Token Linkages: Effective Realizability and Decision Boundaries

Wuming Pan

Version 7 — November 9, 2026

Abstract

Token linkages use structural witnesses and two original heap-preserving tree Token maps. For finite ranked signatures, their recognizable behaviors admit an effective characterization: a convolution-recognizable relation is realizable exactly when it has no root core obstruction and its language of canonically forced leaf tiles is finite. A four-state annotation of the relation automaton constructs this language. With fixed maximum rank, realizability and the least uniform tile-size bound are computable in polynomial time; constructing a full realization need not have polynomial output size. The canonical decomposition requires a core check before descending through matching constructors. We retain recognizable pullback composition, EXPTIME-complete equivalence of explicit linkage presentations, and undecidable equivalence after yield. Context can be internalized, while weighted behavior records witness multiplicity. Comparisons with Raoult's rational tree grammars, TATA ground tree transducers, word synchronization, and quasi-relabeling relations locate the constructor-fixing fragment without claiming a new general transducer formalism. Node-size bounds permit shorter tree outputs but limit unbounded contraction in a fixed finite presentation.

Keywords. Token Space; context internalization; tree bimorphisms; recognizable tree relations; effective realizability; behavioral equivalence; yield.

1 Input meaning and structural implementation

A linkage

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

is directed externally from input records in X to output records in Y. Its common domain S organizes structural witnesses; it is not the source of meaning. For an input $x \in H_X$, its behavior is

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

Both legs are original Token maps. A context mechanism can restrict witnesses or specify a context update without changing that definition.

The goal is a structural model of sequence transformation, motivated by the sequence-transduction role of Transformers [2]. An input/output relation alone is not an attention implementation or a learning algorithm. Likewise, a table realization is one implementation, not a normal-form requirement on all linkages. We impose explicit finite-state hypotheses and distinguish tree-observed behavior from behavior observed after flattening. The former admits an effective realizability test and an EXPTIME-complete presentation-equivalence problem; the latter already contains the undecidable equivalence problem for rational word relations. Neither observation changes the two original maps.

Three distinctions are essential. First, () is a node, not an erasing substitution. Second, internalizing contextual behavior is weaker than an isomorphism of unrestricted span categories. Third, an automaton's state count is not its complete description size. The proofs below use these distinctions rather than assuming that context annotations, pullback generators, and intermediate paths can be identified without loss.

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 2.1 (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 4 applies to the full tuple syntax, including empty-forest records. From Section 5 onward, 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.

3 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 example, the required equations $(c, p) \mapsto (d, p)$ and $(c, q) \mapsto (f, q)$, $d \neq f$, would force two values for the same base letter c. A linkage can use two distinct witnesses whose left images both contain c. Context dependence is organized in the witnesses and their applicability, without changing either map.

4 Context as an ordinary Token interface

This section records the interface construction needed to carry context into the finite-state theory. Context does not change either original map. For a tree-or-empty-forest record put $\langle t \rangle = (t)$ when $t \neq \varepsilon$ and $\langle \varepsilon \rangle = ()$. This injective boxing commutes with every original map. With fresh context letters c_γ, define

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

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

Theorem 4.1 (Context internalization)**.** *For $C_L \subseteq \Gamma_X \times H_S \times \Gamma_Y$, use fresh letters $k_{\gamma,\delta}$ and set*

$$\begin{aligned} B_{S_C} &= B_S \sqcup \{k_{\gamma,\delta} : (\gamma, \delta) \in \Gamma_X \times \Gamma_Y\}, \\ H_{S_C} &= \{(k_{\gamma,\delta}, \langle s \rangle) : (\gamma, s, \delta) \in C_L\}, \\ \ell'_0(k_{\gamma,\delta}) &= c_\gamma, \qquad r'_0(k_{\gamma,\delta}) = c_\delta. \end{aligned}$$

Keeping the old generator values gives an ordinary linkage between the encoded interfaces with exactly the encoded contextual behavior. If contexts are finite and all slices $\{s : (\gamma, s, \delta) \in C_L\}$ and endpoint heaps are recognizable, so are the encoded heaps.

Proof. A displayed apex record has images $(c_\gamma, \langle \widehat{\ell}(s) \rangle)$ and $(c_\delta, \langle \widehat{r}(s) \rangle)$. These establish heap preservation and behavioral equality. Finite-state recognition follows by finite union of the boxed and prefixed slice languages. An arbitrary context predicate need not have recognizable slices. □

No context letter is erased. In the elementary word category the same construction prefixes $k_{\gamma,\delta}$ to the witness word and uses letter-valued projections; empty tuples are not essential for internalization.

For $N = (Y \leftarrow Q \rightarrow Z)$ let P be the tree pullback of Theorem 5.1, with its canonical heap bijection

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

Existential update composition declares

$$(\gamma, D(s, t), \eta) \in C_{N \circ L} \iff \exists \delta : \ (\gamma, s, \delta) \in C_L \wedge (\delta, t, \eta) \in C_N. \tag{5}$$

Proposition 4.2 (Comparison under composition)**.** *Composing the internalized linkages and internalizing (5) have the same endpoint relation. A commuting map of spans, surjective on heaps, sends $(k_{\gamma,\delta,\eta}, \langle D(s, t) \rangle)$ to $(k_{\gamma,\eta}, \langle D(s, t) \rangle)$. It need not be an isomorphism.*

Proof. Disjoint context/data alphabets separate pullback generators into context triples and data generators. Send a triple to its endpoint pair and keep the data generators fixed. This is an original map; existential quantification proves surjectivity. Two distinct middle contexts for one data witness give two composite records but only one existentially annotated record. □

Thus internalization does not identify span isomorphism classes with endpoint relations. Retaining intermediate paths matters for counts and weights. The finite-slice hypothesis is the connection with the recognizable fragment below.

5 Recognizable linkages as restricted tree bimorphisms

For the rest of the paper 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:

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

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.

5.1 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 map D in (4).

Theorem 5.1 (Recognizable pullbacks and an explicit state bound)**.** *The above construction is the pullback in the finite ranked original-map category when equipped with*

$$H_P = \widehat{\pi}_S^{-1}(H_S) \cap \widehat{\pi}_Q^{-1}(H_Q). \tag{6}$$

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.

1id1:core:20261109-0001@v1 1id1.net

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 (6) 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 (6). Composition retains the original endpoint heaps and original composed maps. □

For the full syntax used in Section 4, 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 5.2 (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.

5.2 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.

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 6.4 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 6.9. 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 5.3 (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 6.3 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.

6 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. 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

$$\begin{aligned} j_L(b) &= \mathrm{conv}(\ell_0(b), r_0(b)), \qquad j_L(e) = (e, e), \\ j_L(\tau_n(z_1, \ldots, z_n)) &= (\tau_n, \tau_n)(z_1, \ldots, z_n). \end{aligned} \tag{7}$$

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

Theorem 6.1 (Recognizable paired behavior)**.** *For a recognizable linkage,*

$$j_L(s) = \mathrm{conv}(\widehat{\ell}(s), \widehat{r}(s)), \qquad j_L(H_S) = \{\mathrm{conv}(x, y) : (x, y) \in \mathcal{R}_L\}. \tag{8}$$

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

$$q + \sum_{b \in B_S} (|\,\mathrm{conv}(\ell_0(b), r_0(b))| - 1) \tag{9}$$

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

Proof. Induction proves (8): 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 (9).

□

6.1 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 6.2 (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 $\max(|u|, |v|) \le k$.

1id1:core:20261109-0001@v1 1id1.net

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 6.3 (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) $\mathrm{conv}(R)$ *is recognizable and R is uniformly leaf-tile synchronous.*

Given a recognizer of $\mathrm{conv}(R)$ *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 6.1 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 (7) 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 6.4 (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.

6.2 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 6.5 (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 $c(x,y) \le k$.*

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. □

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 6.6 (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 $r \ge 2$, 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

$$k_{\min} = \max(\{1\} \cup \{|u|,|v| : \mathrm{conv}(u,v) \in \mathcal{F}_R\}). \tag{10}$$

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 6.5, conditions (i)–(ii) are precisely the existence of a uniform decomposition bound. Apply Theorem 6.3. The least bound is (10), 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:

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

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 6.7 (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 6.1 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.

6.3 The exact complexity for explicit presentations

Theorem 6.8 (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 6.1 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

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

All other transitions go to $\dagger$, 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.

1id1:core:20261109-0001@v1 1id1.net

6.4 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 6.9 (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 $(x, y) \in \mathcal{R}_L$,

$$\frac{|x|}{M_\ell} \le |y| \le M_r|x|. \tag{11}$$

Proof. Every constructor and core occurrence contributes one node. Each base occurrence contributes $|f_0(b)| \ge 1$ 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 5.1 computes the preimages. Applying the two size bounds to the common witness proves (11). □

Here $|()| = 1 = |b|$, so $b \mapsto ()$ does not reduce node count along a leg. This does *not* imply $|y| \ge |x|$ 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,

$$|\widehat{r}(s)| - |\widehat{\ell}(s)| = \sum_{b \in B_S} \mathrm{occ}_b(s)(|r_0(b)| - |\ell_0(b)|). \tag{12}$$

The actual restriction for a *fixed finite presentation* is bounded multiplicative node distortion (11) 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.

7 Witness weights as tree series on paired trees

Let K be a commutative semiring, for example $\mathbb{N}$ or $\mathbb{R}_{\ge 0}$. A weighted tree automaton assigns a finite state vector to each tree by summing products of transition weights, followed by final weights. We use the standard finite-tree semantics [5]. Suppose $a : \mathsf{T}(\Sigma_S) \to K$ is recognizable and vanishes off the apex heap. Define the paired series

$$Z_L(\mathrm{conv}(x, y)) = \sum_{\substack{s \in H_S \\ \widehat{\ell}(s)=x,\ \widehat{r}(s)=y}} a(s), \tag{13}$$

and zero on other paired trees. The sum is finite by Proposition 6.9; no complete-semiring or convergence assumption is needed for this pushforward.

Theorem 7.1 (Weighted paired behavior and composition)**.** *The series (13) is recognizable by a weighted tree automaton. For two weighted linkages, give a composite witness $D(s, t)$ the weight $a(s)b(t)$. This is a recognizable series on the pullback alphabet, and*

$$Z_{N \circ L}(x, z) = \sum_y Z_L(x, y) Z_N(y, z), \tag{14}$$

where the convolution notation on arguments is suppressed and only finitely many terms can contribute for a fixed input x.

Proof. Use the ground-macro construction of Theorem 6.1, now preserving weights. For each source leaf transition, put its weight on the macro root and weight one on auxiliary steps. Old constructor transitions keep their weights. Each target run corresponds to a source run together with its macro expansions. If distinct source choices yield identical target transitions, retain parallel transitions or add their weights; do not merge them Booleanly. This gives exactly the sum (13).

For inverse application along a projection, a pullback generator g receives the state vector obtained by evaluating its fixed image $\pi_S(g)$ in the first weighted automaton; constructors retain their transitions. Induction gives $a(\widehat{\pi}_S(w))$, and similarly for the other projection. Their product is recognized by the tensor-product state construction, using commutativity to group the weights. The support lies in (6). Finally the heap bijection (4) partitions composite witnesses by the middle record, yielding (14). □

Taking $a = 1_{H_S}$ over $\mathbb{N}$ gives the original witness count. Use a deterministic or otherwise unambiguous recognizer for this indicator: assigning weight one to all transitions of an arbitrary nondeterministic recognizer counts accepting runs, which need not equal the number of witness records. The resulting series lives on *paired trees*; it is not automatically a series on inputs alone without an output encoding and a summation convention.

Probability normalization is a separate step. If $0 < \sum_y Z_L(x, y) < \infty$, row normalization defines a kernel on that input. The finite-fiber result makes each sum finite for finite real weights, but positivity still requires an available witness of positive weight. Closure under row normalization is not established by the recognizable-series construction above, and normalizing composite counts need not equal composing separately normalized kernels. For instance, one first-stage witness to each of y_1, y_2 and then one witness to z_1 versus three to z_2 give respectively $1/4, 3/4$ after composite-count normalization, versus $1/2, 1/2$ after local normalization. A normalized conditional model can compose as a Markov kernel, but that fact alone does not make an arbitrary recognizable linkage a "Markov tree automaton." In context internalization, weights must also sum over the intermediate context records collapsed by Proposition 4.2.

8 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 8.1 (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. □

This establishes the unrestricted yield boundary, rather than leaving it as a conjecture. 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 Further questions and relation to Counting

The characterization now decides realizability and produces the least tile bound. It does not settle optimization of complete presentations or broader observation models. 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 the efficiently computable $k_{\min}$ compare with this cost, including the potentially exponential alphabet generated by composition?

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 6.8 and 8.1 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.

1id1:core:20261109-0001@v1 1id1.net

Verification. The scripts retain the earlier pullback, context, convolution, count, and yield checks. New checks compare the canonical cut against exhaustive decompositions, construct the four-state product and forced-tile automaton, and exercise empty, finite, infinite, and core-obstructed relations. They also check least bounds and reconstruction of a realizing linkage. Finite tests supplement the proofs; complexity and undecidability claims follow from the stated arguments.

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] E. Filiot, I. Jecker, C. Löding, and S. Winter, On equivalence and uniformisation problems for finite transducers, *ICALP 2016*, LIPIcs **55**, 125:1–125:14. Publisher's version.

[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, v1, 9 November 2026 (manuscript V13). `1id1:core:20261109-0002@v1`. `https://1id1.net/id/core/20261109-0002/v1/`.