A Myhill-Nerode Theorem beyond Trees and Forests via Finite Syntactic Categories Internal to Monoids

Tobias Heindel

Department of Computer Science, University of Leipzig, Leipzig, Germany

October 28, 2017

Abstract. The paper introduces recognizable languages as inverse images of sets of arrows from finite categories internal to monoids. The first result is the Myhill-Nerode Theorem as a conservative extension of the classic result for tree languages. The second result shows that a language of planar acyclic circuit diagrams whose gates have non-empty lists of input and output ports is recognizable if, and only if, it is accepted by an automaton in the sense of Bossut.
The proof of the Myhill-Nerode Theorem hinges on a suitable counterpart of the syntactic monoid, dubbed syntactic category, which is obtained by endowing the syntactic congruence of a recognizable language with the structure of a finite category internal to monoids. This opens up a venue for future research on Eilenberg type correspondences for pseudovarieties of finite categories internal to monoids.

Introduction

Circuit diagrams have been gaining popularity over the last decade in cross-disciplinary work between quantum physics, logic, and computation [BS10] in which they serve as a common graphical, yet formal syntax. In view of a whole line of research that extends formal language theory beyond words, trees, and forests [Boj15,AMU15,BW08,Jon96,TSG88], it is only natural to extend the language theorist's tool chest to cover "quantum syntax" [CK17], starting with the structural aspects of the most basic fragment: planar acyclic circuit diagrams. The latter have already been studied from an automata theoretic point of view [Bos86,BDW95]—in the guise of (embeddings of) planar directed acyclic graphs—while their intimate connection to semi-Thue systems—the very origin of formal language theory—was discovered and studied by Computer Science pioneer Günter Hotz [Hot68] in the '60s.

The specific, open problem that we set out to tackle is complementation for Bossut's automata model [Bos86, Chapter II.2.10], which we shall solve for a non-trivial class of circuit diagrams as a corollary of the two main results of the paper: since the generalized Myhill-Nerode Theorem (Theorem 1) ensures closure of recognizable languages under complements, coincidence of the classes of recognizable and automaton acceptable languages (Theorem 2) implies that automata can be complemented. Now, let us quickly review circuit diagrams via

rewriting diagrams in semi-Thue systems, Bossut's automaton model, and their mathematical structure for a better understanding of the open problem and the notion of recognizability that we shall study.

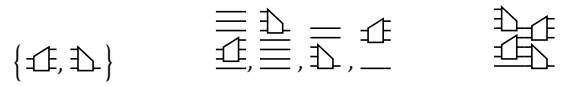


Fig. 1. A semi-Thue system over {1}, a rewriting sequence, and its rewriting diagram

Circuit diagrams, their automata, and their mathematical structure. A planar acyclic circuit diagram (PLACID) can be described as a *rewriting diagram* of a semi-Thue system, i.e., a string rewriting system, over the singleton alphabet {1}, by which we mean a sequence of rewriting steps up to switching of adjacent rewriting steps that act on non-overlapping parts of a word.[1] A rule of such a Semi-Thue system is essentially a pair of numbers and will be illustrated as a gate with m input ports on the left and n output ports on the right, as on the left in Figure 1; a word 1^n will be represented by n parallel wires, and a rewriting step is rendered as a gate with parallel wires above and below, as in the middle of Figure 1; finally, rewriting diagrams arise by joining matching wires in a rewriting sequence, as on the right in Figure 1.

Bossut's automata for unlabelled PLACIDS are semi-Thue systems over a general finite alphabet of *states* together with regular languages of initial and final *words*, resp.; their accepting runs are those that rewrite initial to final state words. Thus "transitions" of the automaton are gates whose input and output ports are labelled with states and a run is a circuit diagram in which each wire is labelled with a state. Finally, a PLACID belongs to the language of the automaton if it can be obtained from an accepting run by replacing all states by the letter 1, which amounts to removing all wire labels. For a given semi-Thue system it is an open problem whether the complement of the language of a given automaton is again automaton acceptable [Bos86, Chapter II.2.10].

Last but not least, PLACIDS have intuitive operations of sequential and parallel composition. These operations are associative and their respective identities are diagrams of parallel wires and the empty diagram; moreover they satisfy the so-called *interchange law*

$$\left(\begin{array}{c} m_1 \boxed{\mathcal{F}_1} n_1 \\ \oplus \\ m_2 \boxed{\mathcal{F}_2} n_2 \end{array}\right) ; \left(\begin{array}{c} n_1 \boxed{\mathcal{G}_1} k_1 \\ \oplus \\ n_2 \boxed{\mathcal{G}_2} k_2 \end{array}\right) = \begin{array}{c} m_1 \boxed{\mathcal{F}_1} n_1 \boxed{\mathcal{G}_1} k_1 \\ m_2 \boxed{\mathcal{F}_2} n_2 \boxed{\mathcal{G}_2} k_2 \end{array} = \left(\begin{array}{c} m_1 \boxed{\mathcal{F}_1} n_1 ; n_1 \boxed{\mathcal{G}_1} k_1 \\ \oplus \\ m_2 \boxed{\mathcal{F}_2} n_2 ; n_2 \boxed{\mathcal{G}_2} k_2 \end{array}\right)$$

[1] Note that we start by considering only unlabelled and uncoloured PLACIDS.

as illustrated in the above display where $\mathcal{F}_1, \mathcal{F}_2, \mathcal{G}_1, \mathcal{G}_2$ are circuit diagrams with the respective numbers of input and output ports. It was again Günter Hotz [Hot65] who discovered that these compositions endow PLACIDS with the mathematical structure that the contemporary category theorist would identify as categories internal to monoids with free monoids of objects that are freely generated by a set of gates. Removing the freeness requirements yields general categories internal to monoids (see Definition 1), which are to PLACIDS what monoids are to words.

Main Contributions. The paper introduces recognizability for languages of arrows in an arbitrary category internal to monoids as a conservative extension of recognizability for tree and forest languages, identifying trees with their natural PLACID counterpart. The first result is the Myhill-Nerode Theorem for a suitable notion of syntactic congruence such that a language is recognizable if, and only if, its syntactic congruence is of finite index. The second result shows that, under the assumption that all gates of PLACIDS have non-empty lists of input and output ports,[2] the recognizable languages coincide with the automaton acceptable languages, relying on Francis Bossut's automaton model. Finally, we apply the results to give a (partial) answer to Bossut's open problem of automata complementation [Bos86, Chapter II.2.10].

Related work. The paper uses a direct generalization of finite syntactic categories [TSG88] to finite syntactic categories internal to monoids. The idea to replace syntactic monoids by richer structures has been explored in the literature using several related approaches [AMU15,Boj15,ÉW05]. The two distinguishing features of the general theory that we shall develop in the present paper are the use of properly finite structures as language recognizers and a canonical way of treating tree and forest languages as languages of PLACIDS.

The relation to recognizable algebras of a monad [Boj15] is via the word *algebra*. However, in the present paper, the notion of algebra is the one for monoidal theories—replacing the product preserving functors of Lawvere theories by functors that preserve an abstract monoidal product. Changing the notion of algebra leads to subtle differences in how the language theory for trees and forests is embedded. In the present paper, trees are essentially preclones as in the work of Ésik and Weil [ÉW05]. The difference to forest algebras [BW08] is the richer structure of "sequential" composition of forests in a category. Finally, there are obvious connections to recognizable languages from a free monoid in a monoidal category [AMU15]; however, the automaton model that we shall consider for the present paper [Bos86,BDW95] does not work on objects of a

[2] The restriction to gates with input and output ports is exactly the one that is necessary to obtain a bijective correspondence of PLACIDS and their dual representation as planar embeddings of direct acyclic graphs with a unique source and a unique sink node: wires become edges and gates faces (a.k.a. regions or cells) such that the face of a gate lies between the two paths of its input and output wires and the source and target nodes of the two paths are the same. The whole paper could be written using categories internal to semi-groups instead of monoids, as discussed at the end of Section 4.

monoidal category, but on generators of a product category (see Section 3), which retains the syntactic character of automata.

Structure of the paper. The core of the paper is structured in two parts: first, we introduce recognizability abstractly and develop the Myhill-Nerode Theorem, before we consider the special case of automaton acceptable languages of PLACIDS and apply the Myhill-Nerode Theorem to give a partial answer to the open question of automata complementation [Bos86, Chapter II.2.10]. The preliminaries are split accordingly, deferring the rigorous formalization of circuit diagrams to Section 3, the starting point of the second part. We discuss directions for future work and its related work in a separate section before we conclude.

1 Background: monoids, trees, graphs, categories, functors

We first introduce notational conventions for mathematical structures that the reader should be familiar with (Section 1.1). Then we review categories (Definition 1) and functors (Definition 2) internal to monoids in Section 1.2. The latter concepts allow for a succinct definition of recognizable language in Section 2. No further knowledge of category theory is required.

1.1 Notational conventions

We introduce notation as we define a representative list of concepts that we shall use without explanation.

Sets, relations, and functions. The set of natural numbers, including 0, is denoted by $\mathbb{N}$. Let A, B be two arbitrary sets. The *Cartesian product* of these sets, denoted by $A \times B$, consists of all ordered pairs, i.e., $A \times B = \{(a, b) \mid a \in A, b \in B\}$. A *relation* between these sets is a subset $R \subseteq A \times B$ of the Cartesian product $A \times B$ of which the first factor A is the *domain* while the second one B is the *codomain* of the relation; $R\colon A \nrightarrow B$ stands for $R \subseteq A \times B$ and $a\ R\ b$ for $(a, b) \in R$. A relation $f\colon A \nrightarrow B$ is a *function* if each element of its domain A is related to exactly one element of the codomain B; we write $f\colon A \to B$ to express that f is a function. The *direct image* of a subset $X \subseteq A$ of the domain of a function is $f(X) = \{f(a) \mid a \in A\}$; the *inverse image* of a subset $Y \subseteq B$ of the codomain is $f^{-1}(Y) = \{a \in A \mid f(a) \in Y\}$. Given two functions $f\colon A \to C, g\colon A \to D$ with the same domain, the corresponding function $\langle f, g\rangle\colon A \to C \times D$ to the Cartesian product of their codomains maps every element $a \in A$ to the pair $\langle f, g\rangle(a) = (f(a), g(a))$, consisting of the respective images under the two functions.

Monoids and their fibred products. Given two monoids $(M, \cdot, e), (N, \circ, i)$, their product $(M \times N, \cdot \times \circ, (e, i))$ consists of pairs of elements from the monoid M and the monoid N and is equipped with the point-wise application of the two multiplication operations, i.e., $(m, n)(\cdot \times \circ)(m', n') = (m \cdot m', n \circ n')$, for all $m, m' \in M$ and $n, n' \in N$. The *fibred product* of pairs of monoid homomorphisms $f\colon M \to K$

and $g\colon N \to K$ between monoids M, N, K is the largest sub-monoid $P \subseteq M \times N$ of the product $M \times N$ such that $f(m) = g(n)$ holds for all elements $(m,n) \in P$; following tradition, we denote the fibred product by $M \times_K N$ whenever the pair of monoid homomorphisms f, g is clear from the context.

Free monoids and alphabetic homomorphisms. The free monoid over any non-empty set A is denoted by $(A^*, \cdot, \epsilon)$; its elements are called *words* and the composition $u \cdot v$ of words $u, v \in A^*$ is denoted by juxtaposition uv. The *length* of a word $u \in A^*$ is denoted by $|u|$. A function $f : A \to X$ to an arbitrary set X extends uniquely to a monoid homomorphism $f^*\colon A^* \to X^*$ such that $f^*(a) = f(a)$ for all $a \in A$; such unique extensions of functions are called *alphabetic homomorphisms*.

Ranked alphabets, trees, congruences. A *ranked alphabet* is a finite set Σ of *symbols* with a function $|_|\colon \Sigma \to \mathbb{N}$ that maps symbols to their *arity*. A *tree* over a ranked alphabet Σ is inductively defined as follows. Any symbol a of arity $0 = |a|$ is a tree and if a is a symbol of arity $n = |a|$ and $\boldsymbol{t} = t_1, \ldots, t_n$ is a sequence of n trees, then $a(\boldsymbol{t})$ is a tree. A *congruence relation* on trees is an equivalence relation $\equiv \subseteq \mathcal{T}_\Sigma \times \mathcal{T}_\Sigma$ over the set of trees $\mathcal{T}_\Sigma$ such that $a(\boldsymbol{t}) \equiv a(\boldsymbol{t}')$ if $t_i \equiv t_i'$ $(i = 1, \ldots, n)$.

Graphs and graph morphisms. A *directed multigraph* (or *graph* for short) is a quadruple $G = (V, E, s, t)$ of a set of *vertices* or *nodes* V, a set of *edges* E, and two functions $s, t\colon E \to V$ that map each edge to its *source* and *target* node, resp.; for convenience, we always assume node and edge sets to be disjoint. We write $e\colon u \to v$ in G to express that an edge $e \in E$ has source $s(e) = u \in V$ and target $t(e) = v \in V$. A *graph morphism* from a graph $G = (V, E, s, t)$ to a graph $H = (V', E', s', t')$ is a pair $\phi = (\phi_0, \phi_1)$ of a function on nodes $\phi_0\colon V \to V'$ and a function on edges $\phi_1\colon E \to E'$ that preserves sources and targets of edges, i.e., such that the equations $\phi_0 \circ s = s' \circ \phi_1$ and $\phi_0 \circ t = t' \circ \phi_1$ hold.

1.2 Monoidification: in general and in particular

Many mathematical structures give rise to richer ones if monoids play the role of sets and we replace functions by monoid homomorphisms. For example, *a graph internal*[3] *to monoids* is a tuple of a monoid of nodes V, a monoid of edges E, and two monoid homomorphisms $s, t\colon E \to V$, mapping edges to their sources and targets, respectively. The set of paths in a graph is a standard example of a category. Monoidification yields the central mathematical structure of the paper whose finite instances will be the main protagonists.

Definition 1 (Categories in $\mathscr{Mon}$). *A* category internal to monoids *(or* category in $\mathscr{Mon}$*, for short), is a tuple* $\mathbb{C} = (C_0, C_1, \mathrm{dom}, \mathrm{cod}, \circ, \mathrm{id})$ *of*

- *two monoids* C_0, C_1 *whose elements are called* objects *and* arrows*, resp., whose multiplications are called* parallel compositions *and are denoted by* $\oplus$*, and whose neutral elements are called* unit object *and* unit arrow*, resp.,*

[3] We adhere to the usual terminology.

– *two monoid homomorphisms* $\mathrm{dom}, \mathrm{cod} \colon C_1 \to C_0$ *that assign to each arrow its* domain *and* codomain*, resp.,*
– *a third monoid homomorphism* $\circ \colon C_1 \times_{C_0} C_1 \to C_1$ *that assigns to each element* $(g, f) \in C_1 \times_{C_0} C_1$ *of the fibred product of* dom *and* cod*, i.e., to each pair of arrows* $(g, f) \in C_1 \times C_1$ *s.t.* $\mathrm{dom}\, g = \mathrm{cod}\, f$*, its* sequential composition $g \circ f$*, and*
– *a fourth monoid homomorphism* $\mathrm{id} \colon C_0 \to C_1$ *that gives the* identity arrow id_X *on each object* $X \in C_0$

such that

$$g \circ f \colon X \to Z, \qquad h \circ (g \circ f) = (h \circ g) \circ f$$
$$\mathrm{id}_X \colon X \to X, \qquad \mathrm{id}_Y \circ f = f, \text{ and} \qquad f = f \circ \mathrm{id}_X,$$

for all $X, Y, Z, V \in C_0$ *and all* $f, g, h \in C_1$ *that satisfy* $f \colon X \to Y$, $g \colon Y \to Z$, *and* $h \colon Z \to V$ *(where we write* $k \colon U \to W$ *to abbreviate* $\mathrm{dom}(k) = U$ *and* $\mathrm{cod}(k) = W$ *for any arrow* $k \in C_1$*). It is* finite *if* C_1 *and* C_0 *are finite.*

We often use the semicolon to write compositions, reversing the argument order, thus making $f ; g$ an alias of $g \circ f$, for all $f \colon X \to Y$ and $g \colon Y \to Z$ in $\mathbb{C}$.

Example 1 (Relations between natural numbers). For each natural number n, we have its *lower set* $\underline{n} := \{i \in \mathbb{N} \mid i < n\}$. Relations between lower sets of natural numbers form a category in $\mathscr{Mon}$ in the following manner. First, the object monoid $(\underline{\mathbb{N}}, \oplus, \varnothing)$ has lower sets of natural as elements (i.e., $\underline{\mathbb{N}} = \{\underline{n} \mid n \in \mathbb{N}\}$) and its parallel composition $\oplus$ is defined by $\underline{n} \oplus \underline{m} = \underline{n + m}$ (for all $n, m \in \mathbb{N}$). Next, the arrow monoid $(\mathcal{R}_{\underline{\mathbb{N}}}, \oplus, \mathrm{id}_\varnothing)$ has

– relations $R \colon \underline{m} \nrightarrow \underline{n}$ between lower sets of natural numbers as elements,[4]
– the homomorphism that maps pairs of relations $R \colon \underline{m} \nrightarrow \underline{n}$, $S \colon \underline{p} \nrightarrow \underline{q}$ to their disjoint union $R \oplus S \colon \underline{m + p} \nrightarrow \underline{n + q}$ as parallel composition where the *disjoint union* $R \oplus S$ relates two elements $i \in \underline{m + p}, j \in \underline{n + q}$ if either $i \; S \; j$ or $(i - p) \; R \; (j - q)$ holds, and
– the empty relation $\mathrm{id}_\varnothing \colon \varnothing \nrightarrow \varnothing$ as unit.

Finally, adding the usual relational composition brings us to the data of a category $\big((\mathcal{R}_{\underline{\mathbb{N}}}, \oplus, \mathrm{id}_\varnothing), \mathrm{dom}, \mathrm{cod}, \circ, \mathrm{id}\big)$ whose last component id maps every lower set $\underline{n}$ to its identity function[5] $\mathrm{id}_{\underline{n}} \colon \underline{n} \nrightarrow \underline{n}$.

The idea of internalization extends to (homo-)morphisms of structures. For example a morphism of graphs in $\mathscr{Mon}$ is a graph morphism on the underlying graph that also respects the monoid structure. Structure preserving maps between categories are called *functors* and monoidification results in the second fundamental concept that we shall rely on to define recognizability.

Definition 2 (Functor between categories in $\mathscr{Mon}$). *Let* $\mathbb{C}, \mathbb{D}$ *be two categories in* $\mathscr{Mon}$*. A* functor *from* $\mathbb{C}$ *to* $\mathbb{D}$*, denoted* $F \colon \mathbb{C} \to \mathbb{D}$*, is a pair* $F = (F_0, F_1)$ *of two*

[4] Recall that relations include information about their domain and codomain.
[5] Note that we have defined functions as relations with additional properties.

monoid homomorphisms $F_0 \colon C_0 \to D_0$, $F_1 \colon C_1 \to D_1$ *between object and arrow monoids, resp., such that the equations*

$$F_0 \circ \mathrm{dom} = \mathrm{dom} \circ F_1 \qquad F_1(g \circ f) = F_1(g) \circ F_1(f)$$
$$F_0 \circ \mathrm{cod} = \mathrm{cod} \circ F_1 \qquad F_1(\mathrm{id}_X) = \mathrm{id}_{F_0(X)}$$

hold for all $f \colon X \to Y$, $g \colon Y \to Z$ *in* $\mathbb{C}$ *and all* $X \in C_0$. *A functor* F *is surjective if its components* F_0, F_1 *are surjective functions.*

Thus, a functor preserves the graph structure of a category, distributes over composition and maps identity arrows to identity arrows.

2 Myhill-Nerode via universal recognizers

We set out to study languages of "generalized words", such as trees, forests, and—as detailed later—circuit diagrams. The essential common trait of "generalized words" of a language is that we can find a suitable category internal to monoids that contains them as arrows. The main result of the section will be a direct analogue of the classic Myhill-Nerode Theorem for languages of "generalized words". The running example of a category internal to monoids is the following variation of forests, related to the work of Ésik and Weil [ÉW05] on trees.

Example 2 (Forests of pruned trees). We reserve a special symbol for labelling leaves in trees, called *wire* and denoted by ı. Thus, we consider ranked alphabets that have the wire as a distinguished symbol of arity 0; its behaviour will be different from that of a variable, as discussed at the end of the example. Leaves that are labelled as wires are called *pruned twigs* and they are endowed with the order in which an in-order tree traversal would visit them. A pruned tree is called *integral* if none of its leaves are pruned twigs. A *pruned forest* is a list of pruned trees. Often, we drop the qualification of being pruned.

We define a category of forests in $\mathscr{Mon}$ as follows. Its arrows are forests, i.e., lists of pruned trees $[\![t_1, \ldots, t_n]\!]$. Parallel composition of forests is list concatenation, denoted by $\|$. Concerning sequential composition, note that a forest of n trees can be grafted into a forest that has n pruned twigs by taking trees from the first list and grafting them on the pruned twigs from the other forest, one after another, e.g., $[\![a(\imath, c), b, d(c)]\!]; [\![\imath, a\big(d(\imath), \imath\big)]\!] = [\![a(\imath, c), a\big(d(b), d(c)\big)]\!]$.

Formally, the object monoid is the monoid of natural numbers $(\mathbb{N}, +, 0)$, the domain of a forest is its length as a list, and its codomain is the total count of pruned twigs. Observe that two forests can be composed (sequentially) by grafting the trees of the first forest—to be thought of as scions—onto the pruned twigs of the second forest, whenever their numbers match; otherwise the sequential composition is "undefined".

Note that, typically, different trees are "substituted" for twigs during sequential composition. If one wanted to describe grafting via substitution, each leave would require a different variable. One can achieve this by numbering twigs using initial segments of the positive integers, adding offsets during parallel

composition. In fact, the monoid of natural numbers $(\mathbb{N}, +, 0)$ performs this bookkeeping of "interfaces"! Finally, note that the example can be adapted to unranked ordered trees without difficulty.

We start with a definition of recognizable languages of arrows in categories internal to monoids and a short discussion of their basic properties in Section 2.1. Then we define the counterpart of syntactic monoids, which will be pivotal in the proof of the Myhill-Nerode Theorem (Section 2.2).

2.1 Recognizable languages and their basic closure properties

In definitions and results, we assume a fixed category $\mathbb{C} = (C_0, C_1, \mathrm{dom}, \mathrm{cod}, \circ, \mathrm{id})$ internal to monoids whose unit object and arrow are denoted by O and id_{O}, respectively. Recall that we refer to the monoid operation $\oplus$ on arrows (resp. objects) as *parallel* composition while the composition operation $\circ$ is qualified as *sequential*; the semicolon is an alternative notation for sequential composition, which takes arguments in the opposite order. With all preparations in place, we proceed to the main definition.

Definition 3 (Recognizable language in a category in *Mon*). *A* language *in the category $\mathbb{C}$ is a subset $L \subseteq C_1$ of its arrows. A functor $F\colon \mathbb{C} \to \mathbb{D}$ to a category $\mathbb{D}$ in* ***Mon*** recognizes *a language $L \subseteq C_1$ if the language is the inverse image of its direct image under the functor, i.e., if $F_1^{-1}\big(F_1(L)\big) = L$. A language $L \subseteq C_1$ is* recognizable *if it is recognized by some functor into a finite category in* ***Mon****.*

Example 3 (Tree languages as a special case). Every language of trees induces a language of integral trees, considering trees as singleton lists of pruned trees, or more formally, mapping a tree t to its singleton forest $[\![t]\!]$. Later, we shall see that a language of trees L is recognizable in the usual sense if, and only if, its embedding into forests of pruned trees is recognizable according to Definition 3.

A basic example of a language that is not recognizable is

$$\Big\{ a(\mathrm{I})^n \parallel a(\mathrm{I})^n \;\Big|\; n \in \mathbb{N} \Big\},$$

where the symbol a has rank 1, $a(\mathrm{I})^0 = \mathrm{I}$, and $a(\mathrm{I})^{n+1} = a(\mathrm{I})^n a(\mathrm{I})$. This language can be pictured as pairs of Bamboo shoots with the same number of segments. Note that the wire corresponds to the empty word and the tree $a(\mathrm{I})^n$ of height n to the n-th power a^n of the "letter" a. In fact, every recognizable word language induces a recognizable forest language by replacing each letter a by $a(\mathrm{I})$, its application to a twig, and considering the result as a singleton list.

Recognizable languages are closed under all Boolean operations, images (direct and inverse), and composition (sequential and parallel). We now turn to the first fundamental result about recognizable languages.

2.2 The Myhill-Nerode theorem and the syntactic category

The goal is a Myhill-Nerode Theorem whose formulation matches the original one for word languages virtually verbatim. After giving the straightforward definitions of congruence relations and their quotients, existence of universal language recognizers leads to a suitable notion of syntactic congruence with a corresponding Myhill-Nerode Theorem.

Congruence relations and their quotients. Congruence relations of monoids are closed w.r.t. (sequential) composition; the counterpart for categories in $\mathscr{Mon}$ are also closed w.r.t. parallel composition.

Definition 4 (Congruences). *A* congruence *relation on the category $\mathbb{C}$ is an equivalence relation $\equiv \subseteq C_1 \times C_1$, relating arrows of $\mathbb{C}$, such that, for all quadruples of arrows $f, g, h, k \in C_1$,*

- *$f \equiv g$ implies $h \oplus f \oplus k \equiv h \oplus g \oplus k$ and*
- *$f \equiv g$ implies $h; f; k \equiv h; g; k$, if both compositions $h; f; k$, $h; g; k$ are defined.*

A congruence on trees can be seen as a special case of a congruence on forests of pruned trees.

Example 4 (Congruences of trees). Consider a congruence $\approx$ on trees over a ranked alphabet. After adding the wire $\mid$ to the ranked alphabet, the relation $\approx$ induces an equivalence relation on integral trees. We can extend it to a congruence on forests of pruned trees—without adding new relations between integral trees—as follows. Two pruned trees t, t' with $n > 0$ twigs are related, in symbols $t \approx t'$, iff, for all pairs of sequences $\boldsymbol{t} = t_1, \ldots, t_n$ $\boldsymbol{t}' = t'_1, \ldots, t'_n$ of (integral) trees that are *index-wise congruent* in the sense that $t_i \approx t'_i$ holds for all $i \in \{1, \ldots, n\}$, we have

$$(t_1 \parallel \cdots \parallel t_n); t \approx (t'_1 \parallel \cdots \parallel t'_n); t'.$$

Next, we extend $\approx$ to forests of pruned trees such that any two forests $\boldsymbol{t} = [\![t_1, \ldots, t_n]\!]$ $\boldsymbol{t}' = [\![t'_1, \ldots, t'_n]\!]$ shall be congruent, in symbols $\boldsymbol{t} \approx \boldsymbol{t}'$, iff they are index-wise congruent, i.e., if $t_i \approx t'_i$ $(i = 1, \ldots, n)$. One can verify that the result is the least congruence relation that extends the original congruence $\approx$ from trees to forests of pruned trees; moreover, equivalence classes of trees are in bijective correspondence to equivalence classes of integral trees. Thus, the embedding of trees into PLACIDS extends to congruence relations on trees.

Functors in $\mathscr{Mon}$ induce congruences on their domain categories, relating pairs of arrows (and objects) if they have the same image. In fact, every congruence can be obtained in this way.

Lemma 1. *For every congruence $\equiv \subseteq C_1 \times C_1$, there is a functor $F \colon \mathbb{C} \to \mathbb{D}$ such that a pair of arrows is related by the congruence if, and only if, they have the same image under the functor, i.e., such that $F_1(f) = F_1(g)$ iff $f \equiv g$ holds for all $f, g \in C_1$.*

The proof of the lemma amounts to showing that we can endow the congruence with the structure of a category in $\mathscr{Mon}$ in the following manner.

Definition 5 (Quotient category). *Let $\equiv \subseteq C_1 \times C_1$ be a congruence. Its* object congruence*, denoted by $\doteq$, contains a pair of objects $(X, Y) \in \doteq$ iff there is a pair of equivalent arrows $(f, g) \in \equiv$ such that $(X, Y) = (\mathrm{dom}\, f, \mathrm{dom}\, g)$ or $(X, Y) = (\mathrm{cod}\, f, \mathrm{cod}\, g)$. The* quotient category *of the congruence $\equiv$, denoted by $\mathbb{C}/\equiv$, has*

- *equivalence classes of the congruence $\equiv$ as arrows, equipped with the extension of parallel composition to classes that is defined by the equation $[f]_\equiv \oplus [g]_\equiv = [f \oplus g]_\equiv$, for all $f, g \in C_1$, whose unit is the class of the unit arrow $[\mathrm{id}_\mathrm{O}]_\equiv$,*
- *equivalence classes of the object congruence $\doteq$ as objects, equipped with the extension of parallel composition to classes, defined by the equation $[X]_\doteq \oplus [Y]_\doteq = [X \oplus Y]_\doteq$, for all $X, Y \in C_0$, whose unit is the class of the unit object $[\mathrm{O}]_\doteq$,*
- *domain and codomain homomorphisms that are defined by the equations*

$$\mathrm{dom}\big([f]_\equiv\big) = [\mathrm{dom}\, f]_\doteq \qquad \mathrm{cod}\big([f]_\equiv\big) = [\mathrm{cod}\, f]_\doteq,$$

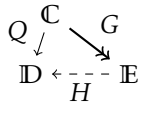
  for all $f \in C_1$,
- *the extension of sequential composition to equivalence classes, defined by the equation $[g]_\equiv \circ [f]_\equiv = [g \circ f]_\equiv$, for all arrows $f\colon X \to Y$, $g\colon Y \to Z$ in $\mathbb{C}$, as sequential composition,*
- *equivalence classes of identities as identities, i.e., $\mathrm{id}_{[X]_\doteq} = [\mathrm{id}_X]_\equiv$, for all $X \in C_0$.*

Finally, the functor that embeds object and arrows into their equivalence class is called quotient functor *of the congruence $\equiv$ and is denoted by $[_]\colon \mathbb{C} \to \mathbb{C}/\equiv$.*

Note that there is a tight correspondence between quotient functors and surjective functors that maps every surjective functor to the quotient functor of the congruence of image-equality under the functor. We now turn to the relevant special case of congruences and their quotients.

The Myhill-Nerode Theorem via universal recognizers. We now generalize the Myhill-Nerode Theorem for word or tree languages in terms of suitable counterparts of the syntactic congruence and its monoid. The core technical property is existence of a universal recognizer.

Lemma 2 (Universal recognizers). *Let $L \subseteq C_1$ be a language. There exist a category $\mathbb{D}$ in $\mathcal{Mon}$ and a functor $Q\colon \mathbb{C} \to \mathbb{D}$ that recognizes L such that for every surjective functor $G\colon \mathbb{C} \to \mathbb{E}$ that recognizes L there exists a unique functor $H\colon \mathbb{E} \to \mathbb{D}$ such that $Q = H \circ G$.*



The proof of Lemma 2 is deferred to Appendix A. It implies that the following counterparts of syntactic congruence and syntactic monoid are well-defined.

Definition 6 (Syntactic congruence and category). *Given a language $L \subseteq C_1$, its* syntactic congruence*, denoted by $\equiv_L$, is the coarsest congruence relation $\equiv_L \subseteq C_1 \times C_1$ on C_1 such that for all pairs of arrows $(f, g) \in \equiv_L$ and all arrows $h, k \in C_1$,*

- *$h \oplus f \oplus k \in L$ iff $h \oplus g \oplus k \in L$, and*
- *$h; f; k \in L$ iff $h; g; k \in L$, if both compositions $h; f; k$, $h; g; k$ are defined.*

The syntactic category *of L is the quotient $\mathbb{C}/\equiv_L$ of its syntactic congruence.*

The usual syntactic congruence of tree languages is subsumed.

Example 5. Similar to Example 4, we identify trees with integral ones and embed trees into forests by mapping a tree t to its singleton forest $[\![t]\!]$. Moreover, every tree language L induces a forest language, denoted by $[\![L]\!]$. The least extension $\hat{\equiv}_L$ of the syntactic congruence $\equiv_L$ such that $t \equiv_L t'$ implies $[\![t]\!] \hat{\equiv}_L [\![t']\!]$ coincides with the syntactic congruence $\equiv_{[\![L]\!]}$ of the lifted language $[\![L]\!]$. Clearly, if $\equiv_{[\![L]\!]}$ is of finite index, then so is $\equiv_L$; the converse is true as well (see Appendix B).

Now, let us prove the first result

Theorem 1 (Myhill-Nerode internal to monoids). *For all categories $\mathbb{C}$ internal to monoids, for all languages $L \subseteq C_1$ of arrows in $\mathbb{C}$, the following are equivalent.*

1. *The language L is recognizable.*
2. *The syntactic congruence $\equiv_L$ of the language L is of finite index.*
3. *The syntactic category $\mathbb{C}/\equiv_L$ of L is finite.*

Proof. The Myhill-Nerode Theorem follows from Lemma 2 since index finiteness of the syntactic congruence implies recognizability while the converse follows as every other recognizing functor induces a congruence that is finer than the syntactic congruence and we can assume w.l.o.g. that recognizing functors are surjective.

In the second part of the paper, we shall apply the theorem beyond tree and forest languages.

3 PLACIDS as free PROs, or X-categories, or magmoids

Coexistence of at least three different names for the very same concept is reason enough to digress, especially if it is not the trinity of Turing computable, lambda definable, and recursive. First, we have MacLane's notion of product and permutation category, which he has chosen to abbreviate as PROP [Mac65, p. 97]; leaving out the permutations, we are left with a PRO—a *product*[6] *category*. Günter Hotz called them X-categories [Hot68, p. 210], making his notation for the monoid operation part of the name. Finally, for graph automata, the notion of magmoid is used [BK08]. We are following MacLane's terminology.

PROs. Having the choice among several definitions and characterizations of PROs, we are relying on Steve Lack's observation [Lac04] that PROs and their morphisms are instances of categories and functors internal to monoids—which is essentially the way that they have been defined by Hotz [Hot68, p. 210].

[6] Such a "generalized tensor product" is not to be confused with the Cartesian product.

Definition 7 (PROs and their morphisms). *A* product category *(or* PRO*, for short) is a category in $\mathscr{Mon}$ whose object monoid is free. A* PRO-morphisms *between* PROS $\mathbb{C}, \mathbb{D}$ *is a functor $F\colon \mathbb{C} \to \mathbb{D}$ whose object homomorphism F_0 is alphabetic.*[7]

The category of relations on lower sets of the natural numbers from Example 1 is an example of a PRO as the monoid of natural numbers is isomorphic to the free monoid $\{1\}^*$. Moreover, forming the opposite of relations induces a PRO morphism on $\mathcal{R}_{\mathbb{N}}$. The remainder of the section is devoted to the characterization of free PROS, which are to PROS what word monoids are to monoids, and the counterpart of alphabetic homomorphisms; the latter concepts will allow for a succinct formal definition of Bossut's automaton model.

From alphabets to signatures. As described in the introduction, signatures can be understood as string rewriting systems over words in which rewriting rules are labelled by symbols. The formal definition is as follows.

Definition 8 (PRO signatures). *Let $\mathfrak{C} \neq \emptyset$ be a non-empty set of* colours[8]*. A $\mathfrak{C}$-coloured* signature *is a triple $\Sigma = (\Sigma, s, t)$ such that we obtain a graph $(\mathfrak{C}^*, \Sigma, s, t)$ by adding $\mathfrak{C}^*$, the set of $\mathfrak{C}$-words, as set of nodes; its elements $a \in \Sigma$ are called* symbols*. A symbol $a\colon u \to v \in \Sigma$ is a* terminator *(resp.,* initiator*) if its source (resp. target) is the empty word.*

From now on, all signatures are assumed to be finite.

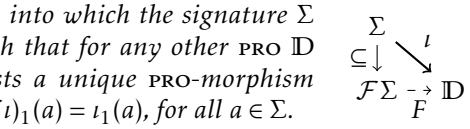
Example 6 (String rewriting systems). String rewriting systems over an alphabet A are essentially A-coloured signatures of PROS in which there is at most one edge from any node u to another node v: any symbol $a\colon u \to v$ is a string rewriting rule $u \rightsquigarrow v$.

From free monoids to free PROS*.* As alphabets of free monoids come with canonical embeddings, signatures follow suit.

Definition 9 (Signature embeddings). *An* embedding *of a $\mathfrak{C}$-coloured signature (Σ, s, t) into a* PRO $\mathbb{D}$ *is a graph morphism $\iota\colon (\mathfrak{C}^*, \Sigma, s, t) \to (D_0, D_1, \mathrm{dom}, \mathrm{cod})$ whose function on nodes $\iota_0\colon \mathfrak{C}^* \to D_0$ is an alphabetic homomorphisms.*

Finally, PLACIDS over a given set of basic gates are characterized, up to isomorphism, as the free PRO over the signature with gate types as symbols.

Fact 1 ("The" free PRO) *Let $\mathfrak{C} \neq \emptyset$ be a set of colours and let (Σ, s, t) be a $\mathfrak{C}$-coloured signature. There exists a* PRO $\mathcal{F}\Sigma$ *into which the signature Σ embeds via the inclusion $\Sigma \subseteq \mathcal{F}\Sigma$ such that for any other* PRO $\mathbb{D}$ *and embedding $\iota\colon \Sigma \to \mathbb{D}$, there exists a unique* PRO*-morphism $\mathcal{F}\iota\colon \mathcal{F}\Sigma \to \mathbb{D}$ that restricts to ι, i.e., $(\mathcal{F}\iota)_1(a) = \iota_1(a)$, for all $a \in \Sigma$.*



From now on we use the terms PLACID and free PRO interchangeably and symbols will also be called (types of) gates. We shall use mainly the letters p, r to range over arrows of the free PRO $\mathcal{F}\Sigma$ and use $u, v, w, x, y, z, \ldots$ for objects.

[7] Recall that alphabetic homomorphisms are extensions of functions between generators.
[8] This is the standard terminology for product and permutation categories [YJ15].

From alphabetic morphisms to morphisms of free PROs. As alphabets are generalized to signatures, graph morphisms take the place of functions.

Definition 10 (Signature morphism). *Given signatures* Γ *and* Σ*, a* signature morphism *from* Γ *to* Σ *is a graph morphism* $\phi\colon \Gamma \to \Sigma$ *whose node function* ϕ_0 *is an alphabetic homomorphism.*

Signature morphisms have unique extensions to free PROs.

Fact 2 *Let* Γ *and* Σ *be signatures. Every signature morphism* $\phi\colon \Gamma \to \Sigma$ *has a unique extension* $\mathcal{F}\phi\colon \mathcal{F}\Gamma \to \mathcal{F}\Sigma$ *such that* $(\mathcal{F}\phi)_0 = \phi_0$ *and* $(\mathcal{F}\phi)_1(a) = \phi_1(a)$*, for all* $a \in \Gamma$*.*

Now all prerequisites are in place to show that the notion of recognizable language is sound and complete w.r.t. Bossut's automaton model for a natural class of circuit diagrams.

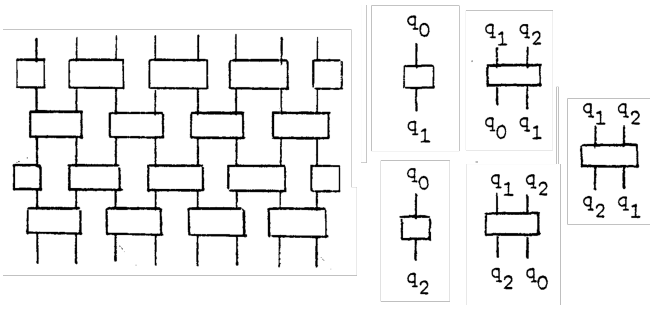


Fig. 2. Bossut's Brick wall generating rules [Bos86, p. 78f.][9]

4 PRO automata and recognizable languages of PRO arrows

It remains to provide an automaton model that corresponds to the abstract notion of recognizable language to justify the name of Theorem 1. Bossut's model will turn out to be the perfect candidate—at least if every gate has input and output ports: its runs can be seen as the simplest form of *tilings* from graph acceptors [Tho91] and its initial and final state words allow to embed word and tree automata without complications; moreover, they have an extension to graphs [BK08], which will be the topic of future work.

We thus proceed to formally define Francis Bossut's automata in the setting of PROs, show that all automaton acceptable languages are recognizable, and that the converse is true if signatures do contain neither initiators nor terminators; the case of trees was already treated in Example 5 (see also Appendix B). That

[9] The illustrations have been rearranged.

Bossut's automata have complements—at least if source and target words of all symbols are non-empty—follows as an immediate corollary from the Myhill-Nerode Theorem. We fix a set of colours $\mathfrak{C} \neq \emptyset$ and a $\mathfrak{C}$-coloured signature $\Sigma = (\Sigma, s, t)$ for the remainder of this section.

Definition 11 (PRO automaton). *A* PRO automaton *over the signature Σ is a tuple $\mathcal{A} = (Q, \Gamma, \delta, I, F)$ of a finite set of* states *Q, a Q-coloured* working signature *(Γ, s', t'), a signature morphism $\delta\colon \Gamma \to \Sigma$, called* transition function*, and regular sets of* initial *(resp.,* final*) state words $I \subseteq Q^*$ (resp., $F \subseteq Q^*$); it is* normal *if the function $\langle s', \delta_1, t' \rangle\colon \Gamma \to Q^* \times \Sigma \times Q^*$ is injective.*[10]

A run *in an automaton $\mathcal{A}$ as above is an arrow $r\colon \tilde{q} \to \tilde{q}'$ in the free* PRO *$\mathcal{F}\Gamma$ and it is* accepting *if its domain $\tilde{q}$ and codomain $\tilde{q}'$ are initial and final state words, resp., in symbols, $\tilde{q} \in I$ and $\tilde{q}' \in F$. The* language *of an automaton, denoted by $\mathcal{L}(\mathcal{A})$, is the image of the accepting runs under the extension $\mathcal{F}\delta$ of the signature morphism δ to a* PRO*-morphism $\mathcal{F}\delta\colon \mathcal{F}\Gamma \to \mathcal{F}\Sigma$, in symbols*

$$\mathcal{L}(\mathcal{A}) = \mathcal{F}\delta_1\Big(\Big\{r\colon \tilde{q} \to \tilde{q}' \text{ in } \mathcal{F}\Gamma \;\Big|\; \tilde{q} \in I, \tilde{q}' \in F\Big\}\Big).$$

A language is automaton acceptable *if it is the language of some* PRO *automaton.*

Example 7 (Brick walls [Bos86, p. 78f.]). Figure 2 reproduces Bossut's graphical illustration of automata transitions for an automaton that accepts well-structured "brick walls" with the language $q_0\{q_1q_2\}^*q_0 \cup \{q_1q_2\}^+$ for both initial and final states over the set of states $\{q_0, q_1, q_2\}$.

The first result about PRO-automata is soundness of the automaton model or, equivalently, completeness of recognizability.

Proposition 1. *Every automaton acceptable language is recognizable.*

Proof. First, we show that the set of accepting runs of an automaton $\mathcal{A} = (Q, \Gamma, \delta, I, F)$ forms a recognizable language. For this, let $\mathcal{A}_I$ and $\mathcal{A}_F$ be deterministic finite automata that accept the initial and final state words, resp., i.e., such that $I = L(\mathcal{A}_I)$ and $F = L(\mathcal{A}_F)$. Now, two runs are syntactically congruent if their domain and codomain words lead to the same states in $\mathcal{A}_I$ and $\mathcal{A}_F$, respectively. This however already implies that the syntactic congruence of accepting runs is of finite index. The proposition follows by closure of recognizability under direct images.

The converse is true if we rule out terminators and initiators.

Theorem 2. *Let $\mathfrak{C} \neq \emptyset$ be a set of colours, let (Σ, s, t) be a $\mathfrak{C}$-coloured signature without terminators and initiators, and let $L \subseteq \mathcal{F}\Sigma$ be a language of* PLACIDS*. The language L is recognizable if, and only if, it is automaton acceptable.*

[10] This means, for all symbols $a, b\colon \tilde{q} \to \tilde{q}'$ in Γ that are parallel in Γ and have the same δ-image $\delta_1(a) = \delta_1(b)$, the equality $a = b$ of symbols follows.

This theorem is an important "soundness check" for recognizability via functors; the proof is deferred to Appendix C. We can combine it with the Myhill-Nerode Theorem to conclude that Bossut's automata can be complemented.

Corollary 1. *The complement of every language of a* PRO *automaton over a signature* Σ *that does neither contain initiators nor terminators is automaton acceptable.*

For general signatures, the correspondence of Theorem 2 fails, in general, even if we can recover recognizable tree and forest languages via trivial encodings of tree and forest automata. The following is an example of a recognizable language that is not automaton recognizable. As signature, we take the simplest one with terminators and initiators, namely $\{ \triangleleft, \triangleright \}$. The language $L = \{ \triangleleft\triangleright \}$ is recognizable but not automaton acceptable; the least automaton acceptable language that contains L also contains every PLACID of the form $\triangleleft\triangleright \oplus \cdots \oplus \triangleleft\triangleright$. In the end, this indicates that the natural habitat for Bossut's automata are categories internal to semi-groups, replacing free monoids by free semi-groups everywhere. The semi-group setting still allows to embed trees and forests, replacing each symbol of rank 0 be a corresponding leaf colour.

5 Related and future work

Besides further work on PRO-automata as acceptors of recognizable languages of arrows of categories internal to semi-groups, there are essentially two major directions for future work: the extension of the general theory to encompass graph languages [BK08] and future work on classes of recognizable languages whose syntactic categories form pseudovarieties of finite categories in $\mathscr{Mon}$.

Towards PROPS*, graphoids, and beyond.* The definition of recognizable language transfers to symmetric monoidal categories (SMC): a language of arrows in a SMC is *recognizable* if it coincides with the inverse image of its image under a functor to a finite SMC. Similarly, the defining equations of graphoids [BK08] have counterparts in SMCs with suitable extra structure, leading to a (2-)category of *graphoidal* categories. The definition of recognizability carries over, virtually verbatim. The goal is to find the general structure behind these and similar generalizations, ideally with a blueprint for a common automaton model.

Pseudovarieties à la Eilenberg . Given the recent advances on the structure of Eilenberg's Theorem [UACM16,Sal17], the natural next step are Eilenberg type correspondences for languages in (symmetric) monoidal categories, obtained by "monoidification" of existing work on pseudovarieties of finite categories [TSG88]. The broad goal of this second line of future work is a better understanding of first-order definable languages of trees, forests, and circuit diagrams.

6 Conclusion

We have introduced an abstract notion of recognizability that conservatively extends the classic notion for word and tree languages and is general enough to

reason about languages of planar acyclic circuit diagrams, the basic fragment of graphical syntax for quantum processes [CK17]. The two main results are the Myhill-Nerode Theorem and a soundness result for languages of PLACIDS whose gates have at least one input and one output port, showing that recognizable languages are exactly the automaton acceptable ones.

The main results rely on the syntactic category of languages—internal to monoids. Syntactic categories of PLACID languages are essentially algebras of free monoidal theories. The generalization to symmetric monoidal theories, which amounts to dropping the restriction of planarity of circuit diagrams, is the natural next step. Syntactic categories of recognizable languages in any (symmetric) monoidal theory are finite algebras. This opens up the field of algebraic language theory for studies beyond tree and forest languages in direct analogy to work on syntactic categories [TSG88].

Acknowledgements. The author would like to thank Henning Urbat for pointing out the connections to work on recognizability in finite categories and providing a short summary of the state of the art of Eilenberg type correspondences. Moreover, the author is grateful for constructive discussions with Andreas Maletti about Bossut's automata model, which brought up the topic of complementation.

References

AMU15. Jirí Adámek, Stefan Milius, and Henning Urbat. Syntactic monoids in a category. In Lawrence S. Moss and Paweł Sobociński, editors, *Proceedings of CALCO 2015*, pages 1–16, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

BDW95. Francis Bossut, Max Dauchet, and Bruno Warin. A Kleene theorem for a class of planar acyclic graphs. *Information and Computation*, 117(2):251–265, 1995.

BK08. Symeon Bozapalidis and Antonios Kalampakas. Graph automata. *Theoretical Computer Science*, 393(1):147–165, 2008.

Boj15. Mikołaj Bojańczyk. Recognisable languages over monads. In Igor Potapov, editor, *Proceedings of DLT 2015*. Springer International Publishing, 2015.

Bos86. Francis Bossut. *Rationalité et reconnaissabilité dans les graphes*. PhD thesis, Université Lille, Sciences et Technologies, 1986.

BS10. John Baez and Mike Stay. Physics, topology, logic and computation: a Rosetta Stone. In *New structures for physics*, pages 95–172. Springer, 2010.

BW08. Mikołaj Bojańczyk and Igor Walukiewicz. Forest algebras. *Automata and logic: history and perspectives*, pages 107–132, 2008.

CK17. Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017.

ÉW05. Zoltán Ésik and Pascal Weil. Algebraic recognizability of regular tree languages. *Theoretical Computer Science*, 340(2):291–321, 2005.

Hot65. Günter Hotz. Eine Algebraisierung des Syntheseproblems von Schaltkreisen I. *Elektronische Informationsverarbeitung und Kybernetik*, 1(3):185–205, 1965.

Hot68. Günter Hotz. Reduktionssätze über eine Klasse formaler Sprachen mit endlich vielen Zuständen. *Mathematische Zeitschrift*, 104(3):205–221, Jun 1968.

Jon96. Peter R. Jones. Profinite categories implicit operations and pseudovarieties of categories. *Journal of Pure and Applied Algebra*, 109(1):61–95, 1996.
Lac04. Stephen Lack. Composing PROPs. *Theory and Applications of Categories*, 13(9):147–163, 2004.
Mac65. Saunders MacLane. Categorical algebra. *Bulletin of the American Mathematical Society*, 71(1), 1965.
Por08. Hans E. Porst. On categories of monoids, comonoids, and bimonoids. *Quaestiones Mathematicae*, 31(2):127–139, 2008.
Sal17. Julian Salamanca. Unveiling Eilenberg-type correspondences: Birkhoff's theorem for (finite) algebras + duality. *CoRR*, abs/1702.02822, 2017.
Tho91. Wolfgang Thomas. On logics, tilings, and automata. In *Proceedings of ICALP 1991*, Lecture Notes in Computer Science, pages 441–454. Springer-Verlag, 1991.
TSG88. Denis Thérien and Małgorzata Sznajder-Głodowski. Finite categories and regular languages. *Banach Center Publications*, 1(21):395–402, 1988.
UACM16. Henning Urbat, Jirí Adámek, Liang-Ting Chen, and Stefan Milius. One Eilenberg theorem to rule them all. *CoRR*, abs/1602.05831, 2016.
YJ15. Donald Yau and Mark W. Johnson. *A foundation for PROPs, algebras, and modules*, volume 203. American Mathematical Society, 2015.

A Existence of universal recognizers

There is a short proof for the existence of universal recognizers for languages of a category $\mathbb{C}$ in $\mathscr{Mon}$ (Lemma 2) using results of Porst [Por08]; Lemma 2 implies that the syntactic congruence and category of any language $L \subseteq C_1$ is well-defined. The proof first establishes that the category of categories in $\mathscr{Mon}$ with functors as morphisms is cocomplete, relying on a general result [Por08]; for simplicity, avoiding issues of size, we assume that the category $\mathbb{C}$ is countable, i.e., C_1 and C_0 are countable sets.

Lemma 2 (Universal recognizer). *Let $L \subseteq C_1$ be a language with C_1 countable. There exist a category $\mathbb{D}$ in $\mathscr{Mon}$ and a functor $Q\colon \mathbb{C} \to \mathbb{D}$ that recognizes L such that for every surjective functor $G\colon \mathbb{C} \to \mathbb{E}$ that recognizes L there exists a unique functor $H\colon \mathbb{E} \to \mathbb{D}$ such that $Q = H \circ G$.*

Proof. The category of small categories $\mathscr{Cat}$ is locally presentable. Equipping it with the Cartesian product yields the closed symmetric monoidal category $(\mathscr{Cat}, \times, \mathbb{1})$. Now, categories in $\mathscr{Mon}$ are essentially the same as (pseudo-)monoid objects in the monoidal category $(\mathscr{Cat}, \times, \mathbb{1})$: the monoid operations on objects and arrows of any category $\mathbb{D}$ in $\mathscr{Mon}$ correspond to a functor $\oplus\colon \mathbb{D} \times \mathbb{D} \to \mathbb{D}$, the unit object and arrow can be seen as a single functor $\mathrm{id}\colon \mathbb{1} \to \mathbb{D}$, and these functors satisfy the monoid laws, thus making $\mathbb{D}$ a *monoid over* $(\mathscr{Cat}, \times, \mathbb{1})$ [Por08, § 2.2].

As $\mathscr{Cat}$ is locally presentable and we are using the Cartesian product $\times$ as monoidal product, the monoidal category $(\mathscr{Cat}, \times, \mathbb{1})$ is *admissible* in the sense of Porst [Por08], and, therefore, the category of (pseudo-)monoids in the monoidal category $(\mathscr{Cat}, \times, \mathbb{1})$ is locally presentable [Por08, p. 7], which in turn implies

that it has all small colimits. Hence (using the simplifying assumption that $\mathbb{C}$ is at most countable) we can take the colimit of all surjective recognizers of the language $L \subseteq C_1$; also, there is at least one surjective recognizer, namely $\mathrm{id}_{\mathbb{C}} \colon \mathbb{C} \to \mathbb{C}$. The ("diagonal" of) the colimit of all surjective recognizers—a wide pushout—is the universal recognizer.

The proof also implies that the universal recognizer identifies two arrows $f, g \in C_1$ if, and only if, some recognizer identifies them. Syntactic congruence corresponds to image equality w.r.t. "the" universal recognizer and the syntactic category is a canonical choice of representative from the isomorphism class of universal recognizers.

B Tree languages: soundness and completeness.

The usual notion of recognizability for tree languages can be embedded into the more general framework of the present paper via an encoding of tree automata. In fact, Bossut's automata model contains tree automata as a special case using signatures (Σ, s, t) that are $\{1\}$-coloured and consist of symbols $a \in \Sigma$ with exactly one output port (i.e., $|t(a)| = 1$) and adding the two requirements that the empty word be the only initial state word and that all final state words be single letters. Thus, we can define tree automata as follows.

Definition 12 (Tree automaton). *A* tree automaton *is a normal* PRO*-automaton* $(Q, \Gamma, \delta, I, F)$ *over a* $\{1\}$*-coloured signature* (Σ, s, t) *such that*

- *all symbols* $a \in \Sigma$ *have* $1 = t(a)$ *as target,*
- *the initial state set* $I = \{\epsilon\}$ *contains only the empty word, and*
- *final state words are single states, i.e.,* $F \subseteq Q$.

Thus, the automaton acceptable tree languages are all recognizable, by Proposition 1. The remaining task is to show that every recognizable language $L \subseteq \mathcal{F}\Sigma$ of the free PRO that consists only of (integral) trees is the language of an automaton.

Lemma 3 (Completeness of tree automata). *Let* (Σ, s, t) *be a* $\{1\}$*-coloured signature all whose symbols* $a \in \Sigma$ *have* 1 *as target, i.e.,* $t(a) = 1$. *Every recognizable language* $L \subseteq \mathcal{F}\Sigma$ *of the free* PRO $\mathcal{F}\Sigma$ *whose elements are trees, i.e., such that* $t \colon \epsilon \to 1$ *in* $\mathcal{F}\Sigma$ *holds for all* $t \in L$*, is the language of a tree automaton.*

The proof amounts to the construction of an automaton whose runs encode evaluation of a functor to a finite category in $\mathscr{M}on$ that recognizes the language.

Proof. Assume that the language L is recognized by a functor $G \colon \mathcal{F}\Sigma \to \mathbb{E}$ to a finite category $\mathbb{E}$ in $\mathscr{M}on$. Given the signature (Σ, s, t), the language $L \subseteq \mathcal{F}\Sigma$, and the functor $G \colon \mathcal{F}\Sigma \to \mathbb{E}$ that recognizes L, we construct a corresponding automaton $\mathcal{A}_{\Sigma,L,G} = (Q, \Gamma, \delta, I, F)$ as candidate for an automaton that accepts the language L.

– The set of states is $Q = E_1$, the set of arrows of $\mathbb{E}$, whose elements are referred to as *actions*.
– The working signature
$$\Gamma = \bigcup_{a \in \Sigma} \{a\} \times E_1^{|s(a)|}$$
is the set of Σ-symbols whose input ports are labelled with actions.
– The source function $s' \colon \Gamma \to Q^*$ is defined by
$$(s'(a, \boldsymbol{f}))_i = f_i$$
for all symbols $(a, \boldsymbol{f}) \in \Gamma$ of the signature and all $i \in \{1, \ldots, |s(a)|\}$ where $\boldsymbol{f} = f_1 \cdots f_{|s(a)|}$ is the list of action labels on input ports.
– The target function $t' \colon \Gamma \to Q^*$ maps each symbol $(a, \boldsymbol{f}) \in \Gamma$ with the action word $\boldsymbol{f} = f_1 \cdots f_{|s(a)|}$ as source to
$$t'(a, \boldsymbol{f}) = t'(a, \boldsymbol{f})_1 = \left(\bigoplus_{k=1}^{|s(a)|} f_k \right); G(a),$$ [11]
the action of the (first and only) output port, which is the parallel composition of all actions f_k that are present at input ports, followed by the action of the symbol $a \in \Sigma$, i.e., the image of a under the functor G.
– The transition function $\delta \colon \Gamma \to \Sigma$ projects each symbol $(a, \boldsymbol{f}) \in \Gamma$ of the working signature to the first component a.
– The set of initial state words is the singleton $\{\epsilon\}$, containing the empty word, as required by the definition of tree automaton.
– Finally, a state is final iff it belongs to the image of the language under G, i.e., $F = G_1(L)$.

Runs of the automaton now merely evaluate the functor G while ascending the tree, storing results for sub-trees in the states of the output port of gates at the roots of the respective sub-trees of the run. We omit the straightforward induction on the tree structure, showing that, for each tree t, there is exactly one run r that is "projected" to t via $\mathcal{F}\delta$, i.e., s.t. $\mathcal{F}\delta_1(r) = t$, and that the codomain of this run is $G(t)$, the value of the functor applied to the tree, which is a final state (word) iff $t \in L$.

The above automata construction will be extended in the proof of Theorem 2.

C Theorem 2 via forest decompositions of circuit diagrams

The remainder of the appendix describes the forest structure of PLACIDS for signatures without terminators and initiators and gives the proof of Theorem 2.

[11] Thus, if the number of input ports of the symbol a is $0 = |s(a)|$, we have $t'(a, \boldsymbol{f}) = \mathrm{id}_O; G(a) = G(a)$.

C.1 The entangled forest structure of PLACIDs

The main observation is that absence of both terminators and initiators in signatures results in an entangled forest structure of planar acyclic circuit diagrams: the roots of the forests are gates whose first output ports are wired

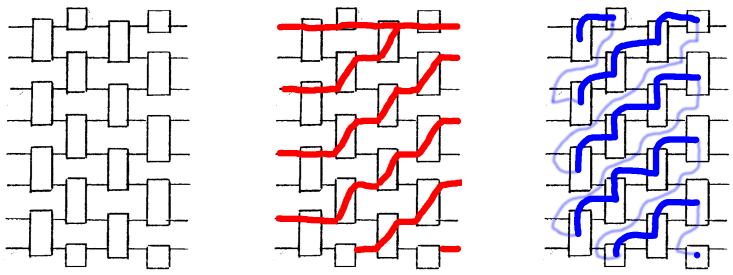


Fig. 3. A brick wall, its forest structure, and the left-most-upper-most-first order

to the "output interface"; in particular, gates at the right border of drawings are output gates. The children of a gate are all those gates whose first output ports are connected to some of its input ports. An example of such a forest structure is illustrated in the middle of Figure 3, using a small "brick wall" of Bossut. Its forest consists of a first tree with proper branching "on top", followed by a number of "diagonal" linear branches "underneath"; a technical complication arises from the fact that the trees are intertwined, reminiscent of ivy that entangles trees.

Nevertheless, the idea of the tree automaton construction from the proof of Lemma 3, namely that runs evaluate a recognizing functor on all (sub-)trees, can be re-used. More detailed, after runs of the automaton have evaluated the functor on all "trees" of the skewed forest, the word automaton for the final state words collects and composes the results for all "trees". The main technical difficulty is the "skew factor", which we shall overcome by retaining enough information about the "offset" of gates relative to the bottom of the circuit diagram in the states of the automaton. To describe this formally, we first give a detailed account of the skewed forest structure of PLACIDs, which will serve us in the proof of Theorem 2.

C.2 The skewed forest structure of PLACIDs, formally

We order gates according to their position in the list of gates that is obtained by repeatedly removing the left-most-upper-most gate from a given diagram and appending it to the list of gates, starting from the empty list. An example of this ordering of gates is illustrated in the rightmost drawing of Figure 3: by following the blue line from top-left to bottom-right, we are visiting the gates in upper-most-left-most-first order. The corresponding formal property of PLACIDs is the following.

Lemma 4 (Canonical factorization). *For a $\mathfrak{C}$-coloured signature (Σ, s, t) such that $s(a) \neq \epsilon \neq t(a)$, for all $a \in \Sigma$, every* PLACID *$p \in \mathcal{F}\Sigma$ has a unique factorization of the form*

$$\begin{aligned} p &= \fatsemi_{i=1}^{n} (\mathrm{id}_{u_i} \oplus a_i \oplus \mathrm{id}_{v_i}) \\ &= (\mathrm{id}_{u_1} \oplus a_1 \oplus \mathrm{id}_{v_1}); \cdots ; (\mathrm{id}_{u_n} \oplus a_n \oplus \mathrm{id}_{v_n}) \end{aligned} \tag{1}$$

of symbols $a_i \in \Sigma$ with identities on colour words $u_i, v_i \in \mathfrak{C}^*$ ("above" and "below", resp.) such that there does not exist any index $j \in \{1, \ldots, n-1\}$ such that $u_{j+1} s(a_{j+1})$ is a prefix of u_j.

Proof. The result follows by a straightforward induction on the number of gates.



Fig. 4. Two different factorizations of the same circuit diagram

The left one of the factorizations in Figure 4 is not in left-most-upper-most-first order. The canonical factorization can be obtained by repeatedly switching a gate at which the condition of Lemma 4 is violated with its successor gate. The canonical factorization on the right in Figure 4 is obtained by pushing the first and third gate one position to the right.

We shall now formalize the skewed forest structure of a canonical factorization

$$\big(\mathrm{id}_{u_1} \oplus a_1 \oplus \mathrm{id}_{v_1}\big); \cdots ; \big(\mathrm{id}_{u_n} \oplus a_n \oplus \mathrm{id}_{v_n}\big).$$

For an index $i \in \{1, \ldots, n\}$, the sequential factor $\mathrm{id}_{u_i} \oplus a_i \oplus \mathrm{id}_{v_i}$ is called *gate* of index i (or *gate i*, for short), the word v_i is the *lower offset* of gate i, and u_i is its *upper offset*. The gate of index i is an *output gate* if all following upper offsets are longer, i.e., if $|u_j| > |u_i|$, for all $j > i$. For example, the output gates in Figure 3 are those with indices $6, 10, 14, 17$ and 18. The output gates form the roots of the skewed forest structure.

Given a gate i that is not an output gate, its *parent* is the gate with the smallest index j such that $|u_j| \leq |u_i|$ (and thus, the upper offset u_i must be a proper prefix of $u_j s(a_j)$ because the characteristic property of canonical factorizations would be violated otherwise); reciprocally, gate i is the *child* of gate j if gate j is the the parent of gate i. Note that among the children of a gate i, the one with the highest index, called the *youngest child*, has index $i - 1$. Gate i is a *leaf* if it is not the parent gate of any other gate. Note that leaves do not necessarily lie at the left boundary of drawings; e.g., in Figure 3, gates 15 and 18 are leaves.

With these formal definitions of the skewed forest structure in place, we shall factorize the placid $p \colon x \to y$ according to the format

$$p = \fatsemi_{m=1}^{|y|} \mathrm{id}_{y_{[1,m-1]}} \oplus \mathcal{T}_m$$

where $\mathrm{id}_{y_{[1,m-1]}}$ is the prefix of y of length $m-1$ and the PLACID $\mathcal{T}_m$ is what we shall define as the *skewed pseudo-tree* at m, for each number $m \in \{1,\ldots,|y|\}$; we call such a number $m \in \{1,\ldots,|y|\}$ an *output wire* of the PLACID p. The intuitive idea behind the subsequent definition of skewed pseudo-trees $\mathcal{T}_m$ is that the tree of an output gate i with all its children and indirect descendents can be assigned a unique "address" $m \in \{1,\ldots,|y|\}$, namely $|u_i|+1$. For example, the first output ports of (roots of) skewed trees in Figure 4 have addresses $1,2,4,6,8$, the tree $\mathcal{T}_8$ is just the output gate 18 and $\mathcal{T}_1$ is the top-most proper tree of the first six gates, but "offset" by four parallel wires "below". All other output wires, i.e., those that do not correspond to any first output port of some output gate, will be assigned "dummy trees" $\mathcal{T}_m$ that consist of identities that "cut trough the ivy that entangles the skewed trees", but "do not separate parents from their children". Keeping the *upper tabs* $y_{[1,m-1]}$ separately will turn out to be convenient. We now give the final formal definition—skewed (pseudo-)trees—that we shall use in the proof of the theorem.

For any gate i, let N_i be the total number of children of gate i, and moreover let $\{\mathcal{G}_i(1),\ldots,\mathcal{G}_i(N_i)\}$ be the set of indices of all its children such that $l<k$ implies $\mathcal{G}_i(l)<\mathcal{G}_i(k)$; moreover, for each child gate $l \in \{\mathcal{G}_i(1),\ldots,\mathcal{G}_i(N_i)\}$ of gate i, let $\bar{u}_l$ be the suffix of the upper offset u_l of the child such that $u_l = u_i\bar{u}_l$. The *skewed tree* of a gate i, is defined by

$$\mathcal{T}[i] = \begin{cases} a_i \oplus \mathrm{id}_{v_i} & \text{if gate } i \text{ is a leaf} \\ \fatsemi_{l=1}^{N_i} \mathrm{id}_{\bar{u}_{\mathcal{G}_i(l)}} \oplus \mathcal{T}[\mathcal{G}_i(l)]; (a_i \oplus \mathrm{id}_{v_i}) & \text{if } N_i > 0. \end{cases} \tag{2}$$

We extend the definition of skewed trees to output ports of all gates. The *skewed pseudo-tree* of *output port* $l \in \{1,\ldots,|t(a_i)|$ of a gate i is

$$\mathcal{T}[i.l] = \begin{cases} \mathcal{T}[i] & \text{if } l = 1 \\ \mathrm{id}_{t(a_i)_{[l,|t(a_i)|]}} \oplus \mathrm{id}_{v_i} & \text{if } l > 1 \text{ where } t(a_i)_{[l,|t(a_i)|]} \text{ is} \\ & \quad \text{the suffix of } t(a_i) \text{ of length } 1+|t(a_i)|-l. \end{cases} \tag{3}$$

In the end, we are interested in referencing skewed pseudo-trees via wires at the output interface of the PLACID p through a number $m \in \{1,\ldots,|y|\}$. The first port of any gate i, denoted $(i.1)$, is an *output wire* if gate i is an output gate; its *output address* is $@(i.1) = |u_i|+1$, provided that i is an output gate. For "secondary" ports $l \in \{2,\ldots,|t(a_i)|\}$, it is slightly non-trivial to formally describe the situations in which they are output wires. The set of *causal successors above* an output port $(i.l)$ (with $l \geq 2$) is

$$\lceil i.l \rceil = \left\{ j > i \;\middle|\; \begin{array}{l} t(a_i)_{[l,|t(a_i)|]} v_i \text{ is a suffix of } v_j \\ \quad \text{where } t(a_i)_{[l,|t(a_i)|]} \text{ is the suffix of } t(a_i) \text{ of length } 1+|t(a_i)|-l. \end{array} \right\}$$

If $\lceil i.l \rceil$ is not empty, the largest index in this set will be denoted by $(i.l)^{\gg}$; if $\lceil i.l \rceil = \emptyset$, we put $(i.l)^{\gg} = i$. The *potential address* of the port $(i.l)$ is

$$?(i.l) = |u_{(i.l)^{\gg}} t(a_{(i.l)^{\gg}}) v_{(i.l)^{\gg}}| - (|t(a_i) v_i| - l).$$

The port $(i.l)$ is an *output wire* if $|u_k| \geq ?(i.l)$ holds for all $k > (i.l)^{\gg}$ and then its *output address* is $@(i.l) = ?(i.l)$. Intuitively, an output port l of a gate i is an output wire if it is not connected to the input port of any other gate. Finally, the *skewed pseudo-tree* of an output address $m \in \{1, \ldots, |y|\}$ where $y = y_1 \cdots y_{|y|}$ is defined by

$$\mathcal{T}_m = \begin{cases} \mathrm{id}_{y_m} \oplus \mathrm{id}_{\mathrm{dom}(p_2)} & \text{if } p = p_1 \oplus \mathrm{id}_{y_m} \oplus p_2 \\ \mathcal{T}[i.l] & \text{if } m = @(i.l). \end{cases}$$

The first case covers the case of wires that lead from the input interface to the output interface of the PLACID, including, but not limited to the case of $n = 0$; otherwise, there is a unique output port that has the relevant output address. By construction, we obtain

$$p = \mathop{\fatsemi}_{m=1}^{|y|} \mathrm{id}_{y_{[1,m-1]}} \oplus \mathcal{T}_m \tag{4}$$

where $y_{[1,m-1]}$ is the prefix of y of length $m-1$.

C.3 The proof of Theorem 2

We shall now prove the second result, which extends the soundness check for the proposed notion of recognizability to PLACID languages of Bossut's automata.

Theorem 2. *Let $\mathfrak{C} \neq \emptyset$ be a set of colours, let (Σ, s, t) be a $\mathfrak{C}$-coloured signature without terminators and initiators, and let $L \subseteq \mathcal{F}\Sigma$ be a language of* PLACIDS. *The language L is recognizable if, and only if, it is automaton acceptable.*

Proof. Assume that the language L is recognized by a functor $G\colon \mathcal{F}\Sigma \to \mathbb{E}$ to a finite category $\mathbb{E}$ in $\mathscr{Mon}$. Given the signature (Σ, s, t), the language $L \subseteq \mathcal{F}\Sigma$, and the functor $G\colon \mathcal{F}\Sigma \to \mathbb{E}$ that recognizes L, we construct a corresponding automaton $\mathcal{A}_{\Sigma,L,G} = (Q, \Gamma, \delta, I, F)$ as candidate for an automaton that accepts the language L.

- The set of states is the Cartesian product $Q = \mathfrak{C} \times (E_0 \times E_1)$ such that a state $(\mathfrak{c}, (X, f))$ consists of a colour $\mathfrak{c}$ of the signature Σ of the language, an object $X \in E_0$ from the finite category $\mathbb{E}$, called *strut*, and an arrow $f \in E_1$ from the finite category $\mathbb{E}$, referred to as *action*.
- The working signature

  $$\Gamma = \bigcup_{a \in \Sigma} \{a\} \times (E_0 \times E_1)^{|s(a)|}$$

  is the set of Σ-symbols with a strut and action label on each input port.
- The source function $s'\colon \Gamma \to Q^*$ of the working signature is defined by

  $$(s'(a, \boldsymbol{l}))_i = (s(a)_i, l_i)$$

  for all symbols $(a, \boldsymbol{l}) \in \Gamma$ of the working signature and all input port numbers $i \in \{1, \ldots, |s(a)|\}$ where $s(a) = s(a)_1 \cdots s(a)_{|s(a)|}$ and $\boldsymbol{l} = l_1 \cdots l_{|s(a)|}$.

– The target function $t'\colon \Gamma \to Q^*$ maps each symbol $(a, \boldsymbol{l}) \in \Gamma$ with $l_i = (X_i, f_i)$ (for all $i \in \{1, \ldots, |s(a)|\}$) to $t'(a, \boldsymbol{l})$ such that, for all $j \in \{1, \ldots, |t(a)|\}$,

$$t'(a, \boldsymbol{l})_j = \Big(t(a)_j, \big(Y_j, g_j\big)\Big)$$

where

- the strut

$$Y_j = G\Big(t(a)_{[j+1,|t(a)|]}\Big) \oplus X_{|s(a)|}$$

is the strut of the last input port of $(a, \boldsymbol{l})$ augmented with the image of the suffix of $t(a)$ of length $|t(a)| - j$ under G and

- the action g_j depends on the output port number j: we put

$$g_j = \begin{cases} \Big(\mathop{\fatsemi}_{i=1}^{|s(a)|} \mathrm{id}_{G\left(s(a)_{[1,i-1]}\right)} \oplus f_i\Big); \big(G(a) \oplus \mathrm{id}_{X_{|s(a)|}}\big) & j = 1 \\ \mathrm{id}_{G(t(a)_{[j,|t(a)|]})} \oplus \mathrm{id}_{X_{|s(a)|}} & j > 1 \end{cases}$$

where $s(a)_{[1,i-1]}$ is the prefix of $s(a)$ of length $i - 1$ and $t(a)_{[j,|t(a)|]}$ is the suffix of $t(a)$ of length $1 + |t(a)| - j$. Thus, the action of the first output port is the sequential composition of all actions f_i that are present at input ports—each one extended by the image of the first $i - 1$ input ports of the gate type $a \in \Sigma$ under G "on top"—followed by the action of the symbol $a \in \Sigma$ according to the functor G—extended in parallel with the strut of its last input port"below"; the case $j = 1$ corresponds to Equation (2). Thus, for output port numbers $j \in \{2, \ldots, |t(a)|\}$, we use a "dummy action" of an identity of the image of all output ports under G starting from and including the j-th one—composed in parallel with the identities of the strut of the last input port; the case $j > 1$ corresponds to the second case in Equation (3).

– The transition function δ projects each symbol $(a, \boldsymbol{l}) \in \Gamma$ of the working signature to its first component a.
– The set of initial state words is

$$\left\{ \Big(\mathfrak{c}_1, \big(Y_1, \mathrm{id}_{G(\mathfrak{c}_1) \oplus Y_1}\big)\Big) \cdots \Big(\mathfrak{c}_n, \big(Y_n, \mathrm{id}_{G(\mathfrak{c}_n) \oplus Y_n}\big)\Big) \;\middle|\; \forall i \in \underline{n}.\, Y_i = G(\mathfrak{c}_{i+1} \cdots \mathfrak{c}_n) \right\}$$

where $\mathfrak{c}_{i+1} \cdots \mathfrak{c}_n = \epsilon$ if $i = n$, and thus $Y_n = \mathrm{O}$ if $n > 0$.
– The set of final state words is

$$\left\{ \Big(\mathfrak{c}_1, \big(Y_1, g_1\big)\Big) \cdots \Big(\mathfrak{c}_n, \big(Y_n, g_n\big)\Big) \;\middle|\; G_1(L) \ni \mathop{\fatsemi}_{i=1}^{n} \mathrm{id}_{G(\mathfrak{c}_1 \cdots \mathfrak{c}_{i-1})} \oplus g_i \right\}$$

where $\mathfrak{c}_1 \cdots \mathfrak{c}_{i-1} = \epsilon$ if $i = 1$.

The set of initial state words $I \subseteq Q^*$ is regular as its reversal I' can be accepted by an automaton $\mathcal{A}'_I$ with states $Q'_I = E_0$, unique initial state $\mathrm{O} \in I'_I$, final states $F'_I = Q'_I = E_0$, and the partial transition function δ'_I given by

$$\delta'_I\Big(X, \big(\mathfrak{c}, (Y, g)\big)\Big) = \begin{cases} G(\mathfrak{c}) \oplus X & \text{if } Y = X \text{ and } g = \mathrm{id}_{G(\mathfrak{c}) \oplus Y} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

This automaton initializes the struts and actions of all input wires of a run, where struts retain partial information about lower offsets of gates.

The set of final state words $F \subseteq Q^*$ is accepted by the automaton $\mathcal{A}_F$ with states $Q_F = E_0 \times E_1$, set of initial states $I_F = \{(\mathrm{O}, \mathrm{id}_Z) \mid Z \in C_0\}$, set of final states $F_F = \{(X, f) \mid f \in G_1(L)\}$, and the partial transition function δ_F given by

$$\delta_F\Big((X,f),\big(\mathfrak{c},(Y,g)\big)\Big) = \begin{cases} \Big(X \oplus G(\mathfrak{c}), \big(f;(\mathrm{id}_X \oplus g)\big)\Big) & \text{if } \mathrm{cod}(f) = \mathrm{dom}(\mathrm{id}_X \oplus g) \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Intuitively, the automaton $\mathcal{A}_F$ first guesses the G-image of the domain of the PLACID to be accepted or rejected and then composes successively the evaluations of the functor G on skewed pseudo-trees of the PLACID that are stored in the actions at the output wires of the run.

The definition of the transition function $\delta\colon \Gamma \to \Sigma$ is *deterministic*: for a given symbol $a \in \Sigma$ of the language signature and a state word $\tilde{q}$ of length $\tilde{q} = |s(a)|$, there is exactly one matching symbol in Γ; moreover, for each PLACID $p\colon x \to y$, there is exactly one initial state word $\tilde{q} \in I_I$ that is mapped to x via $\mathcal{F}\delta$, namely the sequence $\tilde{q} = q_1 \cdots q_{|x|}$ with

$$q_i = \Big(x_i, \big(G\left(x_{[i+1,|x|]}\right), \mathrm{id}_{G\left(x_{[i,|x|]}\right)}\big)\Big)$$

for all $i \in \{1, \ldots, |x|\}$ where $x_{[i+1,|x|]}$ and $x_{[i,|x|]}$ are the suffixes of x of length $|x| - i$ and $1 + |x| - i$, respectively. Thus, for each PLACID $p\colon x \to y$, there is a unique run $r\colon \tilde{q} \to \tilde{q}'$ with $\mathcal{F}\delta_1(r) = p$.

Finally, by definition of the transition function δ and using Equations 2 and 3, for every run $r\colon \tilde{q} \to \tilde{q}'$ with $\tilde{q}' = q'_1 \cdots q'_{|y|}$ and $q'_m = (y_m, (Y_m, g_m))$ for each output address m, we have $g_m = G(\mathcal{T}_m)$; moreover, the run is accepting if, and only if, $G(p) \in G(L)$, by Equation 4 and the construction of $\mathcal{A}_F$, since

$$G(p) = \fatsemi_{m=1}^{|y|} \mathrm{id}_{G(y_{[1,m-1]})} \oplus G(\mathcal{T}_m).$$