

Persistent Scaffold Computation

Draft 4 correction folio - 10 September 2026

What remains established

The paper's generic simultaneous-batch compiler and its conditional opaque-element staging theorem remain useful. They explain how a bounded finite-record transition can be packed into one scaffold node once the transition's implementation interface and resource bounds have been established.

What is corrected

The earlier edition treated Kaplan-Tarjan's catenable-steque theorem as if it also supplied FPRD's stronger finite-record interface: explicit bounded allocation, bounded access to old addresses, opacity of the complete stored cell, and one concrete strict implementation satisfying all of these at once. The source theorem does not state that conjunction. Example 9.6 therefore gives a **candidate application**, not a completed second proof of the even-palindrome result.

The remaining bridge

1	Give one literal finite tagged pointer program for the required persistent catenable sequence operations.
2	Prove its correspondence with the abstract sequence semantics and regularity invariant.
3	Audit every branch for opacity, fresh allocation, and access to old addresses; symbolic bounds are acceptable.
4	Redo the palindrome client's resource inventory against that exact interface.

A separate client audit narrows the required interface to empty, single, uncons, and catenation. That simplification improves the specification but does not remove the hard catenation bridge. The established real-time transfer proof of the palindrome result is unaffected.

Scope of this correction

This folio corrects the direct-palindrome application and its source attribution. It is not an external review, a complete proof audit of the rest of this paper, or a claim that the remaining bridge is impossible. The preserved Draft 3 body follows for historical and mathematical context; where it conflicts with this folio, this folio controls.

Persistent Scaffold Computation: Causal Presentations, Sewing, and Feedback

Joshua Gay
FPRD Lab, Insight Forge

Draft 3, 23 August 2026

Abstract

Scaffolding automata were introduced by Loff, Moreira, and Reis to characterize the languages of parsing expression grammars: a scaffolding automaton reads its input once and, at every letter, appends one new node of bounded degree to a permanent graph, choosing the new node’s edges among the nodes within a fixed distance of the previous one. Their language-theoretic power is known. This paper studies the graphs themselves: what a construction history builds, what an observer standing at the newest node can learn by following edges, when two different histories are indistinguishable to such an observer, and—the main question—which *local* rewrites of a history connect it to every history that is indistinguishable from it.

We separate four things that are easy to conflate: the physical history (which nodes exist and which edges they carry), its *behavioral figure* at each moment (the smallest graph with the same observations), the *strong trace* (the sequence of all those figures), and the *final figure* alone. For fixed degree and distance we prove that the reachable part of a history is always totally ordered by reachability; that on such graphs a simple pairwise *guarded fold* rule computes the behavioral figure; and, as the main theorem, that two rewrites—renaming an edge by a shorter path to the same node, and replacing a pointer to an old self-looping node by the new node’s own self-reference—form a terminating and confluent presentation of strong-trace equivalence, with a linear bound on the number of rewrites. Equality of final figures only is a harder problem, because a later node can make an earlier difference invisible; we define the *causal support* that a final figure actually depends on, prove that the remaining data are freely variable, and prove two exact results in restricted settings (a two-row theorem relative to a fixed past, and a complete theorem for degree one and distance one). We say precisely what is not proved.

Finally we use the calculus on a demanding application: a direct even-palindrome recognizer needs a fresh record to own a fresh sequence whose last cell points back to that record, a same-step cycle that no ordinary allocation discipline can build. We prove that every bounded one-step update with an arbitrary finite graph of fresh records compiles into a scaffolding automaton, and explain why the cycle is harmless. Palindromes are a stress test of the calculus, not its subject. The paper is self-contained: it defines grammars, PEGs, scaffolding automata, bisimulation, and every rewrite it uses, and it makes no novelty claim for the established coalgebraic, automata-minimization, or term-graph ideas it builds on.

Contents

1	From a finite presentation to a history algebra	3
1.1	The machine and the question	3
1.2	In what sense this is an “algebra”	4

1.3	What is new here, and what is inherited	4
1.4	The viewpoint	4
1.5	How to read this paper	5
2	Background	5
2.1	Words and languages	5
2.2	Context-free grammars and parsing expression grammars	5
2.3	Scaffolding automata	5
2.4	From automata to histories	6
3	Histories, unfoldings, and observations	7
3.1	The finite row alphabet	7
3.2	Port-word behavior	8
3.3	Full abstraction	8
3.4	Why coalgebra, and what a figure is	9
4	Chronological order and navigation	10
5	Static guarded folding	11
5.1	Rewriting vocabulary	11
5.2	Guarded folds	11
6	The causal presentation calculus	13
6.1	Strong equivalence and semantic sewing	13
6.2	One-state extensions	13
6.3	The two primitive strong moves	14
6.4	SELF-eager prefixes	14
7	Final observation and replay support	16
7.1	A noncommuting boundary	17
8	Two calibrated sewing theorems	17
8.1	Unexposed acyclic two-row factorizations	17
8.2	The unary whole space	19
8.3	A path-level quotient	20
9	Feedback, receipts, and a repaired allocation model	21
9.1	The receipt knot	21

9.2	Bounded simultaneous record transducers	22
9.3	Staging a strict persistent sequence	23
9.4	The conditional application	24
10	Open boundaries	25
11	Assurance, provenance, and nonclaims	25
12	Conclusion	26
A	Move and preservation table	27
B	The central implication map	27

1 From a finite presentation to a history algebra

1.1 The machine and the question

Loff, Moreira, and Reis (LMR) introduced scaffolding automata as a machine model for parsing expression grammars [12]; Section 2 defines both. For now the picture suffices. At each input position the machine reads a fixed-radius neighborhood of the current top of a persistent graph, chooses a finite label, and appends one new node. Each new edge is missing, points to the fresh node itself, or follows a bounded path from the preceding top. LMR prove that a language has a PEG if and only if its reversal is decided by such a machine [12, Theorem 16].

The usual observation made of such a machine is acceptance: does the run end in an accepting state? This paper keeps more of the generated object visible. A run has a physical graph at every time; an observer standing at the top and following edges sees, at every time, a *behavioral figure*; and at the end there is a *final figure*. Those are different quotients of the same history, and the paper’s question is:

When two finite construction histories generate the same behavior, which local fold, unfold, routing, erasure, and feedback moves connect them?

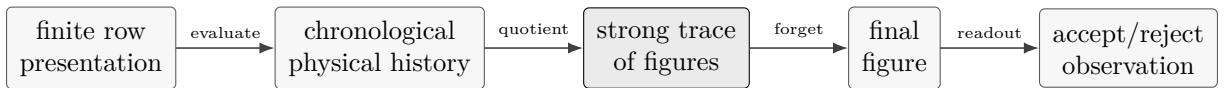


Figure 1: The observation ladder. A *row* is one construction step written down (Definition 3.1); a *history* is a sequence of rows; its *physical history* is the graph it builds; the *figure* at a time is the smallest graph an observer at the top cannot distinguish from the real one; the *strong trace* keeps every figure and the *final figure* only the last. Every arrow can identify presentations. No reverse implication is automatic.

The answer has two layers. For every fixed finite degree and observation distance, two graph-local rewrites give a terminating, confluent presentation of the histories that have the same behavioral

figure after every prefix (Theorem 6.8). Equality of only the final figure is harder because it forgets intermediate behavior; we prove what we can (Section 7–8) and state exactly what remains open (Section 10).

1.2 In what sense this is an “algebra”

We call the system of histories, behavioral equivalences, and rewrites an *algebra* in the informal sense in which one speaks of “the algebra of regular expressions”: a collection of objects, operations on them (appending a row, rewriting a row), and equations between the results. Where a formal algebraic structure is meant, it is named and proved: the monoid of rows under concatenation (Section 3), the transformation monoid of port navigation and its $\mathcal{R}$ -triviality (Corollary 4.2), and the rewriting system with its normal forms (Theorem 6.8). No claim is made that the calculus is an instance of any particular variety, category, or polygraph; whether it is, is an open comparison (Section 10).

1.3 What is new here, and what is inherited

The paper leans on three established theories and does not rename them. *Bisimulation* (two states are equivalent when every observation and every successor behavior matches) and the quotient of a system by it are standard objects of coalgebra [15, §2, §5–6]; the familiar instance is the minimal deterministic automaton obtained from the residuals (derivatives) of a language [3, 13]. Bottom-up merging of equivalent states and incremental minimization of acyclic deterministic automata are established [5, §2]. Cyclic term graphs, their unwinding into infinite trees, copying, hidden nodes, and confluence have a substantial theory [2]. Section 11 lists what is inherited from each.

The target here is the intersection of four restrictions that none of those theories imposes together:

1. one immutable layer is appended per input symbol;
2. every old address is read from the preceding top within one fixed radius;
3. a *strong* move must preserve the figure after every prefix; and
4. soundness must survive every common future continuation.

The resulting local rules live in the generated scaffold graph. They need not be bounded contiguous substring rules in the linear word of rows.

1.4 The viewpoint

Finitely Presented Recognition Dynamics, abbreviated FPRD, is the research viewpoint used in this work: a finite grammar or machine description is treated as a *presentation* that generates an internal recognition *dynamic*, and the accepted language is treated as one *observation* of that dynamic. FPRD Lab is the AI-assisted research project in which the present arguments were developed; nothing in this paper requires trust in that project, and Section 11 states plainly what was checked and by whom. In this paper the viewpoint has a concrete form, Figure 1: changing an observation may merge histories without changing the rows that generated them; changing a presentation may alter the physical history while preserving every behavioral figure. The algebra begins by keeping both phenomena explicit.

1.5 How to read this paper

Section 2 supplies the background: words and languages, context-free grammars and PEGs, LMR’s scaffolding automaton with a drawn example, and the step from automata to *arbitrary* construction histories, which is where this paper’s objects begin. Section 3 defines histories, unfoldings, figures, and the two equivalences, with one worked history that recurs throughout. Sections 4–6 develop the strong-trace calculus: the chain theorem, guarded folds, and the two rewrites with their normal form. Sections 7–8 treat final observation. Section 9 gives the feedback application. Sections 10–11 state open problems, what is inherited, and how the work was produced. Every proof states its strategy first; every example says what generalizes.

2 Background

2.1 Words and languages

An *alphabet* is a finite set of letters; a *word* is a finite sequence of letters, ϵ is the empty word, and Σ^* is the set of all words over Σ . A *language* is a set of words. The *reversal* of $w = c_1 \cdots c_n$ is $w^R = c_n \cdots c_1$. A *monoid* is a set with an associative operation and an identity; Σ^* under concatenation, with identity ϵ , is the *free monoid* on Σ : every word is a unique sequence of letters.

2.2 Context-free grammars and parsing expression grammars

A *context-free grammar* has terminal symbols, non-terminal symbols, a start symbol, and rewriting rules $A \rightarrow \varphi$ with one non-terminal on the left [4, pp. 120–122]; it *generates* the words obtainable from the start symbol by repeatedly replacing a non-terminal by a right-hand side. For instance $S \rightarrow 0S0 \mid 1S1 \mid \epsilon$ generates the even-length binary palindromes $\text{EPAL}_2 = \{ww^R\}$. The reading is existential: a word is in the language if *some* derivation produces it.

A *parsing expression grammar* (PEG) [7, §3] uses similar rule shapes with an operational meaning. Expressions are built from letters, non-terminals, ϵ , and **FAIL** by sequence e_1e_2 , *ordered choice* e_1/e_2 , and the predicates **!e** and **&e**; each non-terminal A has one rule $A \leftarrow R(A)$. Recognition $\text{Rec}_G(e, x)$ reads x from the left and either fails or succeeds consuming a prefix: a letter consumes itself; a sequence runs its parts in order; ordered choice runs e_1 and *commits to it if it succeeds*, trying e_2 only if e_1 fails; **!e** succeeds (consuming nothing) exactly when e fails [12, Definition 3, Algorithm 1]. The procedure is deterministic. A grammar is *total* if recognition never loops, and the language of a total grammar is $L(G) = \{x : \text{Rec}_G(S, x) = x\}$, the words on which the start symbol succeeds consuming the *whole* input (“complete match”); PEG is the class of such languages [12, Definition 4]. Order is semantic: with $S \leftarrow a/ab$, the input ab is not recognized, because the choice commits to a and leaves b unconsumed [7, p. 7]. PEGs recognize some non-context-free languages, such as $\{a^n b^n c^n\}$ [12, p. 3], while LMR conjectured that the context-free language EPAL_2 has no PEG [12, Conjecture 7]; the comparison of the two classes is not syntactic and goes through the machine model below.

2.3 Scaffolding automata

Definition 2.1 (scaffold [12, Definitions 10–11]). Let $d \geq 1$ and let Γ be a finite alphabet. A (d, Γ) -scaffold with top t has nodes $0, 1, \dots, t$; a label $L(v) \in \Gamma \cup \{\emptyset\}$ for each node, with $L(0) = \emptyset$

(the *base* is unlabelled); and for each node v and each *port* $i \in [d] = \{0, \dots, d-1\}$ a target $e_v(i) \in \{0, \dots, v\} \cup \{\emptyset\}$: every edge points to an older node or to the node itself, or is missing. A *port word* $p \in [d]^*$ is followed from a node by taking port p_1 , then port p_2 of the node reached, and so on; it is *valid* from v if no missing port is met. The k -*neighbourhood* $N_k(v)$ is the labelled tree of depth k obtained by unfolding ports from v ; it records labels and missing ports, not node numbers.

Definition 2.2 (scaffolding automaton [12, Definitions 12–15]). A scaffolding automaton $A = \langle \Sigma, d, \Gamma, k, Q, \delta, q_0, F \rangle$ has an input alphabet, a degree d , a working alphabet Γ , a *distance* $k \geq 0$, a finite control (Q, q_0, F) , and a transition function $\delta : Q \times \Sigma \times N_k(d, \Gamma) \rightarrow Q \times \Gamma \times ([d]^{\leq k} \cup \{\text{SELF}, \emptyset\})^d$. Reading a letter in state q with top t , it computes $(q', \gamma, p_0, \dots, p_{d-1}) = \delta(q, \sigma, N_k(t))$ and appends node $t+1$ with label γ and ports: the endpoint of p_i followed from t if p_i is a valid port word; missing if $p_i = \emptyset$ or invalid; the new node $t+1$ itself if $p_i = \text{SELF}$. It accepts a word if the state after the last letter is in F .

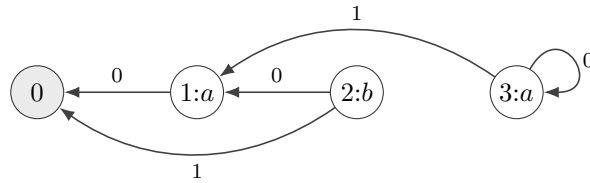


Figure 2: A scaffold of degree 2 after three steps, built by the rows $(a; \epsilon, \text{MISSING})$, $(b; \epsilon, 0)$, $(a; \text{SELF}, 0)$ in the notation of Definition 3.1: node 1 points to the base; node 2 points to node 1 (the old top, descriptor ϵ) and, via port word 0 from node 1, to the base; node 3 points to itself and, via port word 0 from node 2, to node 1. An observer at node 3 who follows ports sees labels a , then a again forever along port 0; it cannot see that the node is the same each time.

Theorem 2.3 (Loff–Moreira–Reis [12, Theorem 16, p. 21]). *A language L is in PEG if and only if L^R is decided by some scaffolding automaton.*

The reversal is not an artifact: the automaton commits letter by letter from the left, and LMR’s grammar re-enacts its run from the other end. This paper uses the theorem only in the labelled example of Section 9; none of its own results depends on it.

2.4 From automata to histories

Everything from Section 3 to Section 8 concerns the *graphs* that scaffolding automata can build, not any particular automaton. A run of an automaton on an input word writes down, at each step, one *row*: the label chosen and the d descriptors chosen. Conversely, any sequence of rows is a legal construction (an invalid port word simply yields a missing edge), whether or not some finite control would produce it. We therefore study all finite row sequences, called *histories*, and ask about their behavior as seen by an observer at the top. The finite control, the input word, the distance bound on what the control may *read*, and acceptance play no role until Section 9. In particular the rewrites of Sections 6–8 are mathematical operations on histories; we do not claim that an automaton could perform them while running (Section 6.3 says this once more where it matters).

3 Histories, unfoldings, and observations

3.1 The finite row alphabet

Fix a degree $d \geq 1$, a distance $k \geq 0$, and a finite working alphabet Γ . Write $[d] = \{0, \dots, d-1\}$ for the port alphabet and

$$\Delta_{d,k} = \{\text{MISSING}, \text{SELF}\} \cup [d]^{\leq k}$$

for the set of *descriptors*.

Definition 3.1 (row, history). A *row* is an element of $\text{Row}_{d,k,\Gamma} = \Gamma \times \Delta_{d,k}^d$: one label and one descriptor per port. A *history* is a finite sequence of rows. The empty word $\epsilon \in [d]^{\leq k}$ is a valid descriptor: it names the preceding top.

Definition 3.2 (Evaluation). A history

$$H = (a_t; \delta_{t,0}, \dots, \delta_{t,d-1})_{t=1}^n$$

is evaluated from an unlabelled edgeless base node 0. At time t , append node t with label a_t . Its port i is missing if $\delta_{t,i} = \text{MISSING}$, points to t if $\delta_{t,i} = \text{SELF}$, and otherwise points to the endpoint obtained by following $\delta_{t,i}$ from node $t-1$ in the old scaffold. An invalid path gives a missing edge. We write S_j for the scaffold after the first j rows.

The set of histories is the free monoid $\text{Row}_{d,k,\Gamma}^*$ as syntax. Appending a row is a total operation on evaluated histories because invalid paths have the defined missing outcome.

Example 3.3 (a worked history). Take $d = 1$, $k = 1$, $\Gamma = \{0, 1\}$, so every descriptor is one of MISSING, SELF, ϵ , 0 (the only port word of length 1). The history

$$E = (1; \text{SELF}) (1; \epsilon) (0; 0)$$

evaluates as follows (Figure 3). Row 1 creates node 1 with label 1 and a self-loop. Row 2 creates node 2 with label 1 pointing to the old top, node 1. Row 3 creates node 3 with label 0 whose port follows port word 0 from node 2: node 2's port leads to node 1, so node 3 points to node 1. Node 2 is not reachable from node 3, yet its port cell was *consulted* when node 3's edge was evaluated; this distinction drives Section 7. What generalizes: every history is evaluated this way, one row at a time, reading only the previous top's surroundings. What is incidental: the labels and the length.

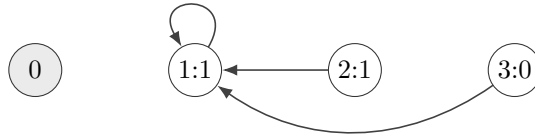


Figure 3: The worked history $E = (1; \text{SELF})(1; \epsilon)(0; 0)$ evaluated. From node 3 the observer sees label 0, then 1, then 1 forever.

3.2 Port-word behavior

For a rooted evaluated scaffold (S, t) , following a port word from t either reaches a node or runs into a missing port. Adjoin a value $\perp_e$ for the latter. The *behavioral unfolding* of (S, t) is the function

$$\mathcal{U}_{S,t} : [d]^* \longrightarrow \Gamma \cup \{\text{UNLABELLED}, \perp_e\}$$

that sends a valid word to the label of its endpoint (with UNLABELLED for the base) and an invalid word to $\perp_e$. The unlabelled base is therefore observable and is not a missing edge. The unfolding is everything an observer at t can learn by following ports and reading labels; it is what LMR's transition function sees, restricted to words of length at most k .

Two rooted scaffolds are *behaviorally equivalent*, written $(S, t) \simeq (T, u)$, when their unfoldings agree. For a history H of length n , let

$$\text{Fig}_j(H) = M(\mathcal{U}_{S_j,j})$$

denote the canonical minimal figure of its length- j prefix, defined in Section 3.4. The *strong trace* and *final observation* are

$$\text{Tr}(H) = (\text{Fig}_1(H), \dots, \text{Fig}_n(H)), \quad \pi(H) = \text{Fig}_n(H).$$

Definition 3.4 (Two congruences). Equal-length histories are *strongly equivalent*, $H \equiv_s H'$, when $\text{Tr}(H) = \text{Tr}(H')$. They are *finally equivalent*, $H \equiv_f H'$, when $\pi(H) = \pi(H')$.

Strong equivalence implies final equivalence. The converse fails because a later row may make an earlier difference unreachable or behaviorally irrelevant.

Example 3.5. With $d = k = 1$, the histories

$$(0; \text{MISSING}) (0; \text{MISSING}) (0; \text{MISSING}) \quad \text{and} \quad (1; 0) (1; 0) (0; 0)$$

have the same final figure—a single node labelled 0 with a missing port—because in the second history every descriptor 0 follows a port that is itself missing. Their strong traces differ already at time 1 (labels 0 versus 1). Many histories share a final figure while their strong traces differ; the displayed pair is the smallest fact used below.

3.3 Full abstraction

The first theorem says that the unfolding is the right notion of behavior: it is determined by finite observations, it is exactly bisimilarity, it is preserved by every common future, and any difference is eventually visible.

Theorem 3.6 (Future-context full abstraction). *For rooted degree- d scaffolds over one working alphabet:*

1. *equal unfoldings, equality of every finite rooted neighborhood, and deterministic labelled bisimilarity are equivalent;*
2. *unfolding equivalence is preserved by every common scaffold row and hence by every common continuation; and*
3. *unequal unfoldings are distinguished after finitely many steps by a distance-two unary scaffold continuation.*

Here a *bisimulation* between two rooted scaffolds is a relation on nodes that relates the roots and, whenever it relates x to y , requires that x and y have the same label and that for every port either both are missing or the targets are again related; two roots are *bisimilar* if some bisimulation relates them. “Unary” means that the continuation uses only one port of each new node; “distance two” means its descriptors have length at most 2 (whatever the distance of the original scaffolds). “Distinguished” means: there is a single sequence of rows which, appended to either scaffold, produces tops whose depth-2 neighbourhoods differ, so that a finite control reading those neighbourhoods can accept one and reject the other.

Proof. Strategy. (1) is a restatement; (2) is an induction on the query word; (3) constructs a “cursor” that walks along a shortest distinguishing port word, one letter per row.

The depth- m neighborhood is exactly the restriction of the unfolding to port words of length at most m , which proves the first equivalence. The relation “same unfolding” is a bisimulation, and conversely a bisimulation forces equal unfoldings by induction on the word.

Suppose the old unfoldings agree and append the same row. The observed depth- k neighborhoods agree. For a query word starting at fresh port i , a missing descriptor fails on both sides; a **SELF** descriptor returns to the respective fresh root; and an old descriptor p reduces the comparison to $\mathcal{U}_{S,i}(pw) = \mathcal{U}_{T,u}(pw)$. The fresh labels agree. Induction on the remaining query length proves equality of the new unfoldings. Iterating proves continuation stability.

For the converse, choose a shortest port word $p_1 \cdots p_m$ on which the unfoldings differ. If $m = 0$, the current labels already differ. Otherwise a unary tester stores a cursor in one port. Its first row points along p_1 ; at the next step the distance-two path through the cursor and p_2 advances it; and so on. All proper prefixes behave identically by minimality. The last neighborhood exposes the first label or missing-edge difference, which finite control records in acceptance. $\square$

Corollary 3.7. *Both $\equiv_s$ and $\equiv_f$ are stable under a common right suffix of rows.*

3.4 Why coalgebra, and what a figure is

The reader who knows finite automata has met the idea of this subsection already. A language L has, for each word p , a *residual* (or derivative) $p^{-1}L = \{w : pw \in L\}$; the language is regular exactly when it has finitely many residuals, and those residuals are the states of its minimal deterministic automaton [3, Def. 3.1, Thm. 4.3], [13]. Two states of any automaton are equivalent when they accept the same words, and merging equivalent states gives the minimal automaton. Coalgebra is the general form of this idea: a *system* is described by what can be observed at a state (here, a label) and which states are reached next (here, the port targets); two states are *bisimilar* when every observation and every successor behavior matches; the quotient by bisimilarity is the smallest system with the same behavior [15, §2, §5–6]. We use only this much of the theory, in the deterministic labelled-graph instance, and we prove the facts we need directly.

For a valid port word p , write $\mathcal{U}_p(w) = \mathcal{U}(pw)$ for the *residual unfolding* at p : what an observer sees after first walking along p . Let $\mathcal{R}(\mathcal{U})$ be the finite set of distinct valid residual unfoldings. A map f from the nodes of one rooted labelled port graph to those of another is a *functional bisimulation* if it sends root to root, preserves labels, and for every node x and port i either both x and $f(x)$ lack port i or $f(x.i) = f(x).i$. Two rooted labelled port graphs are *isomorphic* if there is a bijective functional bisimulation between them.

Theorem 3.8 (Minimal residual figure). *Every finite rooted scaffold has a canonical finite behavioral figure $M(\mathcal{U})$ whose states are $\mathcal{R}(\mathcal{U})$, whose root is $\mathcal{U}_\epsilon$, and whose port i sends $\mathcal{U}_p$ to $\mathcal{U}_{pi}$ when defined. The physical reachable graph maps onto $M(\mathcal{U})$ by a surjective functional bisimulation. Two rooted scaffolds are behaviorally equivalent if and only if their minimal figures are isomorphic.*

Proof. Every residual is represented by a physical reachable node, so there are at most as many residuals as nodes. Residualization commutes with every port ($\mathcal{U}_p = \mathcal{U}_q$ implies $\mathcal{U}_{pi} = \mathcal{U}_{qi}$), which makes labels and successors well defined and makes the node-to-residual map a functional bisimulation. Distinct residual functions have distinct unfoldings and cannot be bisimilar. The construction depends only on $\mathcal{U}$, proving minimality and uniqueness up to rooted labelled port-graph isomorphism. $\square$

Example 3.9. For the worked history E of Example 3.3, the residuals from node 3 are: $\mathcal{U}_\epsilon$ (label 0, then 1 forever) and $\mathcal{U}_0 = \mathcal{U}_{00} = \dots$ (label 1 forever). The figure $\text{Fig}_3(E)$ has two states, a 0-labelled root pointing to a 1-labelled state with a self-loop. The physical node 2 is not reachable and plays no part.

Remark 3.10. Physical aliasing is finer than behavior. Two ports may point to one node in one history and to two bisimilar nodes in another; their minimal figures still agree. Alias identity becomes observable only when a presentation explicitly stores it in finite labels.

4 Chronological order and navigation

Write $x \preceq y$ when x is reachable from y by following ports, including $x = y$; since edges point to older nodes or to the node itself, $x \preceq y$ implies that x was created no later than y , so later nodes sit *above* earlier ones. A finite port graph is *weakly acyclic* when its only directed cycles are self-loops. Its rooted component is a *reachability chain* when $\preceq$ is a total order there.

Theorem 4.1 (Generated-chain theorem). *Let S be a legally generated chronological scaffold with current top t .*

1. *Its physical graph and its minimal figure are weakly acyclic.*
2. *The nodes reachable from t , and the states of the minimal figure, are totally ordered by reachability.*
3. *Conversely, every finite behaviorally reduced labelled rooted reachability-chain port graph, with an optional least unlabelled edgeless state, has an exact legal scaffold presentation at some finite distance.*

Part (2) is worth a pause: it says that, from the top, *any two* reachable nodes are comparable, one reachable from the other. The reason is that a new node can only point into the previous top's reachable set, which is a chain by induction.

Proof. A non-SELF edge of node t reaches an older node, so creation time never increases along a port path and can stay fixed only on a self-loop. If the behavioral quotient had a cycle through two classes, repeating a word around the cycle from a physical representative would give an infinite path of nonincreasing times. It is eventually constant, so every later edge is one physical self-loop and all classes on the alleged cycle coincide.

Let R_j be the component reachable from the top after step j . Every old target of the new top is already in R_{j-1} , so $R_j \setminus \{j\} \subseteq R_{j-1}$. Induction makes R_j a chain, and quotienting preserves total comparability.

Conversely enumerate the desired states $q_1 \prec \dots \prec q_m$. When q_j is appended, each strict successor is some q_i at or below q_{j-1} and is therefore reached from q_{j-1} by a port word. Use that word, **SELF** for self-loops, and missing for absent edges. A shortest simple word has length at most $j - 2$, so one distance at most $m - 1$ realizes all rows. $\square$

A transformation f of a finite totally ordered set is *regressive* if $xf \leq x$ for every x (we write the action on the right). In a monoid M , Green's relation $\mathcal{R}$ is defined by $s\mathcal{R}t$ iff $sM^1 = tM^1$, i.e. $s = tu$ and $t = sv$ for some $u, v \in M^1$; M is $\mathcal{R}$ -trivial when every $\mathcal{R}$ -class is a singleton [14, Ch. V §1, p. 99; Ch. X §1.3].

Corollary 4.2 (Regressive navigation). *After adjoining a least missing sink to a generated minimal figure and numbering its chain, every port acts regressively. Its port-word transition monoid is a submonoid of the full regressive transformation monoid and is $\mathcal{R}$ -trivial.*

Proof. Each successor is at or below its source. If transformations f, g are $\mathcal{R}$ -related, write $f = gu$ and $g = fv$. Pointwise regressiveness gives $xf \leq xg$ and $xg \leq xf$ for every state x , hence $f = g$. $\square$

Boundary. This is the algebra of historical port navigation inside one figure. It says nothing about the streamed input language, and nothing about the finite control's routing of roots, which is a separate finite interface.

5 Static guarded folding

5.1 Rewriting vocabulary

A *rewriting system* on a set is a binary relation $\rightarrow$ (“one step”). An object is *irreducible*, or a *normal form*, if no step applies to it. The system is *terminating* if there is no infinite sequence of steps, and *confluent* if whenever an object rewrites (in zero or more steps) to two objects, those two rewrite to a common object. A terminating confluent system gives every object exactly one normal form, and two objects are related by the equivalence generated by $\rightarrow$ exactly when they have the same normal form; this is the *Church–Rosser* property. We use these words in exactly this sense.

5.2 Guarded folds

Let G be a finite deterministic labelled partial port graph and let E be a bisimulation equivalence already imposed on it (an equivalence relation that is a bisimulation). Work in the quotient G/E , whose states are the classes and whose ports are well defined because E is a bisimulation.

Definition 5.1 (Elementary guarded fold). Two distinct quotient states x, y admit a guarded fold when their labels agree and, at every port, the two successors are both missing, are the same quotient state, or lie in $\{x, y\}$. Equivalently, identifying x and y makes their one-step successor profiles equal.

The last case is the guarded recursive clause. It permits self-reference at either member of the pair but no larger unresolved cycle.

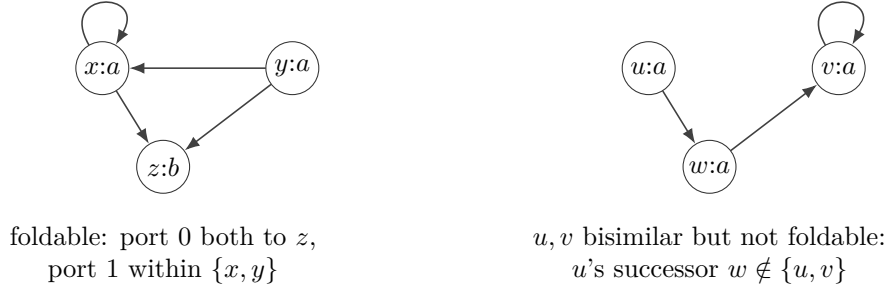


Figure 4: Left: a guarded fold of x and y (degree 2; port 0 drawn down, port 1 drawn across). Right (degree 1): all three of u, w, v are bisimilar, but the pair (u, v) fails the guard; the fold must proceed through (w, v) first, then $(u, [wv])$.

Theorem 5.2 (Guarded folds generate bisimilarity). *On every finite weakly acyclic deterministic labelled partial port graph, the equivalence generated by elementary guarded folds is exactly the greatest bisimulation equivalence.*

Proof. Strategy. Soundness is immediate. For completeness, find, in any sound quotient that is not yet fully reduced, a bisimilar pair that satisfies the guard: take a minimal unreduced class and look at its sinks.

Each fold is sound because corresponding successors become equal in the new quotient.

Two facts keep the induction well posed. First, the greatest bisimulation quotient of a finite weakly acyclic graph is weakly acyclic: the argument of Theorem 4.1(1) applies verbatim with any topological numbering in place of creation time. Second, a guarded fold preserves weak acyclicity: a new cycle through the folded class $[xy]$ would pass through a state $z \notin \{x, y\}$ that is a successor of x or y and reaches x or y ; since x and y have identical non-pair successors, z is a successor of both, so the old graph had a cycle $z \rightarrow \dots \rightarrow y \rightarrow z$ of length at least two, contradicting weak acyclicity. Hence every sound quotient reached by folds is weakly acyclic, and “minimal in strict reachability” below is meaningful.

For completeness, suppose a sound quotient still has a nonsingleton greatest bisimulation class. Choose such a class C minimal in strict reachability of the behavioral quotient. Every strict successor class below C is already a singleton.

The edges between distinct current states inside C form a finite DAG. If it has two sinks x, y , an internal successor of either sink can only be its own self-loop, while corresponding external successors are the same singleton; therefore x, y are guarded-foldable. If it has one sink x , choose a minimal state y of $C \setminus \{x\}$. Every strict internal successor of y is x , and every internal successor of x is its self-loop. External successors are again equal, so x, y are guarded-foldable.

Another sound fold is therefore available until the greatest bisimulation quotient is reached. $\square$

Corollary 5.3 (Static normal form). *Orient each guarded fold toward its quotient. The system terminates and is confluent up to rooted labelled port-graph isomorphism. Its normal form is the minimal residual figure.*

Proof. Every fold reduces the state count. An irreducible graph is behaviorally reduced by Theorem 5.2; every reduction preserves the unfolding; and Theorem 3.8 gives one reduced figure up to

isomorphism. □

This theorem contains the usual bottom-up acyclic merge [5, §2] as a special case. The extra local case is a self-recursive pair. Static normalization alone, however, may remove a physical state and therefore does not yet define a time-respecting rewrite of construction histories: the next section asks which rewrites of the *rows* realize it.

6 The causal presentation calculus

6.1 Strong equivalence and semantic sewing

A *semantic step replacement* changes one row over a fixed preceding prefix when the two possible fresh roots are behaviorally equivalent.

Proposition 6.1 (Semantic causal sewing). *Two strongly equivalent length- n histories are connected by at most n semantic step replacements.*

Proof. Align the rows from left to right. Once the prefixes through $j - 1$ are literally equal, replaying the old j th row over that prefix preserves its behavior by Theorem 3.6; the target row has the same j th figure by strong equivalence. Replace it and continue. □

So there is no *semantic* obstruction: strongly equivalent histories are always connected by row replacements. The real problem is locality: the side condition “the two fresh roots are behaviorally equivalent” is a global statement about unfoldings, and we want a fixed family of moves each of which is justified by looking at a bounded piece of the scaffold.

6.2 One-state extensions

Let Q be a behaviorally reduced old live figure (the reachable part of the old scaffold, assumed to have no two bisimilar nodes). It is convenient to describe a new row not by descriptors but by what its ports *reach*: a new row has the guarded one-variable form

$$R = (a; t_0, \dots, t_{d-1}), \quad t_i \in Q \cup \{X, \perp_e\},$$

where X denotes the fresh root itself. We say the fresh root *presents* a behavior, namely its unfolding; it presents an old residual $q \in Q$ if its unfolding equals that of q .

Lemma 6.2 (One-state extension dichotomy). *Suppose two such fresh roots have the same behavior.*

1. *If the behavior is not already a state of Q , their missing and SELF positions agree, and their old target agrees at every other port.*
2. *If the behavior is an old state $q \in Q$, each row is a guarded unroll of q : it copies every missing and strict-successor port, while a self-loop of q may target either q or the fresh variable X .*

Proof. In the first case, a fresh SELF on one side and an old target q on the other would make the common fresh behavior equal to q , a contradiction. Thus the SELF masks agree. Corresponding old targets are bisimilar and are therefore the same state of reduced Q ; missing positions also agree.

In the second case, compare the fresh root with q . A non-SELF target must be the unique reduced successor of q . A fresh SELF is possible exactly where that successor is q itself. These conditions are also sufficient: identity on Q together with the fresh-root/ q pair is a bisimulation. $\square$

6.3 The two primitive strong moves

Fix the total order

$$\text{MISSING} < \text{SELF} < p \quad (p \in [d]^{\leq k})$$

on descriptors, where port words are ordered by length and then lexicographically (so ϵ is the least port word), and order rows with the same label by comparing descriptor tuples position by position.

Definition 6.3 (Endpoint detour D). Replace one old-top word by another bounded word with the same physical endpoint. All invalid words share the missing endpoint. Orient the move toward the least name of that endpoint.

Definition 6.4 (Literal guarded unroll U). Let a fresh row point, at some port, to an old physical node q that has a self-loop at that port, and suppose the fresh root presents the residual of q : its label equals q 's, and at every port its target is missing exactly when q 's is, is the physical successor of q where that successor is not q , and is either q or SELF where q has a self-loop. If at least one such port points to the old q , replace all of them by SELF and choose least endpoint names at the other ports. Orient in that direction.

The simultaneous recursive part of U is *absorption* A ; its remaining changes are endpoint detours. The guard is physical and bounded: it names q within the observed scaffold and inspects its fixed port tuple.

Example 6.5 (the two moves on the worked history). In $E = (1; \text{SELF})(1; \epsilon)(0; 0)$, row 2 points to node 1, which has a self-loop and the same label, and node 2's unfolding equals node 1's; so U applies and rewrites row 2 to $(1; \text{SELF})$. After that rewrite, row 3's descriptor 0 is followed from the new node 2, whose port is now a self-loop, so the endpoint is node 2; the least name for node 2 from the old top is ϵ , and D rewrites row 3 to $(0; \epsilon)$. The result $(1; \text{SELF})(1; \text{SELF})(0; \epsilon)$ has the same figure as E at every time. What generalizes: an unroll can change the physical endpoints of later rows, and detours then rename them.

These are mathematical graph-local rewrites with explicit endpoint-equality certificates. The calculus does not assert that an LMR automaton can discover or execute a normalizing move from finite control, nor that the move occupies a bounded contiguous interval in the chronological row word.

6.4 SELF-eager prefixes

Call a canonical history *SELF-eager* when it uses SELF at every available self-loop coordinate and least names elsewhere; by the order fixed above, this is the same as being the least row at every step.

Lemma 6.6 (Canonical history exists). *Every realizable finite strong trace has a unique lexicographically least history. It is obtained from left to right by appending the least row that realizes the next figure over the already chosen prefix.*

Proof. Assume the canonical prefix through time $j - 1$ has been chosen and take any original realization of the full trace. Its length- $(j - 1)$ prefix has the same rooted unfolding as the canonical prefix. Replay the original j th row over the canonical prefix. Theorem 3.6 says that the new root has the required j th figure, so the finite set of realizing rows is nonempty and has a unique least element. Induction constructs the history; leastness at the first differing row proves uniqueness. $\square$

Lemma 6.7 (Reduced live prefixes). *Every root-reachable component at every prefix of a SELF-eager history is behaviorally reduced.*

Proof. Induct on prefix length. If the fresh behavior is new, it adds one new residual above a reduced old component. If it repeats an old residual q , Lemma 6.2 says that every strict successor lies below q and that only a self-loop coordinate can choose between old q and fresh SELF. SELF-eagerness chooses the fresh root. It therefore reaches itself and strict descendants but not the old physical q . The fresh root replaces q as the unique live representative. $\square$

Theorem 6.8 (Convergent strong-trace presentation). *For fixed finite d, k, Γ , oriented endpoint detours and literal guarded unrolls form a terminating confluent graph-local presentation of strong equivalence. Two histories have the same strong trace if and only if they reduce to the same SELF-eager history. A length- n history normalizes in at most dn row moves when U counts as one whole-row move.*

Proof. Strategy. Soundness of each move; termination by a lexicographic measure; then show that an irreducible history is the canonical one of Lemma 6.6, by induction from left to right.

Both moves are sound. A detour preserves a physical endpoint. For an unroll, identity on the old prefix together with the fresh-root/ q pair is a bisimulation. Theorem 3.6 carries either replacement through the unchanged suffix.

Lexicographically order the chronological descriptor word. A detour strictly decreases one path name. An unroll replaces at least one old path by smaller SELF and never increases an earlier descriptor. Thus reduction terminates.

Induct left to right through an irreducible history. Its old prefix is already SELF-eager and reduced by Lemma 6.7. Compare the fresh row with the least row supplied by Lemma 6.6 for the same next figure. If the residual is new, Lemma 6.2 gives the same missing/SELF mask and the same old residual targets. Reducedness turns those targets into the same physical nodes; any remaining difference is a detour redex. If the residual repeats old q , a path back to q at a self-loop is an unroll redex, and a nonleast path to any fixed strict successor is a detour redex. Therefore an irreducible row is exactly the SELF-eager least row.

Each strong-trace class has one irreducible history, so termination gives confluence and completeness. Normalizing from left to right, a new residual needs at most d detours. A repeated residual needs at most d detours or one whole-row unroll; because $d \geq 1$, the total is at most dn . $\square$

Remark 6.9 (What “coherence” means here). Theorem 6.8 is object-level Church–Rosser coherence: equal strong traces have one normal form, so any two reduction sequences from one history end at the same place. It does *not* say that any two such sequences are themselves related by a finite list of elementary identities between sequences (“two-dimensional cells”). We reserve the phrase *path-level coherence* for results that actually present those relations between sequences of moves.

7 Final observation and replay support

Final equivalence forgets intermediate figures. A row that is unreachable in the final rooted graph can nevertheless have been *queried* while a later descriptor was evaluated: in the worked history E , node 2 is not reachable from node 3, but node 3’s descriptor 0 was evaluated by reading node 2’s port. Ordinary graph garbage (unreachable nodes) is therefore too small a notion of “irrelevant data”; the right notion follows the evaluation.

We call the descriptor $\delta_{t,i}$ written in row t a *port cell* and the label a_t a *label cell*; these are the syntactic coordinates of a history. Evaluating a port cell that holds a port word p from the old top *queries* the port cells of the nodes it passes through, including the one at which an invalid path fails.

Definition 7.1 (Causal endpoint support). Let R_H be the created nodes reachable from the final top. Support every label cell and every port cell of a node in R_H . Whenever a supported port cell was created by following an old-top path, support every older port cell actually queried along that evaluation, including the cell at which an invalid path fails. Close backward to the least such set C_H .

Example 7.2. For E : $R_E = \{3, 1\}$; the supported label cells are those of rows 3 and 1; the supported port cells are those of rows 3 and 1 and, because row 3’s descriptor 0 queried node 2’s port, also the port cell of row 2. Row 2’s label is unsupported.

Theorem 7.3 (Replay invariance). *Suppose an equal-length history H' agrees with H on the labels of R_H and on every descriptor cell in C_H . Then the final rooted concrete scaffolds of H and H' are identical. Moreover $R_{H'} = R_H$ and $C_{H'} = C_H$.*

Proof. Proceed chronologically through supported descriptor cells. Every dependency of a supported cell is an earlier supported cell, so induction makes its evaluation traverse the same physical endpoints on both sides. The outgoing edges and labels of all final-reachable nodes are therefore identical, giving the same rooted concrete scaffold and the same reachable set. Recomputing the least backward closure follows the same descriptor traces, hence gives the same support. $\square$

Definition 7.4 (Garbage fibre). $\mathcal{G}(H)$ is the set of histories agreeing with H on the data named in Theorem 7.3. A G -move changes any collection of the remaining data.

Theorem 7.5 (Causal-garbage cube). *Every garbage fibre is the Cartesian product of the finite value sets of its unsupported label and descriptor coordinates. Simultaneous G factors into single-coordinate changes. Fill a triangle for three values of one coordinate and a square for changes of two distinct coordinates. The resulting path two-complex is simply connected.*

The last sentence is a statement about *sequences* of garbage moves, of the kind Remark 6.9 calls path-level: form the graph whose vertices are the histories of the fibre and whose edges are single-coordinate changes, glue in a two-dimensional face for every triangle and every square named in the theorem, and the claim is that every closed loop of moves can be contracted across those faces—any two ways of making the same garbage change are related by the displayed elementary identities.

Proof. Support stability in Theorem 7.3 makes every unsupported coordinate independent. Order the coordinates. Commuting squares move all changes of the first coordinate to the front, then the second, and so on. Same-coordinate triangles compose each block to the direct change between its endpoint values. Every path reduces to one coordinate-ordered path; a loop reduces to the empty path. $\square$

7.1 A noncommuting boundary

Let S denote strong normalization (Theorem 6.8) and let J (“junking”) choose least values outside causal support. Both preserve the final figure, but in general

$$S \circ J \neq J \circ S.$$

An absorption can move a repeated residual to a newer physical carrier. Junking first may erase the label needed to perform that absorption; absorbing first makes the older carrier garbage. Thus the two normalizers can reach distinct common fixed points with one final figure already in the unary distance-one three-row fragment.

Concretely, take the worked history $E = (1; \text{SELF})(1; \epsilon)(0; 0)$.

	row 1	row 2	row 3	comment
E	$(1; \text{SELF})$	$(1; \epsilon)$	$(0; 0)$	node 3 $\rightarrow$ 1; node 2 unreachable but queried
$S(E)$	$(1; \text{SELF})$	$(1; \text{SELF})$	$(0; \epsilon)$	row 2 absorbed; row 3 renamed to the new carrier
$J(S(E)) = E_{SJ}$	$(0; \text{MISSING})$	$(1; \text{SELF})$	$(0; \epsilon)$	row 1 now entirely garbage
$J(E)$	$(1; \text{SELF})$	$(0; \epsilon)$	$(0; 0)$	row 2’s label unsupported, set to 0; its port cell kept
$S(J(E)) = E_{JS}$	$(1; \text{SELF})$	$(0; \epsilon)$	$(0; 0)$	already strong-normal: node 2 no longer presents node 1

Both final roots are labelled 0 and point to a 1-labelled self-loop, but E_{SJ} and E_{JS} are distinct and irreducible under both normalizers.

Boundary. Object-level confluence for strong traces plus cubical coherence inside each garbage fibre does not present their join. The missing operation changes which chronological node carries a final residual and therefore changes the support boundary itself.

8 Two calibrated sewing theorems

This section does not assert a complete final-equivalence calculus at arbitrary degree and distance. It proves one local two-row theorem and one global unary theorem. Their different hypotheses show where the remaining problem lies. “Sewing” is used literally: given two histories with the same final figure, we want a sequence of local moves that stitches one into the other while every move preserves the final figure.

8.1 Unexposed acyclic two-row factorizations

Fix a behaviorally reduced old prefix with top q , append one node r (the *router*), then one more node z (the *consumer*). The router’s ports are its *slots*. A consumer port whose descriptor is a nonempty port word is a *path demand*; its first letter i is its *selector*, naming router slot i . Assume no consumer port points to r by the empty descriptor. Whenever a consumer word begins with router slot i , assume that slot contains an old-prefix path α_i , not **SELF** or missing. Thus a consumer word $i\beta$ reaches

$$q_{\alpha_i\beta},$$

the node reached from q by the composite word $\alpha_i\beta$. Call such a block *unexposed* (the consumer does not point at the router itself) and *acyclic* (the used slots point into the past). Because every

consumer port points into the old prefix, the router r is unreachable from z and hence from every later top: no later row can see or query r . Consequently r 's label and its *unused* slots are outside causal support, while a slot used by some consumer demand is queried and supported.

Three final-only moves are useful.

1. A *routing crossing* injectively reassigns used router slots and changes each first selector to follow the reassignment.
2. A *cut slide* moves one letter between α_i and β when both bounded descriptors remain legal.
3. A *dedicated relation exchange* changes (α, β) to (α', β') in a slot selected by exactly one consumer port, under the guard $q_{\alpha\beta} = q_{\alpha'\beta'}$.

The first two refactor a fixed composite word. The third uses a relation in the pointed old navigation action and is the move that crosses between different words with one old endpoint.

Theorem 8.1 (Unexposed two-row sewing). *Fix $d, k \geq 1$, and a behaviorally reduced old prefix. Consider two unexposed acyclic two-row extensions with the same consumer label, the same valid nonmissing path-demanding consumer coordinates, and literal agreement as missing or SELF on every other consumer coordinate. If corresponding path demands have the same final old residual behavior, the blocks are connected by endpoint detours, causal-garbage moves, continuation-aware routing crossings, and dedicated relation exchanges. Every common suffix is preserved.*

Here “the same final old residual behavior” means that the nodes reached in the old prefix by corresponding demands have equal unfoldings, and “continuation-aware” records that a routing crossing rewrites the consumer’s selectors together with the router’s slots; in the present setting no later row can reach the router, so the common suffix is literally unchanged.

Proof. Strategy. Make each demand own a private router slot, then change the slots one at a time under the endpoint guard, then undo the private splitting.

Reducedness makes equal old residual behaviors the same physical old state. Canonicalize every missing consumer coordinate to literal missing by a detour.

If several path demands use one first selector, fewer router slots than demands are currently used. Copy that router descriptor into an unsupported slot by G (legal, since unused slots are outside support) and retarget one consumer word to the copy by a detour; its complete physical endpoint is unchanged. Iteration makes the used selectors injective. Routing crossings put both blocks into one fixed injective slot placement.

Now each demand has a dedicated slot. Its current and target factorizations reach the same old physical state. One dedicated relation exchange changes that pair without affecting another demand. Process all demands, align the unused cells and router label by G , and reverse the target-side selector-splitting steps. Every move preserves the consumer endpoint vector, so the consumer-rooted concrete graph is identical and every common suffix replays identically. $\square$

Remark 8.2. Dedicated relation exchange has a bounded graph-local guard for fixed d, k , but it is not a rowwise endpoint detour. It explicitly cites equality between two composite old paths across the router/consumer cut. Whether it is derivable by refactoring older rows is open. Theorem 8.1 is therefore a presentation *relative to the pointed endpoint congruence of the old prefix*; it is not a finite relation list independent of that old action.

8.2 The unary whole space

Now set $d = k = 1$. Every descriptor is one of

$$\text{MISSING}, \quad \text{SELF}, \quad \epsilon, \quad 0.$$

Each state of the final figure is represented by at least one physical node; a node whose residual is a state of the final figure and which is reachable from the final top is a *carrier* of that state. After absorbing repeated live presentations of a terminal recursive residual (a self-looping final state carried by several consecutive nodes each pointing to the previous one), the final live figure is a reduced chain, and each of its states has exactly one carrier. Rows between two consecutive carriers that are not themselves carriers are *relay rows*: they are not final-reachable, but their port cells pass the lower carrier upward to the next one.

Lemma 8.3 (Unique relay). *If consecutive states $u > v$ of that final reduced chain are carried at nonadjacent construction times, the supported descriptor segment between them is, up to endpoint detours at a recursive bottom,*

$$\epsilon 0 0 \cdots 0.$$

Proof. The first row after the lower carrier selects that carrier by ϵ . Every later supported relay row, including the upper carrier's edge to the lower residual, must use 0 to pass the same endpoint upward; missing or SELF would terminate the relay, while a later ϵ would make the immediately preceding row final-reachable and insert another live residual between u and v . If the endpoint is a self-loop residual, ϵ and 0 can name the same old endpoint and one detour chooses the displayed form. $\square$

A *temporal carrier slide* exchanges the adjacent descriptor patterns

$$(\epsilon, 0) \longleftrightarrow (0, \epsilon)$$

at rows $(t, t+1)$, provided rows $t-1$ and t carry the same label. The endpoint calculation is literal. On the left, row t points to row $t-1$ and row $t+1$ reaches row $t-1$ through row t 's port, so row $t-1$ carries the residual and row t is a relay. On the right, row t points to row $t-1$'s target and row $t+1$ points to row t ; since rows $t-1$ and t have the same label and now the same target, row t carries the residual that row $t-1$ carried, and row $t-1$ has become garbage. The final figure is unchanged although the intermediate figure and the support carrier have changed.

Theorem 8.4 (Unary final presentation). *Fix a length n , a nonempty finite working alphabet, degree one, and distance one. Two histories have the same final behavioral figure if and only if they are connected by endpoint detours, absorption, causal-garbage changes, and temporal carrier slides. Every final fibre contains one literal right-packed normal form determined by n and the final figure.*

“Right-packed” means that the carriers of the final chain occupy the latest possible consecutive rows, and everything below them is in a canonical form.

Proof. Strategy. Soundness is known for each move. For completeness, push every carrier upward by slides until the carriers are consecutive at the top, then canonicalize what is left below.

Soundness has been proved for each move. For completeness, absorb every final-live finite unroll of a recursive residual. The remaining live chain is behaviorally reduced.

Apply Lemma 8.3 between each two consecutive live residuals. Prepare the first hidden relay label by a garbage move (the relay row's label is unsupported). An inverse temporal slide advances the lower carrier through that relay time; if it is recursive, absorption restores the SELF presentation. Each slide decreases the sum of gaps between consecutive essential carriers. Working from the final root downward terminates with the whole reduced live chain in the latest possible consecutive rows.

Canonicalize endpoint spellings by detours. Below the last essential carrier there are three cases. A missing terminal leaves only causal garbage; a recursive terminal leaves a canonical SELF bottom block; and a terminal at the distinguished base has the unique supported relay $\epsilon 0 \cdots 0$, with only its unused labels free. Garbage moves choose the least remaining data. The result depends only on n and the final figure, and is the right-packed history $\text{RP}_n(F)$. Histories in one final fibre both reach that form. $\square$

8.3 A path-level quotient

In the unary theorem, call two histories of one final fibre *chamber-equivalent* if they are connected by strong and garbage moves alone; a *chamber* is a class. The remaining coordinates of a chamber are the creation times $t_1 < \cdots < t_m$ of the essential residual carriers, omitting the arbitrary physical carrier of a terminal SELF residual. Temporal slides move one carrier into one adjacent vacant time. The set of increasing carrier tuples is smaller than the unrestricted set of m -subsets of $\{1, \dots, n\}$: the final root is always carried by the final node, so $t_m = n$. If the reduced chain has an ordinary missing or base terminal, put $r = m - 1$ and $c = n - m$. Then

$$1 \leq t_1 < \cdots < t_{m-1} < t_m = n, \quad \lambda_i = t_i - i,$$

and $0 \leq \lambda_1 \leq \cdots \leq \lambda_r \leq c$. These row lengths are the order ideals of the rectangle poset $[r] \times [c]$. If the terminal is recursive, its arbitrary physical SELF carrier is omitted. The pure recursive case $m = 0$ is a singleton; for $m \geq 1$, time 1 is reserved below the first retained finite carrier, so $r = m - 1$, $c = n - m - 1$, and $\lambda_i = t_i - i - 1$. This gives the order ideals of $[m - 1] \times [n - m - 1]$.

For a finite poset P , Ardila, Owen, and Sullivant construct a cube complex $X(P)$ whose vertices are order ideals and whose cubes remove arbitrary subsets of the maximal elements of one ideal; their general poset-with-inconsistent-pairs theorem makes $X(P)$ a rooted CAT(0) cube complex [1, Definition 2.3 and Theorem 2.5].

Proposition 8.5 (Carrier-surgery CAT(0) geometry). *For a fixed unary distance-one final figure and history length, the temporal slide graph on strong/garbage chambers is the undirected Hasse graph of the order-ideal lattice of the rectangle above. Filling every Boolean family of independent slides gives the CAT(0) cube complex $X(P)$. If the rectangle has r rows and c columns, then the complex has rc hyperplanes, dimension $\min(r, c)$, edge distance*

$$d(t, s) = \sum_{i=1}^r |t_i - s_i|,$$

and coordinatewise-median carrier tuples. In particular, its diamond-filled two-skeleton is simply connected.

Proof. The unique-relay lemma shows that strong and garbage moves preserve the essential tuple and that all histories with one tuple lie in one chamber. The temporal endpoint calculation realizes exactly an adjacent move into a vacancy, which is a cover relation in the componentwise order.

Conversely, the unique-relay construction realizes every allowed cover. In the λ coordinates a cover adds or removes one maximal rectangle box, so the filled cubes are exactly those of $X(P)$ and the cited theorem gives $\text{CAT}(0)$.

There is one hyperplane per rectangle box. A cube is an antichain of boxes, whose maximum size in a rectangle is $\min(r, c)$. Any path between two ideals must cross each box in their symmetric difference, and removing the boxes of the first difference in maximal order and adding the second in minimal order meets that bound; symmetric-difference size is the displayed carrier distance. The median ideal is

$$(I \cap J) \cup (J \cap K) \cup (K \cap I),$$

whose row lengths are the coordinatewise medians.

Orient its edges upward. At a maximal-rank vertex of a loop, equal incident edges cancel as a backtrack. Otherwise the two downward moves remove distinct maximal boxes and form a commuting lattice diamond. Replacing the peak by the other two sides strictly lowers the maximal-rank profile. Iteration contracts the loop. $\square$

Boundary. Proposition 8.5 is path-level coherence only after collapsing each strong/garbage chamber. Theorem 7.5 gives path coherence inside a garbage fibre. A finite elementary presentation of all mixed strong, garbage, and temporal derivations is not proved here.

9 Feedback, receipts, and a repaired allocation model

The preceding calculus was developed intrinsically. A direct palindrome construction supplies a deliberately hostile application because it forces a fresh feedback edge. Everything proved in this section is proved here; the palindrome construction itself is described only as far as the reader needs to see the knot, and its theorems are not assumed (Section 9.4).

9.1 The receipt knot

Consider a recognizer for $\text{EPAL}_2 = \{ww^R\}$ that keeps, for the input read so far, its longest even-palindromic suffix and a persistent sequence (“rope”) describing the letters to the left of that suffix. Whenever the recognizer discovers a new palindrome, it creates a fresh *record* for that occurrence; and because old records can never be modified, it must store in the fresh record everything a later step may need, including a fresh rope. The companion paper [9] argues that the last cell of that fresh rope must hold a *typed receipt*—a pointer—to the fresh record itself, so that when that cell is later reached the recognizer can find the record without a search. Thus the same input step contains the logical cycle

$$\text{fresh record} \longrightarrow \text{fresh rope} \longrightarrow \text{terminal cell} \longrightarrow \text{fresh record}.$$

An earlier internal version of the allocation model (numbered PC-83 in the project’s records) allowed a fresh field to name only an *already constructed* fresh slot, and its finite checker imposed the same topological order. Figure 5 is therefore a genuine proof defect in that version: the claimed source transition was outside its own model. The rest of this section is the repair, stated for arbitrary bounded record programs.

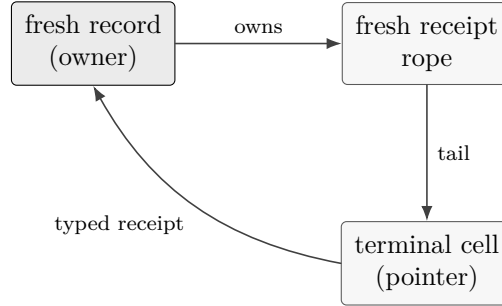


Figure 5: The same-step receipt knot. A fresh allocator that can only point to records that already exist cannot construct this finite strongly connected component.

9.2 Bounded simultaneous record transducers

Definition 9.1 (Simultaneous record batch). Fix constants s (roots), A (fresh slots), q (pointer fields per slot), and r (old dereference radius), together with finite control, tag, and payload alphabets. A transition may inspect tags and payloads reached within r old logical field dereferences from its roots. From that old observation it chooses the next control and one finite batch of at most A records. Each fresh field and next root targets missing, a bounded old address, or *any* active slot of the same batch. No fresh slot is inspected while the plan is selected, and control does not branch on physical pointer identity.

The last clause permits self, forward, and mutual fresh references. The batch is a finite labelled graph specified all at once. Three small batches show what is allowed: a self-loop (one slot pointing to itself), a forward reference (slot 0 pointing to slot 1), and a mutual cycle (slots 0 and 1 pointing to each other).

Theorem 9.2 (Simultaneous-batch compiler). *Every bounded simultaneous record transducer with parameters (s, A, q, r) has an exact letter-synchronous scaffolding-automaton implementation of degree*

$$d = s + Aq$$

and observation/update distance at most $r + 1$.

Proof. Strategy. Pack the whole batch into one node, use ports for fields, use SELF with a slot tag for every fresh target, and check that decoding is exact.

One physical scaffold node represents the whole logical batch. Its finite label records the active mask; every active tag and payload; and, for every root and field, the finite target-slot tag.

Ports $0, \dots, s - 1$ represent roots. Port $s + iq + j$ represents field j of logical slot i . A decoded pointer is a pair (v, i) of a physical historical node and a slot. An old logical address beginning at root u and following fields $j_1, \dots, j_m$, $m \leq r$, compiles to the physical word

$$u, \quad s + i_0q + j_1, \quad \dots, \quad s + i_{m-1}q + j_m,$$

where each next slot i_h is read from the finite target tag at the preceding edge. The observed depth- $(r + 1)$ neighborhood contains this information.

Use a missing descriptor for missing, the compiled word for an old target, and physical SELF for every fresh target. In the last case the new label supplies the requested target slot.

Assume the old physical heap decodes correctly. Induction along a logical old address proves that its physical word reaches the same old pointer. Missing decodes immediately. A fresh descriptor reaches the node currently being appended and its tag selects exactly the requested fresh slot. Applying these three cases to every root and field makes the decoded new finite adjacency table identical to the source batch, including all cycles. This is equality of finite labelled graphs; no recursive graph unfolding or fixed point is needed. Run induction gives exact simulation. All alphabets and plan branches are finite, proving the stated degree and distance bounds. $\square$

Remark 9.3 (Where the cycle lives). There is no contradiction with Theorem 4.1. The physical scaffold still has only backward edges and self-loops. A logical record pointer is the larger coordinate (v, i) , not the physical node v alone. Following a physical **SELF** edge keeps v fixed while its finite edge tag changes the logical slot from i to i' . An arbitrary finite cycle in the decoded fresh batch is therefore represented by one physical self-loop together with finite slot dynamics. The coordinate realization has moved cyclicity from physical time into the finite decoder state.

9.3 Staging a strict persistent sequence

The compiler solves the target representation problem but not the source evaluation problem. A cycle is safe only if the program can choose its allocation skeleton without following the fresh feedback edge. A data structure is *persistent* when old versions survive updates [6, §1], and *purely functional* in Kaplan and Tarjan’s strict sense when built only from the LISP primitives **car**, **cons**, **cdr** (pair construction and projection), with memoization excluded [11, p. 578, footnote 1].

Theorem 9.4 (Opaque-element staging). *Let a strict purely functional sequence operation:*

1. *use only finite constructor tests and a worst-case bounded number of fixed-arity constructor, projection, and pointer steps;*
2. *treat elements of its base type as opaque atoms, with no equality, hashing, pattern match, identity test, or traversal through a base element, while permitting inspection of the representation’s own pairs, buffers, and recursive child structures; and*
3. *return a fixed finite tuple of roots and removed elements.*

For fixed old roots and symbolic new elements, its control branch and fresh structural allocation graph are determined independently of the symbolic values. If one symbol denotes a fresh owner that stores a resulting sequence root, the owner and structural graph form one bounded simultaneous batch and compile by Theorem 9.2.

Proof. Run the bounded operation with distinct symbols in place of new elements. Induct over its bounded primitive steps. A constructor test or projection reads only an old structural record or an earlier fresh structural record. Element opacity prevents the control path from following or branching on a symbol. A constructor step adds one fixed-arity structural record whose fields are old pointers, earlier structural pointers, or symbolic elements. Thus the branch and an acyclic bounded structural allocation graph are fixed before element substitution.

Add the fresh owner slot and replace its symbol by a pointer to that slot. The substitution can create a cycle from the owner through the sequence structure and back through an opaque element field, but it cannot change the selected branch or work bound. Theorem 9.2 represents every edge of the resulting finite graph using old paths, missing, or **SELF** plus a slot tag. $\square$

Kaplan and Tarjan’s Theorem 5.1 gives worst-case constant-time push, pop, inject, and catenation

for regular catenable steques in exactly this strict model [11, p. 592]. Elements at the base type are parameters to those structural operations; the representation inspects its own fixed pair and buffer constructors at recursive levels but never the base element stored at a payload leaf. Hence their theorem supplies the source premise required by Theorem 9.4, not merely an asymptotic slogan.

9.4 The conditional application

Proposition 9.5 (Feedback programs compile). *Let a persistent program, at each input letter, start from a fixed number s of roots and perform a fixed finite composition of strict purely functional sequence operations satisfying the hypotheses of Theorem 9.4 on opaque pointer elements. In addition it may create a bounded number of fresh fixed-arity records whose layouts are determined by the same old observation and whose fields target old records, resulting sequence roots, or records in that fresh finite batch. At most one designated fresh owner’s pointer is stored as a base-type element of those sequences. Suppose the complete per-letter computation, including those sequence operations, follows old logical fields only within some fixed dereference radius r . Then the program is a bounded simultaneous record transducer, and it has an exact letter-synchronous implementation by a finite scaffolding automaton of degree $s + Aq$ and distance at most $r + 1$ for suitable finite A, q .*

Proof. By Theorem 9.4, each sequence operation, run with the fresh owner’s pointer replaced by a symbol, has a control branch and an acyclic structural allocation graph determined by old structure alone; a fixed finite composition of such operations therefore allocates a bounded acyclic skeleton and reads a bounded number of old fields. Adding the bounded fixed-arity records, including the designated owner when present, and substituting that owner for its symbolic base element yields one finite batch of bounded size. Its fields target only bounded old addresses, structural sequence roots, or active fresh slots. This is a transition of Definition 9.1 with A the total allocation bound and q the maximal record arity. Theorem 9.2 compiles it. $\square$

Example 9.6 (the direct even-palindrome recognizer). The companion paper [9] constructs a recognizer for EPAL_2 whose per-letter update is a composition of at most six catenable-steque operations (pop, inject, push, two catenations, and one pop or catenation) on opaque typed receipts, with one fresh owner record per nonreset step whose pointer is injected as the terminal element of a fresh rope. Its two table-rope values and live-rope value are the bounded auxiliary fixed-arity records allowed by Proposition 9.5; their fields contain finite tags, old receipts, or resulting sequence roots. If that construction is correct as claimed there, Proposition 9.5 applies to it, and Theorem 2.3 (with the observation that EPAL_2 is its own reversal) then yields $\text{EPAL}_2 \in \text{PEG}$ by a route that uses neither a Turing machine nor the classical real-time palindrome theorem. This paper does not re-prove the companion paper’s receipt recurrences or its acceptance invariant, and $\text{EPAL}_2 \in \text{PEG}$ is not a result of this paper: it is a theorem of the earlier transfer paper [10] and, by the second route, of [9]. What the example establishes here is narrower and entirely proved above: the same-step cycle that broke the topologically ordered allocator is no obstacle for a scaffold.

What this does and does not say. Proposition 9.5 repairs the allocation seam in general. It does not assert that any particular program satisfies its hypotheses; that is a separate proof obligation for each application.

10 Open boundaries

The proved general object is strong-trace equivalence. The broader program begins where final observation forgets that trace.

1. **Unrestricted final presentation.** Determine whether a finite graph-local move family presents final behavioral equivalence at all fixed degrees and distances. In plain terms: is there, for every d, k , a finite list of local rewrites connecting any two histories with the same final figure? Dedicated relation exchange may be a primitive or may be derivable by refactoring older rows.
2. **Elementary higher coherence.** The strong normalizer gives one object normal form, garbage fibres are cubical, and unary carrier chambers form distributive lattices. A finite set of cells presenting all mixed derivations—elementary identities between sequences of moves of all three kinds—is open.
3. **Support handoff.** A temporal slide can transfer one visible label obligation across a causal cut. Classify its interactions (critical pairs) with detours and absorption.
4. **Navigation monoids.** Characterize which finitely generated submonoids of the regressive chain monoid occur at fixed degree, distance, and uniform finite control.
5. **Functional sharing versus branching.** Separate ordinary immutable record DAGs from unrestricted scaffold navigation first at the dynamic or observational level, before seeking a language separation.
6. **Comparator theorem.** Construct the precise structure-preserving translation, if one exists, from the guarded causal presentation calculus to an established cyclic term-graph rewriting system [2] or to the categorical presentation of term graphs [8]. Novelty should be judged only after that comparison.

11 Assurance, provenance, and nonclaims

The theorem proofs in this paper are ordinary written mathematical proofs. Executable enumeration is used separately to find counterexamples and corroborate finite instances. No proof assistant is used as a stronger label for the mathematics here.

Layer	What it establishes
Written proof	the stated theorem for all objects satisfying its hypotheses
Finite audit	exact behavior over one explicitly bounded search domain
Source audit	the external theorem or model says what the paper attributes to it
Specialist review	an independent reader has attacked correctness and positioning
Novelty review	the contribution has been compared broadly enough to support priority language

The public package includes a theorem inventory, proof audit, source audit, finite audit code, and exact output. The finite audits enumerate bounded histories (for instance all unary distance-one histories up to a fixed length), search for split fibres, and check the displayed normalizers; their counts are recorded in the package. They corroborate; they are not the proofs.

What is inherited. Coalgebraic minimization and functional bisimulation [15]; ordinary acyclic-state merging [5]; general cyclic term-graph fold/unfold and confluence [2]; the scaffold model and its PEG correspondence [12]; strict real-time catenable steques [11]. No unrestricted final-equivalence or finite higher-coherence theorem is claimed; the palindrome example is not a theorem of this paper; and at this review stage no novelty claim is made.

What the viewpoint contributed. The separation in Figure 1 repeatedly changed the mathematical target: treating physical endpoint equality and behavioral equality as different observations produced the unfolding quotient and exposed pure aliases as presentation data; retaining every prefix figure produced a right-stable strong congruence and the detour/unroll normal form; forgetting prefixes only afterward exposed temporal carrier motion and causal support as final-observation coordinates; treating a receipt as a typed coordinate rather than a semantic existence assertion exposed the same-step allocation cycle; and moving from topological allocation to a simultaneous coordinate realization turned that apparent impossibility into ordinary **SELF** feedback with finite slot state. Each arrow of

presentation $\longrightarrow$ exact dynamics $\longrightarrow$ observation $\longrightarrow$ change of representation

is a theorem obligation. In particular, a universal graph embedding is not automatically a bounded coordinate realization, and final language equality does not imply equality of causal histories.

Provenance and AI assistance. The arguments were developed in FPRD Lab, an AI-assisted research project. The mathematics is the author’s responsibility; AI systems drafted and checked text and code under the author’s direction, and an adversarial pass found the allocation defect of Section 9 in an internal draft before publication. The primary sources cited were read in full and the exact statements used are quoted with locators; copies and extracted text are held locally and are not part of the public repository. This paper and the companion paper [9] were developed contemporaneously; neither treats the other as established, and every theorem of this paper is proved from its own text and the cited literature.

12 Conclusion

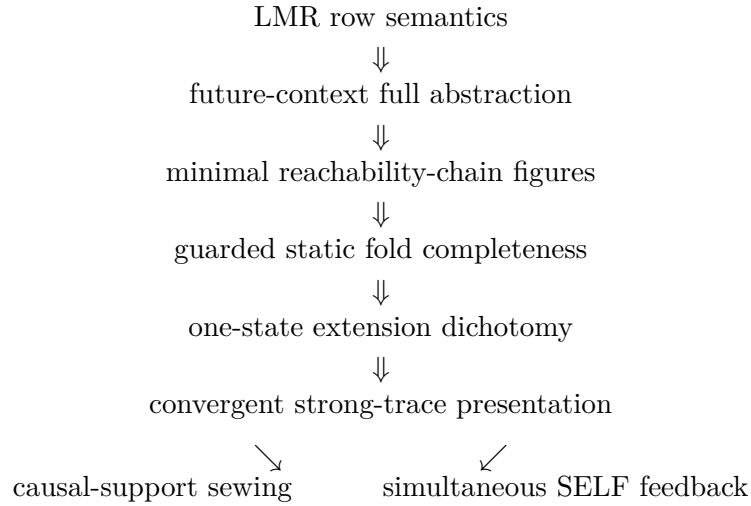
Persistent scaffold computation has an intrinsic algebra before one asks which language a machine recognizes. Its generated behavioral figures are ordered residual systems. Guarded static folds collapse equivalent presentations; chronological detours and one **SELF**-unroll lift that collapse to a convergent presentation of full strong traces. Final observation then reveals a second layer—causal support, garbage cubes, temporal carriers, and cross-cut sewing—whose unrestricted presentation remains open.

The receipt knot explains why this distinction matters computationally. The semantic record exists, but a topologically ordered presentation cannot build it. In a simultaneous scaffold coordinate system the same cycle is a finite batch of **SELF** edges with slot tags. That change of representation repairs the allocation seam of the palindrome application, but more importantly it exhibits the general principle: complexity can move between semantic structure, presentation dynamics, and observation without disappearing.

A Move and preservation table

Move	Local action	Guaranteed preservation
D endpoint detour	rename one bounded word with the same physical endpoint	every prefix figure and every common suffix
A/U absorption, unroll	replace an old recursive carrier by fresh SELF	every prefix figure and every common suffix
G garbage change	change data outside one source-certified replay support	final rooted concrete scaffold
T temporal slide	move a unary residual carrier across one adjacent time cut	final behavioral figure
R routing crossing	reassign demanded router slots injectively and transport selectors	final concrete consumer endpoints
E^{rel} relation exchange	exchange two dedicated factorizations with one old endpoint	final concrete consumer endpoints

B The central implication map



References

- [1] Federico Ardila, Megan Owen, and Seth Sullivan. Geodesics in $CAT(0)$ cubical complexes. *Advances in Applied Mathematics*, 48(1):142–163, 2012. <https://doi.org/10.1016/j.aam.2011.06.004>.
- [2] Zena M. Ariola and Jan Willem Klop. Equational term graph rewriting. *Fundamenta Informaticae*, 26(3–4):207–240, 1996. <https://doi.org/10.3233/FI-1996-263401>. Earlier version: CWI report CS-R9552, 1995 (the version inspected).
- [3] Janusz A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, 1964. <https://doi.org/10.1145/321239.321249>.

- [4] Noam Chomsky and Marcel-Paul Schützenberger. The algebraic theory of context-free languages. In P. Braffort and D. Hirschberg, editors, *Computer Programming and Formal Systems*, pages 118–161. North-Holland, 1963.
- [5] Jan Daciuk, Stoyan Mihov, Bruce W. Watson, and Richard E. Watson. Incremental construction of minimal acyclic finite-state automata. *Computational Linguistics*, 26(1):3–16, 2000. <https://aclanthology.org/J00-1002/>.
- [6] James R. Driscoll, Neil Sarnak, Daniel D. Sleator, and Robert E. Tarjan. Making data structures persistent. *Journal of Computer and System Sciences*, 38(1):86–124, 1989. [https://doi.org/10.1016/0022-0000\(89\)90034-2](https://doi.org/10.1016/0022-0000(89)90034-2).
- [7] Bryan Ford. Parsing expression grammars: A recognition-based syntactic foundation. In *Proceedings of POPL 2004*, pages 111–122. ACM, 2004. <https://doi.org/10.1145/964001.964011>.
- [8] Tobias Fritz and Wendong Liang. Free gs-monoidal categories and free Markov categories. *Applied Categorical Structures*, 31(2):21, 2023. <https://doi.org/10.1007/s10485-023-09717-0>. arXiv:2204.02284.
- [9] Joshua Gay. Persistent receipts and simultaneous records: a direct scaffold for even palindromes. FPRD Lab review paper, Draft 2, 22 August 2026. <https://gitlab.com/insight-forge/boxlab/fprd-lab/-/tree/main/publications/direct-palindrome-scaffolding>.
- [10] Joshua Gay. Real-time multitape languages transfer to parsing expression grammars. FPRD Lab review paper, released 5 August 2026. <https://gitlab.com/insight-forge/boxlab/fprd-lab/-/tree/main/publications/persistent-stacks-scaffolding>.
- [11] Haim Kaplan and Robert E. Tarjan. Purely functional, real-time dequeues with catenation. *Journal of the ACM*, 46(5):577–603, 1999. <https://doi.org/10.1145/324133.324139>.
- [12] Bruno Loff, Nelma Moreira, and Rogério Reis. The computational power of parsing expression grammars. *Journal of Computer and System Sciences*, 111:1–21, 2020. <https://doi.org/10.1016/j.jcss.2020.01.001>. Expanded preprint arXiv:1902.08272v2 (2020); definition and theorem numbers in this paper follow the preprint.
- [13] Anil Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544, 1958. <https://doi.org/10.1090/S0002-9939-1958-0135681-9>.
- [14] Jean-Éric Pin. *Mathematical Foundations of Automata Theory*. Lecture notes, MPRI; author-hosted, version retrieved 22 August 2026. <https://www.irif.fr/~jep/PDF/MPRI/MPRI.pdf>.
- [15] Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000. [https://doi.org/10.1016/S0304-3975\(00\)00056-6](https://doi.org/10.1016/S0304-3975(00)00056-6).

Afterword

Persistent scaffold computation: causal presentations, sewing, and feedback

Methodological and contribution addendum. The paper's mathematical scope and review status remain as stated in the paper.

The Lab's approach

FPRD means **Finitely Presented Recognition Dynamics**. Its starting point is to distinguish a finite description, the possibly infinite process it generates, and what an observer can learn from that process. Equal answers need not come from equal internal structures. Conversely, an infinite state space may have a short, exact description and useful coordinates; failure to compress it into finitely many states need not end the investigation.

A small example: balanced parentheses

The grammar $S \rightarrow (S)S \mid \epsilon$ presents the language of balanced parentheses. To recognize a prefix, keep the number of unmatched opening parentheses, or a failure state if a closing parenthesis arrived too early. An opening parenthesis increments the count. A closing parenthesis decrements a positive count; at zero it enters failure. Failure persists. At the end, accept exactly when the count is zero.

The prefixes $($ and $(($ leave the same future task: any suffix that completes one completes the other. The prefix $(($ leaves a different task: $)$ completes the first two prefixes but not this one. These future tasks are **residuals**. There are infinitely many counts, yet a finite set of update rules generates their dynamics. This is a classical automata example, not a discovery of FPRD Lab. It illustrates the research question: which coordinates preserve all relevant futures, and what does updating or reading those coordinates cost?

From viewpoint to proof

In practice, the Lab fixes the observable and the allowed operations, compares presentations of the same problem, and seeks an exact translation or invariant. A failed shortcut is useful when a counterexample identifies the missing coordinate or hypothesis. Small computations test particular uncertainties; a proof must explain why the result holds beyond those tests. Costs, relation sides, boundary conditions, and the direction of each translation remain explicit.

AI-assisted exploration, source comparison, executable checks, and adversarial reconstruction support this work. Their agreement is not automatically independent evidence: shared assumptions must be exposed. A formal development certifies its stated formal theorem, not every interpretation attached to it. Each paper retains its own evidence and review status.

The approach is most useful when a poor representation hides a reusable construction, a compatibility condition, or a sharp obstruction. Outside recognition theory, it can guide the search for a structural reformulation without producing a literal residual dynamical system. The account below distinguishes that methodological influence from the ingredients of the proof.

Contribution to this publication

This addendum describes the current HTML reading edition. The preceding PDF pages retain their earlier edition; the afterword does not silently revise them.

FPRD's contribution to this paper

This paper turns the distinction between presentation and observation into its main mathematical object. A row word, the physical history it builds, the behavior visible after every prefix, the final rooted behavior, and the final acceptance answer retain progressively different information. The Lab asked which transformations preserve each one.

That separation produced two different problems. Preserving the whole sequence of prefix observations leads to a strong-trace congruence and a guarded detour/unroll normal form. Preserving only the final observation permits additional changes in when useful records are carried. The paper analyzes those changes without promoting final agreement to agreement under every future context.

Its causal-support construction also distinguishes an unreachable node from an irrelevant construction step. A final edge can depend on an old edge that was copied when the final graph was built. A valid garbage change must respect that replay dependency. Tracking those dependencies supplies the sewing argument and, in the unary case, a concrete geometry of carrier placements.

The feedback analysis exposed a second representation error: an allocation model requiring fresh records to be built in topological order could not express a cycle required by a receipt. Simultaneous finite slots and the scaffold's existing SELF descriptor repair that interface. The paper proves the compiler principle separately from its companion's palindrome application.

Inherited ingredients and the boundary of the contribution

Loff, Moreira, and Reis supply the machine model. Coalgebraic minimization, bisimulation, persistent data structures, and cyclic term-graph rewriting are established sources, credited in the paper. The Lab's contribution is the source-sensitive causal presentation calculus and its exact preservation theorems, including the distinction between replay support and final reachability.

The finite history checks helped locate failures; the causal arguments establish the general statements. No complete proof-assistant formalization is claimed. The unrestricted final-equivalence calculus remains open, and comparison with existing term-graph presentations remains a separate obligation. A normal form for a semantic object is not automatically a bounded online procedure for finding it.

Where to inspect the contribution: the paper, Sections 3–9 and 11, including the move-preservation table and the repaired allocation model.