Aperiodicity of Rational Functions Is PSPACE-Complete*

Emmanuel Filiot[1], Olivier Gauwin[2], and Nathan Lhote[2]

1 **Université Libre de Bruxelles, Belgium**
2 **Université de Bordeaux, LaBRI, CNRS, France**
3 **Université Libre de Bruxelles, Belgium; and**
Université de Bordeaux, LaBRI, CNRS, France

Abstract

It is known that a language of finite words is definable in monadic second-order logic – MSO – (resp. first-order logic – FO –) iff it is recognized by some finite automaton (resp. some aperiodic finite automaton). Deciding whether an automaton A is equivalent to an aperiodic one is known to be PSPACE-complete. This problem has an important application in logic: it allows one to decide whether a given MSO formula is equivalent to some FO formula. In this paper, we address the aperiodicity problem for functions from finite words to finite words (transductions), defined by finite transducers, or equivalently by bimachines, a transducer model studied by Schützenberger and Reutenauer. Precisely, we show that the problem of deciding whether a given bimachine is equivalent to some aperiodic one is PSPACE-complete.

1998 ACM Subject Classification F.4.3 Mathematical Logic and Formal Languages

Keywords and phrases Rational word transductions, decision problem, aperiodicity, bimachines

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2016.13

1 Introduction

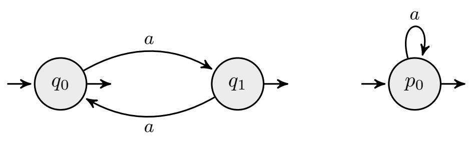
Rational languages and the aperiodicity problem

The theory of rational languages (of finite words) is robust, due to many characterizations coming from different domains, such as computation, logic and algebra. For instance, it is well-known that a language is rational iff it is recognized by some finite automaton, iff it is definable in monadic second-order logic with one successor (MSO), iff its right syntactic congruence (also known as Myhill-Nerode congruence) has finite index. The latter algebraic characterization is closely related to the existence of a unique minimal deterministic automaton for every rational language, the states of which are the classes of the right syntactic congruence. Connections between computation, logic and algebra have been established for subclasses of rational languages. Perhaps the most important example, based on seminal works by Schützenberger [21], McNaughton and Papert [16], is the class of aperiodic languages, characterized by *aperiodic* automata, *first-order* logic (FO), and *aperiodic* right syntactic congruences. See also [23] for other classes.

Thanks to the (effective) logic-automata connections, results in logic can be obtained from results in automata, which are well-suited for algorithmic analysis. An important example

* This work was partially supported by the *ExStream* project (ANR-13-JS02-0010), the ARC project *Transform* (Federation Wallonia-Brussels) and the Belgian FNRS CDR project *Flare*. Emmanuel Filiot is research associate at F.R.S.-FNRS.

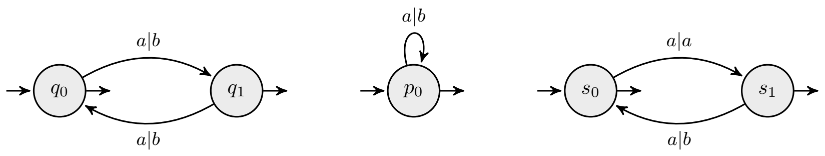
which motivates this paper is the FO in MSO definability problem: given an MSO formula, is it equivalent to some FO formula, over finite strings? In automata-theoretic terms, this amounts to decide whether a given automaton $\mathcal{A}$ is equivalent to an aperiodic one. We illustrate this problem on an example, by considering the following deterministic automata which both recognize a^* (all states are final, and all states but q_1 are initial):



Roughly, an automaton is aperiodic if for some n, for all words u, u^n and u^{n+1} behave the same with respect to their effect on states. For instance, in the left automaton, any word has one of the two following behaviours: either sending q_0 to q_0 and q_1 to q_1, or q_0 to q_1 and q_1 to q_0. This automaton is not aperiodic because a^n and a^{n+1} have necessarily two different behaviours, for all $n \geq 0$. However, this left automaton is equivalent to the right one, which is aperiodic. In general, it is not easy to see whether some automaton $\mathcal{A}$ is equivalent to an aperiodic one, and this problem is known to be PSpace-complete when $\mathcal{A}$ is deterministic To decide it, the connection between automata and algebra plays an important role. Indeed, since aperiodic automata and aperiodic right congruences both characterize the same class of languages, it suffices to (1) minimize $\mathcal{A}$ into the unique minimal automaton $\mathcal{A}_m$ (which is an effective representation of the right syntactic congruence of $L(\mathcal{A})$) and (2) decide whether $\mathcal{A}_m$ is aperiodic. It is well-known that step (1) is in PTime since $\mathcal{A}$ is deterministic, and step (2) is known to be in PSpace [22], and this is optimal [5]. In this paper, our goal is to extend this decidability result to functions of finite words, called transductions.

Rational transductions and the aperiodicity problem

A transduction is a function of finite words. Rational transductions are the transductions realized by finite automata with outputs, called *transducers* [2]. As an example, consider the three following transducers:

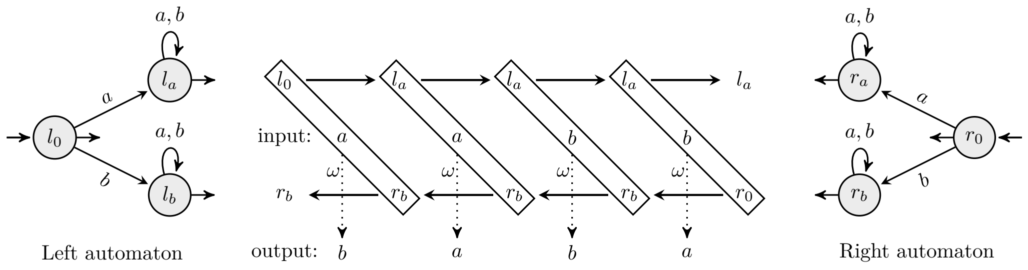


The left one maps any word of the form a^n to b^n. The middle one realizes the same transduction, and the right one maps any word of the form a^{2n} to $(ab)^n$, and any a^{2n+1} to $(ab)^n a$. *Aperiodic rational transductions* are the transductions realized by transducers with aperiodic underlying input automata. E.g., the transducer on the left is not aperiodic, but is equivalent to the middle one, which is aperiodic. Hence both transducers realize an aperiodic rational transduction. However, the transducer on the right is not aperiodic, and is not equivalent to any aperiodic transducer. The left and right transducers are almost the same, but one realizes an aperiodic rational transduction while the other does not. It shows that to decide whether a transducer is equivalent to an aperiodic one, outputs must be taken into account as well, reasoning only on the underlying input automata is not sufficient.

The *aperiodicity problem* asks, given some effective representation of a rational transduction, whether this transduction is aperiodic. It has been shown in [11] that two-way transducers (resp. aperiodic two-way transducers [14, 4]) are expressively equivalent to

MSOT (resp. FOT), a formalism introduced by Courcelle in the general context of graph transductions [7]. This equivalence carries over to the subclass of rational functions (resp. aperiodic rational functions) by considering MSOT (resp. FOT) with the natural restriction of *order-preserving* [3, 12]. As for languages, solving the aperiodicity problem also solves the logic definability problem FOT in MSOT (resp. order-preserving FOT in order-preserving MSOT). As we have seen, for rational languages, the aperiodicity checking procedure heavily relies on the existence of a unique minimal deterministic automaton. However in the setting of transductions, determinism is not sufficient to capture all rational transductions. In transducer theory, the notion of determinism is called *sequentiality*, a transducer being sequential if its underlying input automaton is deterministic. Consider the transduction swap that swaps the first and last letter of a word, *i.e.* maps any word of the form $\sigma w\beta$, where σ, β are symbols and w is a word, to $\beta w\sigma$. If the alphabet has more than one symbol, the transduction swap cannot be realized by a sequential transducer, although it can be easily shown that it is rational. The reason is that any transducer realizing it should guess in advance the last symbol β, by using non-determinism.

To overcome this issue, it has been shown that any rational transduction is the composition of a (left) sequential and a right sequential transduction [10]. In other words, any rational transduction can be realized by composing a sequential transducer that reads input words from right to left, and a sequential transducer that reads words from left to right. This idea has been captured in a single deterministic device called *bimachine*, introduced by Schützenberger [20] and studied by Eilenberg [9], and Reutenauer and Schützenberger [17]. Intuitively, a bimachine is made of two deterministic automata $\mathcal{L}$ and $\mathcal{R}$, and some output function ω, and works as follows: $\mathcal{R}$ processes an input word w from right to left and annotates it with its states. Symmetrically, $\mathcal{L}$ processes w from left to right and annotates it with its states. Finally, the output function ω is applied to any triple (r, σ, l) of the annotated word, where r is a state of $\mathcal{R}$, σ is an input symbol of w, and l is a state of $\mathcal{L}$. For example, consider again the transduction swap on the alphabet $\{a, b\}$. It is realized by the following bimachine with a left deterministic automaton that remembers whether the prefix read so far is empty (state l_0), starts with a (state l_a) or starts with b (state l_b). Symmetrically, the deterministic right automaton remembers information about suffixes. Finally, the output function ω maps any triple of the form (l_0, σ, r_β) or (l_β, σ, r_0) to β, and any other triple (l, σ, r) to σ, for $\sigma, \beta \in \{a, b\}$. An execution on $aabb$ is illustrated on the next figure:



Contributions

A bimachine is aperiodic if its two left and right automata are aperiodic. Aperiodic bimachines define exactly aperiodic rational transductions [18]. In this paper, our main result is to provide optimal complexity (PSPACE-complete) of the aperiodicity problem for rational transductions represented by bimachines.

We detail our contributions more precisely. In language theory, solving the aperiodicity problem relies on the existence of a unique deterministic automaton. For transductions, there is no unique minimal (deterministic) bimachine in general, but a canonical bimachine attached to every rational transduction has been defined by Reutenauer and Schützenberger [17], and this machine can be effectively constructed from a transducer or a bimachine realizing the transduction. As a first contribution, we show that a rational transduction is aperiodic iff its canonical bimachine is aperiodic. As a consequence, this gives an algorithm to solve the aperiodicity problem: (1) construct the canonical bimachine and (2) check whether its left and right automata are aperiodic, using the algorithm of [5].

Unfortunately, step (1) cannot be done in PTime and this is unavoidable: the canonical bimachine may be exponentially bigger than the initial bimachine. Instead of constructing the canonical bimachine, we show that it is sufficient to construct another minimal bimachine of polynomial size, which is aperiodic iff the function it realizes is aperiodic rational. This other bimachine is constructed via a PTime generalization of a minimization procedure for automata to bimachines. This yields in the end a PSpace algorithm, whose correctness is proved based on the aperiodicity of the canonical bimachine. The lower bound is immediate as it is already the case of deterministic automata.

Comparison with [13]

The aperiodicity problem was already shown to be decidable in [13], however with a more general procedure working for any (decidable) variety of congruences (e.g. the class of commutative congruences). More precisely, it is shown in [13] that the following problem is decidable: given a transducer, is it equivalent to some transducer whose transition congruence belongs to some decidable variety $\mathbf{V}$. It is shown that a transduction is in $\mathbf{V}$ iff one the minimal bimachines is in $\mathbf{V}$, and hence decidability comes as follows: construct the set of all minimal bimachines (shown to be finite) and test whether one of them belongs to $\mathbf{V}$. Instantiated by the variety of aperiodic congruences, this solves the aperiodicity problem, however with non-optimal complexity (several exponentials). Moreover, it is shown in [13] that the canonical bimachine of Reutenauer and Schützenberger does not necessarily preserve the equalities of a variety in general. In this paper, we show instead that for aperiodic congruences, the canonical bimachine is necessarily aperiodic if the transduction is, a result which is crucial to obtain an optimal procedure.

A last improvement compared to [13] is the following: in [13], we defined a rational transduction to be aperiodic (and more generally in a variety $\mathbf{V}$) if it is realized by an *unambiguous* aperiodic transducer (or an unambiguous $\mathbf{V}$-transducer). This definition was motivated by the fact that unambiguous transducers already capture all rational functions. For a general variety $\mathbf{V}$, this left open the problem of whether any transduction realized by a $\mathbf{V}$-transducer is realizable by an unambiguous $\mathbf{V}$-transducer. In this paper, we close this problem for the case of aperiodicity: as we show, a transduction is realized by some aperiodic transducer (not necessarily unambiguous) iff its canonical bimachine is aperiodic, and any aperiodic bimachine can be turned into an aperiodic unambiguous transducer.

2 Rational languages and transductions

Words and languages

An alphabet Σ is a finite set of symbols, and a word over Σ is an element of the free monoid Σ^* whose neutral element is denoted by ϵ. For $w \in \Sigma^*$, we write $|w|$ for its length and in particular, $|\epsilon| = 0$. For a non-empty word w and $i \in \{1, \dots, |w|\}$, we denote by $w[i]$ the ith symbol of w. For $u, v \in \Sigma^*$, we write $u \preceq v$ is u is a prefix of v and in this case we denote by $u^{-1}v$ the unique word v' such that $v = uv'$.

By $u \wedge v$ we denote the *longest common prefix* of any two words u and v, and by $\|u,v\|$ the value $|u|+|v|-2|u \wedge v|$. It is well-known that $\|.,.\|$ defines a distance. Finally, a *language* $L \subseteq \Sigma^*$ is a set of words.

Finite automata

A *finite automaton* (or just automaton for short) over an alphabet Σ is a tuple $\mathcal{A} = (Q, I, F, \Delta)$ where Q is a finite set of states, $I \subseteq Q$ (resp. $F \subseteq Q$) is a set of initial (resp. final) states, and $\Delta \subseteq Q \times \Sigma \times Q$ is a transition relation. A *run* r of an automaton $\mathcal{A} = (Q, I, F, \Delta)$ on a word $w \in \Sigma^*$ of length n is a word $r = q_0 \dots q_n$ over Q such that $(q_i, w[i+1], q_{i+1}) \in \Delta$ for all $i \in \{0, \dots, n-1\}$. The run r is accepting if $q_0 \in I$ and $q_n \in F$. A word is accepted by $\mathcal{A}$ if there exists an accepting run of $\mathcal{A}$ over it. The *language recognized by* $\mathcal{A}$ is the set $[\![\mathcal{A}]\!]$ of words accepted by $\mathcal{A}$. We write $p \xrightarrow{w}_{\mathcal{A}} q$ (or simply $p \xrightarrow{w} q$) whenever there exists a run r on w such that $r[1] = p$ and $r[|r|] = q$. An automaton $\mathcal{A}$ is *deterministic* if $|I| = 1$ and for any two rules $(p, \sigma, q_1), (p, \sigma, q_2) \in \Delta$, it holds that $q_1 = q_2$. It is *unambiguous* if for any word there exists at most one accepting run of $\mathcal{A}$ on it. It is *complete* if for every state $p \in Q$ and symbol $\sigma \in \Sigma$, $(p, \sigma, q) \in \Delta$ for some $q \in Q$.

Congruences and recognizability

Equivalently, rational languages can be defined as the languages recognized by congruences of finite index. We present these notions. Let Σ be an alphabet and let $\sim$ be an equivalence relation on Σ^*. We say that $\sim$ is a *right congruence* (resp. *left congruence*) if it satisfies $u \sim v \Rightarrow u\sigma \sim v\sigma$ (resp. $u \sim v \Rightarrow \sigma u \sim \sigma v$) for all $u, v \in \Sigma^*, \sigma \in \Sigma$. A *congruence* is both a left and right congruence. For $u \in \Sigma^*$, the equivalence class of u is denoted by $[u]_\sim$ (or $[u]$ if $\sim$ is clear from the context), and $\Sigma^*/_\sim = \{[u]_\sim \mid u \in \Sigma^*\}$ is called the quotient of Σ^* by $\sim$. We say that $\sim$ has *finite index* if $\Sigma^*/_\sim$ is finite. Concatenation naturally extends to congruence classes as follows: for all $u, v \in \Sigma^*$, $[u]_\sim [v]_\sim = [uv]_\sim$. Since $\sim$ is a congruence, the latter is well-defined. With this operation, $\Sigma^*/_\sim$ forms a monoid whose neutral element is $[\epsilon]_\sim$.

▶ **Example 1.** We will extensively use the following examples of congruences in this paper: the *syntactic congruence* $\equiv_L$ of a language L, the *transition congruence* $\approx_{\mathcal{A}}$ of an automaton $\mathcal{A}$ with set of states Q and if $\mathcal{A}$ is deterministic with initial state q_0, the *right transition congruence* $\sim_{\mathcal{A}}$. They are defined as follows, for any two words $u, v \in \Sigma^*$

$$\begin{array}{ccccllcl}
u & \equiv_L & v & \Leftrightarrow & (\forall x, y \in \Sigma^*, & xuy \in L & \Leftrightarrow & xvy \in L) \\
u & \approx_{\mathcal{A}} & v & \Leftrightarrow & (\forall p, q \in Q, & p \xrightarrow{u}_{\mathcal{A}} q & \Leftrightarrow & p \xrightarrow{v}_{\mathcal{A}} q) \\
u & \sim_{\mathcal{A}} & v & \Leftrightarrow & (\forall q \in Q, & q_0 \xrightarrow{u}_{\mathcal{A}} q & \Leftrightarrow & q_0 \xrightarrow{v}_{\mathcal{A}} q)
\end{array}$$

In particular, if $\mathcal{A}$ is deterministic and complete, then $[u]_{\sim_{\mathcal{A}}}$ can be identified with the state of $\mathcal{A}$ reached by u from the initial state. In this paper, we often make this identification, implicitly assuming that $\mathcal{A}$ is complete[1], and rather denote $[u]_{\mathcal{A}}$ instead of $[u]_{\sim_{\mathcal{A}}}$.

A language $L \subseteq \Sigma^*$ is *recognized by a congruence* $\sim$ if it is equal to the union of some equivalence classes of $\Sigma^*/_\sim$, *i.e.* $L = \{u \in \Sigma^* \mid [u] \in P\}$ for some $P \subseteq \Sigma^*/_\sim$. E.g., L is recognized by $\equiv_L$, by taking $P = L/_{\equiv_L}$, and any language L is rational iff it is recognized by a congruence of finite index (see for instance [23]).

[1] Any automaton can be made complete in polynomial-time.

Equivalence relations can be compared with respect to their granularity: if $\sim_1, \sim_2$ are two equivalence relations on Σ^*, we say that $\sim_1$ is *finer* than $\sim_2$ (or $\sim_2$ is *coarser* than $\sim_1$), if for all $u, v \in \Sigma^*$ such that $u \sim_1 v$, it holds $u \sim_2 v$. We write $\sim_1 \sqsubseteq \sim_2$ to mean that $\sim_1$ is finer than $\sim_2$. E.g., $\equiv_L$ is the coarsest congruence recognizing L, for any language L. In this paper, we also compare deterministic automata $\mathcal{A}_1, \mathcal{A}_2$ with respect to their right transition congruence, by saying that $\mathcal{A}_1$ is *finer* than $\mathcal{A}_2$ if $\sim_{\mathcal{A}_1} \sqsubseteq \sim_{\mathcal{A}_2}$. We write $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$ when $\mathcal{A}_1$ is finer than $\mathcal{A}_2$. For example, the minimal deterministic automaton recognizing a language L is the coarsest deterministic automaton recognizing L with respect to $\sqsubseteq$.

Rational transductions and finite transducers

A *transduction* f over a finite alphabet Σ is a partial function from Σ^* to Σ^*, whose domain is denoted by $\mathrm{dom}(f)$. We are interested in the class of rational transductions, defined by finite transducers. A *finite transducer* [2] (or just transducer for short) over an alphabet Σ is a tuple $\mathcal{T} = (\mathcal{A}, o, i, t)$ where $\mathcal{A} = (Q, I, F, \Delta)$ is a finite automaton, $o : \Delta \to \Sigma^*$ is the *output function*, $i : I \to \Sigma^*$ is the *initial function* and $t : F \to \Sigma^*$ is the *final function*. The transducer $\mathcal{T}$ realizes a binary relation $[\![\mathcal{T}]\!] \subseteq \Sigma^* \times \Sigma^*$ defined as follows. Let $r = q_0 \dots q_n$ be a run of $\mathcal{A}$ on some word u. We write $q_0 \xrightarrow{u|v}_{\mathcal{T}} q_n$ (or simply $q_0 \xrightarrow{u|v} q_n$) whenever $q_0 \xrightarrow{u}_{\mathcal{A}} q_n$ and $v = o(q_0, u[0], q_1) \dots o(q_{n-1}, u[n], q_n)$. If r is an accepting run and $w = i(q_0) v t(q_n)$ then we say that (u, w) is *realized* by $\mathcal{T}$, call u an *input* word and w an *output* word. The *relation realized* by $\mathcal{T}$ is the set $[\![\mathcal{T}]\!] = \{(u, w) \mid (u, w) \text{ is realized by } \mathcal{T}\}$.

A transducer $\mathcal{T} = (\mathcal{A}, o, i, t)$ is *functional* if it realizes a transduction (a function). This property is decidable in PTime (see [1] for instance). $\mathcal{T}$ is called *unambiguous* (resp. *sequential*) if $\mathcal{A}$ is unambiguous (resp. deterministic). In both cases $[\![\mathcal{T}]\!]$ is a transduction and we denote $(u, w) \in [\![\mathcal{T}]\!]$ by $[\![\mathcal{T}]\!](u) = w$. The class of *rational transductions* (resp. *sequential transductions*) is defined as the class of transductions realized by finite transducers (resp. sequential transducers). It is also known (see [2]) that a transduction is rational iff it is realized by some unambiguous transducer.

Aperiodicity

We define the notion of aperiodicity for congruences, automata, transducers and rational transductions. First, a congruence $\sim$ on Σ^* is *aperiodic* if for some $n > 0$ and all words $w \in \Sigma^*$, we have $w^n \sim w^{n+1}$. Aperiodicity is stable by taking coarser congruences, *i.e.* if $\sim_1 \sqsubseteq \sim_2$ and $\sim_1$ is aperiodic, then so is $\sim_2$. A deterministic automaton $\mathcal{A}$ is aperiodic if $\approx_{\mathcal{A}}$ is aperiodic. In other words, $\mathcal{A}$ is aperiodic if for some $n > 0$, for all words w and states p, q, we have $p \xrightarrow{w^n} q$ iff $p \xrightarrow{w^{n+1}} q$. Deciding whether a deterministic automaton is aperiodic is PSpace-complete [5]. Since the minimal deterministic automaton $\mathcal{A}_L$ recognizing a language L is the coarsest automaton recognizing L, and aperiodicity is stable by taking coarser congruences, $\mathcal{A}_L$ is aperiodic iff there is some aperiodic deterministic automaton recognizing L. Therefore, deciding whether a deterministic automaton is *equivalent* to some aperiodic one is also PSpace-complete, because minimizing a deterministic automaton can be done in PTime. Finally, a transducer $\mathcal{T} = (\mathcal{A}, o, i, t)$ is *aperiodic* if $\mathcal{A}$ is aperiodic, and a transduction f is *aperiodic rational* (resp. *aperiodic sequential*) if it is realized by some aperiodic transducer (resp. aperiodic sequential transducer).

[2] This type of transducer is sometimes called *real-time transducer* [19]. In the general case, a transition of a transducer may be labelled by any word, even empty. However such a transducer is equivalent to a real-time one if it realizes a function.

3 Bimachines and minimization

In this section we define bimachines, and associated operators *Left* and *Right*. Then we define canonical bimachines, that will be used in Section 4 to prove that the minimal bimachine $\mathit{Left}(\mathit{Right}(\mathcal{B}))$ is aperiodic iff the transduction realized by the bimachine $\mathcal{B}$ is.

3.1 Bimachines

A bimachine is a model of computation, as expressive as (functional) transducers, introduced by Schützenberger in [20]. It is composed of two automata, an automaton reading words deterministically backwards, called a right automaton, and a classical deterministic automaton called here a left automaton. The right automaton acts as a deterministic look-ahead. An output function produces words based on the current symbol, and the states of the left and right automata.

More precisely, a *right automaton* $\mathcal{R} = (Q, I, F, \Delta)$ is an automaton such that I is a singleton, and transitions are backward deterministic: for any transitions (p_1, σ, q),(p_2, σ, q) $\in \Delta$ it holds that $p_1 = p_2$. The only difference with a (classical) automaton lies in the notion of accepting runs (and therefore in the notion of recognized language): a run r is accepting if $r[1]$ is final and $r[|r|]$ is initial. Therefore, a right automaton can be thought of as an automaton reading words backwards. We write $s_2 \xleftarrow{w}_{\mathcal{R}} s_1$ instead of $s_2 \xrightarrow{w}_{\mathcal{R}} s_1$ (with the same meaning) to emphasize that $\mathcal{R}$ is a right automaton, and graphically any transition $(q, \sigma, p) \in \Delta$ is depicted with an arrow from p to q. For instance, the accepting run on ba of the right automaton of the bimachine depicted in Section 1 is $r_b r_b r_0$. The *left transition congruence* $\sim_{\mathcal{R}}$ of $\mathcal{R}$ is defined by $u \sim_{\mathcal{R}} v \Leftrightarrow (\forall r \in R, r \xleftarrow{u}_{\mathcal{R}} r_0 \Leftrightarrow r \xleftarrow{v}_{\mathcal{R}} r_0)$. Implicitly assuming that $\mathcal{R}$ is complete, we identify $[u]_{\sim_{\mathcal{R}}}$ (also just denoted by $[u]_{\mathcal{R}}$) with the state $r \in R$ such that $r \xleftarrow{u}_{\mathcal{R}} r_0$. We say that $\mathcal{R}$ is finer than a right automaton $\mathcal{R}'$ (written $\mathcal{R} \sqsubseteq \mathcal{R}'$) if $\sim_{\mathcal{R}} \sqsubseteq \sim_{\mathcal{R}'}$. A *left automaton* is a deterministic automaton, called 'left' to emphasize its role in the context of bimachines.

A *bimachine* over an alphabet Σ is a tuple $\mathcal{B} = (\mathcal{L}, \mathcal{R}, \omega, \lambda, \rho)$ where $\mathcal{L} = (L, \{l_0\}, F_{\mathcal{L}}, \Delta_{\mathcal{L}})$ is a left automaton, $\mathcal{R} = (R, \{r_0\}, F_{\mathcal{R}}, \Delta_{\mathcal{R}})$ is a right automaton, $\omega : L \times \Sigma \times R \to \Sigma^*$ is the *output function*, $\lambda : F_{\mathcal{R}} \to \Sigma^*$ is the *left final function* and $\rho : F_{\mathcal{L}} \to \Sigma^*$ is the *right final function*. Both automata $\mathcal{L}$ and $\mathcal{R}$ must recognize the same language, *i.e.* $[\![\mathcal{L}]\!] = [\![\mathcal{R}]\!]$.

We now define the *transduction* $[\![\mathcal{B}]\!]$ *realized by* $\mathcal{B}$. First, we extend the function ω to $L \times \Sigma^* \times R$ as follows: for all states $r \in R$ and $l \in L$, all $u \in \Sigma^*$ and $\sigma \in \Sigma$, $\omega(l, \epsilon, r) = \epsilon$, and $\omega(l, u\sigma, r) = \omega(l, u, r')\omega(l', \sigma, r)$ where $l \xrightarrow{u}_{\mathcal{L}} l'$ and $r' \xleftarrow{\sigma}_{\mathcal{R}} r$. Now, the domain $\mathrm{dom}(\mathcal{B})$ of $\mathcal{B}$ is $[\![\mathcal{L}]\!] = [\![\mathcal{R}]\!]$. For all $u \in [\![\mathcal{L}]\!]$, if $l_0 \xrightarrow{u}_{\mathcal{L}} l$ for some $l \in F_{\mathcal{L}}$ and $r \xleftarrow{u}_{\mathcal{R}} r_0$ for some $r \in F_{\mathcal{R}}$, the image of u by $\mathcal{B}$ is defined by $[\![\mathcal{B}]\!](u) = \lambda(r)\omega(l_0, u, r_0)\rho(l)$.

3.2 Left and right bimachine minimization

Sequential transducers can be minimized by producing the outputs as early as possible [6]. No such procedure exist for transducers in general. For bimachines, however, a similar procedure is proposed in [17]. This one applies the "as early as possible" principle, but parameterized by the look-ahead information of the right automaton. We describe this minimization, exhibit some useful properties, and provide a PTIME minimization algorithm.

Let f be a rational transduction realized by a bimachine $\mathcal{B}$ whose right automaton is $\mathcal{R}$. Our goal here is to construct a bimachine $\mathit{Left}_f(\mathcal{B}) = (\mathit{Left}_f(\mathcal{R}), \mathcal{R}, \omega, \lambda, \rho)$, which realizes f and has the minimal left automaton among bimachines realizing f with $\mathcal{R}$ as its right automaton. We first give the construction of a minimal (wrt $\mathcal{R}$) left automaton

$\mathit{Left}_f(\mathcal{R})$ (or simply $\mathit{Left}(\mathcal{R})$ when it is clear from context). For simplicity, we will write $[w]_{\mathcal{R}}$ instead of $[w]_{\sim_{\mathcal{R}}}$ for any word $w \in \Sigma^*$. For any two words w and u, we define [3] $\widehat{f}_{[w]_{\mathcal{R}}}(u) = \wedge\{f(uv) \mid v \in [w]_{\mathcal{R}} \cap u^{-1}\mathrm{dom}(f)\}$.

This word is the longest possible output upon reading u, knowing that the suffix is in $[w]_{\mathcal{R}}$. The states of $\mathit{Left}_f(\mathcal{R})$ will be the classes of the right congruence $\sim_L$ defined by $u \sim_L v$ if for any letter σ, any $w, z \in \Sigma^*$ we have:

- $uz \in \mathrm{dom}(f) \Leftrightarrow vz \in \mathrm{dom}(f)$
- $\widehat{f}_{[\epsilon]_{\mathcal{R}}}(uz)^{-1}f(uz) = \widehat{f}_{[\epsilon]_{\mathcal{R}}}(vz)^{-1}f(vz)$, if $uz, vz \in \mathrm{dom}(f)$
- $\widehat{f}_{[\sigma w]_{\mathcal{R}}}(uz)^{-1}\widehat{f}_{[w]_{\mathcal{R}}}(uz\sigma) = \widehat{f}_{[\sigma w]_{\mathcal{R}}}(vz)^{-1}\widehat{f}_{[w]_{\mathcal{R}}}(vz\sigma)$

Intuitively, the second condition ensures that after reading uz and vz, $\mathit{Left}_f(\mathcal{R})$ outputs the same word by the final output function, and the third condition ensures that the output produced on σ is the same after reading uz and vz. From $\sim_L$ we define the automaton $\mathit{Left}_f(\mathcal{R}) = (\Sigma^*/_{\sim_L}, \{[\epsilon]_{\sim_L}\}, F, \Delta)$ where $F = \{[w]_{\sim_L} \mid w \in \mathrm{dom}(f)\}$ and $\Delta = \{([w]_{\sim_L}, \sigma, [w\sigma]_{\sim_L}) \mid \sigma \in \Sigma,\ w \in \Sigma^*\}$. Finally, the output functions are defined by: $\omega([u]_{\sim_L}, \sigma, [v]_{\mathcal{R}}) = \widehat{f}_{[\sigma v]_{\mathcal{R}}}(u)^{-1}\widehat{f}_{[v]_{\mathcal{R}}}(u\sigma)$, $\lambda([v]_{\mathcal{R}}) = \widehat{f}_{[v]_{\mathcal{R}}}(\epsilon)$ and $\rho([u]_{\sim_L}) = \widehat{f}_{[\epsilon]_{\mathcal{R}}}(u)^{-1}f(u)$.

Symmetrically one can define $\mathit{Right}_f(\mathcal{L})$ (and hence $\mathit{Right}_f(\mathcal{B})$).

The correctness of these constructions was shown in [17], *i.e.* $\mathit{Left}(\mathcal{B})$ and $\mathit{Right}(\mathcal{B})$ both realize f. The following proposition shows that $\mathit{Left}(\mathcal{R})$ and $\mathit{Right}(\mathcal{L})$ are minimal automata for which the bimachines (with fixed $\mathcal{R}$ and $\mathcal{L}$ respectively) realize f.[4]

▶ **Proposition 2.** *If $\mathcal{B} = (\mathcal{L}, \mathcal{R}, \omega, \lambda, \rho)$ is a bimachine, then $\mathcal{L} \sqsubseteq \mathit{Left}(\mathcal{R})$ and $\mathcal{R} \sqsubseteq \mathit{Right}(\mathcal{L})$.*

One contribution of this paper is to show that the left automaton can be minimized in PTime (for a fixed right automaton), and symmetrically for the right automaton.

▶ **Theorem 3.** *Let $\mathcal{B}$ be a bimachine. One can compute $\mathit{Left}(\mathcal{B})$ (and $\mathit{Right}(\mathcal{B})$) in* PTime.

Proof. Let $\mathcal{B} = (\mathcal{L}, \mathcal{R}, \omega, \lambda, \rho)$ be a bimachine realizing a function f with automata $\mathcal{L} = (Q_{\mathcal{L}}, l_0, F_{\mathcal{L}}, \Delta_{\mathcal{L}})$ and $\mathcal{R} = (Q_{\mathcal{R}}, r_0, F_{\mathcal{R}}, \Delta_{\mathcal{R}})$. W.l.o.g. we assume that $\mathcal{L}$ is complete (otherwise we complete it in polynomial time). The algorithm works in two steps: (i) make the output production earliest, (ii) run state minimization on the left automaton.

Step 1: making the bimachine earliest. We construct $\mathcal{B}' = (\mathcal{L}, \mathcal{R}, \omega', \lambda', \rho')$ a bimachine realizing f with the same automata, but with the earliest leftmost possible outputs:

$$\omega'([u]_{\mathcal{L}}, \sigma, [v]_{\mathcal{R}}) = \widehat{f}_{[\sigma v]_{\mathcal{R}}}(u)^{-1}\widehat{f}_{[v]_{\mathcal{R}}}(u\sigma) \quad \rho'([u]_{\mathcal{L}}) = \widehat{f}_{[\epsilon]_{\mathcal{R}}}(u)^{-1}f(u) \quad \lambda'([v]_{\mathcal{R}}) = \widehat{f}_{[v]_{\mathcal{R}}}(\epsilon).$$

These functions are well-defined as they do not depend on the choice of the representatives u (see the proof of Proposition 2). We have to show that these output values can be computed in polynomial time. The algorithm is very close to the ones described in [6], for sequential functions, which is why we only give the main ideas. We first remark, as in the proof of Proposition 2, that for any words u, v, $\widehat{f}_{[v]_{\mathcal{R}}}(u) = \lambda([uv]_{\mathcal{R}})\omega([\epsilon]_{\mathcal{L}}, u, [v]_{\mathcal{R}})\beta([u]_{\mathcal{L}}, [v]_{\mathcal{R}})$ with:

$$\beta([u]_{\mathcal{L}}, [v]_{\mathcal{R}}) = \bigwedge \{w \mid \exists x \in [v]_{\mathcal{R}} \cap u^{-1}\mathrm{dom}(f), \omega([u]_{\mathcal{L}}, x, [\epsilon]_{\mathcal{R}})\rho([ux]_{\mathcal{L}}) = w\}$$

As in [6], to compute the values $\beta([u]_{\mathcal{L}}, [v]_{\mathcal{R}})$ we can take the directed graph of the automaton $\mathcal{L} \times \mathcal{R}$ with the outputs of ω labelling the edges. In order to account for the functions λ and ρ we add two vertices, a source s pointing to the initial states with the edges labelled

[3] As a convention, the longest common prefix of an empty set of words is the empty word, that is: $\wedge\emptyset = \epsilon$.
[4] It was already shown in [17] for total functions, we extend it with a similar proof to our setting, that is, when the function is not total, and the automata of the bimachine both recognize $\mathrm{dom}(f)$.

accordingly by the values of the initial function, and a target vertex t with edges pointing to it from the final states with the edges labelled by the final function. The value of $\beta([u]_{\mathcal{L}}, [v]_{\mathcal{R}})$ can be seen as the longest common prefix of the labels of all the paths starting in $([u]_{\mathcal{L}}, [v]_{\mathcal{R}})$ and ending in the target vertex t. These values can be computed in polynomial time [6].

Step 2: minimizing the left automaton. Now we describe a minimization algorithm inspired by Moore's minimization algorithm for DFA. It consists in computing successively finer equivalence relations $\sim_0, \sim_1, \ldots$ over the states of $\mathcal{L}$. The only difference is in the initial partition, for which we have to make sure that outputs are compatible. More precisely, for all $l, l' \in Q_{\mathcal{L}}$ and $i \geq 0$, we let:

$$\begin{array}{lll} l \sim_0 l' & \text{if} & l \in F_{\mathcal{L}} \Leftrightarrow l' \in F_{\mathcal{L}},\ \rho'(l) = \rho'(l'),\ \text{and } \forall r, \sigma,\ \omega'(l, \sigma, r) = \omega'(l', \sigma, r) \\ l \sim_{i+1} l' & \text{if} & l \sim_i l' \text{ and } \forall \sigma,\ l.\sigma \sim_i l'.\sigma \end{array}$$

where $l.\sigma$ denotes the state reached by $\mathcal{L}$ after reading the letter σ from l. We extend this notation to words in the natural way: $l.u$ is the state such that $l \xrightarrow{u}_{\mathcal{L}} l.u$ (it is unique as $\mathcal{L}$ is deterministic and complete).

Since $\sim_{i+1} \sqsubseteq \sim_i$ for all $i \geq 0$, this sequence converges after at most $|Q_{\mathcal{L}}|$ steps to an equivalence relation that we denote by $\sim_*$. Moreover, $\sim_0$ can be computed in PTime from $\mathcal{B}'$, and each $\sim_i$ can be computed in PTime from $\sim_{i-1}$, for $i > 0$.

We extend the relations $\sim_i$ to Σ^* as follows: $u \sim_i v$ if $l_0.u \sim_i l_0.v$. We show in the long version of this paper that $\sim_* = \sim_L$ (remind that $\sim_L$ is the right congruence associated with $\mathcal{R}$ and used to define *Left*$(\mathcal{R})$). To give an idea of the proof, we first show by induction on $i \geq 0$ that for all $u, v \in \Sigma^*$, $u \sim_i v$ iff for all $z \in \Sigma^i$, all $w \in \Sigma^*$ and all $\sigma \in \Sigma$, we have (i) $uz \in \mathrm{dom}(f)$ iff $vz \in \mathrm{dom}(f)$, (ii) $\rho'([uz]_{\mathcal{L}}) = \rho'([vz]_{\mathcal{L}})$ (if $uz, vz \in \mathrm{dom}(f)$), and (iii) $\omega'([uz]_{\mathcal{L}}, \sigma, [w]_{\mathcal{R}}) = \omega'([vz]_{\mathcal{L}}, \sigma, [w]_{\mathcal{R}})$. This implies that $u \sim_* v$ iff the properties (i)–(iii) holds for all $z \in \Sigma^*$. Finally, we conclude by noticing that $\rho'([uz]_{\mathcal{L}}) = \widehat{f}_{[\epsilon]_{\mathcal{R}}}(uz)^{-1} f(uz)$, and $\omega'([uz]_{\mathcal{L}}, \sigma, [w]_{\mathcal{R}}) = \widehat{f}_{[\sigma w]_{\mathcal{R}}}(uz)^{-1} \widehat{f}_{[w]_{\mathcal{R}}}(uz\sigma)$.

Clearly, if $l \sim_* l'$, then for all $\sigma \in \Sigma$, $l.\sigma \sim_* l'.\sigma$. Moreover, if $l \in F_{\mathcal{L}}$, then since $\sim_* \sqsubseteq \sim_0$, we also get $l' \in F_{\mathcal{L}}$, and conversely. Therefore one can define the left automaton $\mathcal{L}/_{\sim_*} = (Q_{\mathcal{L}}/_{\sim_*}, F_{\mathcal{L}}/_{\sim_*}, [l_0]_{\sim_*}, \delta)$ where $\delta([l]_{\sim_*}, \sigma) = [l.\sigma]_{\sim_*}$, and we have $[\![\mathcal{L}_{\sim_*}]\!] = [\![\mathcal{L}]\!]$.

Finally, $\sim_* = \sim_L$ implies that the bimachine *Left*$_f(\mathcal{B})$ is isomorphic to the bimachine $(\mathcal{L}/_{\sim_*}, \mathcal{R}, \omega'', \lambda', \rho'')$ where $\omega''([u]_{\sim_*}, \sigma, [v]_{\mathcal{R}}) = \omega'([u]_{\mathcal{L}}, \sigma, [v]_{\mathcal{R}})$ and $\rho''([u]_{\sim_*}) = \rho'([u]_{\mathcal{L}})$. Note that these output functions are well-defined since, by definition, $\sim_*$ is compatible with the output functions λ', ω', ρ'. Moreover, $\mathcal{L}/_{\sim_*}$ can be computed in PTime since $\sim_*$ can be computed in PTime, and the output functions ω'', ρ'' can as well be computed in PTime, which concludes the proof. A last remark is that a Hopcroft-like minimization algorithm [15], initialized with $\sim_0$ as well, would be more efficient, but with a more involved proof. ◀

3.3 A minimal and canonical bimachine

For a given function f and two bimachines $\mathcal{B}_1$ and $\mathcal{B}_2$ realizing it, we say that $\mathcal{B}_1$ is *finer* than $\mathcal{B}_2$, denoted again by $\mathcal{B}_1 \sqsubseteq \mathcal{B}_2$, if we have both $\mathcal{L}_1 \sqsubseteq \mathcal{L}_2$ and $\mathcal{R}_1 \sqsubseteq \mathcal{R}_2$, with $\mathcal{L}_i$ and $\mathcal{R}_i$ being the left and the right automata of bimachine $\mathcal{B}_i$, respectively. We say that a bimachine is *minimal* if there is no coarser bimachine realizing the same transduction. There is no unique minimal bimachine in general [17]. In this section, we explain the construction of a minimal and canonical bimachine associated with a rational transduction, the main result of [17]. It relies on (a) a canonical right automaton, that we detail hereafter, and (b) the construction of a minimal left automaton from a right automaton, as described in Section 3.2.

The construction of a *canonical right automaton* is based on a left congruence associated with a function that measures the effect of suffixes on the translation of prefixes. The *left congruence* of a transduction f on Σ is defined $\forall u, v \in \Sigma^*$ by $u \leftharpoonup_f v$ if:

- $\forall w \in \Sigma^*$, $wu \in \mathrm{dom}(f) \Leftrightarrow wv \in \mathrm{dom}(f)$ and
- $\sup\{\|f(wu), f(wv)\| \mid wu, wv \in \mathrm{dom}(f)\} < \infty$

This congruence has finite index if f is rational [17]. The converse does not hold but if additionally f^{-1} preserves language rationality, then f is rational. For the rest of this section $[w]$ denotes the class of w in $\Sigma^*/_{\leftharpoonup_f}$. The *canonical right automaton* for f is $\mathcal{R}_f = (\Sigma^*/_{\leftharpoonup_f}, \{[\epsilon]\}, F, \Delta)$ where $F = \{[w] \mid w \in \mathrm{dom}(f)\}$ and $\Delta = \{([\sigma w], \sigma, [w]) \mid \sigma \in \Sigma,\ w \in \Sigma^*\}$.

▶ Remark. We can define symmetrically the *right congruence* of f by $u \rightharpoonup_f v$ if $\forall w$, $uw \in \mathrm{dom}(f) \Leftrightarrow vw \in \mathrm{dom}(f)$ and $\sup\{\|f(uw), f(vw)\| \mid uw, vw \in \mathrm{dom}(f)\} < \infty$ and based on this right congruence, define the *canonical left automaton* $\mathcal{L}_f$.

The automata $\mathcal{R}_f$ and $\mathcal{L}_f$ are coarser than any right (resp. left) automaton of a bimachine realizing f. This was shown in [13] but only for the case of total functions. The proof is similar in the general case and we give it in the long version of this paper for completeness.

▶ **Proposition 4.** *Let f be a transduction, and let $\mathcal{B} = (\mathcal{L}, \mathcal{R}, \omega, \lambda, \rho)$ be a bimachine realizing f. Then $\mathcal{L} \sqsubseteq \mathcal{L}_f$ and $\mathcal{R} \sqsubseteq \mathcal{R}_f$.*

The *canonical bimachine* [17] associated with a rational transduction f is the bimachine $\mathcal{B}_f = (\mathit{Left}_f(\mathcal{R}_f), \mathcal{R}_f, \omega_f, \lambda_f, \rho_f)$ where:

$$\begin{array}{rcl} \omega_f([u]_{\sim_L}, \sigma, [v]_{\mathcal{R}_f}) & = & \widehat{f}_{[\sigma v]\mathcal{R}_f}(u)^{-1} \widehat{f}_{[v]\mathcal{R}_f}(u\sigma) \\ \lambda_f([v]_{\mathcal{R}_f}) & = & \widehat{f}_{[v]\mathcal{R}_f}(\epsilon) \\ \rho_f([u]_{\sim_L}) & = & \widehat{f}_{[\epsilon]\mathcal{R}_f}(u)^{-1} f(u) \end{array}$$

By its definition, the bimachine $\mathcal{B}_f$ is canonical, *i.e.* does not depend on any description of f. It is also minimal: indeed, suppose that f is realized by a bimachine $\mathcal{B} = (\mathcal{L}, \mathcal{R}, \omega, \lambda, \rho)$ such that $\mathcal{B}_f \sqsubseteq \mathcal{B}$, *i.e.* $\mathit{Left}(\mathcal{R}_f) \sqsubseteq \mathcal{L}$ and $\mathcal{R}_f \sqsubseteq \mathcal{R}$. Then ω (and similarly λ, ρ) can be restricted to $\omega'([u]_{\mathcal{L}}, \sigma, [v]_{\mathcal{R}_f}) = \omega([u]_{\mathcal{L}}, \sigma, [v]_{\mathcal{R}})$, which is well-defined since $\mathcal{R}_f \sqsubseteq \mathcal{R}$, so that the bimachine $(\mathcal{L}, \mathcal{R}_f, \omega', \lambda', \rho')$ realizes f. By Proposition 2 we get $\mathcal{L} \sqsubseteq \mathit{Left}(\mathcal{R}_f)$. Moreover, by Proposition 4, we also have $\mathcal{R} \sqsubseteq \mathcal{R}_f$.

Finally, $\mathcal{B}_f$ is computable when f is given by a bimachine or a transducer [17].

4 Characterization of aperiodic transductions

In this section we show that to decide if a transduction given by a bimachine $\mathcal{B}$ is aperiodic, one only needs to minimize $\mathcal{B}$, *i.e.* to construct $\mathit{Left}(\mathit{Right}(\mathcal{B}))$, which yields a minimal bimachine, and check its aperiodicity (Section 4.2). To prove the correctness of this procedure, we rely on the following characterization proved in Section 4.1: a transduction f is aperiodic if and only if the canonical bimachine $\mathcal{B}_f$ is aperiodic. This is in contrast with other varieties for which the canonical bimachine does not preserve membership in general [13]. The latter characterization does not yield an optimal algorithm since the canonical bimachine may be exponentially larger than the initial bimachine.

4.1 Characterization through canonical bimachine

In this section, given a rational transduction f over some alphabet Σ, we show that it is aperiodic iff the canonical bimachine $\mathcal{B}_f$ associated with f, defined in the previous section, is aperiodic. It relies on two important facts: (i) the left congruence $\leftharpoondown_f$ of an aperiodic transduction is aperiodic (Proposition 5), (ii) any aperiodic rational transduction f can be decomposed into $f = \ell \circ label_{\mathcal{R}_f}$ such that ℓ is realized by some aperiodic sequential transducer, and $label_{\mathcal{R}_f}$ annotates every input position i with the class of the suffix from i by $\leftharpoondown_f$ (Proposition 6). From this decomposition, one can construct an aperiodic bimachine whenever f is aperiodic. First, one shows that $\mathcal{R}_f$ is aperiodic when f is too:

▶ **Proposition 5.** *Let f be a transduction realizable by an aperiodic transducer then the congruence $\leftharpoondown_f$, and so the automaton $\mathcal{R}_f$, are aperiodic.*

A right-sequential transducer is a transducer whose underlying input automaton is a right automaton. A right-sequential transduction is a function realized by a right-sequential transducer. A bimachine can be seen as the composition of a right-sequential transduction annotating the word with states of the right automaton, and a (left-) sequential transduction obtained from the left automaton and the output function ω. We show that any aperiodic rational transduction f can be decomposed into $\ell \circ label_{\mathcal{R}_f}$ such that ℓ can be realized by a sequential aperiodic transducer, and $label_{\mathcal{R}_f}$ annotates the input word with states of $\mathcal{R}_f$.

More precisely, let $\mathcal{R}$ be a right automaton over Σ with set of states Q, and let $\Sigma_{\mathcal{R}} = \{\sigma_q \mid \sigma \in \Sigma, q \in Q\}$. We define the rational transduction $label_{\mathcal{R}} : \Sigma^* \to \Sigma_{\mathcal{R}}^*$, called the labelling function of $\mathcal{R}$, which labels words in Σ^* by states of $\mathcal{R}$. It is defined by the right-sequential transducer $\mathcal{T} = (\mathcal{R}, o, \bar{\epsilon}, \bar{\epsilon})$ where $\bar{\epsilon}$ denotes the constant function which maps any element to ϵ, and with $o(p, \sigma, q) = \sigma_q$.

▶ **Proposition 6.** *Let f be an aperiodic rational transduction. There exists a transduction ℓ such that $f = \ell \circ label_{\mathcal{R}_f}$ and ℓ is realized by a sequential aperiodic transducer.*

Proof. We first show that there exists a *sequential* transduction ℓ such that $f = \ell \circ label_{\mathcal{R}_f}$. This sequential transduction is realized by the left automaton of the canonical bimachine $\mathcal{B}_f$ combined with the output function ω_f. Then, we show that ℓ is *aperiodic* rational, by constructing an aperiodic transducer realizing it, obtained by taking the product of any aperiodic transducer realizing f (which exists by assumption) and $\mathcal{R}_f$, and by ensuring that the information $[u]$ occurring on symbols $\sigma_{[u]}$ is consistent with the information $[u]$ occurring on the states of the product, for all words u. Finally, any aperiodic and sequential transduction can be realized by a transducer which is both sequential *and* aperiodic (*i.e.* sequentialization preserves aperiodicity [13]). Details can be found in the long version. ◀

We can now show our characterization of aperiodic rational transductions:

▶ **Theorem 7.** *A rational function f is aperiodic iff its canonical bimachine is aperiodic.*

Proof. It is known that any bimachine can be transformed into an equivalent (unambiguous) transducer whose underlying automaton is the product of the left and the right automata [13, 17]. Roughly, the transducer has to guess the state of the right automaton, and unambiguity is implied by the fact that the transitions of the right automaton are backward deterministic. The product of two aperiodic automata being aperiodic, this shows the 'if' direction.

We now show the 'only if' direction. By Proposition 5, f can be decomposed into $f = \ell \circ label_{\mathcal{R}_f}$ such that ℓ is realized by some aperiodic sequential transducer $\mathcal{T}_m = (\mathcal{A}_m, o_m, i_m, t_m)$. Based on this decomposition we construct an aperiodic bimachine $\mathcal{B} = (\mathcal{D}, \mathcal{R}_f, \omega, \lambda, \rho)$ realizing f. This will allow to conclude. Indeed, by Proposition 2 we have $\mathcal{D} \sqsubseteq Left(\mathcal{R}_f)$, and since $\mathcal{D}$ is aperiodic, so is $Left(\mathcal{R}_f)$. Since $\mathcal{R}_f$ is aperiodic as well by Proposition 5, it implies that $\mathcal{B}_f$ is aperiodic. Let us now construct $\mathcal{D}$, ω, λ and ρ.

Recall that the input alphabet of $\mathcal{T}_m$ (and $\mathcal{A}_m$) is Σ_Q where $Q = \Sigma^*/_{\leftharpoondown_f}$. To construct $\mathcal{B}$, it is tempting to think that it suffices to take the projection of $\mathcal{A}_m$ on Σ as left automaton, $\mathcal{R}_f$ as right automaton, and to define the output function ω by $\omega(p, \sigma, [u]) = w$ where $p \xrightarrow{\sigma_{[u]}|w} q$ is the transition of $\mathcal{T}_m$ on $\sigma_{[u]}$ with output w. The problem is that the projection of $\mathcal{A}_m$ on Σ is not a deterministic automaton in general, and by determinizing it, one loses the information of which transition of $\mathcal{T}_m$ should be applied. We propose a solution that overcomes this issue, by integrating the information of $\mathcal{R}_f$ in the state of the left automaton. Let us detail this construction. We take $\mathcal{T}_m$ and project input letters to their Σ component, hence we obtain a transducer realizing f which is unambiguous, since $\mathcal{R}_f$ has backward deterministic transitions. Let $\tilde{\mathcal{T}}_m$ denote the obtained transducer, and $\tilde{\mathcal{A}}_m$ its underlying (unambiguous) automaton. We let $\mathcal{D}$ be the automaton obtained by determinization, by subset construction, of the product automaton $\tilde{\mathcal{A}}_m \times \mathcal{R}_f$. States of $\mathcal{D}$ are therefore of the form $2^{Q_m \times \Sigma^*/_{\leftharpoondown_f}}$. The output function ω is defined by:

$$\omega(\{(p_1, [u_1]), \ldots, (p_n, [u_n])\}, \sigma, [v]) = o_m(p_i, \sigma)$$

such that $[\sigma v] = [u_i]$. The state p_i is unique since $\mathcal{T}_m$ is unambiguous. Indeed, let us assume by contradiction that there are two distinct such states p_i, p_j. This would mean that $[u_i] = [u_j]$ and since $\mathcal{R}_f$ has backward deterministic transitions, for a word w which reaches both p_i and p_j in $\tilde{\mathcal{A}}_m$, we have a labelled word z such that $z[k] = w[k]_c$ for $k \in \{1, \ldots, |w|\}$ and c the class of the word $w[k+1] \ldots w[|w|]u_i$. Thus we obtain $q_0 \xrightarrow{z}_{\mathcal{A}_m} p_i$ and $q_0 \xrightarrow{z}_{\mathcal{A}_m} p_j$ which is in contradiction with the deterministic nature of $\mathcal{A}_m$. We define the final output functions naturally: $\lambda([u]) = i_m(q_{o,m})$ and $\rho(\{(p_1, [u_1]), \ldots, (p_n, [u_n])\}) = t(p_i)$ such that $[u_i] = [\epsilon]$ (again, it is unique by unambiguity of $\mathcal{T}_m$).

It remains to show that $\mathcal{B}$ is aperiodic. Aperiodicity of $\mathcal{R}_f$ is obtained by Proposition 5. Aperiodicity of $\mathcal{D}$ is shown in the long version of this paper, as a consequence of the aperiodicity of $\mathcal{T}_m$, $\mathcal{R}_f$ and the fact that subset construction preserves aperiodicity. ◀

▶ Remark. This theorem gives an algorithm to decide aperiodicity of a rational function: computing the canonical bimachine and checking that it is aperiodic. However, computing the left automaton *Left*$(\mathcal{R}_f)$ may cause an exponential blow-up. Consider for example, the transduction $f : \Sigma^* \to \Sigma^*$ defined for $w \in \Sigma^*$, $w_n \in \Sigma^n$ by $f(ww_n) = w_n$ and for $|w| < n$ by $f(w) = w$. Since the distance between the image of two words is bounded by $2n$, the left congruence of f is trivial, so the canonical bimachine of f is just a sequential transducer and needs $O(\Sigma^n)$ states to remember the last n letters of an input word. However this transduction can be realized by a right-sequential transducer with only n states, but one could define a symmetrical example ($f(w_n w) = w_n$) where the bimachine obtained from the canonical left automaton $\mathcal{L}_f$ witnesses an exponential blow-up.

Another consequence is that any aperiodic transduction admits an aperiodic *unambiguous* transducer realizing it. This problem is open for arbitrary varieties.

▶ **Corollary 8.** *Every aperiodic transduction can be realized by an unambiguous aperiodic transducer.*

Proof. As explained in the proof of the ‘if’ direction of Theorem 7, from any bimachine one can construct an equivalent unambiguous transducer by taking the product of the left and right automata of the bimachine, which is aperiodic if the bimachine is aperiodic too. ◀

4.2 Characterization through bimachine minimization and main result

In this section, we prove the main result of this paper, *i.e.* that aperiodicity is PSPACE-complete for functions realized by bimachines. We show that for a bimachine $\mathcal{B}$ it suffices to construct $\mathit{Left}(\mathit{Right}(\mathcal{B}))$ (or $\mathit{Right}(\mathit{Left}(\mathcal{B}))$) and to check aperiodicity of the resulting bimachine. The first step can be done in PTIME and the second step in PSPACE. The operations $\mathit{Left}(\mathit{Right}(\mathcal{B}))$ and $\mathit{Right}(\mathit{Left}(\mathcal{B}))$ are called *bimachine minimization*, as they indeed yield minimal bimachines, as shown by the following proposition:

▶ **Proposition 9.** *Let $\mathcal{B}$ be a bimachine realizing a transduction f. Then $\mathit{Left}(\mathit{Right}(\mathcal{B}))$ and $\mathit{Right}(\mathit{Left}(\mathcal{B}))$ are minimal bimachines realizing f.*

Proof. Based on successive applications of Proposition 2 and given in the long version. ◀

The following result is a key towards the main contribution:

▶ **Proposition 10.** *Let f be a transduction realized by a bimachine $\mathcal{B} = (\mathcal{L}, \mathcal{R}, \omega, \lambda, \rho)$. Then f is aperiodic iff $\mathit{Left}(\mathit{Right}(\mathcal{B}))$ is aperiodic iff $\mathit{Right}(\mathit{Left}(\mathcal{B}))$ is aperiodic.*

Proof. First, we start by some observation: if there are two bimachines realizing f with $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$ as right automata, then $\mathit{Left}_f(\mathcal{A}_2) \sqsubseteq \mathit{Left}_f(\mathcal{A}_1)$. Indeed, if $\mathcal{A}_1$ provides more (*i.e.* finer) information than $\mathcal{A}_2$ on suffixes, then the two equalities of the definition of the right congruence used to define $\mathit{Left}_f(\mathcal{A}_1)$ (see Section 3.2) are "easier" to satisfy since the set of suffixes v taken into account in the definition of $\hat{f}_{[w]_{\mathcal{A}_1}}$ is included in the set of suffixes used in the definition of $\hat{f}_{[w]_{\mathcal{A}_2}}$ for all words w. Symmetrically, if there are two bimachines realizing f with $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$ as left automata, then $\mathit{Right}_f(\mathcal{A}_2) \sqsubseteq \mathit{Right}_f(\mathcal{A}_1)$.

By Proposition 4 we have $\mathcal{L} \sqsubseteq \mathcal{L}_f$ and $\mathcal{R} \sqsubseteq \mathcal{R}_f$, but also $\mathit{Right}(\mathcal{L}) \sqsubseteq \mathcal{R}_f$ since Proposition 4 holds for any bimachine realizing f, and there is a bimachine realizing f with $\mathit{Right}(\mathcal{L})$ as right automaton (the bimachine $\mathit{Right}(\mathcal{B})$). Therefore by the observation, we get $\mathit{Right}(\mathcal{L}_f) \sqsubseteq \mathit{Right}(\mathcal{L})$ and $\mathit{Left}(\mathcal{R}_f) \sqsubseteq \mathit{Left}(\mathit{Right}(\mathcal{L}))$.

By Theorem 7, if f is aperiodic, then $\mathcal{B}_f = (\mathit{Left}(\mathcal{R}_f), \mathcal{R}_f, \omega, \lambda, \rho)$ is aperiodic. Therefore, $\mathit{Left}(\mathit{Right}(\mathcal{L}))$ is aperiodic. Symmetrically, exactly as shown in Theorem 7, it can be shown that $\mathcal{L}_f$ and $\mathit{Right}(\mathcal{L}_f)$ are aperiodic if f is aperiodic, which implies that $\mathit{Right}(\mathcal{L})$ is aperiodic. In conclusion, $\mathit{Left}(\mathit{Right}(\mathcal{B}))$ is aperiodic. ◀

We can now prove the main result of this paper:

▶ **Theorem 11.** *The problem of deciding whether a bimachine $\mathcal{B}$ realizes an aperiodic rational transduction is* PSPACE*-complete.*

Proof. To get the upper-bound, by Proposition 10, it suffices to (i) construct $\mathit{Right}(\mathcal{B})$, (ii) construct $\mathit{Left}(\mathit{Right}(\mathcal{B}))$, and (iii) test whether the left and right automata of the bimachine $\mathit{Left}(\mathit{Right}(\mathcal{B}))$ are aperiodic.

By Theorem 3, steps (i) and (ii) can be done in PTIME, while step (iii) can be done in PSPACE by [5].

The lower bound is obtained from the problem of deciding whether the transition congruence of a minimal deterministic finite automaton is aperiodic, which is PSPACE-hard [5]. The details can be found in the long version of this paper. ◀

5 Perspectives

In [8] it is proved that deciding whether a regular language given by a *non-deterministic* automaton is aperiodic is also PSPACE-complete. As a future work, we want to obtain tight complexity for the following problem: given a (non-deterministic) transducer, does it define an aperiodic transduction? Based on the techniques of this paper, the latter problem could be shown to be in 2EXPTIME, since obtaining the canonical bimachine causes two exponential blow-ups: one for the canonical right automaton and one for the determinization of the transducer over the enriched alphabet. It is however yet unclear whether the techniques of [8] can be combined with the ones of this paper to lower this upper bound.

References

1 Marie-Pierre Béal, Olivier Carton, Christophe Prieur, and Jacques Sakarovitch. Squaring transducers: an efficient procedure for deciding functionality and sequentiality. *Theor. Comput. Sci.*, 292(1):45–63, 2003.

2 Jean Berstel and Luc Boasson. Transductions and context-free languages. *Ed. Teubner*, pages 1–278, 1979.

3 Mikołaj Bojańczyk. Transducers with origin information. In *International Colloquium on Automata, Languages, and Programming (ICALP), Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 26–37. Springer, 2014. doi:10.1007/978-3-662-43951-7_3.

4 Olivier Carton and Luc Dartois. Aperiodic two-way transducers and fo-transductions. In *EACSL Annual Conference on Computer Science Logic (CSL)*, volume 41 of *LIPIcs*, pages 160–174. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.CSL.2015.160.

5 Sang Cho and Dung T. Huynh. Finite-automaton aperiodicity is PSPACE-complete. *Theor. Comput. Sci.*, 88(1):99–116, 1991. doi:10.1016/0304-3975(91)90075-D.

6 Christian Choffrut. Minimizing subsequential transducers: a survey. *Theor. Comput. Sci.*, 292(1):131–143, 2003. doi:10.1016/S0304-3975(01)00219-5.

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

8 Volker Diekert and Paul Gastin. First-order definable languages. In *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*, volume 2 of *Texts in Logic and Games*, pages 261–306. Amsterdam University Press, 2008.

9 Samuel Eilenberg. *Automata, Languages, and Machines. Volume A*. Pure and Applied Mathematics. Academic press, 1974.

10 Calvin C. Elgot and Jorge E. Mezei. On relations defined by generalized finite automata. *IBM Journal of Research and Development*, 9(1):47–68, 1965. doi:10.1147/rd.91.0047.

11 Joost Engelfriet and Hendrik Jan Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Log.*, 2(2):216–254, 2001. doi:10.1145/371316.371512.

12 Emmanuel Filiot. Logic-automata connections for transformations. In *Indian Conference on Logic and Its Applications (ICLA)*, volume 8923 of *Lecture Notes in Computer Science*, pages 30–57. Springer, 2015. doi:10.1007/978-3-662-45824-2_3.

13 Emmanuel Filiot, Olivier Gauwin, and Nathan Lhote. First-order definability of rational transductions: An algebraic approach. In *Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE Computer Society, 2016.

14 Emmanuel Filiot, Shankara Narayanan Krishna, and Ashutosh Trivedi. First-order definable string transformations. In *International Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 29 of *LIPIcs*, pages 147–159.

Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2014. `doi:10.4230/LIPIcs.FSTTCS.2014.147`.

15 John E. Hopcroft. An $N \log N$ Algorithm for Minimizing States in a Finite Automaton. Technical report, Stanford University, Stanford, CA, USA, 1971.

16 Robert McNaughton and Seymour Papert. *Counter-free automata*. M.I.T. Press, 1971.

17 Christophe Reutenauer and Marcel-Paul Schützenberger. Minimization of rational word functions. *SIAM J. Comput.*, 20(4):669–685, 1991. `doi:10.1137/0220042`.

18 Christophe Reutenauer and Marcel-Paul Schützenberger. Variétés et fonctions rationnelles. *Theor. Comput. Sci.*, 145(1&2):229–240, 1995. `doi:10.1016/0304-3975(94)00180-Q`.

19 Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

20 Marcel-Paul Schützenberger. A remark on finite transducers. *Information and Control*, 4(2-3):185–196, 1961.

21 Marcel-Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965.

22 Jacques Stern. Complexity of some problems from the theory of automata. *Information and Control*, 66(3):163–176, 1985. `doi:10.1016/S0019-9958(85)80058-9`.

23 Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhauser Verlag, Basel, Switzerland, Switzerland, 1994.