Token Linkages and Tree Resynchronization: An Effective Characterization

Wuming Pan
College of Computer Science, Sichuan University
Chengdu 610065, China
Correspondence: panwuming@scu.edu.cn

13 November 2026

Abstract

We characterize the relations realized by recognizable Token linkages, whose two legs are original heap-preserving, constructor-fixing Token maps. Over finite ranked signatures, these relations are exactly the tree-automatic relations with a uniform legal leaf-tile bound, and exactly the behaviors of linear tree bimorphisms with uniformly bounded canonical tile cost. The latter characterization is effective from a bimorphism presentation: boundedness is decidable, and in the positive case one can compute the least bound, a convolution automaton, and a linkage realization. The main resynchronization proof derives comparable output origins and bounded delay from a local subtree-size inequality, then encodes a source run by two copies of convolution positions with bounded-distance pointers. Deleting linear rules are handled by an effective preprocessing; the linearity hypothesis cannot be dropped. A direct finite-window construction gives elementary upper bounds. Thus bounded canonical cost defines a decidable subclass of linear bimorphisms on which equivalence is decidable, while the general equivalence problem is not. We also obtain canonical witnesses, an exponential gap between deterministic witness and canonical recognizers, effective closure bounds, and an EXPTIME-complete equivalence problem for explicit linkage presentations. Taking yield makes equivalence undecidable even for those presentations.

Keywords. Tree resynchronization; tree bimorphisms; tree-automatic relations; Token Space; origin geometry; effective realizability; behavioral equivalence.

1 Introduction and main theorem

A Token linkage

$$L = (X \xleftarrow{\ell} S \xrightarrow{r} Y)$$

is directed externally from records in X to records in Y. Its common domain S organizes structural witnesses; the input in X is the source of meaning. Both legs remain original Token maps. Its behavior is

$$\mathcal{R}_L = \{(\widehat{\ell}(s), \widehat{r}(s)) : s \in H_S\}, \qquad \mathrm{Out}_L(x) = \{\widehat{r}(s) : s \in H_S,\ \widehat{\ell}(s) = x\}. \tag{1}$$

The sequence-transduction role of Transformers motivates this distinction [2]; an endpoint relation is not an attention implementation or a learning algorithm. Finite-state witness representations allow structural implementations to be compared without identifying them with input/output tables or changing either original map.

A motivating distinction is the granularity at which context is represented. The whole-sequence carriers in [1] admit contextual numerical updates while heaps specify properties

such as causality. Here an endpoint record may retain individual symbols or smaller blocks as its leaves. Each leg still substitutes those leaves independently, but compatibility with the input selects whole witnesses from H_S. This permits contextual behavior without first treating the complete input sequence as one base element. Position labels alone do not supply the missing context; their effect depends on the chosen encoding and readout. Section 2.1 makes these limits explicit and distinguishes this relation representation from an implementation of a Transformer.

Our main question is when a broader tree bimorphism can be resynchronized into such a linkage. A linear tree homomorphism uses each source variable at most once; it is nondeleting if every variable is used. A bimorphism applies two such homomorphisms to the same recognizable central language. Its output relation may not have a recognizable padded convolution. A *legal tile* is a pair of nonempty endpoint trees, neither a bare core letter. A legal decomposition uses a common tuple-constructor skeleton, equal core leaves, and terminal tiles. The canonical cost $c(x, y)$ is the least possible largest tile size, zero when no tiles are needed, and infinite for a forbidden root core pair; precise definitions are in Definition 6 and Lemma 9.

Theorem 1 (Effective characterization and resynchronization)**.** *Fix finite literal tuple signatures, a finite core, maximum rank $r \geq 2$, and recognizable endpoint heaps. For $R \subseteq H_X \times H_Y$, the following are equivalent:*

(i) R is the behavior of a finite recognizable linkage of original Token maps;
(ii) $\mathrm{conv}(R)$ is recognizable and $\sup_{(x,y)\in R} c(x, y) < \infty$;
(iii) $R = \{(h_1(t), h_2(t)) : t \in C\}$ for a finite linear tree bimorphism whose canonical cost is uniformly bounded.

In (iii), deletion may be allowed or forbidden without changing the class. From a linear bimorphism presentation one can decide uniform boundedness; in the positive case one can compute a convolution automaton, the least canonical-cost bound, and a recognizable linkage realization. Consequently equivalence is decidable for bimorphism presentations in this decidable bounded-cost class.

Proof organization. The equivalence (i)⇔(ii) is proved in Theorem 7, and (i)⇒(iii) follows from the original-map syntax. The reverse implication and boundedness decision are Theorems 13 and 14; Lemma 26 and Corollary 27 remove the nondeletion assumption used in the origin argument. Equivalence then reduces to equality of the constructed convolution languages. □

This is a decidable class inside a larger undecidable equivalence problem: even nondeleting linear bimorphisms can encode arbitrary rational word relations by literal list trees (Proposition 32). The cost condition is imposed on endpoint pairs and is decidable from the broader presentation, rather than being an uncheckable promise about a chosen synchronization. Origins are an internal proof device; the final semantics forgets the center and all origin annotations.

The proof has three substantive steps. First, a legal tile bound compares subtree sizes at every address, including addresses absent on the other side. After finite-output states and jointly silent rules are eliminated, this forces comparable origins and zero net delay on every productive cycle. Second, bounded origin geometry places a source run inside two copies of the convolution domain with bounded-distance pointers. A finite local certificate therefore supplies a convolution recognizer. Third, a four-color forced-tile construction decides the remaining endpoint condition, including core guards, and computes the least bound. A loop test alone does not replace this last step.

The ambient notion of tree-automatic relation is classical [12, 13, 14, 17]. Origin semantics, word resynchronization, and finite-copy logical methods also have established theories; Section 9 makes these comparisons explicit. The result here connects the particular endpoint tile cost to effective resynchronization of linear bimorphisms. Further results distinguish canonical syntax

from compact structural witnesses and tree observation from yield. Context internalization and witness weights are recorded in Sections A and B so that they do not interrupt the main proof.

2 Original Token maps and syntax

Fix a core E, with all base/core unions tagged. A nonempty tuple tree is built from base letters, core letters, and ordered tuple constructors $\tau_n(t_1, \dots, t_n) = (t_1, \dots, t_n)$, $n \geq 0$. The nullary tuple $() = \tau_0$ is distinct from the empty forest ε. Write $\mathsf{Tr}(B \sqcup E)$ for the nonempty trees together with ε. Each child of a tuple is a nonempty tree.

Definition 2 (Original tree Token map)**.** A class is $X = \langle B_X, E, H_X \rangle$ with $H_X \subseteq \mathsf{Tr}(B_X \sqcup E)$. An original tree Token map $f : X \to Y$ is

$$f_0 : B_X \longrightarrow \mathsf{Tr}(B_Y \sqcup E) \setminus (E \cup \{\varepsilon\}), \qquad \widehat{f}(H_X) \subseteq H_Y. \tag{2}$$

The extension fixes E and ε and preserves every tuple constructor. Equality retains all generator values, including unused ones. These are the maps of TTS_E [1].

A linkage is a span in this category, considered up to apex isomorphism when forming the span category. In particular both legs preserve heaps. The substitution $b \mapsto ()$ sends (a, b, c) to $(a, (), c)$, not to (a, c). Flattening or deleting this position would be a separate operation; it is not part of (2).

We use two syntactic scopes. Context internalization in Section A applies to the full tuple syntax, including empty-forest records. In the main finite-state development, fix a finite maximum rank $r \geq 2$, finite bases and core, and the ranked alphabet

$$\Sigma_B = B \sqcup E \sqcup \{\tau_0, \dots, \tau_r\}.$$

Letters and τ_0 have rank zero; τ_n has rank n. Heaps in this finite-state fragment contain only nonempty trees, and all substitution values use this signature. This is a restriction of objects and maps, not a redefinition of a Token map. Unbounded-length sequences can be represented by binary list trees with τ_0 as terminator. Arbitrarily large flat tuple arities instead require an unranked automaton treatment, not supplied by a finite ranked alphabet without an encoding choice.

2.1 Record granularity, positions, and context

Locality for a fixed encoding. An original tree Token map fixes core leaves and every existing constructor, and replaces each base leaf by its assigned nonempty tree. If a tree t has base letter a at address p, then p remains an address of its image and

$$(\widehat{f}(t))|_p = f_0(a).$$

This follows immediately by induction along the address. Thus the image at that leaf depends only on the chosen base element, not on the other leaves of the record. Heap preservation restricts which substitutions are admitted; it does not change this evaluation rule. For flat word Token maps the substitution is letter-to-letter, so it also preserves word length exactly. These statements concern the literal output record, before any additional readout combining its parts.

There are three useful choices for encoding a position and an element.

(1) *Position supplied by order.* In a word $a_1 \cdots a_n$, the flat extension applies the same letter map at each position. In an ordered tree the existing constructor positions remain fixed, although a base leaf may expand into a subtree. The ordered syntax supplies addresses, not a position-dependent substitution rule.

(2) *Position supplied as a separate component.* For a record $\tau_2(i, a)$, the extension satisfies

$$\widehat{f}(\tau_2(i, a)) = \tau_2(\widehat{f}(i), f_0(a)).$$

A core position label or a constructor-only numeral $\tau_1^i(())$ is fixed. A base position label can change, but its substitution and that of a are separate. Changing i to $\pi(i)$ does not move the record to a different tree address; sorting records by the new labels would be an additional readout. If the label contains only i, the rule $i \mapsto n + 1 - i$ for reversal at arbitrary lengths lacks n. Encoding (n, i) removes that information deficit, but still does not itself reorder the literal tree.

(3) *Position incorporated into the base element.* A tagged atom $[i, a] \in \mathbb{N} \times A$ may have a substitution depending jointly on i and a. For instance, $[1, a] \mapsto 0$ and $[2, a] \mapsto 1$ distinguish two occurrences that a bare-letter map must treat identically. Such an atom still does not contain the other input values. An infinite position alphabet is outside the finite-alphabet hypotheses of our decision theorems, but need not lack a finite description: for $A = \{0, 1\}$ the rule $[i, a] \mapsto [i, a \oplus (i \bmod 2)]$ has a finite formula, with $\oplus$ denoting addition modulo two. A bounded position range is finite. Neither finiteness of a formula nor a larger alphabet by itself establishes the recognizable-heap hypotheses used below.

Consequently, there is no encoding-independent impossibility claim here. Packing a whole sequence or its context into one atom permits the carrier function to use that information. With separate position tags, the same atom $[1, a]$ in two records still receives the same substitution, however their other entries differ. The limitation is dependence on information outside the selected atom. A direct map is also single-valued and cannot merge or delete existing tree positions. It may nevertheless identify *values*, since its carrier substitution need not be injective. The nondecrease of tree node count is proved in Proposition 12; replacing a leaf by () is not deletion of its position.

Comparison with whole-sequence specifications. The example "Why the carrier cannot generally be one token vector" in [1] uses causal averaging: (a, b) produces $(a + b)/2$ at the second position, which is not determined by b alone. Its whole-sequence carriers place this dependence inside a function $F(H)$; permutation and prefix-agreement heaps specify properties of that function. This is a valid choice of granularity. That account also gives occurrence-level operation records, uses a common core across lengths, and derives prefix naturality from compatible dependency presentations. Its finite occurrence sets may have real-valued interpretations. Thus it already distinguishes structural computation from its numerical interpretation.

The present question is whether behavior can instead be represented by a finite recognizable linkage at a specified symbolic granularity. Endpoints can retain individual symbols, while witness letters can carry contextual annotations or finite blocks. This does not require them to be vocabulary tokens, nor does it automatically encode real-valued attention in a finite alphabet. A comparison with a concrete model must specify the admitted inputs, endpoint encodings and observations, construct the witness heap and legs, and prove the intended agreement of behavior. The finite-state results of this paper apply only when their representation hypotheses hold.

2.2 One input and two contextual readings

Let $a \in E$ and let c, e, d, f, g be base letters. Put

$$x = (a, c), \qquad y_1 = (a, (e, d, a)), \qquad y_2 = (a, (f, a, g)).$$

An apex with $B_S = \{u, v\}$ and $H_S = \{(a, u), (a, v)\}$ has original Token maps

	u	v
ℓ_0	c	c
r_0	(e, d, a)	(f, a, g)

fixing a. Take $H_X = \{x\}$ and $H_Y = \{y_1, y_2\}$. The same input has two witnesses and two outputs. A context may select the first witness, the second, or both. The letter e in this example is not core.

More generally, contexts are sets Γ_X, Γ_Y, and a transition mechanism is a relation

$$C_L \subseteq \Gamma_X \times H_S \times \Gamma_Y. \tag{3}$$

Its behavior contains $((\gamma, x), (\delta, y))$ exactly when some $s \in H_S$ has left image x, right image y, and $(\gamma, s, \delta) \in C_L$. A fixed-context admissibility predicate is the special case of diagonal transitions (γ, s, γ). The application chooses what contexts denote and how the relation is computed; the mathematical interface is explicit.

A single direct Token map need not implement the linkage's external behavior. For a fully explicit contextual example, let c, p, q, d, f be distinct base letters and require $(c, p) \mapsto (d, p)$ and $(c, q) \mapsto (f, q)$. A direct map would force both $f_0(c) = d$ and $f_0(c) = f$. Instead take witness letters u_p, u_q, v_p, v_q, heap

$$H_S = \{(u_p, v_p), (u_q, v_q)\},$$

	u_p	u_q	v_p	v_q
ℓ_0	c	c	p	q
r_0	d	f	p	q

and endpoint heaps $H_X = \{(c, p), (c, q)\}$ and $H_Y = \{(d, p), (f, q)\}$. Both legs are original Token maps. Compatibility with (c, p) selects (u_p, v_p), while compatibility with (c, q) selects (u_q, v_q). The apex heap couples the witness choices across positions; the two legs separately remain fixed substitutions. Merely tagging c with its position would leave the same first-position atom in both inputs.

For a given input, the left leg is used through its fiber $\{s \in H_S : \widehat{\ell}(s) = x\}$, not as an inverse function. The right leg then supplies the possible outputs. Different expansion on the two legs can produce shorter or longer endpoint records, subject to the size bounds in Proposition 12. Ordered tree addresses express structural position; the origin maps used later in the proof are not positional embeddings, attention weights, or causal masks. Neither a relation with several outputs nor its witness set specifies a sampling distribution or an autoregressive schedule. Weights require their own interpretation (Section B); feedback, stopping rules and numerical evaluation remain separate modeling choices.

3 Recognizable linkages as restricted tree bimorphisms

In the main development use the finite ranked, nonempty-tree fragment of Section 2. A recognizable heap is accepted by a finite bottom-up tree automaton. We use complete deterministic automata when quoting state counts; completion and determinization are effective but may increase the input size. Nondeterministic input presentations are also allowed where stated. All complexity bounds charge an explicit transition list and explicit substitution trees,

with fixed maximum rank. Standard closure and decision facts are recalled from [4, Chapter 1]. A *recognizable linkage* has recognizable interface and apex heaps, finite bases, and finite original-map substitution tables.

A tree bimorphism consists of a recognizable central language H and two tree homomorphisms, with relation $\{(h_1(s), h_2(s)) : s \in H\}$, in the classical setting of Arnold and Dauchet [3]. Our recognizable linkages are bimorphisms with additional endpoint heaps and particularly restricted legs:

$$h(\tau_n(z_1, \ldots, z_n)) = \tau_n(z_1, \ldots, z_n), \qquad h(e) = e,$$

while base constants may map to permitted ground trees. The legs are linear, nondeleting, and productive in standard tree-homomorphism terminology. Replacing a rank-zero base constant by τ_0 deletes no variable or subtree. We do not claim that this restriction has never been studied, or transfer every theorem about unrestricted bimorphisms to it without checking hypotheses. Each individual leg has a direct one-state deterministic top-down description: copy an internal constructor and recurse on its children, fix a core leaf, and emit the prescribed ground tree at a base leaf. The same recursion is evaluated bottom-up. The whole bimorphism also uses an inverse leg and a recognizable witness constraint, so this observation does not identify its behavior with an arbitrary deterministic transducer or with the full top-down/bottom-up classes.

3.1 Finite pullback alphabet and product-state recognition

For $r : S \to Y$ and $m : Q \to Y$, form the base

$$G = \{[s \mid t] : \widehat{r}(s) = \widehat{m}(t),\ s \in B_S \text{ or } t \in B_Q\}.$$

The other component of such a pair cannot be a bare core letter. Its projections are $\pi_S([s \mid t]) = s$ and $\pi_Q([s \mid t]) = t$. Synchronize matching tuples recursively, stopping at these base cases; identical core letters stay fixed. This gives the canonical heap bijection

$$H_P \cong \{(s,t) \in H_S \times H_Q : \widehat{r}(s) = \widehat{m}(t)\}, \qquad D(s,t) \longleftrightarrow (s,t). \tag{4}$$

Theorem 3 (Recognizable pullbacks and an explicit state bound)**.** *The above construction is the pullback in the finite ranked original-map category when equipped with*

$$H_P = \widehat{\pi}_S^{-1}(H_S) \cap \widehat{\pi}_Q^{-1}(H_Q). \tag{5}$$

If complete deterministic automata for H_S, H_Q have p, q states, an automaton for H_P has at most pq states. In particular recognizable linkages are closed under ordinary span composition.

Proof. Equality of two substituted tuple roots forces equal arity and matching children. Stopping whenever either component is a base letter gives a unique decomposition into a tree over $G \sqcup E$. Its inverse is the pair of projection extensions. A commuting cone has unique mediator $b \mapsto D(u_0(b), v_0(b))$; its values are permitted, and (5) ensures heap preservation.

For finiteness, the inverse image of any fixed target tree under a finite substitution is finite. Its preimages are the base letters mapped to it, the matching core letter if present, and tuples assembled recursively from child preimages. This recursion is effective. Each element of G has at least one component in a finite base, so G is finite.

Let the automata have states Q_S, Q_Q and evaluations $\mathrm{ev}_S, \mathrm{ev}_Q$. On G use states $Q_S \times Q_Q$ and the leaf transitions

$$[s \mid t] \longmapsto (\mathrm{ev}_S(s), \mathrm{ev}_Q(t)), \qquad e \longmapsto (\mathrm{ev}_S(e), \mathrm{ev}_Q(e)).$$

Each τ_n transition acts componentwise, including τ_0. The accepting states are $F_S \times F_Q$. Induction gives state $(\mathrm{ev}_S(\widehat{\pi}_S(w)), \mathrm{ev}_Q(\widehat{\pi}_Q(w)))$ on every input tree w, which is exactly (5). Composition retains the original endpoint heaps and original composed maps. □

For the full syntax used in Section A, add $D(\varepsilon, \varepsilon) = \varepsilon$ to the decomposition. The same universal-property proof applies; finiteness of alphabets is not needed for that set-theoretic statement. The automaton bound above is for the expressly stated nonempty finite ranked fragment.

This is a product-state bound after specifying the new alphabet. A complete transition table can have

$$|G| + |E| + \sum_{n=0}^{r} (pq)^n$$

entries. It is not a polynomial bound in the two state counts alone, nor a restatement of a bound on the number of explicitly listed heap records.

Example 4 (Small state set, large pullback alphabet)**.** Take $E = \varnothing$, a fixed binary tuple shape t_n with n leaves, and

$$B_S = \{b\}, \quad r_0(b) = t_n(y), \quad B_Q = \{u, v\}, \quad m_0(u) = m_0(v) = y.$$

Let $H_S = \{b\}$ and let H_Q contain all trees over $\{u, v\}$ and the fixed constructors. There are exactly 2^n generators $[b \mid t_n(w_1, \ldots, w_n)]$, $w_i \in \{u, v\}$, and all belong to H_P. Yet H_S has a two-state complete automaton and H_Q a one-state automaton; the product has only two states. Its alphabet and generator tables carry the exponential cost.

Finite heaps, including every finite table realization, are recognizable. Recognizability therefore provides an effective language representation, not a ban on lookup tables. Compression must be measured separately, charging the alphabet, transitions, and substitution values as well as the states.

4 An exact characterization of tree-observed behavior

For two finite trees x, y, their padded convolution $\mathrm{conv}(x, y)$ has the union of their position sets. Its label at a position is the pair of labels there, using a fresh padding mark $\perp$ where one tree has no node. A label (σ, ρ) has rank $\max(\mathrm{rk}\,\sigma, \mathrm{rk}\,\rho)$, where padding has rank zero; $(\perp, \perp)$ is excluded. The paired ranked alphabet is finite. We call a relation *tree-automatic* exactly when its valid convolution language is recognizable, using the standard convention [17, Section 2]. Not every tree over it is a valid convolution, but the encoding of pairs is injective.

Define a homomorphism j_L on the witness alphabet by

$$\begin{aligned} j_L(b) &= \mathrm{conv}(\ell_0(b), r_0(b)), \qquad j_L(e) = (e, e), \\ j_L(\tau_n(z_1, \ldots, z_n)) &= (\tau_n, \tau_n)(z_1, \ldots, z_n). \end{aligned} \tag{6}$$

This uses the shared internal constructors, not a general property of all pairs of tree homomorphisms.

Theorem 5 (Recognizable paired behavior)**.** *For a recognizable linkage,*

$$j_L(s) = \operatorname{conv}(\widehat{\ell}(s), \widehat{r}(s)), \qquad j_L(H_S) = \{\operatorname{conv}(x, y) : (x, y) \in \mathcal{R}_L\}. \tag{7}$$

If a bottom-up automaton for H_S has q states, the paired behavior has a nondeterministic bottom-up automaton with at most

$$q + \sum_{b \in B_S} (|\operatorname{conv}(\ell_0(b), r_0(b))| - 1) \tag{8}$$

states, where $|t|$ counts nodes. For fixed maximum rank and explicit substitution trees, the construction has polynomial size and running time.

Proof. Induction proves (7): both projections of an internal witness node have the same constructor and child positions. Thus j_L is a linear, nondeleting homomorphism, so the image is recognizable [4, Section 1.4]. Here is the explicit construction underlying the bound.

Retain each old automaton state. Replace a constructor transition $\tau_n(q_1, \ldots, q_n) \to q'$ by $(\tau_n, \tau_n)(q_1, \ldots, q_n) \to q'$, and a core transition by its paired core transition. For each leaf letter b, expand the ground tree $j_L(b)$ using one fresh state for each proper node occurrence, disjoint from old states and other macros. At the root allow every destination q of a source transition $b \to q$. All these destinations share the same proper-node states. This uses $|j_L(b)| - 1$ additional states before simplification; unused macros may be omitted. Since $|j_L(b)| \le |\ell_0(b)| + |r_0(b)|$, the state and transition descriptions are polynomial in the explicit input. An accepting run decomposes into old constructor steps and complete leaf macros, hence witnesses an accepted source tree; conversely every such source tree produces an accepting run. This proves the construction and (8).

□

4.1 The converse: bounded leaf tiles

Call a nonempty endpoint tree *admissible* if it is not a bare core letter; these are precisely the permitted generator images in this fragment. The following definition concerns pairs of endpoint trees, independently of any witness object or linkage.

Definition 6 (Bounded synchronous decomposition)**.** Fix $k \ge 1$. A pair (x, y) has a k-decomposition if it can be derived using the following three clauses, finitely many times:

(i) (e, e) is allowed for each $e \in E$;
(ii) from n allowed pairs (x_i, y_i), form $(\tau_n(x_1, \ldots, x_n), \tau_n(y_1, \ldots, y_n))$, including $n = 0$;
(iii) stop at a *tile* (u, v) where both trees are admissible and $\max(|u|, |v|) \le k$.

A relation is *uniformly leaf-tile synchronous* if one k works for every pair in it. The bound applies to the entire terminal tiles, not merely to a difference of heights or to a common-prefix length.

The decomposition is a shared constructor skeleton with bounded ground pairs at its leaves. It can be nonunique. Allowing tiles with a bare core on one side would change the original-map model; for example (e, b) cannot be a behavior pair when $e \in E$ and b is a base letter.

Theorem 7 (Behavioral characterization)**.** *Fix finite endpoint alphabets, a finite core, fixed maximum rank, and recognizable endpoint heaps. For $R \subseteq H_X \times H_Y$, the following are equivalent:*

(i) R is the behavior of a finite recognizable linkage of original maps;
(ii) $\operatorname{conv}(R)$ is recognizable and R is uniformly leaf-tile synchronous.

Given a recognizer of $\operatorname{conv}(R)$ and a valid bound k, a realization in (i) is effectively constructible. Thus this is a characterization inside the classical convolution-recognizable tree relations, not an identification with all such relations.

Proof. For (i)⇒(ii), Theorem 5 gives recognizability. Choose $k = \max(\{1\} \cup \{|\ell_0(b)|, |r_0(b)| : b \in B_S\})$. A witness tree supplies its own shared constructor skeleton; each base leaf contributes the tile $(\ell_0(b), r_0(b))$ and each core leaf contributes (e, e).

For the converse, let P_k be the finite set of all admissible pairs of endpoint trees of size at most k on each side. Make one fresh base letter $b_{u,v}$ for each $(u, v) \in P_k$, with original substitutions

$$\ell_0(b_{u,v}) = u, \qquad r_0(b_{u,v}) = v.$$

Let j_k be their joint homomorphism from (6) and set

$$H_S = j_k^{-1}(\mathrm{conv}(R)).$$

This heap is recognizable by inverse-homomorphism closure. Explicitly, determinize a recognizer for $\mathrm{conv}(R)$; assign $b_{u,v}$ the state obtained by evaluating $\mathrm{conv}(u, v)$, and use its diagonal constructor and core transitions. Every accepted witness has its two projections in R, so both legs preserve the stipulated heaps. Conversely, a k-decomposition of any $(x, y) \in R$ becomes a witness by replacing each terminal tile by its letter. Its joint image is $\mathrm{conv}(x, y)$; it is therefore in H_S. This proves equality of behaviors. Enumeration of P_k is finite; no small alphabet bound is claimed. □

The tile condition alone says $R \subseteq j_k(\mathsf{T}(\Sigma_{P_k}))$; that part is a syntactic representability requirement. The substantive converse is that recognizability of the specified subrelation makes its witnessing heap recognizable as well. The construction uses local tree pairs, not one generator per complete record, and can realize infinite relations. The next subsection also decides whether any such k exists.

Example 8 (A recognizable relation outside the class)**.** With empty core and base letters a, b, the relation $R = \{(a, \tau_1^n(b)) : n \geq 0\}$ has a recognizable convolution language: a finite automaton checks the initial label and the padded unary tail. For $n > 0$, a decomposition cannot start with a common constructor or a core, so the whole pair must be a tile of size at least $n + 1$. No uniform bound exists. Thus recognizability of a tree relation alone is insufficient. Even finite fibers and bounded size distortion do not suffice: the relation

$$R' = \{(\tau_1^n(a), \tau_2(\tau_1^{n-1}(b), ())) : n \geq 1\}$$

has a recognizable convolution, singleton fibers on both sides, and output size exactly one greater than input size. The unequal root constructors force each whole pair to be one tile, whose size is unbounded.

4.2 Deciding realizability and synthesizing the least bound

A pair is *core-obstructed* if one component is a bare core letter and the components are not the same core letter. Such a pair has no decomposition at its root. This obstruction can occur *inside* a permitted tile, so unconditional descent through matching constructors is incorrect. For example, $((e, a), (b, a))$, $e \in E$, is a permitted size-3 tile, but its child pair (e, b) is not decomposable. It must remain one tile.

For a pair not obstructed at its root, define a canonical cut recursively:

(i) A pair (e, e) contributes no tile.

(ii) If the roots are the same τ_n and no immediate child pair is core-obstructed, descend to all child pairs, including the empty case $n = 0$.

(iii) Otherwise retain the entire pair as one tile.

Let $\mathcal{F}_R$ be the language of convolutions of all tiles retained in these cuts for all pairs in R. It records occurrences as a set of subtree values, not their multiplicities.

Lemma 9 (Optimality of the canonical cut)**.** *For a root-unobstructed pair (x, y), let $c(x, y)$ be the largest value $\max(|u|, |v|)$ among its canonical tiles, with maximum 0 for no tiles. Then a k-decomposition exists exactly when $c(x, y) \leq k$.*

Proof. If the roots are not equal constructors and are not an equal core pair, the whole pair must be a tile. If a matching-constructor pair has a core-obstructed child, descent cannot give a decomposition, so again the whole pair is forced. In the remaining constructor case every child is root-unobstructed. By induction its canonical cut is optimal and has cost at most the child's maximum component size, strictly less than the parent's maximum component size. Descending is therefore no worse than using the parent as a tile. The equal-core and nullary cases have cost zero. These cases exhaust the decomposition clauses. □

Set $c(x, y) = \infty$ for a pair core-obstructed at the root. The canonical cut is the finest permitted skeleton. Other cuts can attain the same global maximum; uniqueness of all optimal presentations is not asserted.

Theorem 10 (Effective realizability and the least tile bound)**.** *Given a bottom-up automaton $\mathcal{A}$ for a relation's valid convolution language, finite endpoint alphabets and core, and fixed maximum rank $r \geq 2$, one can decide whether the relation is the behavior of a finite recognizable linkage of original maps. It is realizable if and only if*

(i) no accepted pair is core-obstructed at the root; and

(ii) the effectively recognizable forced-tile language $\mathcal{F}_R$ is finite.

In the positive case the least bound is

$$k_{\min} = \max(\{1\} \cup \{|u|, |v| : \mathrm{conv}(u, v) \in \mathcal{F}_R\}). \tag{9}$$

If $\mathcal{A}$ has q states and m explicit transitions, an automaton for $\mathcal{F}_R$ uses at most $4q$ states and $4^r m$ transitions. Realizability and $k_{\min}$ in binary are computable in polynomial time for fixed r. A finite linkage can then be synthesized effectively; no polynomial bound on its explicit alphabet or substitution tables is claimed.

Proof. Local annotation. Color each node of a paired tree with one of four symbols $\mathsf{B}, \mathsf{E}, \mathsf{S}, \mathsf{T}$. Color B means a padding component at the root or a core obstruction; E means an equal core pair. Otherwise use S exactly when the root label is (τ_n, τ_n) and no child has color B; all remaining cases get T. The color is determined by the root label and child colors. For valid convolutions at positions reached by descent, the two components are nonempty, and the colors respectively mean obstruction, core, split, tile.

Take the product $\mathcal{A}^+$ of $\mathcal{A}$ with this four-state annotator. It has the stated bounds; determinization is unnecessary. Compute its productive states P. If a productive accepting state has color B, the relation has a root obstruction and cannot be realized. Suppose not.

Reachable cut positions. Start a set D with the productive accepting states. Repeatedly, for every transition $\omega(d_1, \ldots, d_n) \to d$ with $d \in D$ of color S and all $d_i \in P$, add every d_i to D. A state belongs to D exactly when it can occur at a position reached by the canonical descent in an accepting run. One direction follows along the run; for the other, productive sibling trees fill the transition, and the already constructed accepting context supplies the prefix path. Replacing the subtree by any tree with that same product state preserves both acceptance and the ancestor split decisions. The color information is essential here.

Keeping all transitions of $\mathcal{A}^+$ and taking as final states $D \cap (Q \times \{\mathsf{T}\})$ consequently recognizes exactly $\mathcal{F}_R$. In particular the subtree *inside* a retained tile is not subject to further split restrictions. This proves effective recognizability.

Finiteness and optimization. Over a finite ranked alphabet, a uniform bound on both component sizes is equivalent to finiteness of their convolution set. By Lemma 9, conditions

(i)–(ii) are precisely the existence of a uniform decomposition bound. Apply Theorem 7. The least bound is (9), again by optimality.

For an explicit implementation, trim the forced-tile automaton to productive states that can occur in an accepting run. Its language is infinite exactly when its parent-to-child dependency graph has a directed cycle. A productive cycle can be iterated in an accepting context; an acyclic graph bounds height. This is the usual tree-language finiteness test. In the acyclic case compute, for each side $i = 1, 2$, the maximum accepted node count by dynamic programming:

$$M_i(d) = \max_{\omega(d_1,\dots,d_n)\to d} \left(w_i(\omega) + \sum_{j=1}^{n} M_i(d_j) \right), \qquad w_i(\omega) = \begin{cases} 0, & \text{component } i \text{ is } \bot, \\ 1, & \text{otherwise.} \end{cases}$$

Take maxima over final states and include 1. There are at most $4q$ useful states, so the acyclic trees have height at most $4q$ and their node counts have $O(q \log r)$ bits. Products, reachability, trimming, cycle detection, and this integer dynamic program are polynomial for fixed rank.

Synthesis. Enumerate the finite language $\mathcal{F}_R$ itself and introduce one base letter $b_{u,v}$ for each tile $\mathrm{conv}(u, v)$ in it, with leg values u, v. With the resulting joint homomorphism j, set $H_S = j^{-1}(\mathrm{conv}(R))$. This heap is recognizable by inverse-homomorphism closure. Replacing the canonical tiles of any pair in R by their letters gives a witness, while the inverse-image condition excludes all other pairs. Thus the synthesized linkage has behavior exactly R and attains $k_{\min}$. If there are no tiles, the base may be empty. This construction avoids enumerating tiles absent from the relation, but its explicit output can still be much larger than the decision procedure. It does not assert a minimum number of generators. □

The input is a recognizer of valid convolutions, not an arbitrary bimorphism with unknown behavior. Endpoint containment is also part of the specified relation. Valid-convolution syntax can be checked by local padding conditions; if interface containment must be checked against nondeterministic endpoint automata, that is a separate inclusion problem and is not covered by the polynomial-time claim. Computing $k_{\min}$ does not minimize the number of witness letters, automaton states, or total presentation size.

Corollary 11 (Decidability via recognizable tree relations)**.** Behavioral equivalence of two explicit recognizable linkages with common endpoint alphabets is decidable.

Proof. Their relations are equal exactly when the convolution languages constructed in Theorem 5 are equal. Apply classical finite-tree-language equality [4, Section 1.7]. □

The contribution here is the effective reduction and the characterization, not a new equality algorithm for recognizable tree languages. Equality concerns endpoint relations, not witness counts, context histories, or span isomorphism.

4.3 What size and length can change

For a finite original substitution f, set $M_f = \max(\{1\} \cup \{|f_0(b)| : b \in B\})$.

Proposition 12 (Size bounds and finite fibers)**.** *For every nonempty tree s,*

$$|s| \le |\widehat{f}(s)| \le M_f |s|.$$

Every fixed tree has finitely many substitution preimages, effectively computable. Consequently each input of a finite-base linkage has finitely many witness records and outputs, even if its recognizable heap is infinite. For $(x, y) \in \mathcal{R}_L$,

$$\frac{|x|}{M_\ell} \le |y| \le M_r |x|. \tag{10}$$

Proof. Every constructor and core occurrence contributes one node. Each base occurrence contributes $|f_0(b)| \geq 1$ nodes instead of one, giving both inequalities. Over a finite alphabet, only finitely many source trees have size at most a given target tree; enumeration or the preimage recursion in Theorem 3 computes the preimages. Applying the two size bounds to the common witness proves (10). □

Here $|()| = 1 = |b|$, so $b \mapsto ()$ does not reduce node count along a leg. This does *not* imply $|y| \geq |x|$ for the external linkage. For example, $H_S = \{u\}$, $\ell_0(u) = (p, q)$, and $r_0(u) = (z)$ give the shorter output $(p, q) \mapsto (z)$ using two original maps and appropriate singleton heaps. Setting $r_0(u) = ()$ even gives an empty-tuple output. The input can be more expanded than the output relative to the common witness. More precisely,

$$|\widehat{r}(s)| - |\widehat{\ell}(s)| = \sum_{b \in B_S} \mathrm{occ}_b(s)(|r_0(b)| - |\ell_0(b)|). \tag{11}$$

The actual restriction for a *fixed finite presentation* is bounded multiplicative node distortion (10) and finite fibers in both directions. Thus it cannot implement arbitrarily long inputs going to one fixed tree output, or one fixed tree input going to arbitrarily large outputs. This limits comparisons with unrestricted sequence transduction, but does not prohibit ordinary shortening. Flat tuple length, binary-list size, and flattened yield length are different observables. No projection leg is required just to obtain shorter outputs; a different observation changes the theory as below.

5 Tree resynchronization and decidable boundedness

Let $C \subseteq \mathsf{T}(\Omega)$ be recognizable over a finite ranked alphabet, and let h_1, h_2 initially be linear and nondeleting tree homomorphisms into the fixed finite endpoint signatures. Every source variable occurs exactly once in each defining context. Unary variable-only rules are allowed; no productivity assumption is implicit. The source constructors need not be endpoint tuple constructors. Put

$$R_B = \{(h_1(t), h_2(t)) : t \in C\}.$$

These broader homomorphisms are not being declared new Token maps. The goal is to construct a linkage whose legs satisfy the original definition.

Theorem 13 (Effective tree resynchronization)**.** *If* $\sup_{t \in C} c(h_1(t), h_2(t)) < \infty$*, then* $\mathrm{conv}(R_B)$ *is recognizable. A convolution automaton is effectively constructible from the finite bimorphism presentation under this promise; a numerical tile bound need not be supplied.*

Theorem 14 (Deciding uniform canonical cost)**.** *Uniform canonical-cost boundedness is decidable from the finite linear nondeleting bimorphism presentation. In the positive case one can compute a convolution automaton, the least bound, and a finite recognizable linkage realization. The least nonnegative canonical cost is zero when the forced-tile language is empty; the presentation bound* $k_{\min}$ *of* (9) *uses the convention* $k \geq 1$*.*

The proofs first construct the convolution automaton that Theorem 10 requires. Projected languages are recognizable by linear-image closure, so they may be used as interface heaps if these have not been fixed in advance. For the following arguments write $R = R_B$.

5.1 A local consequence of a legal tile bound

For an address p, put $\mathrm{sz}_p(t) = |t\restriction_p|$ when $p \in \mathrm{dom}(t)$ and put $\mathrm{sz}_p(t) = 0$ otherwise.

Lemma 15 (Local size comparison)**.** *If (x, y) has a legal k-decomposition and $K = \max(1, k)$, then for every $p \in \mathrm{dom}(x)$,*

$$\mathrm{sz}_p(x) \leq K \max(1, \mathrm{sz}_p(y)). \tag{12}$$

The symmetric statement also holds.

Proof. If p is inside a component of a terminal tile, the left side is at most k, so the inequality holds even if the other component has no node at p. Otherwise p lies in the common skeleton, and so does its counterpart. The two restricted subtrees have the same skeleton. Every tile occurrence adds between one and K nodes on each side; shared constructor and core nodes add one on each side. Summing gives the inequality. Equal core leaves and a decomposition with no tiles are included by $K \geq 1$. □

This lemma is stronger than a comparison of the sizes of the two entire trees. It is the local fact needed in both pumping arguments below.

5.2 Removing finite output behavior and joint silence

Represent C by a trimmed regular run grammar with productions

$$q \longrightarrow a(q_1, \ldots, q_m).$$

Every state is productive and occurs in an accepting run. If the original recognizer is nondeterministic, use transition-labelled run trees as the center alphabet and let the homomorphisms forget the run annotation. This does not change R. Write C_q for the center language with root state q.

Definition 16. A state q is *finite-output* if both $h_1(C_q)$ and $h_2(C_q)$ are finite. Otherwise it is *output-active*. This refers to output languages, not to the number of center trees.

Lemma 17 (Effective normalization)**.** *One can effectively express R as the union of a finite relation F and the behavior of a finite regular run grammar with linear, nondeleting macros such that:*

(i) every retained state is output-active;
(ii) no retained production has both macros equal to the single variable;
(iii) all retained states and productions occur in accepting derivations.

The active part may be empty.

Proof. Linear images of recognizable tree languages are effectively recognizable; their finiteness is decidable [4, Sections 1.4 and 1.7]. Thus the finite-output states are computable. For such a state compute the exact finite set

$$P_q = \{(h_1(s), h_2(s)) : s \in C_q\}.$$

For precision, enumerate the two finite marginal languages, and retain a candidate (u, v) exactly when

$$C_q \cap h_1^{-1}(\{u\}) \cap h_2^{-1}(\{v\}) \neq \varnothing.$$

Inverse homomorphic images and intersection are effectively recognizable, so these finitely many tests decide the exact joint set. Using the product of the marginals without this test would lose correlations.

For each production whose parent is active, choose a pair from P_{q_j} for each finite-output child. Substitute its two ground components into the respective output macros. Retain only the active child variables, in their source order, and relabel the production by its identity and these finite choices. The new symbol has the number of retained children as its rank.

Its macros remain linear and nondeleting. There are finitely many choices. Finite-output accepting roots contribute their finite joint sets to F.

These replacements preserve the pair language of every active state. One direction evaluates the removed subderivations. Conversely each selected pair has an actual q_j-derivation, and the regular grammar allows these child derivations to be chosen independently. Reinsert them to obtain an old run.

A production with both macros x must now be unary. Regard it as a unit production $q \to q'$. Compute the finite unit-reachability relation and replace unit paths followed by a nonunit production by a single production carrying the latter's two macros. This is ordinary unit-production elimination, with production identities retained in the new alphabet. It preserves all pair languages: every accepting finite run eventually leaves each consecutive unit path. Finally trim again. The surviving states still have their original infinite marginal on at least one side, and no production is jointly silent. □

Remark 18. Eliminating jointly silent *cycles* alone is not enough for the encoding below. We eliminate all jointly silent unary productions, including acyclic ones. Productions silent on just one side are retained. A nullary production always emits a nonempty ground tree on both sides.

5.3 Comparable origins and bounded delay

Work with the active grammar of Lemma 17. For a node v of a run tree, let $\rho_i(v)$ be the address of the root of its component output subtree in $h_i(t)$. Linearity and nondeletion give a unique such occurrence. If child j of a production has variable-hole address $p_{i,j}$ in macro i, then

$$\rho_i(vj) = \rho_i(v)p_{i,j}. \tag{13}$$

The addresses are independent of the child subtree substituted at v.

Lemma 19 (Active crossing is impossible)**.** *Under a uniform legal tile bound, $\rho_1(v)$ and $\rho_2(v)$ are comparable in the prefix order for every node of every accepted active run tree.*

Proof. Suppose an occurrence of state q has incomparable addresses p_1, p_2. Fix its outer run context and all other subtrees. Any $s \in C_q$ can replace the subtree there. In component 2, replacement affects only the subtree at p_2, so its subtree at p_1 is fixed or absent. In component 1, the subtree at p_1 is $h_1(s)$. Lemma 15 uniformly bounds $|h_1(s)|$. Interchanging the components uniformly bounds $|h_2(s)|$. Finite ranked alphabets have only finitely many trees of bounded size, so q is finite-output, contrary to the normalization. □

Make a finite dependency multigraph on the retained states. For each production $q \to a(q_1, \dots, q_m)$ and child j, add an edge $q \to q_j$ labelled $(p_{1,j}, p_{2,j})$ and weight

$$w = |p_{2,j}| - |p_{1,j}|.$$

Let M be the maximum depth of a hole in any macro, with maximum zero if there are no holes. For the algorithm a harmless safe bound is $D = (|Q| + 1)M$.

Lemma 20 (Zero cycles and bounded delay)**.** *Under a uniform legal tile bound, every directed cycle of the trimmed dependency graph has weight zero. Consequently, every accepted active run satisfies*

$$\big||\rho_2(v)| - |\rho_1(v)|\big| \le D. \tag{14}$$

1id1:core:20261109-0001@v2 1id1.net

Proof. A cycle at q gives a one-hole source context P from q to q: choose fixed productive subtrees at the other children along the cycle. Choose an accepting outer context U for q and a fixed bottom tree $s \in C_q$. Every $U[P^m[s]]$ is accepted. Write u_i for the hole address in $h_i(P)$ and r_i for that in $h_i(U)$. The bottom occurrences have origins $r_i u_i^m$. Their depth difference is

$$|r_2| - |r_1| + m(|u_2| - |u_1|).$$

If the cycle weight is nonzero, its absolute value tends to infinity. By Lemma 19, these two addresses are comparable for every m. For all sufficiently large m their ordering is fixed. Suppose the first address is shorter. At this address the first output subtree is the fixed tree $h_1(s)$; the second output subtree contains a path to the deeper bottom origin, whose length tends to infinity. The symmetric inequality in Lemma 15 is violated. The other ordering is identical.

If every cycle has weight zero, remove repeated-state subpaths from any root-to-node dependency path without changing its weight. The remaining simple path has at most $|Q| - 1$ edges, each of absolute weight at most M. Origins start at ε in both components, so this path weight is exactly their depth difference. The stated safe bound follows. □

Lemma 21 (The two structural tests are effective)**.** *It is decidable whether every dependency cycle has weight zero. If so, it is decidable whether all node-origin pairs in all accepted active runs are comparable.*

Proof. In each strongly connected component, test whether edge weights admit a consistent potential $w(q \to q') = \phi(q') - \phi(q)$. This is equivalent to zero total weight on every directed cycle. The bound D of Lemma 20 then holds on all paths, independently of a tile bound.

To test comparability, start at each accepting state with two empty address words. At a graph edge append its two hole words and cancel their common prefix. While comparable, at most one residual word is nonempty and its length is at most D. There are finitely many residual configurations over the finite output child-index alphabet. If cancellation leaves different first indices on both sides, a crossing has occurred; extensions cannot undo it. Test reachability of this condition. Each graph path can occur in an accepting run, by trimming and the independent choice of productive siblings. Thus the test is exact. □

Failure of either test certifies the absence of a uniform tile bound, by the preceding necessary lemmas. Passing both tests will provide a convolution automaton, but does not yet certify a tile bound. In particular core guards and unbounded constructor mismatches must still be tested.

5.4 A finite-copy MSO construction

This section supplies a sufficient regularity result about origins. It does not assume that the defining macros keep constructors fixed.

Lemma 22 (Bounded origin geometry implies tree-automaticity)**.** *Consider a finite regular run grammar with linear, nondeleting macros, no jointly silent unary production, and a known number D. Suppose that in all accepted runs the two origins of every source node are comparable and their depths differ by at most D. Then an automaton for its convolution behavior is effectively constructible.*

Proof. We describe a finite-copy monadic-second-order (MSO) certificate over a valid convolution $z = \mathrm{conv}(x, y)$. The effective equivalence of MSO-definable finite tree languages and recognizable tree languages supplies the automaton [4, Section 3.3]. All copies and pointer choices below range over fixed finite sets, so they are encoded by finitely many unary set variables, not by quantification over arbitrary binary relations.

Anchors and multiplicity. For each source node v, inspect its production. If macro 1 is not the single variable, anchor v at $\rho_1(v)$ on track 1. Otherwise anchor it at $\rho_2(v)$ on track 2. The latter macro is not a single variable, by the normalization. A macro which is not the single variable emits its root symbol. In a fixed output, distinct source occurrences cannot emit the same node: their emitted positions in a homomorphic expansion are disjoint. Hence there is at most one anchor per output track and address, and at most two source nodes anchored at a convolution address. The candidate source domain is therefore a subset of

$$\mathrm{dom}(z) \times \{1, 2\}.$$

Bounded pointers. The two origins are comparable and have depth gap at most D, so each lies within undirected tree distance D of the chosen anchor. If v' is a source child of v, equation (13) gives

$$\mathrm{dist}(a(v), a(v')) \le \mathrm{dist}(a(v), \rho_1(v)) + |p_{1,j}| + \mathrm{dist}(\rho_1(v'), a(v')) \le M + 2D.$$

Here $a(v)$ denotes the address part of its anchor, and M is the maximum hole depth of the current macros. The union of the two output domains is a tree, so these distances are measured in z. A pointer of radius B can be encoded by a word consisting of at most B upward/downward child steps and, for a source target, a track bit. The available words form a finite set. Their endpoints are first-order definable using the child relations of z. Nonexistent or ambiguous choices are rejected; using the unique simple tree path is one explicit convention.

The guessed source tree. Use finite unary labels on the two copies to guess the source domain, its production label and state, each ordered child pointer of radius $M + 2D$, and the two origin pointers of radius D. Enforce that the chosen anchor track agrees with the production's first nonidentity macro, and that the origin on this chosen track is the anchor itself. A source root is selected; both its origin pointers target the root of z. Check that each vertex has exactly the ordered children specified by its production, that these children are distinct, that the root has no parent and every other vertex exactly one, and that every vertex is reachable from the root. The last condition is MSO: every source-vertex subset containing the root and closed under the guessed child edges contains the whole source domain. A subset of the two copies is represented by two monadic sets. Thus this condition introduces no second-order binary relation. These constraints describe a finite ordered source tree. Check the grammar transition at each vertex and acceptance at the root.

Macro checks. For every source vertex v and each component i, require that its guessed origin belongs to component i of z. For each fixed symbol occurrence at address α in that production's macro, require the correct component symbol at $\rho_i(v)\alpha$. For the unique occurrence of variable x_j at hole address $p_{i,j}$, require

$$\rho_i(v_j) = \rho_i(v)p_{i,j}.$$

These are bounded-path formulas because there are only finitely many fixed macros. A variable-only macro requires equality of the parent and child origins and has no fixed-symbol check. All tests concern the actual component labels, so padding and rank violations are rejected.

Soundness. A satisfying certificate supplies a finite accepted source run t. Inducting from its leaves, the macro checks imply that the subtree of output i rooted at $\rho_i(v)$ equals $h_i(t|_v)$: all nonvariable positions are fixed and every variable hole is filled by its checked child subtree. Ranked labels prevent extra children outside the checked expansion. At the root both origins are ε, proving $x = h_1(t)$ and $y = h_2(t)$. Thus no spurious output pair is accepted.

Completeness and effectiveness. An actual accepted source run gives the anchors described above. They are injective within each track, and all its child and origin pointers fit the stated radii. Its labels and pointers satisfy the certificate. Consequently the MSO sentence defines exactly the convolution behavior. Enumerating the finite pointer and production labels and translating this sentence to a tree automaton is an effective construction. □

Remark 23. The explicit reachability condition is convenient but redundant here. Along every source edge, $|\rho_1| + |\rho_2|$ strictly increases: a nonidentity macro places all its holes at positive depth, and every retained production is nonidentity on at least one side. The macro equations therefore exclude cycles. Together with the parent constraints and finiteness, they already force a single rooted tree. Keeping the MSO reachability test makes the certificate's source-tree requirement explicit.

5.5 Putting the constructions together

Proof of Theorem 13. Normalize by Lemma 17, separating the finite relation F. Under the uniform-bound hypothesis, Lemmas 19 and 20 force both structural tests to pass. They supply a computable D for the normalized active part. Lemma 22 constructs its convolution automaton. Add the finite convolution language of F. No numerical tile bound was used to construct that automaton; its existence was used only to justify the structural tests. □

Proof of Theorem 14. Execute the following finite procedure.

1. Trim the run grammar and perform finite-output and joint-unit normalization. Keep the finite exceptional relation F.
2. Test every dependency cycle for zero weight. If this fails, answer no.
3. Using the computable D, test for an active crossing by the finite residual-word construction. If one is reachable, answer no.
4. Otherwise construct the active convolution automaton by Lemma 22, and union it with the finite convolution of F.
5. Apply the forced-tile/root-obstruction test of Theorem 10 to this automaton. Answer yes precisely when that test succeeds. In that case retain its least bound and its synthesized recognizable linkage; report canonical-cost bound zero if its forced-tile language is empty.

Early negative answers are sound by the two necessary geometric lemmas. If neither early test fails, the constructed automaton recognizes exactly R, so the final forced-tile test is both sound and complete. Each step terminates. Empty centers and a wholly finite relation are covered by taking the active part empty. A forbidden pair in F is rejected in the final test. □

5.6 A direct automaton and an elementary upper bound

Proposition 24 (A direct elementary construction)**.** *Let a normalized regular run grammar and two linear nondeleting macros satisfy the bounded-origin hypotheses of Lemma 22, with known bound D. Let $r \geq 2$ bound the output ranks, let M be the maximum hole depth, let L be the maximum depth of any defining macro, and set*

$$B = M + 2D, \qquad H = 2B + D + L + 3.$$

There is a direct finite-tree-automaton construction for the convolution behavior. If N is the total explicit size of the normalized grammar and its macro tables, including the output signature, and $D \leq (|Q| + 1)M$, the resulting nondeterministic automaton has at most $2^{2^{\mathrm{poly}(N)}}$ states and can be constructed within the same coarse time bound. No tightness claim is made.

Proof. Use the two-copy anchor certificate from Lemma 22. Decorate each convolution node with, on each of its two tracks, either "unused" or a production identity, two origin offsets of radius at most D, its ordered child pointers of radius at most B, and a source-root bit. Include an output-root bit. There are finitely many decorated symbols; denote their alphabet by $\mathcal{A}$.

The explicit reachability requirement can be removed. Along every source edge, $|\rho_1| + |\rho_2|$ strictly increases, because at least one macro emits a root and every variable then occurs at positive depth in that component. Thus the macro equations forbid source cycles. In a finite graph, one root with no incoming edge, one incoming edge at every other source vertex, and the specified distinct ordered children therefore force a single source tree.

All remaining certificate constraints are local within radius H in the underlying convolution tree. In particular, all possible incoming parents lie within radius B; their pointer targets, labels, and origin equations are inspected within the stated larger radius. Fixed macro positions need the full macro depth L, not just its hole depth M. Rank and padding conditions, child-state agreement, source-domain membership, pointer validity, and the anchor convention are local as well. The output-root bit is required to be false at every child root and true at the accepting root. Origin pointers of the selected source root must reach this marked global root. A prefix-cut boundary is never treated as an output root. Exactly one track at the output root is marked as the source root, and no other source vertex has that mark.

Here is an explicit bottom-up checker for these local conditions. A state stores the exact rank-aware decorated prefix of depth $2H$ of the current subtree and one Boolean flag. The flag records whether all centers at depth at least H have passed their neighborhood checks. From the child states and the parent label, form the parent's prefix, inherit the flags, and check the newly eligible centers at depth exactly H. Their radius-H neighborhoods are contained in that prefix. At the accepting root, also check all remaining centers at depth less than H, with the actual absence of ancestors above the root. Prefix descriptors distinguish nonexistent positions from positions omitted solely by the depth cutoff. This gives a deterministic automaton over $\mathcal{A}$.

Writing $S_d = 1 + r + \cdots + r^d$, the number of prefix-and-flag states is at most

$$2(|\mathcal{A}| + 1)^{S_{2H}}.$$

Erasing the certificate decorations yields a nondeterministic bottom-up automaton on the convolution alphabet with the same states. Reversing its transitions gives the usual nondeterministic top-down presentation, with the bottom-up accepting states as initial states. Thus no unrestricted MSO translation is required.

For the size estimate, a radius-s offset has at most $1 + (r+1) + \cdots + (r+1)^s$ possible walk words. A certificate label has at most N child pointers, two origin pointers, and one of at most N production identities on each track. Since $r, L, M \le N$ and $D = O(N^2)$, we have $H = O(N^2)$ and $\log|\mathcal{A}| \le \mathrm{poly}(N)$. Also $S_{2H} \le 2^{\mathrm{poly}(N)}$. Substitution in the displayed state bound proves the double-exponential bound. Enumerating the explicit transition table and evaluating the finite neighborhood tests stays within that bound. □

Remark 25 (Scope of the upper bound)**.** The nonelementary worst case of unrestricted MSO-to-automata translation does not give a nonelementary lower bound for this restricted certificate family. Proposition 24 gives an elementary upper bound for the normalized explicit representation. It does not identify the optimal complexity of the original bounded-cost decision problem, and it does not charge a compact substitution DAG as if it were an explicit tree. Nor does it bound the explicit tile alphabet produced by the subsequent canonical-linkage synthesis by the number of convolution-automaton states.

5.7 Deleting rules and the role of linearity

Lemma 26 (Eliminating observable deletion)**.** *Let G be a finite regular run grammar and let h_1, h_2 be linear tree homomorphisms, with deletion permitted. There is an effective procedure which either certifies that their behavior has no uniform legal tile bound, or constructs an equivalent finite linear, nondeleting bimorphism.*

Proof. Trim G, so every retained state and production is productive. For a production $\pi : q \to a(q_1, \ldots, q_m)$, let $I_i(\pi) \subseteq \{1, \ldots, m\}$ be the indices of the variables occurring in its ith output macro. Linearity means that each such variable occurs once. Let $Q^{\leftrightarrow}$ be the least set containing the accepting root states and closed under

$$q \in Q^{\leftrightarrow}, \quad \pi : q \to a(q_1, \ldots, q_m), \quad j \in I_1(\pi) \cap I_2(\pi) \quad \Longrightarrow \quad q_j \in Q^{\leftrightarrow}.$$

These are precisely the states which can occur at a source node whose output survives in both complete output trees. This characterization follows by following the unique variable occurrence on each side. Conversely every such dependency path extends to an accepting run by productivity of the siblings.

For every production with parent in $Q^{\leftrightarrow}$ and every child $j \in I_i(\pi) \setminus I_{3-i}(\pi)$, test whether the live marginal $h_i(C_{q_j})$ is finite. This is effective because linear homomorphic images of recognizable tree languages are recognizable, including for deleting homomorphisms. If one is infinite, fix an accepting outer run context for the parent, use this production, and fix all siblings. Replacing the q_j subtree by $s \in C_{q_j}$ leaves the opposite whole output unchanged. The live whole output contains $h_i(s)$ as a subtree, hence has unbounded size. The root instance of Lemma 15 contradicts every uniform tile bound. This is a sound negative answer.

Suppose all these tests give finite languages. In each relevant production, enumerate the possible live-side output trees for every one-sided child and substitute the selected ground tree into its sole occurrence. Delete all children absent from both macros, which require only that their states be productive. Retain exactly the children in $I_1(\pi) \cap I_2(\pi)$, ordered by their original source indices, and rename their variables consecutively. Use a new source symbol for the original production and the finite choices just made, of rank equal to the number of retained children. Its two output macros now contain every retained variable exactly once, so both are linear and nondeleting. Keep the original parent and retained-child state labels. There are finitely many productions and choices.

The pair language of each state in $Q^{\leftrightarrow}$ is preserved. An old run gives a new run by evaluating the one-sided child images and pruning the omitted children. Conversely, a new run can be expanded: each chosen live-side tree has an actual preimage in its prescribed child state, and a jointly deleted child has a productive derivation. These subtrees can be chosen independently in a regular run grammar. For a one-sided child its unobserved component is irrelevant; for a doubly retained child the same subderivation continues to supply both components. Induction on the finite run proves exact equality of the resulting pair languages, including at the accepting roots. No Cartesian approximation of a joint pair language is involved. The ordinary finite-output and joint-unit normalizations can now be applied to the new nondeleting presentation. □

Corollary 27 (Linear deletion is admissible)**.** Theorems 13 and 14 extend to arbitrary finite linear bimorphisms, allowing deleting homomorphisms. In the three-way characterization, "linear, nondeleting bimorphism" may therefore be replaced by "linear bimorphism".

Proof. Apply Lemma 26. Its negative branch correctly rejects the bounded-cost condition. Otherwise apply Theorems 13 and 14 to the equivalent presentation. Linearity is still essential: the nonlinear diagonal counterexample is unaffected by this preprocessing. □

Remark 28 (Missing origins versus failure of the theorem)**.** Here deletion means omission of source variables from a tree homomorphism macro; the value of every ground tree remains a nonempty ground tree. Deleting homomorphisms do not assign two origins to every source node, so the crossing and delay lemmas cannot be applied directly to that presentation. This is a limitation of the unnormalized proof, not a counterexample to resynchronization. Jointly invisible subtrees are removed; a one-sided subtree either has infinitely many live outputs, certifying failure of a uniform bound, or has finitely many, which can be inlined. After this preprocessing every retained subtree has both origins.

Corollary 29 (An elementary global upper bound)**.** For explicit finite grammar and macro tables and fixed maximum output rank, the bounded-cost decision procedure, its convolution-automaton construction, and equivalence on the positive bounded-cost instances have elementary upper bounds. The least canonical-cost bound is returned in binary. These are coarse upper bounds; no optimal tower height is asserted.

Proof. First remove epsilon transitions from the finite-tree automata used in normalization. If such an automaton has q states and accepts a finite language, every accepting run has at most q nodes on each root-to-leaf path, hence every accepted tree has address depth less than q: a repeated state on an accepting root-to-leaf run path would pump a nonempty tree context and give infinitely many accepted trees. For fixed rank r, its accepted trees therefore have at most $S_{q-1} = 1 + r + \cdots + r^{q-1}$ nodes. Over an alphabet Σ, the crude bound

$$(|\Sigma| + 1)^{S_{q-1}}$$

bounds the number of candidates in an exhaustive finite-language enumeration, by using an additional symbol for an absent position in the full rank-r shape. This enumeration and its membership tests have an elementary bound in the explicit automaton size.

Linear-image automata have polynomial-size regular-grammar constructions; expansion of their finite right-hand sides and epsilon elimination are elementary. Singleton inverse-image tests, finite products, and unit-rule elimination are elementary as well. The normalization used here performs two finite-output expansion stages: the finite live outputs in the deletion step, followed by the exact finite joint-output sets in the active-state step. Each stage enumerates finitely many finite languages and makes finite child choices in explicit productions. Consequently its output size is elementarily bounded in its input size. There is a fixed number of such stages, rather than an input-dependent iteration of expanding representations.

The zero-cycle test is polynomial. The residual-prefix exploration has at most $|Q|(1 + 2\sum_{j=0}^{D} r^j)$ configurations; with $D = O(N^2)$ for the normalized size, this is elementary as well. Composing these bounds with Proposition 24 gives a fixed-height exponential upper bound for the convolution construction. The forced-tile test and binary-size dynamic program preserve elementarity. Finally, equality of the two convolution languages adds the usual finite-tree-automaton determinization bound, which is another fixed exponential. None of these estimates uses unrestricted MSO translation. □

Proposition 30 (Linearity cannot simply be dropped)**.** *With nonlinear nondeleting legs, uniform canonical cost does not imply recognizable convolution, even when that cost is zero and the two legs are identical.*

Proof. Take the regular center $C = \{d(u^n(z)) : n \geq 0\}$ and let both homomorphisms be

$$h(d(x)) = \tau_2(x, x), \qquad h(u(x)) = \tau_1(x), \qquad h(z) = ().$$

The relation is the diagonal on $D = \{\tau_2(\tau_1^n(()), \tau_1^n(())) : n \geq 0\}$. Every pair has cost zero, since all nodes are shared tuple constructors. The language D is not recognizable: in any

deterministic bottom-up recognizer, two unary chains of different lengths reach the same state, so acceptance of the tree with two equal copies would also accept a tree with those two unequal chains. If the diagonal convolution were recognizable, relabeling each diagonal symbol to its component would make D recognizable, a contradiction. □

Example 31 (A core guard forces an unbounded parent tile)**.** Let $E = \{e\}$, let b be a base leaf, and take $C = \{s(u^n(z)) : n \geq 0\}$. Define

$$h_1(s(x)) = \tau_2(e, x), \quad h_2(s(x)) = \tau_2(b, x), \qquad h_i(u(x)) = \tau_1(x), \quad h_i(z) = \tau_0.$$

The pairs are

$$(\tau_2(e, \tau_1^n(\tau_0)),\ \tau_2(b, \tau_1^n(\tau_0))).$$

All source origins coincide, every delay cycle has weight zero, and the only unequal output symbols are the finite leaf pair (e, b). That pair is an illegal tile. It blocks splitting at its parent, which becomes one forced tile of size $n + 3$. Hence the canonical cost is unbounded. Replacing e by a base leaf makes the cost constantly one. Thus passing the two geometric tests does not by itself certify a tile bound. The final forced-tile test detects the core guard.

This is a standard observation; we record it to locate the decidable class.

Proposition 32 (An effective decidable class in an undecidable problem)**.** *Equivalence of finite linear nondeleting tree bimorphisms is undecidable, already for empty core and literal output tuple signatures of maximum rank two. On the class with uniform canonical-cost bound, membership is decidable and equivalence is decidable.*

Proof. For a word $a_1 \cdots a_n$ define the injective list encoding

$$\mathrm{list}(a_1 \cdots a_n) = \tau_2(a_1, \tau_2(a_2, \ldots, \tau_2(a_n, \tau_0) \ldots)), \qquad \mathrm{list}(\epsilon) = \tau_0.$$

For a word w, let $P_w(x)$ be the same prefix list with its final τ_0 replaced by x; in particular $P_\epsilon(x) = x$. The accepted paths of a finite transducer form a regular word language over transition identities. Regard these paths as unary source trees with a nullary terminator z. For a transition t labelled (u_t, v_t) put

$$h_1(t(x)) = P_{u_t}(x), \qquad h_2(t(x)) = P_{v_t}(x), \qquad h_1(z) = h_2(z) = \tau_0.$$

These are linear nondeleting homomorphisms, including empty transition labels. Their behavior is exactly the componentwise injective list encoding of the rational word relation. Equivalence of rational relations given by finite transducers is undecidable [6], so equivalence of these tree bimorphisms is undecidable. No yield is used in this reduction.

For bounded-cost presentations use Theorem 14 and Corollary 27 and compare the resulting convolution automata by classical tree-language equality. This also checks whether a presentation belongs to the decidable class before attempting that reduction. Its synthesis cost is distinct from the EXPTIME bound for already explicit linkage presentations in Theorem 39. □

6 Canonical witnesses and the cost of forgetting annotations

The forced-tile language determines a canonical syntax for a realizable relation. Minimization in this syntax is different from minimizing an arbitrary linkage presentation.

Definition 33 (Canonical witness language)**.** For a realizable relation R on the fixed interfaces, let

$$B_R = \{b_{u,v} : \mathrm{conv}(u, v) \in \mathcal{F}_R\}.$$

Give $b_{u,v}$ the two images u, v, and write j_R for the resulting joint homomorphism. The *canonical witness language* is

$$K_R = j_R^{-1}(\mathrm{conv}(R)) \subseteq \mathsf{T}(\Sigma_{B_R}).$$

The endpoints, core, constructors, and the names $b_{u,v}$ are fixed in this definition. An automaton recognizing K_R is a presentation of that language, not additional data in the definition of the relation.

Theorem 34 (Canonical representation)**.** *For every realizable R, the following statements hold.*

(i) j_R is injective on the entire witness term algebra. Its restriction $K_R \to \mathrm{conv}(R)$ is a bijection, whose inverse is the canonical cut with each retained tile replaced by its letter.

(ii) K_R is recognizable. Its reachable minimal complete deterministic bottom-up automaton is unique up to state renaming over Σ_{B_R}.

(iii) From a valid-convolution automaton for R, this canonical presentation is effectively constructible whenever the realizability test succeeds.

The resulting linkage has one witness for each behavior pair. It is a normal form for the endpoint relation, not for a span with its witness multiplicities.

Proof. Each forced tile is root-unobstructed and has color T in the annotator of Theorem 10: it is neither an equal core pair nor a pair through which canonical descent is allowed. A witness base letter therefore has a T-colored joint image, and a witness core letter has color E. At a witness constructor, every child has either two admissible projections or two identical core projections. No child is core-obstructed, so the joint image has color S, including the nullary case.

Consequently the image root determines whether the witness root is a tile letter, a core letter, or a constructor. In the tile case its two ground components determine the letter uniquely; in the constructor case the same argument applies recursively to every child. This proves injectivity and shows that the canonical cut recovers every witness in the term algebra. Conversely the canonical cut of every pair of R uses only tiles in $\mathcal{F}_R$, so it gives a witness in K_R. This proves (i).

Recognizability follows by inverse homomorphism. For completeness, define the context congruence on witness trees by

$$s \equiv_{K_R} t \quad \Longleftrightarrow \quad \text{for every one-hole context } C, \quad C[s] \in K_R \iff C[t] \in K_R.$$

It has finite index. Its classes give the reachable minimal complete bottom-up automaton; any such automaton has the same classes as states, up to renaming. These are the classical tree Myhill–Nerode and minimization results [4, Section 1.5], applied to this fixed canonical language. Finally, Theorem 10 constructs the finite tile set; inverse image, determinization, and minimization give (iii). □

In particular, equal relations with the same fixed interfaces have the same canonical tile alphabet and isomorphic minimal canonical automata. This does not minimize the alphabet, the number of states, or the total description size among presentations using other witness alphabets. Nor does it preserve the weighted behavior of Section B: a canonical witness represents a pair, whereas several original witnesses may represent that same pair.

Proposition 35 (Forced tiles occur inside generator images)**.** *For a finite recognizable linkage L realizing R, every tile in $\mathcal{F}_R$ occurs in the canonical cut of $(\ell_0(b), r_0(b))$ for some $b \in B_S$. In particular,*

$$|\mathcal{F}_R| \le \sum_{b \in B_S} |\operatorname{conv}(\ell_0(b), r_0(b))|.$$

The bound charges explicitly written substitution trees.

Proof. Canonical descent never stops at a constructor inherited from a witness: its child projections are admissible pairs or identical core pairs, as in Theorem 34. A retained tile is therefore inside the paired image of a witness base occurrence. Descent within that image is exactly its own canonical cut. The retained roots occupy disjoint positions in each finite paired image, so their number is at most its node count. Taking the union over all base letters proves the bound; unused generators can only enlarge it. □

Thus explicit generator tables bound the number of canonical tile values. The deterministic automaton of the canonical language can nevertheless grow exponentially, even when all tiles have size one.

Theorem 36 (Exponential witness compression)**.** *For each $n \ge 1$, there is a relation R_n with empty core and maximum rank 2 such that:*

(i) it has a recognizable linkage with three base letters, two identical letter-valued legs, and a complete deterministic apex automaton with $n + 5$ states, with exactly one accepted witness for each behavior pair;

(ii) every deterministic bottom-up automaton for its canonical witness language has at least 2^n reachable states.

The first presentation has an explicit description of polynomial size in n. The exponential gap concerns deterministic recognition after witness annotations are forgotten, not a lower bound for arbitrary nondeterministic presentations.

Proof. For $w \in \{0,1\}^*$ use the left-associated list encoding

$$\operatorname{enc}(\epsilon) = (), \qquad \operatorname{enc}(wa) = \tau_2(\operatorname{enc}(w), a).$$

Let $L_n = \{w : |w| \ge n \text{ and the } n\text{th symbol from the end is } 1\}$, and set

$$R_n = \{(\operatorname{enc}(w), \operatorname{enc}(w)) : w \in L_n\}.$$

Universal endpoint heaps suffice. In the witness base $\{0, 1, m\}$, both legs send m to 1 and fix $0, 1$. Choose the heap

$$H_n = \{\operatorname{enc}(w) : w \in \{0,1\}^* m \{0,1\}^{n-1}\}.$$

Here the same encoding is used over the enlarged leaf alphabet. Its projections are precisely the pairs of R_n: the marker chooses the required occurrence of 1. Its position is uniquely determined by the projected word, so each behavior pair has exactly one accepted witness. Both legs are original heap-preserving Token maps.

A word automaton for the marked language uses a state p before the marker, states $q_0, \dots, q_{n-1}$ counting the symbols after it, and a rejecting state d. On $0, 1$, state p loops; on m it goes to q_0. On $0, 1$, state q_i goes to q_{i+1} for $i < n - 1$; every further symbol from q_{n-1}, and every second marker, goes to d. State d is absorbing, and only q_{n-1} accepts. To obtain a complete bottom-up tree automaton, add three disjoint states $\lambda_0, \lambda_1, \lambda_m$ for the leaf letters. Give τ_0 state p and set $\tau_2(q, \lambda_a) \to \delta(q, a)$ for word states q; all other transitions, including unary constructors, go to d. This recognizes exactly H_n and has $n + 5$ states. Its complete rank-two transition table has $O(n^2)$ entries; the substitution tables are constant-size.

The canonical tiles of R_n are exactly $(0,0)$ and $(1,1)$, both of size one. Thus its canonical language is $\mathrm{enc}(L_n)$ with the two leaf letters renamed. For each $u \in \{0,1\}^n$ consider $\mathrm{enc}(u)$. If $u \neq v$, choose an index $i \in \{1,\dots,n\}$ with $u_i \neq v_i$. Appending $i-1$ zeroes is a one-hole left-list context, and

$$u0^{i-1} \in L_n \quad \Longleftrightarrow \quad u_i = 1.$$

It distinguishes $\mathrm{enc}(u)$ from $\mathrm{enc}(v)$. There are 2^n such pairwise context-distinguishable trees, proving (ii). □

This is the classical deterministic-state explosion for a marked-position language, realized with two unchanged Token maps. The witness heap stores an annotation that a deterministic recognizer can check with few states. Passing to the canonical endpoint representation removes that annotation. Hence S can compress a structural implementation even though meaning is assigned to the input in X. The theorem does not measure neural parameter counts or prove a learning or generalization advantage.

7 Closure properties and their limits

For fixed finite interface signatures and recognizable heaps write $\mathcal{L}(X,Y)$ for their finite recognizable linkage behaviors. Let $U_k(X,Y)$ consist of all pairs in $H_X \times H_Y$ admitting a legal k-decomposition. This universe includes core restrictions and fixes the tree encoding.

Theorem 37 (Effective closure)**.** *The classes $\mathcal{L}(X,Y)$ are effectively closed under finite unions, intersections, and differences; relational converse sends $\mathcal{L}(X,Y)$ to $\mathcal{L}(Y,X)$; and relational composition sends $\mathcal{L}(X,Y) \times \mathcal{L}(Y,Z)$ to $\mathcal{L}(X,Z)$. For each fixed k, $U_k(X,Y)$ belongs to $\mathcal{L}(X,Y)$, and its subrelations in this class are closed under complement relative to $U_k(X,Y)$. If R,S have legal tile bounds $k,h \geq 1$, respectively, then bounds for $R \cup S$, $R \cap S$, $R \setminus S$, and R^{-1} are, respectively,* $\max(k,h)$, $\min(k,h)$, k, *and* k. *When composable, $S \circ R$ has bound kh. These are valid bounds, not assertions that each is the least bound.*

Proof. Tree-automatic relations are effectively closed under Boolean operations and component exchange: use the corresponding operations on their convolution automata, with complement restricted to valid convolutions and the stipulated interfaces. Tile decomposability passes to subrelations, unions take the larger bound, and exchanging components preserves each tile size. The characterization of Theorem 7 therefore gives the first four constructions.

For composition, use Theorem 3. To obtain the displayed numerical bound, choose realizations of R,S whose generator images have size at most k,h, respectively, as in the characterization proof. A pullback generator $[s \mid t]$ has at least one base-letter component. If s is a base letter, its middle image has size at most k, whence $|t| \leq k$ by the nonshrinking substitution bound of Proposition 12. The composite left image has size at most k and the right image at most hk. If t is a base letter the symmetric argument gives the same product bound. The composite witnesses supply the required decompositions.

Finally, form the finite tile alphabet P_k from Theorem 7. The linear joint image of its entire witness term algebra is recognizable and consists exactly of the valid convolutions of k-decomposable pairs. Restricting to $H_X \times H_Y$ gives a recognizer for $U_k(X,Y)$. It has tile bound k, so it is realizable. Relative complement is then the difference construction already established. □

The automaton closure facts used here are classical; the inherited bounds explain why the constructor-fixing realizability restriction survives these operations. A finite tile universe does not mean that U_k is a finite relation: an unbounded shared skeleton is allowed.

Proposition 38 (Failure of unrestricted complement and iteration)**.** *Already for empty core and universal interfaces over a base letter a, $\mathcal{L}$ is not closed under complement in the entire interface product, or under transitive or reflexive-transitive closure.*

Proof. The full diagonal is realized by an identity linkage. Its complement contains $(a, \tau_1^m(a))$ for every $m \geq 1$, an infinite fiber forbidden by Proposition 12. Thus unrestricted complement need not be realizable, although its convolution is recognizable.

For iteration take

$$R = \{(\tau_1^n(a), \tau_1^{n+1}(a)) : n \geq 0\}.$$

Use one witness letter b, heap $\{\tau_1^n(b) : n \geq 0\}$, and the original substitutions $b \mapsto a$ and $b \mapsto \tau_1(a)$. This is a recognizable linkage with tile bound 2. Both R^+ and R^* contain $(a, \tau_1^m(a))$ for all $m \geq 1$, so neither is realizable. Here it is uniform tile boundedness that fails. The convolution of R^+ is recognizable by a unary-chain automaton; for R^* add the recognizable full diagonal. □

Operation on behaviors	Closure in the finite recognizable fragment
Finite union, intersection, difference	Yes
Converse and composition	Yes
Complement within $U_k(X, Y)$	Yes
Complement within $H_X \times H_Y$	Not in general
Transitive or reflexive-transitive closure	Not in general

8 Equivalence and observation boundaries

8.1 The exact complexity for explicit presentations

Theorem 39 (EXPTIME-completeness)**.** *Fix maximum rank $r \geq 2$. Behavioral equivalence of finite recognizable linkages, given by explicit bottom-up automata and explicit substitution trees, is EXPTIME-complete under polynomial-time many-one reductions. Hardness already holds with $E = \varnothing$, complete deterministic apex recognizers, and two identical letter-valued legs in each linkage. Endpoint heaps can be universal.*

Proof. For the upper bound, Theorem 5 constructs polynomial-size nondeterministic automata. Their equivalence is in EXPTIME [4, Section 1.7]. Inputs are promised to be valid original-map presentations. If validity is to be checked as well, image-language inclusion in the specified endpoint heaps also has an EXPTIME upper bound.

For hardness, start with the classical EXPTIME-hard equivalence problem for nondeterministic bottom-up tree automata. We may use alphabets with nullary and binary symbols: encode a ranked node by its symbol tag and a binary list of children. Expanding each explicit transition into a chain of transitions is polynomial, and the common tree encoding is injective, hence preserves language equality. Fix such a binary input alphabet $F = F_0 \sqcup F_2$. We give a polynomial construction from an automaton $A = (Q, F, \Delta, F_A)$ to a linkage with a deterministic apex recognizer.

Use a witness letter u_ρ for each leaf rule $\rho : a \to q$, and k_ρ for each binary rule $\rho : f(q_1, q_2) \to q$. A run tree is encoded recursively by u_ρ at a leaf and by $\tau_2(k_\rho, \tau_2(s_1, s_2))$ at a binary node. A complete deterministic automaton checks this encoding using disjoint states

$$Q \;\sqcup\; \{P_{p,q} : p, q \in Q\} \;\sqcup\; \{K_\rho : \rho \text{ binary}\} \;\sqcup\; \{\dagger\}.$$

Its non-rejecting rules are

$$u_{(a \to q)} \longmapsto q, \qquad k_\rho \longmapsto K_\rho,$$
$$\tau_2(p, q) \longmapsto P_{p,q}, \qquad \tau_2(K_\rho, P_{p,q}) \longmapsto v \quad (\rho : f(p, q) \to v).$$

1id1:core:20261109-0001@v2 1id1.net

All other transitions go to †, and only states in $F_A \subseteq Q$ are accepting. Tagged state families make these rules deterministic. There are $|Q| + |Q|^2 + |\Delta_2| + 1$ states; the complete table is polynomial for fixed rank. Its language is exactly the encodings of accepting runs.

In the common endpoint base take a fresh letter c_a for each $a \in F_0$ and d_f for each $f \in F_2$. Both legs are the letter substitution

$$h(u_{(a\to q)}) = c_a, \qquad h(k_{(f(p,q)\to v)}) = d_f.$$

They preserve universal endpoint heaps. With the injective encoding

$$C(a) = c_a, \qquad C(f(t_1, t_2)) = \tau_2(d_f, \tau_2(C(t_1), C(t_2))),$$

the linkage behavior is precisely $\{(C(t), C(t)) : t \in L(A)\}$. Apply the construction to both input automata. The two behaviors are equal if and only if their original languages are equal. This proves the lower bound even with deterministic apex presentations and without leaf expansion. □

Merely taking identity legs would establish this lower bound for *nondeterministic* apex input presentations. The run-letter construction is needed for the stronger deterministic-input statement: many witness letters collapse to the same observed letter and hide nondeterministic choices. The bound does not cover succinct substitution DAGs or grammar compression.

8.2 Flattened observation changes the decision problem

Take $E = \varnothing$. The yield of a tuple tree is the word of its base leaves in left-to-right order:

$$\mathrm{yd}(b) = b, \qquad \mathrm{yd}(\tau_n(t_1, \ldots, t_n)) = \mathrm{yd}(t_1) \cdots \mathrm{yd}(t_n), \qquad \mathrm{yd}(()) = \epsilon.$$

Define $\mathcal{R}_L^{\mathrm{yd}}$ by applying yield to both components of (1). Here ϵ is the empty *word*; this observation does not substitute an empty forest for a generator and does not alter either original map. Yield forgets constructors. It is not a nonlinear ranked-tree homomorphism in the sense of copying variables.

Theorem 40 (Undecidable equivalence after yield)**.** *Every rational word relation is the yield-observed behavior of a recognizable linkage of original maps, already with binary list-shaped witness heaps. Consequently equivalence of yield-observed behaviors is undecidable, even with finite alphabets, empty core, and complete deterministic apex recognizers.*

Proof. A finite-state transducer has finitely many transitions t, each labelled by a pair of words (u_t, v_t), where either word may be empty. Its accepted transition paths form a regular language over the finite alphabet of transition identities. Encode a path $t_1 \cdots t_n$ as the right-associated list

$$[t_1, \ldots, t_n] = \tau_2(b_{t_1}, \tau_2(b_{t_2}, \ldots \tau_2(b_{t_n}, ()) \ldots)).$$

The empty path is encoded by (). The language of accepting path lists is recognizable. It has a deterministic bottom-up recognizer: a suffix can record its starting state and whether its endpoint is accepting; transition identities supply their source and target states, so adjoining a head checks adjacency. Equivalently, use the standard effective regular-word-to-list-tree conversion and determinize.

Choose a binary list tree $W(w)$ for each word w, including $W(\epsilon) = ()$, and put $\ell_0(b_t) = W(u_t)$ and $r_0(b_t) = W(v_t)$. These are permitted nonempty trees, even for empty words. With universal endpoint heaps both legs are original maps. Taking yield of the two images concatenates exactly the transition labels of the path, so the resulting relation is exactly that of the transducer. The construction is effective. Equivalence of rational word relations given by finite transducers is classically undecidable [6]; a decision procedure for these linkage behaviors would decide it. The deterministic recognizer conversion is effective, which suffices for this undecidability reduction. □

The larger class with arbitrary recognizable tree-shaped witnesses need not consist only of rational relations. Decidable restrictions of yield observation remain meaningful; general rational relations already suffice for the negative result. After yield, even $b_t \mapsto ()$ can contribute an empty word, so unboundedly many tree witnesses can have one fixed observed word. The finite-tree-fiber argument cannot be carried through this quotient.

9 Position among established relation classes

We compare *tree-observed* behaviors on literal tuple signatures. A change of tree encoding or taking yield can change these comparisons.

Tree-automatic relations. A relation of finite trees is tree-automatic when its padded convolution language is recognizable. This is an established notion in automatic-structure theory [12, 13, 14]. For finite ranked alphabets our convention agrees with [17, Section 2]: the position set is the union of the component position sets, missing labels are padded, and a paired label has the maximum of the two ranks. Our characterization selects those tree-automatic relations satisfying the uniform admissible leaf-tile condition. It does not introduce tree-automaticity itself or a new general equivalence algorithm for that class.

Origins and resynchronization. Origins have an established role in transducer theory. Bojańczyk treats the source position of each output letter as part of the semantics of regular word functions [19]. For trees, Filiot, Maneth, Reynier, and Talbot show that retaining origins changes the decidability of several equivalence and definability problems [20]. Our endpoint relation forgets origins and center witnesses. Origins are used inside the proof to establish a finite convolution recognizer; equivalence remains equality of ordinary endpoint relations, not origin equivalence.

For words, Figueira and Libkin characterize synchronization classes by bounded shift, lag, and shiftlag, with effective canonical resynchronizations [21]. Descotte, Figueira, and Puppis decide containment of classes controlled by regular synchronization languages and construct relation-preserving resynchronizations when containment holds [22]. Thus controlling asynchronous computations by finite displacement information is an established idea. The canonical leaf-tile cost here is a condition on paired ranked output trees; it is not identified with any of those word parameters.

A closer tree comparison is Winter's decision procedure for a given bound on origin distance between top-down transductions [24]. That distance compares, for the same output node and the same input tree, two possible source origins. Our $\rho_1(v), \rho_2(v)$ instead compare the two output-root addresses induced by one center node v. We derive their comparability and a computable displacement bound from endpoint tile cost, after finite-output normalization. Winter's origin-close semantics and our origin-forgetting endpoint equivalence are different decision problems; bounded delay should not be used interchangeably for these notions.

Finite-copy logical descriptions also have a substantial history: Engelfriet and Maneth characterize MSO-definable tree translations using finite-copying macro tree transducers with regular look-ahead [23]. Our two-copy certificate uses that general logical perspective, but its bounded pointers are derived from the geometry of the two homomorphic outputs. The result established here is the effective passage from uniform canonical tile cost of a linear bimorphism to a convolution automaton, and its combination with the forced-tile test to decide bounded cost. It is not a claim to introduce origin semantics, finite-copy MSO methods, or synchronization itself.

Synchronized tree-tuple languages. The synchronized tree-tuple languages of Gouranton, Réty, and Seidl [15], and the logic-program treatment by Limet and Salzer [16], synchronize tuple derivations. This must be distinguished from positional synchronization by padded convolution. These grammar formalisms can express dependencies between branches; their synchronized language classes are not alternative names for tree-automatic relations. The comparison of [18, Section 1] explicitly distinguishes overlapping-code regular relations, rational tree relations, and synchronized language classes. We work with literal ranked-tree convolutions and the additional constructor-fixing tile restriction, without transferring results between these uses of synchronization on terminology alone.

Raoult's rational tree grammars. Raoult's multivariable productions synchronize the components of a tuple of nonterminals [7, Definition 2.5]. A recognizer for H_S gives such a grammar directly. Replace each state q by a pair nonterminal A_q. For $\tau_n(q_1, \ldots, q_n) \to q$ use

$$A_q \longrightarrow (\tau_n(X_1, \ldots, X_n), \tau_n(Y_1, \ldots, Y_n)), \qquad (X_i, Y_i) \text{ an instance of } A_{q_i}.$$

For a base transition $b \to q$ use $A_q \to (\ell_0(b), r_0(b))$; core and nullary constructors give their diagonal pairs. Accepting states supply the axioms, or a fresh pair axiom chooses one. Induction identifies its generated relation with the linkage behavior. The containment is strict on the literal signature: the relation R' in Example 8 is generated by

$$B \to (a, b) \mid (\tau_1(X), \tau_1(Y)), \quad A \to (\tau_1(X), \tau_2(Y, ())), \qquad (X, Y) \in B,$$

yet has no finite linkage realization. These are instances of Raoult's general grammar class, not a claim about every restricted subclass in that paper.

TATA ground tree transducers. Under TATA's Definition 5, a GTT uses two bottom-up automata to reduce subtrees to shared states beneath an arbitrary identical context [4, Section 3.2.1]. This literal class and our behaviors are incomparable: every such GTT accepts the full diagonal (use a context with no holes), whereas a linkage can realize just $\{(a, b)\}$, $a \neq b$. Conversely a one-state GTT accepting every tree on each side realizes the universal relation, which has infinite fibers and fails Proposition 12. Both comparisons use common universal endpoint alphabets. Variants with additional regular restrictions must be compared separately; we do not silently identify them with TATA's stated convention.

Word synchronization and other bimorphisms. Frougny–Sakarovitch synchronization and bounded length discrepancy concern word relations [8, 9]. They are not the same condition as bounding each terminal tree tile. For example, a list of n occurrences of u, with $\ell_0(u) = a$ and $r_0(u) = (a, a)$, has tile bound 3 but yields (a^n, a^{2n}). The word-length difference is unbounded and its padded word convolution is not regular. Thus tree synchronization here does not become word synchronization on taking yield.

Quasi-relabelings, formerly called quasi-alphabetic homomorphisms, permit one-level constructor patterns and send leaf letters to leaf letters [10, Definition 1]. Their behavior class also differs from ours, not merely their homomorphism syntax.

Proposition 41 (Separation from quasi-relabeling relations)**.** *On literal tuple signatures, already with empty core, recognizable linkage behaviors and quasi-relabeling relations are incomparable.*

Proof. For a quasi-relabeling h, the height bound $\mathrm{ht}(s) \leq \mathrm{ht}(h(s)) \leq \mathrm{ht}(s)+1$ is [10, Proposition 1]. Consequently the two outputs of any quasi-relabeling bimorphism differ in height by at most one. A singleton linkage with $\ell_0(u) = a$ and $r_0(u) = \tau_1(\tau_1(a))$ violates this bound.

Conversely, take the recognizable source language $\{s(t^n(z)) : n \geq 0\}$ with unary symbols s, t and leaf z. Let both homomorphisms send $t(x)$ to $\tau_1(x)$; send z to a on the left and b on the right; send $s(x)$ to $\tau_1(x)$ on the left and $\tau_2(x, c)$ on the right. These are quasi-relabelings, and their relation is

$$\{(\tau_1^{n+1}(a), \tau_2(\tau_1^n(b), c)) : n \geq 0\}.$$

Every pair has unequal root constructors. Its only possible linkage decomposition starts with the entire pair as a tile, of unbounded size as n grows. The characterization in Theorem 7 excludes it. □

These comparisons locate the precise constructor-fixing fragment. They do not exclude equivalence to another established presentation or a result elsewhere in the literature; bibliographic priority is not inferred from a search.

Remark 42 (Internal node labels through an explicit encoding)**.** Fixing tuple constructors is part of the original Token-map definition. For position-preserving relabeling of named constructors of equal rank, one may instead encode a rank-m node by

$$\mathrm{enc}(\sigma(t_1, \ldots, t_m)) = \tau_2(b_\sigma, \tau_m(\mathrm{enc}(t_1), \ldots, \mathrm{enc}(t_m))),$$

using fresh base tags b_σ; existing core leaves remain fixed. A witness tag $k_{\sigma,\rho}$ maps to b_σ on the left and b_ρ on the right. All tuple constructors are still fixed, so these are original Token maps. A recognizable language over paired node labels has a recognizable encoded witness heap by linear-image closure. The encoding is injective and has maximum rank $\max(2, d)$ when the named signature has maximum rank d. This is a specified change of records, not literal internal-constructor relabeling by an original map. The comparisons above continue to use their stated literal signatures; arbitrary child permutations are not supplied by this encoding.

10 Further questions and relation to Counting

The effective characterization now starts either from a convolution automaton or from a linear bimorphism. The canonical normal form solves minimization over its fixed syntax, while Theorem 36 separates this from optimization over arbitrary witnesses. Three further directions remain.

Q1. Minimal structured presentations. For a realizable relation R, minimize

$$|B_S| + \|\mathcal{A}_S\| + \sum_{b \in B_S} (|\ell_0(b)| + |r_0(b)|),$$

charging the full automaton description. For a realizable relation, exhaustive search through finite presentations combined with decidable equivalence gives an effective minimum under a fixed finite description encoding. What useful complexity bounds, approximations, or structural lower bounds avoid that search? How does $k_{\min}$, polynomial-time computable from a given convolution automaton, compare with this cost, including the potentially exponential alphabet generated by composition? Can the gap between the canonical and annotated presentations be bounded under additional restrictions on substitution tables or the recoverability of witness annotations? Witness uniqueness alone does not remove the gap in Theorem 36.

Q2. Alternative representations and observations. What are the tight bounds for DAG- or grammar-compressed substitution values, or for unranked tuples with hedge automata? Which restrictions on yield observation, ambiguity, or permitted transition paths restore decidable behavioral equivalence? Theorems 39 and 40 settle the explicit ranked case and the unrestricted yield case, respectively.

Q3. Tests versus representations. Counting studies hom-set observations $\hom(d, S)$ and the carrier budget $b(n)$ of separating tests [11]. Minimum automaton states instead measure recognition of a heap language. Under which finite-target, encoding, and size restrictions can one bound either invariant by the other, or design behavior-separating tests that retain witness multiplicities?

These quantities have different domains and equivalence notions. Infinite recognizable heaps may have infinite hom sets; quantitative comparisons need explicit finiteness or truncation assumptions. Recognizability does not exclude finite lookup tables, and publication priority for this restricted bimorphism class is not established by the characterization alone.

Verification supplement. The accompanying scripts audit finite instances of pullbacks, context, convolution, weighted and yield behavior, forced tiles, canonical witnesses, and closure bounds. The additional script checks the local subtree inequality, finite weighted dependency graphs, residual-prefix crossing detection, two-copy anchors, and examples of observable-deletion elimination. They are regression checks of finite constructions, not a proof of the general resynchronization theorems or an implementation of the full automaton synthesis. The mathematical arguments above supply those results. No neural-training experiment is used as evidence for the claims. OpenAI Codex assisted exploratory mathematical reasoning, drafting of proof arguments, and preparation and debugging of the finite verification programs. The recorded script outputs are finite computational checks; AI-generated suggestions and AI feedback are not independent mathematical verification or peer review.

A Context as an ordinary Token interface

This appendix also permits empty-forest records in the full tuple syntax. Put $\langle t \rangle = (t)$ for $t \neq \varepsilon$ and $\langle \varepsilon \rangle = ()$. This injective boxing commutes with original maps. For fresh base letters c_γ, encode an interface by

$$B_{X[\Gamma_X]} = B_X \sqcup \{c_\gamma : \gamma \in \Gamma_X\}, \qquad H_{X[\Gamma_X]} = \{(c_\gamma, \langle x \rangle) : \gamma \in \Gamma_X, x \in H_X\}.$$

This is a specified record encoding, not an asserted tensor product.

Theorem 43 (Context internalization)**.** *For the transition relation C_L of (3), take fresh base letters $k_{\gamma,\delta}$ and define*

$$H_{S_C} = \{(k_{\gamma,\delta}, \langle s \rangle) : (\gamma, s, \delta) \in C_L\},$$
$$\ell'_0(k_{\gamma,\delta}) = c_\gamma, \qquad r'_0(k_{\gamma,\delta}) = c_\delta.$$

Together with the old base and generator values, these are original maps whose linkage has exactly the encoded contextual behavior. Finite context sets and recognizable endpoint heaps and slices $\{s : (\gamma, s, \delta) \in C_L\}$ give recognizable encoded heaps.

Proof. A displayed witness maps to $(c_\gamma, \langle \widehat{\ell}(s) \rangle)$ and $(c_\delta, \langle \widehat{r}(s) \rangle)$, proving heap preservation and behavioral equality. Finite unions of prefixed, boxed slice languages are recognizable. □

No letter is erased. Prefixing a fresh letter also works for flat words; empty tuples are not essential. Arbitrary context predicates need not have recognizable slices.

For composable linkages $L = (X \leftarrow S \rightarrow Y)$ and $N = (Y \leftarrow Q \rightarrow Z)$, use the pullback records $D(s, t)$ of (4). Existential context composition is

$$(\gamma, D(s,t), \eta) \in C_{N \circ L} \iff \exists \delta : \ (\gamma, s, \delta) \in C_L \wedge (\delta, t, \eta) \in C_N. \tag{15}$$

Proposition 44 (Comparison under composition)**.** *Composing internalized linkages and internalizing* (15) *give the same endpoint relation. A commuting original map of their spans, surjective on heaps, need not be an isomorphism.*

Proof. Disjoint context and data alphabets separate the pullback generators. Send the context generator $k_{\gamma,\delta,\eta}$ to $k_{\gamma,\eta}$ and fix the data generators. Existential quantification gives surjectivity on heaps. Two middle contexts for one data witness yield two composite records but one existentially annotated record, proving possible noninjectivity. □

Intermediate paths therefore matter for counts and weights even when endpoint relations agree.

B Witness weights as tree series on paired trees

Let K be a commutative semiring and let $a : \mathsf{T}(\Sigma_S) \to K$ be a recognizable weighted-tree series, zero off H_S, with the standard finite-tree semantics [5]. Define, and extend by zero to invalid paired trees,

$$Z_L(\mathrm{conv}(x,y)) = \sum_{\substack{s\in H_S \\ \widehat{\ell}(s)=x,\ \widehat{r}(s)=y}} a(s). \tag{16}$$

The sum is finite by Proposition 12.

Theorem 45 (Weighted paired behavior and composition)**.** *The paired series* (16) *is recognizable. If a composite witness $D(s,t)$ has weight $a(s)b(t)$, this is a recognizable series on the pullback alphabet, and, suppressing convolution notation,*

$$Z_{N\circ L}(x,z) = \sum_{y} Z_L(x,y)Z_N(y,z). \tag{17}$$

Only finitely many terms contribute for a fixed input.

Proof. In the ground-macro construction of Theorem 5, put each leaf-transition weight on the macro root and weight one on its auxiliary transitions. Retain constructor weights. If source choices produce identical transitions, keep parallel transitions or add their weights. Runs then give exactly (16).

For a pullback generator g, evaluate its fixed image $\pi_S(g)$ in the first weighted automaton to obtain its leaf state vector; keep constructor transitions. Induction gives $a(\widehat{\pi}_S(w))$, and similarly on the other side. Their product is recognized by the tensor construction; commutativity permits grouping the weights. The heap bijection (4) partitions witnesses by the middle record and proves (17). □

For witness counts take $a = 1_{H_S}$ over $\mathbb{N}$, using a deterministic or unambiguous recognizer. Weight one on every transition of an arbitrary nondeterministic recognizer would count accepting runs instead of witnesses. These are series on paired trees; marginalizing over outputs requires its own encoding and summation convention.

Over $\mathbb{R}_{\geq 0}$, positive row totals permit probability normalization. Recognizable-series closure under this normalization is not established here. Normalization also need not commute with composition: one witness to each of y_1, y_2, followed by respectively one witness to z_1 and three to z_2, gives $1/4, 3/4$ after composite-count normalization but $1/2, 1/2$ after local normalization. Markov-kernel composition is therefore a separate probabilistic choice, not an automatic description of every weighted linkage. Context weights must sum over the intermediate records collapsed by Proposition 44.

Declarations

Author contributions. Wuming Pan: conceptualization, methodology, formal analysis, investigation, software, validation, writing—original draft, and writing—review and editing.

Funding. This research received no external funding.

Declaration of competing interest. The author declares no known competing financial interests or personal relationships that could have appeared to influence the work reported here.

Data and code availability. The seven finite verification programs and their recorded outputs are provided in `Token_Linkages_Verification_Supplement.zip`. They audit finite instances and examples, not the full synthesis algorithm or the general proofs. The preprint, typesetting source and verification material are available at the fixed release `https://1id1.net/id/core/20261109-0001/v2/`.

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process

During preparation of this work, the author used OpenAI Codex for manuscript drafting and revision, exploratory mathematical reasoning and proof drafting, reference lookup, and preparation and debugging of verification code. The author directed the work, reviewed and edited the resulting material, and takes responsibility for the arguments, evidence, interpretation and final text. AI feedback does not constitute independent peer review. The role and limits of the finite computational checks are described in the verification supplement paragraph above.

References

[1] W. Pan, *Token Space: A Category Theory Framework for AI Computations*, arXiv:2404.11624v4, revised 2 October 2026.

[2] A. Vaswani et al., Attention is all you need, *Advances in Neural Information Processing Systems* **30** (2017). arXiv:1706.03762.

[3] A. Arnold and M. Dauchet, Morphismes et bimorphismes d'arbres, *Theoretical Computer Science* **20**(1) (1982), 33–93. doi:10.1016/0304-3975(82)90098-6.

[4] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi, *Tree Automata Techniques and Applications*, October 14, 1999 edition, Chapters 1 and 3. Publicly available text.

[5] Z. Fülöp and H. Vogler, *Weighted Tree Automata—May it be a little more?*, 2022, revised 2026. arXiv:2212.05529.

[6] T. V. Griffiths, The unsolvability of the equivalence problem for Λ-free nondeterministic generalized machines, *Journal of the ACM* **15**(3) (1968), 409–413. doi:10.1145/321466.321473.

[7] J.-C. Raoult, Rational tree relations, *Bulletin of the Belgian Mathematical Society–Simon Stevin* **4**(1) (1997), 149–176. Journal full text.

[8] C. Frougny and J. Sakarovitch, Synchronized rational relations of finite and infinite words, *Theoretical Computer Science* **108**(1) (1993), 45–82. Publisher's record.

[9] C. Frougny and J. Sakarovitch, Number representation and finite automata, in V. Berthé and M. Rigo (eds.), *Combinatorics, Automata and Number Theory*, Cambridge University Press, 2010, 34–107; Section 2.6. Authors' chapter manuscript.

[10] A. Maletti and C. I. Tîrnăucă, Properties of quasi-relabeling tree bimorphisms, *International Journal of Foundations of Computer Science* **21**(3) (2010), 257–276. doi:10.1142/S0129054110007234; Authors' full text.

[11] W. Pan, *Counting in Token Space: Finite Witnesses, Linear Tests and Regular Heaps*, 1id1 preprint, v2, 9 November 2026 (manuscript V14). `1id1:core:20261109-0002@v2`. `https://1id1.net/id/core/20261109-0002/v2/`.

[12] A. Blumensath, *Automatic Structures*, Diploma thesis, RWTH Aachen University, 1999. Author's full text.

[13] A. Blumensath and E. Grädel, Automatic structures, *Proceedings of LICS 2000*, IEEE Computer Society, 2000, 51–62. Authors' full text.

[14] A. Blumensath and E. Grädel, Finite presentations of infinite structures: Automata and interpretations, *Theory of Computing Systems* **37** (2004), 641–674. doi:10.1007/s00224-004-1133-y.

[15] V. Gouranton, P. Réty, and H. Seidl, Synchronized tree languages revisited and new applications, in F. Honsell and M. Miculan (eds.), *Foundations of Software Science and Computation Structures*, FoSSaCS 2001, LNCS **2030**, Springer, 2001, 214–229. doi:10.1007/3-540-45315-6_14.

[16] S. Limet and G. Salzer, Tree tuple languages from the logic programming point of view, *Journal of Automated Reasoning* **37**(4) (2006), 323–349. doi:10.1007/s10817-006-9064-8.

[17] C. Löding and S. Winter, Synthesis of deterministic top-down tree transducers from automatic tree relations, *Information and Computation* **253** (2017), 336–354. doi:10.1016/j.ic.2016.07.013; Authors' full text.

[18] Y. Boichut, J. Chabin, and P. Réty, Towards more precise rewriting approximations, *Journal of Computer and System Sciences* **104** (2019), 131–148. doi:10.1016/j.jcss.2017.01.006.

[19] M. Bojańczyk, Transducers with origin information, in *Automata, Languages, and Programming*, ICALP 2014, LNCS **8573**, Springer, 2014, 26–37. doi:10.1007/978-3-662-43951-7_3; Author's preprint.

[20] E. Filiot, S. Maneth, P.-A. Reynier, and J.-M. Talbot, Decision problems of tree transducers with origin, *Information and Computation* **261**, Part 2 (2018), 311–335. doi:10.1016/j.ic.2018.02.011; Authors' manuscript.

[21] D. Figueira and L. Libkin, Synchronizing relations on words, *Theory of Computing Systems* **57**(2) (2015), 287–318. doi:10.1007/s00224-014-9584-2; Authors' manuscript.

[22] M. E. Descotte, D. Figueira, and G. Puppis, Resynchronizing classes of word relations, *ICALP 2018*, LIPIcs **107** (2018), 123:1–123:13. doi:10.4230/LIPIcs.ICALP.2018.123.

[23] J. Engelfriet and S. Maneth, Macro tree transducers, attribute grammars, and MSO definable tree translations, *Information and Computation* **154**(1) (1999), 34–91. doi:10.1006/inco.1999.2807.

[24] S. Winter, Decision problems for origin-close top-down tree transducers, *MFCS 2021*, LIPIcs **202** (2021), 90:1–90:16. doi:10.4230/LIPIcs.MFCS.2021.90; Full version.