

Real-Time Multitape Languages Transfer to Parsing Expression Grammars

via Persistent Stacks in Scaffolding Automata

Joshua Gay*

31 July 2026

Abstract

We prove that the reversal of every language recognized by a finite deterministic strict real-time multitape Turing machine has a parsing expression grammar. The construction represents each work tape by two persistent stacks. One newly appended scaffold node stores the simultaneous update of every stack root, using one named backward port per stack and distance two. This gives the containment

$$\text{rev}(\text{RT-MTTM}) \subsetneq \text{PEG}.$$

Properness follows from the online lower bound proved by Loff–Moreira–Reis. Combining the transfer with Slisenko’s real-time palindrome recognizer and the single-head simulation of Leong and Seiferas yields

$$\{ww^R : w \in \{0, 1\}^*\} \in \text{PEG},$$

as a first major corollary, disproving Conjecture 7 of Loff–Moreira–Reis. The new machine-to-scaffold simulation has been formalized in Lean 4.

1 Introduction

Parsing expression grammars (PEGs), introduced by Ford [1], are recognition-based grammars whose ordered choice operator gives them behavior quite different from context-free grammars. Loff, Moreira, and Reis introduced *scaffolding automata* and proved that they characterize PEG languages up to reversal [4, Theorem 16]. They also proposed the following test of the formalism’s power:

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

the language of even-length binary palindromes. Their Conjecture 7 states that $\text{EPAL}_2 \notin \text{PEG}$.

The key observation of this paper is that one scaffold node can act as a persistent cons cell for several logical stacks at once. Its finite label is a tuple of stack-top symbols, and its named backward ports point independently to the persistent predecessor of each stack. Thus one physical append operation can perform a fixed tuple of logical memory updates.

This yields the main transfer theorem.

*ChatGPT, Claude, and Codex were used for conjecture exploration, source checking, Lean formalization, computational auditing, and editorial assistance. The author is responsible for the mathematical claims and the final manuscript.

Theorem 1.1 (Real-time transfer). *Let L be recognized by a finite deterministic strict real-time multitape Turing machine with a fixed positive number of ordinary single-head work tapes. The machine receives one input symbol and makes one global transition per step, has no input-length advice, and performs no post-input computation. Then*

$$L^R \in \text{PEG}.$$

Together with the online lower bound already proved by Loff–Moreira–Reis, the transfer gives a proper class containment:

$$\text{rev}(\text{RT-MTTM}) \subsetneq \text{PEG}.$$

The proof has two parts. First, a fixed collection of stacks, each performing one push, keep, or pop operation per input symbol, is simulated letter-for-letter by a distance-two scaffolding automaton. Second, the usual two-stack zipper represents each Turing tape. The resulting scaffold has degree $2m$ for an m -tape machine.

Slisenko constructs a real-time multitape palindrome recognizer [6]. Leong and Seiferas give an ordinary single-head multitape implementation, using twenty tapes [3]. A parity bit restricts the recognizer to even lengths. Since $\text{EPAL}_2^R = \text{EPAL}_2$, theorem 1.1 gives the announced consequence.

Corollary 1.2. $\text{EPAL}_2 \in \text{PEG}$. *Hence Conjecture 7 of Loff–Moreira–Reis is false.*

The historical direction is worth noting. Loff, Moreira, and Reis explicitly investigated whether reversals of PEG languages admit efficient online Turing-machine recognition and refuted that proposed containment [4, Section 5.3]. The construction here travels from strict real-time machines to scaffolding automata and then applies their characterization.

2 Models and conventions

For a word $x = x_1 \cdots x_n$, write $x^R = x_n \cdots x_1$, and for a language L , write $L^R = \{x^R : x \in L\}$. All alphabets and state sets below are finite.

2.1 Scaffolding automata

We use the model of Loff, Moreira, and Reis [4, Definitions 10–15]. A scaffold of degree d consists of a sequence of labelled nodes. Each node has d named ports, and every port is either missing or points to a node no later than its source. The scaffold begins with an unlabelled base node.

On each input symbol, the automaton appends exactly one labelled node. Its transition sees the state and the depth- k unfolding of the old top: a partial, labelled, d -ary tree that records labels along named port paths but not node identity. Each port of the new node may be left missing, may target the new node itself, or may target the endpoint of a port word of length at most k starting at the old top. The fixed constants d and k are the *degree* and *distance*.

Acceptance depends on the finite control state immediately after the last input symbol. The characterization we use is:

Theorem 2.1 (Loff–Moreira–Reis). *For every language K ,*

$$K \in \text{PEG} \iff K^R \text{ is decided by a scaffolding automaton.}$$

In the sufficient direction, their proof harmlessly duplicates states so that the initial state is used only before the first symbol [4, p. 23]. We will use this normalization when applying theorem 2.1.

2.2 Strict real-time multitape machines

We use the letter-synchronous real-time convention of Rosenberg [5]: after each arriving input symbol, the machine performs exactly one transition and the resulting state reports the answer for the prefix then seen.

Definition 2.2. A *strict real-time m -tape machine* M , where $m \geq 1$, has a finite input alphabet Σ , finite tape alphabet T with a blank symbol, finite control Q , and m ordinary two-way work tapes with one head per tape. Its transition function has the form

$$\delta : Q \times \Sigma \times T^m \longrightarrow Q \times (T \times \{-1, 0, +1\})^m.$$

At each step exactly one new input symbol is supplied, one global transition is made, and every work-tape head writes and moves by at most one cell. Acceptance of a prefix is determined by the resulting control state.

The input is an arriving symbol, not a preloaded input tape. In particular, the transition receives neither the total input length nor an end-of-input work interval.

3 Parallel persistent stacks

We first isolate the mechanism responsible for the transfer. Consider a deterministic finite-control machine with $s \geq 1$ stacks over a finite alphabet A . On every input step it reads the s top symbols and performs one of

$$\text{push}(a), \quad \text{keep}, \quad \text{pop}$$

on each stack. Popping an empty stack leaves it empty.

We construct a scaffolding automaton of degree s and distance two. Reserve port j for stack j , and take the scaffold alphabet to be

$$\Gamma = (A \cup \{\perp\})^s.$$

For a scaffold node v , let $\lambda_j(v)$ denote component j of its label and let $e_j(v)$ be its port- j target. An unlabelled or missing node has component $\perp$.

The stack rooted at v is decoded recursively by

$$D_j(v) = \begin{cases} \varepsilon, & \lambda_j(v) = \perp, \\ a D_j(e_j(v)), & \lambda_j(v) = a \in A, \end{cases} \quad (1)$$

with $D_j(\emptyset) = \varepsilon$. The recursion terminates because the construction uses only strictly backward targets.

3.1 The update table

Let v be the old scaffold top and v^+ the new one. Component j of v^+ and port j are selected as follows:

operation	new component $\lambda_j(v^+)$	new target $e_j(v^+)$
push(a)	a	v , named by ε
keep	$\lambda_j(v)$	$e_j(v)$, named by j
pop	$\lambda_j(e_j(v))$	$e_j(e_j(v))$, named by jj

Every label read and every target named by this table occurs in the depth-two unfolding of v . The same newly appended node carries all s component and port choices simultaneously.

The empty-stack case requires one invariant. Extend $\lambda_j(\emptyset) = \perp$, and define

$$\text{CleanEmpty}(S) \iff \forall v, j, \lambda_j(v) = \perp \implies \lambda_j(e_j(v)) = \perp. \quad (2)$$

Thus a node that terminates stack j points, if anywhere, to another node that also terminates stack j . This prevents an empty pop from exposing a stale symbol through a historical port.

Lemma 3.1 (Persistent-stack update). *Suppose the old scaffold is strictly backward and satisfies CleanEmpty. After appending the node specified by the table, the decoded stack at v^+ is exactly the result of the selected stack operation. The new scaffold is strictly backward and again satisfies CleanEmpty.*

Proof. Fix a stack j .

For a push, (1) gives

$$D_j(v^+) = aD_j(v).$$

For a keep with a nonempty stack, the new component copies $\lambda_j(v)$ and the new port skips directly to $e_j(v)$, so the same decoded list is obtained. If the stack is empty, the copied component is $\perp$, and the result is empty.

For a pop of a nonempty stack, write

$$D_j(v) = aD_j(e_j(v)).$$

The new component is the first component of this tail and the new target is its predecessor, hence $D_j(v^+) = D_j(e_j(v))$. If the old stack is empty, CleanEmpty implies $\lambda_j(e_j(v)) = \perp$, so the new decoded stack remains empty.

All three targets are named by port words and therefore lie at or below the old top; none names the new node. Strict backwardness follows. It remains to check CleanEmpty at the new node. The push case is vacuous. In the keep case it is inherited from v . In the pop case it is inherited one node deeper, from $e_j(v)$. Old nodes are unchanged. $\square$

Theorem 3.2 (Parallel persistent stacks). *Every deterministic letter-synchronous finite-control machine with a fixed positive number s of stacks, performing at most one push, keep, or pop per stack and per input symbol, is simulated exactly by a degree- s , distance-two scaffolding automaton.*

Proof. Interpret the unlabelled base as the empty root for every stack. It is strictly backward and satisfies CleanEmpty. The scaffold state is the stack machine's finite control. On each input symbol, the transition reads the old top components, computes the stack machine's next control and operations, and appends the tuple-valued node specified above.

Induction on the input length, using lemma 3.1, shows that the controls agree and that every decoded scaffold stack equals its source stack after every prefix. Therefore the accepting states agree. $\square$

4 From work tapes to persistent stacks

Represent work tape i by a zipper

$$(L_i, c_i, R_i),$$

where c_i is the scanned symbol, L_i lists the cells strictly to the left of the head in nearest-first order, and R_i lists the cells strictly to the right in nearest-first order. The scanned symbols $(c_1, \dots, c_m)$ are stored in finite control. Blank cells beyond the finite lists are supplied by the fixed blank convention.

Suppose the machine writes b_i and then moves its i -th head. The zipper updates are:

$$\begin{aligned} \text{right: } (L_i, c_i, R_i) &\mapsto (b_i L_i, \text{head}_{\sqcup}(R_i), \text{tail}(R_i)), \\ \text{left: } (L_i, c_i, R_i) &\mapsto (\text{tail}(L_i), \text{head}_{\sqcup}(L_i), b_i R_i), \\ \text{stay: } (L_i, c_i, R_i) &\mapsto (L_i, b_i, R_i). \end{aligned} \tag{3}$$

Consequently each ordinary tape transition performs at most one operation on each of the two stacks L_i, R_i . All m tapes update in parallel.

Lemma 4.1 (Tape zipper). *Every strict real-time m -tape machine is simulated letter-for-letter by a finite-control machine with $2m$ stacks, performing at most one push, keep, or pop on each stack per input symbol.*

Proof. Use the representation (3). The source transition depends only on the finite control, the arriving input symbol, and the m scanned symbols. The new scanned symbols are determined by the written symbols, move directions, and the old top symbols of the $2m$ stacks. Hence they can be placed in the next finite-control state. Equation (3) assigns one stack operation to each zipper half. A direct induction shows equality of the source and decoded tape contents after every prefix. $\square$

Theorem 4.2 (Strict machine-to-scaffold simulation). *Let M be a strict real-time m -tape machine, $m \geq 1$. There is a scaffolding automaton A_M of degree $2m$ and distance two such that*

$$L(A_M) = L(M).$$

After every input prefix, the two persistent stacks associated with each tape, together with its scanned symbol in finite control, reconstruct the source tape exactly. In particular, the reconstructed and source tapes agree at every integer offset from the head, including their implicit blank tails, and the source control states agree.

Proof. Compose lemma 4.1 with theorem 3.2. The product scaffold alphabet and augmented control are finite because m , the tape alphabet, and the source control are finite. Exactly one scaffold node is appended for each input symbol. $\square$

5 The real-time transfer

Theorem 5.1. *If L is recognized by a strict real-time multitape machine, then*

$$L^R \in \text{PEG}.$$

Equivalently,

$$\text{rev}(\text{RT-MTTM}) \subseteq \text{PEG}.$$

Proof. By theorem 4.2, a scaffolding automaton A decides L . If its initial state can recur, apply the state-duplication normalization used in the sufficient direction of theorem 2.1; this preserves the language. Put $K = L^R$. Then $K^R = L$ is decided by a scaffolding automaton, so theorem 2.1 gives $K \in \text{PEG}$. $\square$

Corollary 5.2 (Proper class containment).

$$\text{rev}(\text{RT-MTTM}) \subsetneq \text{PEG}.$$

Proof. Inclusion is theorem 5.1. Loff, Moreira, and Reis prove that there is a language $H \in \text{PEG}$ such that neither H nor H^R belongs to $\text{Online}(t(n))$ for any $t(n) = o(n/(\log n)^2)$ [4, Theorem 23]. Every strict real-time multitape machine from definition 2.2 is an $\text{Online}(O(1))$ machine after ignoring the latter model’s auxiliary input-length advice. Since $O(1) = o(n/(\log n)^2)$, $H^R \notin \text{RT-MTTM}$, and therefore $H \notin \text{rev}(\text{RT-MTTM})$. $\square$

Remark 5.3 (Construction size). The Loff–Moreira–Reis proof is constructive, so theorem 5.1 effectively yields a finite PEG. Direct expansion is impractical. For the twenty-tape implementation used below, the scaffold degree is 40, and a depth-two neighborhood has $1 + 40 + 40^2 = 1641$ positions. Even with a two-symbol tape alphabet, the tuple label set has at least 3^{40} elements; the ambient family of fully labelled depth-two neighborhoods alone has more than 10^{31000} members. This size is a property of the direct generic compiler, not of the language itself.

6 Even-length palindromes

Let

$$\text{PAL}_2 = \{x \in \{0, 1\}^* : x = x^R\}.$$

Slisenko gives a real-time multitape Turing-machine recognizer for PAL_2 [6]. His input convention supplies one new letter per machine tick, and the construction finds palindromic prefixes synchronously; the real-time palindrome theorem is stated at the end of the construction [6, pp. 132, 137, 139]. Galil later gave an independent real-time multitape palindrome algorithm [2].

Leong and Seiferas place this result in the ordinary single-head multitape model. They state that their real-time implementation of Slisenko’s recognizer uses twenty single-head tapes [3, p. 166]. Their technical setup applies Hartmanis–Stearns speedup to replace a fixed bounded number of composite cycles by one conventional machine step, using an enlarged finite tape alphabet and no time loss [3, p. 167]. Thus the recognizer fits definition 2.2.

Theorem 6.1. *The even-length binary palindrome language has a parsing expression grammar:*

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

Proof. Add one parity bit to the finite control of the real-time palindrome recognizer and accept precisely the palindromic prefixes of even length. This gives a strict real-time multitape machine for EPAL_2 . By theorem 5.1, $\text{EPAL}_2^R \in \text{PEG}$. Since $\text{EPAL}_2^R = \text{EPAL}_2$, the result follows. $\square$

Corollary 6.2. *Conjecture 7 of Loff, Moreira, and Reis [4] is false.*

7 Formal verification

The new simulation in sections 3 and 4 has been formalized in Lean 4. The development models the operational core of scaffolding automata: the unlabelled base, backward ports, partial tree unfoldings, distance-bounded port words, one-node append steps, runs, and final-state acceptance. It then proves:

- (i) preservation of `CleanEmpty`;
- (ii) correctness of each push, keep, and pop, including pop on an empty stack;
- (iii) equality of all source stacks and decoded scaffold stacks after every input word;
- (iv) equality of each source tape and the tape reconstructed from its two decoded scaffold roots and scanned symbol, including equality at every integer offset;
- (v) equality of the strict multitape machine’s language and the compiled scaffold automaton’s language.

The principal formal endpoints are:

- `realtime_transfer`: the structural induction relating the source and compiled runs;
- `realtime_transfer_tape`: equality of each source tape with the tape reconstructed from the scaffold;
- `realtime_transfer_tape_contents`: agreement at every signed cell offset;
- `realtime_transfer_wellFormed_decides`: distance-two typing, absence of self-targets, and language equality.

The development uses Lean 4.30.0 without Mathlib and contains no admitted proofs or declared axioms. Its theorem dependencies are Lean’s standard logical axioms. The operational formalization is polymorphic over its types; the finiteness hypotheses needed for a finite scaffolding automaton are imposed when the theorem is specialized to the finite source machines used in the scaffold-to-PEG composition.

The formalization stops at the scaffolding automaton. The applications of the published Loff–Moreira–Reis characterization and the classical real-time palindrome results are mathematical composition steps in this paper.

As a separate implementation check, a literal interpreter of the published scaffold transition model was tested with a transferred four-stack middle-symbol machine. Exhaustive tests through length 14, longer structured and random tests, and eight deliberately corrupted variants exercise the unfolding and port-word resolution independently of the Lean development.

8 Discussion

The transfer rests on a simple resource principle: the number of physical updates and the number of logical memory roots updated need not coincide. A scaffold node has a product label and a fixed tuple of named ports, so one append can advance every root in a fixed finite family. The two-stack zipper then converts local movement on each work tape into exactly this form.

This also explains why an explicit midpoint is unnecessary in the palindrome application. The scaffolding automaton does not discover the midpoint by a single stack turn. It simulates the distributed schedule already present in the classical real-time recognizer.

The degree is a fixed parameter of the scaffolding automaton, not a universal bound on the model. In the twenty-tape realization the construction uses degree 40 and distance two. No pointer-identity test is needed: the transition reads only labels along named paths in the published tree unfolding.

9 Conclusion

A fixed tuple of persistent stack roots can be updated by one appended scaffold node. Through the ordinary two-stack representation of a Turing tape, this gives an exact letter-synchronous simulation of strict real-time multitape computation. The Loff–Moreira–Reis characterization therefore places the reversal of every such language in PEG. Applied to the classical real-time palindrome recognizer, the transfer yields a PEG for even-length binary palindromes.

References

- [1] Bryan Ford. Parsing expression grammars: A recognition-based syntactic foundation. In *Proceedings of POPL 2004*, pages 111–122. ACM, 2004.
- [2] Zvi Galil. Palindrome recognition in real time by a multitape Turing machine. *Journal of Computer and System Sciences*, 16(2):140–157, 1978.
- [3] Benton L. Leong and Joel I. Seiferas. New real-time simulations of multihead tape units. *Journal of the ACM*, 28(1):166–180, 1981.
- [4] Bruno Loff, Nelma Moreira, and Rogério Reis. The computational power of parsing expression grammars. *Journal of Computer and System Sciences*, 111:58–72, 2020. <https://doi.org/10.1016/j.jcss.2020.01.001>.
- [5] Arnold L. Rosenberg. Real-time definable languages. *Journal of the ACM*, 14(4):645–662, 1967.
- [6] A. O. Slisenko. A simplified proof of the real-time recognizability of palindromes on Turing machines. *Zapiski Nauchnykh Seminarov LOMI*, 60:123–139, 1976. English translation in *Journal of Soviet Mathematics*, 15:68–77, 1981.