A robust class of transductions beyond functionality

Nicolas Baudru
Aix Marseille Univ, CNRS, LIS, Marseille, France

Louis-Marie Dando
IMDEA Software Institute, Madrid, Spain

Nathan Lhote
Aix Marseille Univ, CNRS, LIS, Marseille, France

Pierre-Alain Reynier
Aix Marseille Univ, CNRS, LIS, Marseille, France

Abstract

The class of regular functions constitutes a new pillar of the theory of word transductions: it admits multiple characterizations (deterministic 2-way transducers, streaming string transducers, regular function expressions and MSO transductions), and numerous closure properties. In this work, we propose a new extension of this class beyond functionality, which enjoys multiple characterizations, including a Kleene-like theorem, as well as several closure properties.

The starting point of our work is an extension of the set of operators introduced by Alur *et al* to characterize regular functions in two directions: first, we allow an ambiguous version of the sum operator, and second, we introduce the Hadamard star of a transduction f, which maps a word u to the language $f(u)^*$. We show this new class of transductions corresponds to (a decidable subclass of) a natural extension of streaming string transducers where the register updates are enriched to allow any regular expression involving the registers. We also identify an expressively equivalent restriction of *non-deterministic* 2-way transducers, which we call *weakly ambiguous*, based on a structural constraint on the ambiguity.

The resulting class of transductions inherits many of the closure properties of regular functions (apart from composition). In addition, it is closed by Hadamard star, union and pre-composition with regular functions. Finally, we show one can effectively decide whether a 2-way transducer is weakly ambiguous.

2012 ACM Subject Classification Models of computation; Formal languages

Keywords and phrases transducers, regular expressions, register automata

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Introduction

One of the fundamental results in language theory is the characterization of regular languages by means of finite state automata, regular expressions and Monadic Second-Order formulae. While automata are particularly convenient for algorithmic purposes, regular expressions allow specifications in a declarative manner, and are widely used in practical applications.

This theory has been extended in numerous directions, including finite and infinite trees. Another natural extension is moving from languages to transductions, namely, functions that map input words over an input alphabet A to (sets of) output words over an output alphabet B. In this setting, transducers constitute a fundamental extension of automata. Contrary to finite state automata, transducers are not robust under classical modifications in the model, as nondeterminism and two-wayness increase their expressive power.

The class of functions realized by deterministic two-way transducers, so-called *regular functions*, has attracted recently a strong interest [6, 7, 17, 14, 20, 13, 12]. It is very expressive and allows the description of natural transformations that are not definable by one-way transducers (*e.g.* duplicate the input word, or produce its mirror image). This class enjoys a logical characterization using Monadic Second-Order graph transductions interpreted on

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:33
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

strings [16], is equivalent to functional or unambiguous two-way transducers [16], and can be defined using the model of copyless streaming string transducers (SST) [1], which are a one-way model updating write-only registers which store strings over the output alphabet.

In [3], Alur, Freilich and Raghothaman showed a Kleene-like theorem for regular functions. They introduced a set of combinators to form expressions, called *regular function expressions* (RFEs), and showed their equivalence regarding expressiveness with the model of SST. RFEs allow *unambiguous* versions of natural operators such as sum, Cauchy and Hadamard product, and Kleene iteration, as well as their mirror images and another more involved iteration operator. Alternative proofs of this equivalence have been proposed in the last years, starting from deterministic [14] or unambiguous [5] two-way transducers.

The work in this paper follows this trend. Here, we aim to get a Kleene-like theorem for a class of transductions that goes *beyond functionality.* Our starting point has been to extend RFEs to non-functional transductions in a very natural way, by allowing an *ambiguous* version of the sum operator and introducing the *Hadamard star* of a transduction f, that maps a string u to $f(u)^*$. The new expressions are called *regular relation expressions* (RREs). They define a new class of transductions that we believe is relevant for the following reasons. First it contains regular functions and is expressive enough to capture several interesting non-functional transductions, such as:

- The Subsequence relation that associates to each word u all the subsequences of u.
- The Iterative-Star relation, with domain $(ba^+)^*b$, that associates to each word $ba^{n_1}ba^{n_2}\dots ba^{n_i}b$ all the words $ba^{x_1 n_1}ba^{x_2 n_2}\dots ba^{x_i n_i}b$ with $x_1,\dots,x_i \in \mathbb{N}$.
- The k-Evaluator relation that associates to each regular expression whose number of nested union or Kleene star combinators is less than k every word belonging to its associated language.

Note that the last two cannot be defined by a nondeterministic SST.

Then, this class inherits all of the closure properties of regular functions (except for composition), and is additionally closed under union, Hadamard star and pre-composition with regular functions. Last but not least, it admits characterizations in terms of two quite natural extensions of automaton-like models that we also introduce. The first one, called *SST with regular updates* (RSST), is an SST that produces regular expressions over the output alphabet B. The associated transduction maps a word to the language denoted by its corresponding output in the RSST. If the number of nested union and Kleene-star combinators in the output expressions is bounded, then the RSST is called *nl-bounded.* The second one, called *weakly ambiguous two-way transducer* (W2NFT), is a two-way transducer with a total order over its set of states. Because of nondeterminism, several runs are possible for a given input word. To be weakly ambiguous, we require that all these runs *synchronize* on the largest state. Now, we formally state the main result of this paper.

▶ **Theorem 1.** *Let f be a word-to-word transduction. The following are equivalent:*

- *f is denoted by a regular relation expression.*
- *f is recognized by a weakly ambiguous two-way transducer.*
- *f is recognized by a nl-bounded streaming string transducer with regular updates.*

Moreover, one can decide whether a non-deterministic transducers is weakly ambiguous and whether an RSST is nl-bounded.

Organization of the paper The models we consider are presented in Section 2. Our main result, together with important closure properties of our class of transductions, are given in Section 3. Section 4 describes the translation of a W2NFT into an RRE. Lastly, a discussion is conducted in Section 5. Omitted proofs can be found in the Appendix.

Related works In [2], a non-deterministic version of SST is studied. In particular, it is shown that the model is equivalent to non-deterministic MSO transductions (NMSOT). This form of non-determinism is incomparable to the one we study in this work: in NMSOT, every input word is mapped to a finite set of output words, while we may have infinite sets using Hadamard star. On the other hand, NMSOT allow to encode the transduction that maps any word u to the set of words vv, with v subword of u. This is not possible in our model as it requires to make the same guess of the positions to keep twice. In addition, to the best of our knowledge, no presentation of NMSOT by means of expressions is known.

In [8], the authors aim at exhibiting a set of expressions to capture the expressiveness of the whole class of non-deterministic two-way transducers. This constitutes a challenging open problem, and a solution is provided for the case where both the input and output alphabets are unary. In [4], a Kleene-like theorem is given for the whole class of non-deterministic 2-way transducers. However, the regular expressions proposed are rather machine oriented as they roughly encode the moves of the transducer, step-by-step. There is thus a lack of high-level operators, more amenable to an easy specification of transformations.

In [12], the authors use an incomparable definition of RRE without Hadamard star but with an ambiguous version of the Cauchy product and chained star operator. They show that such RREs can be expressed as the pre-composition of a 2-way reversible transducer with a 1-way-nondeterministic transducer. The latter parses the input word, non-deterministically adding parenthesis to disambiguate it according to the RRE, while the former evaluates the tagged word to a single output word. So the non-determinism consists in the different ways an RRE can parse an input word. In our work, we tackle a different problem: input words are always parsed by our RREs without ambiguity. However, the evaluation of an input word is done non-deterministically and then yields a possibly infinite set of output words.

2 Models

2.1 Preliminaries

Let Σ be a finite alphabet, the empty word is denoted ε, and the set of words on Σ is denoted Σ^*. The length of a word $w \in \Sigma^*$ is denoted $|w|$. Given a non-empty word $w \in \Sigma^*$, its positions are numbered using integers $i \in \{0, \dots, |w|-1\}$ and $w[i]$ is the letter at position i.

Given two languages $L_1, L_2 \subseteq \Sigma^*$, we say that L_1, L_2 are *unambiguously concatenable* if any word $u \in L_1L_2$ uniquely decomposes into vw with $v \in L_1$ and $w \in L_2$. The language $L \subseteq \Sigma^*$ is *unambiguously iterable*[1] if any word $u \in L^*$ uniquely decomposes into $u_1 \dots u_n$, for some $n \geq 0$, with each $u_i \in L$.

We consider the set $\mathcal{U}(\Sigma)$ of non-null regular expressions over Σ. We represent them using the following grammar : $\mathcal{U}(\Sigma) \ni \alpha : \mathbf{1} \mid a \in \Sigma \mid \alpha_1\alpha_2 \mid [\alpha_1 + \alpha_2] \mid \langle \alpha_1 \rangle$ where $\alpha_1, \alpha_2 \in \mathcal{U}(\Sigma)$. The term $\langle \alpha_1 \rangle$ stands for α_1^*. This grammar has the advantage to make easier the evaluation of the expression during a left-to-right parsing, since the next operator is fully determined by the type of the encountered "parenthesis", namely [or $\langle$. Given $\alpha \in \mathcal{U}(\Sigma)$, we denote by $L(\alpha) \subseteq \Sigma^*$ its associated language. It is well known that regular expressions allow to describe the class of regular languages over Σ, denoted Reg_Σ.

A classical parameter often considered when dealing with expressions is the nesting level of parenthesis. It is defined recursively as follows: if $b \in \Sigma$ and $\alpha_1, \alpha_2 \in \mathcal{U}(\Sigma)$, then $nl(\mathbf{1}) = nl(b) = 0$, $nl(\alpha_1\alpha_2) = \max(nl(\alpha_1), nl(\alpha_2))$, $nl([\alpha_1 + \alpha_2]) = 1 + \max(nl(\alpha_1), nl(\alpha_2))$ and $nl(\langle \alpha_1 \rangle) = 1 + nl(\alpha_1)$. Note that we do not take into account the concatenation.

[1] Also called a code in the literature.

Given two finite alphabets A, B, a *transduction* from A^* to B^* is a relation between A^* and B^* (*i.e.* a subset of $A^* \times B^*$). It can also be seen as a partial map from A^* to $\mathcal{P}(B^*)$.

2.2 Regular Relation Expressions

Given two finite alphabets A and B, a *regular relation expression* f denotes a partial function $[\![f]\!]$ from A^* to Reg_B, whose domain is written $\mathrm{dom}(f)$.

▶ **Definition 2** (Regular Relation Expressions (RRE for short)). *Given two finite alphabets A and B, the class of* regular relation expressions *is the smallest class of functions from A^* to Reg_B that satisfies the following properties:*

- *it contains the* constant functions L/v *where $L \subseteq Reg_A \setminus \{\emptyset\}$ and $v \in B^*$. Its domain is $\mathrm{dom}(L/v) = L$ and for all $u \in L$, $[\![L/v]\!](u) = \{v\}$.*
- *if f, g are RREs, then the* sum $f \oplus g$ *is an RRE such that* $\mathrm{dom}(f \oplus g) = \mathrm{dom}(f) \cup \mathrm{dom}(g)$ *and for all $u \in \mathrm{dom}(f \oplus g)$, $[\![f \oplus g]\!](u) = \bigcup_{h \in \{f,g\} \mid u \in \mathrm{dom}(h)} [\![h]\!](u)$.*
- *if f, g are RREs, then the* Hadamard product $f \otimes g$ *is an RRE such that* $\mathrm{dom}(f \otimes g) = \mathrm{dom}(f) \cap \mathrm{dom}(g)$ *and for all $u \in \mathrm{dom}(f \otimes g)$, $[\![f \otimes g]\!](u) = [\![f]\!](u) \cdot [\![g]\!](u)$.*
- *if f is an RRE, then the* Hadamard star $f^{\otimes}$ *is an RRE such that* $\mathrm{dom}(f^{\otimes}) = \mathrm{dom}(f)$ *and for all $u \in \mathrm{dom}(f^{\otimes})$, $[\![f^{\otimes}]\!](u) = [\![f(u)]\!]^*$.*
- *if f, g are RREs such that* $\mathrm{dom}(f)$ *and* $\mathrm{dom}(g)$ *are unambiguously concatenable, then the* Cauchy product $f \bullet g$ *is an RRE such that* $\mathrm{dom}(f \bullet g) = \mathrm{dom}(f)\mathrm{dom}(g)$, *and for all $u = u_1 u_2$ with $u_1 \in \mathrm{dom}(f)$ and $u_2 \in \mathrm{dom}(g)$: $[\![f \bullet g]\!](u) = [\![f]\!](u_1) \cdot [\![g]\!](u_2)$.*
- *if f is an RRE and if $L \subseteq Reg_A \setminus \{\emptyset\}$ is unambiguously iterable and such that $L^k \subseteq \mathrm{dom}(f)$, then the k-*chained star $f^{\circledast,L,k}$, *and the* left k-chained star $f^{\overleftarrow{\circledast},L,k}$, *are RREs such that* $\mathrm{dom}(f^{\circledast,L,k}) = \mathrm{dom}(f^{\overleftarrow{\circledast},L,k}) = L^{\geqslant k}$, *and for all $u = u_1 u_2 \dots u_n$ with $u_i \in L$ for all i:*

$$[\![f^{\circledast,L,k}]\!](u) = [\![f]\!](u_1 \dots u_k) \cdot [\![f]\!](u_2 \dots u_{k+1}) \cdots [\![f]\!](u_{n-k+1} \dots u_n)$$

$$[\![f^{\overleftarrow{\circledast},L,k}]\!](u) = [\![f]\!](u_{n-k+1} \dots u_n) \cdots [\![f]\!](u_2 \dots u_{k+1}) \cdot [\![f]\!](u_1 \dots u_k)$$

▶ Remark 3. Actually, the 2-chained star and its left version suffice to define RREs. Indeed, other k-chained stars can be defined from them. For instance, the 3-chained star $f^{\circledast,L,3}$ is equivalent to $g \stackrel{\text{def}}{=} ((f \bullet L/\varepsilon) \otimes (L/\varepsilon \bullet f))^{\circledast,L^2,2}$ on the domain $(L^2)^{\geqslant 2}$ of g. It follows that $f^{\circledast,L,3}$ can be expressed as $f \oplus g \oplus ((g \bullet L/\varepsilon) \otimes (L^*/\varepsilon \bullet f))$. However, 3-chained star naturally appears in our proofs when constructing RREs from non-deterministic transducers.

Regular function expressions (RFEs) of [3, 5, 14] can be seen as a restriction of the class of regular relation expressions in which the Hadamard star is forbidden and the sum $f \oplus g$ is authorized only if f and g have disjoint domains. Other operators are introduced, but they are redundant. They can be derived from those presented here (see [5] and [14]). For instance, the Kleene star of a function f, noted here $f^{\circledast}$, simply corresponds to $f^{\circledast,\mathrm{dom}(f),1}$.

The addition to the original model of an ambiguous version of the sum together with Hadamard star, two natural operators, constitutes the starting point of our work. They help to design a new interesting class of transductions, some examples are presented below.

▶ **Example 4.** Come back to the first two transductions presented in Section 1. The Subsequence relation can be expressed as $f_{Sub} = \varepsilon/\varepsilon \oplus (a/\varepsilon \oplus a/a \oplus b/\varepsilon \oplus b/b)^{\circledast}$, and the Iterative-Star relation as $f_{IS} = b/b \oplus \left(\left(b/b \bullet ((a/a)^{\circledast})^{\otimes} \right)^{\circledast} \bullet b/b \right)$.

On the other hand, the Suffix relation f_{Suf} that associates to a word u all the suffixes of u cannot be specified by an RRE. Intuitively, this would require to ambiguously split the word u into $u_1 u_2$ and output the suffix only. This cannot be done with unambiguous Cauchy product or chained star.

▶ **Example 5** (Evaluation of regular expressions). Let $U_k \subseteq \mathcal{U}(\Sigma)$ be the set of expressions with a nesting level at most k. This set can be seen as a regular set of words over $\Sigma \cup \{\mathbf{1}, [, +,], \langle, \rangle\}$. Interestingly, we can define an RRE $f_{eval,k}$ that associates to each expression $\alpha \in U_k$ the language denoted by α. It is inductively built as follows:

- $k = 0$: let $Id = (\mathbf{1}/\varepsilon) \oplus \oplus_{b\in\Sigma}(b/b)$ be the function that evaluates a letter of Σ. Clearly, the base case is the RRE $f_{eval,0} = (Id)^{\circledast}$ that evaluates expressions with nesting level 0.
- The RRE $f_{eval,k} = (Id \oplus f_{eval,+,k} \oplus f_{eval,*,k})^{\circledast}$ decomposes an expression into sub-expressions that are $\mathbf{1}$, a letter of Σ, a union expression $[\cdot + \cdot]$ or a Kleene expression $\langle\cdot\rangle$. It evaluates each sub-expression using Id, $f_{eval,+,k}$ or $f_{eval,*,k}$ according to its type:
  - $f_{eval,+,k} = ([/\varepsilon) \bullet ((f_{eval,k-1} \bullet (+/\varepsilon) \bullet U_{k-1}/\varepsilon) \oplus (U_{k-1}/\varepsilon \bullet (+/\varepsilon) \bullet f_{eval,k-1})) \bullet (]/\varepsilon)$ simply inductively evaluates the left operand or the right operand of a union expression, and makes the union of the results.
  - $f_{eval,*,k} = (\langle/\varepsilon) \bullet (f^{\otimes}_{eval,k-1} \oplus U_{k-1}/\varepsilon) \bullet (\rangle/\varepsilon)$ inductively evaluates the operand of a Kleene expression and iterates the result.

2.3 Streaming String Transducers with Regular Updates

▶ **Definition 6** (Streaming String Transducers with Regular Updates (RSSTs for short)). *Given two finite alphabets A and B, a* streaming string transducer with regular updates *$\mathcal{S}$ over (A, B) is a tuple $\mathcal{S} = (Q, \mathfrak{i}, F, \delta, \mathcal{X}, \mu, \nu)$ where $\mathcal{A} = (Q, \mathfrak{i}, F, \delta)$ is a deterministic finite state automaton over A, i.e. Q is a finite set of states, $\mathfrak{i} \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and δ mapping from $Q \times A$ to Q. This automaton is equipped with a finite set of registers $\mathcal{X}$, an update function $\mu : Q \times A \times \mathcal{X} \to \mathcal{U}(B \cup \mathcal{X})$ and an output function $\nu : F \to \mathcal{U}(B \cup \mathcal{X})$, where $\mathcal{U}(B \cup \mathcal{X})$ denotes the set of regular expressions as specified in preliminaries.*

Intuitively, along an execution of an RSST, registers $X \in \mathcal{X}$ contain a word of $\mathcal{U}(B)$. Each transition step of $\mathcal{A}$ triggers register updates that depend on the current state and input letter. When $\mathcal{A}$ reaches a final configuration with final state $q \in F$, the RSST $\mathcal{S}$ outputs the regular expression obtained by substituting in $\nu(q)$ the registers with their values.

Formally, a valuation of the registers is a function $\chi : \mathcal{X} \to \mathcal{U}(B)$. We extend this notion to regular expressions α of $\mathcal{U}(\mathcal{X} \cup B)$ writing $\chi(\alpha)$ to denote the regular expression α in which each register X is replaced with $\chi(X)$. A configuration of $\mathcal{S}$ is a triple (q, χ, i) where $q \in Q$, χ is a valuation that describes the current value of registers and i is the position of the reading head on the input word. The initial configuration is $(\mathfrak{i}, \chi_0, 0)$, where χ_0 maps every register to $\mathbf{1}$. Two configurations (q, χ, i) and (q', χ', i') are consecutive on $u \in A^*$ if $\delta(q, u[i]) = q'$, $i' = i + 1$ and, for every $X \in \mathcal{X}$, $\chi'(X) = \chi(\mu(q, u[i], X))$. A run on u is a sequence of consecutive configurations on u. It is accepting if it starts from the initial configuration and ends in a final configuration $(q, \chi, |u|)$ with $q \in F$. In this case, the RSST $\mathcal{S}$ outputs the regular expression $\chi(\nu(q))$. Since $\mathcal{S}$ is deterministic, there is at most one accepting run on u for all $u \in A^*$. Thus, $\mathcal{S}$ describes a transduction $[\![\mathcal{S}]\!]$ from A^* to $\mathcal{U}(B)$. Since an RSST outputs a regular expression, we can also define the *evaluated semantics* of $\mathcal{S}$ as the word-to-word relation $[\![\mathcal{S}]\!]_{eval}$ over (A, B) that maps any word $u \in A^*$ to the regular language $L([\![\mathcal{S}]\!](u)) \subseteq B^*$.

Streaming string transducers (SST) [1] are simply RSSTs whose updates are restricted to words in $(B \cup \mathcal{X})^*$ (*i.e.* union and Kleene operators are forbidden).

Copyless SSTs is a classical restriction well-studied in the literature. The copyless property states that, for all states $q \in Q$ and letter $a \in A$, a register X can appear at most once in all the regular expressions in $\{\mu(q, a, X) \mid X \in \mathcal{X}\}$ and at most once in the regular expression $\nu(q)$. In this paper, we consider copyless RSSTs only.

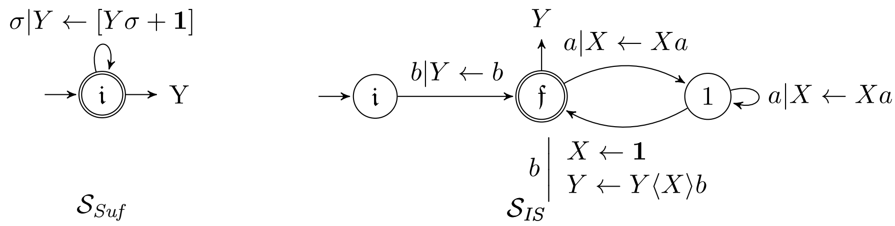


Figure 1 Two examples of RSSTs. The one on the right is nl-bounded.

▶ **Example 7.** Figure 1 depicts two copyless RSSTs. The RSST $\mathcal{S}_{Suf}$ recognizes the Suffix relation and $\mathcal{S}_{IS}$ recognizes the Iterative-Star relation of Example 4. We recall that the Suffix relation cannot be specified by an RRE.

If we look more closely at $\mathcal{S}_{Suf}$, we can see it outputs regular expressions whose nesting level (as defined in Subsection 3.1) depends on the size of the input. On the other hand, if we identify the image of an input word u under an RRE f as a regular expression $\alpha \in \mathcal{U}(B)$ (this is quite simple), one can check that the nesting level of α is bounded by the number of operators used in f. This observation leads us to consider the restriction below that we will prove to be equivalent to RREs.

▶ **Definition 8.** *An RSST $\mathcal{S}$ is* nl-bounded *by n if all the regular expressions in the image of $[\![\mathcal{S}]\!]$ have nesting level at most n. We say that $\mathcal{S}$ is* nl-bounded *if it is for some n.*

For instance, the RSST $\mathcal{S}_{IS}$ of Figure 1 is nl-bounded (by 1). In contrast, $\mathcal{S}_{Suf}$ is not.

By analysing the updates of registers along simple cycles, one can prove:

▶ **Proposition 9.** *Given an RSST $\mathcal{S}$, one can decide whether $\mathcal{S}$ is nl-bounded in* PTIME.

2.4 Weakly Ambiguous Two-Way Finite State Transducers

When studying two-way automata and transducers, it is classical to use additional symbols $\vdash$ and $\dashv$ to surround the input word, thus allowing the two-way device to identify the beginning and the end of the input. Given a finite alphabet A, we let $A_{\vdash\dashv} = A \cup \{\vdash, \dashv\}$.

▶ **Definition 10** (Two-way finite state automata (2NFA for short))**.** *Given a finite alphabet A, a* two-way (non-deterministic) finite state automaton *over A is a tuple $\mathcal{A} = (Q_\rightarrow, Q_\leftarrow, \mathfrak{i}, \mathfrak{f}, \delta)$ where $Q = Q_\rightarrow \uplus Q_\leftarrow$ is a finite set of states, $\mathfrak{i} \in Q_\rightarrow$ is the initial state, $\mathfrak{f} \in Q_\rightarrow$ is the final state. The transition relation δ is included in the union of the following relations:*

- $(\{\mathfrak{i}\} \cup Q_\leftarrow) \times \{\vdash\} \times Q_\rightarrow$*;*
- $Q \setminus \{\mathfrak{i}, \mathfrak{f}\} \times A \times Q \setminus \{\mathfrak{i}, \mathfrak{f}\}$*;*
- $Q_\rightarrow \setminus \{\mathfrak{i}, \mathfrak{f}\} \times \{\dashv\} \times (Q_\leftarrow \cup \{\mathfrak{f}\})$.

The automaton is deterministic *if δ is a partial function from $Q \times A_{\vdash\dashv}$ to Q.*

We describe the behaviors of a 2NFA $\mathcal{A}$ on some input word u in $A^*_{\vdash\dashv}$. A *configuration* of $\mathcal{A}$ is a pair $(q, i) \in Q \times \mathbb{N}$, where i is the position of the reading head. The reading head always points between symbols of u, and possibly on the left of the first one and on the right of the last one. The type of states, $Q_\rightarrow$ or $Q_\leftarrow$, indicates whether the next input letter read is on the right or on the left of the reading head. Two configurations (q, i) and (q', i') are *consecutive* on u if $0 \leqslant i + m_q < |u|$, $(q, u[i + m_q], q') \in \delta$ and $i' = i + m_q + m_{q'} + 1$, where m_q (respectively $m_{q'}$) equals 0 or -1 depending on whether q (respectively q') belongs to $Q_\rightarrow$ or $Q_\leftarrow$. Thus, the reading head moves right (respectively left) when a transition with two states in $Q_\rightarrow$ (respectively in $Q_\leftarrow$) is fired. Otherwise, it does not move.

A *run* r *on* $u@i,j$ from p to q is any finite sequence $(q_0,i_0)\cdots(q_n,i_n)$ of consecutive configurations on u that starts at configuration (p,i) and ends at configuration (q,j). As usual in two-way automata, one can define when two runs r_1 and r_2 can be concatenated, in which case we write this concatenation as $r_1 :: r_2$ (see Appendix C.1 for a formal definition). This notation is extended to sets of runs in the expected way. At many places, we distinguish runs according to the way in which they go through a word:

- r has type LL if $q_0 \in Q_{\rightarrow}$, $q_n \in Q_{\leftarrow}$, $i_0 = i_n$ and $i_0 < i_j$ for all $0 < j < n$;
- r has type RR if $q_0 \in Q_{\leftarrow}$, $q_n \in Q_{\rightarrow}$, $i_0 = i_n$ and $i_j < i_0$ for all $0 < j < n$;
- r has type LR if $q_0, q_n \in Q_{\rightarrow}$, $i_0 < i_n$ and $i_0 < i_j < i_n$ for all $0 < j < n$;
- r has type RL if $q_0, q_n \in Q_{\leftarrow}$, $i_n < i_0$ and $i_n < i_j < i_0$ for all $0 < j < n$;
- r is a *return* run if it is LL or RR, and a *transversal* run if it is LR or RL;
- r is *proper* if it is a return or transversal run and $i_0, i_n \in \{0, |u|\}$.

In particular, a proper LL-run (respectively RR-run) starts and ends at position 0 (respectively position $|u|$). Note that no end marker can be read along a return run, and a traversal run can read them at most once. So all traversal proper runs are on words in $\{\vdash, \varepsilon\} \cdot A^* \cdot \{\varepsilon, \dashv\}$.

A run is *accepting* if the first configuration is $(\mathfrak{i}, 0)$ and the last one is $(\mathfrak{f}, |u|)$. Accepting runs are only possible for words with end markers, namely of the form $\vdash u \dashv$ with $u \in A^*$. They are always proper and of type LR. Note that the final configuration does not allow additional transitions. The word language $L(\mathcal{A})$ of a 2NFA $\mathcal{A}$ consists of the set of words $u \in A^*$ such that there exists an accepting run on $\vdash u \dashv$.

We recall the standard notion of *transition monoid* of $\mathcal{A}$, denoted $M_{\mathcal{A}}$, which is included in $\mathcal{P}(Q^2)$, and such that the mapping φ from A^* to $M_{\mathcal{A}}$ that associates to a word $u \in A^*$ the set of pairs (p,q) such that there is a proper run from p to q on u, is a morphism of monoids.

We say that $\mathcal{A}$ has a *finite degree of ambiguity* if there exists some integer k such that for any word $u \in A^*$, there are at most k accepting runs of $\mathcal{A}$ on $\vdash u \dashv$. Otherwise, we say that $\mathcal{A}$ is *infinitely ambiguous*.

We define the projection $pos : (Q \times \mathbb{N})^* \to \mathbb{N}^*$ that erases the states of a run to keep only the sequence of positions of the reading head. In addition, we also define for every state k the projection $\pi_k : (Q \times \mathbb{N})^* \to (\{k\} \times \mathbb{N})^*$ that erases from a run the configurations that are not in $\{k\} \times \mathbb{N}$, and we set $pos_k = pos \circ \pi_k$. Then, for a run r, $pos_k(r)$ represents the sequence of reading head positions at which the state k occurs along r.

▶ **Definition 11.** *Let $\mathcal{A}$ be a* 2NFA*, k be a state of $\mathcal{A}$ and $i \in \mathbb{N}$.*

A set R of runs synchronizes on (k,i) if (k,i) appears in all $r \in R$.

A set R of runs is k-synchronized if $\{pos_k(r) \mid r \in R\}$ is a singleton.

A set R of runs is k-stationary if $\{pos_k(r) \mid r \in R\} \subseteq \{j\}^+$ for some $j \in \mathbb{N}$.

From now on, we consider a total order $\prec$ on the states of Q, and identify Q with $\{1, \ldots, |Q|\}$. We define the *rank* of a run cr, with c a configuration, as the greatest state occurring in r. Note that the first configuration is not considered. Let $e = (p,k,q) \in Q^3$. We denote $R(e,L)$ as the set of *proper* runs on $u \in L$ from p to q of rank k. For readability, we simply write $R(e,u)$ when $L = \{u\}$.

We finally have all the necessary tools to define *weakly ambiguous* automata: intuitively such an automaton may be infinitely ambiguous, but different runs on a same input word should have similar behaviour w.r.t. a state of highest rank. This structural condition emerges naturally when looking at 2NFT built from expressions. For instance when defining a 2NFT for the union of two 2NFTs, one introduces a new state over which all runs synchronize: they start in the new state then non-deterministically jump to one of the two 2NFTs. Such a "hierarchical" definition is a useful approach to build weakly ambiguous 2NFT, as done in the proof of Proposition 18.

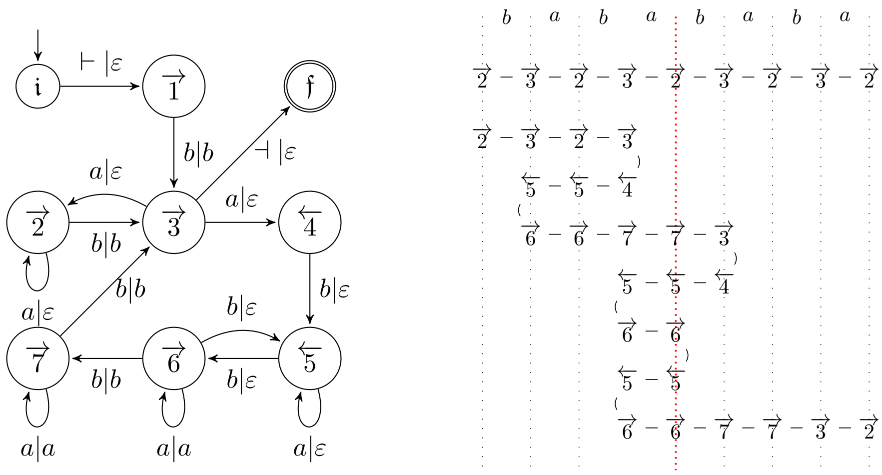


Figure 2 A weakly ambiguous 2NFT $\mathcal{T}_{IS}$ that recognizes f_{IS}, and two of its LR proper runs on $(ba)^4$ from state 2 to state 2. We can see they are 3-synchronized.

▶ **Definition 12.** *A* 2NFA $\mathcal{A}$ *is* weakly ambiguous *with respect to* $\prec$ *if for all words* $u \in A^*$ *and all* $(p, k, q) \in Q^3$*, the set* $R((p, k, q), u)$ *is either empty,* k*-synchronized or* k*-stationary.*

Note that deterministic 2NFAs are trivially weakly ambiguous w.r.t. any order since, in this case, the sets $R((p, k, q), u)$ contain at most one run. One can decide weak ambiguity:

▶ **Proposition 13.** *One can decide whether a* 2NFA *is weakly ambiguous with respect to a given order over its states, in* ExpTime.

▶ **Definition 14** (Two-Way Finite State Transducers (2NFTs for short))**.** *Given two finite alphabets* A *and* B*, a* two-way finite state transducer *from* A *to* B *is a pair* $\mathcal{T} = (\mathcal{A}, out)$*, where* $\mathcal{A} = (Q_{\rightarrow}, Q_{\leftarrow}, \mathfrak{i}, \mathfrak{f}, \delta)$ *is a* 2NFA *over* A*, and* $out : \delta \rightarrow B^*$ *is an output function that maps transitions of* $\mathcal{A}$ *to words over* B*.*

Intuitively, $\mathcal{T}$ extends $\mathcal{A}$ with a one-way left-to-right output tape containing elements of B^*. When a transition $t \in \delta$ is fired, the word $out(t)$ is appended to the right of the output tape. The word written on the output tape at the end of a run r is denoted $output(r)$.

A 2NFT $\mathcal{T}$ thus defines a transduction $[\![\mathcal{T}]\!]$ from[2] A^* to Reg_B. Its domain is $\text{dom}(\mathcal{T}) = L(\mathcal{A})$. For all $u \in \text{dom}(\mathcal{T})$, $v \in [\![\mathcal{T}]\!](u)$ if v is the output of an accepting run on $\vdash u \dashv$.

A 2NFT is deterministic (2DFT) or weakly ambiguous (W2NFT) if its underlying automaton is. So the class of 2DFTs is strictly included in the class of W2NFTs. In particular, any regular function (*i.e.* recognized by a 2DFT) is recognized by a W2NFT.

Lastly, the *transition monoid* of $\mathcal{T}$, denoted as $M_{\mathcal{T}}$, is defined as the one of $\mathcal{A}$.

▶ **Example 15.** Figure 2 depicts a weakly ambiguous 2NFT $\mathcal{T}_{IS}$ with order $\mathfrak{i} \prec \mathfrak{f} \prec 1 \prec 2 \prec 4 \prec 5 \prec 6 \prec 7 \prec 3$. The arrows in the states, represented by circles, indicate the reading direction. It has domain $(ba^+)^*b$ and recognizes the transduction f_{IS} of Example 4. Two LR proper runs on $(ba)^4$ from state 2 to itself are depicted in Figure 2. In the second run, we can see that state 5 occurs multiple times at the same position. The piece of run between these two occurrences can be repeated any number of time, giving rise to new runs: $\mathcal{T}_{IS}$ does not have a finite degree of ambiguity. All these runs have rank 3 and are 3-synchronized.

---

[2] It is easy to verify that for every $u \in A^*$, $[\![\mathcal{T}]\!](u)$ is a regular language on B.

3 Main result

3.1 Preliminary properties

The following properties give a clue as to why the class of transductions we study behaves nicely, namely the good closure properties it enjoys.

The following proposition is proved using decomposition theorems: according to [15], any rational function is the composition of a sequential and a co-sequential function. Moreover, using the result of Krohn-Rhodes [19], sequential functions can be further decomposed.

▶ **Proposition 16.** *RSSTs are closed by RRE operations. Moreover this preserves nl-boundedness.*

The next proposition is shown using a result of [11] stating that regular functions can be realized by *reversible* transducers, and that pre-composition with reversible transducers is well-behaved.

▶ **Proposition 17.** W2NFTs *are closed by pre-composition with a regular function.*

Finally the evaluation relation which inputs a regular expression (of bounded nesting level) can be realized by a weakly ambiguous transducer.

▶ **Proposition 18.** *For all n, the transduction $f_{eval,n}$ which evaluates a regular expression can be recognized by a* W2NFT $\mathcal{T}_{eval,n}$.

Sketch of proof. We can build two-way transducers for $f_{eval,n}$ by induction over n. Base cases are easy, and the inductive step uses a modular construction which naturally entails that the resulting transducers are weakly ambiguous.

Alternatively, we could use the fact that weakly ambiguous transducers are closed under pre-composition by regular functions to show that they are closed under RRE operations (as is done for RSSTs), and thus subsume RREs. ◀

3.2 Main theorem

Now that we have formally defined the models we study, we can (re)state our main result:

▶ **Theorem 1.** *Let f be a word-to-word transduction. The following are equivalent:*
- *f is denoted by a regular relation expression.*
- *f is recognized by a weakly ambiguous two-way transducer.*
- *f is recognized by a nl-bounded streaming string transducer with regular updates.*

Sketch of proof.
- From RRE to RSST: Using Proposition 16, we only have to notice that constant functions can be realized by nl-bounded RSSTs.
- From RSST to 2NFT: By definition, the semantics of an RSST $\mathcal{S}$ with nesting level n can be expressed as the composition $[\![\mathcal{S}]\!]_{eval} = [\![\mathcal{T}_{\text{eval},n}]\!] \circ [\![\mathcal{S}]\!]$. One can thus see an nl-bounded RSST as a regular function. Using Proposition 17 we know that W2NFTs are closed under pre-composition by regular functions. We can conclude since the evaluation relation can be realized by a W2NFT (Proposition 18).
- From W2NFT to RRE. This last inclusion is proved in the next Section.

◀

▶ Remark 19. Word-to-word regular functions are also characterized as word-to-word MSO transductions [16], in the sense of Courcelle [9]. As a consequence, our class of transductions is equivalent to that of MSO transductions from words to regular expressions, which have a *bounded nested-level*, *i.e.* such that there exists a bound on the nesting level of all the regular expressions they may output. Indeed, the reasoning of the previous proof to go from RSST to 2NFT is also valid for any MSO transduction of bounded nested-level.

4 From Two-Way Transducers to Expressions

Our construction is based on the one of [14] for deterministic 2NFT. It strongly relies on the following unambiguous version of Simon's factorization forest theorem [22]:

▶ **Theorem 20** ([14]). *Let M be a finite monoid and φ be a monoid morphism from A^* to M. For each $m \in M$, there is an ε-free φ-good regular expression E_m such that $L(E_m) = \varphi^{-1}(m) \setminus \{\varepsilon\} \subseteq A^+$.*

In this statement, an ε-free regular expression cannot use ε nor Kleene star, but can use Kleene plus. Goodness means that the expression E_m is unambiguous and that the image $\varphi(L(E))$ of any sub-expression E of E_m is a singleton $\{m_E\}$. As a consequence, Kleene plus connectors only occur on sub-expressions whose image by φ is an *idempotent* element.

The approach of [14] uses the transition monoid $M_{\mathcal{T}}$ and properties of its idempotents. A recap is given in Appendix B. Roughly, the determinism of the transducer entails strong properties on idempotents elements of $M_{\mathcal{T}}$, such as nice decompositions of runs. In this paper, we start from a weakly ambiguous 2NFT $\mathcal{T}$. Because it is non-deterministic, the study of the shape of its runs is more difficult.

4.1 Analysis of the shape of runs

Preliminaries We slightly modify the classical definition of transition monoid to keep track of run ranks. We denote by $M^{\mathsf{r}}_{\mathcal{T}} \subseteq \mathcal{P}(Q^3)$ this new monoid. To each input word $u \in A^*_{\vdash\dashv}$ we associate the set $m = \mu(u) \in M^{\mathsf{r}}_{\mathcal{T}}$ defined by $(p,k,q) \in \mu(u)$ if there is a proper run in $\mathcal{T}$ on u of rank k from p to q. One can verify that $M^{\mathsf{r}}_{\mathcal{T}}$ is a monoid, and that μ is a monoid morphism. For $e = (p,k,q) \in m$, we denote by $R(e,m)$ the set of all proper runs of rank k from p to q on words $u \in \mu^{-1}(m)$, and we have that $R(e,m) = \bigcup_{\mu(u)=m} R(e,u)$. Observe that given an element $e \in m$, all the proper runs in $R(e,m)$ have the same type and the same rank. We define this way the type and the rank of e.

Given an element $m \in M^{\mathsf{r}}_{\mathcal{T}}$, we define the labelled graph $\mathcal{G}_m$ by interpreting elements of m as edges: $(p,k,q) \in m$ yields an edge from p to q labelled by k. An example is given on Figure 3, which corresponds to the W2NFT of Example 15.

Let L_1 and L_2 be two unambiguously concatenable languages and $u = vw \in L_1L_2$, with v in L_1 and w in L_2. We say that a run r on u is L_1, L_2-*quasi-proper* if it starts and ends at positions 0, $|v|$ or $|u|$. Such a run can *uniquely* be decomposed into a sequence $\Delta_{L_1,L_2}(r) = (t_1, \ldots, t_n)$ where the t_i's are proper sub-runs alternatively on v or w such that $r = t_1 :: \cdots :: t_n$. This notion can easily be adapted to the unambiguous Kleene iteration of a language L: given a quasi-proper run r on $u \in L^+$, there exists a unique L-decomposition $\Delta_L(r) = (t_0, t_1, \ldots, t_l)$ of proper sub-runs over L such that $t_0 :: t_1 \cdots :: t_l = r$.

▶ Remark 21. There is a bijection between L-quasi-proper runs r on some word in L^+, and paths ρ of $\mathcal{G}_m$ from p to q. It follows from the (unique) decomposition $r = t_0 :: \cdots :: t_l$, where $\Delta_L(r) = (t_0, \ldots, t_l)$, which corresponds to the path $(p_0,k_0,q_0) \ldots (p_l,k_l,q_l)$ in $\mathcal{G}_m$, with $t_i = (p_i,k_i,q_i)$ for every i. In particular, observe that the rank of r is $\max\{k_i \mid 0 \leqslant i \leqslant l\}$.

Analysis of runs in L^+ From now on, we suppose that L is an unambiguously iterable language whose image m by μ is an idempotent element of M. We first state an easy property:

▶ **Lemma 22.** *The L-decomposition of a quasi-proper run r on $u \in L^*$ cannot contain both an LR-run and an RL-run.*

In general, we can tell nothing about the rank of the runs in the decomposition of r. But interesting properties can be exhibited when the starting and ending states of r are in the

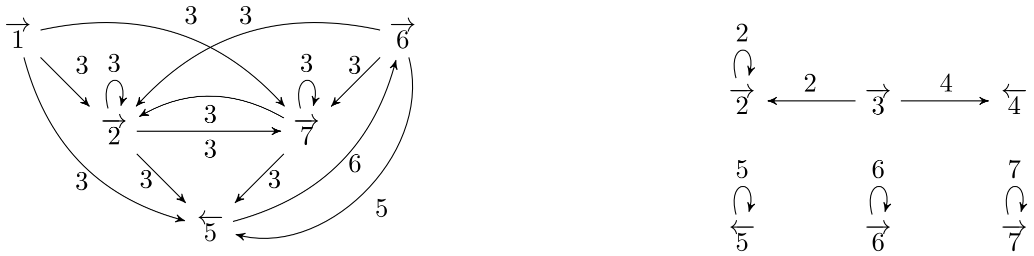


Figure 3 On the left, the graph $\mathcal{G}_{baba}$ associated to the idempotent element m_{baba}. On the right, the graph $\mathcal{G}_a$ associated to the idempotent element m_a.

same (non-trivial) strongly connected component (SCC) of $\mathcal{G}_m$. This relies on the particular structure of the SCCs of $\mathcal{G}_m$, characterized by Proposition 23. We write $p \sim_m q$ if p and q are states of the same *non-trivial* SCC C of $\mathcal{G}_m$. We also use letter C to denote a non-trivial SCC of $\mathcal{G}_m$. The rank of C, denoted k_C, is the maximum of the ranks of its edges.

▶ **Proposition 23.**
1. *All transversal edges of an SCC C of $\mathcal{G}_m$ have the same type, and the same rank as C.*
2. *Let C_1 and C_2 be two non-trivial SCCs of $\mathcal{G}_m$ that contain transversal edges of m. If there is a path from a state of C_1 to a state of C_2 in $\mathcal{G}_m$ then $C_1 = C_2$.*

▶ **Example 24.** Following Example 15 (remember that 3 is the greatest state here), one can check that the element $m_{baba} = \mu(baba)$ is idempotent. Its graph is depicted on Figure 3. As expected, all the transversal edges of the strongly connected component $\{2, 5, 6, 7\}$ have the same type and the same rank 3. Those with a different rank are LL or RR edges. If we look at the graph $\mathcal{G}_a$ of the idempotent element $\mu(a)$, we can see four strongly connected components. Each of them has transversal edges of a single type. The rank of transversal edges in different components can be different.

Thanks to Remark 21, Proposition 23.1 can be reformulated in terms of runs.

▶ **Corollary 25.** *Let r be a quasi-proper run on u from p to q with p, q in an SCC C.*
- *All the transversal runs of $\Delta_L(r)$ have the same type and rank k_C;*
- *All the return runs of $\Delta_L(r)$ have rank less than or equal to k_C.*

▶ **Proposition 26.** *Let $w = w_1 \dots w_n \in L^+$. The runs in $\bigcup_{p,q \in C \cap Q_\rightarrow} R((p, k_C, q), w)$ (resp. $\bigcup_{p,q \in C \cap Q_\leftarrow} R((p, k_C, q), w)$) synchronize on k_C. More precisely, they do it at least once between positions $|w_1 \dots w_i| + 1$ and $|w_1 \dots w_{i+1}|$ for all $0 \leq i < n$.*

Sketch of proof. Consider two proper transversal runs r_1, r_2 on some word $w = w_1 \dots w_n$ in some SCC C. Given $p, q \in Q_\rightarrow \cap C$, we can extend them so as to obtain two transversal runs $ext_{p,q}(r_1)$ and $ext_{p,q}(r_2)$ on the word $w_1 w w_n$, which both start in p and end in q. This is possible as the two runs belong to the same SCC. Since $\mathcal{T}$ is weakly unambiguous, the two extended runs are k_C-synchronized, which is possible by construction only if r_1 and r_2 are. The second part of the corollary holds because of Corollary 25. ◀

From Proposition 23.2, it results that we can decompose any long enough transversal proper run into 3 transversal (quasi-)proper sub-runs. The length of the prefix and the suffix sub-runs depends on the number of states of $\mathcal{T}$. The infix sub-runs "live" in an SCC whose rank is smaller than those of the other sub-runs. Thus, Corollary 25 holds for this sub-run.

▶ **Proposition 27.** *If r is a proper transversal run on $u \in L^{\geqslant 2|Q|+3}$ where Q is the number of states of the* 2NFT*, then it can be decomposed into $r_1 :: r_2 :: r_3$ such that*

- *r_1 is a proper transversal run to p on the prefix u_1 of u in $L^{|Q|+1}$;*
- *r_3 is a proper transversal run from q on the suffix u_2 of u in $L^{|Q|+1}$;*
- *the states p and q are $\sim_m$-equivalent.*
- *the ranks of r_1 and r_3 are greater than, or equal to, the rank of r_2.*

4.2 Building expressions from transducers

Let $\mathcal{T}$ be a weakly ambiguous transducer w.r.t. some order $\prec$ on its states. Without loss of generality, we can suppose that the final state $\mathfrak{f}$ of $\mathcal{T}$ is the largest state because it appears at most once in any run (at the last configuration). We aim to build a RRE $f_{\mathcal{T}}$ equivalent to $[\![\mathcal{T}]\!]$. Our construction relies on the following key lemma.

▶ **Lemma 28.** *For any ε-free μ-good regular expression F and $e = (p, k, q) \in \mu(L(F))$, we can compute an RRE $out_{F,e}$ with domain $L(F)$ such that $[\![out_{F,e}]\!](u) = \{output(r) \mid r \in R(e, u)\}$.*

Intuitively, the proof proceeds by induction on F. The main difficulty arises when considering Kleene iteration. In this case, we use Proposition 27 to show that we can build $out_{F,e}$ as a finite sum by distinguishing the SCC and the inner states p, q. The detailed proof is given in Appendix D. We explain how to use it to get $f_{\mathcal{T}}$. We let $P = \{\mu(\vdash u \dashv) \mid u \in \mathrm{dom}(\mathcal{T})\}$. For each $m \in P$, ε does not belong to $\mu^{-1}(m)$, and by Theorem 20, we can find an ε-free μ-good regular expression E_m for $\mu^{-1}(m)$. We let $e_{\mathfrak{f}} = (\mathfrak{i}, \mathfrak{f}, \mathfrak{f})$. We get by Lemma 28:

$$[\![\mathcal{T}]\!](u) = [\![\bigoplus_{m \in P} out_{E_m, e_{\mathfrak{f}}}]\!](\vdash u \dashv) \quad \text{for all } u \in \mathrm{dom}(\mathcal{T}).$$

Using small technicalities to get rid of endmarkers, one can then derive $f_{\mathcal{T}}$ from $\bigoplus_{m \in P} out_{E_m, e_{\mathfrak{f}}}$.

5 Discussion

We have introduced a class of relations which subsumes regular functions, has several distinct characterizations and enjoys multiple closure properties.

We have also investigated other aspects of this class. Firstly, while we have shown that this class is closed under *pre-composition* by regular functions, it is not closed under *post-composition* by regular functions. For instance the relation which maps a word to any square of a subword is not recognizable by a two-way transducer since one cannot make the same guess of which positions to keep twice. We actually think that it is not even closed under post-compostion by sequential functions. Second, for the sake of simplicity, we have not mentionned yet a rather natural restriction of RRE which would correspond to one-way weakly ambiguous transducers. We strongly believe that such an equivalence should hold by removing all operations which are not one-way and having an unambiguous Kleene star operation. Lastly, the equivalence of two weakly ambiguous transducers is unfortunately undecidable, the classical proof being incidentally valid for weakly ambiguous transducers.

Natural extensions of this work would be to allow ambiguity for the Cauchy product or the chained-star operators. Note however that two-way transducers are not closed under these operations, so such a class would go beyond two-way transducers. One possibility to circumvent this problem would be to consider transducers with *common guess*: such a transducer can non-deterministically guess a coloring of its input and thus perform such operations. Finally, we do not know whether weak ambiguity subsumes finite ambiguity. A sufficient condition is that finitely ambiguous transducers coincide in expressiveness with finite unions of unambiguous transducers, but this is an open problem.

References

1 Rajeev Alur and Pavol Černý. Expressiveness of streaming string transducers. In *FSTTCS*, volume 8 of *LIPIcs.*, pages 1–12. Schloss Dagstuhl. Leibniz-Zent. Inform., 2010.

2 Rajeev Alur and Jyotirmoy V. Deshmukh. Nondeterministic streaming string transducers. In Luca Aceto, Monika Henzinger, and Jirí Sgall, editors, *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*, volume 6756 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2011. URL: `https://doi.org/10.1007/978-3-642-22012-8_1`, `doi:10.1007/978-3-642-22012-8\_1`.

3 Rajeev Alur, Adam Freilich, and Mukund Raghothaman. Regular combinators for string transformations. In *CSL-LICS '14*, pages 9:1–9:10. ACM, 2014.

4 Nicolas Baudru, Louis-Marie Dando, Nathan Lhote, Benjamin Monmege, Pierre-Alain Reynier, and Jean-Marc Talbot. Weighted automata and expressions over pre-rational monoids. In Florin Manea and Alex Simpson, editors, *30th EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference)*, volume 216 of *LIPIcs*, pages 6:1–6:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. URL: `https://doi.org/10.4230/LIPIcs.CSL.2022.6`, `doi:10.4230/LIPIcs.CSL.2022.6`.

5 Nicolas Baudru and Pierre-Alain Reynier. From two-way transducers to regular function expressions. *IJFCS*, 31(6):843–873, 2020. `doi:10.1142/S0129054120410087`.

6 Mikolaj Bojanczyk. Transducers with origin information. In *ICALP 2014*, volume 8573 of *LNCS*, pages 26–37. Springer, 2014.

7 Olivier Carton and Luc Dartois. Aperiodic two-way transducers and fo-transductions. In Stephan Kreutzer, editor, *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany*, volume 41 of *LIPIcs*, pages 160–174. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. URL: `https://doi.org/10.4230/LIPIcs.CSL.2015.160`, `doi:10.4230/LIPIcs.CSL.2015.160`.

8 Christian Choffrut and Bruno Guillon. An algebraic characterization of unary two-way transducers. In *MFCS*, volume 8634 of *LNCS*, pages 196–207. Springer, 2014.

9 Bruno Courcelle. Monadic second-order definable graph transductions: a survey. *Theoret. Comput. Sci.*, 126(1):53–75, 1994. `doi:10.1016/0304-3975(94)90268-2`.

10 Louis-Marie Dando and Sylvain Lombardy. Two-way automata over locally finite semirings. In *International Conference on Descriptional Complexity of Formal Systems*, pages 62–74. Springer, 2018.

11 Luc Dartois, Paulin Fournier, Ismaël Jecker, and Nathan Lhote. On reversible transducers. In *ICALP 2017*, volume 80 of *LIPIcs*, pages 113:1–113:12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

12 Luc Dartois, Paul Gastin, R. Govind, and Shankara Narayanan Krishna. Efficient construction of reversible transducers from regular transducer expressions. In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, pages 50:1–50:13. ACM, 2022. URL: `https://doi.org/10.1145/3531130.3533364`, `doi:10.1145/3531130.3533364`.

13 Luc Dartois, Paul Gastin, and Shankara Narayanan Krishna. Sd-regular transducer expressions for aperiodic transformations. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13. IEEE, 2021. URL: `https://doi.org/10.1109/LICS52264.2021.9470738`, `doi:10.1109/LICS52264.2021.9470738`.

14 Vrunda Dave, Paul Gastin, and Shankara Narayanan Krishna. Regular transducer expressions for regular transformations. *Inf. Comput.*, 282:104655, 2022. URL: `https://doi.org/10.1016/j.ic.2020.104655`, `doi:10.1016/j.ic.2020.104655`.

15 Calvin C. Elgot and Jorge E. Mezei. On relations defined by generalized finite automata. *IBM J. Res. Dev.*, 9(1):47–68, 1965. URL: `https://doi.org/10.1147/rd.91.0047`, `doi:10.1147/rd.91.0047`.

16 Joost Engelfriet and Hendrik Jan Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Transactions on Computational Logic (TOCL)*, 2(2):216–254, 2001.

17 Emmanuel Filiot and Pierre-Alain Reynier. Transducers, logic and algebra for functions of finite words. *ACM SIGLOG News*, 3(3):4–19, 2016. URL: `https://doi.org/10.1145/2984450.2984453`, `doi:10.1145/2984450.2984453`.

18 David Janin. On languages of one-dimensional overlapping tiles. In Peter van Emde Boas, Frans C. A. Groen, Giuseppe F. Italiano, Jerzy Nawrocki, and Harald Sack, editors, *SOFSEM 2013: Theory and Practice of Computer Science*, pages 244–256, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

19 Kenneth Krohn and John Rhodes. Algebraic theory of machines. i. prime decomposition theorem for finite semigroups and machines. *Transactions of the American Mathematical Society*, 116:450–464, 1965. URL: `http://www.jstor.org/stable/1994127`.

20 Anca Muscholl and Gabriele Puppis. The many facets of string transducers (invited talk). In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, volume 126 of *LIPIcs*, pages 2:1–2:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL: `https://doi.org/10.4230/LIPIcs.STACS.2019.2`, `doi:10.4230/LIPIcs.STACS.2019.2`.

21 Giovanni Pighizzini. Nondeterministic one-tape off-line turing machines and their time complexity. *J. Autom. Lang. Comb.*, 14(1):107–124, jan 2009.

22 Imre Simon. Factorization forests of finite height. *Theor. Comput. Sci.*, 72(1):65–94, 1990. URL: `https://doi.org/10.1016/0304-3975(90)90047-L`, `doi:10.1016/0304-3975(90)90047-L`.

A Proofs of Subsection 3.1

The following proposition is proved using decomposition theorems: according to [15], any rational function is the composition of a sequential and a co-sequential function. Moreover, using the result of Krohn-Rhodes [19], sequential functions can be further decomposed.

▶ **Proposition 29.** *RSSTs are closed by pre-composition with a letter-to-letter rational function. Moreover this preserves nl-boundedness.*

Proof. In order to prove this we rely on two decomposition results. The first is the result of Elgot and Mezei [15] which states that any rational function is the composition of a sequential and a co-sequential function. The second is the result of Krohn and Rhodes [19] which says that any letter-to-letter sequential function (ie realized by a Mealy machine) is the composition of two kinds of functions:

- functions realized by Mealy machines where each letter induces a permutation of the states.
- functions realized by 2 state Mealy machines where each letter induces either a constant function over the states or the identity function.

Of course the symmetric result holds for co-sequential functions. Thus we only need to show closure under pre-composition by these simpler classes of functions. This is what we do in Lemmas 30, 31 and 32. The fact that nl-boundedness is preserved is clear since it is a semantic restriction. ◀

▶ **Lemma 30.** *RSSTs are closed by pre-composition with letter-to-letter sequential functions.*

Proof. Let A, B, C be three alphabets. Let $\mathcal{S}_f = (Q_f, \mathfrak{i}_f, F_f, \delta_f, \mathcal{X}_f, \mu_f, \nu_f)$ be a RSST over (B, C) and g a letter-to-letter sequential function. Then g is recognized by a mealy machine $\mathcal{A} = (Q_A, \mathfrak{i}_A, F_A, \delta_A, \lambda_A)$ over (A, B). We build a RRST $\mathcal{S} = (Q, \mathfrak{i}, F, \delta, \mathcal{X}, \mu, \nu)$ over (A, C) such that $[\![\mathcal{S}]\!] = [\![\mathcal{S}_f]\!] \circ [\![\mathcal{A}]\!]$. For readability, the states of $\mathcal{S}_f$ are denoted as $p, p_1, \ldots$, the one of $\mathcal{A}$ as $q, q_1, \ldots$ and those of $\mathcal{S}$ as $s, s_1, \ldots$.

The RSST $\mathcal{S}$ results from the product construction between $\mathcal{S}_f$ and $\mathcal{A}$: $Q = Q_f \times Q_A$, $\mathfrak{i} = (\mathfrak{i}_f, \mathfrak{i}_A)$, $F = F_f \times F_A$. On reading an input letter a from a state $s = (p, q)$, the RSST $\mathcal{S}$ first simulates $\mathcal{A}$ on a from q, which produces an output b, and then simulates $\mathcal{S}_f$ on b. Thus, $\mathcal{S}$ uses the same registers as $\mathcal{S}_f$ and $\delta(p, q) = (p', q')$ if $\delta_A(q, a) = q'$, $\lambda(q, a) = b$, $\delta_f(p, b) = p'$ and $\mu((p, q), b) = \mu(p, b)$. Moreover, for all final states $(p, q) \in F$, $\nu(p, q) = \nu(p)$. Clearly, $\mathcal{S}$ is a RSST since the updates are the same as the ones of $\mathcal{S}_f$. A simple induction on the length of runs shows that $[\![\mathcal{S}]\!] = [\![\mathcal{S}_f]\!] \circ [\![\mathcal{A}]\!]$. ◀

▶ **Lemma 31.** *RSSTs are closed by pre-composition with functions that are recognized by the transpose of two-state Mealy machines where every input letter acts as a constant function or the identity function on the states.*

Proof. Let A, B, C be three alphabets. Let $\mathcal{S}_f = (Q_f, \mathfrak{i}_f, F_f, \delta_f, \mathcal{X}_f, \mu_f, \nu_f)$ be a RSST over (B, C) and $\mathcal{A} = (Q_A, I_A, F_A, \delta_A, \lambda_A)$ be the transpose of a Mealy machine over (A, B) as in the lemma. We denote its two states as $\mathfrak{f}$ and $\bar{\mathfrak{f}}$, both are initial and $\mathfrak{f}$ is final. The transition relation is $\delta \subseteq Q_A \times A \times Q_A$ and the output function is $\lambda_A : \delta \to B$. We build a RRST $\mathcal{S} = (Q, \mathfrak{i}, F, \delta, \mathcal{X}, \mu, \nu)$ over (A, C) such that $[\![\mathcal{S}]\!] = [\![\mathcal{S}_f]\!] \circ [\![\mathcal{A}]\!]$. For readability, the states of $\mathcal{S}_f$ are denoted as $p, p_1, \ldots$, the one of $\mathcal{A}$ as $q, q_1, \ldots$ and those of $\mathcal{S}$ as $s, s_1, \ldots$.

We explain the ideas behind the construction. For a given input word $u \in A^*$, $\mathcal{S}$ simulates at the same time all the runs of $\mathcal{A}$ on u, and for each of these runs r, the (unique) run of

$\mathcal{S}_f$ on the output of r. Since $\mathcal{A}$ is the transpose of a two-state Mealy machine where every input letter a acts as the identity function or as a constant function on the states, either $\delta_A(f,a) \neq \delta_A(\bar{f},a)$ or one of these two transitions is undefined (*). Consequently, it cannot have more than two runs on $\mathcal{A}$ on any word u, and at most two runs of $\mathcal{S}_f$ need to be simulated at the same time for each input word $u \in A^*$. Thus, a state of Q consists in one or two pairs of $Q_f \times Q_A$ (depending on whether one or two runs need to be simulated at the same time). The initial state is $\mathfrak{i} = \{(\mathfrak{i}_f, \mathfrak{f}), (\mathfrak{i}_f, \bar{\mathfrak{f}})\}$. The RSSTs $\mathcal{S}$ uses two copies of $\mathcal{X}_f$: $\mathcal{X} = \mathcal{X}_f \times Q_A$. When $\mathcal{S}$ simulates a transition of $\mathcal{S}_f$, it updates its registers by mimicking the corresponding updates, which ensures that the updates of $\mathcal{S}$ are still regular.

We formally define the transition function, the update function and the output function of $\mathcal{S}$. Let $\varrho : Q_A \times \mathcal{U}(\mathcal{X}_f) \to \mathcal{U}(\mathcal{X})$ such that $\varrho(q,\alpha)$ substitutes in α every register $x \in \mathcal{X}_f$ with $(x,q) \in \mathcal{X}$. For all $v \in Q$ and $a \in A$, we define $\delta(v,a)$ (noted v') and $\mu(v,a)$ (noted σ) as follows: if $(p,q) \in v$, $t = (q,a,q') \in \delta_A$, $\lambda(t,a) = b$ and $\delta_f(p,b) = p'$ then $(p',q') \in v'$ and, for all $x \in \mathcal{X}_f$, $\sigma(x,q') = \varrho(q,\alpha)$ where $\alpha = \mu_f(p,b,x)$. Note that σ is well-defined thanks to (*). Finally, $v \in F$ if $(p,\mathfrak{f}) \in v$ for some $p \in F_f$, and we set $\nu(v) = \varrho(\mathfrak{f}, \nu_f(p))$.

Using a simple induction on the length of input word $u \in A^*$, it is easy to show that the next statement holds: For all $p \in Q_f$ and $q \in Q_A$, there are a run from $\mathfrak{i}_A$ to q on u in $\mathcal{A}$ that outputs v and a run from $(\mathfrak{i}_f, \mathcal{X}_f \to \{\varepsilon\})$ to (p,χ) on v in $\mathcal{S}_f$, if and only if, there is a run from $(\mathfrak{i}, \mathcal{X} \to \{\varepsilon\})$ to some (s,χ') on u in $\mathcal{S}$ such that $(p,q) \in s$. Moreover, whenever these runs exist, we have $\chi'(x,q_n) = \chi(x)$ for all $x \in \mathcal{X}_f$. The proof of the lemma follows when considering accepting runs.

◀

▶ **Lemma 32.** *RSSTs are closed by pre-composition with functions that are recognized by the transpose of Mealy machines where every input letter acts as a permutation on the states.*

Proof. We only give the main ideas behind the construction. An induction on the runs suffices to show the built RSST recognize what is expected. The nl-boundedness is quite obvious.

Let $\mathcal{S}_f$ be a RSST with set of states Q_f and initial state $\mathfrak{i}_f$. Let $\mathcal{A}$ be the transpose of a Mealy machines where every input letter acts as a permutation on the states, with set of states $Q_\mathcal{A}$. This machine is deterministic and complete. Its transition function is injective. All its states are initial, and only one is final, noted $\mathfrak{f}$. We build a RRST $\mathcal{S}$ such that $[\![\mathcal{S}]\!] = [\![\mathcal{S}_f]\!] \circ [\![\mathcal{A}]\!]$. The ideas behind the construction are the follows. For a given input word u, $\mathcal{S}$ simulates at the same time all the runs of $\mathcal{A}$ on u, and for each of these runs r, the (unique) run of $\mathcal{S}_f$ on the output of r. Since all the states of the complete and deterministic machine $\mathcal{A}$ are initial, there are precisely $n = |Q_\mathcal{A}|$ runs on $\mathcal{A}$ on any word u, and as many runs of $\mathcal{S}_f$ to be simulated at the same time (by using a product construction for each run). Thus, a state s of $\mathcal{S}$ consist in a sequence $(p_1,q_1),\ldots,(p_n,q_n)$ of n pairs of $Q_f \times Q_\mathcal{A}$. We arbitrarily choose a state of $\mathcal{S}$ with all its first components at $\mathfrak{i}_f$ as the initial state of $\mathcal{S}$. Note that, any reachable state of $\mathcal{S}$ have pairewise distinct q_i's because the transition function of $\mathcal{A}$ is injective. The RSST $\mathcal{S}$ uses n copies of $\mathcal{X}_f$: $\mathcal{X} = \mathcal{X}_f \times \{1,\ldots,n\}$. When $\mathcal{S}$ simulates a transition t of $\mathcal{S}_f$ from the i-th pair of the sequence, it updates the registers in $\mathcal{X}_f \times \{i\}$ by mimicking the updates associated to t. A sequence $s = (p_1,q_1),\ldots,(p_n,q_n)$ is a final state of $\mathcal{S}$ if it contains a pair $(p,\mathfrak{f})$ with p a final state of $\mathcal{S}_f$. Since, all the q_i's are distinct, there is only one such pair, saying at position i in the sequence. Then, $\mathcal{S}$ mimes the output of $\mathcal{S}_f$ from state p using the corresponding registers in $\mathcal{X}_f \times \{i\}$. ◀

▶ **Proposition 16.** *RSSTs are closed by RRE operations. Moreover this preserves nl-boundedness.*

Proof Sketch. Closure under sum or Hadamard product of two relations f, g defined by (nl-bounded) RSSTs is straightforward: only need to add or concatenate the results of the two RSSTs.

For the closure under unambiguous Cauchy product or chain star, we use the result of Proposition 29: the rational function is used to mark the positions of the decomposition according to the unambiguous product/Kleene star. ◀

▶ **Proposition 17.** W2NFTs *are closed by pre-composition with a regular function.*

Proof. We use a result of [11] stating that any regular function can be defined by a reversible two-way transducer. Here reversible means that any configuration of the underlying automaton has at most one successor (deterministic) and one predecessor (co-deterministic).

Without loss of generality, we can assume that a reversible two-way transducer outputs at most one letter per transition. Moreover, in order to simplify the proofs we further decompose a regular function f into $\phi \circ g$ where g is given by a reversible transducer which outputs exactly one symbol per transition, and a morphism ϕ which erases one particular symbol and is the identity over other symbols. This can easily be obtained by modifying a reversible transducer which outputs at most one letter per transition: each transition which should output ε outputs instead a special symbol $\bar{\varepsilon}$. Then the morphism ϕ erases the extra symbols.

We call a transducer *transition-to-letter* if every transition produces exactly one letter, and a morphism which erases one letter and does not modify the others is called a 1-*erasing* morphism. Hence we only have to show the following claim:

▷ Claim 33.
1. W2NFTs are closed by pre-composition with 1-erasing morphims,
2. W2NFTs are closed by pre-composition with transition-to-letter reversible transducers.

Proof of 1. Let us consider a W2NFT $\mathcal{T}$ with underlying automaton $\mathcal{A}$ over alphabet A, realizing a relation T. Let ϕ be 1-erasing morphism erasing the letter $\bar{\varepsilon}$.

We define a new transducer $\mathcal{T}'$ which realizes $T \circ \phi$. Intuitively, this transducer, when reading a letter $\bar{\varepsilon}$ ignores it and continues in the direction it was moving. The set of states of the new transducer is $Q \uplus \bar{Q}$, where $\bar{Q}$ is a copy of Q. The transitions of $\mathcal{T}'$ over letters different from $\bar{\varepsilon}$ are the same as the transitions of $\mathcal{T}$. Given a state $p \in Q$, we add a transition $(p, \bar{\varepsilon}, \bar{p})$ with no outputs (note that the direction of $\bar{p}$ is the same as the direction of p). We also add transitions $(\bar{p}, \bar{\varepsilon}, \bar{p})$ again with no output. Finally, for any transition (p, a, q) of $\mathcal{T}$, we add a transition $(\bar{p}, a, q)$ with the same output as (p, a, q). Hence any factor of consecutive $\bar{\varepsilon}$ symbols is ignored by the transducer, which just moves trough it to the next regular letter, propagating the state information.

We define the order over $Q \uplus \bar{Q}$ by saying that original states (in Q) are greater than any copy state (in $\bar{Q}$) and then using the order over Q. Given a word $u \in (A \cup \bar{\varepsilon})^*$, let $v = \phi(u)$. The runs of $\mathcal{T}'$ over u are easily obtained from the runs of $\mathcal{T}$ over v by adding factors of states of $\bar{Q}$ over positions labelled by $\bar{\varepsilon}$. Since all states of Q are larger than states of $\bar{Q}$, the sets $\mathcal{R}((p, k, q), u)$ are always empty, k-synchronized, or k-stationary, for $k \in Q$. When $k \in \bar{Q}$, the runs of $\mathcal{R}((p, k, q), u)$ only have states in $\bar{Q}$. However, runs that are only over $\bar{Q}$ are extremely simple: only one state can appear in the run. These runs are forward or backward passes (depending on whether the state is in $\bar{Q}_{\rightarrow}$ or $\bar{Q}_{\leftarrow}$) over words in $(\bar{\varepsilon})^*$ which produce nothing. Hence $\mathcal{T}'$ is a W2NFT. ◀

Proof of 2. Let us consider a W2NFT $\mathcal{T}$ with underlying automaton $\mathcal{A}$ over alphabet A, realizing a relation T. Let f be a function realized by a transition-to-letter reversible transducer $\mathcal{S}$.

We define a new transducer $\mathcal{T}'$ which realizes $T \circ f$. The main idea is to define, as in [11], a transducer simulating $\mathcal{T}$ over the image by f of its input word u. Thus, to move to the right over $f(u)$, the automaton simulates one computation step of $\mathcal{S}$, and to move to the left, it simulates one step of computation of $\mathcal{S}$ but backwards, which is possible since $\mathcal{S}$ is reversible.

We denote Q the set of states of $\mathcal{T}$, P the set of states of $\mathcal{S}$, δ the transition relation of $\mathcal{T}$ and γ the transition function of $\mathcal{S}$. We also denote γ' the inverse of the γ relation, which is also functional. We denote the set of states of $\mathcal{T}'$ by $Q' = P \times Q$. We define $Q'_{\rightarrow} = P_{\rightarrow} \times Q_{\rightarrow} \cup P_{\leftarrow} \times Q_{\leftarrow}$ and $Q'_{\leftarrow} = P_{\rightarrow} \times Q_{\leftarrow} \cup P_{\leftarrow} \times Q_{\rightarrow}$. The idea is that when $\mathcal{T}$ has to move forward, we simulate $\mathcal{S}$, thus the direction of the state is the same as the direction of the $\mathcal{S}$ component. Conversely, when $\mathcal{T}$ has to move back, we need to simulate $\mathcal{S}$ in reverse, thus inverting the direction of the $\mathcal{S}$ component. Given a transition $(p_1, a, p_2) \in \gamma$ which produces b in $\mathcal{S}$ and a forward transition (from $Q_{\rightarrow}$ to $Q_{\rightarrow}$) (q_1, b, q_2), we add a transition $((p_1, q_1), a, (p_2, q_2))$. The idea is that with the information given by a and p_1, $\mathcal{T}'$ can simulate $\mathcal{T}$ over the corresponding position labelled by b. Similarly, if (q_1, b, q_2) is a right-to-right transition, we add the transition $((p_1, q_1), a, (p_2, q_2))$. When (q_1, b, q_2) is either a left-to-left or a backward (right-to-left) transition, we need to move the virtual reading head of $\mathcal{T}$ to the left. In that case, for any transition $(p_1, a, p_2) \in \gamma'$, we add a transition $((p_1, q_1), a, (p_2, q_2))$.

We want to show that the obtained transducer is a W2NFT. Let $u \in A^*$ and let $v = f(u) \in B^*$.

Let ρ be the run of $\mathcal{S}$ over u, and let ρ' be a run of $\mathcal{T}$ over v, with maximal state k. We describe the corresponding run of $\mathcal{T}'$ over ρ''. We can define the *origin function* of v as the function which maps a position of v to the position of u that was read in $\mathcal{S}$ when the position was produced. When $\mathcal{T}'$ is virtually over a position i of v, it is actually over position $o(i)$ of u. Moreover, the $\mathcal{S}$ state of the configuration is exactly the state where the ith output was produced, which is the one of the ith configuration of ρ. Thus ρ'' is simply ρ' where a configuration (q, i) is replaced by a configuration $(p_i, q, o(i))$ where p_i is the state of the ith configuration of ρ. What is key here is that the configurations of ρ'' where k appears only depend on the configurations of ρ' where k appears. We choose any order of $P \times Q$ which is compatible with the order over Q.

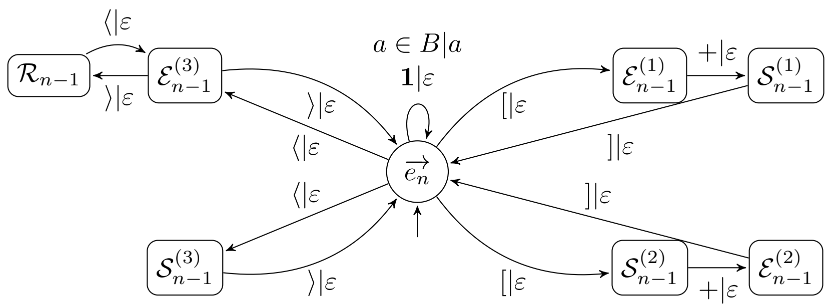
Let ρ'', λ'' be two runs in $\mathcal{R}(((p_1, q_1), (p_2, k), (p_3, q_3)), u)$, with $p_1, p_2, p_3 \in P$. We denote by ρ, ρ' the corresponding runs over u, v of $\mathcal{S}, \mathcal{T}$ respectively, and similarly for λ, λ'. Note that $\lambda = \rho$ since $\mathcal{S}$ is deterministic. Since the highest state appearing in both ρ', λ' is k, we have $\rho', \lambda' \in \mathcal{R}((q_1, k, q_3), v)$. If these runs are k-synchronized, then $pos_k(\rho') = pos_k(\lambda')$. We can obtain $pos_{(p_2,k)}(\rho'')$ by replacing configurations (k, i) by $((p_2, k), o(i))$ when p_2 is the state of the ith configuration of ρ. Since $\rho = \lambda$, we thus have that $pos_{(p_2,k)}(\rho'') = pos_{(p_2,k)}(\lambda'')$ meaning that ρ'', λ'' are (p_2, k)-synchronized. Similarly, assuming $pos_k(\rho'), pos_k(\lambda') \subseteq j^+$, we get $pos_{(p_2,k)}(\rho''), pos_{(p_2,k)}(\lambda'') \subseteq o(j)^+$ (the non-emptiness is by assumption), hence $\{\rho'', \lambda''\}$ is (p_2, k)-stationary. ◀

◀

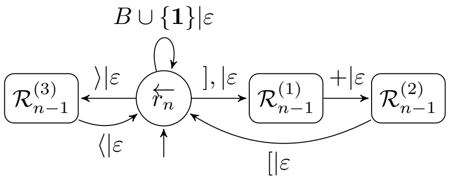
► **Proposition 18.** *For all n, the transduction $f_{eval,n}$ which evaluates a regular expression can be recognized by a* W2NFT $\mathcal{T}_{eval,n}$.

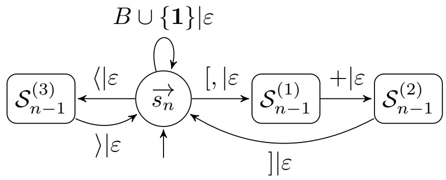
Proof. Figure 4 depicts three transducers $\mathcal{E}_n$, $\mathcal{R}_n$ and $\mathcal{S}_n$. They are designed inductively and modularly. In these pictures, circles represent states (the arrow in the states describes the reading direction) and rectangles with rounded corners represent a new instance of a transducer. An arrow to (resp. from) a rectangle is actually an edge to the initial state (resp. from the final state) of the instance it represents. Base cases are not represented here as they are trivial: $\mathcal{E}_0$, $\mathcal{R}_0$ and $\mathcal{S}_0$ are simply restricted to their initial state alone, with the self-loop and without component.

The transducer $\mathcal{E}_n$ of Figure 4a (with final state e_n) recognizes the evaluation transduction $f_{eval,n}$ from Example 5: it outputs the language denoted by a regular expression with nesting level n. One can show that this transducer is weakly ambiguous. ◀



(a) Transducer $\mathcal{E}_n$ evaluates any regular expression with nesting level n, *i.e.* it outputs the language denoted by the expression.





(b) Transducer $\mathcal{R}_n$ returns to the beginning of any regular expression with nesting level n, without producing anything.

(c) Transducer $\mathcal{S}_n$ skips any regular expression with nesting level n, without producing anything.

Figure 4 Weakly ambiguous 2NFT $\mathcal{E}_n$ for the evaluation function $f_{eval,n}$ of Example 5.

B Recap of the approach for deterministic two-way transducers

The approach of [14] applies Theorem 20 to the transition monoid $M_{\mathcal{T}}$. Let us consider some ε-free φ-good expression F. Given a sub-expression E of F, an RFE $f_{E,p,q}$ is built for each pair (p,q) of $m_E = \varphi(L(E)) \in M_{\mathcal{T}}$. For all $u \in L(E)$, $[\![f_{E,p,q}]\!](u)$ equals the output of the unique run r of $\mathcal{T}$ from p to q on u. We give an overview of the main ingredients of the construction by considering the most tricky case where $E = E_1^+$ and $p,q \in Q_{\rightarrow}$. It results from the study of the shape of LR proper runs r from p to q on words $u \in L(E)$. (see Figure 5):

1. Since E is unambiguous, u uniquely decomposes into $u_1 \dots u_n$ with each u_i in $L(E_1)$.
2. Since $\mathcal{T}$ is deterministic and since m_E is idempotent, r decomposes into a sequence $t_1, r_1, \dots, t_{n-1}, r_{n-1}, t_n$ where each r_i is a run on $u_i u_{i+1}@|u_i|, |u_i|$ and each t_i is a proper LR run on u_i.
3. The r_i's (and then the t_i's) have the same starting and ending states.
4. Each r_i decomposes into the same sequence $s_1, \dots, s_l$ of proper RR or LL runs;
5. The numbers l of sub-runs, as well as the starting state q_j and the ending state q_{j+1} of each s_j depend on E, p and q only. So they are the same for any proper run from p to q on any word in $L(E)$.

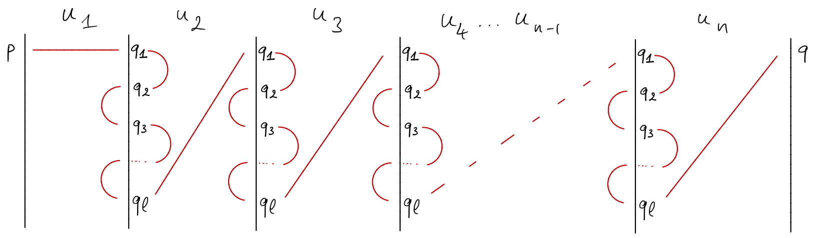


Figure 5 Decomposition of an LR-run on a word of $\phi^{-1}(m)$ with m an idempotent element of the transition monoid of a 2DFT.

Guided by the shape of the runs, we can build $f_{E,p,q}$: by induction hypothesis, we get the RFEs f_{E_1,p,q_1}, $f_{E_1,q_l,q}$, f_{E_1,q_l,q_1}, and all the $f_{E_1,q_j,q_{j+1}}$'s. Then, we can use l Cauchy products to combine them into an RFE f of domain $L(E_1)^2$ that captures the outputs of all the possible pieces of runs between two consecutive states q_1. The 2-chained star of f yields an RFE for the runs from the first q_1 to the last q_1 (which is equal to q by determinism). Finally, the latter is combined with f_{E_1,p,q_1} to get $f_{E,p,q}$.

In this paper, we start from a weakly ambiguous 2NFT $\mathcal{T}$. Because it is non-deterministic, the study of the shape of its runs is more difficult. In particular, the previous items 3-5 fail: not only do the runs $r_i :: t_{i+1}$ no longer have the same starting and ending states, but they also decompose in different ways, with a different number of components, possibly unbounded. In addition, runs from p to q on words in L decompose differently.

C Proofs of Subsection 4.1

The goal of this section is to study the shape of the runs r of a W2NFT $\mathcal{T}$. This study is done by decomposing r into proper sub-runs and determining their type and their rank depending on the type and the rank of r. Interesting results are obtained when r is a run on a word that corresponds to an idempotent element of the underlying transition monoid of $\mathcal{T}$. We will exploit these properties in the next section in order to get RREs from 2WFTs.

C.1 Concatenation of runs

The formal definition of concatenation of overlapping runs of a 2NFA is recalled below, inspired by the approach of [18].

Let u be a word. For all $0 \leqslant i, j \leqslant |u|$, we denote $u_{i,j}$ as the factor of u between positions i and j. Note that $u_{i,i} = \varepsilon$ and $u_{i,j} = u_{j,i}$ for all i, j. Let $\leqslant_p$ stand for the prefix order over A^* and $\leqslant_s$ stand for the suffix order over words. We define two operators on words, $\vee_p$ and $\vee_s$:

- $u \vee_p v$ equals u if $v \leqslant_p u$, or v if $u \leqslant_p v$, or undefined otherwise;
- $u \vee_s v$ equals u if $v \leqslant_s u$, or v if $u \leqslant_s v$, or undefined otherwise.

Let $r_1 = c_1 \dots c_n$ be a run on $u@i, j$ from p_1 to q_1 and $r_2 = c'_1 \dots c'_m$ be a run on $v@k, l$ from p_2 to q_2. They are concatenable if $q_1 = p_2$ and $w_1 = u_{0,j} \vee_s v_{0,k}$ and $w_2 = u_{j,|u_1|} \vee_p v_{k,|u_2|}$ are defined. When it is possible, the concatenation of r_1 and r_2, noted $r_1 :: r_2$, is the run $c''_1 \dots c''_{n+m-1}$ from p_1 to q_2 on $w_1 w_2 @c, d$ where $c = i + |w_1| - j$ and $d = l + |w_1| - k$, defined by:

- for all $1 \leqslant i \leqslant n$, $c''_i = (q, h + |w_1| - j)$ if $c_i = (q, h)$;
- for all $1 \leqslant i \leqslant m$, $c''_{i+n-1} = (q, h + |w_1| - k)$ if $c'_i = (q, h)$.

We extend the concatenation operator to sets of runs: $R_1 :: R_2$ consists of all runs $r_1 :: r_2$ such that r_1 and r_2 are two concatenable runs of R_1 and R_2 respectively. It is distributive over union. Note also that, given an order over the states, the concatenation of two runs r_1 and r_2 of ranks k_1 and k_2, when it exists, is a run of rank $\max(k_1, k_2)$.

C.2 Transition monoid for weakly ambiguous automata

▶ **Example 34.** Let's consider the weakly ambiguous 2NFT of Figure 2. The element $m_{ba} = \mu(ba)$ of its transition monoid contains the following triples: $(1,3,2)$, $(2,3,2)$, $(6,7,7)$ and $(7,3,2)$ of type LR; $(1,3,5)$, $(2,3,5)$, $(6,5,5)$ and $(7,3,5)$ of type LL; and $(5,6,6)$ of type RR. For the element $m_{baba} = \mu(baba)$, the LR triples are all the $(x,3,y)$ with $x \in \{1,2,6,7\}$ and $y \in \{2,7\}$. Its LR or RR triples are the same as for $\mu(ba)$. One can check that m_{baba} is idempotent (m_{ba} is not), and that $\mu^{-1}(m_{baba}) = (ba^+)^{\geqslant 2}$. We will see later in the section that it is not a coincidence if all the LR triples of m_{baba} have the same rank.

C.3 Proof of Lemma 22

▶ **Lemma 35.** *If r is a proper return run on $u \in L^*$, then its L-decomposition is r.*

Proof. This is equivalent to prove that the proper LL-run (resp. RR-run) r is actually a run on u_1 (resp. u_n). We prove it for proper LL-runs using an induction on their rank $k \in Q$. The proof for proper RR-runs is similar. Without loss of generality, we can suppose that $n \geqslant 3$ since every LL-run on u is also a run on uv for all $v \in L^+$. The base case and the inductive one are proved by contradiction.

Suppose that the LL-run r is not on u_0. Let p and p' be the starting and ending states of r and k be its rank. Since $\mu(L)$ is idempotent, there also exist:

- a proper LL-run r_0 from p to p' with rank k on u_0 (and then on $u_0u_1u_2$),
- a proper LL-run r_1 from p to p' with rank k on $u_0u_1u_2$ that is not a run on u_0u_1. This means there is in r_1 a configuration (p'', i) with $i \geqslant |u_0u_1|$.

Base case: $k = 1$. Then by definition of the rank, only state k appears in r_1 and r_0 (except for the first one that is p). So r_1 and r_0 cannot be k-synchronized nor k-stationary, which contradicts the fact that $\mathcal{T}$ is weakly ambiguous.

Inductive case. Since $\mathcal{T}$ is weakly ambiguous, r_0 and r_1 are either k-stationary or k-synchronized. In both cases, this means that state k appears at positions less that $|u_0|$. It follows that all the LL-sub-runs of r_1 that start and end at position $|u_0|$ have ranks less than k. The latter ones can be seen as proper LL-runs on $u_1 \dots u_n$. Then the induction hypothesis implies these runs are actually on u_1, and consequently, that r_1 is a run on u_0u_1. Hence, a contradiction. ◀

Proof of Lemma 22. By contradiction, suppose that the L-decomposition $\Delta_L(r) = (t_0, \dots, t_l)$ of r contains a LR-run and a RL-run. Let i and j ($i < j$) the least indexes such that t_i is LR and t_j is RL (or the reverse). Then $(t_i, t_{i+1}, \dots, t_j)$ is the L-decomposition of a LL or RR proper run, which contradicts Lemma 35. ◀

C.4 Proof of Proposition 23

The next property is a direct consequence of the idempotence of m.

▶ **Lemma 36.** *Let p and q two states of $\mathcal{G}_m$. If there is a path from p to q in $\mathcal{G}_m$ using transversal edges, then there also exists a path from p to q using exactly one transversal edge.*

Proof. Let $\rho = (p_0, k_0, q_0) \dots (p_l, k_l, q_l)$ be a path between p and q using $c \geqslant 2$ transversal edges. Let (p_i, k_i, q_i) and (p_j, k_j, q_j) be the first and the last ones. Let $u \in L$. Then for each $i \leqslant i' \leqslant j$, there exists a proper run $t_{i'}$ from $p_{i'}$ to $q_{i'}$ of rank $k_{i'}$ on u, and consequently $r = t_i :: \cdots :: t_j$ is an L-quasi-proper run from p_i to q_j on some power of u. Using Lemma 22 and Remark 21, we deduce that r is actually a proper transversal run on u^c. Then $(p_i, k, q_j) \in m$ for some rank k. Replacing $(p_i, k_i, q_i) \dots (p_j, k_j, q_j)$ with (p_i, k, q_j) in ρ gives the desired path. ◀

Proof of Proposition 23.1. Let C be a scc of $\mathcal{G}_m$. Let (p, k, q) be a transversal element of C. Since C is a scc, we can find a path $\rho = \prod_{i=0}^{n}(p_i, k_i, q_i)$ that starts and ends with the edge (p, k, q) and that goes through all edges of C (with possible edge repetitions).

Let $u \in L$ (we recall that $\mu(L) = m$). By definition, for each i, $(p_i, k_i, q_i) \in m$ implies that we can find a proper run r_i from p_i to q_i of rank k_i on u. It follows that $r_0 :: \cdots :: r_n$ is a quasi-proper run on some power of u from p_0 to q_n with rank $k_C = \max\{k_i \mid 0 \leqslant i \leqslant n\}$, namely the rank of C. The L-decomposition of r is $\Delta_L(r) = (r_0, \dots, r_n)$. As a first consequence, all traversal elements of C are of the same type (by Lemma 22). In addition, all transversal edges being of the same type, r is actually a proper transversal run on u^c where c is the number of transversal sub-runs of $\Delta_L(r)$. By idempotence, it follows that (p, k_C, q) is a transversal edge in $\mathcal{G}_m$, and thus in C.

By construction, (p, k_C, q) appears in ρ, saying at position j. Then, replacing r_0 with r_j in $\Delta_L(r)$ leads to another proper transversal run r' from p to q with rank k_C. So, r and r' synchronize on k_C which is only possible if $k = k_C$. ◀

Proof of Proposition 23.2. We prove it by contradiction.

Thanks to Lemma 36, we can find a state p of C_1 and a path $\rho_p = (p, k_p, p')\rho'_p$ from p to p such that (p, k_p, p') is a transversal edge and ρ'_p contains return edges only. Similarly, we can find a state q of C_2 and a path $\rho_q = \rho'_q(q', k_q, q)$ from q to q such that (q', k_q, q) is a transversal edge and ρ'_q contains return edges only.

Suppose there exists a path from the C_1 to C_2. Then there is also a path ρ_{pq} from p to q in $\mathcal{G}_m$. By Lemma 36, we can suppose that ρ_{pq} consists of exactly one transversal edge (p, k_{pq}, q).

Now consider the paths $\rho_1 = \rho_p\rho_{pq}\rho_q\rho_q\rho_q$ and $\rho_2 = \rho_p\rho_p\rho_p\rho_{pq}\rho_q$. These two paths contain precisely five transversal edges, all the same type (by Lemma 22). Let $u \in L$. Following Remark 21, we can find two proper transversal runs $r_1 = r_p :: r_{pq} :: r_q :: r_q :: r_q$ and $r_2 = r_p :: r_p :: r_p :: r_{pq} :: r_q$ both on u^5 from p to q where r_{pq} is a proper transversal run on u from p to q with some rank k_{pq}, r_p is a run from p to p with some rank k_p and r_q is a run from q to q with some rank k_q.

Since the rank of r_1 and r_2 is $k = \max\{k_p, k_q, k_{pq}\}$, these two runs synchronize on k. So it is for the proper sub-runs $r'_1 = r_{pq} :: r_q :: r_q$ and $r'_2 = r_p :: r_p :: r_{pq}$. Since $k = \max\{k_p, k_q, k_{pq}\}$, the state k necessary appears in the prefix $r_{pq} :: r_q$ of r'_1 or the prefix r_p of r'_2. So the k-synchronisation of r'_1 and r'_2 entails that $k = k_p$. But in this case, there exists $|u^2| < j \leqslant |u^3|$ such that (k, j) is a configuration of the prefix $r_p :: r_p$ on $u^5@|u|, |u^3|$ of r'_2. By synchronization, (k, j) is also a configuration of the suffix $r_q :: r_q$ on $u^5@|u^2|, |u^4|$ of r'_1. Then there exists a run on $u^5@|u^2|, |u^3|$ from q to p. This means by Remark 21 that a path from q to p exists in $\mathcal{G}_m$ and then $q \sim_m p$. ◀

C.5 Proof of Proposition 27

Proof of Proposition 27. Let $n \geqslant 2|Q| + 3$ and $u \in L^n$. Let r be a proper transversal run on u and $\Delta_L(r) = (t_1, t_2, \dots, t_l)$ be its L-decomposition where each t_i is a run from some

p_i to some q_i with some rank k_i. This decomposition contains at least $n \leqslant l$ transversal sub-runs. Moreover, they have all the same type (Lemma 22).

Let i be the integer such that t_i is the $|Q|+1$-th transversal edge in $\Delta_L(r)$, and j be the integer such that t_j is the $n-(|Q|+1)$-th transversal edge in $\Delta_L(r)$. Then $r_1 = t_1 :: \cdots :: t_i$ and $r_3 = t_j :: \cdots :: t_l$ are as expected. We let $r_2 = t_{i+1} :: \cdots :: t_{j-1}$.

By Remark 21 there is a path $\rho = (p_1, k_1, q_1) \ldots (p_l, k_l, q_l)$ in $\mathcal{G}_m$. Furthermore, there are $|Q|$ transversal edges before (p_i, k_i, q_i), and $|Q|$ transversal edges after (p_j, k_j, q_j) in ρ. Then two $\sim_m$-equivalent states necessary appear in the prefix $\rho_1 = (p_1, k_1, q_1) \ldots (p_i, k_i, q_i)$ of ρ as well as in the suffix $\rho_2 = (p_j, k_j, q_j) \ldots (p_l, k_l, q_l)$ of ρ. By Proposition 23.2 these four states, as well as all intermediate states, are $\sim_m$-equivalent. In particular, $p_i \sim_m q_i \sim_m p_j \sim_m q_j$.

Since t_i is transversal run from p_i, t_j is a transversal run to q_j and $p_i \sim_m q_j$, Corollary 25 ensures that t_i and t_j have a rank greater than the sub-run r_2. It follows that r_1 and r_3 have a rank greater than, or equal to, the one of r_2. ◀

D Proofs of Subsection 4.2

We detail some points of the construction of Section 4.2 not developed in the main section.

D.1 Dealing with the endmarkers

We recall we get:

$$[\![\mathcal{T}]\!](\eta(\vdash u \dashv)) = [\![\bigoplus_{m \in \mu(\vdash L \dashv)} out_{E_m, e_\mathfrak{f}}]\!](\vdash u \dashv) \quad \text{for all } u \in L_\mathcal{T}.$$

The RRE $f'_\mathcal{T} = \bigoplus_{m \in \mu(\vdash L \dashv)} out_{E_m, e_\mathfrak{f}}$ has domain $\vdash L \dashv$, whereas we need a RRE with domain $L_\mathcal{T}$. The reader will easily able to check that the way we will construct each expression $out_{E_m, e_\mathfrak{f}}$ ensures that the following two properties are satisfied: (1) all its sub-expressions f are on a domain included in A^+ or $\{\vdash\}A^*$ or $A^*\{\dashv\}$ or $\{\vdash\}A^*\{\dashv\}$; (2) the Hadamard products always operate on two RREs with the same domain. We can take advantage of these properties to define inductively from each sub-expression f a new RRE $\zeta(f)$ of domain $\eta(\mathrm{dom}(f))$ such that $[\![\zeta(f)]\!](\eta(v)) = [\![f]\!](v)$: if $\mathrm{dom}(f) \subseteq A^*$, then $\zeta(f) = f$, otherwise,

- if f equals $\mathrm{dom}(f)/v$, then $\zeta(f) = \eta(\mathrm{dom}(f))/v$;
- if $f = f_1 \odot f_2$ then $\zeta(f) = \zeta(f_1) \odot \zeta(f_2)$ for all $\odot \in \{\oplus, \bullet, \otimes\}$;
- if $f = f_1^{\otimes}$, then $\zeta(f) = \zeta(f_1)^{\otimes}$.

Note that if $f = f_1^{\circledast, L, k}$, then $\mathrm{dom}(f) \subseteq A^+$. Moreover, because of their definition domains, if $\mathrm{dom}(f_1)$ and $\mathrm{dom}(f_2)$ are unambiguously concatenable, so it is for $\eta(\mathrm{dom}(f_1))$ and $\eta(\mathrm{dom}(f_2))$. The proof that $\eta(f)$ is as expected is immediate, using a simple induction and properties (1) and (2).

As a direct consequence, the RRE $f_\mathcal{T} = \zeta(f'_\mathcal{T})$ has domain $L_\mathcal{T}$ and is equivalent to $\mathcal{T}$.

D.2 Proof of Lemma 28

We first recall this lemma:

▶ **Lemma 28.** *For any ε-free μ-good regular expression F and $e = (p, k, q) \in \mu(L(F))$, we can compute an RRE $out_{F,e}$ with domain $L(F)$ such that $[\![out_{F,e}]\!](u) = \{output(r) \mid r \in R(e, u)\}$.*

Let r be a run and k be a state that appears in r. We denote the prefix sub-run of r to the first occurrence of k as $pr_k(r)$, and the suffix sub-run of r from the first occurrence of k as $su_k(r)$.

We will prove the following property:

▶ **Lemma 37.** *For any ε-free μ-good regular expression F and $e = (p, k, q) \in \mu(L(F))$, we can compute two RREs $pr_{F,e}$ and $su_{F,e}$ with domain $L(F)$ such that:*

- $[\![pr_{F,e}]\!](u) = \{output(pr_k(r)) \mid r \in R(e, u)\}$,
- $[\![su_{F,e}]\!](u) = \{output(su_k(r)) \mid r \in R(e, u)\}$.

We first explain why this result allows to prove Lemma 28. This immediately follows from the next Lemma.

▶ **Lemma 38.** *If $\mathcal{T}$ is weakly ambiguous then $[\![out_{F,e}]\!] = [\![pr_{F,e} \otimes su_{F,e}]\!]$.*

Proof. By definition, their domains are equal. We only prove that $[\![pr_{F,e}]\!] \otimes [\![su_{F,e}]\!] \subseteq [\![out_{F,e}]\!]$, the other one being trivial. Let $u \in L(F)$ and $\alpha \in [\![pr_{F,e} \otimes su_{F,e}]\!](u)$. By definition of Hadamard product, there are α_1 and α_2 such that $\alpha = \alpha_1\alpha_2$, $\alpha_1 \in [\![pr_{F,e}]\!](u)$ and $\alpha_2 \in [\![su_{F,e}]\!](u)$. So, by definition of the function $pr_{F,e}$ and $su_{F,e}$, there exist two runs $r_1, r_2 \in R(e, u)$ such that $\alpha_1 = output(pr_k(r_1))$ and $\alpha_2 = output(su_k(r_2))$. Since $\mathcal{T}$ is weakly ambiguous, r_1 and r_2 are k-stationary or k-synchronized. In both cases, this implies that $r = pr_k(r_1) :: su_k(r_2)$ is a run in $R(e, u)$. Clearly, $output(r) = \alpha_1\alpha_2$. ◀

We turn now to the proof of Lemma 37. Let F be an ε-free μ-good regular expression, $\mu(L(F)) = \{m_F\}$ and $\hat{e} = (\hat{p}, \hat{k}, \hat{q}) \in m_F$. We will now express the transductions $pr_{F,\hat{e}}$ and $su_{F,\hat{e}}$ as regular relation expressions using a structural induction on F.

Base case and union case

Suppose that $F = a \in V$. Let $m = \mu(a)$ and $\hat{e} = (\hat{p}, \hat{k}, \hat{q}) \in m$. Then, by construction of M, there is a transition $t = (\hat{p}, a, \hat{q}) \in \delta$ such that $\hat{k}$ equals $\hat{q}$. We set $pr_{F,\hat{e}} = a/out(t)$ and $su_{F,\hat{e}} = a/\varepsilon$.

Suppose that $F = F_1 + F_2$ and let $L = L(F)$, $L_1 = L(F_1)$ and $L_2 = L(F_2)$. Since the expression F is good, we deduce that $\mu(L) = \mu(L_1) = \mu(L_2) = \{m\}$. Let $\hat{e} = (\hat{p}, \hat{k}, \hat{q}) \in m$. We set $pr_{F,\hat{e}} = pr_{F_1,\hat{e}} \oplus pr_{F_2,\hat{e}}$ and $su_{F,\hat{e}} = su_{F_1,\hat{e}} \oplus su_{F_2,\hat{e}}$.

Concatenation case

Suppose that $F = F_1 \cdot F_2$ and let $L = L(F)$, $L_1 = L(F_1)$ and $L_2 = L(F_2)$. Since the expression F is good, $\mu(L)$, $\mu(L_1)$ and $\mu(L_2)$ are singletons, respectively noted $\{m_F\}$, $\{m_{F_1}\}$ and $\{m_{F_2}\}$. Let's $\hat{e} = (\hat{p}, \hat{k}, \hat{q}) \in m_F$. We compute the regular relation expressions $pr_{F,\hat{e}}$ and $su_{F,\hat{e}}$ by analyzing the different ways the runs in $R(\hat{e}, L)$ decompose w.r.t. L_1 and L_2.

Let u be in L and r be a proper run on u from $\hat{p}$ to $\hat{q}$ with rank $\hat{k}$ (namely $r \in R(\hat{e}, u)$). Since F is good, L_1 and L_2 are unambiguously concatenable. So, u uniquely decomposes into vw with v in L_1 and w in L_2. Let $\Delta_{L_1,L_2}(r) = (t_1, \ldots, t_n)$ be the decomposition of r w.r.t. L_1 and L_2. Then, the t_i's are proper sub-runs from some p_i to some q_i on some k_i alternatively on v or w and such that $t_1 :: \cdots :: t_n = r$. More precisely, t_1 is on v if $\hat{p} \in Q_\rightarrow$ while t_n is on v if $\hat{q} \in Q_\leftarrow$. Otherwise, they are on w.

We aim to build a RRE $pr_{F,\hat{e}}$ such that $[\![pr_{F,\hat{e}}]\!](u) = \{output(pr_{\hat{k}}(r)) \mid r \in R(\hat{e}, u)\}$ for all $u \in L$. So, only the prefix $pr_{\hat{k}}(r)$ of r to the first occurrence of state $\hat{k}$ is of interest. Since r has rank $\hat{k}$, we have necessary that

$$pr_{\hat{k}}(r) = t_1 :: \cdots :: t_{l-1} :: pr_{\hat{k}}(t_l) \tag{1}$$

where l is the first index such that t_l has rank $\hat{k}$.

We abstract this decomposition by the word $(x_1, e_1) \ldots (x_l, e_l)$, over the alphabet $M_{F_1,F_2} = \{1, 3\} \times m_{F_1} \cup \{2, 4\} \times m_{F_2}$, such that $e_i = (p_i, k_i, q_i)$, $x_1 = 1$ if $\hat{p} \in Q_\rightarrow$ (otherwise $x_1 = 2$)

and all the x_i's are alternatively equal to 1 or 2. The first component x_i tells us if e_i comes from m_{F_1} or m_{F_2} and makes the abstraction independent of the type of run (RL, LR, RR or LL). This word contains precisely one element of rank k, the last one. We tag this element by replacing it with $(3, e_l)$ or $(4, e_l)$ depending on $x_l = 1$ or 2. The resulting word is denoted $\sigma_{<\hat{k}}(r)$. We also set $L^{<\hat{k}}_{F,\hat{e}} = \{\sigma_{<\hat{k}}(r) \mid r \in R(\hat{e}, u), u \in L(F)\}$. This set is not empty and does not contain the empty word. It is easy to prove from Equation 1 that the next equation holds. For all $v \in L_1$ and $w \in L_2$:

$$pr_{\hat{k}}(R(\hat{e}, vw)) = \bigcup_{(x_1,e_1)\ldots(x_l,e_l) \in L^{<\hat{k}}_{F,\hat{e}}} \left(\prod_{i=1}^{n-1} R(e_i, u_i) \right) pr_{\hat{k}}(R(e_l, u_l)) \tag{2}$$

where u_i equals v if $x_i \in \{1, 3\}$, or w otherwise. Thus, $L^{<\hat{k}}_{F,\hat{e}}$ abstracts the runs in $pr_{\hat{k}}(R(\hat{e}, L))$.

▶ **Lemma 39.** *$L^{<\hat{k}}_{F,\hat{e}}$ is a regular language over M_{F_1,F_2}.*

Proof. We can easily define a language $L_{F,\hat{e}}$ that abstracts precisely all the possible decomposition of runs over L_1L_2 from p to q of rank $\hat{k}$: the language $L_{F,\hat{e}}$ contains all words $(x_1, (p_1, k_1, q_1)) \ldots (x_n, (p_n, k_n, q_n))$ in $M^*_{F_1,F_2}$ such that

- we have $k_i \leqslant \hat{k}$ for all $1 \leqslant i \leqslant n$, and $k_j = \hat{k}$ for some j; $x_1 = 1$ iff $\hat{p} \in Q_{\rightarrow}$, and for all $2 \leqslant i \leqslant n$, $x_i = 1$ if $x_{i-1} = 2$, otherwise $x_i = 2$;
- $p_1 = \hat{p}$, $q_n = \hat{q}$ and for all $0 < i < n$, $q_i = p_{i+1}$.

This language is clearly regular. Now tag each word of $L_{F,\hat{e}}$ by replacing the first occurrence of a letter $(x_j, (p_j, k_j, q_j))$ with $k_j = \hat{k}$ with $(3, (p_j, k_j, q_j))$ or $(4, (p_j, k_j, q_j))$ depending on $x_j = 1$ or 2, and called $L'_{F,\hat{e}}$ the resulting language. It is also clearly regular. Since $L^{<\hat{k}}_{F,\hat{e}}$ consists of the prefixes of $L_{F,\hat{e}}$ such that the only element of rank $\hat{k}$ is the last one, it is also regular. ◀

Consider for each $e_1 \in m_{F_1}$ and $e_2 \in m_{F_2}$ the RREs built using the induction hypothesis:

$$\widehat{out}_{F_1,e_1} = out_{F_1,e_1} \bullet L_2/\varepsilon \qquad \widehat{out}_{F_2,e_2} = L_1/\varepsilon \bullet out_{F_2,e_2}$$
$$\widehat{pr}_{F_1,e_1} = pr_{F_1,e_1} \bullet L_2/\varepsilon \qquad \widehat{pr}_{F_2,e_2} = L_1/\varepsilon \bullet pr_{F_2,e_2}.$$

Each of them has domain $L = L(F)$. From any regular expression E over M_{F_1,F_2} that does not use $\mathbf{0}$ as atom, we can inductively build a RRE $\nu(E)$ with domain L as follows:

- $\nu(\varepsilon) = L/\varepsilon$;
- if $E = (x_i, e)$ and $x_i \in \{1, 2\}$ then $\nu(E) = \widehat{out}_{F_{x_i},e}$;
- if $E = (x_i, e)$ and $x_i \in \{3, 4\}$, then $\nu(E) = \widehat{pr}_{F_{x_i-2},e}$;
- if $E = E_1 + E_2$ then $\nu(E) = \nu(E_1) \oplus \nu(E_2)$;
- if $E = E_1 \cdot E_2$ then $\nu(E) = \nu(E_1) \otimes \nu(E_2)$;
- if $E = E_1^*$ then $\nu(E) = \nu(E_1)^{\otimes}$.

▶ **Lemma 40.** *Let E be a regular expression over M_{F_1,F_2} that does not use $\mathbf{0}$ as atom. For all $u \in L$, we have*

$$[\![\nu(E)]\!](u) = \bigcup_{\alpha \in L(E)} [\![\nu(\alpha)]\!](u).$$

Proof. We proceed by induction on the structure of E. We give the proof for the star case only, that is when $E = E_1^*$. The other cases are quite simple. Let $L_{E_1} = L(E_1)$.

$$
\begin{aligned}
[\![\nu(E_1^*)]\!](u) &= [\![\nu(E_1)^{\otimes}]\!](u) && \text{(def. of } \nu) && (3)\\
&= ([\![\nu(E_1)]\!](u))^* && \text{(def. of Had. star)} && (4)\\
&= \bigcup_{i=0}^{\infty} ([\![\nu(E_1)]\!](u))^i && \text{(def. Kleene star)} && (5)\\
&= \bigcup_{i=0}^{\infty} \left(\bigcup_{\alpha \in L(E_1)} [\![\nu(\alpha)]\!](u) \right)^i && \text{(by induction)} && (6)\\
&= \{\varepsilon\} \cup \bigcup_{i=1}^{\infty} \bigcup_{(\alpha_1,\ldots,\alpha_i) \in L_{E_1}^i} \prod_{j=1}^{i} [\![\nu(\alpha_j)]\!](u) && && (7)\\
&= \{\varepsilon\} \cup \bigcup_{i=1}^{\infty} \bigcup_{(\alpha_1,\ldots,\alpha_i) \in L_{E_1}^i} [\![\bigotimes_{j=1}^{i} \nu(\alpha_j)]\!](u) && \text{(def. of Had. prod.)} && (8)\\
&= \{\varepsilon\} \cup \bigcup_{i=1}^{\infty} \bigcup_{(\alpha_1,\ldots,\alpha_i) \in L_{E_1}^i} [\![\nu(\alpha_1 \ldots \alpha_i)]\!](u) && \text{(def. of } \nu) && (9)\\
&= [\![\nu(\varepsilon)]\!](u) \cup \bigcup_{i=1}^{\infty} \bigcup_{\alpha \in L_{E_1}^i} [\![\nu(\alpha)]\!](u) && && (10)\\
&= \bigcup_{\alpha \in L_{E_1}^*} [\![\nu(\alpha)]\!](u) && && (11)
\end{aligned}
$$

◀

We pick up a regular expression $E_{F,\hat{e}}$ (without $\mathbf{0}$) denoting the non-empty language $L_{F,\hat{e}}^{<k}$, and set $pr_{F,\hat{e}} = \nu(E_{F,\hat{e}})$.

▶ **Lemma 41.** *For all $u \in L(F)$, we have $[\![pr_{F,\hat{e}}]\!](u) = output(pr_k(R(\hat{e}, u)))$.*

Proof. Since $F = F_1 \cdot F_2$ is a good expression, u uniquely decomposes into vw with $v \in L(F_1)$ and $w \in L(F_2)$. Equation 2 immediately implies that $output(pr_k(R(\hat{e}, vw)))$ is equal to

$$
\bigcup_{(x_1,e_1)\ldots(x_n,e_n) \in L_{F,\hat{e}}^{<k}} \left(\prod_{i=1}^{n-1} output(R(e_i, u_i)) \right) output(pr_k(R(e_n, u_n)))
$$

where u_i equals v (resp. w) if x_i equals 1 (resp. 2).

By induction, this is equal to

$$
\bigcup_{(x_1,e_1)\ldots(x_n,e_n) \in L_{F,\hat{e}}^{<k}} \left(\prod_{i=1}^{n-1} [\![out_{F_{x_i},e_i}]\!](u_i) \right) [\![pr_{F_{x_n},e_n}]\!](u_n) \qquad (12)
$$

So we have the following equalities:

$$(12) = \bigcup_{(x_1,e_1)\dots(x_n,e_n)\in L^{<k}_{F,\hat{e}}} \left[\!\!\left[\bigotimes_{i=1}^{n-1} \widehat{out}_{F_{x_i},e_i} \otimes \widehat{pr}_{F_{x_n},e_n}\right]\!\!\right](vw) \tag{13}$$

$$= \bigcup_{(x_1,e_1)\dots(x_n,e_n)\in L^{<k}_{F,\hat{e}}} [\![\nu((x_1,e_1)\dots(x_n,e_n))]\!](vw) \quad \text{def. of } \nu \tag{14}$$

$$= \bigcup_{\alpha\in L^{<k}_{F,\hat{e}}} [\![\nu(\alpha)]\!](vw) \tag{15}$$

$$= [\![\nu(E_{F,\hat{e}})]\!](vw) \quad \text{by Lemma 40} \tag{16}$$

$$= [\![pr_{F,\hat{e}}]\!](vw) \quad \text{def. of } pr_{F,\hat{e}} \tag{17}$$

◀

The construction of $su_{F,\hat{e}}$ is very similar. In this case, we are interested in the suffix $su_k(r)$ of r from the first occurrence of state k. Then, if r decomposes into $t_1 :: \cdots :: t_n$ then Equation 1 becomes

$$su_k(r) = su_k(t_l) :: t_{l+1} :: \cdots :: t_n \tag{18}$$

where l is the first index such that t_l has rank k. The function $\sigma_{<k}$ and the language $L^{<k}_{F,\hat{e}}$ are adapted accordingly. In particular, it is now the first element of each word in $L^{<k}_{F,\hat{e}}$ that is tagged with 3 or 4. Finally, the function ν changes slightly: if $E = (x_i, e)$ and $x_i \in \{3, 4\}$, then $\nu(E) = \widehat{su}_{F_{x_i-2},e}$.

Kleene iteration case

Suppose that $F = F_1^+$, and let $L = L(F_1)$. Then $L(F) = L^+$. Since F is μ-good, $\{m_F\}$ is equal to $\{m_{F_1}\}$. Moreover, m_F is idempotent and L is unambiguously iterable. We distinguish two cases depending on the type of $\hat{e}$.

Suppose that $\hat{e}$ is LL, namely $\hat{p} \in Q_\to$ and $\hat{q} \in Q_\leftarrow$ (the RR case is similar). In this case, we can show from Lemma 22 that $R(\hat{e}, L^+) = R(\hat{e}, L)$. So we use the induction hypothesis and set: $pr_{F,\hat{e}} = pr_{F_1,\hat{e}} \bullet L^*/\varepsilon$ and $su_{F,\hat{e}} = su_{F_1,\hat{e}} \bullet L^*/\varepsilon$.

Suppose now that $\hat{e}$ is LR, namely $\hat{p}, \hat{q} \in Q_\to$ (the RL case is similar). We only describe the main ideas to build $pr_{F,\hat{e}}$, those for $su_{F,\hat{e}}$ being similar. First, following the approach developed for the concatenation, we can build the RREs $pr_{F_1^i,\hat{e}}$ for any $i \leqslant 2|Q|+2$ where $|Q|$ is the number of states of the transducer. We show below how to build the RRE $pr_{F_1^{\geqslant 2|Q|+3},\hat{e}}$. The RRE $pr_{F,\hat{e}}$ is then computed as the sum of all of them.

A long proper LR run r in $R(\hat{e}, L^{\geqslant 2|Q|+3})$ can be decomposed into $r_1 :: r_2 :: r_3$ as described in Proposition 27. In this decomposition, states p and q belong to a same (non-trivial) SCC C of $\mathcal{G}_{m_F}$ and are in $Q_\to$ (as $\hat{p}$). Lemma 22 entails that r_1, r_3 and all the transversal runs of $\Delta(r_2)$ have the same type as r. So they are all proper LR runs. Furthermore, by Corollary 25, r_2 and all transversal runs of $\Delta(r_2)$ have the same rank k_C, and the return ones have rank less than (or equal to) k_C.

We present the main ideas of the proof in the simpler case where r_2 is always a proper LR run. The general case only adds some uninteresting technical details that makes a little more complicate the decomposition of Equation (19) below. With this assumption, we can partition the set $R(\hat{e}, L^{\geqslant 2|Q|+3})$ according to the intermediate states p and q that appear in the decomposition and the ranks k_1 and k_2 of r_1 and r_2 as follows:

$$R(\hat{e}, L^{\geqslant 2|Q|+3}) = \biguplus_{\substack{p,q \text{ in a scc } C \text{ of } \mathcal{G}_m \\ \hat{k}=\max(k_1,k_2)}} R((\hat{p}, k_1, p), L^{|Q|}) :: R((p, k_C, q), L^{\geqslant 3}) :: R((q, k_2, \hat{q}), L^{|Q|}) \quad (19)$$

We let $e = (p, k_C, q)$. The last technical difficulty of the proof is to define an RRE $out'_{F,e}$ with domain $L^{\geqslant 3}$ and which maps any word $v \in L^{\geqslant 3}$ to $output(R(e, v))$. Once this is done, the expected RRE $pr_{F_1^{\geqslant 2|Q|+3},\hat{e}}$ can be obtained using adequate combinations of the RREs $out'_{F,e}$, $pr_{F_1^{|Q|},(\hat{p},k_1,p)}$ and $su_{F_1^{|Q|},(q,k_2,\hat{q})}$, depending on whether $\hat{k}$ equals k_1 or k_2 (recall that $k_C \leqslant k_1, k_3$ by Proposition 27).

We detail now the construction of $out'_{F,e}$. Let $w \in L^3$ that uniquely decomposes into $w_1 w_2 w_3$ with $w_i \in L$. Let $p', q' \in C \cap Q_\rightarrow$. Any run $r \in R((p', k_C, q'), w)$ can be decomposed into three sub-runs: the prefix $\hat{pr}_{k_C}(r)$ of r that ends to the first occurrence of k_C between positions $|w_1| + 1$ and $|w_1 w_2|$; the suffix $\hat{su}_{k_C}(r)$ of r that starts from the first occurrence of k_C between positions $|w_1 w_2| + 1$ and $|w_1 w_2 w_3|$; and the remaining infix $\hat{in}_{k_C}(r)$ of r. Proposition 26 implies that the sets $\hat{pr}_{k_C}(R(p', k_C, q'), w)$, $\hat{su}_{k_C}(R(p', k_C, q'), w)$ and $\hat{in}_{k_C}(R((p', k_C, q'), w))$ do not depend on p' and q'. More generally, for all $v \in L^{\geqslant 3}$ with $v = v_1 \dots v_l$ its unique decomposition (L is unambiguously iterable), we have

$$R(e, v) = \hat{pr}_{k_C}(R(e, v_1 v_2 v_3)) :: \prod_{2 \leqslant i \leqslant l-1} \hat{in}_{k_C}(R(e, v_{i-1} v_i v_{i+1})) :: \hat{su}_{k_C}(R(e, v_{l-2} v_{l-1} v_l))$$

Again, we can adapt the approach used for the concatenation to build RREs $\hat{pr}_{F_1,e}$, $\hat{su}_{F_1,e}$ and $\hat{in}_{F_1,e}$ that map any word $w \in L^3$ to $output(\hat{pr}_{k_C}(R(e, w)))$, $output(\hat{su}_{k_C}(R(e, w)))$ and $output(\hat{in}_{k_C}(R(e, w)))$, respectively. We get:

$$\begin{aligned} output(R(e, v)) &= [\![\hat{pr}_{F_1,e}]\!](v_1 v_2 v_3) \prod_{2 \leqslant i \leqslant l-1} [\![\hat{in}_{F_1,e}]\!](v_{i-1} v_i v_{i+1}) [\![\hat{su}_{F_1,e}]\!](v_{l-2} v_{l-1} v_l) \\ &= [\![\hat{pr}_{F_1,e}]\!](v_1 v_2 v_3) [\![\langle \hat{in}_{F_1,e} \rangle^{\circledast, L, 3}]\!](v_1 \dots v_l) [\![\hat{su}_{F_1,e}]\!](v_{l-2} v_{l-1} v_l) \\ &= [\![(\hat{pr}_{F_1,e} \bullet L^*/\varepsilon) \otimes \langle \hat{in}_{F_1,e} \rangle^{\circledast, L, 3} \otimes (L^*/\varepsilon \bullet \hat{su}_{F_1,e})]\!](v) \end{aligned}$$

The latter RRE is the expected $out'_{F,e}$.

E Proof of Proposition 13

In this section we prove the decidability of the property of weak ambiguity. To do so, since this is a property of the runs, we will first define crossing sequences automata, that capture families of runs in a nice way, and exhibit some interesting properties, which we will use later on for the decidability.

E.1 Crossing sequences

▶ **Definition 42.** *Let $\mathcal{A} = (Q_\rightarrow, Q_\leftarrow, \mathfrak{i}, \mathfrak{f}, \delta_\mathcal{A})$ be a two-way automaton. The* crossing sequence *of a run ρ at position i is the sequence of states obtained by the projection π_i from $(Q \times \mathbb{N})^*$ to $(Q \times \{i\})^*$ applied to the run: we keep only the states that were in a configuration at position i.*

The crossing sequence of the run ρ of the automaton $\mathcal{A}$ at position i over the word u is noted $CS^{(u)}_\mathcal{A}(\rho, i)$, or $CS(\rho, i)$ when the rest is clear from context.

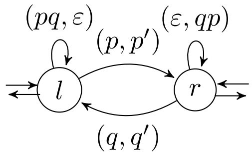


Figure 6 Automaton recognizing a-joinable crossing sequences.

We will want to know when two given crossing sequences can be consecutive, *i.e.* when there exists a run to wich both belong, one at position i ant the other at position $i+1$. This is a local property, in the sense that it does not depend on the whole word (*cf. e.g.* [21]). Formally:

▶ **Definition 43.** *a pair (c, c') of crossing sequences is said to be a-*joinable*, with $a \in A_{\vdash\dashv}$, if with $c = c_1 \dots c_n$ and $c' = c'_1 \dots c'_m$, the pair of words (c, c') over the alphabet $(Q_\rightarrow + Q_\leftarrow)^+$ is accepted by the automaton $\mathcal{T}_a = (Q, I, F, \delta)$, drawn in Fig.6, with:*

- *$Q = I = F = \{l, r\}$ two states, both initial and final,*
- *δ is the following set of transitions, for all $p, p' \in Q_\rightarrow$ and $q, q' \in Q_\leftarrow$:*
  - *from l to r, labeled by (p, p'), if $(p, a, p') \in \delta_\mathcal{A}$,*
  - *from r to l, labeled by (q, q'), if $(q, a, q') \in \delta_\mathcal{A}$,*
  - *from l to l, labeled by (pq, ε), if $(p, a, q) \in \delta_\mathcal{A}$,*
  - *from r to r, labeled by (ε, qp), if $(q, a, p) \in \delta_\mathcal{A}$,*

These objects are often defined for deterministic automata, because then there is a finite number of crossing sequences of accepting runs: no state could appear twice on the same crossing sequence. This allows to construct an automaton based on such objects, that recognize accepting runs of the original deterministic automaton.

However it is still possible to build a finite automaton based on these objects in the non-deterministic case, if we consider crossing sequences where we allow states to repeat at most once (*cf. e.g.* [10]). In a sense, a crossing sequence with a repetition is a witness of an infinite number of actual crossing sequences with unbounded size, because the run between the two occurences of the repeating state can be repeated as often as wanted. Note that if we construct an automaton with such crossing sequences, allowing at most two occurences of the states, we will not recognize all runs of the automaton, but only the ones that take at most once any loop they encounter, loop meaning here a run that goes from a configuration back to itself. However this information is enough for our usage, because from such runs one could rebuild all missing runs.

▶ **Definition 44.** *Let $\mathcal{A}$ be a two-way automaton. A* crossing sequence with one repetition*, noted $CS1(\rho, i)$, is a crossing sequence in which no states appears more than twice. Note that there is only a finite number of possible such crossing sequences. We note RCS this finite set.*

We want to build an automaton from these objects that will allow us to capture proper runs of the initial automaton. But since we will be interested by runs of rank k, and LL or RR runs may have any word as prefix or suffix, this construction is not exactly the classical one.

Let $\mathcal{A}$ be a two-way automaton, and k, p, q states of $\mathcal{A}$. We want an automaton whose accepting runs are in bijection with runs of $R(p, k, q)$, the union of $R((p, k, q), u)$ for all $u \in A^*$ [3], that do not go more than twice through a given state at the same position in a

[3] We will consider afterwards the product of this automaton with itself, ensuring that we only consider one word at the same time.

given word.

How we do so will depend on the nature, forward or backward, of p and q, but the main idea is always the same, having crossing sequences in states, and a transition whenever the crossing sequences are joinable.

We then trim this automaton to only keep runs that go through a state -a crossing sequence of $\mathcal{A}$- that contains a k. This can be done by making a copy of the automaton with a flag remembering wether or ot we have already encountered a state containing k. This step is ommited for clarity.

If p and q are both forward:

▶ **Definition 45.** *The* LR-automaton of crossing sequences with repetitions *of $\mathcal{A}, p, q$ is the one-way automaton $(Q_{LR}, I_{LR}, F_{LR}, \delta_{LR})$ with:*

- $Q_{LR} = RCS \cap (Q_{\rightarrow} \times Q_{\leftarrow})^* \times Q_{\rightarrow}$,
- $I_{LR} = \{p\}$,[4]
- $F_{LR} = \{q\}$,[5]
- $\delta_{LR}(c, a)$ *is the set of all crossing sequences c' such that (c, c') is a-joinable.*

If p and q are both backward:

▶ **Definition 46.** *The* RL-automaton of crossing sequences with repetitions *of $\mathcal{A}, p, q$ is the one-way automaton $(Q_{RL}, I_{RL}, F_{RL}, \delta_{RL})$ with:*

- $Q_{RL} = RCS \cap (Q_{\leftarrow} \times Q_{\rightarrow})^* \times Q_{\leftarrow}$,
- $I_{RL} = \{q\}$,
- $F_{RL} = \{p\}$,
- $\delta_{RL}(c, a)$ *is the set of all crossing sequences c' such that (c, c') is a-joinable.*

When the proper runs are of the form LL or RR, remark that any word on which such a run is valid can be extended to another valid support, adding any suffix for LL-runs, and any prefixes for RR-runs. Hence we will need to add a special state to handle this property.

If p is forward and q backward:

▶ **Definition 47.** *The* LL-automaton of crossing sequences with repetitions *of $\mathcal{A}, p, q$ is the one-way automaton $(Q_{LL}, I_{LL}, F_{LL}, \delta_{LL})$ with:*

- $Q_{LL} = RCS \cap (Q_{\rightarrow} \times Q_{\leftarrow})^+ \cup \{\triangleleft\}$,
- $I_{LL} = \{(p, q)\}$,
- $F_{LL} = \triangleleft$,
- $\delta_{LL}(c, a)$ *is the set of all crossing sequences c' such that (c, c') is a-joinable. We also add $(\triangleleft)$ to this set if (c, ε) is a-joinable; and for all a, $\triangleleft \in \delta_{LL}(\triangleleft, a)$.*

If p is backward and q forward:

▶ **Definition 48.** *The* RR-automaton of crossing sequences with repetitions *of $\mathcal{A}, p, q$ is the one-way automaton $(Q_{RR}, I_{RR}, F_{RR}, \delta_{RR})$ with:*

- $Q_{RR} = RCS \cap (Q_{\leftarrow} \times Q_{\rightarrow})^+ \cup \{\triangleright\}$,
- $I_{RR} = \triangleright$,
- $F_{RR} = \{(p, q)\}$,
- $\delta_{RR}(c, a)$ *is the set of all crossing sequences c' such that (c, c') is a-joinable. We also add c to the set $\delta_{RR}(\triangleright, a)$ if (ε, c) is a-joinable; and for all a, $\triangleright \in \delta_{LL}(\triangleright, a)$.*

---

[4] We do not allow proper runs to go again through the starting position.

[5] Likewise for the ending position of the run.

Algorithm 1 **Aut_Synch**($\mathcal{A}$)

```
for k in states of A in decreasing order (for <) do
    for p, q in states of A do
        if not Runs(A, <, k, p, q) then
            return False
        end if
    end for
end for
return True
```

▶ **Proposition 49.** *Accepting runs of the automaton of crossing sequences with repetitions of* $\mathcal{A}, p, q$ *are in bijection with proper runs in* $R((p,k,q),u)^{(<3)}$*, runs of* $R(p,k,q)$ *that do not go more than twice through a given state at the same position in a given word.*

Proof. This comes from the fact that by construction, runs of $\mathcal{A}$ and sequences of crossing sequences (of unbouded length) are in bijection. ◀

E.2 The Decidability Algorithm

We want to show that the following problem is decidable:

Weakly Ambiguous?
Input: A two-way automaton $\mathcal{A}$.
Output: The boolean value of the proposition "$\mathcal{A}$ is weakly ambiguous."

One possible way of solving this is to enumerate all possible orders, until we find one that checks the property needed for weak-ambiguity. This means that we need to exhibit an algorithm to solve the following problem:

Aut_Synch?
Input: A two-way automaton $\mathcal{A}$, $<$ a total order on states of $\mathcal{A}$.
Output: The boolean True iff for all states k, p, q, and all words u, $R((p,k,q),u)$ is either k-synchronous or k-stationary.

Assuming we have an algorithm **Runs**$(\mathcal{A}, <, k, p, q)$ that can say whether $R((p,k,q),u)$ is either k-synchronous or k-stationary for all words u, the algorithm 1 solves **Aut_Synch?**.

But before describing such an algorithm **Runs**, we are going to need a few lemmas.

▶ **Lemma 50.** *Let $\mathcal{A}$ be an automaton, $<$ an order on its states, u a word, and k, p, q states of $\mathcal{A}$.*

$R((p,k,q),u)$ is neither k-stationary nor k-synchronous if and only if one of the following two properties is true:

- *there exists two runs $\rho_1, \rho_2 \in R((p,k,q),u)$ and a position i such that k appears at position i in ρ_1 and does not appear at position i in ρ_2.*
- *there exists a run $\rho \in R((p,k,q),u)$ and two positions i, j such that k appears in ρ twice at positions i and at least once in position j.*

Proof. Assume $R((p,k,q),u)$ is neither k-stationary nor k-synchronous.

This means that $\{Pos_k(\rho) \mid \rho \in R((p,k,q),u)\}$ is not a singleton, *i.e.* there are two runs $\rho_1, \rho_2 \in R((p,k,q),u)$ such that $Pos_k(\rho_1) \neq Pos_k(\rho_2)$.

Now, with P_1 the set of positions in $Pos_k(\rho_1)$, and P_2 the same for ρ_2, consider the two following cases:

- $P_1 \neq P_2$. This means that there exists a position i such that k appears at position i in exactly one of the two runs, meaning that the first property holds.
- $P_1 = P_2$. Therefore, the occurrences of k in ρ_1 and ρ_2 must be in different order. By non-stationarity, we can assume that there exists two positions i, j where k appears. Therefore, we have $\rho_1 = \sigma_1(k,i)\sigma_2(k,j)\sigma_3$, and $\rho_2 = \tau_1(k,j)\tau_2(k,i)\tau_3$. Consider now the run $\tau_1(k,j)\tau_2(k,i)\sigma_2(k,j)\sigma_3$. This run belong to $R((p,k,q),u)$ and checks the second property.

This proves the implication.

For the other direction, consider $P = \{Pos_k(\rho) \mid \rho \in R((p,k,q),u)\}$. If the first property holds, P is not a singleton, and we therefore don't have k-synchronization. Furthermore $Pos_k(\rho_2)$ will not be a subset of i^+, and neither will P for any position j because i belongs to P from ρ_1. In the second case, the occurrence of k twice at the same position is proof that there exists an infinity of runs that can be build from this one by looping on the part between these two occurrences. The set $\{Pos_k(\rho) \mid \rho \in R((p,k,q),u)\}$ is therefore not a singleton, and since there is another occurrence of k in at least one other position, it is not a subset of i^+ either. Therefore, in both cases, $R((p,k,q),u)$ is neither k-stationary nor k-synchronous. ◀

▶ **Lemma 51.** *Let $\mathcal{A}$ be an automaton, $<$ an order on its states, u a word, and k, p, q states of $\mathcal{A}$.*

With $R((p,k,q),u)^{(<3)}$ the set of runs of $R((p,k,q),u)$ whose crossing sequences do not contain the same state more than two times, $R((p,k,q),u)$ is neither k-stationary nor k-synchronous if and only if one of the following two properties is true:

- *there exists two runs $\rho_1, \rho_2 \in R((p,k,q),u)^{(<3)}$ and a position i such that k appears at position i in ρ_1 and does not appear at position i in ρ_2.*
- *there exists a run $\rho \in R((p,k,q),u)^{(<3)}$ and two positions i, j such that k appears in ρ twice at positions i and at least once in position j.*

This is the lemma above, but with the restriction that the runs in the second part of the equivalence can be chosen among $R((p,k,q),u)^{(<3)}$. To prove this, consider first that if there is a run of $R((p,k,q),u)$ where the state k appears several times at the same position, infinitely many runs of $R((p,k,q),u)$ can be constructed from it. Indeed, if $\rho = \sigma_0(k,i)\sigma_1(k,i)\sigma_2(k,i)\ldots\sigma_n(k,i)\sigma'$ belong to $R((p,k,q),u)$, then so do all runs in the language $\sigma_0(k,i)\left((\sigma_1+\sigma_2+\ldots+\sigma_n)(k,i)\right)^*\sigma'$. In particular, $\sigma_0(k,i)\sigma'$ is in $R((p,k,q),u)$. Meaning that for all runs in $R((p,k,q),u)$ there exist a run in $R((p,k,q),u)$ where no state is visited twice. In a sense, we can get rid of the loops.

This operation applied to ρ_1 and ρ_2 of the first property of the equivalence gives the result. For the second property, we do basically the same thing, except that we keep one occurrence of the loop, to ensure that we do not get rid of the portion of the run containing k at position j.

Therefore, from that lemma, we can write Algorithm 2.

E.3 Correctness of the algorithm Runs

Remark first that since by Property 49, an accepting run of $\mathcal{C}$ describes a proper run of $R((p,k,q),u)^{(<3)}$, an accepting run of $\mathcal{D}$ describes two proper runs of $R((p,k,q),u)^{(<3)}$ on the same word, and states of $\mathcal{D}$ are two crossing sequences at the same position. This will allows us to prove that the algorithm checks the properties of Lemma 51. For convenience, let us note:

Algorithm 2 **Runs**$(\mathcal{A}, <, k, p, q)$

```
C ← Repeated_crossing_sequences_automaton(A, k, p, q)
D ← Trim(C × C)
for all distinct pairs of states (c1, c2), (c3, c4) of D belonging to an accepting run of D do
    if k appears in c1 XOR k appears in c2 then
        return False
    end if
    if k appears in each c1, c2, c3, c4 AND k appears twice in one of (c1, c2, c3, c4) then
        return False
    end if
end for
return True
```

- P_1: there exists two runs $\rho_1, \rho_2 \in R((p,k,q),u)^{(<3)}$ and a position i such that k appears at position i in ρ_1 and does not appear at position i in ρ_2.
- P_2: there exists a run $\rho \in R((p,k,q),u)^{(<3)}$ and two positions i, j such that k appears in ρ twice at positions i and at least once in position j.
- A_1: There exists a state (c_1, c_2) of $\mathcal{D}$ such that k appears in c_1 XOR k appears in c_2.
- A_2: k appears in each c_1, c_2, c_3, c_4 AND k appears twice in one of (c_1, c_2, c_3, c_4).

We prove the following:

(i) $P_1 \Rightarrow A_1$;
(ii) $A_1 \Rightarrow P_1$;
(iii) $P_2 \Rightarrow A_1 \vee A_2$;
(iv) $A_2 \Rightarrow P_2$.

i: $P_1 \Rightarrow A_1$. P_1 means that there exists a run of $\mathcal{D}$ and a state (c_1, c_2) on this run, where the state k appears exactly in one of the two crossing sequences c_1, c_2.

ii: $A_1 \Rightarrow P_1$. Existence of a state means existence of a run of $\mathcal{D}$, and therefore of two runs of $\mathcal{C}$, for which there is a position where k appears only in one of these runs.

iii: $P_2 \Rightarrow A_1 \vee A_2$.

P_2 means that there exists two crossing sequences c_1, c_3, at two different positions, where c_1 contains k twice, and c_3 at least once.

Now, in pairs of states of $\mathcal{D}$, to capture c_1 and c_3 on the same run, it means that there exists c_2, c_4, such that the pair is one of $(c_1,c_2),(c_3,c_4)$, $(c_3,c_4),(c_1,c_2)$, $(c_2,c_1),(c_4,c_3)$, or $(c_4,c_3),(c_2,c_1)$.

Assume both c_2 and c_4 contain k. We immediately have A_2. But if one of them does not contain k, it means that the state of $\mathcal{D}$ from which it comes is comprised of two crossing sequences, one that contains k, and the other who does not: this is A_1.

iv: $A_2 \Rightarrow P_2$. Assume wlog that k appears twice in c_1. There is a run of $R((p,k,q),u)^{(<3)}$ that contains both c_1 and c_3. Which is the run for P_2.