source-realizable pair or replay evidence that reifies the Core witness. A P2 or route-optimality claim identifies the Establishment semantics and any objective annotations whose correspondence is assumed or tested. Later refinements cannot become earlier premises. Unsupported source semantics produce an explicit unresolved result. Finite completion domains remain declared components of the Boundary when the external source leaves values unconstrained.

The Cedar gate now supplies one exact P1 conformance record for one frozen bounded front end. Its evidentiary force comes from the locked declaration, official source oracle, separate source/Core execution paths, full reachable-state and Q-subset comparison, and source reification of exported witnesses. It does not elevate unrelated front ends to P1 and does not establish P2. The bounded world construction, source-expression derivation, acquisition model, official Cedar runtime, and harness and bundle integrity remain qualified components of the experiment; they were pinned and replayably recorded rather than mathematically proved.

More generally, a conformance record must state which external parsing and typing, source-to-Core derivation, finite completion-domain declarations, evidence acquisition, transition models, and external oracles remain trusted or

independently checked. The Boundary must also make the Section 3.1 cut-adequacy assumption explicit: consequence-relevant source transitions between the evidence cut and the realized action effect cannot disappear merely because the front end does not represent them. The Core can then verify the compiled closure problem only within that declared boundary. Source conformance and mechanism commissioning are complementary rather than interchangeable. Section 5 supplies bounded real-system evidence for selected declared Establishment and consequence semantics, including prospective route selection and implementation transfer; those runs do not automatically raise an unrelated front end to P1 or P2.

# 5. System Commissioning and Transfer

Section 4 asks whether a carrier-specific front end preserves the source semantics needed for the claim it exports. This section tests a different link. When a model-relative result relies on a declared Establishment, consequence mapping, or evidence gate, commissioning asks whether that declared effect survives contact with a concrete system at the stated action cut, and whether frozen selection or checking logic transfers without semantic repair. The assurance stack therefore has three distinct links: Section 3 establishes model-relative determinacy and synthesis; Section 4 qualifies source-to-Core preservation; Section 5 qualifies selected mechanism and evidence semantics against real systems.

An external action-level claim inherits every applicable qualification. Success at one link does not repair a failure at another: commissioning can qualify mechanism or evidence semantics without establishing P0, P1, or P2 for that system, and the Cedar P1 result does not establish the operativity of an unrelated mechanism.

The unit of evidence is consequently a specific link rather than an undifferentiated system-level pass: effect semantics at a cut, generic route selection over declared mechanism semantics, prospective selection before the outcome is known, confound-controlled consequence evidence, or transfer of frozen checking logic. The evaluation is organized around the corresponding threats to interpretation. A model-relative route can fail at a real effect cut, depend on domain-specific compiler logic, reflect a mechanism selected only after its success was observed, classify an unrelated request failure as operative revocation, or require implementation-specific repair. The commissioning sequence addresses those alternatives separately. Repeated 200-trial runs establish repeatability under the fixed adversarial schedules used here; they are not statistical estimates of behavior outside the declared schedules and system boundaries.

## 5.1 Operativity at a real effect cut

The first commissioning isolates the distinction introduced in Section 1 at a Linux system call boundary. A writable memfd was created before a child inherited its descriptor. Each condition completed and verified its control change before the child attempted pwrite through that already established descriptor.

In the administrative condition, fchmod set mode 0444 and fstat verified the mode in all 200 trials; the inherited descriptor still wrote successfully in all 200. In the operative conditions, F_SEAL_WRITE was established and verified before the write attempt. Both the expert control and the compiler-selected route blocked all 200 writes with EPERM. Linux specifies that file seals take effect when successfully added and that F_SEAL_WRITE prevents modification of file contents [1].

**Table 5.** Linux operativity commissioning at an already established write path.

| Condition | Control verified | Post-control write through inherited descriptor |
|---|---|---|
| **fchmod 0444 + fstat** | 200 / 200 | succeeded 200 / 200 |
| **Expert F_SEAL_WRITE** | 200 / 200 | succeeded 0 / 200; EPERM 200 / 200 |
| **Compiler-selected F_SEAL_WRITE** | 200 / 200 | succeeded 0 / 200; EPERM 200 / 200 |

F_SEAL_WRITE was supplied in the declared Establishment surface with its profile semantics. The compiler selected that Establishment because its declared effect made the post-declaration write consequence invariant across the modeled realizations, and the real run then checked the profile claim against kernel behavior. The result establishes one bounded effect cut in which verified administrative metadata and an operative control produce different consequences for the same preexisting write capability.

### 5.2 Cross-system reuse and prospective selection

A second set of tests examines whether route selection depends on domain-specific compiler logic or on knowledge gained from the eventual real-system result. Cross-system reuse came first. One frozen compiler executable, containing zero branches keyed to SQLite or Linux, processed both a SQLite lost-update profile and the Linux write profile. In SQLite 3.46.1 [36] with WAL mode, whose documented concurrency model provides snapshot isolation across separate database connections [37], an adversary committed after the final baseline observation and before the client effect. The naive sequence lost the intervening update in 200 of 200 trials. Both an expert compare-and-swap route and the compiler-selected CAS route lost 0 of 200. The same compiler selected F_SEAL_WRITE for the Linux profile described above.

The profiles declared the semantics of CAS and F_SEAL_WRITE as part of their trust boundary. This stage tests generic route selection over declared mechanism semantics, while the real-system runs test whether those declarations match the commissioned mechanisms.

A prospective filesystem test then separated route selection from the real run. The profile modeled a publication step in which another actor could create the destination after the client prepared its source but before publication. The declared Establishment surface included ordinary rename and a no-replace binding. Before commissioning, the frozen compiler selected RENAME_NOREPLACE_BIND. A paired negative profile that omitted the no-replace Establishment returned no closure route, preserving the capability boundary from Section 3.8. The prediction bundle was sealed while its commissioning status still recorded NOT YET RUN. The seal fixed the selected route and profile semantics before the real-system outcome, implementing Section 3.3.

The real-system schedule then inserted the competing publication immediately before the client publication step. Ordinary rename replaced the destination in 200 of 200 trials. The compiler-selected RENAME_NOREPLACE path preserved the competing destination in all 200 trials, with 0 of 200 clobbers, and returned EEXIST in 200 of 200, matching the expert no-replace control. Linux documents RENAME_NOREPLACE as a renameat2 flag that leaves an existing destination in place and returns an error when that destination already exists [23].

This sequence separates two bounded transfer claims. The SQLite and Linux runs show reuse of one compiler across distinct mechanism families under declared profile semantics. The filesystem run adds prospective route selection under that frozen compiler, including a negative profile that leaves the problem unresolved when the required Establishment is absent. The mechanism semantics remain explicit profile inputs. Their real-system commissioning tests the declared effect claims, while any source-to-Core preservation claim remains separately governed by Section 4.

### 5.3 Authority evidence at a measured cut

OAuth adds a setting in which authority events and protected-resource outcomes can be captured as machine records. RFC 7009 specifies token invalidation while acknowledging propagation delay across servers [2]. RFC 7662 permits protected resources to cache introspection responses and warns that a revoked token can remain usable during the resulting stale information window [3]. The commissioning question is therefore path-specific and time-bounded:

> *Is this access token's revocation effective against this exact protected resource HTTP action path at this measured cut?*

SAFE_AT_CUT and UNSAFE are profile-level consequence classifications, not aliases for Core CLOSED and OPEN. A directly observed post-revocation acceptance can be a determinately adverse outcome at the measured path even though its profile verdict is UNSAFE; missing evidence instead leaves the profile OPEN. The positive gate below is therefore an evidentiary rule for assigning SAFE_AT_CUT at this qualified cut, not a redefinition of closure as goodness. It is also an operational classification, not a claim that revocation is the uniquely identified causal explanation for a blocked request; the paired controls and timing gates constrain specified alternatives while the claim remains limited to the measured path and cut.

The active commissioner issues two distinct access tokens from the same client and scope, establishes target and control baselines on the selected HTTP method and URL, revokes only the target token, optionally introspects it, and then performs a control before, target, control after sandwich. Raw bearer tokens remain in memory; the evidence record stores SHA-256 fingerprints, machine timestamps, endpoint results, the HTTP method, and the resource URL.

Post-revocation target acceptance is direct evidence that the selected path remains operative and yields the profile verdict UNSAFE. A blocked target supports SAFE_AT_CUT only when every positive gate below is satisfied. Introspection contributes authority-side evidence; the protected-resource probe supplies the action-path outcome.

**Table 6.** Positive SAFE_AT_CUT gates and the alternative explanations they constrain.

| Positive SAFE_AT_CUT gate | Confound controlled |
|---|---|
| Target pre-revocation success | establishes that the target token could use the selected path before revocation |
| Distinct control pre-revocation success | establishes an unrevoked control baseline |
| Successful revocation response | records acknowledgement of the target revocation request |
| Same HTTP method and URL | binds the before and after probes to the same selected action path |
| Target probe before natural expiry | excludes token expiry as the blocking explanation |
| Control succeeds immediately before and after target probe | shows the measured resource path remained available around the target check |
| Target post-revocation response is 401 or 403 | establishes that the target token no longer succeeds on the measured path at the cut |

Missing positive gates remain OPEN. Post-revocation target acceptance yields UNSAFE even when the revocation endpoint itself failed, because the action effect has already been observed. When introspection reports active:false while the resource still accepts the target, the checker records AUTHORITY_RESOURCE_SPLIT_BRAIN.

The positive gate was also enumerated exhaustively over its declared ten-dimensional Boolean abstraction. Among 1,024 well-formed configurations, exactly one satisfied every SAFE_AT_CUT condition, 512 were UNSAFE because post-revocation target acceptance dominated the remaining signals, and 511 were OPEN. The frozen checker was compared with the declared classification oracle over all 1,024 configurations and emitted no SAFE_AT_CUT result where that oracle was non-SAFE. This is a conformance check over the checker gate abstraction, not independent evidence that the abstraction exhausts OAuth protocol behavior; behavior outside it remains governed by the qualified profile.

### 5.4 Transfer without semantic repair

The final commissioning step holds the OAuth evidence format and semantic checker fixed while changing the implementation behind the authority and resource path. The same generic runner and checker were first exercised against OAuthLib 3.3.1, an independently maintained OAuth protocol library [24]. The live condition produced SAFE_AT_CUT with subtype SANDWICHED_PAIRED_CONTROL_CLOSURE. A deliberately stale protected-resource produced UNSAFE with subtype AUTHORITY_RESOURCE_SPLIT_BRAIN. The semantic repair count was zero, and the checker contained zero OAuthLib-specific semantic branches.

The next run used Keycloak 26.7.2, released on August 19, 2026 [25]. Keycloak documents RFC 7009 revocation and RFC 7662 introspection endpoints in its OpenID Connect interface [26]. Before evaluation, six frozen semantic artifacts were checked against their recorded hashes. The archived commissioning record binds the run to release commit 289376b142480b4d600aca7acb1e4651862ed2a1 and container image digest quay.io/keycloak/keycloak@sha256:831330513f55695572286e521f94fcd3c7e285250ed5b848090265a33192f669. The live condition again produced SAFE_AT_CUT, while the controlled stale resource condition produced AUTHORITY_RESOURCE_SPLIT_BRAIN. Semantic repair remained zero and the checker retained zero Keycloak-specific semantic branches.

The stale resource condition in both implementations is a harness-supplied negative control. Its evidentiary role is to test whether the frozen checker continues to distinguish authority-side inactivity from continued protected-resource acceptance under a deliberately split state.

**Table 7.** Frozen OAuth checker transfer across the two commissioned implementations.

| Runtime | Live condition | Controlled stale condition | Semantic repair | Provider-specific checker branches |
|---|---|---|---|---|
| OAuthLib 3.3.1 | SAFE_AT_CUT | UNSAFE / AUTHORITY_RESOURCE_SPLIT_BRAIN | 0 | 0 |
| Keycloak 26.7.2 | SAFE_AT_CUT | UNSAFE / AUTHORITY_RESOURCE_SPLIT_BRAIN | 0 | 0 |

These runs support a bounded implementation transfer claim for the OAuth profile: the same frozen checker separated the profile's SAFE_AT_CUT live condition from authority-resource divergence across both commissioned implementations without semantic repair. The Keycloak result is bound to the pinned local runtime and archived image digest. Other OAuth flows, revocation mechanisms, deployment topologies, and provider configurations remain separate empirical questions.

Table 8 summarizes the distinct evidentiary role of each commissioning stage. The rows are complementary rather than a ladder in which a later result subsumes an earlier one: each controls a different rival explanation, and each claim remains limited to its stated boundary.

**Table 8.** Commissioning map: each stage addresses a distinct threat to interpretation.

| Stage | Threat addressed | Bounded result |
|---|---|---|
| **Linux effect cut** | verified administrative state may be mistaken for operative control | mode 0444 was verified while the inherited write succeeded 200 / 200; F_SEAL_WRITE blocked 200 / 200 |
| **SQLite and Linux cross-system reuse** | compiler behavior may depend on domain-specific branches | one frozen compiler with zero SQLite or Linux branches selected CAS and F_SEAL_WRITE; selected routes matched expert controls in 200-trial runs |
| **Prospective filesystem publication** | route may be selected only after observing the successful mechanism | RENAME_NOREPLACE_BIND was selected and sealed before commissioning; absent from the negative profile, no route was returned; real no-replace path clobbered 0 / 200 |
| **OAuth paired control gate** | a blocked target may reflect outage, path mismatch, or expiry | SAFE_AT_CUT requires same path, preexpiry, paired controls, and target block; across 1,024 declared gate states the checker emitted no SAFE result where the declared oracle was non-SAFE |
| **OAuthLib transfer** | checker may depend on the local authority implementation | frozen checker returned live SAFE and controlled stale UNSAFE with zero repair and zero provider-specific branches |
| **Keycloak transfer** | checker may require provider-specific semantic repair | pinned 26.7.2 runtime returned live SAFE and controlled stale UNSAFE with six frozen hashes matched, zero repair, and zero provider-specific branches |

Taken together, the sequence provides heterogeneous but bounded evidence for the path from semantic closure to operative use; it is not a single cross-carrier equivalence test. The Linux run separates verified administrative state from operative effect on an established capability path. The SQLite/Linux profiles test reuse of one carrier-neutral route selector over declared mechanism semantics. The filesystem run adds prospective selection and a negative capability boundary before commissioning. The two OAuth implementations test one frozen evidence checker across distinct implementations without provider-specific semantic repair. Across these stages, the compiler or checker remains frozen where the transfer claim requires it, while mechanism semantics and source qualifications remain explicit. The release archive preserves the corresponding machine records, prediction seals, manifests, and verifiers for replay. These experiments pressure different links in the assurance chain; they establish neither P1/P2 for the commissioned systems nor general carrier neutrality.

# 6. Inspectable Assurance Records

Sections 3 through 5 establish different kinds of assurance. Inspectability is not an additional assurance layer; its role is to preserve the claim and its boundary. An inspection record should carry the model-relative result, the evidence

needed to challenge it, and the qualifications that prevent it from silently becoming a stronger claim. The reference Inspector implements that record contract.

Four invariants define the inspectability contract. First, claim fidelity: the exported status, obligation family, witnesses, and Route must agree with the underlying analysis, and unresolved states must remain unresolved. Second, challengeability: an OPEN result carries a concrete materiality witness; a selected inclusion-minimal obligation set carries removal witnesses for each selected Proposition; and an empty discernibility edge is reported as UNRESOLVABLE UNDER DECLARED EVIDENCE SURFACE rather than repaired by inventing a discriminator. Third, qualification preservation: Boundary scope, cut adequacy, source-preservation level, and applicable mechanism evidence remain explicit, and a model-relative Route is not promoted to a source-system Route without the required P2 evidence. Fourth, replay identity: canonical source, declaration, state, and record identities are bound to the exported object so that the same qualified analysis can be reproduced and compared.

**Table 9.** Inspectability contract for an exported Consequence Closure assurance record.

| Record element | What it makes inspectable | What it does not establish |
|---|---|---|
| **Declaration and Boundary identity** | Which action cut, realization model, evidence state, candidate vocabulary, and Establishment surface the result is about | That the declared source model is complete or that the cut is adequate unless separately established |
| **Closure result, obligation family, and Witnesses** | Why the current result is CLOSED, OPEN, IMPOSSIBLE, or unresolvable, and why each Proposition in a selected inclusion-minimal set is necessary | Minimum cardinality, minimum cost, uniqueness, or correctness outside the declared Boundary |
| **Declared Route** | Which realizable Establishment policy closes every modeled branch and which branch outcomes remain possible | Transfer to a source-system Route, route completeness, or source-level optimality without the applicable P2 and objective evidence |
| **Preservation and mechanism qualifications** | Which source-to-Core and real-system links have actually been checked, and which remain unestablished | That success at one assurance link repairs or substitutes for another |
| **Digests and replay references** | Identity, deterministic replay, and comparison of the exact declaration, state, evidence bundle, and exported record | External origin, authenticity, freshness, or truth of the referenced evidence |

On a successful exact analysis, the record exposes the complete inclusion-minimal obligation family computed for the declared finite problem; deterministic selection of one member is only a presentation choice. If the exact work budget is exceeded, the reference implementation fails without emitting a partial or approximate family. Obligation and mechanism remain separate throughout: the record can show both what semantic distinctions are sufficient and what declared Establishment policy can close the gap without treating one as a substitute for the other.

Inspector v0.5.0 [38] serves as a portable reference implementation of this contract. Semantic authority remains with the underlying declaration, evidence, and conformance records. The implementation recomputes the supported local analyses and presents them in a proof-oriented order: status, obligation family and certificate, Route, qualifications, and source lineage. Missing source or cut evidence remains explicitly unestablished. It runs locally without requiring a server or external network request, keeping inspection and replay independent of a hosted service.

Artifact validation is contract-oriented rather than interface-oriented. The release gate matches exact inclusion-minimal obligation families to a brute-force oracle on 300 randomized finite models, exercises incomparable minimal sets and unresolvable vocabularies, and fails closed when the declared exact work budget is exceeded. It also checks deterministic export, stale-state clearing, hostile display text, and absence of external network requests; the frozen OAuth and Keycloak checks and the canonical Cedar Run-002 evidence are carried without carrier-specific Core branches. These tests support faithful transport and replay; they do not raise the assurance level of the source or mechanism claims carried by the record.

The methodological purpose is to close the loop between formal result and reviewable evidence. A recipient can reconstruct the declaration, replay the conclusion, inspect the counterexample or minimality certificate, and see which qualifications still limit export to the source system. The Inspector is useful only insofar as it makes those claims easier to challenge without changing their meaning.

# 7. Conclusion

This note isolates a failure mode that sits between familiar assurance layers: a system can have valid authority, accurate observations, and correctly configured controls while the consequence of the action at hand still depends on unresolved state. Consequence Closure makes that residual dependence the object of assurance. At a declared action cut, it asks whether all admissible realizations consistent with the evidence agree on the specified consequence; when they do not, it exposes the distinctions and realizable Establishments needed to eliminate the dependence. The target complements authorization, planning, enforcement, and runtime control rather than replacing them.

The work supports that target in three layers. At the model level, closure and semantic sufficiency are exact relational questions; the materiality hypergraph characterizes inclusion-minimal obligation sets, and finite CEGIS either returns one or exposes a declared-vocabulary boundary. At the source level, P0, P1, and P2 separate the preservation required for status, obligation, and Route claims. The Cedar pressure case first demonstrated that complete-world agreement can preserve final decisions while distorting partial-state assurance, and the later frozen official-runtime differential confirmed exact P1 on its declared bounded surface without post-outcome semantic repair. At the systems level, commissioning tests distinct mechanism and evidence links rather than treating successful execution as a substitute for source preservation. The inspectability contract carries the resulting claims, witnesses, and qualifications forward without strengthening them.

For increasingly heterogeneous agent infrastructure, this suggests a useful interface boundary. Current standards work is already confronting action-specific authority, delegation, interoperable agent identity, least privilege, and audit across software-agent boundaries [17], [18]. Consequence Closure addresses an adjacent question that those mechanisms do not answer by themselves: whether the evidence and operative state at the action boundary are sufficient to fix the specified consequence. Systems need not converge on one policy language or control plane to target that question; a front end can preserve the consequence-relevant distinctions, Establishments, and qualifications while the underlying identity and enforcement mechanisms remain heterogeneous.

The present evidence is bounded. General source derivation, P2 transfer across further carriers, evidence origin and authenticity, and broader carrier generality remain open empirical and formal questions. They mark the present boundary of the assurance chain. The practical objective is correspondingly precise: before a consequential machine action, make it possible to inspect what can still change the consequence, why that distinction is material, what realizable step can establish closure, and what evidence warrants carrying the conclusion from the model to the source system and the operative action path.

# References

[1] Linux man-pages project, "F_GET_SEALS, F_ADD_SEALS - get/add file seals," Linux man-pages 6.18, F_GET_SEALS(2const), sec. F_SEAL_WRITE, Jul. 20, 2025. [Online]. Available: https://man7.org/linux/man-pages/man2/F_GET_SEALS.2const.html. Accessed: Aug. 24, 2026.

[2] T. Lodderstedt, Ed., S. Dronia, and M. Scurtescu, "OAuth 2.0 Token Revocation," RFC 7009, Aug. 2013, doi: 10.17487/RFC7009.

[3] J. Richer, Ed., "OAuth 2.0 Token Introspection," RFC 7662, Oct. 2015, doi: 10.17487/RFC7662.

[4] J. H. Saltzer and M. D. Schroeder, "The Protection of Information in Computer Systems," Proceedings of the IEEE, vol. 63, no. 9, pp. 1278-1308, Sep. 1975, doi: 10.1109/PROC.1975.9939.

[5] OASIS, eXtensible Access Control Markup Language (XACML) Version 3.0, E. Rissanen, Ed., OASIS Standard, Jan. 22, 2013. [Online]. Available: https://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html. Accessed: Aug. 24, 2026.

[6] J. Park and R. Sandhu, "The UCONABC Usage Control Model," ACM Transactions on Information and System Security, vol. 7, no. 1, pp. 128-174, Feb. 2004, doi: 10.1145/984334.984339.

[7] M. K. Aguilera, M. Ji, M. Lillibridge, J. MacCormick, E. Oertli, D. Andersen, M. Burrows, T. Mann, and C. A. Thekkath, "Block-Level Security for Network-Attached Disks," in 2nd USENIX Conference on File and Storage Technologies (FAST 03), San Francisco, CA, USA, Mar. 2003, USENIX Association. [Online]. Available: https://www.usenix.org/conference/fast-03/block-level-security-network-attached-disks. Accessed: Aug. 24, 2026.

[8] T. C. Son and C. Baral, "Formalizing Sensing Actions: A Transition Function Based Approach," Artificial Intelligence, vol. 125, nos. 1-2, pp. 19-91, Jan. 2001, doi: 10.1016/S0004-3702(00)00080-1.

[9] P. Bertoli, A. Cimatti, M. Roveri, and P. Traverso, "Strong Planning under Partial Observability," Artificial Intelligence, vol. 170, nos. 4-5, pp. 337-384, Apr. 2006, doi: 10.1016/j.artint.2006.01.004.

[10] P. Bulychev, F. Cassez, A. David, K. G. Larsen, J.-F. Raskin, and P.-A. Reynier, "Controllers with Minimal Observation Power: Application to Timed Systems," in Automated Technology for Verification and Analysis (ATVA 2012), LNCS 7561, S. Chakraborty and M. Mukund, Eds. Berlin, Germany: Springer, 2012, pp. 223-237, doi: 10.1007/978-3-642-33386-6_19.

[11] J. Ligatti, L. Bauer, and D. Walker, "Edit Automata: Enforcement Mechanisms for Run-Time Security Policies," International Journal of Information Security, vol. 4, nos. 1-2, pp. 2-16, Feb. 2005, doi: 10.1007/s10207-004-0046-8.

[12] M. R. Clarkson and F. B. Schneider, "Hyperproperties," Journal of Computer Security, vol. 18, no. 6, pp. 1157-1210, 2010, doi: 10.3233/JCS-2009-0393.

[13] T. Terauchi and A. Aiken, "Secure Information Flow as a Safety Problem," in Static Analysis, 12th International Symposium (SAS 2005), LNCS 3672. Berlin, Germany: Springer, 2005, pp. 352-367, doi: 10.1007/11547662_24.

[14] F. Ranzato and F. Tapparo, "Generalized Strong Preservation by Abstract Interpretation," Journal of Logic and Computation, vol. 17, no. 1, pp. 157-197, Feb. 2007, doi: 10.1093/logcom/exl035.

[15] N. Gorji and S. Rubin, "Sufficient Reasons for Classifier Decisions in the Presence of Domain Constraints," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, no. 5, pp. 5660-5667, Jun. 2022, doi: 10.1609/aaai.v36i5.20507.

[16] A. Solar Lezama, "Program Synthesis By Sketching," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-176, Dec. 19, 2008. [Online]. Available: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-176.html. Accessed: Aug. 24, 2026.

[17] National Institute of Standards and Technology, "AI Agent Standards Initiative," created Feb. 17, 2026, updated Apr. 20, 2026. [Online]. Available: https://www.nist.gov/artificial-intelligence/ai-agent-standards-initiative. Accessed: Aug. 24, 2026.

[18] H. Booth, W. Fisher, R. Galluzzo, and J. Roberts, "Accelerating the Adoption of Software and Artificial Intelligence Agent Identity and Authorization," NIST National Cybersecurity Center of Excellence, Concept Paper, Feb. 5, 2026. [Online]. Available: https://www.nccoe.nist.gov/sites/default/files/2026-02/accelerating-the-adoption-of-software-and-ai-agent-identity-and-authorization-concept-paper.pdf. Accessed: Aug. 24, 2026.

[19] D. Dean and A. J. Hu, "Fixing Races for Fun and Profit: How to Use access(2)," in 13th USENIX Security Symposium (USENIX Security 04), San Diego, CA, USA, Aug. 2004, USENIX Association, pp. 195-206. [Online]. Available: https://www.usenix.org/legacy/publications/library/proceedings/sec04/tech/full_papers/dean/dean.pdf. Accessed: Aug. 24, 2026.

[20] E. W. Dijkstra, "Guarded Commands, Nondeterminacy and Formal Derivation of Programs," Communications of the ACM, vol. 18, no. 8, pp. 453-457, Aug. 1975, doi: 10.1145/360933.360975.

[21] J. W. Cutler, C. Disselkoen, A. Eline, S. He, K. Headley, M. Hicks, K. Hietala, E. Ioannidis, J. Kastner, A. Mamat, D. McAdams, M. McCutchen, N. Rungta, E. Torlak, and A. M. Wells, "Cedar: A New Language for Expressive, Fast, Safe, and Analyzable Authorization," Proceedings of the ACM on Programming Languages, vol. 8, OOPSLA1, Art. 118, 28 pp., Apr. 2024, doi: 10.1145/3649835.

[22] Cedar Policy Project, "cedar-policy-cli v4.12.0," GitHub release cedar-policy-cli-v4.12.0, Jul. 28, 2026, Cedar Language v4.5, commit fdcbaed32bdb8c8d13e4eaf2b58db5555e9fb8c5. [Online]. Available: https://github.com/cedar-policy/cedar/releases/tag/cedar-policy-cli-v4.12.0. Accessed: Aug. 24, 2026.

[23] Linux man-pages project, "rename, renameat, renameat2 - change the name or location of a file," Linux man-pages 6.18, rename(2), sec. RENAME_NOREPLACE, Feb. 8, 2026. [Online]. Available: https://man7.org/linux/man-pages/man2/rename.2.html. Accessed: Aug. 24, 2026.

[24] OAuthLib project, "OAuthLib 3.3.1," PyPI release, Jun. 19, 2025. [Online]. Available: https://pypi.org/project/oauthlib/3.3.1/. Accessed: Aug. 24, 2026.

[25] Keycloak, "Keycloak 26.7.2 released," Aug. 19, 2026. [Online]. Available: https://www.keycloak.org/2026/08/keycloak-2672-released. Accessed: Aug. 24, 2026.

[26] Keycloak, "Securing applications and services with OpenID Connect," secs. "Introspection endpoint" and "Token Revocation endpoint." [Online]. Available: https://www.keycloak.org/securing-apps/oidc-layers. Accessed: Aug. 24, 2026.

[27] W. W. Armstrong, "Dependency Structures of Data Base Relationships," in Information Processing 74: Proceedings of IFIP Congress 74. Amsterdam, The Netherlands: North-Holland, 1974, pp. 580-583.

[28] A. Nash, L. Segoufin, and V. Vianu, "Views and Queries: Determinacy and Rewriting," ACM Transactions on Database Systems, vol. 35, no. 3, Art. 21, 41 pp., Jul. 2010, doi: 10.1145/1806907.1806913.

[29] Z. Pawlak, Rough Sets: Theoretical Aspects of Reasoning about Data, Theory and Decision Library, Series D, vol. 9. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1991, doi: 10.1007/978-94-011-3534-4.

[30] A. Skowron and C. Rauszer, "The Discernibility Matrices and Functions in Information Systems," in Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory, R. Slowinski, Ed., Theory and Decision Library, Series D, vol. 11. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1992, pp. 331-362, doi: 10.1007/978-94-015-7975-9_21.

[31] S. Jiang, R. Kumar, and H. E. Garcia, "Optimal Sensor Selection for Discrete-Event Systems with Partial Observation," IEEE Transactions on Automatic Control, vol. 48, no. 3, pp. 369-381, Mar. 2003, doi: 10.1109/TAC.2003.809144.

[32] A. Darwiche and A. Hirth, "On the Reasons Behind Decisions," in ECAI 2020, G. De Giacomo et al., Eds., Frontiers in Artificial Intelligence and Applications, vol. 325. Amsterdam, The Netherlands: IOS Press, 2020, pp. 712-720, doi: 10.3233/FAIA200158.

[33] A. Ignatiev, N. Narodytska, and J. Marques-Silva, "Abduction-Based Explanations for Machine Learning Models," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 1, pp. 1511-1519, 2019, doi: 10.1609/aaai.v33i01.33011511.

[34] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi, Reasoning About Knowledge. Cambridge, MA, USA: MIT Press, 1995, doi: 10.7551/mitpress/5803.001.0001.

[35] A. Baltag, "To Know is to Know the Value of a Variable," in Advances in Modal Logic, vol. 11, L. Beklemishev, S. Demri, and A. Mate, Eds. London, U.K.: College Publications, 2016, pp. 135-155. [Online]. Available: https://www.aiml.net/volumes/volume11/Baltag.pdf. Accessed: Aug. 24, 2026.

[36] SQLite Project, "SQLite Release 3.46.1 On 2024-08-13," Aug. 13, 2024. [Online]. Available: https://www.sqlite.org/releaselog/3_46_1.html. Accessed: Aug. 24, 2026.

[37] SQLite Project, "Isolation In SQLite," updated Apr. 18, 2022. [Online]. Available: https://www.sqlite.org/isolation.html. Accessed: Aug. 24, 2026.

[38] M. Lee, "Consequence Closure Inspector," Version 0.5.0, Zenodo, Aug. 2026, doi: 10.5281/zenodo.22095595.