Aperiodicity of rational functions is PSpace-complete
Emmanuel Filiot, Olivier Gauwin, Nathan Lhote

To cite this version:
Aperiodicity of rational functions is PSpace-complete

Emmanuel Filiot¹, Olivier Gauwin², and Nathan Lhote¹,²

¹ Université Libre de Bruxelles
² Université de Bordeaux, LaBRI, CNRS

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.2016.0

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 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
Aperiodicity of rational functions is PSpace-complete

terms, this amounts to decide whether a given automaton $A$ 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 $A$ is equivalent to an aperiodic one, and this problem is known to be PSPACE-complete when $A$ 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 $A$ into the unique minimal automaton $A_m$ (which is an effective representation of the right syntactic congruence of $L(A)$) and (2) decide whether $A_m$ is aperiodic. It is well-known that step (1) is in PTIME since $A$ 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 $\sigma, \beta$ 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 $\beta$, 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 $L$ and $R$, and some output function $\omega$, and works as follows: $R$ processes an input word $w$ from right to left and annotates it with its states. Symmetrically, $L$ processes $w$ from left to right and annotates it with its states. Finally, the output function $\omega$ is applied to any triple of the form $(r, \sigma, l)$ of the annotated word, where $r$ is a state of $R$, $\sigma$ is an input symbol of $w$, and $l$ is a state of $L$. 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 $\omega$ maps any triple of the form $(l_0, \sigma, r_\beta)$ or $(l_\beta, \sigma, r_0)$ to $\beta$, and any other triple $(l, \sigma, r)$ to $\sigma$, 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 ($\text{PSpace}$-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 V. It is shown that a transduction is in V iff one the minimal bimachines is in 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 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 V) if it is realized by an unambiguous aperiodic transducer (or an unambiguous V-transducer). This definition was motivated by the fact that unambiguous transducers already capture all rational functions. For a general variety V, this left open the problem of whether any transduction realized by a V-transducer is realizable by an unambiguous 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 \( \Sigma \) is a finite set of symbols, and a word over \( \Sigma \) is an element of the free monoid \( \Sigma^* \) whose neutral element is denoted by \( \epsilon \). For \( w \in \Sigma^* \), we write \( |w| \) for its length and in particular, \( |\epsilon| = 0 \). For a non-empty word \( w \) and \( i \in \{1, \ldots, |w|\} \), we denote by \( w[i] \) the \( i \)th 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 \( u \land v \) 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 \( \Sigma \) is a tuple \( A = (Q, I, F, \Delta) \) 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 \( A = (Q, I, F, \Delta) \) on a word \( w \in \Sigma^* \) of length \( n \) is a word \( r = q_0 \ldots q_n \) over \( Q \) such that \( (q_i, w[i+1], q_{i+1}) \in \Delta \) for all \( i \in \{0, \ldots, n-1\} \). The run \( r \) is accepting if \( q_0 \in I \) and \( q_n \in F \). A word is accepted by \( A \) if there exists an accepting run of \( A \) over it. The language recognized by \( A \) is the set \( \{u \in \Sigma^* | w \sim_A u \} \) of words accepted by \( A \). We write \( p \xrightarrow{w} q \) (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 \( A \) 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 \( A \) 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 \( \Sigma \) be an alphabet and let \( \sim \) be an equivalence relation on \( \Sigma^* \). 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 \([u]_\sim\) if \( \sim \) is clear from the context), and \( \Sigma^*/\sim = \{[u]_\sim | u \in \Sigma^* \} \) is called the quotient of \( \Sigma^* \) by \( \sim \). We say that \( \sim \) has finite index if \( \Sigma^*/\sim \) is finite. Concatenation naturally extends to congruence classes as follows: for all \( u, v \in \Sigma^* \), \([uv]_\sim = [uv]_\sim\). Since \( \sim \) is a congruence, the latter is well-defined. With this operation, \( \Sigma^*/\sim \) forms a monoid whose neutral element is \( [e]_\sim \).

**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 \( \approx_A \) of an automaton \( A \) with set of states \( Q \) and if \( A \) is deterministic with initial state \( q_0 \), the right transition congruence \( \sim_A \). They are defined as follows, for any two words \( u, v \in \Sigma^* \)

\[
\begin{align*}
\equiv_L & : u \equiv_L v \iff (\forall x, y \in \Sigma^*, xuy \in L \iff xvy \in L) \\
\approx_A & : u \approx_A v \iff (\forall p, q \in Q, p \xrightarrow{u} q \iff p \xrightarrow{v} q) \\
\sim_A & : u \sim_A v \iff (\forall q \in Q, q_0 \xrightarrow{u} q \iff q_0 \xrightarrow{v} q)
\end{align*}
\]

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

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. \( L = \{ u \in \Sigma^* | [u] \in P \} \) 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 \cite{23}).

Equivalence relations can be compared with respect to their granularity: if \( \sim_1 \sim_2 \), \( \sim_2 \) are two equivalence relations on \( \Sigma^* \), we say that \( \sim_1 \) is finer than \( \sim_2 \) (or \( \sim_2 \) is coarser than

\(^1\) Any automaton can be made complete in polynomial-time.
Aperiodicity of rational functions is PSpace-complete

\( \sim_1 \), if for all \( u, v \in \Sigma^* \) such that \( u \sim_1 v \), it holds \( u \sim_2 v \). We write \( \sim_1 \subseteq \sim_2 \) 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 \( A_1, A_2 \) with respect to their right transition congruence, by saying that \( A_1 \) is finer than \( A_2 \) if \( \sim_{A_1} \subseteq \sim_{A_2} \). We write \( A_1 \subseteq A_2 \) when \( A_1 \) is finer than \( A_2 \). For example, the minimal deterministic automaton recognizing a language \( L \) is the coarsest deterministic automaton recognizing \( L \) with respect to \( \subseteq \).

Rational transductions and finite transducers

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

A transducer \( T = (A, o, i, t) \) is functional if it realizes a transduction (a function). This property is decidable in PTIME (see [1] for instance). \( T \) is called unambiguous (resp. sequential) if \( A \) is unambiguous (resp. deterministic). In both cases \( [T] \) is a transduction and we denote \( (u, w) \in [T] \) by \( [T](u) = w \). 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 \( \Sigma^* \) 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 \( \sim_1 \subseteq \sim_2 \) and \( \sim_1 \) is aperiodic, then so is \( \sim_2 \). A deterministic automaton \( A \) is aperiodic if \( \approx_A \) is aperiodic. In other words, \( A \) is aperiodic if for some \( n > 0 \), for all words \( w \) and states \( p, q \), we have \( \xrightarrow{w^n} q \) iff \( \xrightarrow{w^{n+1}} q \). Deciding whether a deterministic automaton is aperiodic is PSPACE-complete [5]. Since the minimal deterministic automaton \( A_L \) recognizing a language \( L \) is the coarsest automaton recognizing \( L \), and aperiodicity is stable by taking coarser congruences, \( A_L \) 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 \( T = (A, o, i, t) \) is aperiodic if \( A \) 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 \( \text{Left}(\text{Right}(\mathcal{B})) \) isaperiodic if 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 \((p_1, \sigma, q),(p_2, \sigma, q) \in \Delta \) 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 \( s_2 \overset{\lambda}{\to} \mathcal{R} \) \( s_1 \) instead of \( s_2 \overset{\lambda}{\to} \mathcal{R} \), (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 \( r_0r_0r_0 \). The left transition congruence \( \sim_{\mathcal{R}} \) of \( \mathcal{R} \) is defined by \( u \sim_{\mathcal{R}} v \Leftrightarrow (\forall r \in R, r \overset{\lambda}{\to} \mathcal{R} r_0 \Leftrightarrow r \overset{\lambda}{\to} \mathcal{R} r_0) \). Implicitly assuming that \( \mathcal{R} \) is complete, we identify \([u]_{\mathcal{R}} \) (also just denoted by \([u]_{\mathcal{R}} \)) with the state \( r \in R \) such that \( r \overset{\lambda}{\to} \mathcal{R} r_0 \). 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 \( \Sigma \) is a tuple \( \mathcal{B} = (\mathcal{L}, \mathcal{R}, \omega, \lambda, \rho) \) where \( \mathcal{L} = (L, \{l_0\}, F_L, \Delta_L) \) is a left automaton, \( \mathcal{R} = (R, \{r_0\}, F_R, \Delta_R) \) is a right automaton, \( \omega : L \times \Sigma \times R \to \Sigma^* \) is the output function, \( \lambda : F_R \to \Sigma^* \) is the left final function and \( \rho : F_L \to \Sigma^* \) 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 \( \omega \) 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 \( l \overset{u}{\to} \mathcal{L} l' \) and \( r' \overset{\sigma}{\to} \mathcal{R} r \). Now, the domain \( \text{dom}(\mathcal{B}) \) of \( \mathcal{B} \) is \([\mathcal{L}] = [\mathcal{R}]\). For all \( u \in [\mathcal{L}] \), if \( l_0 \overset{u}{\to} \mathcal{L} l \) for some \( l \in F_L \) and \( r \overset{\lambda}{\to} \mathcal{R} r_0 \) for some \( r \in F_R \), 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 \( \text{Left}(\mathcal{B}) = (\text{Left}(\mathcal{R}), \mathcal{R}, \omega, \lambda, \rho) \), 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.
Aperiodicity of rational functions is PSpace-complete

Let \( \text{Left}_f(\mathcal{R}) \) (or simply \( \text{Left}(\mathcal{R}) \) when it is clear from context). For simplicity, we will write \([w]_\mathcal{R}\) instead of \([w]_{\sim_L} \) for any word \( w \in \Sigma^* \). For any two words \( w \) and \( u \), we define
\[
\hat{f}_{[w]_\mathcal{R}}(u) = \Lambda\{f(wv) \mid v \in [w]_\mathcal{R} \cap u^{-1}\text{dom}(f)\}.
\]
This word is the longest possible output upon reading \( u \), knowing that the suffix is in \([w]_\mathcal{R}\). The states of \( \text{Left}_f(\mathcal{R}) \) will be the classes of the right congruence \( \sim_R \) defined by \( u \sim_L v \) if for any letter \( \sigma \), any \( w, z \in \Sigma^* \) we have:
\[
\begin{align*}
uz \in \text{dom}(f) & \iff vz \in \text{dom}(f) \\
\hat{f}_{[w]_\mathcal{R}}(uz) = f_{[w]_\mathcal{R}}(vz) & \iff uz, vz \in \text{dom}(f)
\end{align*}
\]

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

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

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

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

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 \( \text{Left}(\mathcal{B}) \) (and \( \text{Right}(\mathcal{B}) \)) 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:
\[
\omega'([u]_{\mathcal{L}'}, \sigma, [v]_{\mathcal{R}'}) = \hat{f}_{[v\sigma]_{\mathcal{R}'}}(u) \quad \lambda'([v]_{\mathcal{R}'}) = \hat{f}_{[v]_{\mathcal{R}'}}(\epsilon).
\]

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 \( u, v \), \( \hat{f}_{[v]_{\mathcal{R}'}}(u) = \lambda([uv]_{\mathcal{R}'})\omega([v]_{\mathcal{L}'}, u, [v]_{\mathcal{R}'}) \beta([u]_{\mathcal{L}'}, [v]_{\mathcal{R}'}) \) with:
\[
\beta([u]_{\mathcal{L}'}, [v]_{\mathcal{R}'}) = \Lambda\{w \mid \exists x \in [v]_{\mathcal{R}'} \cap u^{-1}\text{dom}(f), \omega([u]_{\mathcal{L}'}, x, [v]_{\mathcal{R}'}) = w\}
\]

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 \( \omega \) labelling the edges. In order to account for the functions \( \lambda \) and \( \rho \) we add two vertices, a source \( s \) pointing to the initial states with the edges labelled 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 \( \beta([u]_{\mathcal{L}'}, [v]_{\mathcal{R}'}) \)

---

3 As a convention, the longest common prefix of an empty set of words is the empty word, that is: \( \Lambda\emptyset = \epsilon \).

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 \( \text{dom}(f) \).
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 finer equivalence relations inspired by Moore’s minimization algorithm for DFA.

The construction of a minimal bimachine in general [17]. In this section, we explain the construction of a minimal left automaton from a right automaton, as described in Section 3.2.

**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 \(\mathcal{B}_1 \subseteq \mathcal{B}_2\), if we have both \(\mathcal{L}_1 \subseteq \mathcal{L}_2\) and \(\mathcal{R}_1 \subseteq \mathcal{R}_2\), 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 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 \(\Sigma\) is defined \(\forall u,v \in \Sigma^*\) by \(u \sim f v\) if:

\[
\begin{align*}
&\forall w \in \Sigma^*, wu \in \text{dom}(f) \Leftrightarrow wv \in \text{dom}(f) \quad \text{and} \\
&\sup\{\|f(wu), f(wv)\| \mid wu, wv \in \text{dom}(f)\} < \infty
\end{align*}
\]
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 $\Sigma^*/\sim_f$. The canonical right automaton for $f$ is $R_f = (\Sigma^*/\sim_f, \{e\}, F, \Delta)$ where $F = \{[w] \mid w \in \text{dom}(f)\}$ and $\Delta = \{[[\sigma w], [\sigma w]] \mid \sigma \in \Sigma, w \in \Sigma^*\}$.

**Remark.** We can define symmetrically the right congruence of $f$ by $u \sim_f v$ if $\forall u, uw \in \text{dom}(f) \Rightarrow uw \in \text{dom}(f)$ and $\sup\{|f(uw), f(vw)| \mid uw, vw \in \text{dom}(f)\} < \infty$ and based on this right congruence, define the canonical left automaton $L_f$.

The automata $R_f$ and $L_f$ 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 Appendix for completeness.

**Proposition 4.** Let $f$ be a transduction, and let $B = (L, R, \omega, \lambda, \rho)$ be a bimachine realizing $f$. Then $L \subseteq L_f$ and $R \subseteq R_f$.

The canonical bimachine [17] associated with a rational transduction $f$ is the bimachine $B_f = (\text{Left}(R_f), R_f, \omega_f, \lambda_f, \rho_f)$ where:

$$
\omega_f([u]_L, \sigma, [v]_R_f) = f_u^{\omega}([\sigma]_R_f)\left(\omega_u^{-1}f_u^{\omega}([\sigma]_R_f)\right)
$$

$$
\lambda_f([v]_R_f) = f_v^{\lambda}([\epsilon]_R_f)
$$

$$
\rho_f([u]_L) = f_u^{\rho}([\epsilon]_R_f)
$$

By its definition, the bimachine $B_f$ 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 $B = (L, R, \omega, \lambda, \rho)$ such that $B_f \subseteq B$, i.e. $\text{Left}(R_f) \subseteq L$ and $R_f \subseteq R$. Then $\omega$ (and similarly $\lambda$ and $\rho$) can be restricted to $\omega'(u_L, \sigma, v_R_f) = \omega(u_L, \sigma, v_R)$, which is well-defined since $R_f \subseteq R$, so that the bimachine $(L, R_f, \omega', \lambda', \rho')$ realizes $f$. By Proposition 2 we get $L \subseteq \text{Left}(R_f)$. Moreover, by Proposition 4, we also have $R \subseteq R_f$.

Finally, $B_f$ 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 $B$ is aperiodic, one only needs to minimize $B$, i.e. to construct $\text{Left}(\text{Right}(B))$, 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 $B_f$ 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 $\Sigma$, we show that it is aperiodic iff the canonical bimachine $B_f$ associated with $f$, defined in the previous section, is aperiodic. It relies on two important facts: (i) the left congruence $\sim_f$ of an aperiodic transduction is aperiodic (Proposition 5), (ii) any aperiodic rational transduction $f$ can be decomposed into $f = \ell \circ \text{label}_R_f$ such that $\ell$ is realized by some aperiodic sequential transducer, and $\text{label}_R_f$ annotates every input position $i$ with the class of the suffix from $i$ by $\sim_f$ (Proposition 6). From this decomposition, one can construct an aperiodic bimachine whenever $f$ is aperiodic. First, one shows (in Appendix) that $R_f$ is aperiodic when $f$ is too:
Proposition 5. Let $f$ be a transduction realizable by an aperiodic transducer then the congruence $\sim_f$, 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 $\omega$. We show that any aperiodic rational transduction $f$ can be decomposed into $\ell \circ \text{label}_{\mathcal{R}_f}$ such that $\ell$ can be realized by a sequential aperiodic transducer, and $\text{label}_{\mathcal{R}_f}$ annotates the input word with states of $\mathcal{R}_f$.

More precisely, let $\mathcal{R}$ be a right automaton over $\Sigma$ with set of states $Q$, and let $\Sigma_\mathcal{R} = \{\sigma_q \mid \sigma \in \Sigma, q \in Q\}$. We define the rational transduction $\text{label}_{\mathcal{R}} : \Sigma^* \to \Sigma_\mathcal{R}^*$, called the labelling function of $\mathcal{R}$, which labels words in $\Sigma^*$ 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 $\epsilon$, and with $o(p, \sigma, q) = \sigma_q$.

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

Proof. We first show that there exists a sequential transduction $\ell$ such that $f = \ell \circ \text{label}_\mathcal{R}_f$. This sequential transduction is realized by the left automaton of the canonical bimachine $B_f$ combined with the output function $\omega_f$. Then, we show that $\ell$ 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 Appendix.

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 $f = \ell \circ \text{label}_\mathcal{R}_f$, such that $\ell$ is realized by some aperiodic sequential transducer $T_m = (A_m, \sigma_m, \epsilon_m, t_m)$. 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 $\mathcal{D} \subseteq \text{Left}(\mathcal{R}_f)$, and since $\mathcal{D}$ is aperiodic, so is $\text{Left}(\mathcal{R}_f)$. Since $\mathcal{R}_f$ is aperiodic as well by Proposition 5, it implies that $\mathcal{B}_f$ is aperiodic. Let us now construct $\mathcal{D}$, $\omega$, $\lambda$ and $\rho$.

Recall that the input alphabet of $T_m$ (and $A_m$) is $\Sigma_Q$ where $Q = \Sigma^*/\sim_f$. To construct $\mathcal{B}$, it is tempting to think that it suffices to take the projection of $A_m$ on $\Sigma$ as left automaton, $\mathcal{R}_f$ as right automaton, and to define the output function $\omega$ by $\omega(p, \sigma, [u]) = w$ where $p \xrightarrow{\sigma_u} q$ is the transition of $T_m$ on $\sigma_u$ with output $w$. The problem is that the projection of $A_m$ on $\Sigma$ is not a deterministic automaton in general, and by determinizing it, one loses the information of which transition of $T_m$ 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.

$\blacksquare$
Aperiodicity of rational functions is PSpace-complete

Let us detail this construction. We take $T_m$ and project input letters to their $\Sigma$ component, hence we obtain a transducer realizing $f$ which is unambiguous, since $R_f$ has backward deterministic transitions. Let $\tilde{T}_m$ denote the obtained transducer, and $\tilde{A}_m$ its underlying (unambiguous) automaton. We let $D$ be the automaton obtained by determinization, by subset construction, of the product automaton $\tilde{A}_m \times R_f$. States of $D$ are therefore of the form $2^{Q_m \times \Sigma^*/f}$. The output function $\omega$ is defined by:

$$\omega((p_1, [u_1]), \ldots, (p_n, [u_n]), \sigma, v) = \sigma_m(p_1, \sigma)$$

such that $[\sigma v] = [u_i]$. The state $p_i$ is unique since $T_m$ 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 $R_f$ has backward deterministic transitions, for a word $w$ which reaches both $p_i$ and $p_j$ in $\tilde{A}_m$, 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 $q_0 \xrightarrow{z} A_m p_i$ and $q_0 \xrightarrow{z} A_m p_j$ which is in contradiction with the deterministic nature of $A_m$. We define the final output functions naturally: $\lambda([u]) = t_m(q_m, 0)$ and $\rho((p_1, [u_1]), \ldots, (p_n, [u_n])) = t(p_i)$ such that $[u_i] = [c]$ (again, it is unique by unambiguity of $T_m$).

It remains to show that $D$ is aperiodic. Aperiodicity of $R_f$ is obtained by Proposition 5. Aperiodicity of $D$ is shown in Appendix, as a consequence of the aperiodicity of $T_m$, $R_f$ 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 $Left(R_f)$ 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 $f(wu_n) = w_n$ 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 $L_f$ 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 $B$ it suffices to construct $Left(Right(B))$ (or $Right(Left(B))$) and to check aperiodicity of the resulting bimachine. The first step can be done in PTIME and the second step in PSPACE. The operations $Left(Right(B))$ and $Right(Left(B))$ are called bimachine minimization, as they indeed yield minimal bimachines, as shown by the following proposition:

**Proposition 9.** Let $B$ be a bimachine realizing a transduction $f$. Then $Left(Right(B))$ and $Right(Left(B))$ are minimal bimachines realizing $f$. 


Proof. Based on successive applications of Proposition 2 and given in Appendix.

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 $\text{Left}(\text{Right}(\mathcal{B}))$ is aperiodic iff $\text{Right}(\text{Left}(\mathcal{B}))$ is aperiodic.

Proof. First, we start by some observation: if there are two bimachines realizing $f$ with $A_1 \sqsubseteq A_2$ as right automata, then $\text{Left}(f(A_2)) \sqsubseteq \text{Left}(f(A_1))$. Indeed, if $A_1$ provides more (i.e. finer) information than $A_2$ on suffixes, then the two equalities of the definition of the right congruence used to define $\text{Left}(f(A_1))$ (see Section 3.2) are “easier” to satisfy since the set of suffixes $v$ taken into account in the definition of $\hat{f}_{[w]}^{\text{A}_1}$ is included in the set of suffixes used in the definition of $\hat{f}_{[w]}^{\text{A}_2}$ for all words $w$. Symmetrically, if there are two bimachines realizing $f$ with $A_1 \sqsubseteq A_2$ as left automata, then $\text{Right}(f(A_2)) \sqsubseteq \text{Right}(f(A_1))$.

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

By Theorem 7, if $f$ is aperiodic, then $\mathcal{B}_f = (\text{Left}(\mathcal{R}_f), \mathcal{R}_f, \omega, \lambda, \rho)$ is aperiodic. Therefore, $\text{Left}(\text{Right}(\mathcal{L}))$ is aperiodic. Symmetrically, exactly as shown in Theorem 7, it can be shown that $\mathcal{L}_f$ and $\text{Right}(\mathcal{L}_f)$ are aperiodic if $f$ is aperiodic, which implies that $\text{Right}(\mathcal{L})$ is aperiodic. In conclusion, $\text{Left}(\text{Right}(\mathcal{B}))$ 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 $\text{Right}(\mathcal{B})$, (ii) construct $\text{Left}(\text{Right}(\mathcal{B}))$, and (iii) test whether the left and right automata of the bimachine $\text{Left}(\text{Right}(\mathcal{B}))$ 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 Appendix.

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 $2\text{ExpTime}$, 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.
0:14  Aperiodicity of rational functions is PSpace-complete

References


A

Appendix

A.1 Proof of Proposition 2

Proof. Let \( B = (L, \mathcal{R}, \omega, \lambda, \rho) \), and \( f \) be the function it realizes. Let \( u \sim_{L} v \), we will show that \( u \sim_{L} v \) (using the same notations as above). First we have that \( \forall \sigma, \ uz \in \text{dom}(f) \iff vz \in \text{dom}(f) \) since \( L \) recognizes \( \text{dom}(f) \). Let \( u \) and \( v \) be two words, then we have that:

\[
\beta([u]_L, [w]_\mathcal{R}) = \bigwedge \{ x \mid \exists y \in [w]_\mathcal{R} \cap u^{-1}\text{dom}(f), \ (\omega([u]_L, y, [e]_\mathcal{R}) \rho([uy]_L) = x) \}
\]

Hence we have, for all \( z \) such that \( uz, vz \in \text{dom}(f) \):

\[
\hat{f}_{[z]_\mathcal{R}}(uz) = (\lambda([uz]_\mathcal{R}) \omega([e]_\mathcal{L}, uz, [e]_\mathcal{R}) \beta([uz]_\mathcal{L}, [e]_\mathcal{R})^{-1} \lambda([uz]_\mathcal{R}) \omega([e]_\mathcal{L}, uz, [e]_\mathcal{R}) \rho([uz]_L))
\]

Similarly, we have for two words \( w, z \) and a letter \( \sigma \):

\[
\hat{f}_{[\sigma w]_\mathcal{R}}(uz) = (\lambda([uw\sigma w]_\mathcal{R}) \omega([e]_\mathcal{L}, uz, [\sigma w]_\mathcal{R}) \beta([uz]_\mathcal{L}, [\sigma w]_\mathcal{R})^{-1} \lambda([uw\sigma w]_\mathcal{R}) \omega([e]_\mathcal{L}, uz, [\sigma w]_\mathcal{R}) \rho([uz]_L))
\]

Hence we have that \( u \sim_{L} v, \sim_{L} \) is indeed the coarsest left congruence among the bimachines realizing \( f \) with \( \mathcal{R} \) as their right automaton.

A.2 Proof of \( \sim_s = \sim_L \) in Theorem 3

Proof. We want to show that \( \sim_s = \sim_L \). By induction on \( i \geq 0 \), we first show that for all \( u, v \in \Sigma^*; u \sim_{L} v \) iff for all \( z \in \Sigma^i \), all \( \sigma \in \Sigma \) and all \( w \in \Sigma^* \):

1. \( uz \in \text{dom}(f) \) iff \( vz \in \text{dom}(f) \),
2. \( \rho'(\{uz\}_L) = \rho'(\{vz\}_L) \) if \( uz, vz \in \text{dom}(f) \),
3. \( \omega'([uz]_L, \sigma, [w]_\mathcal{R}) = \omega'([vz]_L, \sigma, [w]_\mathcal{R}) \).

At rank 0, condition (1) comes from the condition \( l \in F_L \) iff \( l' \in F_L \) in the definition of \( \sim_0 \), and conditions (2) and (3) are immediate consequences of the definition of \( \omega' \) and \( \rho' \) respectively.

Now, suppose it is true at rank \( i \geq 0 \). Then, \( u \sim_{i+1} v \) iff \( u \sim_{i} v \) and \( u^{\beta} \sim_{i} v^{\beta} \) for all \( \beta \in \Sigma \). By induction hypothesis on \( u^{\beta} \sim_{i} v^{\beta} \), we get that \( u^{\beta}z \in \text{dom}(f) \) iff \( v^{\beta}z \in \text{dom}(f) \) for all \( z \in \Sigma^i \) and \( \beta \in \Sigma \), hence \( uz' \in \text{dom}(f) \) iff \( vz' \in \text{dom}(f) \) for all \( z' \in \Sigma^{i+1} \). Similarly, we get \( \omega'(\{uz\}_L, \sigma, [w]_\mathcal{R}) = \omega'(\{vz\}_L, \sigma, [w]_\mathcal{R}) \) for all \( z \in \Sigma^i \), all \( \sigma, \beta \in \Sigma \), which is equivalent to \( \omega'(\{uz\}_L, \sigma, [w]_\mathcal{R}) = \omega'(\{vz\}_L, \sigma, [w]_\mathcal{R}) \) for all \( z' \in \Sigma^{i+1} \) and all \( \sigma \in \Sigma \). Similarly, by definition of \( \rho' \) one immediately gets \( \rho'(\{uz\}_L) = \rho'(\{vz\}_L) \) for all \( z \in \Sigma^{i+1} \) if \( uz, vz \in \text{dom}(f) \). The converse is shown similarly, concluding the proof of the characterization of \( \sim_i \) by properties 1, 2, 3.
Now, since $\sim_*$ is the fixpoint of the sequence $(\sim_i)_{i \geq 0}$, we have $u \sim_* v$ iff properties 1, 2, 3 holds for all $z \in \Sigma^*$.

Finally, we conclude by the following observation, which is a direct consequence of the definition of $\omega'$ and $\rho'$: for all $u, z \in \Sigma^*$ and $\sigma \in \Sigma$:

$$\omega'(\langle uz \rangle_{\mathcal{L}}, \sigma, [w]_{\mathcal{R}}) = \hat{f}_{\omega(u)}^{-1}(u\sigma u) \hat{f}_{[w]_{\mathcal{R}}}^{-1}(u\sigma u)$$

and therefore the conditions 1, 2, 3 just corresponds exactly to conditions of the definition of $\sim_\mathcal{L}$.

### A.3 Proof of Proposition 4

**Proof.** Let $f$ be a transduction realized by the bimachine $\mathcal{B} = (\mathcal{L}, \mathcal{R}, \omega, \lambda, \rho)$. Let us show that $\mathcal{R} \subseteq \mathcal{R}_f$ and the proof for left automaton is completely symmetrical. Let $u \sim \mathcal{R} v$, we will show that $u \sim_f v$. For any word $w$, $wu \in \text{dom}(f) \Leftrightarrow wv \in \text{dom}(f)$ since $\mathcal{R}$ recognizes $\text{dom}(f)$. Let us assume that $wu, wv \in \text{dom}(f)$ for some word $w$. Let $l = l_0 \ldots l_{|wu|}$ and $r = r_0 \ldots r_0$ denote the runs on $wu$ of $\mathcal{L}$ and $\mathcal{R}$, respectively. Similarly, let $l' = l_0 \ldots l_{|wv|}$ and $r' = r_0 \ldots r_0$ denote the runs on $wv$ of $\mathcal{L}$ and $\mathcal{R}$, respectively. Since $u \sim \mathcal{R} v$ and $\mathcal{R}$ is deterministic, the ends (from right to left) of the runs $r$ and $r'$ are identical, i.e. for $0 \leq i \leq |w|$, we have $r_i = r_i'$. Hence for $x = f(wu) \land f(wv)$ we have

$$|x^{-1}f(wu)| \leq |\omega(l_{|w|}, u, r_0)\rho(l_{|wu|})|$$

Finally we obtain $|f(wu), f(wv)| \leq k(|u| + |v| + 2)$ where $k$ is the maximum length of any word in the ranges of $\omega, \lambda$ and $\rho$, which proves that $u \sim_f v$.

### A.4 Proof of Proposition 5

**Proof.** Let $\mathcal{T} = (\mathcal{A}, q, i, t)$ be a transducer realizing a transduction $f$ and let $\mathcal{A} = (Q, I, F, \Delta)$. We only have to show that $\approx \mathcal{A} \subseteq \sim_f$, because aperiodicity is stable by taking coarser congruences. Let $u, v, w \in \Sigma^*$ such that $u \approx \mathcal{A} v$. By definition of $\approx \mathcal{A}$, we have $wu \in \text{dom}(f)$ iff $wv \in \text{dom}(f)$. Now, suppose that $wu \in \text{dom}(f)$. We show that $\|f(wu), f(wv)\|$ does not depend on $w$. Since $wu \in \text{dom}(f)$, there is a run $q_0 \xrightarrow{w} q_0 \xrightarrow{u} q_0 \xrightarrow{v} q_0 \xrightarrow{F}$ with $q_0 \in I$ and $q_0 \in F$. Since $u \approx \mathcal{A} v$, we have a run $q_0 \xrightarrow{w} q_0 \xrightarrow{u} q_0 \xrightarrow{v} q_0 \xrightarrow{F}$. Let $i = i(q_0)$ and $t = t(q_0)$, then $\|f(wu), f(wv)\| = \|x, y\|$ which does not depend on $v$, and is bounded. Hence $u \sim_f v$.

### A.5 Proof of Proposition 6

The proof of Proposition 6 is based on two propositions.

**Proposition 12.** Let $f$ be a rational function. There exists a sequential transducer $\ell$ such that $f = \ell \circ \text{label}_{\mathcal{R}_f}$.

**Proof.** Let us consider the canonical bimachine realizing $f$: $\mathcal{B}_f = (\text{Left}(\mathcal{R}_f), \mathcal{R}_f, \omega_f, \lambda_f, \rho_f)$, and using the same notations as in the previous section, let $\sim_{\mathcal{L}}$ denote the right congruence of $\text{Left}(\mathcal{R}_f)$. Then we define a sequential transducer $\ell = (A, q, \epsilon, t)$ with $A = (Q, I, F, \delta)$ realizing $\ell$ with:

- $Q_\ell = \{([u]_{\mathcal{L}}, [v]) \mid vu \in \text{dom}(f)\} \cup \{q_0\}$ for some fresh state $q_0$ (remind that $[v]$ denotes the class of $v$ by the canonical left congruence $\sim_f$),
- $\delta_\ell = \{([u]_{\mathcal{L}}, [\sigma v]), ([\sigma v], ([u]_{\mathcal{L}}, [v])) \} \cup \{([q_0, \sigma[v]], ([\sigma]_{\mathcal{L}}, [v])) \mid \sigma v \in \text{dom } f\}$.
Aperiodicity of rational functions is PSpace-complete

Let \( \ell \) be an aperiodic transduction. There exists an aperiodic transduction \( \ell \) such that \( f = \ell \circ \text{label}_{R_f} \).

**Proof.** Let \( T \) be an aperiodic transducer realizing \( f \) with underlying automaton \( A = (Q, I, F, \delta) \). We first define a transducer \( T' \) with underlying automaton \( A' = (Q', I', F', \delta') \) defined by the product of \( A \) and \( R_f \):

\[
Q' = Q \times \Sigma^*/\sim_f, \quad I' = I \times \text{dom}(f)/\sim_f, \quad F' = F \times \{[e]\}, \quad \delta' = \{(p, [u]), (q, [u]) | \exists p \in \Sigma, (q, [u]) \in \delta(p, [u])\}
\]

The outputs of transducer \( T' \) are the same as the ones of \( T \), ignoring the extra information given by \( R_f \), i.e. \( \delta'(p, [u]) = \delta(p) \) for any state \( p \) of \( A \) and any word \( u \). Therefore \( T' \) is nothing else than \( T \) whose states have been enriched by the information of \( R_f \). The automaton \( A' \) is aperiodic since it is the product of \( A \) which is aperiodic by assumption and \( R_f \) which is aperiodic by Proposition 5. Indeed the transition congruence of a product automaton is coarser than the intersection of the transition congruences of its components. Now we transform the labels of the transitions of \( T' \) to obtain the automaton \( A_f \) of a transducer \( T_f \) realizing \( f \). For each transition \( (p, [u]) \xrightarrow{\sigma}{A'} (q, [u]) \) we set \( (p, [u]) \xrightarrow{\sigma}{A_f} (q, [u]) \).

By construction of \( T_f \), we have \( f = \ell \circ \text{label}_{R_f} \). Let us show that \( T_f \) is aperiodic. Let \( u \in (\Sigma \times \Sigma^*/\sim_f)^* \) be a word labelled by states of \( R_f \) and let \( \lambda: (\Sigma \times \Sigma^*/\sim_f)^* \to \Sigma^* \) be the natural projection morphism.

Let us assume that for \( n \) large enough we have \( (p, [u_1]) \xrightarrow{u^n}{A_f} (q, [u_2]) \). We show that \( (p, [u_1]) \xrightarrow{u^n}{A_f} (q, [u_2]) \). The converse is similar and omitted. By definition of \( T_f \), we necessarily have \([u_1] = [u^n v]\) for some \( v \in \Sigma^* \), and \([u_2] = [v]\). Hence we have to show that \( (p, [u^n v]) \xrightarrow{u^n}{A_f} (q, [v]) \). We can assume \( (p, [u^n v]) \xrightarrow{u^n}{A_f} (q, [v]) \). By aperiodicity of \( A \) we have \( p \xrightarrow{\tilde{u}^{n+1}} A q \). Furthermore, by definition of \( A_f \) and the fact that there is a run of the form \( (p, [u^n v]) \xrightarrow{u^n}{A_f} (q, [v]) \), we necessarily have \( u^n \equiv \sigma_{\Sigma} \equiv \sigma_{\Sigma} \in \Sigma \). This implies that \([v] \in R_f \), hence \([u^n v] = [v] \) since \( \sim_f \) is a left congruence. Finally we obtain \( (p, [v] = [u^n v]) \xrightarrow{u^n}{A_f} (q, [v]) \).

**Proof of Proposition 6**

According to Propositions 12 and 13, \( f \) is equal to the composition \( \ell \circ \text{label}_{R_f} \), with \( \ell \) an aperiodic and sequential transduction. It was shown in [13] that a transduction realizable by an aperiodic transducer and realizable by a sequential transducer is realizable by a transducer which is both sequential and aperiodic.

### A.6 Proof of Theorem 7 (continued)

To show that \( D \) is aperiodic, first we show that \( \tilde{A}_m \) is aperiodic. Let \( u \) be a word such that \( p \xrightarrow{u^n}{\tilde{A}_m} q \). Let \( v \in \Sigma_f \) such that \( \tilde{v} = \tilde{u}^n \) (the projection of \( v \) on \( \Sigma \)) and \( p \xrightarrow{\tilde{u}^n}{A_m} q \). Let \( w \) be such that \( v[[v]] = u[[u][w]] \). Since \( R_f \) is aperiodic, there exists \( k \) such that \( [u^k w] = [u^{k+1} w] \).
Hence we have: \( p \xrightarrow{n'} A_m \xrightarrow{u^k} \tilde{A}_m q \) (for some large enough \( n' \)). Let \( v' \) be such that \( \tilde{v}' = u^{n'} \) and \( p \xrightarrow{v'} A_m p' \). Then we have \( v'[|v'|-|u|] = u[i]|w[i]|u^k w \) for any \( i \in \{0, \ldots, n'\} \). Hence we must have \( \tilde{v}' = x^n \) where \( x[i] = u[i] \), where \( c \) is the class of the word \( u[i+1] \ldots u[|u|]u^k w \).

Then we obtain \( p \xrightarrow{x^{n'+1}} A_m p' \) which means by aperiodicity that \( p \xrightarrow{x^{n'+1}} A_m p' \). Hence \( p \xrightarrow{x^{n'+1}} A_m p' \xrightarrow{u^k} \tilde{A}_m q \) and \( \tilde{A}_m \) is aperiodic. Since \( \mathcal{R}_f \) is aperiodic, we also obtain that the product automaton \( \tilde{A}_m \times \mathcal{R}_f \) is aperiodic. Subset construction preserves aperiodicity, therefore \( \mathcal{D} \) is aperiodic, which concludes the proof.

### A.7 Proof of Proposition 9

**Proof.** We only show the result for \( \text{Right}(\text{Left}(\mathcal{B})) \), the other result being proved symmetrically. Let \( \mathcal{B} = (\mathcal{L}, \mathcal{R}, \omega, \lambda, \rho) \). Suppose that \( \text{Right}(\text{Left}(\mathcal{B})) \) is not minimal and let \( \mathcal{B}' = (\mathcal{L}', \mathcal{R}', \omega', \lambda', \rho') \) be a bimachine realizing \( f \) such that \( \text{Right}(\text{Left}(\mathcal{B})) \subseteq \mathcal{B}' \). Then, \( \text{Right}(\text{Left}(\mathcal{B})) \) has \( \text{Left}(\mathcal{R}) \subseteq \mathcal{L}' \) as left automaton, and \( \text{Right}(\text{Left}(\mathcal{R})) \subseteq \mathcal{R}' \) as right automaton.

By Proposition 2, we have \( \mathcal{L} \subseteq \text{Left}(\mathcal{R}) \) and \( \mathcal{R} \subseteq \text{Right}(\text{Left}(\mathcal{R})) \), hence

\[
\mathcal{L} \subseteq \text{Left}(\mathcal{R}) \subseteq \mathcal{L}' \quad \mathcal{R} \subseteq \text{Right}(\text{Left}(\mathcal{R})) \subseteq \mathcal{R}'
\]

Since \( \mathcal{R} \subseteq \mathcal{R}' \), we can restrict the right automaton of \( \mathcal{B}' \) to \( \mathcal{R} \) and obtain a bimachine realizing \( f \) with \( \mathcal{L}' \) as left automaton, and \( \mathcal{R} \) as right automaton. This bimachine just ignores the extra information given by \( \mathcal{R} \), by setting its output function to \( ([u]_{\mathcal{L}'}, [\sigma, [v]_{\mathcal{R}}]) \mapsto \omega'([u]_{\mathcal{L}'}, [\sigma, [v]_{\mathcal{R}}]) \). If \( \mathcal{L}' \) is strictly coarser than \( \text{Left}(\mathcal{R}) \), this contradicts the minimality of \( \text{Left}(\mathcal{R}) \) among bimachines realizing \( f \) with \( \mathcal{R} \) as right automaton (Proposition 2).

Now, suppose that \( \mathcal{R}' \) is strictly coarser than \( \text{Right}(\text{Left}(\mathcal{R})) \). We also show a contradiction, by a similar argument. Since \( \text{Left}(\mathcal{R}) \subseteq \mathcal{L}' \), we can restrict the left automaton of \( \mathcal{B}' \) to \( \text{Left}(\mathcal{R}) \) (the output function ignores the finer information given by \( \text{Left}(\mathcal{R}) \)) and obtain a bimachine realizing \( f \) with \( \text{Left}(\mathcal{R}) \) as left automaton, and \( \mathcal{R}' \) as right automaton. This yields a contradiction since \( \mathcal{R}' \) is strictly finer than \( \text{Right}(\text{Left}(\mathcal{R})) \) and \( \text{Right}(\text{Left}(\mathcal{R})) \) is the minimal automaton among bimachines realizing \( f \) with \( \text{Left}(\mathcal{R}) \) as right automaton. \( \blacktriangleleft \)

### A.8 Proof of Theorem 11

**Proof.** It remains to show the lower bound.

Let \( \mathcal{A} \) be a minimal complete DFA recognizing a language \( M \subseteq \Sigma^* \), with set of states \( Q \) and initial state \( q_0 \). We construct (in polynomial time) a bimachine \( \mathcal{B} \) such that \( \mathcal{B} \) realizes an aperiodic rational function iff \( \mathcal{A} \) is aperiodic. Since the left and right automata of a bimachine must recognize the same language, we cannot construct \( \mathcal{B} \) such that \( \text{dom}(\mathcal{B}) = M \) (with \( \mathcal{A} \) as left automaton for instance), because a right automaton recognizing \( M \) can be exponentially larger than \( \mathcal{A} \) in general. Instead, we will define \( \mathcal{B} \) such that for all \( u \in \Sigma^* \), \( |\mathcal{B}|(u) = r \), where \( r \) is the run of \( \mathcal{A} \) on \( u \). Note that \( \text{dom}(\mathcal{B}) = \Sigma^* \).

To do this, we let \( \mathcal{B} = (\mathcal{A}', \mathcal{A}_\top, \omega, \lambda, \rho) \) where \( \mathcal{A}' \) is the automaton \( \mathcal{A} \) where all states are final, \( \mathcal{A}_\top \) is a single-state complete automaton whose state \( q_{\top} \) is both initial and accepting. Note that both \( \mathcal{A}' \) and \( \mathcal{A}_\top \) recognizes the same language \( \Sigma^* \). The function \( \omega : Q \times \Sigma \times \{q_{\top}\} \) is defined by \( \omega(q, \sigma, q_{\top}) = q \) for all \( q \in Q \). The left final function \( \rho \) is defined by \( \rho(q) = q \) for all \( q \in Q \), and the right terminal function \( \lambda \) is defined by \( \lambda(q_{\top}) = \epsilon \).

We claim that \( \mathcal{B} \) realizes an aperiodic rational function iff \( \mathcal{A} \) is aperiodic. Let us prove this statement. Suppose that \( \mathcal{A} \) is aperiodic, then clearly, \( \mathcal{A}' \) is aperiodic as well, and
A\top is trivially aperiodic, hence B is aperiodic (and therefore realizes an aperiodic rational function).

Conversely, suppose that B realizes an aperiodic rational function f = \lfloor B \rfloor. By Theorem 7, the canonical bimachine B_f is aperiodic. Since dom(f) = \Sigma^* and for all w, u, v \in \Sigma^*, ||f(wu), f(wv)|| is bounded by |u| + |v|, the canonical right automaton of R_f of B_f is trivial and isomorphic to A\top. Now, let us study the right congruence ~L defined with respect to R_f.

First, since R_f is trivial, there is only one class in ~R_f, the class \Sigma^*. Consider the function \hat{f}_{\Sigma^*}. To any word u it maps the word \hat{f}_{\Sigma^*}(u) = \bigwedge \{ f(\sigma v) \mid v \in \Sigma^* \}, which is equal to r_u, the unique run of A on u. Hence \hat{f}_{\Sigma^*}(u) = f(u).

Now, take u, v \in \Sigma^*. Note that u \equiv_{\text{dom}(f)} v since dom(f) = \Sigma^*. Then \hat{f}_{\Sigma^*}(uz)^{-1} f(uz) = f(uz)^{-1} f(uz) = \epsilon = f(vz)^{-1} f(vz) = \hat{f}_{\Sigma^*}(vz)^{-1} f(vz). Let \sigma \in \Sigma. Then \hat{f}_{\Sigma^*}(uz)^{-1} \hat{f}_{\Sigma^*}(us\sigma) = f(uz)^{-1} f(uz\sigma) = q if q is the last state of the run r_{uz\sigma}. Similarly, \hat{f}_{\Sigma^*}(vz)^{-1} \hat{f}_{\Sigma^*}(vz\sigma) = f(vz)^{-1} f(vz\sigma) = p if p is the last state of the run r_{vz\sigma}. From this we deduce that u \sim_L v if and only if for all z \in \Sigma^*, for all \sigma \in \Sigma, the runs r_{uz\sigma} and r_{vz\sigma} ends with the same state. In particular, this implies that u \sim_A v. Indeed, let q such that q_0 \xrightarrow{\sigma}_A q, and let q' such that q_0 \xrightarrow{\sigma}_A q'. We show that q = q'. By taking z = \epsilon, we have that for all \sigma \in \Sigma, \delta_A(q, \sigma) = \delta_A(q', \sigma). By minimality of A, this implies q = q'. Conversely, if u \sim_A v, then there exists q \in Q such that q_0 \xrightarrow{\epsilon}_A q and q_0 \xrightarrow{\epsilon}_A q. Therefore for all z \in \Sigma^* and \sigma \in \Sigma, since A is deterministic, we get that r_{uz\sigma} and r_{vz\sigma} ends with the same state, and hence u \sim_L v.

We have just shown that u \sim_L v if u \sim_A v, which implies that Left_f(R_f) is isomorphic to A, and therefore A is aperiodic.