Salus: Modular Trading and Solving Infrastructure for Decentralized Markets

Technical Whitepaper

John Whitton

Document version: 0.2 Draft · 2026-08-14

Abstract

Decentralized-market infrastructure makes decisions while its inputs are changing. Liquidity moves while a system ingests state, maintains topology, evaluates routes, estimates costs, simulates execution, and decides whether a result is still usable. Salus treats that as a systems problem before treating it as a trading problem.

The implementation separates durable topology from live state, retains a warm catalogue of known routes, maps changed components back to affected routes, evaluates exact inputs against labelled state, and makes execution another guarded lifecycle. Its proposition is not a claim of a universal profitable strategy. It is that a decision is useful only when the system can explain its state boundary, inputs, policy, handoff, and later evidence.

Salus has implemented foundations for liquidity mapping, state-aware route evaluation, bounded runtime coordination, dry-run and submission preparation, replay, and retained diagnostics. It can support arbitrage-oriented evaluation and common foundations for liquidation, solving, portfolio management, market making, and additional strategies. Profitable after-gas production execution has not been established. Appendix A records engineering capacity, not an opportunity, execution, settlement, or profit result.

Contents

1 Overview

Salus began with the practical question of whether a large, changing liquidity surface could be inspected quickly enough to support an informed decision. The answer was not simply to enumerate more cycles or start more workers. A route catalogue may be durable while reserves, balances, fees, quote context, and execution conditions are perishable. Treating those facts as one "opportunity" makes a system hard to inspect and easy to overstate.

The current implementation divides responsibility: ingestion owns ordered observations; persistence owns durable read models; discovery owns topology and known route membership; evaluation owns a fixed-input model; execution owns a guarded handoff; and evidence owns later observations. A result can be rejected for incomplete state, invalid arithmetic, unsupported semantics, or a newer block. Those are useful engineering results, not failures to hide.

Salus evolved from a solver-oriented prototype into a modular Rust runtime with a clearer separation between topology, state, calculation, policy, and outcome. The current limitation is clear: retained runs establish bounded implementation capacity. The latest cited retained runs selected zero after-gas routes; they do not establish profitable execution.

The Salus Work gateway is the concise project case study. This Whitepaper is the self-contained long-form system narrative; focused Research and Architecture pages retain their specialist evidence and reusable patterns.

2 Opportunity

Liquidity is fragmented across protocols with different state models and update behaviour. A searcher, solver, protocol operator, treasury, liquidator, or market maker needs a reusable answer to prior questions: what changed; which known decisions could it affect; which inputs are current enough to calculate; what does a model say for an exact amount; and what later evidence is needed before any external action is allowed.

This is infrastructure, not an assertion that every participant has the same strategy. Arbitrage focuses on one bounded conversion. Liquidation adds account health and collateral eligibility. Solving adds a user mandate. Portfolio work adds time, exposure, capital, and accounting. Market making adds two-sided quotes, inventory, adverse selection, and hedging. Shared infrastructure is useful precisely because it does not erase those economic differences.

Speed alone is insufficient. Freshness, correct state, cost estimation, funding, transaction construction, inclusion, and inventory or counterparty risk can each determine whether a positive calculation is useful. Evidence- first development can slow an initial implementation, but reduces the risk of optimizing a strategy whose inputs, execution path, or economics cannot support a real decision.

3 Solution

3.1 Liquidity Mapping

Salus normalizes observed liquidity components and token relationships into durable read models and a graph-shaped runtime representation. It enumerates a bounded catalogue and records component-to-route membership. When a component changes, the runtime recalls known affected routes rather than rebuilding the universe. Membership narrows work; it is not a cached profitability or execution decision.

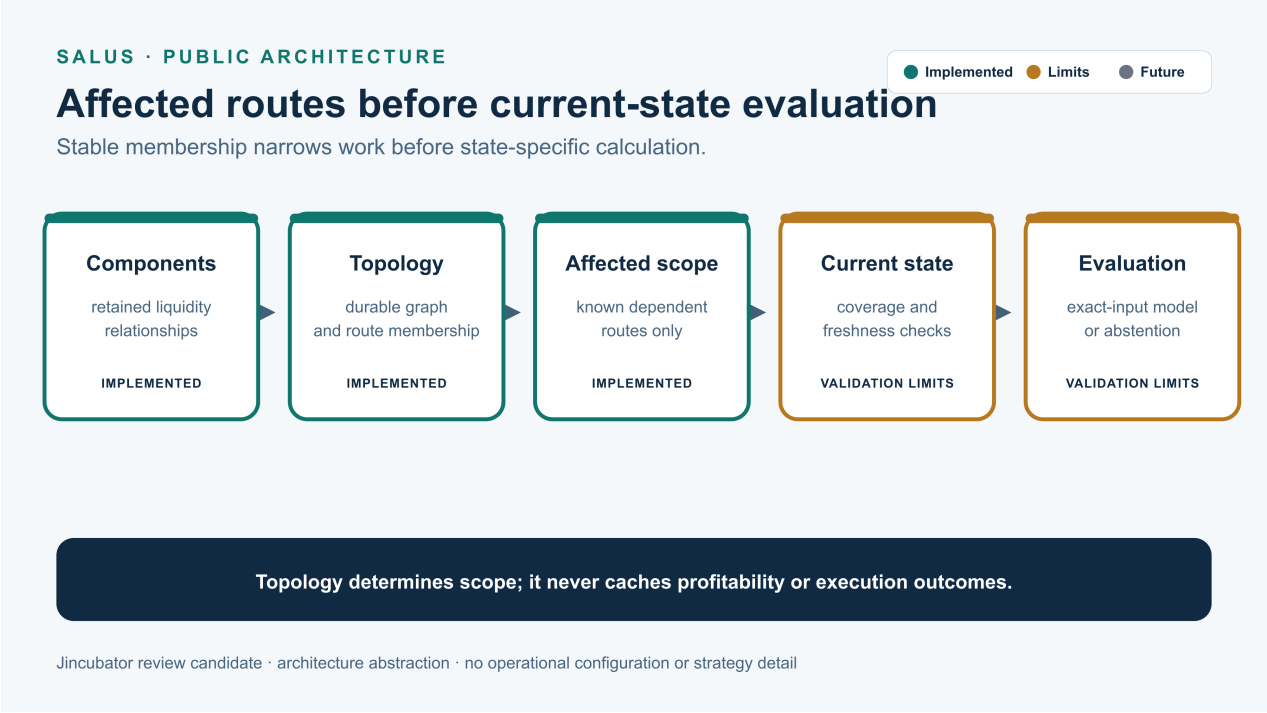

Figure 1. A topology index narrows work to known affected routes; it does not cache profitability, gas, or an execution result.

3.2 Flash Loans

For supported arbitrage transaction shapes, Salus can attach one optional atomic-funding source to a route. The funding asset is the route's start asset. The attachment step excludes a source already used by the route and incompatible V3/V4 overlaps, then deterministically orders the remaining eligible sources by represented fee, provider priority, and component identity. That is a source-local eligibility policy, not a complete comparison of inclusion probability or total execution cost. [1] [2]

Funding metadata travels with the route into evaluation and execution planning. Known available balance can bound the principal plus rounded funding fee; absent or incomplete funding information is not treated as a free loan. The evaluator accounts for funding cost separately, and dry-run/live planning require complete metadata, repayment safeguards, a current handoff, and the configured preflight path. For supported atomic transactions this can provide trading principal and repay it in the same transaction. It does not remove infrastructure, RPC, operational, or gas expenditure; an unavailable loan, stale state, failed preflight, non-inclusion, or reverted transaction can still make the decision unusable or costly.

3.3 State Ingestion

Typed stream and replay adapters feed block-labelled state changes into the runtime. The ingestion boundary preserves source identity, ordering context, and the distinction between an observed delta and complete coverage. Route preparation verifies every required leg before evaluation. Incomplete or unsupported input is an explicit abstention, not a partial success.

3.4 Strategy Evaluation

Evaluation accepts an ordered route, block-scoped state, and exact input. Each leg is simulated with its protocol state and checked arithmetic. Missing state, invalid amounts, unsupported paths, and stale generations become typed rejections. At a high level an arbitrage model asks whether $\mathrm{net}(x)$ remains positive after input, protocol fees, funding, estimated gas, and other represented costs. That calculation is a model, not a trade or profit claim.

3.5 Execution

Execution separately rechecks freshness, cost context, authorization, and preflight conditions. It can construct dry-run or submission intent, retain the handoff, and reconcile later evidence. Submission is not inclusion; inclusion is not finality; finality is not settlement; settlement is not realized economic profit. Missing evidence remains unresolved. Private provider configuration, bidding logic, thresholds, route-selection details, opportunity identities, and active parameters are excluded from this public description.

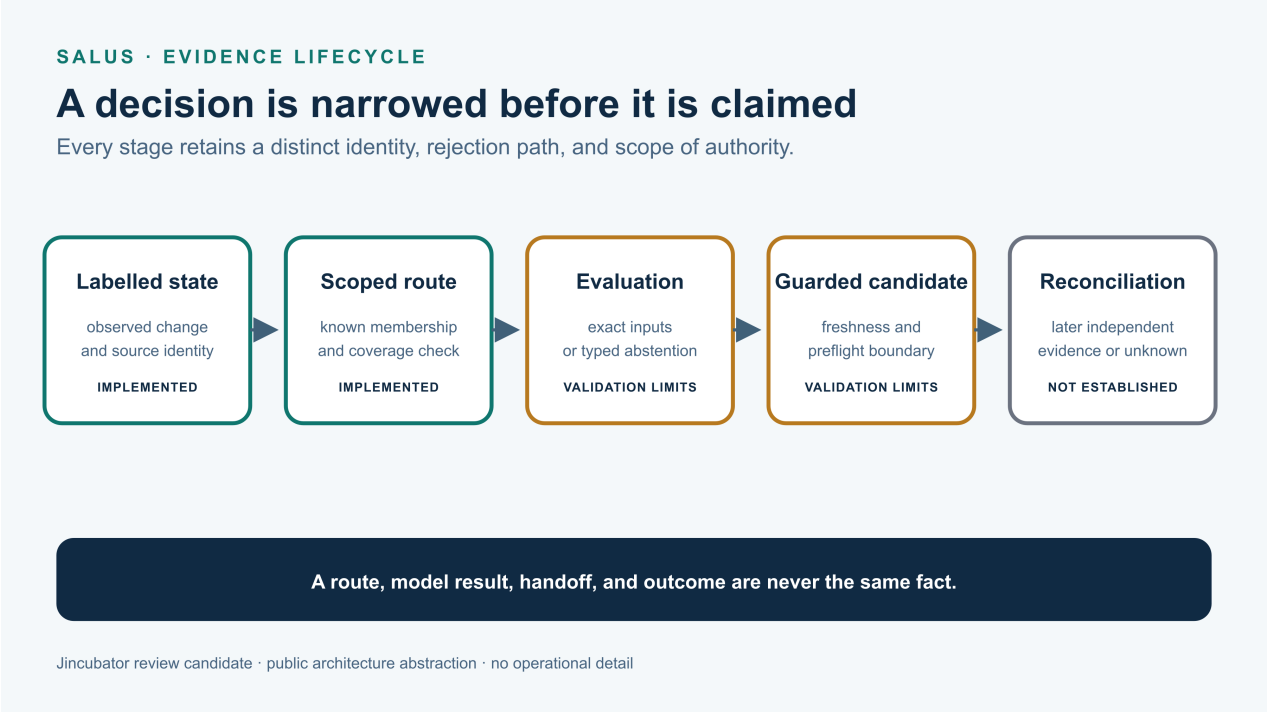

Figure 2. Every transition either narrows a candidate set or adds a distinct evidence record.

4 Use Cases

Use case	Shared Salus contribution	Current status
Arbitrage	route scope, fixed-input evaluation, cost classification, guarded handoff	implemented with validation limits
Liquidation	state, evaluation, funding, execution, and evidence foundations	proposed / future
Solving	state, simulation, mandate validation, and evidence boundaries	proposed / future
Portfolio Management	read models, accounting facts, evaluator extensions, and risk boundaries	proposed / future
Market Making	state, pricing/evaluation primitives, execution controls, and evidence	proposed / future
Additional Strategies	reusable topology, state, policy, and evidence contracts	proposed / future

4.1 Arbitrage

An arbitrage actor asks whether an ordered conversion for one exact input produces a positive modelled result after represented costs. Salus contributes topology, affected-route scope, state coverage, per-leg simulation, bounded amount search, cost classification, and a guarded handoff. It abstains when a route lacks coverage, a protocol state is unsupported, a cost input is absent, an amount is invalid, or newer state supersedes the work.

The public implementation record identifies promoted support classes for Uniswap V2, Sushiswap V2, PancakeSwap V2, Uniswap V3, PancakeSwap V3, plain Uniswap V4, and filtered VM-backed Balancer V2 behaviour. This is a support/simulation classification, not a claim that every deployment, pool, or route is currently indexed or executable. Public retained topology evidence is aggregate rather than name-by-name: 3,415 graph tokens, 4,552 retained liquidity components, and 2,401,108 maximum-four-hop catalogue routes. Configuration, experiments, plans, and restricted identities remain internal. Unsupported families reject rather than silently degrading.

The model can use atomic funding where a supported transaction shape permits it, including a flash-loan boundary. It does not require upfront trading principal for that modelled structure, but still requires infrastructure, RPC, operational, and gas expenditure; unsuccessful transactions can incur unreimbursed costs. The next evidence gate is a traceable after-gas candidate with independently observed execution and settlement. It is not established.

4.2 Liquidation

Liquidation asks whether an account crosses a protocol-defined eligibility boundary, whether debt can be repaid, whether collateral can be recovered and converted, and whether financing, competition, gas, and timing leave a viable model. Morpho and Aave are public examples of distinct liquidation mechanics; they are not evidence that Salus is integrated with or operating on either. Morpho's published liquidation material illustrates why account health, oracle state, debt, collateral, and recovery rules must stay protocol-specific. [3]

Salus can reuse ingestion, topology, exact calculations, funding boundaries, dry-run controls, and reconciliation. A liquidation extension needs position and oracle state, eligibility/health calculation, debt repayment, collateral recovery, routing, accounting, and competition controls. These are proposed / future capabilities. The next evidence gate is a replayable protocol-specific eligibility and recovery model before a bounded shadow or canary process.

4.3 Solving

Solving is not arbitrage with a different name. An intent owner sets authority constraints on assets, price, amount, fees, time, and settlement. A solver may optimize only within that mandate. CoW Protocol, UniswapX, and 1inch are public examples of ecosystem surfaces where user authority changes the decision boundary; this document claims no integration or commercial relationship. [4] [5] [6]

Salus could contribute state, quote/simulation, exact calculations, mandate validation, and execution evidence. It would require an intent adapter, canonical mandate representation, protocol-specific settlement logic, and a proof that a candidate preserves constraints. It is proposed / future. The next evidence gate is replayable intent and constraint validation with explicit abstention on incomplete mandate or state data.

4.4 Portfolio Management

Portfolio management evaluates exposure over time rather than one route. A treasury, allocator, or inventory owner needs capital, financing, concentration, liquidity, accounting, and risk constraints. Salus can reuse read models, deterministic accounting facts, fixed-input evaluation, and execution evidence, but it has no published portfolio policy.

The approved public-safe PEX1 boundary is limited to the lesson that apparent transaction profit may not explain portfolio or inventory-backed economics. Operator identities, transaction-specific economics, ordering fees, and multi-transaction profitability analysis remain restricted. Portfolio Management is proposed / future; its next gate is reproducible portfolio state and counterfactual accounting, not a route-evaluation benchmark.

4.5 Market Making

Market making maintains two-sided quotes while managing bid-ask spread, inventory, adverse selection, funding, hedging, rebalancing, and protocol liquidity. Arbitrage can align venues but is not the market-making strategy. Salus provides current state, controlled evaluation, simulation, durable accounting facts, and guarded execution/evidence. A market-making extension would need inventory, quote, pricing, exposure, hedge, and order-lifecycle models. It is proposed / future. The next evidence gate is an offline or passive quote/inventory model with retained mark, fill, and exposure facts.

4.6 Additional Strategies

Additional public-safe strategy families include cross-protocol conversion, liquidation conversion, intent fulfilment, inventory rebalancing, and protocol-liquidity support. Analytics, replay, telemetry, evidence, and operator controls support strategies; they are not strategies themselves. A new strategy must name its actor, inputs, objective, capital and fee model, authority boundary, abstention conditions, execution path, and independently observed outcome.

5 What Has Been Built

Liquidity mapping. Salus persists normalized topology, warms a route catalogue, and records component-to-route membership so changed components can recall known affected routes. The published aggregate snapshot contains 3,415 graph tokens, 4,552 retained liquidity components, and 2,401,108 maximum-four-hop catalogue routes. These are topology measures, not opportunities or profits.

Flash loans. For a supported route shape, Salus attaches deterministic funding metadata for the route start asset, checks source compatibility, carries fee and available-balance information where it is known, and applies repayment guards during dry run and live planning. A supported atomic transaction can borrow and repay principal within the transaction; gas, infrastructure, operational expenditure, route eligibility, liquidity, funding fee, repayment, and preflight remain separate constraints.

Profitability calculation. Exact input/output amounts combine with known protocol fees, estimated funding cost, estimated gas, and other represented costs. A retained calculation is a modelled result, never shorthand for a realized economic outcome.

Optimal amount calculation. A bounded search explores feasible amounts, rejects invalid or unsupported regions, and selects only current, policy-eligible work. It abstains when state, cost, arithmetic, or freshness is incomplete.

Execution. Dry-run, preflight, submission preparation, and reconciliation are distinct surfaces. A simulation, acknowledgement, or receipt is not silently promoted into final economic proof.

Runtime and evidence infrastructure. Bounded queues, stage ownership, freshness checks, replay inputs, durable read models, diagnostics, and retained artifacts make the runtime inspectable without turning a capacity measure into a commercial claim.

6 Future Work

Future work is direction and evidence gates, not a delivery commitment. Its priorities will be driven by profitability and market evidence that is not yet complete; a catalogue or evaluator speed does not establish that evidence.

Profitability analysis. A provisional competitor-analysis program can measure transaction-local observable inventory change and visible costs only when identity, trace, valuation, and accounting boundaries are explicit. A stronger conclusion still needs realized accounting, context attribution, inventory, counterfactuals, and reconciliation; unmerged research is not a completed Salus result.

Machine learning. The route-survival direction is a replayable research surface: features must be point-in-time correct, labels immutable, and capture provenance retained. Models begin with abstention and passive analysis; they do not receive execution authority merely because a prediction is available.

Arbitrage advancement. The next path is after-gas selection, current preflight, guarded submission, settlement reconciliation, and realized- economics evidence. The cited retained runs selected zero after-gas routes, so profitable or repeatable execution remains unestablished.

Liquidation support. Morpho is the proposed first protocol-specific model, with Aave as a comparison. It requires bounded position and oracle state, health/eligibility, funding, collateral conversion, settlement, and replay evidence before shadow, canary, or live claims.

Solving support. CoW Protocol, UniswapX, and 1inch need distinct mandate, quote, settlement, and evidence adapters. A valid solver candidate must remain within the user's mandate and abstain when mandate or state evidence is incomplete.

Portfolio management. Capital, inventory, exposure, accounting, risk, and counterfactual boundaries must be explicit. Route-level model output cannot stand in for a portfolio policy or a reconciled portfolio result.

Market making. Two-sided pricing, inventory, adverse selection, hedging, and protocol-liquidity support require their own state and evidence contracts. The first useful proof is passive or offline quote/inventory evidence, not an arbitrage benchmark.

Additional strategies. The authoritative future-work record includes protocol-liquidity support, MEV-aware solving, volatility, market intelligence, and other strategy research only where each has a defined actor, objective, capital/fee model, authority boundary, and abstention conditions.

AI-assisted investigation. Passive research, replay, postmortems, and evidence repair may improve understanding before they influence policy. Any later policy use needs a separately verified decision contract.

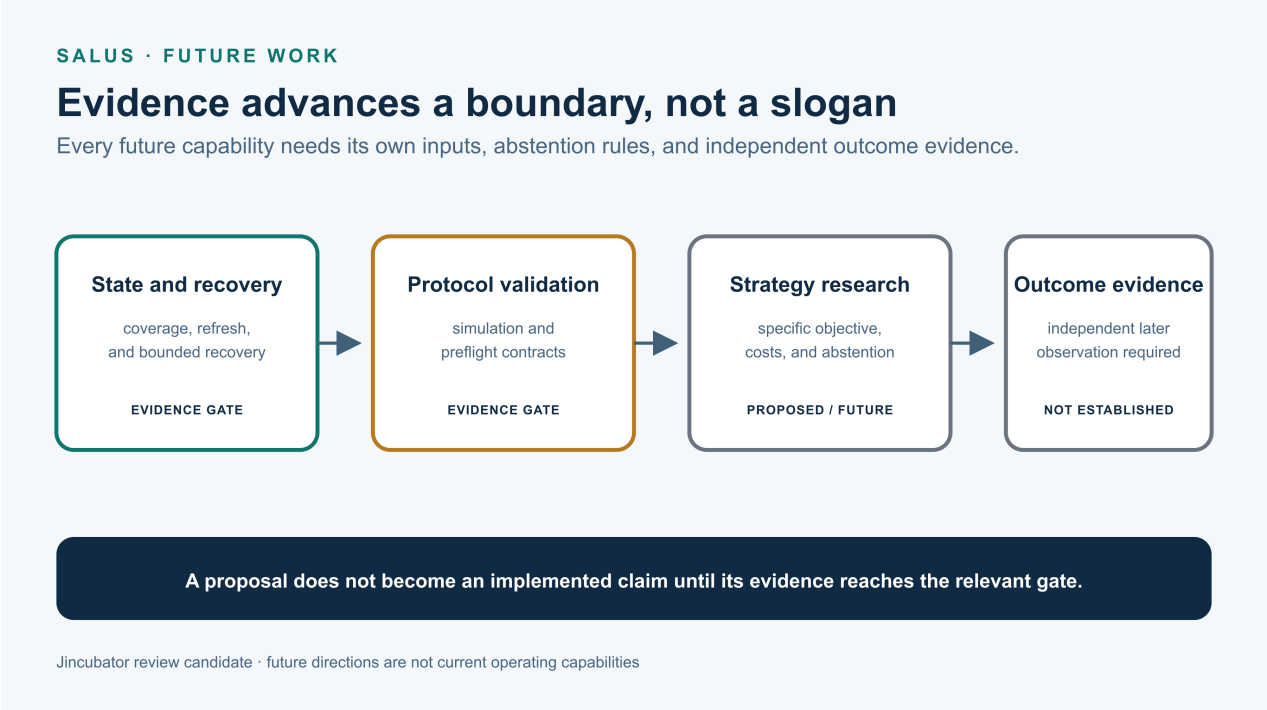

Figure 3. Future directions advance only when their own evidence gates, not a shared platform narrative, are satisfied.

7 Appendix A — Performance and Engineering Evidence

Retained measurements describe a named engineering workload. They are not a production scorecard and cannot be compared without workload, chain, implementation revision, route definition, environment, and time boundary. Canonical evidence pins and detailed retained artefacts remain in private governance records; this public summary retains approved aggregates.

Measurement	Retained result	Establishes	Does not establish
Ethereum graph tokens	3,415	topology scale in the cited snapshot	current coverage, liquidity, or opportunity count
Ethereum liquidity components	4,552	normalized retained component scope	live availability or executable depth
Maximum-four-hop catalogue routes	2,401,108	bounded route-universe construction	evaluated, profitable, selected, or submitted routes
Ethereum sustained evaluation rate	33,578 evaluations/s	evaluator capacity for the cited workload	execution, inclusion, settlement, or profit
Base evaluation measurements	bounded retained sustained/peak runs	workload-scoped capacity	cross-chain or commercial throughput promise

A **catalogue route** is known topology. A **route evaluation** is model work for an input and state frame. An **opportunity** requires a defined policy. A **selected after-gas route** requires cost and selection evidence. A **submitted transaction**, **execution**, and **realized profit** require later independent observations. The cited retained runs selected zero after-gas routes, so this appendix does not establish profitable execution.

Historical material used different snapshots, route universes, chains, worker allocations, and reporting surfaces. Its durable conclusion is architectural: affected-route scope, bounded workers, queue wait/service observations, stale work, and replay artifacts make bottlenecks inspectable. One measurement does not

retroactively validate another environment or commercial outcome.

8 Appendix B — Mathematical Models and Route Evaluation

Market graph model. Salus builds a deterministic directed adjacency map from normalized tokens and components. A multi-token component contributes a directed edge for each ordered input/output token pair; the edge retains the component identity and protocol classification. The catalogue is therefore a durable topology record: it says which ordered paths are known, not whether their current state is complete or economically useful.

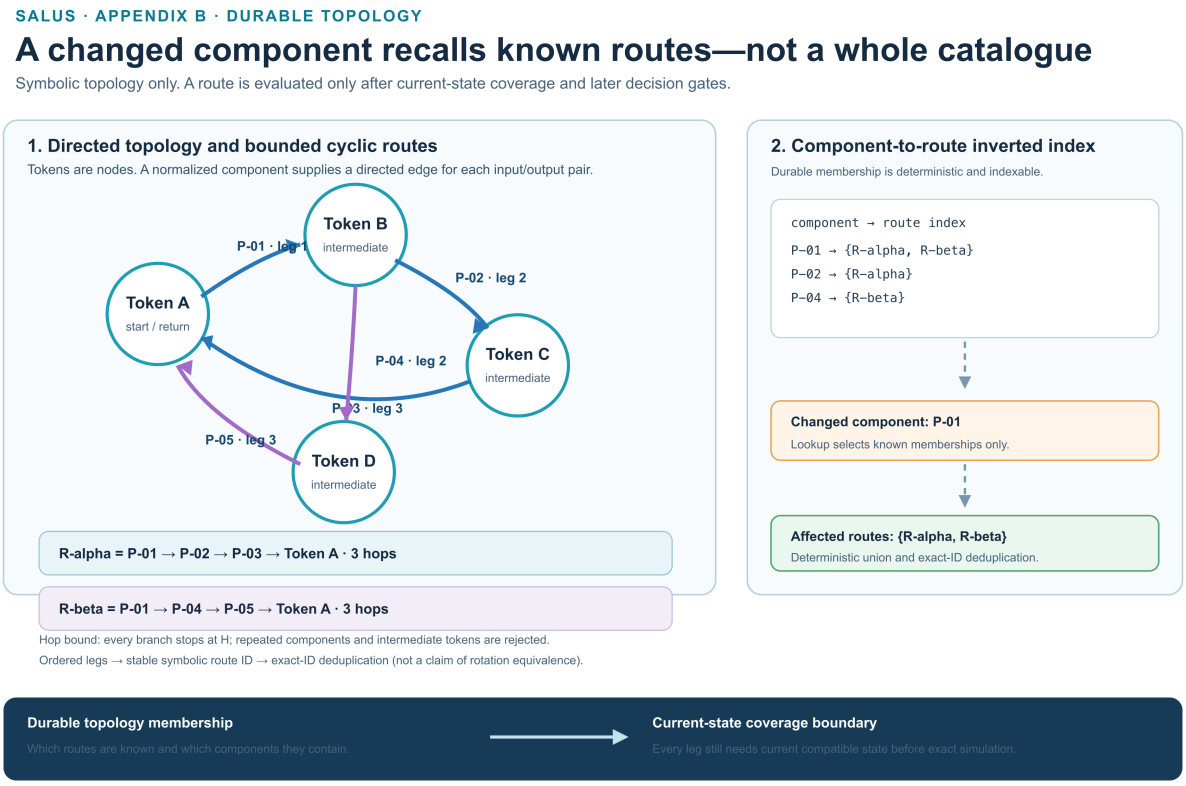

Figure 4. A symbolic, bounded route catalogue. Component P-01 is shared by two known routes, so a P-01 change recalls R-alpha and R-beta from the deterministic component-to-route index. Current-state coverage is a later gate; topology membership does not cache profitability.

Route construction. The verified discovery method is **bounded depth-first search with backtracking**. Discovery begins from each ordered start token and walks outgoing directed edges. `RouteSearchState` carries the path, ordered legs, visited intermediate tokens, and used components. `max_hops` stops a branch before it grows beyond its configured bound. An edge may close a route only when it returns to the start token after at least one prior leg; otherwise repeated intermediate tokens and repeated components are rejected. The result is a bounded cyclic candidate, not a live opportunity.

The implementation does not show a separate rotation-equivalence canonicalizer. Instead, it uses the ordered leg representation as the canonical input to a stable identity: the component sequence, plus optional funding attachment fields, is encoded and Keccak-256 hashed. A `BTreeMap` retains one candidate

per exact route identity, so duplicate serialisations are removed deterministically. This is ID-based deduplication; this Whitepaper does not claim that every rotation of an economically similar cycle is collapsed into one identity.

Conceptual algorithm.

1. Normalize tokens and components, then build ordered directed adjacency.
2. For each eligible start token, run bounded depth-first search with backtracking; close only a return-to-start cycle within `max_hops`.
3. Reject repeated components and intermediate tokens, retain ordered legs, and create a stable route identity; deduplicate equal identities.
4. Persist validated route summaries and hydrate an in-memory catalogue whose `RouteIndex` maps route IDs by component, token, and start token.
5. On a labelled component change, take the deterministic `BTreeSet` union of affected route IDs. A generation-tagged acceleration cache may serve the scope only when it agrees with the catalogue; otherwise lookup falls back to the catalogue index.
6. Check catalogue generation and current-state coverage before exact-input per-leg simulation. Missing state, invalid arithmetic, unsupported protocol semantics, or a bounded-admission decision produces a typed skip.
7. Perform bounded amount search, attach compatible funding and cost terms, recheck freshness, then select or return typed abstention. A selected model result is only a guarded execution handoff.

The trade-off is deliberate. Discovery cost grows with branching and the hop bound, while a warm route catalogue avoids rediscovering topology for every state change. The inverted index uses memory for direct recall of known affected routes. Exact simulation costs more than static edge tests, and explicit candidate, time, amount, and worker bounds make resource use inspectable. Freshness may still invalidate completed work that was otherwise calculated correctly.

Affected-route indexing. `RouteIndex` holds routes by ID plus ordered component, token, and start-token memberships. Insertion adds every route leg and, where present, the optional flash component to the component membership. For a set of changed components, lookup unions the associated sorted ID sets. That narrows computation to known memberships; it neither proves state coverage nor a positive result.

Bellman–Ford comparison. Bellman–Ford can identify theoretical negative cycles when exchange relationships are reduced to static negative-log weights under simplifying assumptions [7]. That is useful for graph analysis, but it does not meet Salus's execution-grade calculation contract. This is a design suitability comparison, not a claim that an earlier team decision rejected the algorithm.

Concern	**Static Bellman–Ford / negative-log abstraction**	**Verified Salus approach**
Edge weight	One static scalar	Amount- and state-dependent protocol transition
Slippage and price impact	Not represented by one fixed weight	Exact-input per-leg simulation
Concentrated liquidity	Difficult to flatten safely	Protocol-specific state transition
Fees, rounding, and units	Approximated in weights	Checked at each ordered leg
Invalid regions	Awkward or static	Typed invalid or abstention outcome
Route universe	Detected from a current weighted graph	Bounded durable route catalogue

Concern	Static Bellman–Ford / negative-log abstraction	Verified Salus approach
Changed state	Typically rerun graph calculation	Affected-route lookup, then evaluation
Funding and gas	External to cycle detection	Explicit funding and cost terms
Freshness	External	Generation and block-scoped validation
Execution evidence	Absent	Guarded handoff followed by later evidence

Fixed-input route evaluation. For ordered route input x_0, route-specific protocol state s_i, and leg function f_i, sequential evaluation is:

$$x_{i+1} = f_i(x_i, s_i)$$

Each leg consumes the prior output. The evaluator preserves token units and decimals, represents protocol fees at the relevant leg, uses checked arithmetic, and treats absent state, invalid amounts, or non-representable outputs as invalid evaluations rather than partial answers.

Optimal-amount search. The bounded input-search objective is:

$$x^* = \arg\max_{x \in D} \mathrm{net}(x)$$

The feasible domain D includes represented token amounts, route-specific state coverage, protocol compatibility, funding availability, and declared search bounds. The verified search uses bounded candidate probing, expansion, and refinement rather than claiming a global optimum. Search may stop because an interval is exhausted, a budget is reached, a region is invalid, or evidence is insufficient. It must not assume monotonicity: fees, rounding, price impact, and route composition can create non-linear or invalid regions.

Flash-loan selection. For a route r, candidate provider p, and input x, the funding boundary admits only a compatible source with the route start asset, represented fee, and—where it is observed—enough available balance:

$$\mathrm{eligible}(r, p, x) = \mathrm{compatible}(r, p) \wedge L_p \geq x + \mathrm{fee}_p(x)$$

The source pin implements deterministic ordering among eligible sources; the available-liquidity term is only as current as the observed source state.

Profitability calculation. The modelled net result includes funding and is:

$$\mathrm{net}(x) = x_n - x - \mathrm{protocolFees}(x) - \mathrm{fundingCost}(x) - \mathrm{gasCost}(x) - \mathrm{externalCosts}(x)$$

A positive $\mathrm{net}(x)$ is not a selected candidate or outcome. A missing/stale cost, unsupported path, or state change causes abstention.

Freshness, rejection, and abstention. For a changed component set ΔC, affected-route selection is:

$$R(\Delta C) = \bigcup_{c \in \Delta C} \mathrm{RouteIndex}(c)$$

The changed-component scope supplies deterministic membership, not profitability. The union is bounded before evaluation; hydration/generation checks establish catalogue membership and coverage checks establish usable state for each leg.

Evaluation-to-execution handoff. The deterministic flow exposes rejection and abstention gates rather than presenting one unconditional success path.

Figure 5. State coverage, invalid simulation, incompatible funding or costs, and stale work each terminate in typed abstention. Guarded handoff is not execution, settlement, or realized profit.

The algorithm deliberately separates scope, lookup, coverage, fixed-input simulation, bounded search, cost classification, freshness validation, selection/abstention, and guarded execution handoff. It does not turn a topology index into a profitability cache or a model result into a transaction outcome.

9 References

1. Uniswap Labs, 2026. Flash integrations for Uniswap v3. Accessed 2026-08-14.
2. Uniswap Labs, 2026. Flash accounting in Uniswap v4. Accessed 2026-08-14.
3. Morpho Association, 2026. Liquidation. Accessed 2026-08-14.
4. CoW Protocol, 2026. CoW Protocol documentation. Accessed 2026-08-14.
5. Uniswap Labs, 2026. UniswapX overview. Accessed 2026-08-14.
6. 1inch, 2026. 1inch documentation. Accessed 2026-08-14.
7. Bellman, Richard, 1958. On a Routing Problem, *Quarterly of Applied Mathematics* 16(1), 87–90.