Automata in the Category of Glued Vector Spaces*

Thomas Colcombet[1] **and Daniela Petrişan**[1]

1 **CNRS, IRIF, Univ. Paris-Diderot, Paris 7, France**
{thomas.colcombet,petrisan}@irif.fr

Abstract

In this paper we adopt a category-theoretic approach to the conception of automata classes enjoying minimization by design. The main instantiation of our construction is a new class of automata that are hybrid between deterministic automata and automata weighted over a field.

1998 ACM Subject Classification F.1.1

Keywords and phrases hybrid set-vector automata, automata minimization in a category

Digital Object Identifier 10.4230/LIPIcs.MFCS.2017.52

1 Introduction

In this paper we introduce a new automata model, *hybrid set-vector automata*, designed to accept weighted languages over a field in a more efficient way than Schützenberger's weighted automata [13]. The space of states for these automata is not a vector space, but rather a union of vector spaces "glued" together along subspaces. We call them hybrid automata, since they naturally embed both deterministic finite state automata and finite automata weighted over a field. In Section 2 we present at an informal level a motivating example and the intuitions behind this construction, avoiding as much as possible category-theoretical technicalities. We use this example to guide us throughout the rest of the paper.

A key property that the new automata model should satisfy is minimization. Since the morphisms of "glued" vector spaces are rather complicated to describe, proving the existence of minimal automata "by hand" is rather complicated. Therefore we opted for a more systematic approach and adopted a category-theoretic perspective for designing *new forms of automata* that enjoy *minimization by design*. In particular, we introduce the category of "glued" vector spaces in which these automata should live and we analyse its properties that render minimization possible.

Starting with the seminal papers of Arbib and Manes, see for example [3] and the references therein, and of Goguen [10], it became well established that category theory offers a neat understanding of several phenomena in automata theory. In particular, the key property of minimization in different contexts, such as for deterministic automata (over finite words) and Schützenberger's automata weighted over fields [13], arises from the same categorical reasons (existence of some limits/colimits and an (epi,mono)-factorization system [3]).

There is a long tradition of seeing automata either as algebras or coalgebras for a functor. However, in the case of deterministic automata, the algebraic view does not capture the

* This work was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No.670624), and by the DeLTA ANR project (ANR-16-CE40-0007). The authors also thank the Simons Institute for the Theory of Computing where this work has been partly developped.

accepting states, while the coalgebraic view does not capture the initial state. In the coalgebraic setting one needs to consider the so-called pointed coalgebras, see for example [1], where minimal automata are modelled as well-pointed coalgebras. The dual perspective of automata seen as both algebras or coalgebras, as well as the duality between reachability and observability, has been explored more recently in papers such as [4–6].

Here we take yet another approach to defining automata in a category. The reader acquainted with category theory will recognise that we see automata as functors from an input category (that specifies the type of the machines under consideration, which in this paper is restricted to word automata) to a category of output values. We show that the next ingredients are sufficient to ensure minimization: the existence of an initial and of a final automaton for a language, and a factorization system on the category in which we interpret our automata.

For example, deterministic and weighted automata over a field are obtained by considering as output categories the categories Set of sets and functions and Vec of vector spaces and linear maps, respectively. Since Set and Vec have all limits and colimits, it is very easy to prove the existence of initial and final automata accepting a given language. In both cases, the minimal automaton for a language is obtained by taking an epi-mono factorization of the unique arrow from the initial to the final automaton.

Notice that the initial and the final automata have infinite (-dimensional) state sets (spaces). If the language at issue is regular, that is, if the unique map from the initial to the final automaton factors through a finite (-dimensional) automaton then, automatically, the minimal automaton will also be finite (-dimensional). However, this relies on very specific properties of the categories of sets and vector spaces, namely on the fact that the full subcategories $\mathsf{Set}_{\mathsf{fin}}$ of finite sets and $\mathsf{Vec}_{\mathsf{fin}}$ of finite dimensional vector spaces are closed in Set, respectively in Vec, under both quotients and subobjects.

Coming back to hybrid set-vector automata, we define them as word automata interpreted in an output category $\mathtt{Glue}(\mathsf{Vec})$ which we obtain as the completion of Vec under certain colimits, and can be described at an informal level as "glueings" of arbitrary vector spaces. The definition of this form of cocompletion $\mathtt{Glue}(\mathcal{C})$ of a category $\mathcal{C}$ is the subject of Section 4.

We are interested in those hybrid automata for which the state object admits a finitary description, which intuitively can be described as *finite* glueings of *finite dimensional* spaces. For this reason we will consider the subcategory $\mathtt{Glue}_{\mathsf{fin}}(\mathsf{Vec}_{\mathsf{fin}})$ of $\mathtt{Glue}(\mathsf{Vec})$. It turns out that $\mathtt{Glue}_{\mathsf{fin}}(\mathsf{Vec}_{\mathsf{fin}})$ is closed under quotients in $\mathtt{Glue}(\mathsf{Vec})$ but, crucially, *it is not closed under subobjects*. For example, a glueing of infinitely many one-dimensional spaces is a subobject of a two-dimensional space, but only the latter is an object of $\mathtt{Glue}_{\mathsf{fin}}(\mathsf{Vec}_{\mathsf{fin}})$.

This is the motivation for introducing a notion of *$(E_{\mathcal{S}},M_{\mathcal{S}})$-factorization of a category $\mathcal{C}$ through a subcategory $\mathcal{S}$*. This is a refinement of the classical notion of factorization system on $\mathcal{C}$ and is used for isolating the semantical computations (in $\mathcal{C}$) from the automata themselves (with an object from $\mathcal{S}$ as "set of configurations").[1] We show how it provides a minimization of "$\mathcal{S}$-automata for representing $\mathcal{C}$-languages". A concrete instance of this is a factorization system on $\mathtt{Glue}(\mathsf{Vec})$ through $\mathtt{Glue}_{\mathsf{fin}}(\mathsf{Vec}_{\mathsf{fin}})$, which plays a crucial role in proving the existence of minimal $\mathtt{Glue}_{\mathsf{fin}}(\mathsf{Vec}_{\mathsf{fin}})$-automata for recognizing weighted languages.

The rest of the paper is organised as follows. We first develop a motivating example of a hybrid set-vector space automaton in Section 2. We then identify in Section 3 the category-theoretic ingredients that are sufficient for a class of automata to enjoy minimization. We

[1] This distinction is usually not necessary, and we are not aware of its existence in the literature. It is crucial for us, thus we cannot use already existing results from the coalgebraic literature, e.g. [1].

then turn to our main contribution, namely the description and the study of the properties of (finite-)mono-diagrams in a category, in Section 4. We conclude in Section 5 with a discussion of some of the design choices we made in this paper.

2 The hybridisation of deterministic finite state and vector automata

In this section, we (rather informally) describe the motivating example of this paper: the construction of a family of automata that naturally extends both deterministic finite state automata and finite automata weighted over a field in the sense of Schützenberger (i.e., automata in the category of finite vector spaces). The intuition should then support the categorical constructions that we develop in the subsequent sections.

Set automata (deterministic automata). Let us fix ourselves an alphabet $\mathtt{A}$. A *deterministic automaton* (or *set automaton*) is a tuple

$$\mathcal{A} = (Q, i, f, (\delta_a)_{a\in\mathtt{A}}),$$

in which Q is a *set of states*, i is a map from a one element set $1 = \{0\}$ to Q (i.e. an *initial state*), f is a map from Q to a two elements set $2 = \{0, 1\}$ (i.e. a *set of accepting states*), and δ_a is a map from Q to itself for all letters $a \in \mathtt{A}$. Given a word $u = a_1 \dots a_n$, the automaton *accepts* the map $[\![\mathcal{A}]\!](u)\colon 1 \to 2$ defined as:

$$[\![\mathcal{A}]\!](u) = f \circ \delta_u \circ i \qquad \text{where} \quad \delta_{a_1\dots a_n} = \delta_{a_n} \circ \dots \circ \delta_{a_1}.$$

We recognize here the standard definition of a deterministic automaton, in which a word u is accepted if the map $[\![\mathcal{A}]\!](u)$ is the constant 1, and rejected if it is the constant 0.

Vector space automata (automata weighted over a field). Now, we can use the same definition of an automaton, this time with Q a vector space (over, say, the field $\mathbb{R}$), i a linear map from $\mathbb{R}$ to Q, f a linear map from Q to $\mathbb{R}$ (seen as a $\mathbb{R}$-vector space as usual), and δ_a a linear map from Q to itself. In other words, we have used the same definition, but this time in the category of vector spaces. Given a word u, a vector space automaton $\mathcal{A}$ computes $[\![\mathcal{A}]\!](u)\colon \mathbb{R} \to \mathbb{R}$ as the composite described above. Since a linear map from $\mathbb{R}$ to $\mathbb{R}$ is only determined by the image of 1, this automaton can be understood as associating to each input word u the real number $[\![\mathcal{A}]\!](u)(1)$. We will informally refer to such automata in this section as *vector space automata*. Let us provide an example.

Leading example. For a word $u \in \{a, b, c\}^*$ let $|u|_a$ denote the number of occurrences of the letter a in u. Let us compute the map F which, given a word $u \in \{a, b, c\}^*$, outputs $2^{|u|_a}$ if it contains an even number of b's and no c's, and 0 in all other cases. This is achieved with the vector space automaton $\mathcal{A}^{\mathsf{vec}} = (Q^{\mathsf{vec}} = \mathbb{R}^2, i^{\mathsf{vec}}, f^{\mathsf{vec}}, \delta^{\mathsf{vec}})$ where for all $x, y \in \mathbb{R}$,

$$i^{\mathsf{vec}}(x) = (x, 0), \qquad \delta_a^{\mathsf{vec}}(x, y) = (2x, 2y), \qquad \delta_b^{\mathsf{vec}}(x, y) = (y, x),$$
$$f^{\mathsf{vec}}(x, y) = x, \qquad \delta_c^{\mathsf{vec}}(x, y) = (0, 0).$$

One easily checks that indeed $[\![\mathcal{A}^{\mathsf{vec}}]\!](u)(1) = F(u)$ for all words $u \in \mathtt{A}^*$.

Can we do better? It is well known from Schützenberger's seminal work that the vector space automaton $\mathcal{A}^{\mathsf{vec}}$ is minimal, both in an algebraic sense (to be described later) as well as at an intuitive level in the sense that no vector space automaton could recognize F with a dimension one vector space as configuration space: $\mathcal{A}^{\mathsf{vec}}$ is "dimension minimal."

However, let us think for one moment on how one would "implement" the function F as an online device that would get letters as input, and would modify its internal state accordingly. Would we implement concretely $\mathcal{A}^{\mathsf{vec}}$ directly? Probably not, since there is a more economic[2] way to obtain the same result: we can maintain 2^m where m is the number of a's seen so far, together with one bit for remembering whether the number of b's is even or odd. Such an automaton would start with 1 in its unique real valued register. Each time an a is met, the register is doubled, each time b is met, the bit is reversed, and when c is met, the register is set to 0. At the end of the input word, the automaton would output 0 or the value of the register depending on the current value of the bit. If we consider the configuration space that we use in this encoding, we use $\mathbb{R} \uplus \mathbb{R}$ instead of $\mathbb{R} \times \mathbb{R}$. Can we define an automata model that would faithfully implement this example?

A first generalization: disjoint unions of vector spaces. A way to achieve this is to interpret the generic notion of automata in the category of finite disjoint unions of vector spaces (*duvs*). One way to define such a *finite disjoint unions of vector spaces* is to use a finite set N of '*indices*' $p, q, r \dots$, and to each index p associate a vector space V_p. The 'space' represented is then $\{(p, \vec{v}) \mid p \in N,\ \vec{v} \in V_p\}$. A 'map' between duvs represented by (N, V) and (N', V') is then a pair $h : N \to N'$ together with a linear map f_p from V_p to $V'_{h(p)}$ for all $p \in N$. It can be seen as mapping each $(p, \vec{v}) \in N \times V_p$ to $(h(p), f_p(\vec{v}))$. Call this a *duvs map*. Such duvs maps are composed in a natural way. This defines a category, and hence we can consider *duvs automata* which are automata with a duvs for its state space, and transitions implemented by duvs maps.

For instance, we can pursue with the computation of F and provide a duvs automaton $\mathcal{A}^{\mathsf{duvs}} = (Q^{\mathsf{duvs}}, i^{\mathsf{duvs}}, f^{\mathsf{duvs}}, \delta^{\mathsf{duvs}})$ where $Q^{\mathsf{duvs}} = \{(s, x) \mid s \in \{\mathtt{even}, \mathtt{odd}\},\ x \in \mathbb{R}\}$ (considered as a disjoint union of vector spaces with indices `even` and `odd` and all associated vector spaces $V_{\mathtt{even}} = V_{\mathtt{odd}} = \mathbb{R}$). The maps can be conveniently defined as follows:

$$
\begin{array}{lll}
i^{\mathsf{duvs}}(x) = (\mathtt{even}, x) & \delta_a^{\mathsf{duvs}}(\mathtt{even}, x) = (\mathtt{even}, 2x) & \delta_a^{\mathsf{duvs}}(\mathtt{odd}, x) = (\mathtt{odd}, 2x) \\
f^{\mathsf{duvs}}(\mathtt{even}, x) = x & \delta_b^{\mathsf{duvs}}(\mathtt{even}, x) = (\mathtt{odd}, x) & \delta_b^{\mathsf{duvs}}(\mathtt{odd}, x) = (\mathtt{even}, x) \\
f^{\mathsf{duvs}}(\mathtt{odd}, x) = 0 & \delta_c^{\mathsf{duvs}}(\mathtt{even}, x) = (\mathtt{even}, 0) & \delta_c^{\mathsf{duvs}}(\mathtt{odd}, x) = (\mathtt{odd}, 0)
\end{array}
$$

This automaton computes the expected F. It is also obvious that such automata over finite disjoint unions of vector spaces generalize both deterministic finite state automata (using only 0-dimensional vector spaces), and vector space automata (using only one index). However, is it the joint generalization that we hoped for? The answer is no...

Minimization of duvs automata. We could think that the above automaton $\mathcal{A}^{\mathsf{duvs}}$ is minimal. However, it involved some arbitrary decisions when defining it. This can be seen in the fact that when δ_c^{duvs} is applied, we chose to not change the index (and set to null the real value): this is arbitrary, and we could have exchanged `even` and `odd`, or fixed it abitrarily to `even`, or to `odd`. All these *variants* would be equally valid for computing F.

It is a bit difficult at this stage to explain the non-minimality of these automata since we did not introduce the proper notions yet. Let us try at a high level, invoking some standard automata-theoretic concepts. The first remark is that every configuration in Q^{duvs} is 'reachable' in this automaton: indeed $(\mathtt{even}, x) = i^{\mathsf{duvs}}(x)$ and $(\mathtt{odd}, x) = \delta_b^{\mathsf{duvs}} \circ i^{\mathsf{duvs}}(x)$ for all $x \in \mathbb{R}$. Hence there is no hope to improve the automaton $\mathcal{A}^{\mathsf{duvs}}$ or one of its variants by some form of 'restriction to its reachable configurations'. Only 'quotienting of configurations' remains. However, one can show that none among $\mathcal{A}^{\mathsf{duvs}}$ and the variants mentioned above is the quotient of another. Keeping in mind the Myhill-Nerode equivalence, we should instead

[2] Under the reasonable assumption that maintaining a real is more costly than maintaining a bit.

merge the configurations $(\texttt{even}, 0)$ and $(\texttt{odd}, 0)$ since these are observationally equivalent:

$$f^{\texttt{duvs}} \circ \delta_u^{\texttt{duvs}}(\texttt{even}, 0) = 0 = f^{\texttt{duvs}} \circ \delta_u^{\texttt{duvs}}(\texttt{odd}, 0) \qquad \text{for all words } u \in \mathtt{A}^*.$$

However, the quotient duvs obtained by merging $(\texttt{even}, 0)$ and $(\texttt{odd}, 0)$, albeit not very intuitive, consists of one index associated to a two dimensional vector space, which is essentially an indexed version of the vector space automaton $\mathcal{A}^{\texttt{vec}}$ computed before. At this stage, we understand that minimising in the category of duvs is not very helpful, as we do not obtain the desired optimisation.

How to proceed from here. The only reasonable thing to do is indeed to merge $(\texttt{even}, 0)$ and $(\texttt{odd}, 0)$, but we have to be more careful about the precise meaning of 'quotient'. A possibility is to add explicitly equivalence classes in the definition of the automaton. However, category theory provides useful concepts and terminology for defining these objects: colimits, and more precisely the free co-completion of a category. In the previous paragraph, we have shown that the category of duvs – which is itself the free completion of Vec with respect to finite coproducts – is not a good ambient category for our purposes. We need more colimits, so that the notion of 'quotient' is further refined. At the other extreme, we could consider the free completion with respect to all colimits, which, informally, consists of objects obtained from the category using copying and gluing. We will explain later in Section 5 why we choose to not use this completion. Intuitively, by adding all colimits we glue the vector spaces "too much", and not only we loose a geometric intuition of the objects we are dealing with, but we may run into actual technical problems when it comes the existence of minimal automata.

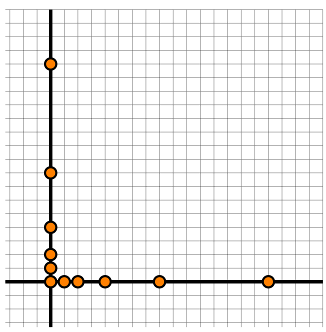
Instead, we restrict our attention to a class of colimits (which strictly contains coproducts) for which different spaces in the colimit can be "glued" together along subspaces, but which do not contain implicit self folding (i.e., such that an element of a vector space is not glued to a distinct element of the same vector space, directly or indirectly). E.g., we can describe 'two one-dimensional spaces, the 0-dimensional subspaces of which are identified through a linear bijection'. In this way we obtain the new category of *glued vector spaces* and *hybrid set-vector space automata*, corresponding to Glue(Vec)-automata in the rest of the paper.

Generic arguments of colimits provide the language for describing these objects, but do not solve the question of minimality. In particular, we are interested in automata whose space of configurations is a *finite* colimit belonging to the class described above. The categorical development in this work addresses the minimization problem for hybrid automata.

An intuition in the case of gluing of vector spaces. In the case of gluing of vector spaces, it is possible to isolate a combinatorial statement that plays a crucial role in the existence of minimal hybrid set-vector automata:

> *(a)* Any subset of a finite-dimensional vector space admits a minimal cover as a finite union of subspaces. *(b)* Furthermore, there is a unique such cover which is a union of subspaces which are incomparable with respect to inclusion.

For instance, in the original vector space automaton $\mathcal{A}^{\texttt{vec}}$, the states that are reachable in fact all belong to $\mathbb{R} \times \{0\} \cup \{0\} \times \mathbb{R}$, and this is the minimal cover as in (a) of these reachable configurations. This subset of $\mathbb{R}^2$ has the structure of two $\mathbb{R}$-spaces. These happen to intersect at $(0, 0)$, hence it is necessary to glue them at 0 to faithfully represent this set of reachable configurations. Thanks to (b) this decomposition is canonical, and hence can be used for describing the automaton.



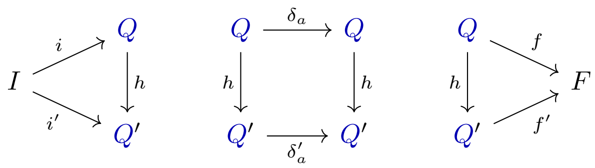
3 Automata in a category

In this section, we provide the general definition for a (finite word) automaton in a category. We also isolate properties guaranteeing the existence of minimal automata. Though presented differently, the material in the first subsection is essentially a slight variation around the work of Arbib and Manes [3], which introduced a notion of automaton in a category and, moreover, highlighted the connection between factorization systems of the ambient category, duality and minimization. In the remaining subsections we develop a refinement of this approach to minimization, and introduce a notion of factorization system through a subcategory.

3.1 Automata in a category, initial automaton, final automaton

▶ **Definition 3.1** (automata). Let $\mathcal{C}$ be a locally small category, I and F be objects of $\mathcal{C}$, and $\mathtt{A}$ be some alphabet. An *automaton $\mathcal{A}$ in the category $\mathcal{C}$* (over the alphabet $\mathtt{A}$), for short a *$\mathcal{C}, I, F$-automaton* (or simply *$\mathcal{C}$-automaton* when I and F are obvious in the context), is a tuple (Q, i, f, δ), where Q is an object in $\mathcal{C}$ (called the *state object*), $i\colon I \to Q$ and $f\colon Q \to F$ are morphisms in $\mathcal{C}$ (called *initial* and *final morphisms*), and $\delta\colon \mathtt{A} \to \mathcal{C}(Q, Q)$ is a function associating to each letter $a \in \mathtt{A}$ a morphism $\delta_a\colon Q \to Q$ in $\mathcal{C}$. We extend the function δ to $\mathtt{A}^*$ as with δ_ϵ being the identity morphism on Q and $\delta_{wa} = \delta_a \circ \delta_w$ for all $a \in \mathtt{A}$ and $w \in \mathtt{A}^*$.

A *morphism of $\mathcal{C}, I, F$-automata* $h\colon \mathcal{A} \to \mathcal{A}'$ is a morphism $h\colon Q \to Q'$ in $\mathcal{C}$ between the state objects which commutes with the initial, final and transition morphisms:



(1)

▶ **Example 3.2.** The two guiding instantiation of this definition are as follows. When the category $\mathcal{C}$ is Set, $I = 1$ and $F = 2$, we recover the standard notion of a deterministic and complete automaton (over the alphabet $\mathtt{A}^*$). In the second case, when $\mathcal{C}$ is Vec over a base field $\mathbb{K}$, $I = \mathbb{K}$ and $F = \mathbb{K}$, we obtain $\mathbb{K}$-weighted automata. Indeed, if Q is isomorphic to $\mathbb{K}^n$ for some natural number n, then linear maps $i\colon \mathbb{K} \to Q$ are in one-to-one correspondence with vectors $\mathbb{K}^n$. The same holds for linear maps $f\colon Q \to \mathbb{K}$, hence i and f are simply selecting an initial, respectively, a final vector.

▶ **Definition 3.3** (languages and language accepted). A *$\mathcal{C}, I, F$-language* (or *$\mathcal{C}$-language* when I and F are clear from the context) is a function $L\colon \mathtt{A}^* \to \mathcal{C}(I, F)$. We say that $\mathcal{A}$ *accepts the language L* if $L(w) = [\![\mathcal{A}]\!](w) := f \circ \delta_w \circ i$ for all $w \in \mathtt{A}^*$. Let $\mathsf{Auto}_{\mathcal{C}}(L)$ denote the *category of $\mathcal{C}, I, F$-automata for L*, that is, the category whose objects are $\mathcal{C}, I, F$-automata that accept the language L and whose arrows are morphisms of $\mathcal{C}, I, F$-automata[3].

▶ **Lemma 3.4.** *If the coproduct $\coprod_{w \in \mathtt{A}^*} I$ exists in $\mathcal{C}$, then $\mathsf{Auto}_{\mathcal{C}}(L)$ has an initial object $\mathtt{init}_{\mathcal{C}}(L)$. If the product $\prod_{w \in \mathtt{A}^*} F$ exists in $\mathcal{C}$, then $\mathsf{Auto}_{\mathcal{C}}(L)$ has a final object $\mathtt{final}_{\mathcal{C}}(L)$.*

In the case of Set, these automata are well known. The first one has as states $\mathtt{A}^*$, as initial state ε, and when it reads a letter a, its maps w to wa. Its final map sends the

[3] If $\mathcal{A}$ accepts the language L and $h\colon \mathcal{A} \to \mathcal{A}'$ is a morphism of $\mathcal{C}, I, F$-automata, then $\mathcal{A}'$ also accepts the language L. Hence, $\mathsf{Auto}_{\mathcal{C}}(L)$ is a 'connected component' in the category of all $\mathcal{C}, I, F$-automata.

state w to $L(w)(0)$. There exists one and exactly one morphism from this automaton to each automata for the same language. The generalisation of this construction is that the state space is the coproduct of A^*-many copies of I. The final automaton is known as the automaton of 'residuals'. Its set of states are the maps from $\mathtt{A}^*$ to 2. The initial state is L itself, and when reading a letter a, the state S is mapped to $w \mapsto S(aw)$. The final map sends S to $S(\varepsilon)$. The generalisation of this construction is that the state space is the product of A^*-many copies of F.

3.2 Factorizations through a subcategory

It is important in the development of this paper to distinguish the category $\mathsf{Auto}_{\mathcal{C}}(L)$ in which the initial and final automata for a language L exist (recall Lemma 3.4) and which contains 'infinite automata', from the subcategory, named $\mathsf{Auto}_{\mathcal{S}}(L)$ that is used for the concrete automata (with state object in $\mathcal{S}$) which are intended to be algorithmically manageable. In this section, we provide the concept of factorizing through a subcategory, which articulates the relation between these two categories.

▶ **Definition 3.5** (factorization through a subcategory)**.** Assume $\mathcal{S}$ is a subcategory of $\mathcal{C}$. An arrow $f\colon X \to Y$ in $\mathcal{C}$ is called *$\mathcal{S}$-small* if it factors through some object S of $\mathcal{S}$, that is, f is the composite $X \xrightarrow{u} S \xrightarrow{v} Y$ for some $u\colon X \to S$ and $v\colon S \to Y$.

A *factorization system through $\mathcal{S}$ on $\mathcal{C}$* (or simply a *factorization system on $\mathcal{C}$* if $\mathcal{C} = \mathcal{S}$) is a pair $(E_{\mathcal{S}}, M_{\mathcal{S}})$ where $E_{\mathcal{S}}$ and $M_{\mathcal{S}}$ are classes of arrows in $\mathcal{C}$ so that the codomains of all arrows in $E_{\mathcal{S}}$, the domains of all arrows in $M_{\mathcal{S}}$ are in $\mathcal{S}$, and the following conditions hold:

1. $E_{\mathcal{S}}$ and $M_{\mathcal{S}}$ are closed under composition with isomorphisms in $\mathcal{S}$, on the right, respectively left side.
2. All $\mathcal{S}$-small arrows in $\mathcal{C}$ have an *$(E_{\mathcal{S}}, M_{\mathcal{S}})$-factorization*, that is, if $f\colon X \to Y$ factors through an object of $\mathcal{S}$, then there exists $e \in E_{\mathcal{S}}$ and $m \in M_{\mathcal{S}}$, such that $f = m \circ e$.
3. The *unique $(E_{\mathcal{S}}, M_{\mathcal{S}})$-diagonalization property* holds: for each commutative diagram

$$\begin{array}{ccc} X & \xrightarrow{e} & T \\ {\scriptstyle f}\downarrow & \swarrow{\scriptstyle u} & \downarrow{\scriptstyle g} \\ S & \xrightarrow[m]{} & Y \end{array} \tag{2}$$

with $e \in E_{\mathcal{S}}$ and $m \in M_{\mathcal{S}}$, there exists a unique *diagonal*, that is, a unique morphism $u\colon T \to S$ such that $u \circ e = f$ and $m \circ u = g$.

Using standard techniques, we can prove that whenever $(E_{\mathcal{S}}, M_{\mathcal{S}})$ is a factorization system through $\mathcal{S}$ on $\mathcal{C}$, both classes $E_{\mathcal{S}}$ and $M_{\mathcal{S}}$ are closed under composition, their intersection consists of precisely the isomorphisms in $\mathcal{S}$, and, as expected, that $(E_{\mathcal{S}}, M_{\mathcal{S}})$-factorizations of $\mathcal{S}$-small morphisms are unique up to isomorphism.

▶ **Example 3.6.** Instantiating $(\mathcal{C}, \mathcal{S})$ to be $(\mathsf{Set}, \mathsf{Set}_{\mathsf{fin}})$ yields a natural factorization system through $\mathsf{Set}_{\mathsf{fin}}$ on Set (as the restriction of the standard (epi,mono)-factorization system on Set to $\mathsf{Set}_{\mathsf{fin}}$-small morphisms, i.e., the maps of finite image). Over these categories $\mathsf{Set}_{\mathsf{fin}}$-automata are deterministic finite state automata inside the more general category of Set-automata which are deterministic (potentially infinite) automata. The example $(\mathsf{Vec}, \mathsf{Vec}_{\mathsf{fin}})$ was already mentioned. In this case, being $\mathsf{Vec}_{\mathsf{fin}}$-small is equivalent to having finite rank.

Notice that for $(\mathcal{C}, \mathcal{S}) = (\mathsf{Set}, \mathsf{Set}_{\mathsf{fin}})$ or $(\mathsf{Vec}, \mathsf{Vec}_{\mathsf{fin}})$, the factorization systems through the subcategories are obtained simply by restricting the factorization systems on Set, respectively Vec. This is because, in these cases $\mathcal{S}$ is closed under quotients and subobjects in $\mathcal{C}$. The category $\mathtt{Glue}_{\mathsf{fin}}(\mathsf{Vec}_{\mathsf{fin}})$ used in this paper is closed under quotients, but in general not under subobjects (and this is the important reason for this extension of the standard notion of factorization). This is also a case in which there is a factorization system in the category $\mathcal{C}$, that coincide over $\mathcal{S}$ with factorizing through $\mathcal{S}$, but for which factorizing in $\mathcal{C}$ of an $\mathcal{S}$-small morphism and factorizing it through $\mathcal{S}$ yield different results.

A factorization system on $\mathcal{C}$ lifts naturally to categories of $\mathcal{C}$-valued functors. Automata being very close in definition to such a functor category, factorization systems also lift to them. Lemma 3.7 shows that this is also the case for factorization systems through $\mathcal{S}$, assuming of course that the input and output objects I and F belong to $\mathcal{S}$.

▶ **Lemma 3.7.** *If $(E_{\mathcal{S}}, M_{\mathcal{S}})$ is a factorization system through $\mathcal{S}$, then $(E_{\mathsf{Auto}_{\mathcal{S}}(L)}, M_{\mathsf{Auto}_{\mathcal{S}}(L)})$ is a factorization system through $\mathsf{Auto}_{\mathcal{S}}(L)$ on $\mathsf{Auto}_{\mathcal{C}}(L)$, where $E_{\mathsf{Auto}_{\mathcal{S}}(L)}$ (resp. $M_{\mathsf{Auto}_{\mathcal{S}}(L)}$) contains these $\mathsf{Auto}_{\mathcal{C}}(L)$-morphisms that belong to $E_{\mathcal{S}}$ (resp. to $M_{\mathcal{S}}$) as $\mathcal{C}$-morphisms.*

3.3 Minimization through a subcategory

In this section, we show how the joint combination of having initial and final automata for a language, as given by Lemma 3.4, together with a factorization system through a subcategory $\mathcal{S}$ yields the existence of a minimal $\mathcal{S}$-automaton for small $\mathcal{C}$-languages.

We make the *following assumptions*: $(E_{\mathcal{S}}, M_{\mathcal{S}})$ is a factorization system through $\mathcal{S}$ on $\mathcal{C}$, and L is a $\mathcal{C}$-language accepted by some $\mathcal{S}$-automaton such that there exist an initial $\mathtt{init}_{\mathcal{C}}(L)$ $\mathcal{C}$-automaton and a final $\mathcal{C}$-automaton $\mathtt{final}_{\mathcal{C}}(L)$ for L.

▶ **Definition 3.8** (minimal automaton)**.** The *minimal $\mathcal{C}$-automaton for L*, denoted $\mathtt{min}_{\mathcal{S}}(L)$, is the[4] $\mathcal{S}$-automaton for L obtained by $(E_{\mathsf{Auto}_{\mathcal{S}}(L)}, M_{\mathsf{Auto}_{\mathcal{S}}(L)})$-factorization of the unique $\mathsf{Auto}_{\mathcal{S}}(L)$-small morphism from $\mathtt{init}_{\mathcal{C}}(L)$ to $\mathtt{final}_{\mathcal{C}}(L)$.

▶ **Theorem 3.9.** *For all $\mathcal{S}$-automata $\mathcal{A}$ for L satisfying the above assumptions, we have*

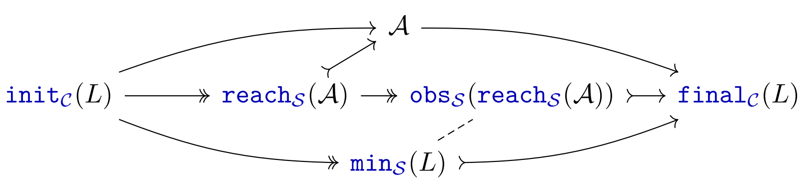
$$\mathtt{min}_{\mathcal{S}}(L) \cong \mathtt{obs}_{\mathcal{S}}(\mathtt{reach}_{\mathcal{S}}(\mathcal{A})) \cong \mathtt{reach}_{\mathcal{S}}(\mathtt{obs}_{\mathcal{S}}(\mathcal{A}))\,,$$

in which

- *$\mathtt{reach}_{\mathcal{S}}(\mathcal{A})$ is the result of applying an $(E_{\mathsf{Auto}_{\mathcal{S}}(L)}, M_{\mathsf{Auto}_{\mathcal{S}}(L)})$-factorization to the unique $\mathsf{Auto}_{\mathcal{S}}(L)$-morphism from $\mathtt{init}_{\mathcal{C}}(L)$ to $\mathcal{A}$, and*
- *$\mathtt{obs}_{\mathcal{S}}(\mathcal{A})$ is the result of applying an $(E_{\mathsf{Auto}_{\mathcal{S}}(L)}, M_{\mathsf{Auto}_{\mathcal{S}}(L)})$-factorization to the unique $\mathsf{Auto}_{\mathcal{S}}(L)$-morphism from $\mathcal{A}$ to $\mathtt{final}_{\mathcal{C}}(L)$.*

This theorem does not only state the existence of a minimal automaton, it also makes transparent how to make effective its construction: if one possesses both an implementation of $\mathtt{reach}_{\mathcal{S}}$ and $\mathtt{obs}_{\mathcal{S}}$, then their sequencing minimises an input automaton. From the above theorem it immediately follows that $\mathtt{min}_{\mathcal{S}}(L)$ is both an $E_{\mathsf{Auto}_{\mathcal{S}}(L)}$-quotient of a $M_{\mathsf{Auto}_{\mathcal{S}}(L)}$-subobject of $\mathcal{A}$ and a $M_{\mathsf{Auto}_{\mathcal{S}}(L)}$-subobject of an $E_{\mathsf{Auto}_{\mathcal{S}}(L)}$-quotient of $\mathcal{A}$: the minimal automaton divides every other automaton for the language.

Proof idea. The proof is contained in the following commutative diagram, in which $\twoheadrightarrow$ denotes $\mathsf{Auto}_{\mathcal{C}}(L)$-morphisms in $E_{\mathsf{Auto}_{\mathcal{S}}(L)}$, and $\rightarrowtail$ $\mathsf{Auto}_{\mathcal{C}}(L)$-morphisms in $M_{\mathsf{Auto}_{\mathcal{S}}(L)}$:



[4] It is unique up to isomorphism according to the diagonal property.

That $\texttt{obs}_{\mathcal{S}}(\texttt{reach}_{\mathcal{S}}(\mathcal{A}))$ is an $(E_{\mathsf{Auto}_{\mathcal{S}}(L)}, M_{\mathsf{Auto}_{\mathcal{S}}(L)})$-factorization of the unique $\mathsf{Auto}_{\mathcal{C}}(L)$-morphism from $\texttt{init}_{\mathcal{C}}(L)$ to $\texttt{final}_{\mathcal{C}}(L)$ follows since $E_{\mathsf{Auto}_{\mathcal{S}}(L)}$ is closed under composition. By the unique diagonal property, it is isomorphic to $\texttt{min}_{\mathcal{S}}(L)$. The case $\texttt{reach}_{\mathcal{S}}(\texttt{obs}_{\mathcal{S}}(\mathcal{A}))$ is symmetric. ◀

3.4 A special case of factorization through

So far, the description of factorization and minimization of automata is very generic. Hereafter, the classes $E_{\mathcal{S}}$ and $M_{\mathcal{S}}$ are constructed along a particular principle which we describe now. In the next sections we will instantiate this construction when $\mathcal{S}$ is the subcategory $\texttt{Glue}_{\mathsf{fin}}(\mathsf{Vec}_{\mathsf{fin}})$ of glued vector spaces.

In this section we fix an (E, M)-factorization system on $\mathcal{C}$ and a subcategory $\mathcal{S} \hookrightarrow \mathcal{C}$.

▶ **Definition 3.10.** An *$\mathcal{S}$-extremal epimorphism*[5] in $\mathcal{C}$ is an arrow $e\colon X \to S$ in $\mathcal{C}$, with S an object in $\mathcal{S}$, such that if $e = m \circ g$ where m is in M with domain in $\mathcal{S}$, then m is an isomorphism. We set $M_{\mathcal{S}}$ to be the class of arrows in M with domain in $\mathcal{S}$, and $E_{\mathcal{S}}$ to be the class of $\mathcal{S}$-extremal epimorphisms.

▶ **Definition 3.11.** An *$M_{\mathcal{S}}$-subobject in $\mathcal{S}$* of an object X of $\mathcal{C}$ is an equivalence class up to isomorphism of a morphism $S \to X$ belonging to M, where S is an object of $\mathcal{S}$. The $M_{\mathcal{S}}$-subobject $S \to X$ is called *proper* if it is not an isomorphism.

▶ **Lemma 3.12.** *Assume the following conditions hold:*
1. *all arrows in M are monomorphisms in $\mathcal{C}$,*
2. *$\mathcal{S}$ is closed under E-quotients, i.e., if $e\colon S \to T$ is in E with S in $\mathcal{S}$, then T is isomorphic to an object of $\mathcal{S}$,*
3. *the intersection of a nonempty set of $M_{\mathcal{S}}$-subobjects of an object X of $\mathcal{C}$ exists and is an $M_{\mathcal{S}}$-subobject of X, and,*
4. *the pullback of an $M_{\mathcal{S}}$-subobject $m\colon S \to T$ of T along a morphism $T' \to T$ in $\mathcal{S}$ is an $M_{\mathcal{S}}$-subobject of T'.*

Then $(E_{\mathcal{S}}, M_{\mathcal{S}})$ is a factorization system through $\mathcal{S}$ on $\mathcal{C}$.

The next lemma ensures that condition 3 of Lemma 3.12 can be replaced with the weaker version involving only binary intersections of $M_{\mathcal{S}}$-subobjects, provided that any infinite descending chain of $M_{\mathcal{S}}$-subobjects eventually stabilises (of course, up to isomorphism).

▶ **Lemma 3.13.** *Assume that there are no infinite descending chains of proper $M_{\mathcal{S}}$-subobjects*

$$X \longleftarrow\!\!\prec S_1 \longleftarrow\!\!\prec S_2 \longleftarrow\!\!\prec \dots$$

and furthermore that the intersection of any two $M_{\mathcal{S}}$-subobjects of an object X of $\mathcal{C}$ exists and is an $M_{\mathcal{S}}$-subobject of X, then condition 3 in the hypothesis of Lemma 3.12 holds.

The proof simply uses finite partial intersections in order to create a strictly descending chain of $M_{\mathcal{S}}$-subobjects. By assumption, this construction has to stop, and the last element of the sequence happens to be the intersection of the entire family.[6]

[5] Note that $\mathcal{S}$-extremal epimorphisms need not be epimorphisms in $\mathcal{C}$.

[6] The attentive reader will have recognised in this argument part of the reason why every subset of a finite-dimensional vector space admits a minimal cover as a finite union of subspaces.

4 Gluing of categories

We turn now to the central construction of this paper: given a category $\mathcal{C}$ and a subcategory $\mathcal{S}$, we construct a category $\mathtt{Glue}(\mathcal{C})$ of "gluings of objects in $\mathcal{C}$" that has both $\mathcal{C}$ and $\mathtt{Glue}_{\mathsf{fin}}(\mathcal{S})$ – the category of "finite gluings of objects in $\mathcal{S}$" – as subcategories. Under proper assumptions on $\mathcal{C}$ and $\mathcal{S}$, the resulting pair $(\mathtt{Glue}(\mathcal{C}), \mathtt{Glue}_{\mathsf{fin}}(\mathcal{S}))$ satisfies the assumption required for constructing minimisable automata for $\mathtt{Glue}(\mathcal{C})$-languages. Taking $\mathcal{C} = \mathsf{Vec}$ and $\mathcal{S} = \mathsf{Vec}_{\mathsf{fin}}$ we obtain the construction informally described in Section 2.

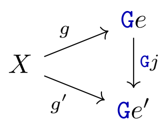
Throughout this section we assume that $\mathcal{C}$ is equipped with a (E, M)-factorization system consisting of strong epimorphisms and monomorphisms.

4.1 The free gluing of a category

When $\mathcal{C}$ is small, it is well known that the Yoneda embedding of $\mathcal{C}$ into the category of presheaves over $\mathcal{C}$ is a free completion of $\mathcal{C}$ under colimits of small diagrams. For a possibly large category, one has to consider instead the category of small presheaves, i.e. small colimits of representable ones, see for example [9]. For our purposes, we found more illuminating and direct to use a syntactic way of describing the colimit completion of a category.

The category of diagrams. Assume $\mathcal{C}$ is a locally small category. The *free colimit completion of $\mathcal{C}$* is the category $\mathsf{Diag}(\mathcal{C})$ whose objects are *diagrams* $\mathtt{F}\colon \mathcal{D} \to \mathcal{C}$ and morphisms between two diagrams $\mathtt{F}\colon \mathcal{D} \to \mathcal{C}$ and $\mathtt{G}\colon \mathcal{E} \to \mathcal{C}$ will be given in Definition 4.1.

To this end we define an equivalence relation on arrows from an arbitrary object X of $\mathcal{C}$ to the objects in the image of $\mathtt{G}$. Assume e, e' are objects in $\mathcal{E}$. We consider the least equivalence relation $\sim_{\mathtt{G}}$ which contains all pairs (g, g'), where $g\colon X \to \mathtt{G}e$, $g'\colon X \to \mathtt{G}e'$ are such that there exists $j\colon e \to e'$ a map in $\mathcal{E}$ with $\mathtt{G}j \circ g = g'$, i.e., the diagram on the right commutes.



We denote by $\widehat{\mathtt{G}}(X)$ the equivalence classes of the relation $\sim_{\mathtt{G}}$.

▶ **Definition 4.1.** A *morphism between diagrams* $\mathtt{F}\colon \mathcal{D} \to \mathcal{C}$ and $\mathtt{G}\colon \mathcal{E} \to \mathcal{C}$ is a map f which associates to each object d in $\mathcal{D}$ an equivalence class $f(d) \in \widehat{\mathtt{G}}(\mathtt{F}d)$, such that whenever $u\colon d \to d'$ is a morphism in $\mathcal{D}$ and $g\colon \mathtt{F}d' \to \mathtt{G}e$ is in the equivalence class $f(d')$, then $g \circ \mathtt{F}u$ is in the equivalence class $f(d)$.

The subcategory of gluings. We are now ready to define the category $\mathtt{Glue}(\mathcal{C})$, which is a restriction of $\mathsf{Diag}(\mathcal{C})$ to M-diagrams, that is, to diagrams that intuitively 'do not quotient'. Recall that $\mathcal{C}$ has a factorization system (E, M) in which E are the strong epimorphisms and M are the monomorphisms.

▶ **Definition 4.2** (*glued category*). An *M-cocone* over a diagram $\mathtt{F}\colon \mathcal{D} \to \mathcal{C}$ is a cocone $(u_d\colon \mathtt{F}d \to X)_{d \in \mathcal{D}}$ such that all the structural components of the cocone u_d are in M. An *M-diagram* is a diagram that has an M-cocone.

The *glued category* $\mathtt{Glue}(\mathcal{C})$ is the subcategory of $\mathsf{Diag}(\mathcal{C})$ over the M-diagrams $\mathtt{F}\colon \mathcal{D} \to \mathcal{C}$. Let $\mathtt{Glue}_{\mathsf{fin}}(\mathcal{C})$ the subcategory of $\mathtt{Glue}(\mathcal{C})$ that has as objects the finite diagrams of $\mathtt{Glue}(\mathcal{C})$.

Notice that, if $\mathtt{F}$ is such a diagram, then we can show that for each morphism $v\colon d \to d'$ in $\mathcal{D}$, we have that $\mathtt{F}u\colon \mathtt{F}d \to \mathtt{F}d'$ is in M (however this is not a characterisation). Also, if there exists a universal cocone for an M-diagram, then this cocone is in particular an M-cocone.

▶ **Lemma 4.3.** *If $\mathcal{C}$ is cocomplete, then $\mathtt{Glue}(\mathcal{C})$ is a full reflective subcategory of $\mathsf{Diag}(\mathcal{C})$, and hence $\mathtt{Glue}(\mathcal{C})$ is a cocomplete category. If $\mathcal{C}$ is furthermore complete, then so is $\mathtt{Glue}(\mathcal{C})$.*

In the automata theoretic application we have in mind, we use this category in order to construct the initial and final automata for a language.

4.2 A factorization system through finite gluings

The category of most interest for us is the full subcategory $\mathtt{Glue}_{\mathsf{fin}}(\mathcal{S})$ of $\mathtt{Glue}(\mathcal{C})$ which consists of the finite M-diagrams over $\mathcal{S}$. In this section we construct in particular, under suitable assumptions, a factorization system through $\mathtt{Glue}_{\mathsf{fin}}(\mathcal{S})$ on $\mathtt{Glue}(\mathcal{C})$, making use of Lemma 3.12. For $\mathcal{S} = \mathsf{Vec}_{\mathsf{fin}}$, this is the category of 'finite gluings of finite vector spaces' that we longly introduced in Section 2.

We define the following classes of morphisms in $\mathtt{Glue}(\mathcal{C})$.

- $Epi_{\mathtt{Glue}(\mathcal{C})}$ consists of the morphisms $f\colon \mathtt{F} \to \mathtt{G}$, where $\mathtt{F}\colon \mathcal{D} \to \mathcal{C}$ and $\mathtt{F}\colon \mathcal{E} \to \mathcal{C}$, such that for all e in $\mathcal{E}$ there exists a representative $f_d\colon \mathtt{F}d \to \mathtt{G}e'$ in the equivalence class $f(d)$ and a morphism $u\colon \mathtt{G}e \to \mathtt{G}e'$, so that $u \sim_{\mathtt{G}} id_{\mathtt{G}e}$ and u factors through the image of f_d.
- $Mono_{\mathtt{Glue}(\mathcal{C})}$ consists of morphisms $f\colon \mathtt{F} \to \mathtt{G}$ such that for all morphisms $u\colon X \to \mathtt{F}d$ and $v\colon X \to \mathtt{F}d'$ such that $f_d \circ u \sim_{\mathtt{G}} f_{d'} \circ v$ (for f_d and $f_{d'}$ in the equivalence classes $f(d)$, respectively $f(d')$), we have that $u \sim_{\mathtt{F}} v$.

One can easily verify that the arrows in $Mono_{\mathtt{Glue}(\mathcal{C})}$ are exactly the monomorphisms in $\mathtt{Glue}(\mathcal{C})$.[7] The next lemma establishes that under mild conditions on $\mathcal{C}$ we have a (strong epi, mono) factorization system on $\mathtt{Glue}(\mathcal{C})$.

▶ **Lemma 4.4.** *Assume $\mathcal{C}$ has intersections. Then $(Epi_{\mathtt{Glue}(\mathcal{C})}, Mono_{\mathtt{Glue}(\mathcal{C})})$ is a (strong epi, mono) factorization system on $\mathtt{Glue}(\mathcal{C})$.*

In what follows we say that a subcategory $\mathcal{S}$ of $\mathcal{C}$ is *well-behaved* if it satisfies the hypothesis of Lemmas 3.12 and 3.13 with respect to the (strong epi, mono) factorization system on $\mathcal{C}$. (In fact condition 3 of Lemma 3.12 can be replaced by its binary version.)

▶ **Theorem 4.5.** Assume $\mathcal{C}$ has intersections and pullbacks. If the subcategory $\mathcal{S}$ of $\mathcal{C}$ is well-behaved, then $\mathtt{Glue}_{\mathsf{fin}}(\mathcal{S})$ is a well-behaved subcategory of $\mathtt{Glue}(\mathcal{C})$.

Some ideas about the proof. This result is an application of Lemmas 3.12 and 3.13. The central combinatorial aspect of this statement is that there exists no infinite strictly descending chains of $Mono_{\mathtt{Glue}(\mathcal{C})}$-subobjects in $\mathtt{Glue}_{\mathsf{fin}}(\mathcal{S})$. For the sake of contradiction, let us consider a descending sequence of diagrams from $\mathtt{Glue}_{\mathsf{fin}}(\mathcal{S})$:

$$\mathtt{F}_0 \xleftarrow{f_1} \mathtt{F}_1 \xleftarrow{f_2} \mathtt{F}_2 \xleftarrow{f_3} \dots$$

We have to prove that it is ultimately constant (up to isomorphism). Let the diagrams be $\mathtt{F}_i : \mathcal{D}_i \to \mathcal{S}$ for all i. The first step is to consider an *aggregation of mono-diagrams*, that is, we construct a big diagram that aggregates all the $\mathtt{F}_i$'s. At the level of objects this diagram contains the disjoint unions of the $\mathcal{D}_i$'s. We call the objects originating from $\mathcal{D}_i$ of *rank i*. Secondly, we prove a *global homogeneity* property of $\mathtt{F}$: given two arrows $g\colon X \to \mathtt{F}d$, $g'\colon X \to \mathtt{F}d'$ with d, d' at the same rank i, then $g \sim_{\mathtt{F}} g'$ if and only if $g \sim_{\mathtt{F}_i} g'$. Finally we prove the existence of an isomorphism g_j from $\mathtt{F}_{j-1}$ to $\mathtt{F}_j$ for some j by analysing the structure of $\mathtt{F}$ and using König's lemma. ◀

We come back to the leading example of $\mathtt{Glue}_{\mathsf{fin}}(\mathsf{Vec}_{\mathsf{fin}})$-automata. Applying Theorem 4.5 for $(\mathcal{C}, \mathcal{S}) = (\mathsf{Vec}, \mathsf{Vec}_{\mathsf{fin}})$ we obtain a factorization system through $\mathtt{Glue}_{\mathsf{fin}}(\mathsf{Vec}_{\mathsf{fin}})$ on $\mathtt{Glue}(\mathsf{Vec})$. Using Lemma 4.3 and Theorem 3.9 we derive that hybrid set-vector automata are minimisable.

[7] As a side remark, we should mention that these classes of arrows correspond precisely to the natural transformations between the induced presheaves that are pointwise injective.

▶ **Corollary 4.6.** *For any* $\mathtt{Glue}(\mathsf{Vec})$*-language accepted by some* $\mathtt{Glue}_{\mathsf{fin}}(\mathsf{Vec}_{\mathsf{fin}})$*-automaton there exists a minimal* $\mathtt{Glue}_{\mathsf{fin}}(\mathsf{Vec}_{\mathsf{fin}})$*-automaton. In particular, any* Vec*-language accepted by a* $\mathsf{Vec}_{\mathsf{fin}}$*-automaton has a minimal* $\mathtt{Glue}_{\mathsf{fin}}(\mathsf{Vec}_{\mathsf{fin}})$*-automaton.*

For the language described in Section 2, and for which the minimal vector automaton has a two-dimensional state space, we obtain a minimal $\mathtt{Glue}_{\mathsf{fin}}(\mathsf{Vec}_{\mathsf{fin}})$-automaton obtained by glueing two one-dimensional spaces at 0. Formally, this is an M-diagram $\mathtt{F}\colon \mathcal{D} \to \mathsf{Vec}_{\mathsf{fin}}$ where $\mathcal{D}$ is a three object poset $\{\bot, 0, 1\}$ with $\bot \leq 0$ and $\bot \leq 1$. The functor $\mathtt{F}$ maps $0, 1$ to one-dimensional spaces, $\bot$ to the zero-dimensional space and the morphisms $\bot \leq 0$ and $\bot \leq 1$ to its inclusions in the respective one-dimensional spaces.

For another example, consider the language which to a word $u \in a^*$ associates the value $\cos(\alpha|u|)$ for some α which is not a rational multiple of π, and whose minimal vector space automaton has a two-dimensional state space. If we used the factorization in $\mathtt{Glue}(\mathsf{Vec})$ we would obtain a glueing of infinitely many one-dimensional subspaces (obtained by rotations with angle α). Thus, it is crucial for our setting to use the *factorization system through* $\mathtt{Glue}_{\mathsf{fin}}(\mathsf{Vec}_{\mathsf{fin}})$. In this case, the minimal $\mathtt{Glue}_{\mathsf{fin}}(\mathsf{Vec}_{\mathsf{fin}})$-automaton also has just a two-dimensional vector space of states.

5 Conclusion

We have introduced a new way to construct automata which, thanks to category-theoretic insights, admits minimal automata 'by design'. The introductory example of hybrid set-vector automata is a convincing instance of this approach, which has both algorithmic merits (in succinctness of the encoding of the state space) and theoretical merits (in that there exists minimal automata). The closest work to our knowledge is the work of Lombardy and Sakarovitch [12] which studies the sequentialisability of weighted automata; in the framework of this paper, this is answering the question whether a vector space automaton is equivalent to a hybrid set-vector automaton for which the state space consists of dimension 1 vector spaces only, glued at 0 (the problem remains open).

At the categorical level, we should say a few words regarding our design choices. First why not use more familiar co-completions such as the Ind-completion of the free co-completion? The answer is that if we did so, we would not obtain the desired behaviour when we restrict our attention to the 'finite' automata. For example finite filtered colimits are not very interesting, while the freely added finite colimits of vector spaces are not closed under quotients in the free co-completion, thus the work in the previous sections cannot be applied.

Another question one may ask is why we haven't used coalgebras, as in [2] or in the work of [1] on well-pointed coalgebras. First, the factorization through a subcategory, which plays a crucial role in our work, is not developed in that setting. Secondly, we believe that the functorial approach to automata, which neatly combines the dual narrative of automata seen as both algebras and coalgebras is worth saying. As we show in [8], we can employ this framework for minimizing subsequential transducers à la Choffrut [7] (by interpreting them as automata in a Kleisli category). This is also an example in which the conditions in Lemma 3.4 are not necessary. We believe, that at least in that situation the functorial approach works slightly smoother than the coalgebraic one [11]. Also, by changing the input category, we can further extend this work to capture tree automata or algebras (for instance monoids).

In the particular model of hybrid set-vector automata the problem of effectiveness remains: we have proved the existence of a minimal automaton for a language, but obtaining the reachable configurations in an effective way is the subject of ongoing work.

References

1 Jirí Adámek, Stefan Milius, Lawrence S. Moss, and Lurdes Sousa. Well-pointed coalgebras. *Logical Methods in Computer Science*, 9(3), 2013.

2 Jiří Adámek, Filippo Bonchi, Mathias Hülsbusch, Barbara König, Stefan Milius, and Alexandra Silva. A coalgebraic perspective on minimization and determinization. In *Proceedings of the 15th International Conference on Foundations of Software Science and Computational Structures*, FOSSACS'12, pages 58–73, Berlin, Heidelberg, 2012. Springer-Verlag. URL: `http://dx.doi.org/10.1007/978-3-642-28729-9_4`, `doi:10.1007/978-3-642-28729-9_4`.

3 Michael A. Arbib and Ernest G. Manes. Adjoint machines, state-behavior machines, and duality. *Journal of Pure and Applied Algebra*, 6(3):313 – 344, 1975. URL: `http://www.sciencedirect.com/science/article/pii/0022404975900286`, `doi:http://dx.doi.org/10.1016/0022-4049(75)90028-6`.

4 Nick Bezhanishvili, Clemens Kupke, and Prakash Panangaden. *Minimization via Duality*, pages 191–205. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. URL: `http://dx.doi.org/10.1007/978-3-642-32621-9_14`, `doi:10.1007/978-3-642-32621-9_14`.

5 Filippo Bonchi, Marcello Bonsangue, Michele Boreale, Jan Rutten, and Alexandra Silva. A coalgebraic perspective on linear weighted automata. *Inf. Comput.*, 211:77–105, February 2012. URL: `http://dx.doi.org/10.1016/j.ic.2011.12.002`, `doi:10.1016/j.ic.2011.12.002`.

6 Filippo Bonchi, Marcello M. Bonsangue, Helle Hvid Hansen, Prakash Panangaden, Jan J. M. M. Rutten, and Alexandra Silva. Algebra-coalgebra duality in Brzozowski's minimization algorithm. *ACM Trans. Comput. Log.*, 15(1):3:1–3:29, 2014.

7 Christian Choffrut. A generalization of Ginsburg and Rose's characterization of G-S-M mappings. In *ICALP*, volume 71 of *Lecture Notes in Computer Science*, pages 88–103. Springer, 1979.

8 Thomas Colcombet and Daniela Petrişan. Automata minimization: a functorial approach. *CALCO*, 72:8:1–8:15, 2017.

9 Brian J. Day and Stephen Lack. Limits of small functors. *Journal of Pure and Applied Algebra*, 210(3):651 – 663, 2007. URL: `http://www.sciencedirect.com/science/article/pii/S0022404906002635`, `doi:http://dx.doi.org/10.1016/j.jpaa.2006.10.019`.

10 J. A. Goguen. Minimal realization of machines in closed categories. *Bull. Amer. Math. Soc.*, 78(5):777–783, 09 1972. URL: `http://projecteuclid.org/euclid.bams/1183533991`.

11 Helle Hvid Hansen. Subsequential transducers: a coalgebraic perspective. *Inf. Comput.*, 208(12):1368–1397, 2010.

12 Sylvain Lombardy and Jacques Sakarovitch. Sequential? *Theor. Comput. Sci.*, 356(1-2):224–244, 2006. `doi:10.1016/j.tcs.2006.01.028`.

13 M.P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2):245 – 270, 1961. URL: `http://www.sciencedirect.com/science/article/pii/S001999586180020X`, `doi:http://dx.doi.org/10.1016/S0019-9958(61)80020-X`.