Algebraic Characterization of FO-Definable Languages of Higher-Dimensional Automata

Enzo Erlich
Université Paris Cité, CNRS, IRIF, F-75013, Paris, France
EPITA Research Laboratory (LRE), Paris, France

Jérémy Ledent
Université Paris Cité, CNRS, IRIF, F-75013, Paris, France

Krzysztof Ziemiański
Institute of Mathematics, University of Warsaw, Poland

Abstract

Higher-dimensional automata (HDA) are a model of concurrency that models simultaneous execution of events using higher dimensional cells. HDA recognize languages of pomsets, a generalization of finite words whose letters are partially ordered. We prove a new algebraic characterization of HDA languages: a language of pomsets is regular if and only if it is the inverse image of a functor from the category of pomsets into a finite category. Furthermore, the language is definable in first-order logic exactly when it is recognized by an aperiodic category, generalizing the McNaughton-Papert theorem to HDA languages. We also investigate a notion of counter-free HDA, and show that if a language is accepted by a counter-free HDA, it must be definable in first-order logic. The converse, however, is still open.

2012 ACM Subject Classification Theory of computation → Concurrency; Theory of computation → Automata extensions; Theory of computation → Logic and verification

Keywords and phrases Higher-dimensional automata, Pomset languages, McNaughton-Papert theorem, Counter-free HDA, Aperiodic category

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2026.32

Related Version *Full Version*: https://arxiv.org/abs/2605.25253

1 Introduction

Over finite words, the class of languages definable in First-Order (FO) logic is a strict subclass of regular languages, which admits several characterizations. For a language $L \subseteq \Sigma^*$, the following are equivalent:

1. L is recognized by a star-free regular expression,
2. L is recognized by an aperiodic monoid,
3. L is recognized by a counter-free automaton,
4. L is definable in first-order logic (FO),
5. L is definable in linear temporal logic (LTL).

The proof of (1) ⇔ (2) is due to Schützenberger [16]; the equivalence between (2), (3) and (4) was proved by McNaughton and Papert [14]; and Kamp [12] showed the last equivalence (4) ⇔ (5).

In this paper, we are interested in extending these results to *Higher-Dimensional Automata* (HDA). HDA are a model of concurrency that enriches standard finite-state automata with higher-dimensional cells, allowing to specify that some events may occur simultaneously. Although they were originally introduced by Pratt [15] in 1991, a proper language theory for HDA was only developed recently [9]. Many classic results of automata theory carry over to HDA languages [10, 11, 3, 2, 6]. Most relevant for our purpose is the work developed

in [6], which proves a variant of Kamp's theorem for HDA. In that paper, the authors define a temporal logic on pomsets that has the same expressive power as first order logic. This is interesting for automatic verification of concurrent programs: model a program using an HDA; specify its desired behavior using LTL on pomsets; and algorithmically check that the program satisfies its intended specification.

Following McNaughton and Papert, our goal is to find two additional characterizations of the class of FO-definable HDA languages: (i) via an algebraic characterization akin to aperiodic monoids, and (ii) using a notion of *counter-free HDA*. Extending these classic results of automata theory to the higher-dimensional setting shows that FO and LTL are robust specification languages for model-checking HDA properties. We leave open the question of finding a notion of star-free regular expression for HDA that captures the same class of languages.

Pomset languages of HDA. While standard finite state automata recognize languages of words, HDA recognize languages of *partially ordered multi-sets*, or *pomsets*. A word can be viewed as a special case of pomset, where the letters are totally ordered; such pomsets correspond to sequential executions. But in general, two letters in a pomset need not be ordered: when two letters are incomparable, we think of them as two events occurring concurrently. More precisely, events are ordered according to Lamport's happens-before partial order [13]. We say that event a *precedes* event b, written $a \to b$, if event a is terminated before b starts. Thus, two events are incomparable (a.k.a. *concurrent*) when the two intervals during which they are executed overlap.

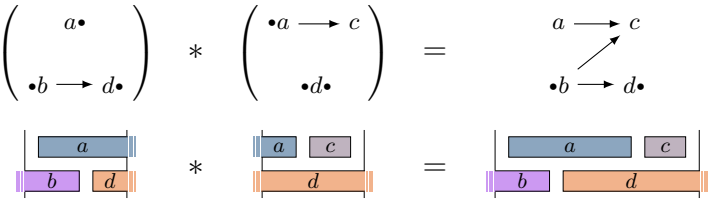


Figure 1 Gluing of pomsets and their interval representation.

Additionally, pomsets are equipped with *interfaces*. The *starting interface* is a set of events that are already started at the beginning of the computation. They are required to be minimal w.r.t. the precedence partial order. Symmetrically, the *terminating interface* is a set of events, maximal w.r.t. precedence, that are still ongoing at the end of the computation. When two pomsets P and Q have matching interfaces (that is, the terminating interface of P is the same as the starting interface of Q), we can *glue* them together to obtain a new pomset $P * Q$.

Figure 1 depicts the gluing of two pomsets, viewed as partial orders (top), and as intervals (bottom). Events that belong to the starting (resp., terminating) interface are marked with a dot to the left (resp., right) of the letter. Gluing plays the role of word concatenation. However, since it is defined only for pomsets with matching interfaces, the underlying algebraic structure is that of a category rather than a monoid. We write iiPoms for the category whose objects are interfaces, and morphisms are pomsets.

Contributions. We first prove in Section 3 the following algebraic characterization of HDA languages: a language of pomsets is regular if and only if it is the inverse image of a functor from iiPoms into a finite category (Theorem 13). The left-to-right implication follows the standard construction of the syntactic monoid associated with a language; in our setting, this is now a *syntactic category*. The converse direction requires a bit more work however.

While, for finite words, a morphism into a finite monoid can immediately be viewed as an automaton using the Cayley graph of the monoid; this is not the case for HDA. What we get instead is what we call an iiPoms-module. To turn it into an HDA, we rely on the canonical suffix presentation of a language of pomsets [11].

Then, in Section 4 we investigate the class of languages that are recognized by a functor into an aperiodic category. A category is aperiodic when every endomorphism x is such that $x^n = x^{n+1}$ for some n. We show in Theorem 33 that the class of languages recognized aperiodic categories is also the class of FO-definable languages. To prove this, we use the so-called ST-decomposition of a pomset. This allows us to view a language of pomsets as a language of words, and relying on results from [6], we can transport the McNaughton-Papert theorem from words to pomsets.

Lastly in Section 5, we define a notion of counter-free HDA, and show that every language accepted by a counter-free HDA is aperiodic. However, the converse direction remains an open question. This is because the geometric structure of HDAs interacts with counter-freeness in a non-trivial way. We explain why a naive attempt at solving the open question fails, and conclude with an example of a pomset language that exhibits some form of counting, despite being aperiodic.

2 Preliminaries

In this section, we briefly recall the definitions about pomsets and HDA that we will need in the rest of the paper. We encourage readers unfamiliar with pomset languages of HDA to read a more gentle introduction to those notions, e.g. [10]. We also rely on a few notions from category theory, namely *categories* and *functors*. The necessary definitions can be found in the first chapter of any category theory textbook, e.g. [4].

Notations. For a category $\mathcal{C}$, we write $\mathsf{Ob}(\mathcal{C})$ for the objects of $\mathcal{C}$, and $\mathsf{Mor}(\mathcal{C})$ for the morphisms. Composition is written in diagrammatic order, i.e., given morphisms $f : A \to B$ and $g : B \to C$, their composition is denoted fg or $f \fatsemi g$ rather than $g \circ f$.

2.1 Higher Dimensional Automata (HDA)

Fix a finite alphabet Σ. An HDA is built from higher-dimensional cells. Each cell is labeled with a finite list of elements of Σ, that we think of as the currently running events. This totally ordered list of events is called a concurrency list, or *conclist*.

▶ **Definition 1** (Conclist). *A conclist $(U, \dashrightarrow, \lambda)$ consists of a finite set U equipped with a strict total order $\dashrightarrow$, and a labeling function $\lambda : U \to \Sigma$.*

A conclist may have several occurrences of the same letter $a \in \Sigma$ running in parallel, in which case the total order $\dashrightarrow$ is crucial to differentiate the corresponding events. We write CList for the set of conclists.

▶ **Definition 2** (HDA). *An HDA is a tuple $(X, \mathsf{ev}, \{\delta^0_{A,U}, \delta^1_{A,U} \mid A \subseteq U \in \mathsf{CList}\}, \bot, \top)$, where:*

- *X is a finite set of cells, and $\mathsf{ev} : X \to \mathsf{CList}$ assigns a conclist to each cell. For a conclist U, we write $X[U] = \{x \in X \mid \mathsf{ev}(x) = U\}$ for the set of cells of type U.*
- *For each $U \in \mathsf{CList}$ and $A \subseteq U$, $\delta^0_{A,U} : X[U] \to X[U - A]$ is called the lower face map, and $\delta^1_{A,U} : X[U] \to X[U - A]$ the upper face map. We often omit the subscript U when it is clear from context. The face maps must satisfy the precubical identities: for all $\mu, \nu \in \{0, 1\}$ and $A \cap B = \varnothing$, we require $\delta^\mu_A \delta^\nu_B = \delta^\nu_B \delta^\mu_A$.*
- *$\bot, \top \subseteq X$ are sets of initial and accepting cells, respectively. Often, $\bot \subseteq X[\varnothing]$.*

We often denote an HDA by its underlying set X. The *dimension* of an HDA is the maximal number of events active in a cell, $\dim(X) = \max\{|\mathsf{ev}(x)| \mid x \in X\}$. A path in an HDA is a sequence of cells, where consecutive cells can be connected in two ways: either by starting some events, that is, moving from a lower face of x to the cell x; or by terminating some events, that is, moving from a cell x to an upper face of x.

▶ **Definition 3** (Path). *A path in an HDA X is a sequence $\alpha = (x_0, \varphi_1, x_1, \dots, x_{n-1}, \varphi_n, x_n)$, where each $x_i \in X$, each $\varphi_i \in \{\nearrow^A, \searrow_A \mid A \in \mathsf{CList}\}$, such that for all $1 \le i \le n$, either*

- *$\varphi_i = \nearrow^A$ for some $A \subseteq \mathsf{ev}(x_i)$, and $x_{i-1} = \delta^0_A(x_i)$, or*
- *$\varphi_i = \searrow_A$ for some $A \subseteq \mathsf{ev}(x_{i-1})$, and $\delta^1_A(x_{i-1}) = x_i$.*

The *source* and *target* of a path are $\mathsf{src}(\alpha) = x_0$ and $\mathsf{tgt}(\alpha) = x_n$, respectively. A path is *accepting* when $\mathsf{src}(\alpha) \in \bot$ and $\mathsf{tgt}(\alpha) \in \top$. To define the language accepted by an HDA, we first need to introduce pomsets.

▶ **Example 4.** A 2-dimensional HDA is depicted in Figure 2. It consists of four 2-dimensional cells, depicted as gray squares, twelve 1-dimensional cells, depicted as edges, and nine 0-dimensional cells, depicted as vertices. Cells are labeled with the associated conclist. For example, the bottom-left square has two events a and b running in parallel, so its associated conclist is $U = \{a \dashrightarrow b\}$, which we sometimes depict vertically as $\left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right]$. From this square, the lower face map $\delta^0_{\{a\},U}$ assigns the leftmost b-labeled edge, where event a has not started yet; and the upper face map $\delta^1_{\{a\},U}$ assigns the middle b-labeled edge, where event a is terminated. Similarly, the two other face maps $\delta^0_{\{b\},U}$ and $\delta^1_{\{b\},U}$ assign the two horizontal a-labeled boundaries of the $\left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right]$-labeled square.

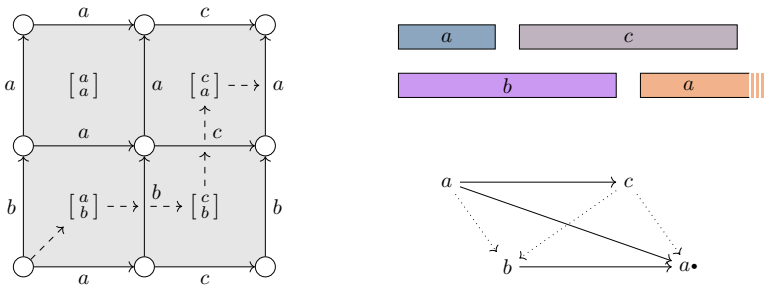


Figure 2 An example of an HDA (left). Cells are labeled with associated conclists. The dashed arrows form a path, the pomset recognized by this path is illustrated at the right-hand side. The bullet at a reflects the fact that the second occurrence of a has not yet been terminated.

2.2 Pomset languages

HDA recognize languages of partially ordered multi-sets, or *pomsets*, a.k.a. labeled partial orders. In fact, we require additional structure: an *event order* $\dashrightarrow$ to distinguish autoconcurrent events; source and target interfaces to allow for concatenation (or *gluing*) of pomsets; and the partial order is always assumed to be an interval order. For conciseness, we overload the word "pomset", always assuming the whole structure.

▶ **Definition 5** (iiPomset). *A pomset over Σ is a tuple $(P, <_P, \dashrightarrow_P, \lambda_P, S_P, T_P)$ where:*

- *$(P, <_P)$ is a finite set equipped with a strict partial interval order called precedence.*
- *$\dashrightarrow_P$ is an acyclic relation on P called the event order, such that for all $x \neq y \in P$, if $x \not<_P y$ and $y \not<_P x$, then either $x \dashrightarrow_P y$ or $y \dashrightarrow_P x$.*

- $\lambda_P : P \to \Sigma$ *is a labeling function.*
- $S_P, T_P \subseteq P$ *are the source (resp. target) interfaces, such that elements of* S_P *(resp.* T_P*) are minimal (resp. maximal) elements w.r.t.* $<_P$*.*

When there is no ambiguity, we denote $<_P, \dashrightarrow_P$ *and* λ_P *as* $<$*,* $\dashrightarrow$ *and* λ*.*

An example pomset is depicted in Figure 2. In pictures, precedence $<_P$ is represented using full arrows, and event order $\dashrightarrow_P$ as dotted arrows, with transitive arrows omitted. The source (resp. target) interfaces are represented as bullets before (resp. after) the label of the event.

A pomset P is called *discrete* when it has an empty precedence order. In that case, the event order $\dashrightarrow$ is a strict total order. Given a conclist $(U, \dashrightarrow, \lambda) \in \mathsf{CList}$ and $A \subseteq U$, we define the following discrete pomsets:

- $\mathsf{id}_U = (U, \varnothing, \dashrightarrow, \lambda, U, U)$ is an *identity*,
- ${}_A\!\uparrow U = (U, \varnothing, \dashrightarrow, \lambda, U - A, U)$ is a *starter*,
- $U\!\downarrow_A = (U, \varnothing, \dashrightarrow, \lambda, U, U - A)$ is a *terminator*.

Intuitively, id_U is an elementary pomset where the events in U are currently active in parallel. In ${}_A\!\uparrow U$, we just started the events A, so that the events U are now active. In $U\!\downarrow_A$, the events U were active, and we terminate the events A. In small dimensions, we write discrete pomsets using a bracket notation, where the event order implicitly goes from top to bottom. For instance, $[\begin{smallmatrix}\bullet^a\\ b\bullet\end{smallmatrix}]$ denotes the starter ${}_{\{b\}}\!\uparrow U$, where U is the two element conclist $a \dashrightarrow b$.

The source and target interfaces S_P, T_P, together with the event order $\dashrightarrow$ and labeling map λ, can be viewed as conclists. For $U, V \in \mathsf{CList}$, we write $\mathsf{iiPoms}(U, V)$ for the set of pomsets with source interface U and target interface V (up to isomorphism). We define the *gluing* operation on pomsets, $* : \mathsf{iiPoms}(U, V) \times \mathsf{iiPoms}(V, W) \to \mathsf{iiPoms}(U, W)$.

▶ **Definition 6** (Gluing). *Let* $P \in \mathsf{iiPoms}(U, V)$ *and* $Q \in \mathsf{iiPoms}(V, W)$*. Let* $f : T_P \to S_Q$ *be the unique conclist isomorphism between the interfaces of* P *and* Q*. Then, we define the pomset* $P * Q = (R, <_R, \dashrightarrow_R, \lambda_R, S_R, T_R)$*, where*

$$R = (P \sqcup Q)/_{x \equiv f(x)} \qquad \lambda_R = \lambda_P \cup \lambda_Q$$
$$<_R = <_P \cup <_Q \cup (P - T_P) \times (Q - S_Q) \qquad S_R = S_P$$
$$\dashrightarrow_R = \dashrightarrow_P \cup \dashrightarrow_Q \qquad T_R = T_Q$$

See Figure 1 for an example of gluing (event order implicitly goes from top to bottom). Gluing is associative and id_U is a neutral element, this data assembles into a category iiPoms, whose objects are conclists, morphisms are pomsets, and composition is gluing.

The *dimension* of a pomset P is the size of a maximal $<_P$-antichain. We write $\mathsf{iiPoms}_{\leq k}$ for the subcategory of iiPoms consisting of pomsets of dimension at most k. Note that $\mathsf{iiPoms}_{\leq k}$ has a finite number of objects: they are conclists of length at most k.

▶ **Definition 7** (Language, subsumption, downward-closure). *A* language *of pomsets is a set of morphisms* $L \subseteq \mathsf{iiPoms}$*. For* $P, Q \in \mathsf{iiPoms}(U, V)$*, we say that* P is subsumed by Q*, written* $P \sqsubseteq Q$*, when there is a bijection* $f : P \to Q$ *preserving the interfaces, labeling and event order, such that* $f(x) <_Q f(y)$ *implies* $x <_P y$*. Intuitively, the pomset* Q *is "more concurrent" than* P*. A language* $L \subseteq \mathsf{iiPoms}$ *is* downward-closed *when* $Q \in L$ *and* $P \sqsubseteq Q$ *implies* $P \in L$*. For a language* L*, we define* $L\!\downarrow = \{P \in \mathsf{iiPoms} \mid P \sqsubseteq Q \text{ for some } Q \in L\}$*, called the downward-closure of* L*.*

Let X be an HDA, and α a path in X. We define the pomset associated with α, $\mathsf{ev}(\alpha) \in \mathsf{iiPoms}(\mathsf{ev}(\mathsf{src}(\alpha)), \mathsf{ev}(\mathsf{tgt}(\alpha)))$, by induction on the length of α:

- If $\alpha = (x_0)$, then $\mathsf{ev}(\alpha) = \mathsf{id}_{\mathsf{ev}(x_0)}$.
- If $\alpha = (\delta^0_A(x), \nearrow^A, x)$, then $\mathsf{ev}(\alpha) = {}_A{\uparrow}\mathsf{ev}(x)$.
- If $\alpha = (x, \searrow_A, \delta^1_A(x))$, then $\mathsf{ev}(\alpha) = \mathsf{ev}(x){\downarrow_A}$.
- If $\alpha = \alpha_1 * \ldots * \alpha_n$ is a concatenation, then $\mathsf{ev}(\alpha) = \mathsf{ev}(\alpha_1) * \ldots * \mathsf{ev}(\alpha_n)$.

▶ **Definition 8** (HDA language). *The language accepted by an HDA X is*

$$\mathsf{Lang}(X) = \{\mathsf{ev}(\alpha) \mid \alpha \textit{ is an accepting path in } X\}$$

A language of pomsets is *regular* if it is accepted by a finite HDA.

▶ Remark 9. An important property of HDA languages is that regular languages are always downward-closed [9]. This is due to the geometric structure of HDA, where whenever a higher-dimensional cell exists, all of its faces must also exist. Thus, they model asynchronous computation, where one can never force events to occur simultaneously. Other formalisms, such as first-order logic or recognizability by finite categories, have no such restriction. Therefore, when comparing expressivity of HDA with other formalisms, we will explicitly restrict to the class of downward-closed languages.

Global assumption. Throughout the paper, we only consider languages of pomsets of bounded dimension, $\mathsf{iiPoms}_{\leq k}$. This is a crucial assumption, as it is not yet known how to properly model concurrent programs with an unbounded numbers of processes using HDA [5].

2.3 First-Order logic on pomsets

Another way to describe pomset languages is via logical characterizations. MSO logic over pomsets was introduced in [2] and shown to be as expressive as HDA, for downward-closed languages. The fragment of First Order (FO) logic has been studied in [6], together with an equivalent LTL-like logic. Here, we focus on the class of FO-definable languages of pomsets. FO formulas are generated by the following grammar, where $a \in \Sigma$ and x, y are variables:

$$\varphi, \psi ::= a(x) \mid \mathbf{s}(x) \mid \mathbf{t}(x) \mid \neg\varphi \mid \varphi \wedge \psi \mid \exists x.\varphi \mid x < y \mid x \dashrightarrow y$$

The semantics is straightforward: variables range over events of the pomset; $<$ and $\dashrightarrow$ stand for precedence and event-order, respectively; and a, $\mathbf{s}$, $\mathbf{t}$ stand for the labeling, starting interface and terminating interface, respectively.

▶ **Definition 10** (Semantics of FO over pomsets). *Consider an FO formula φ, a pomset $P = (P, <_P, \dashrightarrow_P, \lambda_P, S_P, T_P)$, and a valuation ν mapping variables to events of P. Then, the satisfaction relation $P, \nu \models \varphi$ is defined inductively as follows:*

$$\begin{array}{llll}
P,\nu \models a(x) & \textit{if } \lambda_P(\nu(x)) = a & P,\nu \models \exists x.\varphi & \textit{if } \exists p \in P.\ P, \nu[x \mapsto p] \models \varphi \\
P,\nu \models \mathbf{s}(x) & \textit{if } \nu(x) \in S_P & P,\nu \models \mathbf{t}(x) & \textit{if } \nu(x) \in T_P \\
P,\nu \models \neg\varphi & \textit{if } P,\nu \not\models \varphi & P,\nu \models \varphi \wedge \psi & \textit{if } P,\nu \models \varphi \textit{ and } P,\nu \models \psi \\
P,\nu \models x < y & \textit{if } \nu(x) <_P \nu(y) & P,\nu \models x \dashrightarrow y & \textit{if } \nu(x) \dashrightarrow_P \nu(y)
\end{array}$$

When φ has no free variable, we write $P \models \varphi$, and $\mathsf{Lang}(\varphi) = \{P \in \mathsf{iiPoms} \mid P \models \varphi\}$.

FO-definable languages may not be downward-closed. In fact, the class of FO-definable languages is not closed under downward-closure, as shown in Proposition 11. Thus, when comparing expressivity of HDA models with logical (or algebraic) characterizations, we will focus on the class of downward-closed FO-definable languages.

▶ **Proposition 11.** *The language* $\left[\begin{smallmatrix} a \\ a \end{smallmatrix}\right]^*$ *is FO-definable, but* $\left[\begin{smallmatrix} a \\ a \end{smallmatrix}\right]^*\downarrow$ *is not.*

Proof. The language $\left[\begin{smallmatrix} a \\ a \end{smallmatrix}\right]^*$ is recognized by the following FO formula: $\forall x.a(x) \wedge \exists! y.x \parallel y$, where $\exists!$ means "exists unique", defined as usual, and $x \parallel y := \neg(x < y \vee y < x \vee x = y)$. However, $\left[\begin{smallmatrix} a \\ a \end{smallmatrix}\right]^*\downarrow \cap a^* = (aa)^*$, which is not FO-definable [7]. Since FO-definable languages are closed under intersection, $\left[\begin{smallmatrix} a \\ a \end{smallmatrix}\right]^*\downarrow$ cannot be defined in FO. ◀

3 Recognizability for HDA languages

In this section, we propose a notion of algebraic recognizability for pomset languages. Over words, a language $L \subseteq \Sigma^*$ is recognized by a finite monoid M if there exists a monoid morphism $\varphi : \Sigma^* \to M$ and a subset $J \subseteq M$ such that $L = \varphi^{-1}(J)$. It is a well-known result that recognizable languages are exactly the regular languages [8]. In higher dimension, the set iiPoms of all pomsets over a given alphabet has the structure of a category rather than a monoid. The objects are interfaces, and the arrows are pomsets whose domain and codomain are the starting and terminating interfaces, respectively. Thus, we get the following notion of recognizability for pomset languages:

▶ **Definition 12.** *A language* $L \subseteq \mathsf{iiPoms}$ *is* recognizable *if there exists a finite category* $\mathcal{D}$*, a set of morphisms* $K \subseteq \mathsf{Mor}(\mathcal{D})$*, and a functor* $F : \mathsf{iiPoms} \to \mathcal{D}$ *such that* $L = F^{-1}(K)$*.*

The rest of the section is dedicated to proving the following Theorem 13. The first implication (Lemma 15) follows the standard construction of the syntactic monoid. The second implication (consequence of Proposition 22) is less straightforward and requires the notion of iiPoms-module.

▶ **Theorem 13.** *A pomset language* $L \subseteq \mathsf{iiPoms}_{\leq k}$ *is regular if and only if it is recognizable.*

3.1 From regularity to recognizability

To show that any regular language is recognizable, we use a construction similar to the usual syntactic monoid. Taking into account the interfaces, we describe in Definition 14 the *syntactic category* of a language. Then, we prove in Lemma 15 that if the language is regular, then its syntactic category is finite. Since any language is recognized by its syntactic category, this concludes the first direction of Theorem 13.

The definition of the syntactic category can be formulated for any category $\mathcal{C}$. We will then specialize to the case of $\mathcal{C} = \mathsf{iiPoms}$. Given a language $L \subseteq \mathsf{Mor}(\mathcal{C})$ and two morphisms $\alpha, \beta : U \to V$ in $\mathcal{C}$, we write $\alpha \sim \beta$ when $\forall \gamma, \delta.\ \gamma\alpha\delta \in L \Leftrightarrow \gamma\beta\delta \in L$. This is an equivalence relation on $\mathcal{C}(U, V)$, called the syntactic congruence.

▶ **Definition 14.** *The* syntactic category $\mathrm{Syn}(L)$ *of a language* $L \subseteq \mathsf{Mor}(\mathcal{C})$ *has objects* $\mathsf{Ob}(\mathrm{Syn}(L)) = \mathsf{Ob}(\mathcal{C})$*, and morphisms*

$$\mathrm{Syn}(L)(U, V) = \mathcal{C}(U, V)/\!\sim$$

with the composition $[\alpha] \fatsemi [\beta] = [\alpha \fatsemi \beta]$*.*

The syntactic category comes with an obvious projection $\pi_L : \mathcal{C} \to \mathrm{Syn}(L)$ and a language

$$K = \{[\alpha] \mid \alpha \in L\} \subseteq \mathrm{Syn}(L)$$

such that $\pi_L^{-1}(K) = L$.

▶ **Lemma 15.** *If $L \subseteq \mathsf{iiPoms}_{\leq k}$ is regular, then $Syn(L)$ is finite. As a consequence, L is recognizable.*

Proof. Let X be a finite HDA such that $\mathsf{Lang}(X) = L$. For $P \in \mathsf{iiPoms}(U, V)$, let

$$t(P) = \{(x, y) \in X[U] \times X[V] \mid \text{there is a path } \alpha \text{ s.t. } \mathsf{src}(\alpha) = x, \mathsf{tgt}(\alpha) = y \text{ and } \mathsf{ev}(\alpha) = P\}.$$

First, $t(P) = t(P')$ implies $P \sim P'$; second, there is only a finite number of possible values of t, so a finite number of equivalence classes for $\sim$. ◀

3.2 From recognizability to regularity

Now that we have established that any regular language is recognizable, we aim to prove the converse. For languages of words, translating a finite monoid M together with a homomorphism $\varphi : \Sigma^* \to M$ into a finite automaton is fairly straightforward: the states of the automaton are the elements of the monoid, and the transitions are given by $x \xrightarrow{a} y$ if $x \cdot \varphi(a) = y$. For pomset languages, however, the situation is different. Given a functor $F : \mathsf{iiPoms} \to \mathcal{D}$ into a finite category $\mathcal{D}$, the analogue of the above construction does not yield an HDA directly. Instead, what we get is what we call an iiPoms-module, that is, a set equipped with a right action of iiPoms. This can then be turned into an HDA, but it requires some work.

▶ **Definition 16.** *Let $\mathcal{C}$ be a small category. A $\mathcal{C}$-module is a set M equipped with*

- *Functions $\mathsf{src}, \mathsf{tgt} : M \to \mathsf{Ob}(\mathcal{C})$. For $U, V \in \mathsf{Ob}(\mathcal{C})$ let $M(U, V) = \mathsf{src}^{-1}(U) \cap \mathsf{tgt}^{-1}(V)$.*
- *Functions $\mu_{U,V,W} : M(U, V) \times \mathcal{C}(V, W) \to M(U, W)$. We write μ for the partial function $M \times \mathsf{Mor}(\mathcal{C}) \to M$, and $m \cdot \alpha$ for $\mu(m, \alpha)$.*

such that, for all $m \in M(U, V)$, $\alpha \in \mathcal{C}(V, W)$, and $\beta \in \mathcal{C}(W, T)$,

- *$m \cdot \mathsf{id}_V = m$, and*
- *$(m \cdot \alpha) \cdot \beta = m \cdot (\alpha \fatsemi \beta)$.*

▶ **Definition 17.** *A homomorphism of $\mathcal{C}$-modules is a function $\varphi : M \to M'$ such that for any $m \in M$ and $\alpha \in \mathsf{Mor}(\mathcal{C})$, $\mathsf{src}(\varphi(m)) = \mathsf{src}(m)$, $\mathsf{tgt}(\varphi(m)) = \mathsf{tgt}(m)$ and $\varphi(m \cdot \alpha) = \varphi(m) \cdot \alpha$.*

▶ Remark 18. In other words, a $\mathcal{C}$-module M is a family of functors $M(U, -) : \mathcal{C} \to \mathrm{Set}$ indexed by $U \in \mathsf{Ob}(\mathcal{C})$, and a homomorphism of $\mathcal{C}$-modules $M \to M'$ is a family of natural transformations $M(U, -) \Rightarrow M'(U, -)$.

Notice that, for $\mathcal{C} = \Sigma^*$ viewed as a category with one object, a Σ^*-module is a set M together with a right monoid action of Σ^*, that is, a deterministic finite state automaton. Here, we are interested in iiPoms-modules instead. For instance, the syntactic category $\mathrm{Syn}(L)$ is an iiPoms-module, with $[P] \cdot Q = [P * Q]$. More generally, any functor $F : \mathsf{iiPoms}_{\leq k} \to \mathcal{D}$ can be viewed as an $\mathsf{iiPoms}_{\leq k}$-module, by taking $M(U, V) = \{F(P) \mid P \in \mathsf{iiPoms}_{\leq k}(U, V)\}$ for all U, V, and then letting $m \cdot Q = m \fatsemi F(Q)$. Finally, $\mathsf{iiPoms}_{\leq k}$ itself is an $\mathsf{iiPoms}_{\leq k}$-module, with the right action $P \cdot Q = P * Q$.

▶ **Definition 19.** *A language $L \subseteq \mathsf{iiPoms}_{\leq k}$ is recognized by an $\mathsf{iiPoms}_{\leq k}$-module M if there exists a subset $J \subseteq M$ and a homomorphism of $\mathsf{iiPoms}_{\leq k}$-modules $\varphi : \mathsf{iiPoms}_{\leq k} \to M$ such that $L = \varphi^{-1}(J)$.*

Note that in the above definition, the morphism of $\mathsf{iiPoms}_{\leq k}$-modules $\varphi : \mathsf{iiPoms}_{\leq k} \to M$ is entirely determined by the image of the identity pomsets $\mathsf{id}_U \in \mathsf{iiPoms}_{\leq k}(U, U)$ for each interface U. These play the role of the "initial states", if we think of the $\mathsf{iiPoms}_{\leq k}$-module M as some sort of deterministic automaton. When a language L is recognized by an $\mathsf{iiPoms}_{\leq k}$-module, we call the tuple (M, J, φ) a *presentation* of L.

▶ **Example 20.** A presentation for the language $L = [\begin{smallmatrix} a \\ b \end{smallmatrix}]\downarrow + abc$ is given in Figure 3. Each element of the module M is represented by a node in the graph. The labels inside a node indicate the preimage of the element by φ. The elements of J are represented by adding a double border to their nodes. The arrows represent the action of some starter or terminator. For example, from the element $\varphi(b\bullet)$, we can either terminate b by acting on the right by $\bullet b$, to reach $\varphi(b)$; or start a by acting on the right by $[\begin{smallmatrix} a\bullet \\ \bullet b\bullet \end{smallmatrix}]$, to reach $\varphi([\begin{smallmatrix} a\bullet \\ b\bullet \end{smallmatrix}])$. The elements drawn in gray are those from which no element of J is accessible. Their preimage by φ is infinite. Some such elements have been omitted, e.g., $\varphi(c\bullet)$. One can notice that a finite presentation is essentially equivalent to a deterministic complete ST-automaton (see Section 4.1).

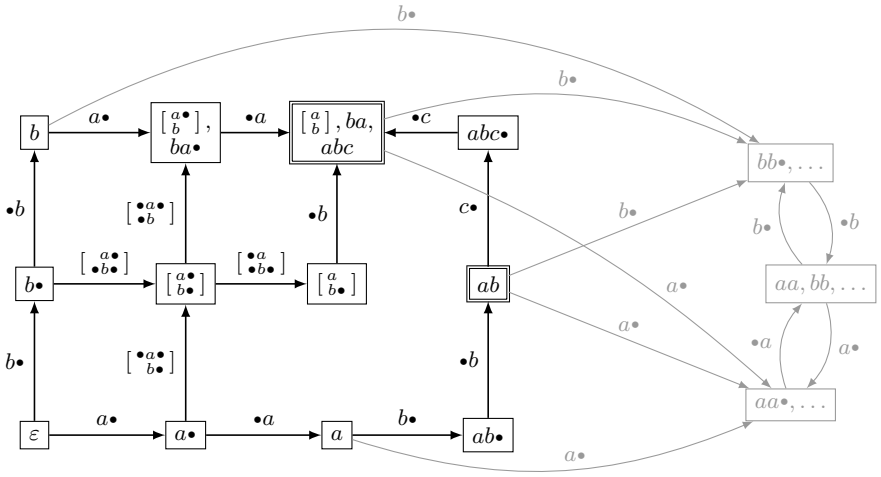


Figure 3 A presentation for $L = [\begin{smallmatrix} a \\ b \end{smallmatrix}]\downarrow + abc$.

Importantly, notice that one may always assume that the morphism $\varphi : \mathsf{iiPoms}_{\leq k} \to M$ is surjective. If that is not the case, we can simply discard the elements of M that are not in the image of φ, so that $(\varphi(\mathsf{iiPoms}_{\leq k}), J \cap \varphi(\mathsf{iiPoms}_{\leq k}), \varphi)$ is a surjective presentation of L.

▶ **Lemma 21.** *Any language $L \subseteq \mathsf{iiPoms}_{\leq k}$ recognizable by a finite category can be recognized by a finite $\mathsf{iiPoms}_{\leq k}$-module.*

Proof. Suppose L is recognized by a functor $F : \mathsf{iiPoms}_{\leq k} \to \mathcal{D}$ into a finite category, with $K \subseteq \mathsf{Mor}(\mathcal{D})$. As we have seen, this gives rise to an $\mathsf{iiPoms}_{\leq k}$-module whose elements are the morphisms of $\mathcal{D}$ in the image of F. Then F can be seen as a morphism of $\mathsf{iiPoms}_{\leq k}$-modules, and since $F^{-1}(K) = L$, this concludes the proof. ◀

Canonical suffix presentation

The reason for introducing the notion of $\mathsf{iiPoms}_{\leq k}$-module is that every finite presentation of a pomset language can be turned into a *canonical suffix presentation*. Given a language $L \subseteq \mathsf{iiPoms}_{\leq k}$ and pomset P, the *prefix quotient* $P\backslash L$ is defined by $P\backslash L = \{Q \in \mathsf{iiPoms}_{\leq k} \mid P * Q \in L\}$. Since pomset languages have a Myhill-Nerode theorem [11], a pomset language L is regular if and only if the set $\mathsf{suff}(L) = \{P\backslash L \mid P \in \mathsf{iiPoms}_{\leq k}\}$ is finite. While the set $\mathsf{suff}(L)$ does not have the structure of a category, it can be equipped with the structure of an $\mathsf{iiPoms}_{\leq k}$-module. This module can then be turned into an HDA using the construction from [11].

▶ **Proposition 22.** *Let $L \subseteq \mathsf{iiPoms}_{\leq k}$ be a language. Then, the presentation $(\mathsf{suff}(L), J, \varphi)$, called the canonical suffix presentation of L, defined by:*

- $\mathsf{src}(P\backslash L) = S_P$ *and* $\mathsf{tgt}(P\backslash L) = T_P$,
- $P\backslash L \cdot Q = (P * Q)\backslash L$,
- $J = \{P\backslash L \mid P \in L\}$,
- $\varphi(P) = P\backslash L$,

recognizes L. Additionally, for any presentation (M, K, ψ) recognizing L where ψ is surjective, there exists a module homomorphism $f : M \to \mathsf{suff}(L)$ such that $(f(M), f(K), f \circ \psi)$ is isomorphic to the canonical suffix presentation $(\mathsf{suff}(L), J, \varphi)$ of L.

Proof. First, $(\mathsf{suff}(L), J, \varphi)$ is well-defined since $P\backslash L \cdot \mathsf{id}_{T_P} = (P * \mathsf{id}_{T_P})\backslash L = P\backslash L$ and $(P\backslash L \cdot Q) \cdot R = (P * Q * R)\backslash L = P\backslash L \cdot (Q * R)$. Then, it is clear from the definitions that $L = \varphi^{-1}(J)$. Now, let us prove that any presentation (M, K, ψ) admits a homomorphism f such that $(f(M), f(K), f \circ \psi) = (\mathsf{suff}(L), J, \varphi)$.

Fix, for any $m \in M$, $f(m) = \psi^{-1}(m)\backslash L$. This is well-defined since if $\psi(P) = \psi(Q) = m$, then $P\backslash L = \{R \in \mathsf{iiPoms}_{\leq k} \mid m \cdot R \in L\} = Q\backslash L$. First, f is a module homomorphism. Observe that $\mathsf{src}(f(\psi(P))) = \mathsf{src}(\psi^{-1}(\psi(P))\backslash L) = \mathsf{src}(P\backslash L) = S_P = \mathsf{src}(\psi(P))$ as ψ is itself a homomorphism. The same goes for $\mathsf{tgt}(f(\psi(P)))$. Now, for any $Q \in \mathsf{iiPoms}_{\leq k}$, $f(\psi(P))Q = (\psi^{-1}(\psi(P))\backslash L)Q = (P\backslash L)Q = PQ\backslash L = \psi^{-1}(\psi(PQ))\backslash L = f(\psi(PQ)) = f(\psi(P)Q)$.

It is immediate that $f \circ \psi(P) = \psi^{-1}(\psi(P))\backslash L = P\backslash L$. Hence, $f(M) = \mathsf{suff}(L)$ and $f \circ \psi = \varphi$. Now, $f(K) = \{\psi^{-1}(\psi(P))\backslash L \mid \psi(P) \in K\} = \{P\backslash L \mid P \in L\} = J$. ◀

▶ **Example 23.** The presentation from Example 20 is isomorphic to the canonical prefix presentation of $L = \left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right]\!\downarrow + abc$. Notice, for instance, that $ab\backslash L = \{\varepsilon, c\}$ whereas $ba\backslash L = \{\varepsilon\}$. Therefore, any presentation of L must have two distinct elements $\varphi(ab) \neq \varphi(ba)$.

Proposition 22 suffices to conclude that recognizable languages of finite dimension are regular. Indeed, if we show that the set $\mathsf{suff}(L)$ is finite, then we can conclude by the Myhill-Nerode theorem for HDA [11] that L is regular. If L is recognizable, it is recognized by a finite presentation (M, K, ψ) where ψ is surjective. Then, Proposition 22 says that there is a module homomorphism $f : M \to \mathsf{suff}(L)$ such that $f(M) = \mathsf{suff}(L)$. Since M is finite, this implies that $\mathsf{suff}(L)$ is also finite.

The proof of the Myhill-Nerode theorem from [11] essentially shows how to construct an HDA from the canonical suffix presentation $(\mathsf{suff}(L), J, \varphi)$. In Section 3.3, we adapt this proof to directly turn any finite module recognizing L into an HDA accepting the same language.

3.3 Direct construction from modules to HDA

In this section, we give another, more direct proof that recognizable languages are regular. The construction of the HDA from an $\mathsf{iiPoms}_{\leq k}$-module is a generalized version of the one in [11]. In this section only, we restrict to languages of pomsets whose source interface is $\varnothing$, meaning that no processes are active at the start of computation. This restriction is made solely for ease of presentation, as it prevents some technicalities (see [11] for details). We write ${}_{\varnothing}\mathsf{iiPoms}$ for the set of pomsets whose source interface is $\varnothing$.

Given a finite presentation (M, J, φ), the U-cells of the resulting HDA are elements of $M(\varnothing, U)$. Face maps are then added in the same way as in [11]. However, for the down faces to be well-defined, we need an additional condition called coherence. In the following, for P a pomset and $A \subseteq P$ a subset of events of P, we write $P - A$ for the subpomset induced by the events of P not belonging to A.

▶ **Definition 24.** *A presentation* (M, J, φ) *is* coherent *if for all* U *and all* $P, Q \in \mathsf{iiPoms}(\varnothing, U)$*:*

$$\varphi(P) = \varphi(Q) \implies \text{for all } A \subseteq U.\ \varphi(P - A) = \varphi(Q - A),$$

▶ **Lemma 25.** *Any recognizable language has a coherent presentation.*

Proof. Fix a presentation (M, J, φ) recognizing a language L. Given $P, Q \in \mathsf{iiPoms}_{\leq k}$, we write $P \sim Q$ if $S_P = S_Q$, $T_P = T_Q$ and, for any $A \subseteq T_P$, $\varphi(P - A) = \varphi(Q - A)$. Then, $\sim$ is an equivalence relation with a finite number of classes, since M is finite and the dimension of L is bounded. Moreover, $\mathsf{iiPoms}_{\leq k}/\sim$ is an $\mathsf{iiPoms}_{\leq k}$-module. We now prove that together with $\psi : P \mapsto [P]$ and $K = \varphi^{-1}(J)$, it defines a coherent finite presentation for L.

First, ${}_{\varnothing}\mathsf{iiPoms}_{\leq k} \xrightarrow{\psi} \mathsf{iiPoms}_{\leq k}/\sim \supseteq K$ is coherent. If $\psi(P) = \psi(Q)$, then by definition $P \sim Q$, which implies that for any $A \subseteq T_P = T_Q$, $P - A \sim Q - A$. Therefore, $\psi(P - A) = \psi(Q - A)$. Finally, it recognizes L: $\psi(P) \in K$ if and only if there exists $Q \sim P$ such that $\varphi(Q) \in J$. Since $\varphi(Q) = \varphi(P)$, this is equivalent to $P \in L$. ◀

We can now construct an HDA from a coherent presentation. Observe that if (M, J, φ) is the canonical suffix presentation of L, the resulting HDA is the Myhill-Nerode HDA of [11].

▶ **Lemma 26.** *Let* (M, J, φ) *be a coherent presentation of a downward-closed language* L*, where* φ *is surjective. Then* X *defined by*

$$X[U] = M(\varnothing, U), \qquad \delta^0_A(\varphi(P)) = \varphi(P - A), \qquad \delta^1_B(m) = m \cdot U{\downarrow_B},$$
$$\bot_X = \{\varphi(\mathsf{id}_\varnothing)\}, \qquad \top_X = J,$$

is a well-defined HDA such that $\mathsf{Lang}(X) = L$*.*

Proof. First, note that since φ is surjective, $X[U] = M(\varnothing, U) = \varphi(\mathsf{iiPoms}(\varnothing, U))$. The map δ^0_A is well-defined: if $\varphi(P) = \varphi(Q)$, then $\varphi(P - A) = \varphi(Q - A)$ because the presentation is coherent. The precubical identities are straightforward to check. For example, if $m = \varphi(P) \in M(\varnothing, U)$, and $A, B \subseteq U$ are disjoint, $\delta^0_A(\delta^1_B(m)) = \varphi((P * U{\downarrow_B}) - A) = \varphi((P - A) * (U - A){\downarrow_B}) = \delta^1_B(\delta^0_A(m))$. So X is a well-defined HDA.

To prove that the language accepted by X is L, we first prove the following claim: for any path α in X, $\mathsf{tgt}(\alpha)\backslash L \subseteq \mathsf{ev}(\alpha)\backslash L$. Note that $\mathsf{tgt}(\alpha) \in M$, and $m\backslash L$ is defined by $m\backslash L = \{P \in \mathsf{iiPoms}_{\leq k} \mid m \cdot P \in J\}$. We prove this by induction over α.

- If $\alpha = (\varphi(\mathsf{id}_\varnothing))$, then $\mathsf{tgt}(\alpha)\backslash L = L = \mathsf{ev}(\alpha)\backslash L$.
- If $\alpha = (\beta, \searrow_B, m)$, then $m = \delta^1_B(\mathsf{tgt}(\beta)) = \mathsf{tgt}(\beta) \cdot U{\downarrow_B}$, where $\mathsf{tgt}(\beta) \in M(\varnothing, U)$. Let $R \in \mathsf{tgt}(\alpha)\backslash L$, that is, $m \cdot R \in J$. Since $m \cdot R = (\mathsf{tgt}(\beta) \cdot U{\downarrow_B}) \cdot R = \mathsf{tgt}(\beta) \cdot (U{\downarrow_B} * R)$, this yields $U{\downarrow_B} * R \in \mathsf{tgt}(\beta)\backslash L$. By induction hypothesis, we obtain $U{\downarrow_B} * R \in \mathsf{ev}(\beta)\backslash L$, which is in turn equivalent to $R \in (\mathsf{ev}(\beta) * U{\downarrow_B})\backslash L = \mathsf{ev}(\alpha)\backslash L$.
- If $\alpha = (\beta, \nearrow^A, m)$, let P be such that $\varphi(P) = m$, and $U = T_P$. Let $R \in \mathsf{tgt}(\alpha)\backslash L$, which means that $m \cdot R = \varphi(P * R) \in J$, that is, $P * R \in L$. One can check (see [11] for a proof) that $(P - A) * {}_A{\uparrow}U \sqsubseteq P$, and since L is downward-closed, $(P - A) * {}_A{\uparrow}U * R \in L$. Since $\mathsf{tgt}(\beta) = \delta^0_A(m) = \varphi(P - A)$, this means that ${}_A{\uparrow}U * R \in \mathsf{tgt}(\beta)\backslash L$. By induction hypothesis, we obtain ${}_A{\uparrow}U * R \in \mathsf{ev}(\beta)\backslash L$. Then, $R \in (\mathsf{ev}(\beta) * {}_A{\uparrow}U)\backslash L = \mathsf{ev}(\alpha)\backslash L$.

With this property, if α is an accepting path, $\mathsf{tgt}(\alpha) \in \top_X = J$, so $\mathsf{id}_U \in \mathsf{tgt}(\alpha)\backslash L$, which implies $\mathsf{id}_U \in \mathsf{ev}(\alpha)\backslash L$, i.e. $\mathsf{ev}(\alpha) \in L$. We have shown $\mathsf{Lang}(X) \subseteq L$.

For the converse, we show that for any pomset $P \in {}_{\varnothing}\mathsf{iiPoms}_{\leq k}$, there is a path α in X starting in $\bot_X$ and ending in $\varphi(P)$, with $\mathsf{ev}(\alpha) = P$. This implies that $L \subseteq \mathsf{Lang}(X)$, since for any $P \in L$, $\varphi(P) \in J$ is an accepting cell. We proceed by induction over the length of an ST-decomposition of $P = P_1 * \cdots * P_n$ (see Section 4.1 for a reminder on ST-decompositions).

If $n = 0$, then $P = \mathsf{id}_\varnothing$ and there is a path from $\bot_X$ to $\bot_X$ labeled by $\mathsf{id}_\varnothing$. Now, assume we have built a path β from $\bot_X$ to $\varphi(P_1 * \cdots * P_{n-1})$ with $\mathsf{ev}(\beta) = P_1 * \cdots * P_{n-1}$. There are two cases. If $P_n = {}_A{\uparrow}U$ is a starter, then we take the path $\alpha = (\beta, \nearrow^A, \varphi(P))$. This is a correct path because $\delta^0_A(\varphi(P)) = \varphi(P - A) = \mathsf{tgt}(\beta)$, because $P - A = (P_1 * \cdots * P_{n-1} * {}_A{\uparrow}U) - A = P_1 * \cdots * P_{n-1}$. Similarly, if $P_n = U{\downarrow}_A$ is a terminator, we take $\alpha = (\beta, \searrow_A, \varphi(P))$ which is a correct path because $\delta^1_A(\mathsf{tgt}(\beta)) = \mathsf{tgt}(\beta) \cdot U{\downarrow}_A = \varphi(P_1 * \cdots * P_{n-1} * U{\downarrow}_A) = \varphi(P)$. ◀

▶ **Example 27.** Consider again the presentation of the language $L = \left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right]{\downarrow} + abc$, depicted in Figure 3. The corresponding HDA, constructed by Lemma 26, is depicted in Figure 4. Notice that the vertical b-transition on the right side of the square does not exist in the original presentation. It corresponds to the element $\varphi(\left[\begin{smallmatrix} a \\ b_\bullet \end{smallmatrix}\right])$ of the module. The definition of HDA requires that this 1-dimensional cell must have a lower face map, which we defined as $\delta^0_{\{b\}}(\varphi(\left[\begin{smallmatrix} a \\ b_\bullet \end{smallmatrix}\right])) = \varphi(a)$. As a consequence, the resulting HDA is not deterministic, even though presentations, by construction, are. This is not surprising, since it was established in [11] that there is no deterministic HDA for this language L.

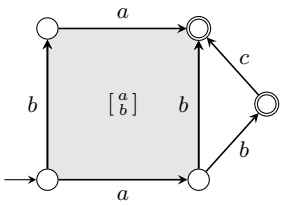


Figure 4 The HDA constructed from the presentation of Example 20 (co-accessible cells only).

4 Aperiodic categories and First-Order logic

For languages of finite words, algebraic characterizations have led to the study of several strict subclasses of regular languages, defined by the properties of their recognizing monoids. Perhaps the most prominent is the class of languages recognized by aperiodic monoids. This class can equivalently be characterized using star-free regular expressions, counter-free automata and first-order logic [16, 14]. Intuitively, these languages can be recognized without needing to count letters modulo a constant $k > 1$. For instance, a^* is star-free, whereas $(aa)^*$ is not, because in the latter requires counting the number of a's modulo 2.

In this section, we extend this class of languages from words to pomsets. Recall that a finite monoid M is *aperiodic* if there exists $n \in \mathbb{N}$ such that, for every $x \in M$, $x^{n+1} = x^n$. Extending this definition to categories and modules is straightforward: we require the same property for every endomorphism $x \in \mathcal{C}(U, U)$. For the particular case of pomsets, this means that we only iterate pomsets with the same source and target interfaces.

▶ **Definition 28.** *A finite category $\mathcal{C}$ is* aperiodic *if there exists $n \in \mathbb{N}$ such that, for every object U and endomorphism $x \in \mathcal{C}(U, U)$, $x^{n+1} = x^n$.*

▶ **Definition 29.** *A $\mathcal{C}$-module M is* counter-free *if there exists $n \in \mathbb{N}$ such that, for every objects U, V and for every $m \in M(V, U)$ and $c \in \mathcal{C}(U, U)$, $m \cdot x^{n+1} = m \cdot x^n$.*

For modules, we prefer to use the term *counter-free*, traditionally used for automata, because we consider them to be closer to automata than to categories. In this section, we aim to prove that aperiodic categories, counter-free modules, and FO logic recognize the same pomset languages. Expressivity of counter-free HDA will be explored in Section 5.

4.1 ST-sequences and ST-automata

First, let us introduce a technical tool called the Starter-Terminator (ST) decomposition of a pomset. In previous work, ST-decompositions have been used to lift various results from words to pomsets, such as the Büchi-Elgot-Trakhtenbrot theorem [2] and Kamp theorem [6]. The key idea is that any pomset of bounded dimension can be represented as a finite word over a finite alphabet, called an ST-sequence. Recall from Section 2.2 that starters and terminators are special elementary pomsets consisting of concurrent events, some of which may be started or terminated. We denote by Ω the set of all starters and terminators (including identities), and $\Omega_{\leq k}$ those of dimension at most k. Note that $\Omega_{\leq k}$ is finite.

▶ **Definition 30.** *A word $w = P_1 P_2 \cdots P_n \in \Omega^*$ is an* ST-sequence *if $T_{P_i} = S_{P_{i+1}}$ for any $i = 1, 2, \ldots, n-1$. In such a case, we define $\mathsf{glue}(w) = P_1 * P_2 * \cdots * P_n \in \mathsf{iiPoms}$.*

Although any ST-sequence defines a unique pomset, a given pomset can be decomposed into several ST-sequences. However, there exists a canonical ST-decomposition of a pomset.

▶ **Proposition 31** ([11])**.** *For any pomset P, there exists a unique ST-sequence w such that $\mathsf{glue}(w) = P$ and w alternates between non-identity starters and terminators.*

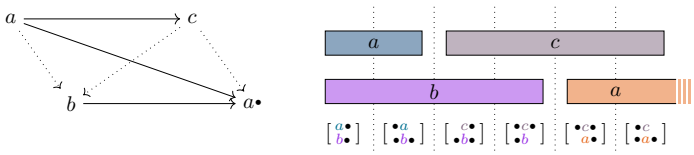


Figure 5 A pomset with its associated sparse ST-decomposition.

Such a decomposition is called *sparse*. As emphasized, non sparse ST-decompositions are not unique. For instance, the pomset depicted in Figure 5 has non-sparse ST-decomposition $[{}^{a}_{\bullet}][{}^{\bullet a}_{b\bullet}][{}^{\bullet a}_{\bullet b\bullet}][{}^{c\bullet}_{\bullet b\bullet}][{}^{\bullet c\bullet}_{\bullet b}][{}^{\bullet c\bullet}_{a\bullet}][{}^{\bullet c}_{a\bullet}]$, with two consecutive starters. Given a language K of ST-sequences, we denote by $\mathsf{glue}(K) = \{\mathsf{glue}(w) \mid w \in K\}$ the language of associated pomsets. Conversely, given a language L of pomsets of dimension k, $\mathsf{ST}(L) = \{w \in \Omega^*_{\leq k} \mid \mathsf{glue}(w) \in L\}$ is the language of ST-decompositions representing pomsets of L. Since ST-sequences are words over a finite alphabet, they can be accepted by finite state automata.

▶ **Definition 32** ([1])**.** *A* k-dimensional ST-automaton *is a tuple $\mathcal{A} = (Q, I, F, \delta, \lambda)$ where (Q, I, F, δ) is a finite state automaton over the alphabet $\Omega_{\leq k}$ and $\lambda : Q \to \mathsf{CList}_{\leq k}$ is a labeling function such that for any transition $q \xrightarrow{P} q'$, $\lambda(q) = S_P$ and $\lambda(q') = T_P$.*

4.2 A McNaughton-Papert theorem for pomset languages

▶ **Theorem 33.** *Let $L \subseteq \mathsf{iiPoms}_{\leq k}$ be a pomset language of bounded dimension k. The following propositions are equivalent:*

1. *L is recognized by an aperiodic category,*
2. *L is recognized by a counter-free iiPoms-module,*
3. *L is defined by an FO formula.*

Proof. (1) ⇒ (2). Let $\mathcal{C}$ be an aperiodic category recognizing L with functor F. We prove that $\mathcal{C}$, considered as an $\mathsf{iiPoms}_{\leq k}$-module as in the proof of Lemma 21, is counter-free. Fix $P, Q \in \mathsf{iiPoms}$. Then, $F(P) \cdot Q^{n+1} = F(P) \fatsemi F(Q)^{n+1} = F(P) \fatsemi F(Q)^n = F(P) \cdot Q^n$, where $\fatsemi$ is the composition in $\mathcal{C}$. So the associated module is counter-free.

(2) ⇒ (3). Let (M, J, φ) be a counter-free presentation for L. Then, consider the ST-automaton $\mathcal{A} = (Q, I, F, \delta, \lambda)$ with:

$$Q = M, \qquad I = \{\varphi(\mathsf{id}_U) \mid U \in \mathsf{CList}_{\leq k}\}, \qquad F = J,$$

$$\delta(m, P) = m \cdot \varphi(P), \qquad \lambda(m) = \mathsf{tgt}(m).$$

First, observe that $\mathcal{A}$ is counter-free (see Section 5 for a reminder of the notion of counter-freeness for word automata). Indeed, for any $m \in Q$ and $w \in \Omega^*_{\leq k}$, taking i such that $m \cdot P^{i+1} = m \cdot P^i$, we have $\delta(m, w^{i+1}) = m \cdot \varphi(\mathsf{glue}(w))^{i+1} = m \cdot \varphi(\mathsf{glue}(w))^i = \delta(m, w^i)$. Further, $\mathsf{Lang}(\mathcal{A}) = \mathsf{ST}(L)$, which implies that the word language $\mathsf{ST}(L)$ is FO-definable [14]. Hence, using a result from [6], $\mathsf{glue}(\mathsf{ST}(L)) = L$ can be recognized by an FO formula over pomsets.

(3) ⇒ (1). It is known from [6] that any pomset language L is defined by an FO formula if and only if $\mathsf{ST}(L)$ is defined by an FO formula over ST-sequences. Since ST-sequences are words over finite alphabets, this implies that there exists an aperiodic monoid recognizing $\mathsf{ST}(L)$ [14]. Fix M to be such a monoid, with $J \subseteq M$ and $\varphi : \Omega^*_{\leq k} \to M$ a monoid morphism such that $\mathsf{ST}(L) = \varphi^{-1}(J)$. Assume that φ is surjective (if this is not the case, then we can replace M by $\varphi(\Omega^*_{\leq k})$). We cannot recognize L using M and φ because one pomset can be recognized by several ST-sequences. Let us define $\sim$ the equivalence relation over M induced by $\varphi(v) \sim \varphi(w)$ whenever $\mathsf{glue}(v) = \mathsf{glue}(w)$. The relation $\sim$ is a congruence on M. Indeed, if $a, m, m' \in M$ and $m \sim m'$, then there exist $v, w, w' \in \Omega^*_{\leq k}$ such that $\varphi(v) = a$, $\varphi(w) = m$, $\varphi(w') = m'$ and $\mathsf{glue}(w) = \mathsf{glue}(w')$. Then

$$\mathsf{glue}(vw) = \mathsf{glue}(v) * \mathsf{glue}(w) = \mathsf{glue}(v) * \mathsf{glue}(w') = \mathsf{glue}(vw'),$$

which implies $am = \varphi(v)\varphi(w) = \varphi(vw) \sim \varphi(vw') = \varphi(v)\varphi(w') = am'$. Similarly, $ma \sim m'a$. Thus, the quotient $M/\sim$ is a monoid. Now, define $\psi : \mathsf{iiPoms}_{\leq k} \to M/\sim$ by the formula $\psi(P) = [\varphi(\mathsf{glue}^{-1}(P))]$. This is valid since for $w, w' \in \Omega^*_{\leq k}$ such that $\mathsf{glue}(w) = \mathsf{glue}(w') = P$ we have $\varphi(w) \sim \varphi(w')$ and hence $[\varphi(w)] = [\varphi(w')]$. Since

$$\psi(\mathsf{glue}(v) * \mathsf{glue}(w)) = \psi(\mathsf{glue}(vw)) = [\varphi(vw)] = [\varphi(v)][\varphi(w)] = \psi(\mathsf{glue}(v))\psi(\mathsf{glue}(w)),$$

ψ is a homomorphism. Further, we choose $J' = \{[m] \mid m \in J\}$. Notice that if $m \sim m'$, then $m \in J$ iff $m' \in J$. To see this, assume $m = \varphi(w)$ and $m' = \varphi(w')$ for some $w, w' \in \Omega^*_{\leq k}$ such that $\mathsf{glue}(w) = \mathsf{glue}(w')$. Then, $w \in \mathsf{ST}(L)$ iff $w' \in \mathsf{ST}(L)$, that is, $\varphi(w) \in J$ iff $\varphi(w') \in J$. From this, we deduce that $L = \psi^{-1}(J')$. Indeed, $\psi(P) \in J'$ is equivalent to $\varphi(w) \sim m$ for some $m \in J$ and $w \in \Omega_{\leq k}$ with $\mathsf{glue}(w) = P$. Then $\varphi(w) \in J$, i.e., $w \in \mathsf{ST}(L)$, i.e., $P \in L$. Finally, M' is aperiodic since $[m]^{n+1} = [m^{n+1}] = [m^n] = [m]^n$. This concludes the proof. ◀

5 Aperiodic categories and counter-free HDA

In the context of words, a deterministic automaton $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ is *counter-free* if there exists $n \in \mathbb{N}$ such that, for any word $w \in \Sigma^*$ and state $q \in Q$, $\delta(q, w^{n+1}) = \delta(q, w^n)$ [14]. Essentially, this says that the transition monoid of the automaton is aperiodic, so the proof that counter-free automata recognize aperiodic languages is fairly straightforward. It is, however, limited to deterministic automata, and it has been established that not all HDA can be determinized [11, Section 6]. In [7], Diekert and Gastin discussed two possible definitions of counter-freeness in the non-deterministic setting of Büchi automata. We adapt the less restrictive of them, Definition 11.5, for HDA.

5.1 Counter-free HDA recognize aperiodic languages

Given an HDA X, a cell $x \in X[U]$ and a pomset $P \in \mathsf{iiPoms}(U, V)$, we denote by $d(x, P)$ the set of cells accessible from x by paths labeled by P.

▶ **Definition 34.** *An HDA X is* counter-free *if there exists $n \in \mathbb{N}$ such that, for any cell $x \in X[U]$ and pomset $P \in \mathsf{iiPoms}(U, U)$, $d(x, P^{n+1}) = d(x, P^n)$.*

▶ **Proposition 35.** *For any counter-free HDA accepting a language L, there exists an aperiodic category recognizing L.*

Proof. If L is recognized by a counter-free HDA, then the associated ST-automaton is counter-free. Hence, the word language $\mathsf{ST}(L)$ is defined by an FO formula [14]. Thus, so is the pomset language L [6]. The conclusion comes from Theorem 33. ◀

▶ **Conjecture 36.** *If $L \subseteq \mathsf{iiPoms}_{\leq k}$ is downward-closed w.r.t. subsumption, and is recognized be an aperiodic category, then L is accepted by a counter-free HDA.*

This statement, which is the converse of Proposition 35, is still open. Though Definition 34 is very similar to other definitions of aperiodicity and counter-freeness, Conjecture 36 is not immediate because of the geometric structure of HDAs imposing that any cell must be equipped with all possible faces. The construction of Lemma 26 that turns an iiPoms-module into an HDA may need to add new cells to guarantee that face maps are properly defined. While this construction does not change the accepted language (assuming it was already downward-closed), it can add non-trivial counters to the structure.

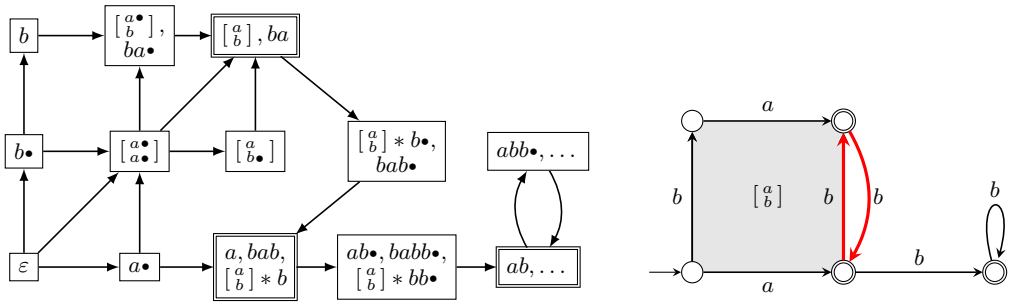
5.2 Creating counters

In this subsection, we show that the HDA construction of Lemma 26 does not preserve aperiodicity. Indeed, to turn an iiPoms-module into an HDA recognizing the same language, the construction of Lemma 26 (adapted from [11]) introduces supplementary lower faces. Such extra cells, required by the HDA structure, may create counters that were not present in the original module.

▶ **Proposition 37.** *There exists a counter-free presentation (M, J, φ) such that the resulting HDA from Lemma 26 is not counter-free.*

Proof. Consider the presentation (M, J, φ) depicted in Figure 6a. In this figure, elements of M are nodes in the graph, and the labels inside the nodes indicate the preimage by φ: when $\varphi(P) = m$, the pomset P is written within the node representing m. Elements of J are the nodes with a double edge. Arrows represent the action of some starter or terminator. This presentation recognizes the downward-closed language $L = \left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right]\!\downarrow b^* + a$. It is aperiodic because the only cycles are $\varphi(ab) \cdot b = \varphi(ab)$ and $\varphi(abb\bullet) \cdot \bullet bb\bullet = \varphi(abb\bullet)$ and both are trivial. However, the HDA constructed from Lemma 26, depicted in Figure 6b, is not counter-free. ◀

This counterexample shows that the most direct approach to proving Conjecture 36 does not work; however, it does not disprove the conjecture itself. The language $L = \left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right]\!\downarrow b^* + a$ can in fact be recognized by a counter-free HDA. More precisely, the Myhill-Nerode HDA for this language, constructed from the canonical prefix presentation, is counter-free. It is the HDA obtained from the one of Figure 6b by merging the three accepting cells.



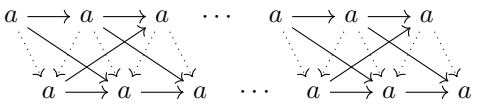
(a) A counter-free presentation for $L = [\begin{smallmatrix} a \\ b \end{smallmatrix}]{\downarrow}\, b^* + a$. **(b)** The resulting HDA, counter red.

Figure 6 A counter-free presentation whose associated HDA has a counter.

5.3 Almost counting aperiodic language

We now showcase an example of an aperiodic language that allows for a certain specific counting. We assume the one-letter alphabet $\Sigma = \{a\}$, and pomsets of dimension bounded by $k = 2$. Let us introduce pomsets $P_0 = \varepsilon$, and for $n \geq 1$,

$$P_{2n} \;=\; a\bullet \left[\begin{smallmatrix} \bullet a a \bullet \\ a \end{smallmatrix}\right]^{n-1} \left[\begin{smallmatrix} \bullet a \\ a \end{smallmatrix}\right] \;=$$



Intuitively, P_{2n} is the 2-dimensional pomset containing $2n$ events labeled by a such that the i-th event is concurrent with the $(i-1)$-th and the $(i+1)$-th events. Furthermore, they are alternating: odd events are on the top (i.e., they are minimal according to event order) and even events are at the bottom (i.e., they are maximal).

▶ **Lemma 38.** *The (non-downward-closed) language* $\{P_{2n} \mid n \geq 0\}$ *is aperiodic.*

Proof. It is easy to verify that this language is described as a set of pomsets P satisfying the following first order formula. For every x, $a(x)$ and there exist y, y' such that one of the following four conditions hold:

- $y \dashrightarrow x \dashleftarrow y'$ and for all $z \neq x, y, y'$, either $z < x$ or $x < z$,
- $y \dashleftarrow x \dashrightarrow y'$ and for all $z \neq x, y, y'$, either $z < x$ or $x < z$,
- $x \dashrightarrow y$ and for all $z \neq x, y,\ x < z$,
- $y \dashrightarrow x$ and for all $z \neq x, y,\ z < x$. ◀

The key idea is that we do not actually need to count the parity of the number of a's. Instead, we keep track of whether the event is up or down according to the event order. If we start up, alternate positions, and finish down, the number of a's will be even. Note that the language $\{P_{2n}\}{\downarrow}$ is not aperiodic, since $\{P_{2n}\}{\downarrow} \cap a^* = (aa)^*$ is not aperiodic.

▶ **Lemma 39.** *The language* $L = \mathsf{iiPoms}_{\leq 2}(\varnothing, \varnothing) \setminus \{P_{2n}\}$ *is downward-closed and aperiodic.*

Proof. The difference of aperiodic languages is aperiodic. Furthermore, L is downward-closed since P_{2n} are maximal pomsets with respect to subsumption in $\mathsf{iiPoms}_{\leq 2}$. ◀

Thus, there exists an aperiodic downward-closed language L such that (1) it contains all pomsets of odd length, and (2) for every $n \geq 0$, there is a pomset of length $2n$ that does not belong to L. We conclude from this example that aperiodic pomset languages admit a limited form of "counting", in contrast to word languages. The language L from Lemma 39, however, can be recognized by a counter-free HDA, so it still does not invalidate the conjecture.

6 Conclusion and future work

We have proved an algebraic characterization of regular languages of HDA: a pomset language is regular if and only if it is the preimage of a functor into a finite category. Furthermore, the finite category recognizing a language is aperiodic if and only if the language is definable in first order logic. This generalizes the McNaughton-Papert theorem to a higher-dimensional setting. Aperiodic pomset languages are peculiar in that they allow for some restricted form of counting, unlike word languages.

We have defined a notion of counter-free HDA, and shown that every language accepted by a counter-free HDA is aperiodic. However, the converse is still an open question: it is not yet well understood how aperiodicity interacts with downward-closure, which is intrinsic to HDAs. One possible way to solve it would be to prove that the HDA constructed from the canonical suffix presentation using Lemma 26 is always counter-free. There does not seem to be a specific reason why this should be the case, but we have not yet found a counterexample. Another, perhaps more promising, approach could be to show that, if an aperiodic language is recognized by a non-counter-free HDA, it is always possible to add some extra cells so as to eliminate the counter without changing the language.

One concrete use-case of the McNaughton-Papert theorem, for word automata, is to decide whether a given regular language is FO-definable: first compute its syntactic monoid, then check whether it is aperiodic. Doing the same for HDA is an interesting research direction: how to effectively compute the syntactic category, is it guaranteed to be aperiodic? Finally, another avenue for future work would be to find a notion of star-free regular expressions that captures the same class of pomset languages. HDA languages do not behave well w.r.t. complementation [3], so finding a good notion of star-free regular expression is not straightforward.

References

1 Amazigh Amrane, Hugo Bazille, Emily Clement, Uli Fahrenberg, and Krzysztof Ziemianski. Presenting interval pomsets with interfaces. In Uli Fahrenberg, Wesley Fussner, and Roland Glück, editors, *Relational and Algebraic Methods in Computer Science - 21st International Conference, RAMiCS 2024, Prague, Czech Republic, August 19-22, 2024, Proceedings*, volume 14787 of *Lecture Notes in Computer Science*, pages 28–45. Springer, 2024. doi:10.1007/978-3-031-68279-7_3.

2 Amazigh Amrane, Hugo Bazille, Uli Fahrenberg, and Marie Fortin. Logic and languages of higher-dimensional automata. In Joel D. Day and Florin Manea, editors, *Developments in Language Theory - 28th International Conference, DLT 2024, Göttingen, Germany, August 12-16, 2024, Proceedings*, volume 14791 of *Lecture Notes in Computer Science*, pages 51–67. Springer, 2024. doi:10.1007/978-3-031-66159-4_5.

3 Amazigh Amrane, Hugo Bazille, Uli Fahrenberg, and Krzysztof Ziemianski. Closure and decision properties for higher-dimensional automata. *Theor. Comput. Sci.*, 1036:115156, 2025. doi:10.1016/J.TCS.2025.115156.

4 Steve Awodey. *Category Theory*. Oxford University Press, Oxford, GB, 2010.

5 Thomas Baronner, Henning Basold, and Márton Hablicsek. Irrationality of process replication for higher-dimensional automata. In Uli Fahrenberg, Wesley Fussner, and Roland Glück, editors, *Relational and Algebraic Methods in Computer Science - 21st International Conference, RAMiCS 2024, Prague, Czech Republic, August 19-22, 2024, Proceedings*, volume 14787 of *Lecture Notes in Computer Science*, pages 46–64. Springer, 2024. doi:10.1007/978-3-031-68279-7_4.

6 Emily Clement, Enzo Erlich, and Jérémy Ledent. Kamp theorem for pomset languages of higher dimensional automata. In Stefano Guerrini and Barbara König, editors, *34th EACSL Annual Conference on Computer Science Logic, CSL 2026, Paris, France, February 23-28, 2026*, LIPIcs, pages 43:1–43:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2026. doi:10.4230/LIPIcs.CSL.2026.43.

7 Volker Diekert and Paul Gastin. First-order definable languages. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*, volume 2 of *Texts in Logic and Games*, pages 261–306. Amsterdam University Press, 2008. URL: https://lsv.ens-paris-saclay.fr/~gastin/Verif/DiekertGastin-FO-07.pdf.

8 Samuel Eilenberg. *Automata, languages, and machines. A*. Pure and applied mathematics. Academic Press, 1974. URL: https://www.sciencedirect.com/bookseries/pure-and-applied-mathematics/vol/59/part/PA.

9 Uli Fahrenberg, Christian Johansen, Georg Struth, and Krzysztof Ziemianski. Languages of higher-dimensional automata. *Math. Struct. Comput. Sci.*, 31(5):575–613, 2021. doi:10.1017/S0960129521000293.

10 Uli Fahrenberg, Christian Johansen, Georg Struth, and Krzysztof Ziemianski. Kleene theorem for higher-dimensional automata. *Log. Methods Comput. Sci.*, 20(4), 2024. doi:10.46298/LMCS-20(4:22)2024.

11 Uli Fahrenberg and Krzysztof Ziemianski. Myhill-Nerode theorem for higher-dimensional automata. *Fundam. Informaticae*, 192(3-4):219–259, 2024. doi:10.3233/FI-242194.

12 Johan Anthony Willem Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, Computer Science Department, University of California at Los Angeles, USA, 1968.

13 Leslie Lamport. On interprocess communication. part I: basic formalism. *Distributed Comput.*, 1(2):77–85, 1986. doi:10.1007/BF01786227.

14 Robert McNaughton and Seymour Papert. *Counter-Free Automata*, volume Research Monograph, 65. The M.I.T. Press, Cambridge, Mass., 1971.

15 Vaughan R. Pratt. Modeling concurrency with geometry. In David S. Wise, editor, *Conference Record of the Eighteenth Annual ACM Symposium on Principles of Programming Languages, Orlando, Florida, USA, January 21-23, 1991*, pages 311–322. ACM Press, 1991. doi:10.1145/99583.99625.

16 Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965. doi:10.1016/S0019-9958(65)90108-7.