Exhaustive Enumeration of Selection Observations in Pure Dataflow Graphs

A Survey and Unified Framework

Bili Dong[1]

Draft 2026-08-09 · landing page · survey record

Abstract

Different inputs can make a shared graph consult different selection nodes. We group allowed inputs when the requested outputs consult the same graph occurrences and obtain the same outcomes. For each group, the target is an exact input formula, a symbolic expression for the requested outputs, and one sample input. We call the grouping function the selection observer and each group an observation fiber. The base model is a finite, acyclic, deterministic pure graph with total primitives. This paper surveys the established approaches that can solve, compile, or specialize this task: guarded symbolic execution, projected model enumeration, decision structures, demand-guided search, geometric region traversal, and compositional summaries. Its main synthesis is a unified terminology and theoretical framework that separates the observer being enumerated from the discovery algorithm and output representation. Within that framework, enabled reachability determines which site outcomes are included; sparse event maps, observed-outcome guards, and totalized reachability-and-outcome coordinates then determine the same input partition. The comparison explains which approaches enumerate the same fibers directly, which require instrumentation or quotienting, which provide stronger guarantees on restricted instances, and which solve only adjacent reduction problems. The result is a problem-centered survey and semantic contract, not a claim of a new generic enumeration paradigm or practical speedup.

1. Introduction

Consider a finite, typed, acyclic, deterministic pure dataflow graph containing nested conditionals, multiplexers, priority selectors, or other finite choice operators. Its primitives are total on their typed domains. A caller supplies an input from a declared domain and requests one or more graph results. The problem of this survey is:

Enumerate every distinct selection observation induced by the requested results, exactly once, together with its exact input guard, residual value, and a witness.

A *selection observation* records the contextual outcome of each selection site reached from the requested roots through declared operand dependencies and selected case edges. It is sparse because a site in an unselected case cone is absent, not merely assigned an unconstrained value. Its inverse image in the caller domain is an *observation fiber*. An exact enumerator returns one record

$$(\text{observation}, \text{guard}, \text{residual}, \text{witness})$$

per nonempty fiber. The guard denotes precisely that fiber; the residual computes the requested values throughout it; and the witness demonstrates feasibility.

[1]The byline names the accountable human author, who directed and gated the work and takes responsibility for its content. OpenAI Codex systems (through GPT-5.6 Sol) provided substantial assistance with literature-search planning, evidence organization, cross-paper synthesis, formal presentation, manuscript drafting and editing, and repository tooling. Anthropic Claude systems (through Fable 5) provided substantial assistance with survey-record migration and manuscript review and revision. AI output is not treated as evidence; literature claims rest on the cited primary sources. The survey record linked in the title metadata documents the working evidence and synthesis trail.

The paper introduces *selection observation*, *selective term graph*, *enabled closure*, *observed-outcome guard*, *observation record*, and *full-fiber blocking* as local names or local refinements for this synthesis; they are not claimed as established terms of art, and the complete four-field contract is not presented as an already established named task. The paper is therefore also a theory/position synthesis of a proposed semantic contract, not evidence that this exact observer already has a demonstrated consumer. *Observer*, *kernel*, *fiber*, *guard*, *residual*, *projection*, and *symbolic execution* retain their established meanings. The six groups below are overlapping solution routes, not a field-wide taxonomy.

This task is easy to misidentify. A monolithic symbolic value describes output semantics but does not enumerate structural observations. A path records a control-flow history rather than the enabled portion of an arbitrary shared graph. A projected model fixes selected logical coordinates but needs explicit reachability instrumentation to represent structural absence. A partial cube may cover several complete observations. A geometric region can give an exact guard and affine residual while quotienting equal behavior or assuming that every site is observed. These objects can look alike while answering different questions.

The survey therefore separates three choices that are often conflated:

- the *observer*, which determines when two inputs belong to the same record;
- the *enumeration mechanism*, which discovers all nonempty fibers; and
- the *representation*, which stores guards, residuals, witnesses, and shared structure.

This separation makes six overlapping routes comparable: guarded symbolic execution, projected enumeration, compiled decision structures, demand-guided search, geometric or parametric specialization, and compositional summaries. Each route has strong precedents, but its name alone does not determine the observer or supply the complete record contract. Section 5 introduces the representative literature only where each route is analyzed.

The paper answers four questions:

- **RQ1:** What consistent terminology precisely defines selection-observation enumeration and distinguishes it from neighboring tasks?
- **RQ2:** Which established approach traditions or routes can enumerate, compile, or specialize the required fibers, and what instrumentation do they require?
- **RQ3:** Which correctness, representation, and complexity guarantees does each route provide under its stated assumptions?
- **RQ4:** Where do the approaches coincide, where do they compute a refinement or quotient of the target observer, and where do they address only an adjacent reduction problem?

The contribution is consequently a problem-centered survey and unified framework:

- A finite selective-term-graph model defines requested-root enabled closure, contextual site identity, sparse observations, exact fibers, and the output record contract.
- An observer-refinement order distinguishes structural non-observation, existential projection, logical don't-care, and equal-behavior quotienting without treating them as interchangeable forms of sparsity.
- Six recurring research traditions and implementation routes are compared under one set of semantic, algorithmic, representational, and complexity dimensions. They overlap rather than forming a flat taxonomy.
- Formal correspondences show that observed-outcome guards and global reachability-and-outcome projection describe the same partition, while guarded summaries agree with flattened graph substitution under explicit sharing and interface conditions.

The framework is a synthesis, not a priority claim for its ingredients. Its purpose is to state one problem precisely enough that results from different communities can be transferred only when their observers and assumptions actually agree.

Section 2 develops the problem through one graph. Section 3 then defines its core semantics independently of any solver, and Section 4 states the survey's evidence boundary. Section 5 presents the six solution routes, Section 6 gives two general enumeration presentations, and Section 7 states the compositional extension. Section 8 separates general costs from stronger specialized frontiers; Section 9 then compares observers, guarantees, and closest

established results. Section 10 states the remaining boundaries and open problems, and Section 11 closes with the main conclusions.

2. Selection observations by example

Let `p` and `r` be Boolean inputs and `x` and `y` be integer inputs. Consider the shared, pure graph

```
q_inner = select(r, x + 1, x + 1)
q_outer = select(p, q_inner, y)
return q_outer
```

Here the first case is selected when the Boolean selector is true, so we label the outcomes `left` (true) and `right` (false). Both arms of `q_inner` are intentionally equal. We request only `q_outer`.

A monolithic value encoding is immediate:

$$\mathrm{ite}(p, \mathrm{ite}(r, x+1, x+1), y).$$

It simplifies extensionally to $\mathrm{ite}(p, x+1, y)$. Neither expression states the structural partition we intend to enumerate. Write q_o and q_i for the outer and inner sites. Table 1 lists the three observations.

Observation	**Exact guard**	**Residual**
$q_o \to \mathrm{right}$	$\neg p$	y
$q_o \to \mathrm{left}, q_i \to \mathrm{right}$	$p \wedge \neg r$	$x+1$
$q_o \to \mathrm{left}, q_i \to \mathrm{left}$	$p \wedge r$	$x+1$

Table 1: Exact observations for the nested-selection example; witnesses are omitted from the table, and one is given below.

The example separates four notions.

First, `q_inner` is *structurally unobserved* when $\neg p$. Assigning it a wildcard in a Boolean implicant would be weaker: a wildcard may also cover executions in which the site is observed but its outcome is irrelevant to some other formula. The totalized observation uses a dedicated `unobserved` sentinel; the sparse API omits the coordinate.

Second, the two p-true records have equal residual expressions. They remain different because the observer records which outcome of `q_inner` was reached. A quotient by output value, residual function, or maximal affine behavior would merge them. That quotient is legitimate for a different objective but is not this observer.

This distinction has a concrete specification role even without assigning operational significance to evaluation order. As a stated hypothetical — no worked instance is developed in this survey — an event-aware validation task may ask whether every source selection outcome remains represented after graph rewriting, lowering, or component substitution (a task that itself requires a declared event correspondence, since such transformations can change the observer; see Section 10). The two inner outcomes then belong to different records despite equal values. A value-only equivalence checker deliberately chooses a coarser observer and merges them. The framework does not declare one policy universally preferable; it makes the policy an explicit part of the problem statement.

Third, the observed-outcome guards contain predicates only for sites actually observed in the corresponding record. The first guard does not say anything about `r`. Its exactness is structural: once `q_outer` chooses `y`, graph reachability cannot enter the cone containing `q_inner`. The later exact observed-outcome guard theorem generalizes this lockstep argument to arbitrary finite shared graphs and multi-case selections.

Fourth, one model per record is not the desired residual. For example, `p = true`, `r = false`, and `x = 0` witnesses the second record, but the residual valid on the whole fiber is $x+1$, not the sampled value 1. Exact enumeration must combine model discovery with symbolic specialization.

This nested example is intentionally different from a dense ReLU network or a flat collection of strict sign tests. When the caller domain excludes all test boundaries, each observation is a total activation vector and established

full-dimensional cell algorithms already enumerate the corresponding guards and affine maps. The unselected nested site is the minimal feature that exposes the graph-relative observation policy.

2.1. Framework at a glance

The concrete observer is easy to compute. Start at the requested root set. At an ordinary operation, walk to every operand. At a selection, always walk to its selector and only to the case roots enabled by that outcome. Record each selection occurrence reached by this backward walk and its outcome. Two inputs belong to the same observation fiber exactly when these records match.

The formal section gives names to the pieces: G is the graph, R the requested root set, A the caller-domain predicate, $D_{G(x,R)}$ the nodes reached by the walk, and $T_{G(x,R)}$ its selection record. For one feasible record τ, its fiber F_τ is the input group with record τ, while Γ_τ is an exact formula for that group. The rest of the paper asks how existing methods discover these groups and represent their formulas, residual output expressions, and sample inputs.

An implementation-oriented reading path is this example, the opening and comparison table of Section 5, the two constructions in Section 6, and Section 11. The intervening definitions and proofs make the contracts and transfer conditions precise.

3. Selection observations: formal model and semantics

This section defines the problem independently of any solver or data structure. The framework separates an observer, its inverse-image fibers, an enumeration mechanism, and an output representation. Its graph model is intentionally narrower than a general programming language: it isolates the finite, acyclic, deterministic, total pure-graph case for which exhaustive observation enumeration is a well-defined finite-output task.

3.1. Notation map

Table 2 collects the symbols used throughout the core definitions. A subscript G names the graph whose structure is being consulted; a fixed caller predicate A and requested-root set R determine the enumeration instance.

Symbol	Meaning	Role in the framework
$G = (V, E, I, O, Q, \lambda)$	Selective term graph: nodes, operand edges, inputs, output roots, selection sites, and labels	The finite semantic object being observed
$\mathcal{X}_G$, x, A, R	Typed input space, one input, caller-domain predicate, requested roots	Fix the inputs and outputs covered by one enumeration instance
Ω_q, κ_q, C_q, $h_{q,\omega}$	Outcomes, classifier, demanded cases, and selected combiner at site q	Determine an observed site's outcome and enabled case edges
val_x, $D_{G(x,R)}$, $T_{G(x,R)}$	Total graph value, enabled closure, and sparse selection observation	Evaluate the graph, expose requested dependencies, and record events
τ, F_τ, Γ_τ	One feasible observation, its input fiber, and its observed-outcome guard	Name one semantic bucket and an exact formula for it
$(\tau, \gamma_\tau, r_\tau, m_\tau)$	Observation, exact guard, residual, and witness	The required output record for one nonempty fiber

Table 2: Core notation. Later sections introduce only route-specific cost and composition symbols.

3.2. Selective term graphs

Definition 1 (selective term graph). A selective term graph is a tuple

$$G = (V, E, I, O, Q, \lambda),$$

where V is a finite node set; I, O, and Q are respectively input nodes, output roots, and selection-site nodes; $I \cap Q = \varnothing$; and $N = V \setminus (I \cup Q)$ is the set of ordinary nodes. Thus I, N, and Q partition V; an output root may belong to any category. Input nodes have no operand positions. The finite typed operand relation $E \subseteq V \times \mathbb{N}_0 \times V$ contains (v, j, u) when operand position j of consumer v is supplied by u. Operand positions are unique per consumer, edges are type-correct, and the consumer-to-operand relation is acyclic. Every ordinary node of arity k has exactly one operand edge at each position $1, \ldots, k$ and no other operand edges. For every selection q, its operand edges are exactly $(q, 0, s_q)$ and $(q, j, c_{q,j})$ for $1 \le j \le m_q$. The label map λ supplies result types and node semantics.

An input valuation belongs to the typed product $\mathcal{X}_G = \prod_{i \in I} \mathcal{D}_i$. An ordinary node $v \in N$ with operands $u_1, \ldots, u_k$ denotes a deterministic total function

$$f_v : \mathcal{D}_{u_1} \times \ldots \times \mathcal{D}_{u_k} \to \mathcal{D}_v.$$

Ordinary nodes follow an *all-operands observation policy*: observing their result observes every operand. This is a declared structural dependency policy, not an operational evaluation order and not an assertion that f_v depends extensionally on every argument.

For readers used to runtime dataflow diagrams, the edge direction here may look backward: an edge points from a consumer to one of its operands. This lets reachability start at the requested results and walk directly toward everything they declare as a dependency.

Definition 2 (generalized selection site). A selection site $q \in Q$ consists of a selector operand s_q, case-root operands $c_{q,1}, \ldots, c_{q,m_q}$, a finite outcome set Ω_q, and a total classifier

$$\kappa_q : \mathcal{D}_{s_q} \to \Omega_q.$$

A case-demand function $C_{q(\omega)} \subseteq \{1, \ldots, m_q\}$ states which case roots are observed for outcome ω. A typed total combiner

$$h_{q,\omega} : \prod_{j \in C_{q(\omega)}} \mathcal{D}_{c_{q,j}} \to \mathcal{D}_q$$

constructs the result from those cases in increasing operand order.

The outcome is semantic, not necessarily the raw selector value. It must carry enough information to determine both the demanded case set and the applicable combiner. An indexed selection may have one outcome per case and one default; a priority selection may identify the first enabled case; and a mask-valued selection may use the complete enabled-case mask as one outcome and demand several roots. If an interface intends to distinguish two raw values, they must be distinct outcomes even when they choose the same cases.

Because the graph is acyclic and every primitive is total, each input x induces a unique *total graph value* $\mathrm{val}_{x(v)}$ at every node. At a selection site, $\omega_{x(q)} = \kappa_{q\left(\mathrm{val}_{x(s_q)}\right)}$, and the value is obtained by applying $h_{q,\omega_{x(q)}}$ to the selected case-root values. Values in unselected case cones remain mathematically defined; the observation judgment below determines which of them are structurally exposed.

3.3. Contextual identity and caller domain

A site inside one graph is identified by its node. Graph substitution prefixes internal sites by the component occurrence, so two calls to the same callee have different identities. Statically bounded iteration also includes the iteration index. This policy preserves sharing within an occurrence while excluding unbounded recursion and loops, which could create an infinite event domain.

The observation is relative to a requested root set $R \subseteq O$ and a caller-domain predicate $A : \mathcal{X}_G \to \mathbb{B}$. All feasibility and completeness claims range over $\mathcal{X}_A = \{x \in \mathcal{X}_G \mid A(x)\}$. An open multi-output component therefore takes R as an interface parameter; observing all outputs unconditionally would expose sites that a caller does not request.

3.4. Enabled closure and selection observation

Definition 3 (enabled operand edge). Fix an input x. Every operand edge of an ordinary node is enabled. Every selector edge $(q, 0, s_q)$ is enabled. A case edge $(q, j, c_{q,j})$ is enabled exactly when $j \in C_{q(\omega_{x(q)})}$. Write E_x for this input-indexed edge relation.

Definition 4 (enabled closure). For $R \subseteq O$, $D_{G(x,R)}$ is the least node set containing R and closed from a consumer to each operand along E_x. Equivalently, it is reachability from R in the enabled consumer-to-operand graph. A selection site is *observed for* (x, R) exactly when it belongs to $D_{G(x,R)}$.

The word *observed* is local terminology for this membership judgment. It does not claim that the source evaluator is lazy. Classical neededness and demand semantics use related backward relevance but impose different semantic conditions; Section 5 compares them.

Definition 5 (selection observation). The selection observation is the dependent finite partial map $T_{G(x,R)}$ —a typed dictionary in which key q can store only an outcome from Ω_q— with

$$\operatorname{dom}\left(T_{G(x,R)}\right) = Q \cap D_{G(x,R)}, \quad T_{G(x,R)}(q) = \omega_{x(q)}.$$

Its totalized presentation uses one fresh sentinel $\text{unobs}_q \notin \Omega_q$ per site:

$$T^{\text{tot}}_{G(x,R)}(q) = \begin{cases} T_{G(x,R)}(q) & \text{if } \ q \in D_{G(x,R)} \\ \text{unobs}_q & \text{otherwise.} \end{cases}$$

For a fixed graph, the partial-map and totalized-vector presentations are isomorphic. The first is the intended sparse record; the second permits direct projection onto the finite product $\prod_{q \in Q} \left(\Omega_q \cup \{\text{unobs}_q\}\right)$.

3.5. Observers, refinements, and representations

Definition 6 (observer and refinement). On a fixed caller domain $\mathcal{X}_A$, an *observer* is a total function $B : \mathcal{X}_A \to Y$. Its observable equivalence is its kernel

$$\ker(B) = \{(x, y) \in \mathcal{X}_A^2 \mid B(x) = B(y)\}.$$

Observer B_1 *refines* observer B_2 when $\ker(B_1) \subseteq \ker(B_2)$. Equivalently, there is a unique factor map $h : \text{im}(B_1) \to \text{im}(B_2)$ such that $B_2(x) = h(B_1(x))$ for every x. Thus B_1 preserves at least every distinction preserved by B_2.

In plain language, an observer is the function used to sort inputs into buckets. A *fiber* is one bucket. Refinement means that knowing the finer bucket is enough to recover the coarser one. A residual is a formula or shared expression that computes the requested result for every input in one bucket.

For fixed G and R, $T_{G(-,R)}$ is the *selection observer*. Its sparse and totalized presentations have the same kernel and are therefore two representations of one observer. By contrast, projecting away site coordinates or coalescing records with equal residual functions generally produces a coarser observer. A control-flow path or execution trace may be finer, and it may also preserve distinctions incomparable with requested-root selection events.

This vocabulary separates semantic objects from encodings. A flat list of guards, a decision tree, a decision diagram, and a projected-model stream may represent the same observer image. A partial cube can instead represent a set of image elements. Calling all of these outputs “partial decisions” obscures whether the underlying partition has changed.

Proposition 1 (determinacy, finiteness, and sharing). For every x and R, $D_{G(x,R)}$ and $T_{G(x,R)}$ are unique, and

$$\left|\left\{T_{G(x,R)} \mid x \in \mathcal{X}_G\right\}\right| \leq \prod_{q \in Q}(1 + |\Omega_q|).$$

If $R_1 \subseteq R_2$, the smaller observation is the restriction of the larger. Moreover,

$$D_{G(x,R_1 \cup R_2)} = D_{G(x,R_1)} \cup D_{G(x,R_2)}$$

and the observation at the union is the compatible partial-map union of the two observations. A shared site therefore occurs once.

Proof. Total graph values and outcomes are unique, and finite reachability has one least closure. The product independently counts one unobserved sentinel or one of the outcomes at each site. Reachability from a union of roots is the union of reachability from each root in the same enabled graph; site outcomes are fixed by x, so the partial maps agree on their shared domain. □

3.6. Least valuation under the declared dependency policy

Extend each value domain with a fresh undefined value $\bot_v$ and order it flatly, $\bot_v \leq d$ for every ordinary value d, with distinct ordinary values incomparable. Order valuations pointwise. An x-consistent partial valuation assigns each node either $\bot_v$ or its total graph value. It is dependency-closed when a defined ordinary node has all operands defined and a defined selection has its selector and every case root required by its concrete outcome defined. This is an obligation, not a prohibition: another requested root or shared consumer may independently require an unselected case root. The valuation is R-complete when every requested root is defined.

Proposition 2 (declared-dependency least valuation). The valuation

$$\nu^*_{x,R}(v) = \begin{cases} \mathrm{val}_{x(v)} & \text{if } \ v \in D_{G(x,R)} \\ \bot_v & \text{otherwise} \end{cases}$$

is the unique least x-consistent, dependency-closed, R-complete valuation. Its support is exactly $D_{G(x,R)}$, and projecting the outcomes of its defined selection sites yields $T_{G(x,R)}$.

Proof. The enabled closure contains R and satisfies exactly the declared predecessor obligations, so $\nu^*_{x,R}$ is admissible. The support of any other admissible valuation is a set containing R and closed under the same enabled edges; least reachability therefore puts $D_{G(x,R)}$ inside that support. Consistency fixes every defined ordinary value, giving the pointwise order and uniqueness. □

This proposition is a graph-reachability result for the stated all-operands policy at ordinary nodes. Equality with a language's semantic least demand requires a separate translation showing that each source operator has precisely these predecessor obligations.

3.7. Fibers and exact observed-outcome guards

Fix G, A, and R. The feasible observation image and the fiber of one τ in that image are

$$\mathcal{T}_{G,A,R} = \left\{T_{G(x,R)} \mid A(x)\right\}, \quad F_\tau = \left\{x \mid A(x) \wedge T_{G(x,R)} = \tau\right\}.$$

Because $T_{G(-,R)}$ is a total function on $\mathcal{X}_A$, its nonempty fibers are pairwise disjoint and cover the caller domain. This elementary inverse-image partition is the semantic enumeration contract.

Definition 7 (exact selection-observation enumeration). An exact solution emits one record $(\tau, \gamma_\tau, r_\tau, m_\tau)$ for every $\tau \in \mathcal{T}_{G,A,R}$ and no other record. Its semantic fields have types $\gamma_\tau : \mathcal{X}_G \to \mathbb{B}$, $r_\tau : \mathcal{X}_G \to \prod_{o \in R} \mathcal{D}_o$, and $m_\tau \in \mathcal{X}_G$. For every typed input x they satisfy

$$\gamma_{\tau(x)} \Leftrightarrow A(x) \wedge T_{G(x,R)} = \tau;$$

$\gamma_{\tau(m_\tau)}$; and whenever $\gamma_{\tau(x)}$ holds,

$$r_{\tau(x)} = \mathrm{val}_x|_R.$$

The four obligations are respectively exhaustive duplicate-free indexing, exact fiber representation, a feasibility witness, and residual correctness throughout the fiber.

These are semantic functions. A concrete implementation may encode a guard as an SMT formula and a residual as a term DAG, provided evaluation has the stated meaning; the contract does not require tabulating either function.

For each site outcome, define

$$\chi_{q,\omega}(d) \Leftrightarrow \kappa_{q(d)} = \omega, \quad p_{q,\omega}(x) \Leftrightarrow \chi_{q,\omega}\big(\mathrm{val}_{x(s_q)}\big).$$

For feasible τ, its *observed-outcome guard* is

$$\Gamma_{\tau(x)} = A(x) \wedge \bigwedge_{q \in \mathrm{dom}(\tau)} p_{q,\tau(q)}(x),$$

It mentions outcomes of observed sites but contains no outcome or absence atom for an unobserved site. The name is semantic rather than syntactic: an outcome predicate may contain negation or an inequality, and expanding it may traverse nested selection expressions. The demanded generator in Section 6 avoids constructing unobserved case cones.

Theorem 1 (exact observed-outcome guard). For every feasible observation τ and every input x,

$$\Gamma_{\tau(x)} \Leftrightarrow A(x) \wedge T_{G(x,R)} = \tau.$$

Proof. The reverse implication follows from the definition. For the forward direction, choose a witness $m \in F_\tau$. Build enabled closures for m and x by the same finite sequence of reachability approximants, beginning at R. Suppose the current approximants agree. Ordinary nodes expose identical operand positions. A selection q in the common approximant is observed at m, hence $q \in \mathrm{dom}(\tau)$; the corresponding conjunct gives $\omega_{x(q)} = \tau(q) = \omega_{m(q)}$. Both evaluations therefore expose the same selector edge and the same case roots $C_{q(\tau(q))}$. Induction makes every approximant, and thus both least closures, equal. Their observed-site domains coincide, and every outcome in that domain is fixed by τ. □

The result relies on each outcome determining its case-demand set. It proves exactness, not literal minimality: an observed-site predicate may still be entailed by A and the remaining predicates.

Theorem 2 (conflict frontier). If $\tau \neq \sigma$ are feasible observations, then some site is observed in both with different outcomes:

$$\exists q \in \mathrm{dom}(\tau) \cap \mathrm{dom}(\sigma) : \tau(q) \neq \sigma(q).$$

Proof. Assume instead that both maps agree wherever both are defined. Their closure approximants begin at the same roots. Ordinary nodes expose the same operands, and every selection in a common approximant has the same outcome, so it exposes the same case roots. The approximants remain equal, yielding equal domains and equal maps, a contradiction. □

Thus two observed-outcome guards are disjoint because they contain incompatible outcome predicates at a shared observed site. This structural separator is stronger than the generic fact that distinct function fibers are disjoint and is the key reason full-fiber blocking does not need explicit unobserved-site literals.

4. Survey scope and evidence basis

This is a problem-centered scoping survey supported by a reproducible systematic search record and evidence map; it is neither a closed systematic map nor a statistical systematic literature review. Discovery and reporting draw on systematic-mapping guidance (Petersen et al., 2015), separate backward and forward snowballing (Wohlin, 2014), and auditable reporting of secondary studies (Kitchenham et al., 2023); no completed reporting checklist is claimed. It supports the comparison of solution mechanisms for the problem in Section 3, not a census of every use of symbolic execution, dataflow, or enumeration.

4.1. Comparison and inclusion

The four research questions in the introduction determine the extraction schema. For each work, the map compares its model and observer, enumerated object, discovery algorithm, output representation, guarantees, complexity charge, and semantic or solver assumptions.

Works in the main comparison have one of four relationships to the target problem. A *direct presentation* enumerates the same observer after explicit notation or reachability instrumentation. A *restricted specialization* solves the same contract under stronger assumptions, such as all sites being observed and all classifiers being affine. An *adjacent comparator* preserves or omits information for a different observer and is included only when it clarifies a semantic or complexity boundary. An *open correspondence* is close enough that a reduction may exist, but this survey has not proved observer and record equality. The classification prevents a neighboring reduction problem from being presented as an alternative implementation of an input-fiber enumerator.

4.2. Discovery, screening, and technical evidence

Discovery deliberately used a broader vocabulary than the final paper. It covered symbolic execution and guarded values; projected and partial model enumeration; decision structures; functional-logic and demand-guided search; dataflow and hardware semantics; geometric, neural, and parametric regions; and observer-relative state or search reduction. Exact queries, result depths, dates, and citation chases are retained in the audited log. Transient result sets are discarded after reconciliation, so the log row is the audit unit; primary bibliographies supplement incomplete citation indexes.

Through 9 August 2026, 340 database queries and 245 backward or forward citation chases yielded 36,828 retrieved and title-triaged record occurrences. Repeated hits and rows rejected before cataloging remain in that occurrence count; it is neither a unique-paper count nor an estimate of a literature population. After deduplication and audit, the current catalog contains 924 works: 120 `deep-read`, 247 `screened`, 422 retained `candidate`, and 135 `excluded` records. Candidate status is an unresolved abstract-screening backlog rather than evidence inclusion or a promise to deep-read the work. The manuscript cites 79 deep-read technical sources and seven screened works; no candidate-status work supports a manuscript claim.

Technical claims rely on primary works with pinpoint definition, algorithm, theorem, complexity, assumption, or example anchors. Abstract- or metadata-level records support only search scope or qualified chronology. The evidence ledger binds manuscript claims to those source anchors and carries their scope and caveats.

4.3. Bounded search snapshot and on-demand updates

The initial campaign was previously described as having *bounded mapping closure*. Publication review withdrew that claim: although every row had a catalog status, 422 rows were still `candidate`, which is not an adjudicated inclusion/exclusion disposition under the clarified protocol. The present artifact is therefore a *bounded search snapshot* relative to the recorded sources, rankings, query depths, and date. Its reviewed evidence supports the comparisons made here, but the unresolved backlog prevents an inference that all plausible close work in the captured set has been assessed.

Updates run on demand when a new mechanism or plausible direct competitor appears and to reduce the candidate backlog. The registered searches were last run through 9 August 2026, when 22 registered queries were rerun at caps of the top 100 relevance- or recency-ranked results. No candidate-status work supports a claim, and no absence claim is drawn from the unresolved rows. The protocol, queries, catalog, log, source notes, syntheses, and evidence and claim ledgers are linked from the survey's landing page.

4.4. Validity threats and AI assistance

The breadth-first discovery campaign may miss work whose terminology does not intersect the registered queries or citation neighborhoods. Database rankings are opaque, metadata services omit or merge records, and exact counts describe captured occurrences rather than a population. The framework itself can bias classification: a theorem proved for a different observer must not be silently transferred after adding instrumentation or changing the output quotient.

Search, extraction, and duplicate-screening used repeated AI-assisted passes under the same project framing. They are repeated checks, not independent human reviews; no inter-rater agreement statistic was computed. The title-page note gives the full authorship, assistance, evidence-use, and public-record disclosure.

5. Solution routes

The target is one exact selection-observation record per nonempty caller-input fiber. Prior approaches become comparable only after fixing that contract. Some can enumerate the fibers directly after instrumentation, some compile the same finite observer into a shared representation, and some solve restricted special cases with stronger guarantees. Table 3 summarizes six recurring, non-exclusive research traditions and implementation routes. They mix mechanisms, representations, and restrictions rather than forming a flat taxonomy.

Route	Discovery object	Natural output	Route to target	Principal boundary
Guarded symbolic execution	Feasible path or merged symbolic state	Guarded residual	Log demanded site outcomes	Paths may refine or cross the target partition
Projected model enumeration	Selected coordinate image	Models, cubes, or compiled cover	Project totalized reachability/outcome coordinates	Cubes may group observations; projection supplies no residual
Decision structures	Compiled finite observer	Tree, BDD, ADD, or related DAG	Compile the totalized observer and residual labels	Compilation size and variable order can dominate
Demand-guided search	Forced inputs, choices, or values	Partial map or value stream	Align demand with requested-root enabled closure	Often lacks exact fiber guards or residuals
Geometric/parametric enumeration	Cells, modes, or critical regions	Polyhedral guard and affine map	Dense signs: direct; critical regions: correspondence needed	Different observers; affine and dimensional assumptions
Compositional summaries	Component relation or guarded pieces	Reusable guarded summary	Parameterize summaries by requested outputs	Exact composition does not imply compact reuse

Table 3: Recurring solution routes expressed in the unified framework.

5.1. Guarded symbolic execution

Classical symbolic execution associates path conditions with symbolic states or substitutions (King, 1976), while DART established the later concrete-plus-symbolic, solver-directed test-generation lineage (Godefroid et al., 2005). Denotational treatments make this correspondence exact under their stated language and merge conditions (Voogd et al., 2025), while multi-path execution merges paths into guarded symbolic values (Sen et al., 2015). Variational execution similarly carries conditional values under configuration contexts and shares computation across many configurations, but its native result is a shared multi-configuration execution rather than one exact inverse-fiber record (Wong et al., 2018). Solver-aided libraries and reusable merging semantics preserve the same basic separation between guards and residual values (Lu & Bodík, 2023; Porncharoenwase et al., 2022).

These systems supply two parts of the target record almost directly: a feasibility guard and a residual. They do not by themselves choose the selection observer. Conventional paths may distinguish branches outside the requested enabled closure, merge histories that retain different requested events, or split one observation because of unrelated control flow. To solve the target problem, evaluation must be demanded from the requested roots, record contextual selection outcomes, preserve graph sharing, and block the entire resulting fiber rather than one execution model.

The closest published algorithms make this gap narrow. PESO enumerates reordered relevant-slice conditions for requested output criteria, carries symbolic outputs and solver-generated tests, and proves conditional exploration completeness under a sound and complete solver (Qi et al., 2013). SPD constructs a shared graph of dependence-relevant path families and guarded symbolic values for queried uses (Santelices & Harrold, 2010). All-values and dependence-guided execution provide further output/value-directed precedents (Denaro, 2012; Wang et al.,

2017). What remains open is not whether output-directed guarded exploration exists, but whether PESO's RSC quotient or SPD's path-family graph, after site instrumentation, gives exactly one target selection fiber rather than a refinement or fragmented cover.

SEDGE is the closest explicitly dataflow-named concolic testing comparator: it uses SMT to synthesize high-level Pig inputs intended to exercise operator cases. Its coverage target and accumulated example dataset are not a duplicate-free exact partition with residuals (Li et al., 2013).

The local generator formalized in Section 6 is therefore not a new symbolic execution paradigm. It is the target observer instantiated in a standard guard-and-residual evaluator. Its useful property is the exact observed-outcome guard: structurally unobserved sites require no absence literal because enabled reachability is already fixed by the observed outcomes.

5.2. Projected model enumeration

Phan's AllSMT enumerates satisfying assignments to designated important Boolean coordinates while returning sampled values for relevant theory variables (Phan & Malacaria, 2015). A finite theory-valued coordinate needs an engine whose projection contract explicitly enumerates its values. Recent projected SAT and SMT methods can emit disjoint partial models and avoid a growing family of ordinary blocking clauses (Spallitta et al., 2024; 2025). Knowledge compilation similarly supports disjoint partial-model enumeration after d-DNNF compilation (Lagniez & Lonca, 2024).

This is the most direct generic reduction. Give every contextual selection site a finite coordinate whose values are its outcomes plus an explicit `unobserved` sentinel. Reachability equations connect that sentinel to requested-root reachability. Projecting the graph formula onto those coordinates then enumerates exactly the totalized selection observer. The construction is conceptually complete but charges the whole-graph value, classifier, case-membership, and reachability encodings.

The output contract still matters. Enumerating every complete projected tuple produces one element per selection observation. A short partial cube can cover many tuples and is therefore a compact cover of the observer image rather than the requested record stream. Disjoint short-model methods make such covers precise, but a consumer that requires one residual and witness per complete observation must refine or annotate the cubes accordingly.

5.3. Compiled decision structures

A decision tree asks only its representation tests along a root-to-leaf route. This is not the same as a graph selection site being unobserved: a compiler may choose entirely different input predicates. Reduced BDDs share Boolean subfunctions canonically under a fixed variable order (Bryant, 1986), and ADDs extend terminals beyond Boolean values (Bahar et al., 1997). Finite observer partitions can also be generated directly as exact input-equivalence classes or satisfiable atoms of a declared observation alphabet (Huang et al., 2024; Krafczyk & Peleska, 2017).

For finite encoded inputs, compiling the function $x \mapsto T^{\mathrm{tot}}_{G(x,R)}$ yields an exact representation of the target partition. Leaves or terminals may additionally carry residual identifiers and witnesses. A tree exposes the sparse sequence of tests made along one route; a diagram exposes shared predicates and subfunctions across many routes. Either can be exponentially smaller or larger than a flat guard list, so a comparison must state whether the output is a stream of records or one shared compiled object.

Compilation also exposes a semantic choice. Reducing nodes that have equal successors preserves the compiled observer, but compiling only the requested output value may erase equal-valued selection events. Neural decision-tree extraction and affine decision structures make this contrast concrete. Nguyen et al. construct EC-DTs and report exact empirical fidelity with contradiction pruning (Nguyen et al., 2020); Affinitree proves semantic preservation and simplifies infeasible or entailed tests (Schlüter & Steffen, 2024). They solve the selection-observation task only when their terminals or internal labels retain the declared selection events.

5.4. Demand-guided evaluation and search

Functional-logic set functions and pull-tabbing establish the representation lineage most directly: set functions record the nondeterministic steps actually executed, the 2010 pull-tab transformation propagates immutable choice identifiers so runtime copies make consistent decisions, and memoized pull-tabbing finally exposes an explicit task-local partial choice map (Alqaddoumi et al., 2010; Antoy & Hanus, 2009; Hanus & Teegen, 2021). Other

pull-tabbing, translations, and memoization use demand-populated decision maps or prove value-set preservation under their own search assumptions (Antoy, 2011; Braßel, 2011; Braßel & Huch, 2007; Hanus & Teegen, 2021; Jost, 2023). Lazy SmallCheck refines just the partial input demanded by a Boolean observation and remains exhaustive over its bounded domain (Runciman et al., 2008). SPLat similarly discovers configuration variables on first read, uses SAT to prune feature-model-infeasible partial assignments, and executes concrete witnesses for distinct claimed test traces (Kim et al., 2013). Classical dataflow analyses compute least or reverse demand for a fixed requested result (Avron & Sasson, 1994; Pingali & Arvind, 1985), and modern bidirectional demand semantics can characterize minimal sufficient partial inputs (Xia et al., 2024).

These results establish that stable sparse choice maps, dynamically discovered configuration decisions, and requested-result demand are not new. Their natural output, however, is commonly a value stream, a partial input, a choice fingerprint, a concrete test trace, or a fixed-input demand set. The target enumerator additionally requires the complete inverse-image guard and a residual valid over that entire guard. A demand system becomes a direct solution only after its demand judgment is proved equal to the enabled closure and its fair search is grouped by complete observation fibers.

The comparison also prevents a terminology error. The base graph is not called lazy: ordinary nodes expose all operands under a declared observation policy, while selection sites demand only the cases chosen by their outcomes. “Demand” here is graph-relative support for a requested observer, not an operational evaluation strategy.

5.5. Geometric and parametric specializations

When every selection is observed, each classifier is the strict sign of a distinct nonconstant affine form, and the caller domain is the ambient space with all classifier boundaries removed, a complete observation is a sign vector and each nonempty fiber is a full-dimensional hyperplane-arrangement cell. Reverse-search and incremental algorithms enumerate those cells exactly with output-sensitive guarantees (Avis & Fukuda, 1996; Ferrez et al., 2005; Rada & Černý, 2018). Exact ReLU analyses likewise enumerate activation patterns or polyhedral regions and often attach affine output maps (Serra et al., 2018; Vincent & Schwager, 2021); star-set reachability and explicit piecewise-affine conversion emit exact guards with affine images, and cell-complex, edge-subdivision, and parallel layerwise variants recover richer exact structure (Berzins, 2023; Drammis et al., 2024; Masden, 2025; Robinson et al., 2020; Tran et al., 2019).

Multiparametric programming provides a closely related guard-and-residual contract. Critical-region algorithms emit polyhedral parameter guards together with affine optimizers and have explicit LP-oracle-relative bounds, building on the positive-semidefinite pLCP precursor (Columbano et al., 2009; Jones & Maciejowski, 2006; Jones & Morari, 2006). Piecewise-affine systems compose upstream affine maps into downstream guards and can minimize regions with equal behavior relative to a supplied arrangement and representation class (Geyer et al., 2008; 2010).

The full-dimensional hyperplane and dense activation-pattern methods are direct specializations under the stated observer and domain restrictions. Assigning boundary points to a side produces closed or lower-dimensional strata and requires separate face or ownership machinery; the cited open-cell guarantees do not transfer automatically. Parametric critical regions instead observe an optimizer basis or active set. They are direct only after an explicit model shows that this identity is exactly the graph’s selection observer; otherwise they are strong guard-and-residual comparators. The full-dimensional cell algorithms analyzed here no longer instantiate the target when nested selections make sites unobserved, the caller predicate cuts across cells, or equal affine maps are merged despite different observed outcomes.

5.6. Compositional guarded summaries

Compositional symbolic execution summarizes component behavior with guarded relations, preconditions, post-conditions, or path fragments (Anand et al., 2008; Godefroid, 2007; Voogd et al., 2025). Guarded piecewise-affine composition substitutes an upstream residual into downstream guards and residuals while discarding infeasible conjunctions (Geyer et al., 2010). Selective computations supply an abstract interface for statically visible, dynamically chosen effects (Mokhov et al., 2019).

The unified framework adds two parameters needed by the target observer: requested output roots determine boundary demand, and contextual prefixes distinguish multiple component occurrences while retaining sharing

within an occurrence. Under full-domain component summaries, guard substitution and namespaced observation union agree with flattened graph evaluation. This is an exact compositional presentation of the same partition.

Exactness does not imply compactness. A component with many outputs can require one summary family per demand mask, caller predicates can split component fibers differently at different occurrences, and residual substitution can duplicate large terms unless sharing is retained. Compositional summaries are therefore a representation and reuse strategy, not a general improvement in enumeration complexity.

These routes overlap rather than form a ranking. Section 9 compares their observers, guarantees, applicability, and closest established boundaries.

6. Full-fiber blocking and projected enumeration

The survey identified two general routes to enumerating the observer image. A local presentation discovers one complete fiber from a model, constructs its residual, and blocks the fiber. A global presentation constructs a whole-graph encoding and projects a conventional model enumerator onto totalized observation coordinates. This section states both in the unified terminology and proves that they enumerate the same observer. The constructions are comparison baselines, not claims of new generic enumeration paradigms. Projection directly supplies the observation index and, with model production, a witness and an exact fiber predicate obtained by fixing the projected tuple. It does not by itself construct a residual; satisfying the full record contract requires exact symbolic specialization of the requested values under the fiber guard, such as the demanded symbolic evaluator used by the local presentation. This paper calls that construction *residualization*; it is not a primitive operation supplied by an SMT solver.

6.1. Symbolic and solver assumptions

Every primitive, classifier outcome, case-membership relation, selected combiner, caller-domain predicate A, and typed input-domain constraint must have an exact symbolic representation. A model-producing oracle for a formula returns `sat(m)`, `unsat`, or `unknown`. Only `unsat` certifies exhaustion; `unknown` produces an explicit incomplete result. We do not assume that the oracle runs in polynomial time or that a decision-only oracle returns a witness for free.

Let $\mathrm{Dom}_{G(x)}$ be the symbolic encoding of the typed input product and write

$$A_{\mathrm{enc}}(x) = \mathrm{Dom}_{G(x)} \wedge A(x).$$

Every solver query below uses A_{enc}. This is explicit because a bit-vector or integer lowering can otherwise admit codes that do not denote a typed graph input.

For an input model $m \in \mathcal{X}_A$, a demanded evaluator carries three objects:

- a conjunction g of observed outcome predicates;
- a partial observation map T; and
- a candidate-local memo μ from graph-node identities to pairs of concrete and symbolic values.

The memo is essential for shared DAGs. Memoizing syntax-tree positions could duplicate one source site or allow inconsistent outcomes.

6.2. Concolic exact-fiber generation

The construction uses the standard DART-style pattern of one concrete model guiding a simultaneous symbolic evaluation (Godefroid et al., 2005); its additional obligation is to generalize that model to the entire exact observation fiber.

Evaluation of a demanded node v returns a concrete value c_v at m and a symbolic residual e_v:

1. An input returns its component of m and its symbolic variable.
2. An ordinary node demands all operands and applies the exact concrete and symbolic primitive.
3. A selection q first evaluates its selector, including any selections in that cone. It computes $\omega = \kappa_{q\left(c_{s_q}\right)}$, records $T(q) = \omega$ once, conjoins $\chi_{q,\omega}\left(e_{s_q}\right)$, evaluates exactly the case roots in $C_{q(\omega)}$, and applies the corresponding symbolic combiner.

4. A memo hit reuses the pair and records no duplicate event.

Only selection outcomes are specialized to m. Replacing an ordinary symbolic value by its sampled concrete value is unsound: it typically weakens the outcome guard while overspecializing the residual. Write

$$\mathrm{Gen}(G, m, R) = (T_m, g_m, r_m),$$

where r_m is the residual tuple for the requested roots and the encoded complete guard is $A_{\mathrm{enc}} \wedge g_m$.

Theorem 3 (generator correctness). Under the symbolic assumptions above and memoization by node identity,

$$x \models A_{\mathrm{enc}} \wedge g_m \Leftrightarrow x \in \mathcal{X}_G \wedge A(x) \wedge T_{G(x,R)} = T_m,$$

and every input satisfying that guard satisfies

$$\mathrm{eval}(r_m, x) = \mathrm{val}_x|_R.$$

The model m itself satisfies the emitted guard.

Proof. View demanded evaluation as a state transformer. A call entered with guard g_{in} returns a guard g_{out} such that g_{out} implies g_{in}, $m \models g_{\mathrm{out}}$, the returned concrete value is $\mathrm{val}_{m(v)}$, the residual evaluates to that value at m, and for every $x \models A_{\mathrm{enc}} \wedge g_{\mathrm{out}}$ the returned residual e_v evaluates to the total graph value of v at x. Guards grow monotonically; consequently, a memoized residual remains valid whenever it is reused because the current guard implies its creation guard. Ordinary primitives preserve this invariant by exactness. At a selection, first evaluating the selector establishes its own post-guard. The new predicate $\chi_{q,\omega}\left(e_{s_q}\right)$ is therefore equivalent under the strengthened guard to the global outcome predicate $p_{q,\omega}(x)$ and selects the same case roots. The complete guard $A_{\mathrm{enc}} \wedge g_m$ is therefore semantically equivalent to Γ_{T_m} from the exact observed-outcome guard theorem, while the residual invariant gives value correctness. Every chosen outcome predicate holds at m, so the guard is nonempty. □

6.3. Full-fiber blocking

Initialize the uncovered formula $U_0 = A_{\mathrm{enc}}$ and repeat:

1. Query the model oracle for U_j.
2. On `unknown`, return all accumulated records marked incomplete.
3. On `unsat`, return them marked complete.
4. On `sat(m_j)`, compute $\left(T_j, g_j, r_j\right) = \mathrm{Gen}\left(G, m_j, R\right)$ and emit $\left(T_j, A_{\mathrm{enc}} \wedge g_j, r_j, m_j\right)$.
5. Set $U_{j+1} = U_j \wedge \neg g_j$.

Previous blockers are not folded into an emitted guard. Every query already assumes A_{enc}, so the incremental blocker need only negate g_j.

Theorem 4 (complete duplicate-free enumeration). Let $K = \left|\mathcal{T}_{G,A,R}\right|$. If the model-producing oracle decides every query, full-fiber blocking performs exactly K satisfiable invocations and one final unsatisfiable invocation. It emits every feasible observation once, with its exact guard, correct residual, and a witness.

Proof. Every model lies in one nonempty fiber, and generator correctness emits that entire fiber. Its blocker prevents repetition and, because distinct fibers are disjoint, removes no other observation. Before all fibers are blocked, an input in an unblocked fiber satisfies U_j. Once all K are blocked, the fiber-partition property makes U_K unsatisfiable. □

This is a $K + 1$ *model-producing invocation count under a unit-cost oracle.* It is not an OutputP, IncP, DelayP, wall-clock, or decision-oracle theorem. Formula construction, serialized guard and residual size, solver work, and the final exhaustion query must all be charged separately.

6.4. Global reachability-and-outcome encoding

The alternative presentation symbolically evaluates every node, including unobserved case cones. Let $e_{v(x)}$ be the exact whole-graph symbolic value of v. For each case position define the direct predicate

$$\eta_{q,j}(d) \Leftrightarrow j \in C_{q\left(\kappa_{q(d)}\right)}.$$

Introduce one Boolean reachability indicator a_v per node. It is true exactly when v is reachable from a requested root through an enabled consumer edge. The acyclic defining equations are

$$a_v \Leftrightarrow [v \in R] \vee \left(\vee_{(w,j,v)\in E, w\in N}\, a_w\right) \vee \left(\vee_{q:v=s_q}\, a_q\right) \vee \left(\vee_{q,j:v=c_{q,j}}\, \left(a_q \wedge \eta_{q,j}\left(e_{s_q}\right)\right)\right).$$

Multiple consumers contribute by disjunction, and the biconditional prevents spurious disconnected reachability. Acyclicity gives a unique valuation of the reachability circuit for every input.

For each selection site introduce a projected coordinate ranging directly over $\Omega_q \cup \left\{\mathrm{unobs}_q\right\}$:

$$z_q = \begin{cases} \kappa_{q\left(e_{s_q}\right)} & \text{if } \ a_q \\ \mathrm{unobs}_q & \text{otherwise,} \end{cases}$$

Let $\Phi_{G,A,R}(x, a, Z)$ conjoin $A_{\mathrm{enc}}(x)$, every displayed reachability biconditional, and every displayed coordinate definition. Project $\Phi_{G,A,R}$ onto $Z = \left(z_q\right)_{q\in Q}$, so its projected image is

$$\left\{Z \mid \exists x, a : \Phi_{G,A,R}(x, a, Z)\right\}.$$

An exact projected enumerator or finite decision-diagram compiler can then enumerate the feasible totalized observations. A Boolean-backed implementation may injectively encode the entire finite domain $\Omega_q \cup \left\{\mathrm{unobs}_q\right\}$ into designated projected atoms; an SMT enumerator with appropriate finite-domain projection may instead retain z_q as a theory term.

Theorem 5 (projection equivalence). For every caller-domain input $x \in \mathcal{X}_A$, the reachability equations have the unique solution $a_v \Leftrightarrow v \in D_{G(x,R)}$, and $Z = T^{\mathrm{tot}}_{G(x,R)}$. Hence projected models of $\Phi_{G,A,R}$ over Z are in bijection with feasible sparse observations. For every typed input x and feasible τ,

$$\left(\exists a : \Phi_{G,A,R}(x, a, \tau^{\mathrm{tot}})\right) \Leftrightarrow \Gamma_{\tau(x)}.$$

Proof. Induct backward over the acyclic consumer-to-operand graph. Requested roots establish the base reachability. An ordinary consumer enables every operand; a selection enables its selector and exactly the cases selected by the direct η predicates. The biconditionals therefore compute precisely the least enabled closure, including disjunctive sharing from multiple consumers. Substitution in the definition of z_q gives the totalized partial map. Therefore the existential formula holds exactly when $A(x)$ holds and the induced totalized observation equals τ^{tot}; the exact observed-outcome guard theorem gives the final equivalence. □

Consequently, naive projected enumeration of complete observation tuples and full-fiber blocking have the same $K + 1$ model-producing invocation count. Full-fiber generation is an input-only, lazily constructed substitution of one projected-assignment blocker. An enumerator that emits a short partial cube has a different output contract: one cube may cover several complete selection observations.

6.5. Representation boundary

Local generation avoids constructing unobserved cones for one record and returns a sparse observation, guard, and residual directly. The global encoding constructs one shared whole-graph value/reachability circuit and reuses it across all records; it can exploit mature projected-enumeration algorithms that avoid a growing sequence of ordinary blocking clauses. Neither dominates in the general model. Classifier and direct case-membership circuits can themselves be exponential in a succinct selector description, and local output can repeat large residual structures unless DAG sharing is explicit.

The equivalence also fixes attribution. Projected enumeration supplies the generic enumerator (Phan & Malacaria, 2015; Spallitta et al., 2025); symbolic execution supplies residual construction; the graph-specific obligation is proving that enabled reachability selects the recorded sites and that the totalized unobserved/outcome coordinates and the observed-outcome guard induce the same observer fibers.

7. Compositional extension

The core framework treats one requested graph independently of any implementation strategy. This section extends that semantics to sequential components. It states when demand-parametric guarded summaries preserve the same observations as graph flattening, including contextual site identity and sharing. The result is an exactness theorem, not a compactness claim.

7.1. Graph substitution and sharing

Consider sequential components $G_1 : X \to Y$ and $H : Y \to Z$ with disjoint site namespaces and a type-preserving bijection $\rho : I_H \to O_{G_1}$. Flattening removes the input nodes of H, retains one context-prefixed copy $c \cdot (V_H \setminus I_H)$, and redirects every edge that targeted $i \in I_H$ to $\rho(i)$. Call the resulting graph F. Two uses of one retained node in either component remain shared; a second component occurrence receives a different contextual prefix.

Embed every downstream node into the flattened graph by

$$\iota_{H(v)} = \begin{cases} \rho(v) \text{ if } v \in I_H, \\ c \cdot v \text{ if } v \notin I_H, \end{cases} \quad O_F = \iota_{H(O_H)}.$$

For input x and requested roots $R \subseteq O_H$, define the boundary valuation $y_i = \mathrm{val}_{x(\rho(i))}$. The downstream enabled closure determines the demanded boundary inputs and hence the upstream root demand:

$$\delta_H = I_H \cap D_{H(y,R)}, \quad R_1 = \rho(\delta_H).$$

Proposition 3 (flattening and contextual sharing). The flattened enabled closure decomposes as

$$D_{F\left(x,\iota_{H(R)}\right)} \cap c \cdot (V_H \setminus I_H) = c \cdot \left(D_{H(y,R)} \setminus I_H\right)$$

and

$$D_{F\left(x,\iota_{H(R)}\right)} \cap V_{G_1} = D_{G_1}(x, R_1).$$

Consequently,

$$T_{F\left(x,\iota_{H(R)}\right)} = T_{G_1}(x, R_1) \cup c \cdot T_{H(y,R)}.$$

The union is compatible and records each shared internal site once.

Proof. A topological induction first gives value substitution: every retained H node in F has the value it has under boundary valuation y, while every G_1 node keeps its value under x. Enabled reachability among retained H nodes therefore follows exactly $D_{H(y,R)} \setminus I_H$. Its boundary crossings are exactly δ_H and are redirected to R_1. Starting from those roots, the upstream restriction is exactly $D_{G_1}(x, R_1)$; no other retained H edge enters G_1. Restricting both sets to selection sites yields the observation equation. Root-demand union preserves sharing, and contextual prefixes make distinct occurrences disjoint. □

Operationally, picture a pipeline $G_1 \to H$. A downstream summary for H says which boundary inputs it needs. Those ports select the requested roots for an upstream summary of G_1; the upstream residuals are then substituted into the downstream guard and residual. The contract below states exactly when that familiar substitution covers each flattened-graph input once.

A *demand-parametric exact summary* is defined by the structural projection

$$\Pi_{G,R}(x) = \left(\delta_{G(x,R)}, T_{G(x,R)}\right), \quad \delta_{G(x,R)} = I_G \cap D_{G(x,R)}.$$

Proposition 4 (demanded-port locality). If $\Pi_{G,R}(x) = (\delta, \tau)$ and a typed input x' agrees with x on δ, then

$$\Pi_{G,R}(x') = (\delta, \tau) \quad \text{and} \quad \mathrm{val}_{x'}|_R = \mathrm{val}_x|_R.$$

Proof. Couple a topological value induction with the reachability approximants of the enabled closure. Every operand of a reached ordinary node is reached, and the selector operand of every reached selection is reached. Therefore every external input capable of changing a reached value, observed outcome, or subsequently enabled edge lies in δ. Agreement on those inputs preserves values on the current closure; equal selection outcomes then preserve the next reachability approximant. Induction gives the same closure, structural projection, and requested values. □

For every $(\delta, \tau) \in \mathrm{im}\big(\Pi_{G,R}\big)$, the summary has exactly one record (g, δ, τ, r) with

$$g : \prod_{i \in \delta} \mathcal{D}_i \to \mathbb{B}, \quad r : \prod_{i \in \delta} \mathcal{D}_i \to \prod_{o \in R} \mathcal{D}_o,$$

such that, for every full component input $x \in \mathcal{X}_G$,

$$g(x|_\delta) \Leftrightarrow \Pi_{G,R}(x) = (\delta, \tau), \quad g(x|_\delta) \Rightarrow r(x|_\delta) = \mathrm{val}_x|_R.$$

This is the *exact-summary contract*. It is a full-domain contract with no independent precondition: if an interface precondition mentions an otherwise undemanded input, that input must be charged as interface support. For fixed R, the lockstep argument behind the exact observed-outcome guard shows that τ determines δ; the demanded-port set remains explicit because it is needed by component interfaces.

For a downstream record $(h, \delta_H, \tau_H, r_H)$ of H at demand R, and an upstream record $(g, \delta_1, \tau_1, r_1)$ of G_1 at demand $\rho(\delta_H)$, the summary contract gives the following exact guard and residual types:

$$h : \prod_{i \in \delta_H} \mathcal{D}_i \to \mathbb{B}, \quad r_H : \prod_{i \in \delta_H} \mathcal{D}_i \to \prod_{o \in R} \mathcal{D}_o,$$

$$g : \prod_{j \in \delta_1} \mathcal{D}_j \to \mathbb{B}, \quad r_1 : \prod_{j \in \delta_1} \mathcal{D}_j \to \prod_{o \in \rho(\delta_H)} \mathcal{D}_o.$$

Thus every expression below consumes precisely the demanded boundary tuple. For a full caller input x, write $x|_{\delta_1}$ for its restriction to the demanded upstream ports. Reindex the upstream residual at the downstream inputs by

$$\hat{r}_1(x)_i = r_1\big(x|_{\delta_1}\big)_{\rho(i)}, \quad i \in \delta_H.$$

Their composed record is

$$\Big(g\big(x|_{\delta_1}\big) \wedge h(\hat{r}_1(x)), \delta_1, \tau_1 \cup c \cdot \tau_H, r_{H(\hat{r}_1(x))}\Big).$$

Infeasible guard conjunctions are discarded.

> **Theorem 6 (exact guarded-summary composition).** The feasible composed records form exactly the structural partition of F: every caller input satisfies one record; its demanded inputs, contextual observation, and residual agree with flattened evaluation; and no two records have the same composite structural projection.

Proof. The concrete boundary value selects one exact downstream record. Its δ_H selects the upstream demand and hence one exact upstream record. Demanded-port locality makes both selections and both residuals depend only on the displayed port restrictions, so the component contracts are well-typed rather than assumptions about ignored inputs. Guard and residual substitution, together with the flattening proposition, proves soundness and coverage. If two composed records had the same structural projection, disjoint contextual namespaces would make both component observations equal. The observed-outcome lockstep argument fixes their demanded boundary sets, so the exact-summary contract makes both component records, and hence the composed records, identical. □

This is equality with flattening, not a compactness theorem. A summary may contain one family for each of $2^{|O_H|}$ root-demand sets, and compatible record products may still be exponential. Guard substitution and exact piecewise residual composition are established techniques (Geyer et al., 2010); the graph-specific content is requested-root demand, demanded-input propagation, contextual identity, and preserved DAG sharing.

8. Complexity and specialized frontiers

The comparison framework must separate record count, representation size, oracle invocations, and actual enumeration complexity. This section first charges the graph, symbolic circuits, solver formulas, and serialized output, then identifies hardness in the general problem and stronger established frontiers for geometric and parametric specializations.

8.1. Parameters and enumeration classes

Table 4 fixes the charge model used below. Formula sizes are shared-DAG sizes unless a serialized representation is named explicitly.

Symbol	Meaning	What is charged
S_G	Weighted source-DAG size	Nodes, edges, case lists, constants, widths, and aggregate leaves
S_{enc}	Actual global value-and-reachability encoding size	Source wiring; reachability equations; primitives; classifiers; outcomes; case membership; selected combiners; multiplexing; typed domains
A_{enc}, $S_{A_{\mathrm{enc}}}$	$A_{\mathrm{enc}} = \mathrm{Dom}_G \wedge A$ and its shared size	Typed-domain encoding plus the caller constraint
n_Q, b_q, K, L	Site count, outcomes at q, feasible observations, and maximum observed sites per record	Structural and output cardinalities
S_τ	Demanded subencoding size for record τ	Candidate-local guard and residual construction
S_{out}, W_{wit}	Declared total output size and serialized witness size	Flat serialization or shared output DAG, plus witnesses
$T_{\mathrm{SMT}}(\varphi)$	Cost of deciding φ and producing a model	Solver work rather than oracle-call count alone

Table 4: Complexity parameters and their charge boundaries.

In particular, S_{enc} may be larger than S_G, and S_{out} is meaningful only together with its declared flat or shared representation.

The source-node count alone is insufficient. A width-w mask-valued selector is one node but can have 2^w mask outcomes. Lowering one multiway site to binary choices can also introduce auxiliary decisions whose projection must be merged back to one source outcome; such an elaboration must be charged separately.

OutputP requires total time polynomial in serialized input plus total output. IncP bounds the time to produce the first k solutions by a polynomial in the input size and k under the stated encoding and balance convention. DelayP bounds the time before the first output, between consecutive outputs, and after the last output by an input polynomial. A compiled decision diagram or d-DNNF is not free preprocessing unless it is declared to be the input. No such classification is proved for the general algorithm here: exact-record membership, SMT solving, coefficient growth, residual equivalence, and flat guard serialization need not be polynomial (Creignou et al., 2019).

8.2. Output count and unavoidable products

Totalization gives the general structural bound

$$K \le \prod_{q \in Q} (1 + b_q).$$

The coordinates are not independent, so the bound may be loose. If every site is always observed, the 1 terms disappear. For n independent observed binary selectors, $K = 2^n$, and any explicit record enumerator needs $\Omega(K)$ output time. Demand-relative omission cannot remove this necessary product. Conversely, a single mask-valued site can have $K = 2^w$ while $L = 1$; trace length alone is not a complexity parameter.

The output quotient also matters. Dense activation cells can differ while implementing the same affine function, and piecewise-affine minimization can merge equal-behavior guards into coarser representations (Geyer et al., 2008). The selection observer retains observed equal-valued outcomes, so those merged output counts cannot be substituted for K without changing the task.

8.3. Cost of full-fiber blocking

Ignoring solver work, memoized candidate-local construction costs

$$O\left(\sum_{\tau\in\mathcal{T}_{G,A,R}} S_\tau\right)\subseteq O(KS_{\mathrm{enc}})$$

shared-DAG operations and creates guard/residual DAGs of the same order. If every flat record copies A_{enc}, output size includes $O\big(KS_{A_{\mathrm{enc}}}\big)$; if records share one immutable reference to A_{enc}, it includes only $O\big(S_{A_{\mathrm{enc}}}\big)$ once. Complete record serialization additionally includes W_{wit}; no representation-independent bound on witness byte length is assumed.

Before model query j, the solver sees

$$A_{\mathrm{enc}}\wedge\bigwedge_{i<j}\neg g_i.$$

An honest solver-time expression is therefore

$$\sum_{j=0}^{K} T_{\mathrm{SMT}}\left(A_{\mathrm{enc}}\wedge\bigwedge_{i<j}\neg g_i\right).$$

The cumulative blocker DAG can grow to $O(KS_{\mathrm{enc}})$, plus one shared copy of A_{enc}. Incremental solving may reuse learned clauses but supplies no general monotone-runtime or polynomial-delay guarantee. The global projected encoding has one shared $O\big(S_{\mathrm{enc}}+S_{A_{\mathrm{enc}}}\big)$ circuit but may spend exponential work compiling or enumerating its projected image. The two presentations have the same $K+1$ model-producing invocation count under naive complete-tuple blocking, not the same runtime or memory.

Formula representation must also preserve graph sharing. In the chain

```
e0 = x
e1 = f(e0, e0)
e2 = f(e1, e1)
...
en = f(e[n-1], e[n-1])
```

the source DAG is linear while recursive tree serialization is exponential. Every size claim therefore refers to a formula DAG, let-bound SMT term, definitional encoding, or measured serialized bytes.

8.4. Decision and counting hardness

The hardness statements in this subsection are inherited boundaries obtained by standard circuit and SAT encodings, not contributions of this survey. Feasibility of one requested outcome is NP-hard even for Boolean graphs: let a selector compute an arbitrary circuit $F(x)$ and ask whether its `true` outcome is feasible. For polynomial-size Boolean circuits over explicitly encoded finite bit-vector inputs, a supplied input verifies feasibility in polynomial time, so that restricted problem is in NP. The general typed model allows primitives without a declared polynomial-time evaluation bound and therefore makes no uniform membership claim.

Counting feasible observations is #P-hard when A is part of the input. Under a Boolean constraint $F(x)$, expose every input bit as an always-observed binary outcome; observations are then injective on satisfying assignments and $K=\#\mathrm{SAT}(F)$. We do not call general image counting #P-complete because many inputs may map to one observation and an upper-bound proof is separate. Checking that a supplied finite guard family covers every Boolean input is coNP-hard by propositional validity.

8.5. Hyperplane cells: a strict all-sites-observed case

Suppose every site is structurally observed. After canonicalization, let $Q_c\subseteq Q$ be a geometric core of $m=|Q_c|$ sites. For each $q\in Q_c$, the classifier is the strict sign of a nonconstant affine form $\ell_q:\mathbb{R}^D\to\mathbb{R}$, and distinct core forms define distinct hyperplanes. Constant and scalar-duplicate site coordinates are outside this core; a fixed reconstruction map restores their deterministic outcomes in each full observation record. Let $A(x)\Leftrightarrow\prod_{q\in Q_c}\ell_{q(x)}\neq 0$, and define each core classifier arbitrarily on zero outside that caller domain. Each feasible core observation is one full-dimensional open cell of the resulting m-hyperplane arrangement, and K is equivalently

the number of such cells and reconstructed observations. The cost symbols below are source-specific: $L_{\mathrm{AF}}(m,d)$ is the arithmetic cost of the LP used by Avis–Fukuda, while L_F and L_R denote the LP costs charged respectively by Ferrez et al. and Rada–Černý. Avis and Fukuda enumerate all K cells in

$$O(KmDL_{\mathrm{AF}}(m,D))$$

arithmetic time with $O(mD)$ working space (Avis & Fukuda, 1996). Sleumer gives $O(Km)$ arithmetic time for fixed D (Sleumer, 1998). For central arrangements, Ferrez, Fukuda, and Liebling give the pre-Rada bound

$$O\Big(KmL_{F(m,D)}\Big)$$

with $O(mD)$ working space (Ferrez et al., 2005). Rada and Černý later give a different incremental sign-prefix formulation with $O\Big(KmL_{R(m,D)}\Big)$ time (Rada & Černý, 2018).

All four algorithms are complete, duplicate-free, and output-polynomial under their stated arithmetic or LP-cost models. Rada–Černý additionally classify their bounded-encoding algorithm in OutputP; the other displayed operation bounds are not silently promoted to coefficient-bit theorems. For the full-dimensional rational zonotope dual to this central-arrangement case, Deza and Pournin give a self-contained rational-bit analysis of a traversal. If B_Z is total generator encoding length, its proof gives

$$O(Km[q(m,D,B_Z)+\log K]) = O(Kp(m,D,B_Z)),$$

for unspecified polynomials q and p, while retaining all visited vertices and generator subsets (Deza & Pournin, 2022). The symmetry-aware White Whale successor can greatly reduce structured instances but gives no stronger generic theorem (Deza et al., 2026). Newer circuit-guided central-arrangement algorithms report large practical improvements without a replacement OutputP, IncP, or DelayP bound (Dussault et al., 2025).

These traversal bounds count core cells. Materializing dense records that restore every noncore coordinate adds $O(K|Q|)$ output work. They do not automatically bound the survey's complete four-field record: residualization must be restricted to an explicitly charged affine/PWA computation or added to the construction and serialization cost, and witness production must also be charged.

The reduction has sharp boundaries. A non-strict classifier assigns boundary points to one side, and an arbitrary A can cut or lower the dimension of a fiber. Such a fiber need not be an open full-dimensional arrangement cell. None of the geometric bounds transfers automatically to general typed bitvector primitives or an arbitrary solver theory.

8.6. Parametric partitions

Jones and Maciejowski enumerate every full-dimensional critical-region basis of a multiparametric LP by reverse search. For their constraint matrix $M \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(M) = m$, parameter dimension d, $e = n - m$, and N_r regions, let $L_{P(a,b)}$ denote the paper's exact LP-oracle cost at the displayed dimensions. Their LP-relative bound is

$$O\Big(N_r\Big[e^2L_{P(d,e)} + eL_{P(m,n)}\Big]\Big),$$

with output-relative constant auxiliary space; each basis reconstructs a polyhedral guard and affine optimizer (Jones & Maciejowski, 2006). The later sufficient-matrix pLCP treatment covers pLP and convex pQP and gives a stronger explicit comparator. For the separate pLCP dimension n, parameter dimension d, and N_g reported bases in general position, let $L_{C(a,b)}$ denote that paper's LP-oracle cost. Columbano, Fukuda, and Jones bound the work by

$$O\left(N_g\left[nL_{C(n,d+1)} + \frac{n^2-n}{2}L_{C(2n,d+1)}\right]\right).$$

Under lexicographic perturbation, for N_p perturbed bases the corresponding bound is

$$O\left(N_p\left[(n^2+n)L_{C(n,d+1)} + \frac{n^3-n}{2}L_{C(2n,d+1)}\right]\right).$$

The perturbed basis count can exceed the number of unperturbed critical domains; the analysis retains an output-sized visited set and supplies neither a coefficient-bit nor a DelayP theorem (Columbano et al., 2009).

Separate work resolves other degeneracy and representation questions. Lexicographic perturbation selects a unique continuous affine optimizer and nonoverlapping basis regions (Jones et al., 2007). A minimum-norm secondary optimization yields, under stated assumptions, a unique continuous selection from the original pQP solution set with an algorithm-independent polyhedral representation (Spjøtvold et al., 2007). Patrinos and Sarimveis discover every full-dimensional convex-pQP neighbor across a facet without nondegeneracy or a facet-to-facet assumption, but state no polynomial total, delay, space, or bit bound (Patrinos & Sarimveis, 2010). Bemporad's rank-deficiency treatment likewise gives practical per-combination analyses rather than a new enumeration-class theorem (Bemporad, 2015). Finally, pLP solution and halfspace polyhedral projection are polynomially interreducible, so renaming one side as projection does not establish a different complexity frontier (Jones et al., 2008).

The parametric objects also need a local correspondence caveat. The full-dimensional critical regions reported by Jones and Maciejowski are closed and can overlap on boundaries; regions that exist only in lower dimension are not emitted. Without a boundary-ownership convention or a caller-domain restriction that removes those boundaries, this is not the disjoint all-input fiber contract of Section 3. Parametric results are therefore adjacent optimizer/basis comparators unless a target-observer correspondence is proved.

8.7. General equivalence-class enumeration

Wang et al. give a different strong frontier: from an explicitly supplied acyclic decomposable AND/OR solution graph with locally colored OR nodes, their `Next` algorithm enumerates every contracted colored solution-tree class exactly once with delay $O(ns)$, where n is graph size and s is the solution size (Wang et al., 2021). This establishes exact quotient enumeration with polynomial delay as prior art. It does not classify the present problem: no general reduction from the selection observer to that decomposable representation is known, and its outputs are class trees rather than exact caller-input guards, residuals, and witnesses.

The resulting complexity claim is intentionally negative but precise: the general theory proves finite exact enumeration and an exact $K + 1$ oracle-call accounting, while established special cases already have stronger output-sensitive algorithms. It proves no general asymptotic advantage from demand-relative sparsity alone.

9. Comparative synthesis

The six routes overlap in mechanism and representation, so listing them does not yet show whether they solve the same problem. This section compares their observers and record guarantees, identifies the closest established results that bound the survey's claims, and answers the four research questions.

9.1. Observer, contract, and representation

The observer-kernel test gives the shortest comparison. Projected enumeration and candidate-local symbolic evaluation are direct general presentations after the instrumentation proved in Section 6. Decision structures are direct when they compile the totalized observer; dense-sign geometry is direct only on its all-sites-observed affine restriction; and demand-guided, parametric, or compositional methods require the correspondence conditions stated in Section 5 and Section 7.

This is more precise than asking whether a method emits a "partial assignment." Structural absence is an observer value; existential projection hides formula coordinates; a logical don't-care represents either value; and equal-residual coalescing takes a quotient. Similar syntax can therefore denote different fibers.

The record contract adds four independent obligations: duplicate-free coverage, guard/fiber equivalence, residual correctness, and a witness. Output representation remains orthogonal: a flat record stream, tree, decision DAG, or polyhedral complex may encode the same observer while charging very different construction and serialization costs. The two tables below separate published native guarantees from the framework correspondence needed to obtain the target contract.

9.2. Complexity takeaway

The general problem can have exponentially many observations and contains hard feasibility and counting special cases. Its $K + 1$ model-query accounting is not an output-sensitive theorem; stronger guarantees belong

to restricted or precompiled inputs and must charge residuals, witnesses, and representation size as Section 8 specifies.

9.3. Closest established results

The nearest results are output-directed and dataflow-specific, while the strongest novelty boundary comes from exact equivalence-class enumeration on a different representation. Table 5 consolidates those published native objects and guarantees.

Work and native object	Coverage guarantee	Guard · residual · witness	Relation to target
PESO: relevant-slice conditions at requested outputs (Qi et al., 2013)	Conditionally complete exploration of finite RSCs	RSC/path condition · symbolic output · solver test	Kernel equality open; may refine or fragment a target fiber
SPD: queried-use path-family graph (Santelices & Harrold, 2010)	Exact mode described as pathwise-equivalent; no numbered end-to-end theorem	Path-family condition · shared guarded values · no exposed witness record	Kernel equality open; dependence families may be finer
Feng et al.: mux functional-space cells (Feng et al., 2004)	Mutually exclusive functional-space partition intended	Boolean control · data expression · no witness	Restricted-close; merge choices can make its kernel coarser
Kanade et al.: bounded discrete trace (Kanade et al., 2009)	Sampled class is underapproximated	Sufficient predicate · transformer · sample	Different trace observer; sampled region is not a full kernel class
Sylvia: RTL path-fragment combination (Ryan & Sturton, 2023)	Feasible combinations are SMT-filtered; stated worst case is exponential	Fragment constraints · no fiber-wide residual · replayable model	Adjacent modular construction; no selection-fiber theorem
Fingerprints / MPT: executed-choice map (Alqaddoumi et al., 2010; Antoy & Hanus, 2009; Hanus & Teegen, 2021)	Search guarantees are source-strategy-specific	No input guard · result value, not residual · task	Different input domain and observer; no kernel order established
SPLat: reachable configuration test trace (Kim et al., 2013)	One execution per distinct trace is claimed; no formal exactness theorem	Partial configuration cylinder · none · execution	Trace kernel may be finer or incomparable to selection events
Wang et al.: colored solution-tree class (Wang et al., 2021)	Every class exactly once; delay $O(ns)$	No input guard · class tree · no witness	Different solution domain; incomparable without a reduction

Table 5: Closest and novelty-bounding published comparisons. "Open" means that this survey does not prove equality with the target observer or one-record contract.

The general routes require a separate table because several rows are framework reductions or restricted correspondences rather than native end-to-end algorithms for the target.

Route and native object	Coverage guarantee	Guard · residual · witness	Framework correspondence
Projected AllSMT: instrumented observation tuple (Phan & Malacaria, 2015)	Complete tuples enumerate the exact image	Existential fiber · no residual · model	Same kernel after faithful activity/outcome instrumentation
BDD/ADD compilation: finite observer function (Bahar et al., 1997; Bryant, 1986)	Exact for the compiled function; BDD canonicity needs fixed order	Terminal preimage · optional terminal residual · optional model	Same kernel iff labels injectively relabel the totalized observer
Hyperplane traversal: strict sign cell (Avis & Fukuda, 1996; Rada & Černý, 2018)	Complete, duplicate-free, output-sensitive under source cost models	Polyhedral cell · separate/charged · method-specific	Same kernel on the boundary-free all-sites-observed affine restriction
Parametric traversal: optimizer basis or region (Columbano et al., 2009; Jones & Maciejowski, 2006)	Full-dimensional basis/region coverage under stated assumptions	Closed polyhedron · affine optimizer · method-specific	Usually different; boundary overlap prevents a disjoint all-input kernel
Guarded component summaries: namespaced observations under demanded ports (Geyer et al., 2010)	Exact under full-domain locality and contextual identity assumptions	Conjoined guard · substituted residual · witness from composed guard	Same kernel as flattening under the composition theorem

Table 6: Route-level guarantees after the instrumentation or restrictions required by the unified framework.

The tables make two novelty boundaries explicit. Output-directed guarded exploration and sparse executed-choice maps are established, and Wang et al. already prove exact polynomial-delay quotient enumeration for a restricted decomposable representation. Their class-tree output does not, however, supply arbitrary caller-input guards, residuals, or witnesses.

Three exact correspondences remain open: whether instrumented PESO plus RSC quotienting yields exactly one selection fiber; whether SPD's dependence families denote the enabled-closure observer rather than a refinement; and whether Feng's actual split/merge algorithm, augmented with immutable contextual event labels, preserves every target observation. The survey does not use their absence of an explicit theorem as evidence that no reduction exists.

The strongest adjacent boundaries remain useful but should not be mistaken for the closest algorithms. Requested-output quotients can merge distinct internal histories, as in common-first-action controller regions (König & Mönnigmann, 2020; Mitze et al., 2021) and exact neural-policy decision trees (Chang et al., 2026). Giua et al. give exact inverse consistency for Petri-net markings under an observed label word (Giua et al., 2003). SymPaths records scheduler choices and proves its symbolic semantics sound and complete against concrete executions (Boer et al., 2020). Lindblad's property-blocked constructor search is a close demand-refinement shape, but its stated soundness and completeness conditions are explicitly unproved (Lindblad, 2008); Korat and the BLISS–LISSA–PLI heap line establish stronger bounded access-guided and feasibility-preservation results for different artifacts (Boyapati et al., 2002; Copia et al., 2022; Copia et al., 2023; Rosner et al., 2015). Finally, Fisler and Vardi show that exact BDD bisimulation minimization can cost more than direct symbolic checking (Fisler & Vardi, 2002). No performance claim follows from the observer-fiber construction alone.

9.4. Adjacent reduction problems

Several broad literatures informed the observer vocabulary without providing alternative implementations of the target contract. Partial-order reduction omits redundant interleavings while preserving reachable states or temporal properties (Alur et al., 2001; Holzmann et al., 1992). Petri-net unfoldings represent concurrent configurations, and goal-directed prefixes can preserve all minimal configurations reaching a requested marking while omitting irrelevant transitions (Bonet et al., 2014; Chatain & Paulevé, 2017). Observer- and property-guided state reduction preserves a declared event language, state quotient, or coverage objective (Aronis et al., 2018; Bugrara & Engler, 2013). These works confirm the general lesson that omission is observer-relative, but their outputs are runs, markings, states, or search objectives rather than caller-input fibers with residual functions.

Whole-network dataflow semantics similarly supplies important boundaries for determinism, stability, and compositionality, but usually observes streams, traces, or network behavior rather than finite requested-root selection maps. The main comparison therefore uses fixed-input least-demand results where they directly explain enabled closure and leaves the broader semantic lineage in the repository synthesis.

9.5. Answers to the research questions

RQ1. The common object is a finite observer on caller inputs. Its nonempty inverse images are fibers, and an exact record attaches a guard, residual, and witness to each fiber. This distinguishes semantic partitioning from the algorithm and data structure used to expose it.

RQ2. The included approaches organize into six recurring, overlapping routes. Local guarded evaluation and global projected enumeration are equivalent general presentations; decision structures compile the observer; demand-guided methods supply sparse discovery mechanisms; dense-sign geometric methods solve a restricted affine specialization; and guarded summaries provide composition.

RQ3. General exactness follows only after all four record obligations are proved. General output-sensitive complexity does not follow from sparse demand or one model per fiber. Stronger bounds belong to restricted geometric, parametric, or compiled inputs and must charge their representations. The route-by-route obligations and assumptions are summarized in Table 5 and Table 6.

RQ4. Approaches induce the same fibers when their observer kernels agree after explicit instrumentation. Their labeled images and record schemas then correspond only after an explicit image bijection. Coordinate projection, equal-behavior merging, path refinement, and property-guided search otherwise produce coarser, finer, or incomparable partitions. This observer test is the framework's primary rule for transferring results across terminology; the two guarantee tables record the known equality, open, and incomparable cases.

10. Boundaries and open problems

10.1. Applicability of selection observations

Selection observations may support event-aware specifications such as checking that graph rewriting, lowering, or component substitution preserves a declared map of source selection events. This is a motivating hypothetical in the present paper, not a demonstrated application: no worked validation case or evaluation is reported. They are not preferable for every client. A value-only client should merge behaviorally equal records, as explicit-control methods do when they group regions that share the requested first action (König & Mönnigmann, 2020) or compile an exact extensional policy tree (Chang et al., 2026); an optimization client may retain active constraints rather than source events; and a diagnostic client may record causal events beyond selections. The framework makes these choices explicit rather than ranking them by a universal notion of precision.

10.2. Semantic scope

The theory assumes a finite, typed, acyclic, deterministic, pure graph. Ordinary primitives, classifiers, and selected combiners are total. Partial operations require a proved caller precondition or an explicit error outcome; silently ignoring division by zero, invalid indexing, assertion failure, or language-specific overflow would invalidate coverage. Cycles, recursion, and unbounded dynamic occurrences can make the event domain infinite and require a bounded, time-indexed, regular, or coinductive observer instead.

The all-operands rule is an observation policy, not an extensional dependence theorem. Every operand of an ordinary node is observed even if algebraic simplification makes it irrelevant. Translating a lazy language or stream-dataflow semantics therefore requires a separate correspondence proof. Likewise, an outcome must determine its demanded cases and combiner; two raw selector values with different structural consequences cannot be silently grouped as one outcome.

Site identity is structural and occurrence-sensitive. Sharing within one graph or component occurrence records a site once, whereas distinct contextual occurrences receive distinct names. A compiler that duplicates, fuses, or lowers selections can change the observer while preserving values. Source and lowered observations need an explicit event map. The exact observed-outcome guard is not necessarily a literal-minimal formula: logical minimization is a later representation transformation and may erase the event whose outcome the record is intended to expose.

10.3. Solver, output, and composition boundaries

Exhaustion depends on exact symbolic encodings and decisive oracle answers. An `unknown`, timeout, unsupported primitive, incomplete theory combination, or inexact floating-point abstraction yields an explicitly incomplete result. A model is a witness for one fiber, not a residual valid throughout it.

The exact $K + 1$ model-producing invocation count is deliberately weak. It does not charge solver work, formula growth, coefficient bits, projection, serialization, or final unsatisfiability. The number of fibers can be exponential, and feasibility and counting contain familiar hard special cases. No general OutputP, IncP, DelayP, compact-summary, or practical-speedup theorem is established. The negative precedent is explicit: BDD bisimulation minimization can cost more than the symbolic invariant check it was meant to accelerate (Fisler & Vardi, 2002).

Representation can dominate the comparison. Flat guards, trees, diagrams, compiled circuits, polyhedral complexes, and residual DAGs may denote the same observer with exponentially different sizes. Exact guarded-summary composition has the same caveat: equality with flattening does not imply reuse. A component with many outputs can require exponentially many demand masks, and different caller predicates can split its fibers differently at each occurrence.

The exact-guard, projection-equivalence, demanded-port locality, and guarded-summary composition results are survey-authored derivations. They have not been peer reviewed, mechanized, or independently verified. In particular, the paper has not proved exact correspondence between its observer/record contract and PESO's relevant-slice conditions, SPD's path-family graph, or Feng et al.'s actual functional-space split/merge algorithm. Those are explicit open reduction questions, not implicit novelty evidence.

The evidence map is also open. The dated search snapshot contains an unresolved candidate backlog, so it supports the comparisons made from adjudicated and deep-read works but not a claim that every captured plausible competitor has been assessed.

10.4. Open theoretical questions

The framework suggests a focused research agenda:

- Which observer languages make selection events, value behavior, active constraints, and diagnostic causality explicit, and which refinement or full-abstraction results relate them?
- Which classes of partial operations, cycles, or recurring component occurrences retain a finite or finitely representable observation image?
- When can a shared decision structure, disjoint cover, or guarded-summary DAG be constructed with output-sensitive delay or polynomial auxiliary space?
- Which structural restrictions make demanded local generation asymptotically preferable to whole-graph reachability projection, rather than merely smaller on one candidate?
- Under what interface and workload conditions do demand-parametric summaries provide reusable compression instead of an exponential family of exact fibers?

These questions keep the observer, enumerator, and representation separate. Improving one does not automatically strengthen the other two.

10.5. Reference implementation and evaluation agenda

This paper reports no implementation or benchmark result. A reference implementation should first validate small finite graphs by exhaustive concrete input enumeration, checking observation identity, guard membership, residual values, witnesses, and coverage. It should then compare local full-fiber blocking, global projected enumeration, finite-input ADD or MTBDD compilation, and specialized geometric traversal on inputs that expose nested unobserved sites, equal-valued observed alternatives, sharing, multiple roots, multiway choices, lower-dimensional fibers, and repeated components. Measurements should report serialized and shared output size, compilation cost, solver effort, peak memory, and final-exhaustion cost—not only record count.

11. Conclusion

Exhaustive enumeration of selection observations is a specific observer-partition problem. For a caller domain and requested roots in a pure shared dataflow graph, each input induces a sparse map of contextual selection outcomes. The exact output is one guard, residual, and witness for every nonempty inverse-image fiber of that map.

The unified framework separates this semantic object from the method used to enumerate it and the structure used to represent it. Guarded symbolic execution, projected model enumeration, decision structures, demand-guided search, geometric traversal, and compositional summaries consequently become comparable without being declared equivalent by terminology alone. Local observed-outcome-guard generation and global reachability-and-outcome projection are two general presentations of the same observer. Decision structures compile it, affine methods solve important restricted specializations with stronger guarantees, and component summaries preserve it only under explicit demand, identity, and interface conditions.

The survey finds no new generic enumeration paradigm or general complexity advantage. Its contribution is a precise problem statement, common vocabulary, transfer criteria, and an evidence-backed analysis of known solution routes. Future progress should therefore state which observer is preserved, which record obligations are met, which representation is charged, and which assumptions support any claimed improvement.

References

Alqaddoumi, A., Antoy, S., Fischer, S., & Reck, F. (2010). The Pull-Tab Transformation. *Proceedings of the Fourth International Workshop on Graph Computation Models*, 1–6. https://web.cecs.pdx.edu/~antoy/homepage/publications/gcm10/paper.pdf

Alur, R., Brayton, R., Henzinger, T., Qadeer, S., & Rajamani, S. (2001). Partial-Order Reduction in Symbolic State-Space Exploration. *Formal Methods in System Design*, *18*(2), 97–116. https://doi.org/10.1023/a:1008767206905

Anand, S., Godefroid, P., & Tillmann, N. (2008). Demand-Driven Compositional Symbolic Execution. *Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science*, *4963*, 367–381. https://doi.org/10.1007/978-3-540-78800-3_28

Antoy, S. (2011). On the Correctness of Pull-Tabbing. *Theory and Practice of Logic Programming*, *11*(4–5), 713–730. https://doi.org/10.1017/S1471068411000263

Antoy, S., & Hanus, M. (2009). Set functions for functional logic programming. *Proceedings of the 11th ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, PPDP '09*, 73–82. https://doi.org/10.1145/1599410.1599420

Aronis, S., Jonsson, B., Lång, M., & Sagonas, K. (2018). Optimal Dynamic Partial Order Reduction with Observers. In *Tools and Algorithms for the Construction and Analysis of Systems* (pp. 229–248). Springer International Publishing. https://doi.org/10.1007/978-3-319-89963-3_14

Avis, D., & Fukuda, K. (1996). Reverse search for enumeration. *Discrete Applied Mathematics*, *65*(1–3), 21–46. https://doi.org/10.1016/0166-218x(95)00026-n

Avron, A., & Sasson, N. (1994). Stability, Sequentiality and Demand Driven Evaluation in Dataflow. *Formal Aspects of Computing*, *6*(6), 620–642. https://doi.org/10.1007/bf03259389

Bahar, R., Frohm, E., Gaona, C., Hachtel, G., Macii, E., Pardo, A., & Somenzi, F. (1997). Algebraic Decision Diagrams and Their Applications. *Formal Methods in System Design*, *10*(2–3), 171–206. https://doi.org/10.1023/a:1008699807402

Bemporad, A. (2015). A Multiparametric Quadratic Programming Algorithm With Polyhedral Computations Based on Nonnegative Least Squares. *IEEE Transactions on Automatic Control*, *60*(11), 2892–2903. https://doi.org/10.1109/tac.2015.2417851

Berzins, A. (2023). Polyhedral Complex Extraction from ReLU Networks Using Edge Subdivision. *Proceedings of the 40th International Conference on Machine Learning, Proceedings of Machine Learning Research*, *202*, 2234–2244. https://proceedings.mlr.press/v202/berzins23a.html

Boer, F. S. de, Bonsangue, M., Johnsen, E. B., Pun, V. K. I., Tapia Tarifa, S. L., & Tveito, L. (2020). SymPaths: Symbolic Execution Meets Partial Order Reduction. In *Deductive Software Verification: Future Perspectives* (pp. 313–338). Springer International Publishing. https://doi.org/10.1007/978-3-030-64354-6_13

Bonet, B., Haslum, P., Khomenko, V., Thiébaux, S., & Vogler, W. (2014). Recent advances in unfolding technique. *Theoretical Computer Science*, *551*, 84–101. https://doi.org/10.1016/j.tcs.2014.07.003

Boyapati, C., Khurshid, S., & Marinov, D. (2002). Korat: automated testing based on Java predicates. *Proceedings of the 2002 ACM SIGSOFT International Symposium on Software Testing and Analysis, Issta02*, 123–133. https://doi.org/10.1145/566172.566191

Braßel, B. (2011). *Implementing Functional Logic Programs by Translation into Purely Functional Programs* [Doctoral dissertation]. https://d-nb.info/1020245336/34

Braßel, B., & Huch, F. (2007). On a Tighter Integration of Functional and Logic Programming. In Z. Shao (Ed.), *Programming Languages and Systems: Vol. 4807. Programming Languages and Systems*. https://doi.org/10.1007/978-3-540-76637-7_9

Bryant, R. E. (1986). Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, *C-35*(8), 677–691. https://doi.org/10.1109/TC.1986.1676819

Bugrara, S., & Engler, D. (2013). Redundant State Detection for Dynamic Symbolic Execution. *USENIX Annual Technical Conference*, 199–211. https://www.usenix.org/conference/atc13/technical-sessions/presentation/bugrara

Chang, K., Dahlin, N., Jain, R., & Nuzzo, P. (2026). Equivalent and Compact Representations of Neural Network Controllers With Decision Trees. *IEEE Transactions on Automatic Control*, *71*(8), 5276–5289. https://doi.org/10.1109/tac.2026.3676368

Chatain, T., & Paulevé, L. (2017). Goal-Driven Unfolding of Petri Nets. *28th International Conference on Concurrency Theory (CONCUR 2017), Leibniz International Proceedings in Informatics*, *85*, 18:1–18:16. https://doi.org/10.4230/LIPIcs.CONCUR.2017.18

Columbano, S., Fukuda, K., & Jones, C. (2009). An output-sensitive algorithm for multi-parametric LCPs with sufficient matrices. In *Polyhedral Computation* (pp. 73–102). American Mathematical Society. https://doi.org/10.1090/crmp/048/04

Copia, J. M., Molina, F., Aguirre, N., Frias, M. F., Gorla, A., & Ponzio, P. (2023). Precise Lazy Initialization for Programs with Complex Heap Inputs. *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*, 752–762. https://doi.org/10.1109/issre59848.2023.00080

Copia, J. M., Ponzio, P., Aguirre, N., Gorla, A., & Frias, M. (2022). LISSA: Lazy Initialization with Specialized Solver Aid. *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering, ASE '22*, 1–12. https://doi.org/10.1145/3551349.3556965

Creignou, N., Kröll, M., Pichler, R., Skritek, S., & Vollmer, H. (2019). A complexity theory for hard enumeration problems. *Discrete Applied Mathematics*, *268*, 191–209. https://doi.org/10.1016/j.dam.2019.02.025

Denaro, G. (2012). All-values symbolic execution. *2012 7th International Workshop on Automation of Software Test (AST)*, 138–144. https://doi.org/10.1109/iwast.2012.6228982

Deza, A., & Pournin, L. (2022). A linear optimization oracle for zonotope computation. *Computational Geometry*, *100*, 101809. https://doi.org/10.1016/j.comgeo.2021.101809

Deza, A., Hao, M., & Pournin, L. (2026). Sizing the White Whale. In *Data Science and Optimization* (pp. 209–230). Springer Nature Switzerland. https://doi.org/10.1007/978-3-032-03844-9_9

Drammis, S., Zheng, B., Srinivasan, K., Berwick, R. C., Lynch, N. A., & Ajemian, R. (2024). Parallel Algorithms for Exact Enumeration of Deep Neural Network Activation Regions. *Arxiv Preprint Arxiv:2403.00860*. https://arxiv.org/abs/2403.00860

Dussault, J.-P., Gilbert, J. C., & Plaquevent-Jourdain, B. (2025). On the B-differential of the componentwise minimum of two affine vector functions. *Mathematical Programming Computation*, *17*(1), 1–52. https://doi.org/10.1007/s12532-024-00266-8

Feng, T., Wang, L.-C., Cheng, K.-T., & Lin, A. C.-C. (2004). Improved Symbolic Simulation by Dynamic Functional Space Partitioning. *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, 42–47. https://doi.org/10.1109/DATE.2004.1268825

Ferrez, J.-A., Fukuda, K., & Liebling, T. (2005). Solving the fixed rank convex quadratic maximization in binary variables by a parallel zonotope construction algorithm. *European Journal of Operational Research*, *166*(1), 35–50. https://doi.org/10.1016/j.ejor.2003.04.011

Fisler, K., & Vardi, M. Y. (2002). Bisimulation Minimization and Symbolic Model Checking. *Formal Methods in System Design*, *21*(1), 39–78. https://doi.org/10.1023/a:1016091902809

Geyer, T., Torrisi, F. D., & Morari, M. (2008). Optimal complexity reduction of polyhedral piecewise affine systems. *Automatica*, *44*(7), 1728–1740. https://doi.org/10.1016/j.automatica.2007.11.027

Geyer, T., Torrisi, F. D., & Morari, M. (2010). Efficient Mode Enumeration of Compositional Hybrid Systems. *International Journal of Control*, *83*(2), 313–329. https://doi.org/10.1080/00207170903159285

Giua, A., Júlvez, J., & Seatzu, C. (2003). Marking Estimation of Petri Nets Based on Partial Observation. *Proceedings of the 2003 American Control Conference*, *1*, 326–331. https://doi.org/10.1109/ACC.2003.1238961

Godefroid, P. (2007). Compositional dynamic test generation. *ACM SIGPLAN Notices*, *42*(1), 47–54. https://doi.org/10.1145/1190215.1190226

Godefroid, P., Klarlund, N., & Sen, K. (2005). DART: directed automated random testing. *ACM SIGPLAN Notices*, *40*(6), 213–223. https://doi.org/10.1145/1064978.1065036

Hanus, M., & Teegen, F. (2021). Memoized Pull-Tabbing for Functional Logic Programming. In *Functional and Constraint Logic Programming* (pp. 57–73). Springer International Publishing. https://doi.org/10.1007/978-3-030-75333-7_4

Holzmann, G. J., Godefroid, P., & Pirottin, D. (1992). Coverage Preserving Reduction Strategies for Reachability Analysis. In *Protocol Specification, Testing and Verification, XII* (pp. 349–363). Elsevier. https://doi.org/10.1016/b978-0-444-89874-6.50028-3

Huang, W.-ling, Krafczyk, N., & Peleska, J. (2024). Exhaustive property oriented model-based testing with symbolic finite state machines. *Science of Computer Programming*, *231*, 103005. https://doi.org/10.1016/j.scico.2023.103005

Jones, C. N., Kerrigan, E. C., & Maciejowski, J. M. (2008). On Polyhedral Projection and Parametric Programming. *Journal of Optimization Theory and Applications*, *138*(2), 207–220. https://doi.org/10.1007/s10957-008-9384-4

Jones, C., Kerrigan, E., & Maciejowski, J. (2007). Lexicographic perturbation for multiparametric linear programming with applications to control. *Automatica*, *43*(10), 1808–1816. https://doi.org/10.1016/j.automatica.2007.03.008

Jones, C. N., & Maciejowski, J. M. (2006). Reverse Search for Parametric Linear Programming. *Proceedings of the 45th IEEE Conference on Decision and Control*, 1504–1509. https://doi.org/10.1109/cdc.2006.377799

Jones, C. N., & Morari, M. (2006). Multiparametric Linear Complementarity Problems. *Proceedings of the 45th IEEE Conference on Decision and Control*, 5687–5692. https://doi.org/10.1109/CDC.2006.377797

Jost, A. M. (2023). *Implementing a Functional Logic Programming Language via the Fair Scheme* [Doctoral dissertation]. https://doi.org/10.15760/etd.3564

Kanade, A., Alur, R., Ivančić, F., Ramesh, S., Sankaranarayanan, S., & Shashidhar, K. C. (2009). Generating and Analyzing Symbolic Traces of Simulink/Stateflow Models. In *Computer Aided Verification* (pp. 430–445). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-02658-4_33

Kim, C. H. P., Marinov, D., Khurshid, S., Batory, D., Souto, S., Barros, P., & D'Amorim, M. (2013). SPLat: lightweight dynamic analysis for reducing combinatorics in testing configurable systems. *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, Esec/fse'13*, 257–267. https://doi.org/10.1145/2491411.2491459

King, J. C. (1976). Symbolic execution and program testing. *Communications of the ACM*, *19*(7), 385–394. https://doi.org/10.1145/360248.360252

Kitchenham, B., Madeyski, L., & Budgen, D. (2023). SEGRESS: Software Engineering Guidelines for REporting Secondary Studies. *IEEE Transactions on Software Engineering*, *49*(3), 1273–1298. https://doi.org/10.1109/tse.2022.3174092

König, K., & Mönnigmann, M. (2020). Accelerating MPC by online detection of state space sets with common optimal feedback laws. *Arxiv Preprint Arxiv:2009.08764*. https://arxiv.org/abs/2009.08764

Krafczyk, N., & Peleska, J. (2017). Effective Infinite-State Model Checking by Input Equivalence Class Partitioning. In *Testing Software and Systems* (pp. 38–53). Springer International Publishing. https://doi.org/10.1007/978-3-319-67549-7_3

Lagniez, J.-M., & Lonca, E. (2024). Leveraging Decision-DNNF Compilation for Enumerating Disjoint Partial Models. *Proceedings of the Twenty-First International Conference on Principles of Knowledge Representation and Reasoning*, 509–519. https://doi.org/10.24963/KR.2024/48

Li, K., Reichenbach, C., Smaragdakis, Y., Diao, Y., & Csallner, C. (2013). SEDGE: Symbolic example data generation for dataflow programs. *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 235–245. https://doi.org/10.1109/ase.2013.6693083

Lindblad, F. (2008). Property Directed Generation of First-Order Test Data. *Trends in Functional Programming*, *8*, 105–123. https://research.chalmers.se/publication/111863

Lu, S., & Bodík, R. (2023). Grisette: Symbolic Compilation as a Functional Programming Library. *Proceedings of the ACM on Programming Languages*, *7*(POPL), 455–487. https://doi.org/10.1145/3571209

Masden, M. (2025). Algorithmic Determination of the Combinatorial Structure of the Linear Regions of ReLU Neural Networks. *SIAM Journal on Applied Algebra and Geometry*, *9*(2), 374–404. https://doi.org/10.1137/24m1646996

Mitze, R., Dyrska, R., Konig, K., & Monnigmann, M. (2021). State space sets with common optimal feedback laws for nonlinear MPC. *2021 23rd International Conference on Process Control (PC)*, 55–59. https://doi.org/10.1109/pc52310.2021.9447473

Mokhov, A., Lukyanov, G., Marlow, S., & Dimino, J. (2019). Selective applicative functors. *Proceedings of the ACM on Programming Languages*, *3*(ICFP), 1–29. https://doi.org/10.1145/3341694

Nguyen, D. T., Kasmarik, K. E., & Abbass, H. A. (2020). Towards Interpretable ANNs: An Exact Transformation to Multi-Class Multivariate Decision Trees. *Arxiv Preprint Arxiv:2003.04675*. https://doi.org/10.48550/arXiv.2003.04675

Patrinos, P., & Sarimveis, H. (2010). A new algorithm for solving convex parametric quadratic programs based on graphical derivatives of solution mappings. *Automatica*, *46*(9), 1405–1418. https://doi.org/10.1016/j.automatica.2010.06.008

Petersen, K., Vakkalanka, S., & Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, *64*, 1–18. https://doi.org/10.1016/j.infsof.2015.03.007

Phan, Q.-S., & Malacaria, P. (2015). All-Solution Satisfiability Modulo Theories: Applications, Algorithms and Benchmarks. *2015 10th International Conference on Availability, Reliability and Security*, 100–109. https://doi.org/10.1109/ares.2015.14

Pingali, K., & Arvind. (1985). Efficient demand-driven evaluation. Part 1. *ACM Transactions on Programming Languages and Systems*, *7*(2), 311–333. https://doi.org/10.1145/3318.3480

Porncharoenwase, S., Nelson, L., Wang, X., & Torlak, E. (2022). A formal foundation for symbolic evaluation with merging. *Proceedings of the ACM on Programming Languages*, *6*(POPL), 1–28. https://doi.org/10.1145/3498709

Qi, D., Nguyen, H. D. T., & Roychoudhury, A. (2013). Path exploration based on symbolic output. *ACM Transactions on Software Engineering and Methodology*, *22*(4), 1–41. https://doi.org/10.1145/2522920.2522925

Rada, M., & Černý, M. (2018). A New Algorithm for Enumeration of Cells of Hyperplane Arrangements and a Comparison with Avis and Fukuda's Reverse Search. *SIAM Journal on Discrete Mathematics*, *32*(1), 455–473. https://doi.org/10.1137/15m1027930

Robinson, H., Rasheed, A., & San, O. (2020). Dissecting Deep Neural Networks. *Arxiv Preprint Arxiv:1910.03879*. https://arxiv.org/abs/1910.03879

Rosner, N., Geldenhuys, J., Aguirre, N., Visser, W., & Frias, M. F. (2015). BLISS: Improved Symbolic Execution by Bounded Lazy Initialization with SAT Support. *IEEE Transactions on Software Engineering*, *41*(7), 639–660. https://doi.org/10.1109/TSE.2015.2389225

Runciman, C., Naylor, M., & Lindblad, F. (2008). Smallcheck and lazy smallcheck: automatic exhaustive testing for small values. *Proceedings of the First ACM SIGPLAN Symposium on Haskell, Icfp08*, 37–48. https://doi.org/10.1145/1411286.1411292

Ryan, K., & Sturton, C. (2023). Sylvia: Countering the Path Explosion Problem in the Symbolic Execution of Hardware Designs. *2023 Formal Methods in Computer-Aided Design (FMCAD)*, 109–120. https://doi.org/10.34727/2023/ISBN.978-3-85448-060-0_19

Santelices, R., & Harrold, M. J. (2010). Exploiting program dependencies for scalable multiple-path symbolic execution. *Proceedings of the 19th International Symposium on Software Testing and Analysis, ISSTA '10*, 195–206. https://doi.org/10.1145/1831708.1831733

Schlüter, M., & Steffen, B. (2024). Affinitree: A Compositional Framework for Formal Analysis and Explanation of Deep Neural Networks. In *Tests and Proofs* (pp. 148–167). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-72044-4_8

Sen, K., Necula, G., Gong, L., & Choi, W. (2015). MultiSE: multi-path symbolic execution using value summaries. *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, Esec/fse´15*, 842–853. https://doi.org/10.1145/2786805.2786830

Serra, T., Tjandraatmadja, C., & Ramalingam, S. (2018). Bounding and Counting Linear Regions of Deep Neural Networks. *Proceedings of the 35th International Conference on Machine Learning*, *80*, 4558–4566. https://proceedings.mlr.press/v80/serra18b.html

Sleumer, N. (1998). Output-sensitive cell enumeration in hyperplane arrangements. In *Algorithm Theory — SWAT´98* (pp. 300–309). Springer Berlin Heidelberg. https://doi.org/10.1007/bfb0054377

Spallitta, G., Sebastiani, R., & Biere, A. (2024). Disjoint Partial Enumeration without Blocking Clauses. *Proceedings of the AAAI Conference on Artificial Intelligence*, *38*(8), 8126–8135. https://doi.org/10.1609/aaai.v38i8.28652

Spallitta, G., Sebastiani, R., & Biere, A. (2025). Disjoint projected enumeration for SAT and SMT without blocking clauses. *Artificial Intelligence*, *345*, 104346. https://doi.org/10.1016/j.artint.2025.104346

Spjøtvold, J., Tøndel, P., & Johansen, T. A. (2007). Continuous Selection and Unique Polyhedral Representation of Solutions to Convex Parametric Quadratic Programs. *Journal of Optimization Theory and Applications*, *134*(2), 177–189. https://doi.org/10.1007/s10957-007-9215-z

Tran, H.-D., Manzanas Lopez, D., Musau, P., Yang, X., Nguyen, L. V., Xiang, W., & Johnson, T. T. (2019). Star-Based Reachability Analysis of Deep Neural Networks. In *Formal Methods – The Next 30 Years* (pp. 670–686). Springer International Publishing. https://doi.org/10.1007/978-3-030-30942-8_39

Vincent, J. A., & Schwager, M. (2021). Reachable Polyhedral Marching (RPM): A Safety Verification Algorithm for Robotic Systems with Deep Neural Network Components. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 9029–9035. https://doi.org/10.1109/icra48506.2021.9561956

Voogd, E., Kløvstad, Å. A. A., Johnsen, E. B., & Wąsowski, A. (2025). Compositional symbolic execution semantics. *Theoretical Computer Science*, *1044*, 115263. https://doi.org/10.1016/j.tcs.2025.115263

Wang, H., Liu, T., Guan, X., Shen, C., Zheng, Q., & Yang, Z. (2017). Dependence Guided Symbolic Execution. *IEEE Transactions on Software Engineering*, *43*(3), 252–271. https://doi.org/10.1109/tse.2016.2584063

Wang, Y., Mary, A., Sagot, M.-F., & Sinaimeri, B. (2021). A General Framework for Enumerating Equivalence Classes of Solutions. *29th Annual European Symposium on Algorithms (ESA 2021), Leibniz International Proceedings in Informatics*, *204*, 80:1–80:14. https://doi.org/10.4230/LIPIcs.ESA.2021.80

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, EASE '14*, 1–10. https://doi.org/10.1145/2601248.2601268

Wong, C.-P., Meinicke, J., Lazarek, L., & Kästner, C. (2018). Faster variational execution with transparent bytecode transformation. *Proceedings of the ACM on Programming Languages*, *2*(OOPSLA), 1–30. https://doi.org/10.1145/3276487

Xia, L.-yao, Israel, L., Kramarz, M., Coltharp, N., Claessen, K., Weirich, S., & Li, Y. (2024). Story of Your Lazy Function's Life: A Bidirectional Demand Semantics for Mechanized Cost Analysis of Lazy Programs. *Proceedings of the ACM on Programming Languages*, *8*(ICFP), 30–63. https://doi.org/10.1145/3674626