

Bounded Simultaneous Record Transducers and a Direct Palindrome Scaffold

Joshua Gay
FPRD Lab, Insight Forge

21 August 2026

Abstract

Loff, Moreira, and Reis conjectured that the language $\text{EPAL}_2 = \{ww^R : w \in \{0,1\}^*\}$ has no parsing expression grammar. An earlier FPRD Lab paper refuted that conjecture by transporting a classical strict real-time multitape palindrome recognizer through scaffolding automata. This paper gives a second route that is intrinsic to persistent records.

The construction maintains the longest even-palindromic suffix together with one reversed occurrence context. Type-indexed receipt ropes make suffix-link occurrences directly available, and a purely functional real-time catenable steque supplies bounded head, tail, insertion, and catenation. A fresh palindrome record and one receipt cell point to one another in the same input step. This exposes a defect in a topologically ordered fresh-allocation model. We repair it by defining bounded simultaneous record transducers: one step may append a fixed finite labelled batch graph with arbitrary intra-batch references. Such batches compile into one scaffold node by representing every fresh edge with **SELF** plus a finite slot tag. An opaque-atom staging theorem shows how a strict acyclic steque allocation skeleton can be tied to its fresh owner only at this simultaneous encoding boundary.

The resulting finite scaffolding automaton recognizes EPAL_2 directly. The Loff–Moreira–Reis correspondence therefore gives $\text{EPAL}_2 \in \text{PEG}$ without using a real-time Turing-machine model or Slisenko’s palindrome theorem. The paper is released for specialist review. It makes no claim that the final membership result is first, and no novelty claim is made pending broader literature review.

1 Result and assurance boundary

A parsing expression grammar (PEG) is a deterministic recognition formalism whose ordered choice and predicates differ fundamentally from the usual nondeterministic reading of a context-free grammar. Loff, Moreira, and Reis introduced scaffolding automata and proved that a language is recognized by a PEG exactly when its reversal is recognized by a finite scaffolding automaton [2]. They conjectured that

$$\text{EPAL}_2 = \{ww^R : w \in \{0,1\}^*\}$$

does not belong to PEG.

FPRD Lab has already refuted that conjecture by compiling strict real-time multitape machines into scaffolding automata and using a classical real-time palindrome theorem. The result here is not a first refutation. Its purpose is to remove the Turing-machine detour and identify the persistent data structure that the scaffold itself executes.

The proof has four layers.

1. The longest even suffix and its reversed left context obey an exact three-case recurrence.
2. Typed mirror receipts and bridge-shadow identities turn that recurrence into a fixed collection of persistent rope operations.
3. Kaplan and Tarjan’s purely functional real-time catenable steque [1] supplies those rope operations with worst-case bounded structural work.
4. A bounded simultaneous record transducer compiles letter-for-letter into one finite scaffolding automaton.

The written arguments are mathematical proofs. The accompanying programs are finite corroborations and regressions, not proofs of the infinite statements. The later OCaml/Rocq implementations of Viennot et al. [3] are used as implementation calibration, not as the functional-correctness premise. No Lean reconstruction of this paper is claimed or required. External specialist review and novelty evaluation remain open.

2 Scaffolds and simultaneous record batches

2.1 The Loff–Moreira–Reis interface

A scaffolding automaton has finite input, control, and working alphabets, a fixed positive degree d , and a fixed observation distance k . Its initial scaffold is one unlabelled base node. On each arriving letter, the transition sees the depth- k labelled tree unfolding of the old top and appends exactly one new labelled node. Each new port is either missing, follows a port word of length at most k from the old top, or is **SELF**, which names the node being created. Acceptance is determined by the finite control after the last input letter. The unfolding carries no node identity, so the transition cannot test whether two observed paths alias.

We use only the following direction of the correspondence.

Theorem 2.1 (Loff–Moreira–Reis [2]). *If a finite scaffolding automaton decides a language L , then L^R belongs to ordinary complete-match PEG.*

The existing FPRD Lean development independently reconstructs this sufficient direction, including automata that use **SELF**. The present paper uses the published theorem; it does not use or claim the converse direction.

2.2 The corrected source model

The receipt construction below creates a fresh owner record P and a fresh rope cell whose receipt points back to P . A source model that allows only earlier fresh slots cannot express this step. We therefore make simultaneous allocation explicit.

Definition 2.2 (Bounded simultaneous record transducer). Fix constants s, A, q, r . A bounded simultaneous record transducer has s current roots, allocates at most A active slots per input symbol, gives each record q pointer fields, and follows at most r old-record fields from its current roots in one step. Its control states, record tags, and nonpointer payloads are finite.

After its bounded old observation, one transition chooses a finite labelled batch graph. Every new

root and field targets one of:

$\emptyset$, an old address of at most r dereferences, any active slot of the fresh batch.

The last case includes self-loops, forward references, and mutual cycles. The whole finite adjacency table is specified simultaneously. No fresh record is inspected while the plan is chosen.

The initial roots may be missing. Any fixed finite initial object can be absorbed into finite control and materialized in the first batch.

Theorem 2.3 (Simultaneous batch compiler). *Every transducer of Definition 2.2 compiles exactly and letter-synchronously into a finite scaffolding automaton of degree*

$$d = s + Aq$$

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

Proof. One physical scaffold node represents the complete logical batch at one input time. Its finite label records the active mask, every tag and payload, and the finite target-slot tag associated with each present root or field.

Ports $0, \dots, s - 1$ represent the next roots. Port $s + iq + j$ represents field $j < q$ of slot $i < A$. Thus a decoded logical pointer is a pair (v, i) of a physical scaffold node and a finite slot number.

Consider an old logical address that starts at root u and follows fields $j_1, \dots, j_m$, with $m \leq r$. The current labelled unfolding reveals the slot reached after every prefix. Compile the address to the raw port word

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

where i_h is the slot tag at the preceding logical pointer. This word has length at most $r + 1$ and reaches exactly the old target.

Use a missing descriptor for $\emptyset$ and the compiled word for an old target. For every fresh target use physical **SELF** and record the intended logical slot in the new finite label. This rule makes no distinction between an earlier target, a later target, a self-loop, or one edge of a mutual cycle.

Assume that the old top decodes to the source roots and old heap. Induction on an old logical address proves that its raw word decodes correctly. Missing decodes to missing. A fresh descriptor reaches the new physical node by **SELF** and then its target-slot tag selects exactly the requested fresh logical record. Applying these facts independently to every root and field makes the decoded finite labelled adjacency table identical to the source batch. This comparison does not recursively unfold a cyclic graph, so no fixed-point semantics is involved. Tags, payloads, control, and acceptance are copied directly.

The missing-root base case and the one-step commuting square give the full run invariant by induction on input length. All parameter ranges and labelled plans are finite, so the compiled automaton is finite. $\square$

3 The occurrence-context recurrence

Let S be the input prefix read so far. Let $P(S)$ be its longest even-palindromic suffix and write

$$S = CP, \quad \ell = C^R.$$

The word ℓ is the left-context stack of the current occurrence; its first letter lies immediately to the left of P .

For an even palindrome P and a letter a , let $D_a(P)$ be the longest proper even-palindromic suffix Q for which

$$P = ZaQ,$$

and put $\omega_a(P) = Z^R$. Both are undefined when no such factorization exists.

Theorem 3.1 (Occurrence-context factorization). *After reading a , the new longest even suffix P' and reversed context ℓ' are exactly*

$$(P', \ell') = \begin{cases} (aPa, \text{tail}(\ell)), & \ell = a\ell_0, \\ (aD_a(P)a, \omega_a(P)\ell), & \ell \notin a\Sigma^*, D_a(P) \text{ defined}, \\ (\epsilon, aP\ell), & \ell \notin a\Sigma^*, D_a(P) \text{ undefined}. \end{cases}$$

Moreover,

$$S \in \text{EPAL}_2 \iff \ell = \epsilon.$$

Proof. Every nonempty even-palindromic suffix of Sa has the form aQa , where Q is an even-palindromic suffix of S immediately preceded by a .

If the current longest suffix P is externally preceded by a , then $Q = P$ is maximal. Removing that external letter removes the first symbol of ℓ . Otherwise every eligible Q is a proper suffix of P , so maximality selects $D_a(P)$. Writing $P = ZaD_a(P)$ shows that the new context is CZ and hence has reversal $Z^R C^R = \omega_a(P)\ell$. If no eligible proper suffix exists, the longest even suffix is empty and the reversed full prefix is $aP\ell$.

Finally ℓ is empty exactly when C is empty, which is exactly when the whole prefix equals its longest even-palindromic suffix. $\square$

The chunks $\omega_a(P)$ themselves satisfy a finite-alphabet recursion. If $L = \lambda(P)$ is the longest proper even-palindromic suffix and

$$P = RbL = LbR^R,$$

write $B_P = R^R$. Then

$$D_b(P) = L, \quad \omega_b(P) = B_P,$$

and for $a \neq b$,

$$D_a(P) = D_a(L), \quad \omega_a(P) = \omega_a(L)bB_P,$$

with the same definedness on both sides. Indeed, every proper even-palindromic suffix of P is a suffix of L ; the bridge letter b selects L itself, while every other letter delegates to L and adds the displayed intervening block.

4 Typed mirror receipts

Theorem 3.1 reduces online recognition to catenating type-indexed chunks. It does not yet say how a fresh occurrence obtains its suffix-link record without a search. The needed object is a typed receipt.

Theorem 4.1 (Typed mirror receipt). *Suppose an update selects a longest suffix occurrence Q of the old input that is immediately preceded by a , and creates $Y = aQa$. Then*

$$\lambda(Y) = \begin{cases} aD_a(Q)a, & D_a(Q) \text{ defined,} \\ \epsilon, & D_a(Q) \text{ undefined.} \end{cases}$$

The palindrome $\lambda(Y)$ has a complete occurrence in the old input and a pre-existing birth record. One old typed pointer can therefore install the suffix-link field of Y .

Proof. Every proper nonempty even-palindromic suffix of $Y = aQa$ is aRa , where R is a proper even-palindromic suffix of Q immediately preceded in Q by a . Maximizing R gives the displayed formula. Since Y is a palindrome, $\lambda(Y)$ is also a proper prefix of Y ; that prefix ends before the newly appended final a , so the complete occurrence lies in the old input.

At the first occurrence of any nonempty palindrome H , it is the longest palindromic suffix of the prefix ending there. Otherwise a longer palindromic suffix would contain H as a proper prefix as well as a suffix, giving an earlier occurrence of H . The online construction has therefore already made a birth record for $\lambda(Y)$. The empty case uses one distinguished empty record. $\square$

The word “typed” is essential. A raw endpoint does not identify the needed record at bounded depth.

Proposition 4.2 (Endpoint-only escape). *For $m \geq 2$, let $S_m = 0^{2m}1100$. Its longest even-palindromic suffix is 001100 and the suffix link is 00 . The mirror occurrence of 00 ends at the historical prefix 0^{2m} , whose longest suffix-link chain is*

$$0^{2m}, 0^{2m-2}, \dots, 00, \epsilon.$$

Recovering 00 from that untyped endpoint takes exactly $m - 1$ suffix-link steps.

This is a representation-specific escape theorem, not a language or scaffold lower bound. It says that the positive construction must transport the typed record pointer itself.

5 Receipt ropes and shadow transport

For an even palindrome A and word $w = c_1 \cdots c_k$, define the anchored path $\Pi(A, w)$ by

$$A_0 = A, \quad A_i = c_i A_{i-1} c_i.$$

Cell i stores $(c_i, \lambda(A_i))$. Its receipt trace is

$$\Lambda(A, w) = (\lambda(A_1), \dots, \lambda(A_k)),$$

and its headered form is

$$\widehat{\Pi}(A, w) = (\lambda(A); (c_1, \lambda(A_1)), \dots, (c_k, \lambda(A_k))).$$

The header is the receipt required if the anchor becomes current; taking a tail promotes the first cell’s receipt to the new header. Anchored paths obey the literal catenation law

$$\Pi(A, uv) = \Pi(A, u) \cdot \Pi(\text{wrap}_u(A), v).$$

The physical palindrome states in a retained context tail can change after a prepend or reset. The next lemma shows that the typed receipt trace does not.

Lemma 5.1 (Bridge-shadow identity). *Suppose $H = RbP$, $\lambda(H) = P$, and P is the longest even-palindromic suffix of $w^R P$. If $w = \epsilon$, or if the first letter of w differs from b , then*

$$\Lambda(H, w) = \Lambda(P, w).$$

Proof. For a letter c and even palindrome X , put $G_c(X) = \lambda(cXc)$. Every proper even-palindromic suffix of X is a suffix of $\lambda(X)$. Hence

$$G_c(X) = \begin{cases} c\lambda(X)c, & c \text{ is the bridge before } \lambda(X), \\ G_c(\lambda(X)), & \text{otherwise.} \end{cases} \quad (1)$$

Write $w = c_1 \cdots c_k$, put $u_i = c_1 \cdots c_i$, and define

$$X_i = u_i^R H u_i, \quad Y_i = u_i^R P u_i, \quad N_i = |P| + i.$$

Both X_i and Y_i end in the same word $T_i = P u_i$, of length N_i . We prove for every $i \geq 1$ that

$$\lambda(X_i) = \lambda(Y_i) = K_i \quad \text{and} \quad |K_i| < N_i. \quad (2)$$

For $i = 1$, the hypothesis $c_1 \neq b$ and (1) give $G_{c_1}(H) = G_{c_1}(P)$. This common palindrome is a proper suffix of $c_1 P c_1$, so its length is at most $|P| < N_1$.

Assume (2) at $i < k$. Since K_i is shorter than the common suffix T_i , the letter immediately before its terminal occurrence lies inside T_i . It is therefore the same bridge letter in X_i and Y_i . Equation (1), with $c = c_{i+1}$, now gives the same next receipt on both sides: it is either $cK_i c$ when c is that common bridge, or $G_c(K_i)$ otherwise.

It remains to retain the strict length bound. In the second case $|G_c(K_i)| \leq |K_i| < N_i < N_{i+1}$. In the first case the only way $|cK_i c|$ could reach N_{i+1} is, by parity, that N_i is odd and $|K_i| = N_i - 1$. Then the common suffix has the form $T_i = cK_i$. Because K_i is a palindrome,

$$(u_i c)^R P = c u_i^R P = c T_i^R = c K_i c$$

is an even-palindromic suffix of $w^R P$ strictly longer than P : the word $u_i c$ is a prefix of w . This contradicts the maximality hypothesis. Thus (2) follows by induction, and its receipt equalities are precisely $\Lambda(H, w) = \Lambda(P, w)$. $\square$

For nonempty Y with suffix link L , write

$$Y = RbL = LbB.$$

When $D_c(Y)$ is defined, define the direct headered rope

$$\Omega_c(Y) = \widehat{\Pi}(cD_c(Y)c, \omega_c(Y)).$$

When it is undefined, define the reset rope

$$Z_c(Y) = \widehat{\Pi}(\epsilon, cY).$$

Theorem 5.2 (Receipt-rope recurrence). *Let ρ_b be the header receipt of $\Omega_b(Y)$. For $c \neq b$,*

$$\Omega_c(Y) = \begin{cases} \text{undefined,} & \Omega_c(L) \text{ undefined,} \\ \Omega_c(L) \cdot (b, \rho_b) \cdot \text{body } \Omega_b(Y), & \text{otherwise,} \end{cases}$$

and, whenever $Z_c(Y)$ is defined,

$$Z_c(Y) = Z_c(L) \cdot (b, \rho_b) \cdot \text{body } \Omega_b(Y).$$

Proof. The chunk recursion above gives the raw-word factorizations. The initial $\Omega_c(L)$ or $Z_c(L)$ portion is literal. At its endpoint, the next b cell has the bridge receipt ρ_b . The remaining raw word is B . Lemma 5.1 identifies every receipt along that retained B tail with the body of $\Omega_b(Y)$. Definedness of the off-bridge direct rope is the same at Y and L because every proper suffix of Y lies under L . $\square$

Theorem 5.3 (Full abstract receipt closure). *Over a fixed alphabet, a fresh occurrence record, its complete direct/reset rope table, and the updated live context are obtained from old receipts by a fixed number of head, tail, singleton, inject, and catenation operations.*

Proof. Suppose the update creates $P = aQa$ and Theorem 4.1 supplies $L = \lambda(P)$. Select the old direct rope $\Omega_a(Q)$ when $D_a(Q)$ is defined and the old reset rope $Z_a(Q)$ otherwise. Its body is nonempty; write it $(b, H)\mathcal{T}$. The fresh bridge data are

$$b_P = b, \quad \Omega_b(P) = (H; \mathcal{T}(a, P)). \quad (2)$$

The last receipt is P itself by the mirror-prefix factorization. The bridge entry (2), Theorem 5.2, and the finite alphabet construct every other direct or reset rope by a constant-size table update.

Let $\mathcal{K}(P, \ell) = \hat{\Pi}(P, \ell)$ be the live context rope. The three cases of Theorem 3.1 give

$$\mathcal{K}' = \begin{cases} \text{tail } \mathcal{K}(P, \ell), & \ell = a\ell_0, \\ \Omega_a(P) \cdot \text{body } \mathcal{K}(P, \ell), & D_a(P) \text{ defined,} \\ Z_a(P) \cdot \text{body } \mathcal{K}(P, \ell), & D_a(P) \text{ undefined.} \end{cases} \quad (3)$$

Lemma 5.1 makes the old body in the mismatch cases literally reusable even though its concrete anchor changes. Equations (2)–(3) use a fixed number of rope primitives and old fields. $\square$

Equation (2) contains the crucial same-step cycle: P stores the root of $\Omega_b(P)$, while its terminal cell stores the receipt P . This is why Definition 2.2, rather than a topologically ordered fresh heap, is the correct source model.

6 Opaque-atom staging

The receipt proof has reduced the online step to a constant number of catenable-sequence primitives. We now justify that these primitives expose a bounded record plan without pretending that an ordinary immutable functional value can tie the cycle in (2).

Theorem 6.1 (Kaplan–Tarjan [1]). *There is a purely functional regular catenable steque whose push, pop, inject, and catenation operations take worst-case constant time and preserve the representation.*

For their paper, Kaplan and Tarjan define “purely functional” to mean built only from `car`, `cons`, and `cdr`; memoization is explicitly excluded. This strict model is stronger for our purpose than a bare unit-cost RAM claim.

Theorem 6.2 (Opaque-atom staging). *Let a strict purely functional operation use finite constructor tests and a worst-case bounded number of `car`, `cons`, and `cdr` operations, without inspecting, comparing, hashing, or identity-testing its stored elements. With new elements replaced by symbolic atoms, its*

branch and fresh structural allocation graph are determined by old structure alone. The graph is finite, acyclic, and uniformly bounded.

If a symbolic element denotes a fresh owner record that stores one resulting root, the owner and skeleton form one bounded simultaneous batch. Substituting the owner slot for the atom is sound, even though it creates a cycle.

Proof. Fix the old structural roots. A worst-case constant bound gives a uniform number K of primitive evaluation steps. Replace each new element by a distinct symbol and follow the evaluation.

A constructor test, `car`, or `cdr` reads an old structural record or a structural record allocated at an earlier step. By hypothesis it never follows an element field. Its result and next branch are therefore independent of the symbols. A `cons` allocates one fixed-arity record whose fields are old pointers, earlier fresh structural pointers, or symbolic elements. Induction over the at most K steps yields a bounded acyclic structural allocation graph independent of the symbolic values.

Now add the fresh owner slot and replace the designated symbol by a pointer to that slot. The resulting cycle crosses from the owner to a structural root and returns through an opaque element field. Structural evaluation never follows the return edge, so neither the sequence denotation nor the resource bound changes. Theorem 2.3 compiles the whole simultaneous batch, including that feedback edge, with `SELF`. $\square$

Theorem 6.1 satisfies the hypotheses for the rope operations in Theorem 5.3. Receipt pairs are generic steque elements. A singleton is a push onto empty; body removal is pop; terminal insertion is inject; and rope composition is catenation. Thus every input update determines a bounded simultaneous record plan.

7 A direct palindrome scaffold

We assemble the construction. Finite control distinguishes the empty initial state and the three update cases. The persistent roots name the current occurrence record and live headered context rope. Each occurrence record has, for the two input letters, one direct or reset rope root, together with its suffix-link and bridge data. This is a fixed record layout.

On input a , inspect the live context header and the a -entry of the current record. Theorem 3.1 selects pop, internal prepend, or reset. Theorems 4.1 and 5.2 provide only old typed receipts. Theorem 5.3 expresses the fresh record, its two-entry table, and the new live context by a constant number of catenable-steque operations. Theorems 6.1 and 6.2 turn those operations into one bounded simultaneous record batch. Theorem 2.3 appends that batch as the one scaffold node required for the arriving letter.

All constructors, letters, table coordinates, and slot tags range over finite sets. The input alphabet is binary and the number of rope operations per step is fixed, so the roots, slots, fields, old observation radius, and fresh allocation bound are constants of one machine. Acceptance is the finite empty-body test from Theorem 3.1.

For completeness, one concrete inventory is as follows. Keep two current roots: the occurrence record and the live headered-rope value. A binary occurrence record has one suffix-link field, one finite bridge-letter payload, and two tagged rope entries. A rope value has a header receipt and a steque body root. Expanding (2)–(3) uses, in the worst branch, one source-body pop, one terminal

inject, one singleton push, two catenations for the off-bridge table entry, and one pop or catenation for the live context: at most six steque calls. If K bounds the primitive steps and allocations of one Kaplan–Tarjan call, the simultaneous batch therefore needs at most $6K + 4$ fresh slots: $6K$ structural slots, one occurrence owner, two table-rope values, and one live-rope value. Its old observation depth is at most $6K + 3$, and its uniform field bound is the maximum of the fixed occurrence, rope, and steque-record arities. These conservative bounds, rather than their numerical optimization, are all the compiler requires.

Theorem 7.1 (Direct palindrome scaffolding). *The binary even-palindrome language is recognized by a finite scaffolding automaton:*

$$\boxed{\text{EPAL}_2 \in \text{SA.}}$$

Proof. The construction above gives one letter-synchronous finite scaffolding automaton. Its decoded record state follows the exact recurrence by induction on the input length. It accepts exactly when the decoded left context is empty, which by Theorem 3.1 is exactly EPAL_2 . $\square$

Corollary 7.2 (Direct PEG theorem).

$$\boxed{\text{EPAL}_2 \in \text{PEG.}}$$

This proof route uses neither a strict real-time multitape machine nor Slisenko’s real-time palindrome theorem.

Proof. Apply Theorem 2.1 to Theorem 7.1. Since the reversal of ww^R is again of the form uu^R , the language EPAL_2 is invariant under reversal. $\square$

8 Modern source calibration

Viennot, Wendling, Gueneau, and Pottier give a later OCaml implementation and a separate Rocq implementation of Kaplan–Tarjan catenable dequeues [3]. Their paper clearly separates the evidence: the OCaml code is tested but not verified; the Rocq code has machine-checked functional correctness; there is no formal correspondence between the two; and the Rocq extraction does not establish the same constant-time bound because runtime indices are retained.

We therefore use Kaplan–Tarjan Theorem 6.1 as the mathematical premise. The modern source is a hostile implementation calibration. At the pinned commit, its operational core is strict and nonrecursive, but the audit found two exact mismatches with an opaque finite-record interface:

1. four classifiers read an exact unbounded buffer length; and
2. the noncatenable empty test uses OCaml polymorphic structural equality.

The first is extensionally replaced by fixed persistent end probes; no classifier uses more than twelve primitive root observations. The second is replaced by matching the outer empty constructor. On valid inputs the original equality already stops at that constructor, before any element is inspected, but the replacement makes opacity explicit. These repairs are checked through length 512. This evidence is not described as verification of the OCaml program and is not needed for Theorem 7.1.

9 Executable evidence and limits

The public package contains five independent Python programs. They check the three-case context recurrence, typed mirror receipts, receipt-rope closure, simultaneous cyclic batch decoding, and the modern catenable-interface calibration. The current complete run includes:

Finite audit	Cases or checks
Occurrence-context word updates	262,142 exhaustive; 100,000 random
Typed mirror updates	131,070 exhaustive; 20,000 random
Legal receipt-tail updates	65,534 exhaustive; 20,000 longer random
Typed old-address checks	5,119,788
Simultaneous compiler squares	250,000 plus three cyclic witnesses
Catenable threshold classifiers	3,063; maximum 12 observations

The programs use exact finite words and pointer graphs. They corroborate the displayed identities and protect against regression, but bounded computation is not the proof of a universal theorem.

Three boundaries should remain visible.

1. The conclusion $\text{EPAL}_2 \in \text{PEG}$ was already proved by the earlier FPRD real-time transfer route. This paper contributes a direct persistent record route.
2. Endpoint-only, type-birth-only, and bounded bridge-chasing failures are representation-specific. They are not lower bounds against all scaffolding automata.
3. External specialist review and a broader novelty search remain open. A complete written proof is not the same coordinate as novelty or review status, and no novelty claim is made here.

The reusable result beyond palindromes is Theorem 2.3: a scaffold node can execute one bounded simultaneous persistent record step, including finite semantic cycles. The direct palindrome construction is the first application studied here, not a claimed classification of all bounded persistent programs or all PEG languages.

Acknowledgment of inherited work

Kaplan–Tarjan catenable steques and the Loff–Moreira–Reis scaffold/PEG correspondence are inherited mathematical results. The Viennot et al. implementations are external work used for calibration. FPRD Lab contributes the occurrence-context formulation, typed receipt system, representation escape audits, simultaneous cyclic source model, tuple-packed compiler, opaque-atom staging bridge, direct composition, and the adversarial correction that exposed the same-step cycle before publication. Detailed attribution is recorded in the accompanying `FPRD-Lab-Contribution.md` and `SOURCE-AUDIT.md`.

References

- [1] 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>.

- [2] Bruno Loff, Nelma Moreira, and Rogerio Reis. The computational power of parsing expression grammars. 2020. <https://arxiv.org/abs/1902.08272v2>.
- [3] Jules Viennot, Arthur Wendling, Armael Gueneau, and Francois Pottier. Verified purely functional catenable real-time dequeues. 2025. <https://arxiv.org/abs/2505.07681v3>.