On Canonical Models for Rational Functions over Infinite Words

Emmanuel Filiot
Université Libre de Bruxelles, Belgium
efiliot@ulb.ac.be

Olivier Gauwin
LaBRI, Université de Bordeaux, France
olivier.gauwin@labri.fr

Nathan Lhote
LaBRI, Université de Bordeaux, France and Université Libre de Bruxelles, Belgium
nlhote@labri.fr

Anca Muscholl
LaBRI, Université de Bordeaux, France
anca@labri.fr

Abstract

This paper investigates canonical transducers for rational functions over infinite words, i.e., functions of infinite words defined by finite transducers. We first consider sequential functions, defined by finite transducers with a deterministic underlying automaton. We provide a Myhill-Nerode-like characterization, in the vein of Choffrut's result over finite words, from which we derive an algorithm that computes a transducer realizing the function which is minimal and unique (up to the automaton for the domain).

The main contribution of the paper is the notion of a canonical transducer for rational functions over infinite words, extending the notion of canonical bimachine due to Reutenauer and Schützenberger from finite to infinite words. As an application, we show that the canonical transducer is aperiodic whenever the function is definable by some aperiodic transducer, or equivalently, by a first-order transduction. This allows to decide whether a rational function of infinite words is first-order definable.

2012 ACM Subject Classification Theory of computation → Transducers

Keywords and phrases transducers, infinite words, minimization, aperiodicty, first-order logic

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2018.30

Related Version A full version of the paper is available at `https://hal.archives-ouvertes.fr/hal-01889429`.

Funding This work was supported by the French ANR projects *ExStream* (ANR-13-JS02-0010) and *DeLTA* (ANR-16-CE40-0007), the Belgian FNRS CDR project *Flare* (J013116) and the ARC project *Transform* (Fédération Wallonie Bruxelles).

Introduction

Machine models, such as automata and their extensions, describe mathematical objects in a finite way. Finite automata, for instance, describe languages (of words, trees, etc). A canonization function $\mathcal{C}$ is a function from and to machine models (not necessarily of the same type) such that, whenever two machines M_1, M_2 describe the same object, then

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 30; pp. 30:1–30:17

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

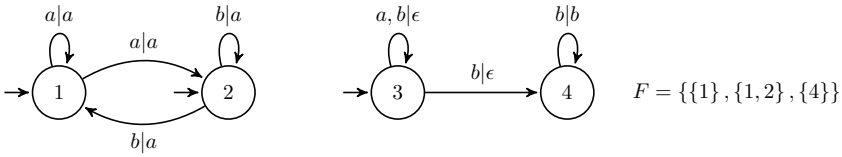

Figure 1 Transducer with Muller sets F realizing the function $f_{\#a}$ mapping any word with infinitely many a to a^ω, otherwise to b^ω.

$\mathcal{C}(M_1) = \mathcal{C}(M_2)$. Accordingly, $\mathcal{C}(M_1)$ is called the *canonical model* of the object described by M_1, and it does not depend on the initial representation of the object. A classical example of canonization is the function which associates with a finite automaton its equivalent minimal deterministic automaton. A canonization function becomes interesting when it satisfies additional constraints like being computable, preserving some algebraic properties, and enjoying minimal models. Canonical models not only shed light on the intrinsic characteristics of the class of objects they describe, but can also serve to decide *definability* problems. For instance, it is well-known that the minimal DFA of a word language L is aperiodic if and only if L is definable in first-order logic [17, 21]. Hence, this allows to decide whether a monadic second-order formula has an equivalent first-order one over words. This result has been extended to infinite words [23, 24, 1, 18], although there is no unique minimal automaton for languages of infinite words (see also [10] for a survey).

Rational functions are functions defined by word transducers. A canonical model for rational functions over *finite* words has been introduced in [20]. This result, which can be considered as one of the jewels of transducer theory, states the existence of a procedure that computes from a given transducer a canonical input-deterministic transducer with look-ahead, called bimachine. For the subclass of functions realized by input-deterministic transducers, called sequential functions, it is even possible to compute a unique and minimal transducer realizing the function [8]. For rational functions, the procedure of [20], though it preserves aperiodicity of the transition congruence of the transducer, does not preserve other congruence varieties, in general. In [14, 15] it was shown how to adapt [20] to obtain a canonization procedure which overcomes this issue. Later it was shown that the first-order definability problem for rational functions is PSpace-c [13]. In a different setting, functions *with origin information* realizable by two-way transducers were shown to have decidable first-order definability [4]. In this paper, we extend the results of [20] and the decidability of first-order definability of [13] to rational functions of infinite words.

Rational functions of infinite words. We consider rational functions of infinite words, *i.e.* functions defined by transducers with Muller acceptance condition. Such machines map any ω-word for which there exists an accepting run to either a finite or an ω-word. Take as example the function $f_{\#a}$ over alphabet $\{a, b\}$ mapping any word containing an infinite number of a to a^ω, and to b^ω otherwise. This function is realized by the transducer of Fig. 1.

The class of *sequential functions* is of particular interest: they are realized by transducers whose underlying input Muller automaton is deterministic. Note that the function $f_{\#a}$ is not sequential, unlike the function f_{ab} of Fig. 2. Sequential functions over infinite words have been studied *e.g.* in [2]. One difference between our setting and [2] is that in the latter paper infinite words are mapped to infinite words, whereas we need also functions that map infinite words to finite words. Deciding whether a rational function is sequential can be done in

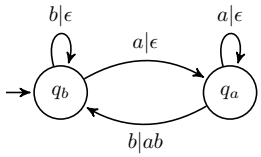

Figure 2 Sequential transducer with Muller condition $F = \{\{q_b\}\}$ realizing the function f_{ab} which maps any word containing a finite number of a's to the subsequence of ab factors, and is undefined otherwise.

PTIME, as shown in [2]. Bimachines for infinite words were introduced in [25] to define the particular class of total letter-to-letter rational functions, and in their counter-free versions, a connection with linear temporal logic was established.

To the best of our knowledge, nothing is known about canonical models for sequential and rational functions over infinite words, and their applications to definability problems in logics.

Contributions.

(1) We provide a characterization of sequential functions by means of the finiteness of a congruence. We give a PTIME procedure which, for any sequential function f given as a transducer whose domain is topologically closed, produces *the* minimal (and hence canonical) sequential transducer $\mathcal{T}_f$ realizing f. When the domain of f is not topologically closed, we extend f to a domain-closed sequential function $\overline{f}$ which coincides with f on its domain. By intersecting $\mathcal{T}_{\overline{f}}$ with some automaton $\mathcal{D}$ recognizing the domain of f, one obtains a canonical transducer for f, as long as $\mathcal{D}$ can be obtained in a canonical way (such a procedure exists, see *e.g.* [7]).

(2) Our main contribution (Theorem 29) is a notion of canonical sequential transducer with look-ahead for any rational function. This canonical transducer is an effectively computable bimachine. Hence we lift results of Reutenauer and Schützenberger [20] on rational transductions from finite to infinite words.

(3) As a side result we lift a result by Elgot and Mezei [11] from finite to infinite words, stating that a function f is rational if and only if $f = g_1 \circ h_1$ (resp. $f = g_2 \circ h_2$) such that h_1, h_2 are letter-to-letter, g_1, h_2 are sequential and h_1, g_2 are right-sequential (*i.e.*, realized by a transducer whose underlying input automaton is prophetic [6]). The existence of such g_1, h_1 was already shown in [5], but the one of g_2, h_2 was left open.

(4) Finally, we show that our procedure which computes a canonical bimachine for any rational function given by a transducer, preserves aperiodicity. As an application, after showing some correspondences between logics and transducers, we obtain the decidability of FO-transductions in MSO-transductions over infinite words.

Overview of the canonization procedure for rational functions. The main idea to get a canonical object for a rational function, inspired by [20], is to add a canonical look-ahead information to the input word, so that the function can be evaluated in a sequential (equivalently, deterministic) manner. We say that the look-ahead "makes the function sequential". By doing so, we can reduce the problem to computing canonical machines for sequential functions. The main difficulty is to define a canonical (and computable) notion of look-ahead which makes the function sequential. Over finite words, the look-ahead information is computed by a co-deterministic automaton, or equivalently, a deterministic automaton reading the input word from right to left (called a right automaton). On infinite words we

need something different, so we use prophetic automata [6] to define look-aheads (called right automata in this paper). Prophetic automata are a special form of co-deterministic automata over infinite words. In Section 3, sequential transducers with look-ahead are formalized via the notion of bimachines, consisting of a left automaton and a right automaton. We show that bimachines over infinite words capture exactly the class of rational transductions. Our goal is to obtain a canonical bimachine, fine enough to realize the function, but coarse enough to preserve algebraic properties like aperiodicity.

Unlike the setting of finite words, some difficulties arise when prefix-independent properties matter (such as for instance that a suffix contains an infinite number of a's). We overcome this issue by defining two kinds of look-ahead information which we combine later on. This decomposition simplifies the overall proof.

The first look-ahead information we define allows one to make any rational function almost sequential, in the sense that it can be implemented by a transducer model which can additionally output some infinite word after processing the whole input, depending on the run (similar to so-called *subsequential* transducers in the case of finite words). We call *quasi-sequential* functions realized by such transducers. They constitute a class with interesting properties. We show that they correspond precisely to transducers satisfying the weak twinning property, a syntactic condition defined in [2]. On the algebraic side, we exhibit a congruence having finite index exactly for quasi-sequential functions.

We then define another kind of canonical look-ahead which makes any quasi-sequential function sequential. Combined together, these two look-aheads turn any rational function into a sequential one: the first one from rational to quasi-sequential, and the second one from quasi-sequential to sequential.

The whole procedure does not yield a minimal bimachine in general. While the minimality question is an important and interesting (open) question, our procedure still has the strong advantages of being canonical, effective, and of preserving aperiodicity. This allows one to answer positively the important question of the decidability of first-order definability for rational functions of infinite words. Detailed proofs are provided in a long version of this paper, available online.

1 Regular languages and rational functions

Finite words, infinite words and languages. An *alphabet* A is a finite set of symbols called *letters*. A *finite word* is a finite sequence of letters, the empty sequence is called the *empty word* and is denoted by ϵ. The set of (resp. non-empty) finite words over A is denoted by A^* (resp. A^+). An infinite sequence of letters is called an *ω-word* (or just an infinite word), we denote by A^ω the set of ω-words and we write $A^\infty = A^* \cup A^\omega$. For a word $x \in A^\omega$ we denote by $\mathrm{Inf}(x)$ the set of letters of x which appear an infinite number of times. The *length* of a word w is written $|w|$, with $|w| = \infty$ if $w \in A^\omega$. Throughout the paper, we often denote finite words by $u, v, \ldots$ and infinite words by $x, y \ldots$

For a non-empty word w and two integers $1 \leq i \leq j \leq |w|$ we denote by $w(i)$ the ith letter of w, by $w(i{:})$ the suffix of w starting at the ith position, by $w({:}i)$ the prefix of w ending at the ith position and by $w(i{:}j)$ the infix of w starting at the ith position and ending at the jth, both included. For two words $u \in A^*$ and $v \in A^\infty$, we write $u \prec v$ if u is a strict prefix of v, *i.e.* there exists a non-empty word $w \in A^\infty$ such that $uw = v$, and we write $u^{-1}v$ for w. For $u, v \in A^\infty$, we write $u \preceq v$ if either $u \prec v$, or $u = v$. We denote by $u \wedge v$ the longest common prefix of u and v. The *delay* $\mathsf{del}(u, v)$ between two words $u, v \in A^\infty$ is the unique pair (u', v') such that $u = (u \wedge v)u'$ and $v = (u \wedge v)v'$. For example, $\mathsf{del}(aab, ab) = (ab, b)$ and $\mathsf{del}(a^\omega, a^\omega) = (\epsilon, \epsilon)$.

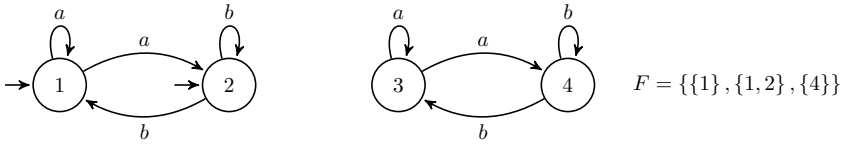

Figure 3 A right automaton (with Muller condition) recognizing $(b^*a)^\omega$. Words with finitely many b's have final run with $\{1\}$, words with finitely many a's have final run with $\{4\}$, and those with infinitely many a's and infinitely many b's have final run with $\{1,2\}$.

A *language* is a set of words $L \subseteq A^\infty$, and by $\bigwedge L$ we denote the longest common prefix of all words in L (if $L \neq \varnothing$). The closure $\overline{L}$ of L is $\{u \in A^\infty \,|\, \forall i \in \mathbb{N},\ i \leq |u|,\ \exists w \text{ s.t. } u(:i)w \in L\}$, *i.e.* the set of words for which any finite prefix has a continuation in L. For instance $\overline{a^*b^\omega} = a^*b^\omega \cup a^\omega$. A word is called *regular* if it is of the form uv^ω with $u, v \in A^*$. In particular any finite word is regular (since $\epsilon^\omega = \epsilon$) and regular ω-words are also called *ultimately periodic*. We say that a regular word uv^ω is in *normal form* if v has minimal length and is minimal in the lexicographic order among all possible decompositions of uv^ω, and v is not a suffix of u (if $v \neq \epsilon$). *E.g.* the normal form of $(ba)^\omega$ is $b(ab)^\omega$. In the sequel we often assume regular words are in normal form.

Automata. A Muller[1] automaton over an alphabet A is a tuple $\mathcal{A} = (Q, \Delta, I, F)$ where Q is a finite set of *states*, $\Delta \subseteq Q \times A \times Q$ is the set of *transitions*, $I \subseteq Q$ is the set of *initial* states, and $F \subseteq \mathcal{P}(Q)$ is called the *final condition*. When there is no final condition, so $F = \mathcal{P}(Q)$, we will omit it. A *run* of $\mathcal{A}$ over a word $w \in A^\infty$ is itself a word $r \in Q^\infty$ of length $|w| + 1$, (with the convention that $\infty + 1 = \infty$) such that for any $1 \leq i < |r|$, we have $(r(i), w(i), r(i+1)) \in \Delta$. A run r is called *initial* if $r(1) \in I$, *final* if $r \in Q^\omega$ and $\mathrm{Inf}(r) \in F$, and *accepting* if it is both initial and final. For a finite word u and two states p, q, we write $p \xrightarrow{u}_{\mathcal{A}} q$ to denote that there is a run r of $\mathcal{A}$ over u such that $r(1) = p$ and $r(|r|) = q$. For an ω-word x, a state p and a subset of states $P \subseteq Q$, we write $p \xrightarrow{x}_{\mathcal{A}} P$ to denote that there is a run r of $\mathcal{A}$ over x such that $r(1) = p$ and $\mathrm{Inf}(r) = P$. A word is *accepted* by $\mathcal{A}$ if there exists an accepting run over it, and the language *recognized* by $\mathcal{A}$ is the set of words it accepts, denoted by $[\![\mathcal{A}]\!] \subseteq A^\omega$. A state p is *accessible* (resp. *co-accessible*) if there exists a finite initial (resp. infinite final) run r such that $r(|r|) = p$ (resp. $r(1) = p$), and an automaton $\mathcal{A}$ is called *accessible* (resp. *co-accessible*) if all its states are. An automaton which is both accessible and co-accessible is called *trim*.

An automaton is called *deterministic* if its set of initial states is a singleton, and any word has at most one initial run. We define a *left automaton* as a deterministic automaton $\mathcal{L} = (Q, \Delta, I)$ with no acceptance condition. We call a *right automaton* an automaton for which any ω-word has exactly one final run[2]. A language is called *ω-regular* if it is recognized by an automaton. It is well-known that every ω-regular language can be recognized by a deterministic (Muller) automaton. Moreover, [6] shows that every ω-regular language can be recognized by a right automaton (even with Büchi condition). Figure 3 shows a right automaton accepting the words with infinitely many a's. Throughout the paper, all automata – except for right automata – are assumed trim, without loss of generality.

[1] We consider the Muller condition since it is more general than Büchi or parity for instance, but most of our results hold for other conditions as well.

[2] Such automata are called prophetic and were introduced in [6].

Transductions. Given two alphabets A, B, we call *transduction* a relation $R \subseteq A^\omega \times B^\infty$ whose domain is denoted by $\mathrm{dom}(R)$. A transducer over A, B is a tuple $\mathcal{T} = (\mathcal{A}, i, o)$ with $\mathcal{A} = (Q, \Delta, I, F)$ the *underlying automaton*, $i : I \to B^*$ the *initial function* and $o : \Delta \to B^*$ the *output function*.

Let u be a finite word of length n, let r be a run of $\mathcal{A}$ over u with $r(1) = p, r(n+1) = q$, and let v be the word $o(p, u(1), r(2)) \cdots o(r(n), u(n), q)$ then we write $p \xrightarrow{u|v}_{\mathcal{T}} q$ to denote that fact. Similarly, for $p \in Q$ and $P \subseteq Q$ we write $p \xrightarrow{x|v}_{\mathcal{T}} P$ to denote that there is a run r of $\mathcal{A}$ over the ω-word x such that $r(1) = p$, $\mathrm{Inf}(r) = P$ and $v = o(p, u(1), r(2))o(r(2), u(2), r(3)) \cdots$. In that case, if $p \in I$ and $P \in F$, let $w = i(p) \cdot v$, then we say that the pair (x, w) is *realized* by $\mathcal{T}$. We denote by $[\![\mathcal{T}]\!]$ the set of pairs realized by $\mathcal{T}$, which we call the *transduction realized* by $\mathcal{T}$. A transducer is called *functional* if it realizes a (partial) function, and in that case we write $[\![\mathcal{T}]\!](x) = w$ rather than $(x, w) \in [\![\mathcal{T}]\!]$. Functionality is a decidable property, see *e.g.* [16], and it can be checked in PTime (see *e.g.* [19]). *In the following all the transductions we consider are functional, and when we speak about functions, we tacitly assume that they are partial.* A transduction is *rational* if it is realized by a transducer. A transducer with a deterministic underlying automaton is called *sequential*, as well as the function it realizes. A transducer with a left (resp. right) underlying automaton is called *left-sequential* (resp. *right-sequential*), and again we extend this terminology to the function it realizes.

Congruences. Given an equivalence relation $\sim$ over a set L, we denote by $[w]^\sim$ (or simply $[w]$) the equivalence class of an element $w \in L$. We say that $\sim$ has *finite index* if the set $L/_\sim = \{[w] \mid w \in L\}$ is finite. Given two equivalence relations $\sim_1, \sim_2$ over the same set we say that $\sim_1$ is *finer* than $\sim_2$ (or that $\sim_2$ is *coarser* than $\sim_1$) if for any u, v we have $u \sim_1 v \Rightarrow u \sim_2 v$. Equivalently we could say that the equivalence classes of $\sim_2$ are unions of equivalence classes of $\sim_1$ or that $\sim_1$ is included (as a set of pairs) in $\sim_2$, which we denote by $\sim_1 \sqsubseteq \sim_2$. A *right congruence* over A^* is an equivalence relation $\sim$ such that for any letter a and any words u, v we have $u \sim v \Rightarrow ua \sim va$. A *left congruence* over A^* (resp. A^ω) is an equivalence relation $\approx$ such that for any letter a and any words u, v we have $u \approx v \Rightarrow au \approx av$. We say that a left congruence is *regular* if it has finite index and any equivalence class is an ω-regular language. In the following all the left congruences will be regular. A *congruence* over A^* is a left and right congruence. A congruence $\equiv$ is *aperiodic* if there exists an integer n such that $\forall u \in A^*$, $u^n \equiv u^{n+1}$.

Given an automaton $\mathcal{A}$ with state space Q, the *right congruence associated with* $\mathcal{A}$ is defined for $u, v \in A^*$ by $u \sim_{\mathcal{A}} v$ if $\forall q \in Q$, there is an initial run of $\mathcal{A}$ over u reaching q if and only if there is one over v. Note that for a trim deterministic automaton, there is a bijection (up to adding a sink state) between Q and the equivalence classes of $\mathcal{A}$. Similarly, the *left congruence associated with* $\mathcal{A}$ is defined for $x, y \in A^\omega$ by $x \approx_{\mathcal{A}} y$ if $\forall q \in Q$ there is a final run of $\mathcal{A}$ over x from q if and only if there is one over y. Given a right automaton there is a bijection between Q and the equivalence classes of $\approx_{\mathcal{A}}$. Finally, the *transition congruence of* $\mathcal{A}$ is defined for $u, v \in A^*$ by $u \equiv_{\mathcal{A}} v$ if $\forall p, q \in Q$, there is a run over u from p to q if and only if there is one over v. An automaton is called *aperiodic* if its transition congruence is aperiodic. A language is called *aperiodic* if there exists an aperiodic automaton recognizing it. A transducer is *aperiodic* if its underlying automaton is aperiodic and in that case the transduction it realizes is called *aperiodic*.

Given a right congruence $\sim$, the *left automaton associated with* $\sim$ is $\mathcal{A}_\sim = (Q_\sim, \Delta_\sim, I_\sim)$: $Q_\sim = A^*/_\sim$, $\Delta_\sim = \{([u], a, [ua]) \mid u \in A^*\}$, $I_\sim = \{[\epsilon]\}$. Given a left congruence $\approx$ and a right automaton $\mathcal{R}$, if $\approx_{\mathcal{R}} \sqsubseteq \approx$ then we say that $\mathcal{R}$ *recognizes* $\approx$. The existence of a canonical automaton for a left congruence is less obvious. From [6] we know that every ω-regular

language can be recognized by a right automaton. We rely on the construction of [6] and, abusing language, we denote the right automaton obtained in the next proposition as the *canonical right automaton recognizing a left congruence*:

▶ **Proposition 1.** Given a (regular) left congruence, we can compute in 2-ExpTime a right automaton recognizing it. Furthermore, this automaton is aperiodic if the congruence is aperiodic.

2 Sequential and quasi-sequential transductions

We define the syntactic congruence associated with any functional transduction over infinite words. Sequential functions are exactly the rational functions having a syntactic congruence of finite index, and being continuous over their domain. When removing this last condition on continuity, we obtain the class of quasi-sequential transductions. These transductions are also characterized by the weak twinning property [2].

We will show that for any sequential function, like in the case of finite words [8], we can define a canonical transducer, with a minimal underlying automaton. This minimal transducer extends the domain of the function to its closure.

▶ **Definition 2** ($\widehat{f}$ and $\overline{f}$)**.** Let $f : A^\omega \to B^\infty$ be a function, we define $\widehat{f} : A^* \to B^\infty$ by $\widehat{f}(u) = \bigwedge \{f(ux) \mid ux \in \mathrm{dom}(f)\}$. In other words, $\widehat{f}$ outputs the longest possible output that f could produce on any word that begins with u. We also define $\overline{f} : A^\omega \to B^\infty$ by setting $\overline{f}(x) = \lim_n \widehat{f}(x(:n))$, for $x \in \overline{\mathrm{dom}(f)}$.

We refer to $\overline{f}$ as the *sequential extension* of f. Note that if f is sequential, then $\overline{f}$ extends f over the closure $\overline{\mathrm{dom}(f)}$ of the domain of f.

▶ **Example 3.** We illustrate these definitions on three rational transductions, described in Table 1.

▶ **Definition 4** (syntactic congruence $\sim_f$)**.** The *syntactic congruence* associated with a transduction f is defined over A^* by $u \sim_f v$ if:

1. $\forall x \in A^\omega$, $ux \in \overline{\mathrm{dom}(f)} \Leftrightarrow vx \in \overline{\mathrm{dom}(f)}$, and
2. either $\widehat{f}(u)$ and $\widehat{f}(v)$ are both ultimately periodic with the same period (in normal form) or they are both finite and $\forall x \in A^\omega$ such that $ux, vx \in \mathrm{dom}(f)$, $\widehat{f}(u)^{-1} f(ux) = \widehat{f}(v)^{-1} f(vx)$.

▶ **Example 5.** Let us illustrate the definition of $\sim_f$ on f_{ab}, as defined in Table 1. The syntactic congruence $\sim_{f_{ab}}$ has only two classes: $[\epsilon]$ and $[a]$. Indeed, if we consider two finite words u and v, condition (1) on the domain is always true, and $\widehat{f_{ab}}(u)$ and $\widehat{f_{ab}}(v)$ are finite (ab-factors in u and v, respectively). Hence $u \sim_{f_{ab}} v$ if and only if $\forall x \in A^\omega$, $\widehat{f_{ab}}(u)^{-1} f_{ab}(ux) = \widehat{f_{ab}}(v)^{-1} f_{ab}(vx)$.

Let us analyze $\widehat{f_{ab}}(u)^{-1} f_{ab}(ux)$. If u does not end with an a, then $\widehat{f_{ab}}(u)^{-1} f_{ab}(ux) = ((ab)^n)^{-1}((ab)^{n+k}) = (ab)^k$ where n and k are the number of ab-factors in u and x, respectively. Now, if u ends with an a and x starts with a b, then a new ab-factor appears in ux and we get $\widehat{f_{ab}}(u)^{-1} f_{ab}(ux) = ((ab)^n)^{-1}((ab)^{n+k+1}) = (ab)^{k+1}$. This means that $\sim_{f_{ab}}$ contains exactly two classes: one for the words ending with an a, and one for the others.

The resulting transducer $\mathcal{T}_{f_{ab}}$ is depicted in Figure 4. Let us check for instance the transition from $[a]$ to $[\epsilon]$ when reading b. We have $[ab] = [\epsilon]$, so $([a], b, [\epsilon]) \in \Delta_{f_{ab}}$. From the definition, $o_{f_{ab}}([a], b, [\epsilon]) = \widehat{f_{ab}}(a)^{-1} \widehat{f_{ab}}(ab) = \epsilon^{-1}.ab = ab$.

▶ **Proposition 6.** Let f be a functional transduction, then $\sim_f$ is a right congruence.

Table 1 Examples of rational transductions, and their associated $\widehat{f}$ and $\overline{f}$ functions.

	f_{ab}	$f_{\#a}$	f_{blocks}
definition	maps a word over $\{a,b\}$ with a finite number of a's to the subsequence of ab-factors.	maps a word x over $\{a,b\}$ to a^ω if x contains an infinite number of a's, and to b^ω otherwise.	maps $u_1\#\dots\#u_n\#v$ where v does not contain $\#$, to $a_1^{\lvert u_1\rvert}\#\dots\#a_n^{\lvert u_n\rvert}\#w$ where $u_i \in \{a,b\}^*$, a_i is the last letter of u_i (if any), $w = a^\omega$ if v has an infinite number of a's, and $w = b^\omega$ otherwise.
A and B	$A = B = \{a,b\}$	$A = B = \{a,b\}$	$A = B = \{a,b,\#\}$
$\text{dom}(f)$	words over $\{a,b\}$ with a finite number of a's	$\{a,b\}^\omega$	words over $\{a,b,\#\}$ with a finite (non-zero) number of $\#$'s
examples	$f_{ab}(abbab^\omega) = abab$, $f_{ab}(b^\omega) = \epsilon$	$f_{\#a}(ab^\omega) = b^\omega$, $f_{\#a}((ab)^\omega) = a^\omega$	$f_{\text{blocks}}((ab\#)^n b^\omega) = (bb\#)^n b^\omega$, $f_{\text{blocks}}(\#(ab)^\omega) = a^\omega$.
$\widehat{f}$	$\widehat{f_{ab}}$ extracts the ab-factors, for instance $\widehat{f_{ab}}(abbabb) = abab$.	reading a finite prefix u does not give any insight on the output, thus $\widehat{f_{\#a}}(u) = \epsilon$	$\widehat{f_{\text{blocks}}}(u_1\#\dots\#u_n\#v) = a_1^{\lvert u_1\rvert}\#\dots\#a_n^{\lvert u_n\rvert}\#$ whenever v does not contain $\#$.
$\overline{f}$	$\overline{f_{ab}}$ is defined over $\text{dom}(\overline{f_{ab}}) = \{a,b\}^\omega$ and $\overline{f_{ab}}((ba)^\omega) = \lim_n \widehat{f_{ab}}((ba)^n) = \lim_n (ab)^{n-1} = (ab)^\omega$.	$\overline{f_{\#a}}(x) = \epsilon$ for every $x \in \{a,b\}^\omega$ as it is based on $\widehat{f_{\#a}}$	$\overline{f_{\text{blocks}}}(u_1\#\dots\#u_n\#v) = a_1^{\lvert u_1\rvert}\#\dots\#a_n^{\lvert u_n\rvert}\#$ whenever v does not contain $\#$.
class	sequential	quasi-sequential	not quasi-sequential

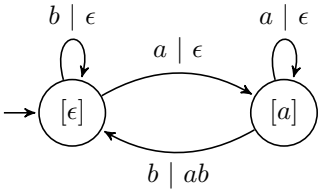

Figure 4 Transducer $\mathcal{T}_{f_{ab}}$.

From $\sim_f$ we define[3] the transducer $\mathcal{T}_f = (\mathcal{A}_f, i_f, o_f)$ with $\mathcal{A}_f = (Q_f, \Delta_f, I_f)$ and:

- $Q_f = A^*/_{\sim_f}$ and $I_f = \{[\epsilon]\}$
- $\Delta_f = \{([u], a, [ua]) \mid u \in A^*, a \in A, \exists x \text{ s.t. } uax \in \text{dom}(f)\}$
- $o_f([u], a, [ua]) = \begin{cases} \widehat{f}(u)^{-1}\widehat{f}(ua) & \text{if } \widehat{f}(ua) \text{ is finite} \\ \beta & \text{if } \widehat{f}(u) = \alpha\beta^\omega,\ \beta \neq \epsilon \\ \alpha & \text{if } \widehat{f}(u) \text{ is finite and } \widehat{f}(u)^{-1}\widehat{f}(ua) = \alpha\beta^\omega,\ \beta \neq \epsilon \end{cases}$
- $i_f([\epsilon]) = \begin{cases} \widehat{f}(\epsilon) & \text{if } \widehat{f}(\epsilon) \text{ is finite} \\ \alpha & \text{if } \widehat{f}(\epsilon) = \alpha\beta^\omega,\ \beta \neq \epsilon \end{cases}$

▶ Remark. Note that, in general, $\sim_f$ may have an infinite index, thus $\mathcal{T}_f$ may be infinite. This is the case for f_{blocks}: for two words $u = u_0\#w$ and $v = u_0\#w'$ with u_0ww' not containing $\#$, $u \sim_{f_{\text{blocks}}} v$ if and only if $|w| = |w'|$ and they end with the same letter. We will define below a subclass of rational transductions, which captures exactly finite $\sim_f$ (Theorem 12).

[3] We check in the long version that $\mathcal{T}_f$ is well-defined.

As shown below, the sequential transducer $\mathcal{T}_f$ computes the sequential extension $\overline{f}$ of f. If f is sequential then f and $\overline{f}$ coincide on $\mathrm{dom}(f)$.

▶ **Proposition 7.** Given a function f, the transducer $\mathcal{T}_f$ realizes $\overline{f}$.

We now focus on sequential transductions, and show first that $\mathcal{T}_f$ can be built in PTime.

▶ **Proposition 8.** There is a PTime algorithm that, for a given sequential transducer $\mathcal{T}$ realizing the function f, computes the transducer $\mathcal{T}_f$.

For sequential transductions we get a characterization, as stated in the next theorem. We will see that the first condition is equivalent to the weak twinning property. Thus, the next theorem adapts a result from [2] to the case where transducers may output finite words.

▶ **Theorem 9.** *A rational function f is sequential if and only if the following conditions hold:*

- $\sim_f$ *has finite index*
- $\overline{f}_{|dom(f)} = f$

If we remove the last restriction $\overline{f}_{|\mathrm{dom}(f)} = f$ in Theorem 9, we obtain a class of transductions where the output can be still generated deterministically (as for sequential transductions), although not necessarily in a progressive manner:

▶ **Definition 10.** A function f is called *quasi-sequential* if it is rational and $\sim_f$ has finite index.

Intuitively, quasi-sequential functions generalize the so-called *subsequential* functions on finite words to infinite words. For subsequential functions there is a final output associated with final states. Quasi-sequential functions can be shown to correspond to sequential transducers where final sets may have an associated word in A^∞. The output of an accepting run with such a final set is obtained by appending the associated word to the output word obtained through the transitions (if finite). Since we do not use this model in the present paper, we do not provide more details in the following. The following property and construction are now taken directly from [2]. As in the latter article, a state is called *constant* if the set of words produced by final runs from this state is a singleton.

▶ **Definition 11** (weak twinning property)**.** A transducer $\mathcal{T}$ is said to satisfy the *weak twinning property* (WTP) if for any initial runs $p_1 \xrightarrow{u|\alpha_1} q_1 \xrightarrow{v|\beta_1} q_1$ and $p_2 \xrightarrow{u|\alpha_2} q_2 \xrightarrow{v|\beta_2} q_2$ the following property holds:

- If q_1, q_2 are not constant then $\mathsf{del}(i(p_1)\alpha_1, i(p_2)\alpha_2) = \mathsf{del}(i(p_1)\alpha_1\beta_1, i(p_2)\alpha_2\beta_2)$
- If q_1 is not constant, q_2 is constant and produces the regular word γ, then either $\beta_1 = \epsilon$ or $i(p_1)\alpha_1\beta_1^\omega = i(p_2)\alpha_2\beta_2\gamma$

Note that if q_2 is constant and $\beta_2 \neq \epsilon$ then $\gamma = \beta_2^\omega$.

The authors of [2] provide a determinization procedure – which we call *subset construction with delays*– which terminates if and only if the transducer satisfies the WTP. We show that actually the procedure gives a transducer realizing the sequential extension of the function and we use this fact in Sec. 4 in order to compute a canonical look-ahead.

▶ **Theorem 12.** *Let $\mathcal{T}$ be a transducer realizing a function f, let $\mathcal{S}$ be the transducer obtained by subset construction with delays. The following statements are equivalent:*

1. *The transducer $\mathcal{T}$ satisfies the WTP*
2. *The transducer $\mathcal{S}$ is finite*
3. *f is quasi-sequential*

Furthermore, if $\mathcal{T}$ is aperiodic then $\mathcal{S}$ is aperiodic as well.

3 Rational transductions

Bimachines over infinite words. A *bimachine* over alphabets A, B is a tuple $\mathcal{B} = (\mathcal{L}, \mathcal{R}, i, o)$ where $\mathcal{L} = (Q_\mathcal{L}, \Delta_\mathcal{L}, \{l_0\})$ is a left automaton, $\mathcal{R} = (Q_\mathcal{R}, \Delta_\mathcal{R}, I, F)$ is a right automaton, $i : I \to B^*$ is the *initial function* and $o : Q_\mathcal{L} \times A \times Q_\mathcal{R} \to B^*$ is the *output function*. We have a semantic restriction that $[\![\mathcal{L}]\!] = \overline{[\![\mathcal{R}]\!]}$. The output produced on an infinite word $w \in [\![\mathcal{R}]\!]$ at position $i \geq 1$ is $\alpha_i = o(l, a, r)$, where l is the state reached in $\mathcal{L}$ after reading the prefix $w(:i-1)$ of w up to position $i-1$ (if defined), r is the state of the unique final run of $\mathcal{R}$ on w (if defined) reached by the suffix $w(i+1:)$ of w from position $i+1$ on, and $a = w(i)$. In other words, the output at position i is determined by the left context up to position $i-1$, the right context from position $i+1$ onwards, and the letter at position i. The output produced on w is $i(r_0)\alpha_1\alpha_2\cdots$, with $r_0 \in I$ the state from which there is a final run of $\mathcal{R}$ on w (if defined). Thus, the right automaton $\mathcal{R}$ provides a look-ahead and the output depends both on the state of $\mathcal{L}$ and the unique final run of $\mathcal{R}$ on the given word. The transduction *realized* by $\mathcal{B}$ is denoted by $[\![\mathcal{B}]\!]$. Note that $[\![\mathcal{B}]\!]$ is defined over $[\![\mathcal{R}]\!]$. A bimachine is called *aperiodic* if both its automata are aperiodic.

▶ **Example 13.** Let us define a bimachine for f_{ab}, the function that outputs ab-factors of the input over $\{a, b\}$, if this input has a finite number of a's. We use as left automaton the underlying automaton of the transducer in Figure 2, without its Muller acceptance condition. This automaton will only be used to store the last letter read. The domain has to be checked by the right automaton, and we choose the one in Figure 3. As output functions, we let $i(q) = \epsilon$ for the initial states of the right automaton, and let $o(q_a, b, r) = ab$ for $r \in \{1, 2\}$, and $o(l, c, r) = \epsilon$ for all other states l, r of the left and right automata, and letter $c \in \{a, b\}$.

Left minimization. We show how to minimize the left automaton of a bimachine with respect to a right automaton $\mathcal{R}$. The procedure is very similar to the minimization for sequential transducers. The objects we use are the same as in Section 2, but relativized to the right context defined by the look-ahead provided by the right automaton $\mathcal{R}$. The bimachine with minimal left automaton with respect to the right automaton $\mathcal{R}$ is the bimachine $\mathcal{B}_f^\mathcal{R}$ defined below.

Recall that the left congruence $\approx_\mathcal{R}$ of a right automaton $\mathcal{R}$ sets $x \approx_\mathcal{R} y$ if the unique state from which there is a final run on x is the same as for y. Let $f : A^\omega \to B^\infty$ be a function and let $\mathcal{R} = (Q_\mathcal{R}, \Delta_\mathcal{R}, I, F)$ be a right automaton recognizing $\mathrm{dom}(f)$. We write $[x]^\mathcal{R}$ for the class of a word x with respect to $\approx_\mathcal{R}$, and, abusing notations, for the state of $Q_\mathcal{R}$ from which words of $[x]^\mathcal{R}$ have a final run. We define $\widehat{f}_x : A^* \to B^\infty$ by setting $\widehat{f}_x(u) = \bigwedge \{f(uy) \mid y \approx_\mathcal{R} x\}$. Note that there are finitely many functions $\widehat{f}_x$, one for each equivalence class of $\approx_\mathcal{R}$. We also define $\overline{f}^\mathcal{R} : A^\omega \to B^\infty$, by setting $\overline{f}^\mathcal{R}(x) = \lim_n \widehat{f}_{x(n+1:)}(x(:n))$. The transduction $\overline{f}^\mathcal{R}$ is defined over $\mathrm{dom}(f)$.

▶ **Definition 14** ($\mathcal{R}$-syntactic congruence)**.** The *$\mathcal{R}$-syntactic congruence of f* is defined over A^* by letting $u \sim_f^\mathcal{R} v$ if:

1. $\forall x \in A^\omega$, $ux \in \mathrm{dom}(f) \Leftrightarrow vx \in \mathrm{dom}(f)$, and
2. for any $x \in A^\omega$, either $\widehat{f}_x(u)$ and $\widehat{f}_x(v)$ are both infinite with the same ultimate period (in normal form) or they are both finite and $\widehat{f}_x(u)^{-1} f(ux) = \widehat{f}_x(v)^{-1} f(vx)$.

Similarly to the sequential case, we define from $\sim_f^\mathcal{R}$ a bimachine $\mathcal{B}_f^\mathcal{R} = \left(\mathcal{L}_f^\mathcal{R}, \mathcal{R}, i_f^\mathcal{R}, o_f^\mathcal{R}\right)$ with right automaton $\mathcal{R}$, and left automaton $\mathcal{L}_f^\mathcal{R} = \left(Q_f^\mathcal{R}, \Delta_f^\mathcal{R}, I_f^\mathcal{R}\right)$ corresponding to $\sim_f^\mathcal{R}$. To simplify notations we denote the congruence class of a word u with respect to $\sim_f^\mathcal{R}$ by $[u]$. Abusing notations we also write $[x]^\mathcal{R}$ for the state of $\mathcal{R}$ from which x has an accepting run.

- $Q_f^{\mathcal{R}} = A^*/_{\sim_f^{\mathcal{R}}}$ and $I_f^{\mathcal{R}} = \{[\epsilon]\}$
- $\Delta_f^{\mathcal{R}} = \{([u], a, [ua]) \mid u \in A^*, a \in A, uax \in \mathrm{dom}(f) \text{ for some } x \in A^\omega\}$
- $o_f([u], a, [x]^{\mathcal{R}}) = \begin{cases} \widehat{f}_{ax}(u)^{-1}\widehat{f}_x(ua) & \text{if } \widehat{f}_x(ua) \text{ is finite} \\ \beta & \text{if } \widehat{f}_{ax}(u) = \alpha\beta^\omega,\ \beta \neq \epsilon \\ \alpha & \text{if } \widehat{f}_{ax}(u) \text{ is finite, } \widehat{f}_{ax}(u)^{-1}\widehat{f}(ua) = \alpha\beta^\omega \\ & \quad \text{and } \beta \neq \epsilon \end{cases}$
- $i_f([x]^{\mathcal{R}}) = \begin{cases} \widehat{f}_x(\epsilon) & \text{if } \widehat{f}_x(\epsilon) \text{ is finite} \\ \alpha & \text{if } \widehat{f}_x(\epsilon) = \alpha\beta^\omega, \beta \neq \epsilon \end{cases}$

We show in the long version of this paper that $\mathcal{B}_f^{\mathcal{R}}$ is well-defined, and exhibit some of its properties. We also describe in the long version a polynomial time algorithm that computes $\mathcal{B}_f^{\mathcal{R}}$ from a bimachine with right automaton $\mathcal{R}$, with a technique similar to the sequential case.

From transducers to bimachines. For the theorem below, recall that $\sim_{\mathcal{A}}$ denotes the right congruence of an automaton $\mathcal{A}$. The left congruence $\approx_{\mathcal{A}}$ of an automaton $\mathcal{A}$ sets $x \approx_{\mathcal{A}} y$ if for every state q of $\mathcal{A}$, there is some final run on x from q if and only if there is one on y.

▶ **Theorem 15.** *Given a transducer with underlying automaton $\mathcal{A}$ and a right automaton $\mathcal{R}$ with $\approx_{\mathcal{R}} \sqsubseteq \approx_{\mathcal{A}}$. Then $\sim_{\mathcal{A}} \sqsubseteq \sim_f^{\mathcal{R}}$ and the bimachine $\mathcal{B}_f^{\mathcal{R}}$ realizes f.*

In particular any aperiodic transduction can be realized by an aperiodic bimachine.

The other direction also holds: from a bimachine we can build an equivalent (unambiguous) transducer, by taking the product of the left and right automata of the bimachine. The construction is not hard but given in the long version. By Theorem 15 and Proposition 1 we obtain:

▶ **Theorem 16.** *A function is rational (resp. rational and aperiodic) if and only if it can be realized by a bimachine (resp. aperiodic bimachine).*

Labelings and bimachines. We define the *labeling function* associated with a right automaton $\mathcal{R} = (Q, \Delta, I, F)$ by the right transducer $\ell(\mathcal{R}) = (\mathcal{R}, i, o)$, with $i(q) = \epsilon$ and $o(p, a, q) = (a, q)$. Intuitively, the labeling function labels each position with the look-ahead information about the suffix provided by $\mathcal{R}$. For a transduction f we define $f_{\mathcal{R}} = f \circ [\![\ell(\mathcal{R})]\!]^{-1}$. Note that $f_{\mathcal{R}}$ is a function, since the labeling is injective (because $\mathcal{R}$ is unambiguous). Thus, $f_{\mathcal{R}}$ corresponds to f defined over words enriched by the look-ahead information of $\mathcal{R}$.

▶ **Proposition 17.** Let f be a transduction and let $\mathcal{R}$ be a right automaton. There exists a bimachine $\mathcal{B}$ realizing f with $\mathcal{R}$ as a right automaton if and only if $f_{\mathcal{R}}$ is left-sequential. Furthermore, assuming that $\mathcal{R}$ is aperiodic, then $\sim_f^{\mathcal{R}}$ is aperiodic if and only if $f_{\mathcal{R}}$ is aperiodic.

We say that a transducer is *letter-to-letter* if its initial output function always outputs the empty word and its output function always outputs a single letter. The following corollary states the classical result of [11] but over infinite words, and generalizes a result of [5].

▶ **Corollary 18.** *For any rational function f, there exists a left-sequential (right-seq. resp.) function g and a letter-to-letter right-sequential (left-seq. resp.) function h such that $f = g \circ h$.*

4 Canonical machines

The goal of this section is to define a canonical bimachine for any rational function. By canonicity we mean that it should be machine-independent. Our ultimate goal is to show that the canonical bimachine suffices to decide the algebraic properties we are interested in. To get a canonical bimachine, we need a right automaton for the look-ahead that is 1) canonical, 2) coarse-grained enough to preserve algebraic properties, and 3) fine-grained enough to obtain a deterministic left automaton (and hence a bimachine).

We define the delay congruence and show that it is the coarsest left congruence such that any automaton $\mathcal{R}$ recognizing it satisfies that $f_{\mathcal{R}}$ is quasi-sequential (Proposition 21). However, this congruence is, in general, too coarse to make $f_{\mathcal{R}}$ sequential. We then introduce the ultimate congruence, and show how to combine these two congruences to build a canonical bimachine.

Let f be a transduction. We define the *delay* between $x, y \in A^\omega$ with respect to f by: $\mathsf{del}_f(x, y) = \{\mathsf{del}(f(ux), f(uy)) \mid ux, uy \in \mathrm{dom}(f)\}$. The following definition is taken from [20, 3].

▶ **Definition 19** (delay congruence)**.** The *delay congruence* of f is defined by setting $x \overset{\Delta}{\approx}_f y$ for $x, y \in A^\omega$ if (1) for all $u \in A^*$, $ux \in \mathrm{dom}(f) \Leftrightarrow uy \in \mathrm{dom}(f)$, and (2) $|\mathsf{del}_f(x, y)| < \infty$.

▶ **Example 20.** Let us illustrate the above definition on f_{blocks} (recall Example 3). We consider $x = u_1\#\dots\#u_n\#v$ and $y = u'_1\#\dots\#u'_n\#v'$ where v and v' are infinite words not containing $\#$. Note that $x \overset{\Delta}{\approx}_{f_{\mathrm{blocks}}} y$ if and only if u_1, u'_1 are either both empty, or end with the same letter. Indeed, if the latter holds then $\mathsf{del}(f(ux), f(uy)) = \mathsf{del}(f(x), f(y))$. Conversely, if both u_1, u'_1 are non-empty but end with different letters, then for any u without $\#$, $\mathsf{del}(f_{\mathrm{blocks}}(ux), f_{\mathrm{blocks}}(uy)) = (f(ux), f(uy))$. If $u_1 = \epsilon$ and u, u'_1 end with different letters, then again, $\mathsf{del}(f_{\mathrm{blocks}}(ux), f_{\mathrm{blocks}}(uy)) = (f(ux), f(uy))$. There are two more classes with respect to $\overset{\Delta}{\approx}_{f_{\mathrm{blocks}}}$, one for infinitely many $\#$, and one for no $\#$.

The look-ahead $\overset{\Delta}{\approx}_{f_{\mathrm{blocks}}}$ provides enough information to transform the blocks deterministically (we only need the last letter before the next $\#$), but not enough information to produce the output after the last $\#$ deterministically.

The following proposition shows that the delay congruence, when used as a look-ahead (see the definition of $f_{\mathcal{R}}$ page 11), transforms any rational function into a quasi-sequential one.

▶ **Proposition 21.** Let f be a transduction and let $\mathcal{R}$ be a right automaton recognizing $\mathrm{dom}(f)$. Then $f_{\mathcal{R}}$ is quasi-sequential iff $\approx_{\mathcal{R}} \sqsubseteq \overset{\Delta}{\approx}_f$. In particular, if f is aperiodic then $\overset{\Delta}{\approx}_f$ is aperiodic.

The delay congruence is minimal, *i.e.* coarsest, among right congruences of bimachines realizing a function, and we show in the long version that it can be computed in PTime from a bimachine.

▶ **Proposition 22.** Given a transducer $\mathcal{T}$ (resp. a bimachine $\mathcal{B}$) with underlying automaton $\mathcal{A}$ (resp. right automaton $\mathcal{R}$) realizing a function f, we have that $\approx_{\mathcal{A}}$ (resp. $\approx_{\mathcal{R}}$) is finer than $\overset{\Delta}{\approx}_f$.

Canonical machine for quasi-sequential functions. As noted in [2], the class of quasi-sequential functions, or equivalently, the class of functions satisfying the WTP, is strictly larger than the class of sequential functions. The last left congruence that we define now will be fine enough to make a quasi-sequential function sequential. By taking the intersection between

this congruence and the left delay congruence we will obtain a congruence that is fine enough to make any rational function sequential. However, it should be noted that this look-ahead is not minimal, in the sense that it is not necessarily coarser than any look-ahead that is fine enough to realize the function.

▶ **Definition 23** (Ultimate congruence)**.** We define the *ultimate congruence* of a rational function f by setting $x \overset{\cup}{\approx}_f y$ for $x, y \in A^\omega$ if the following conditions hold:

- For all $u \in A^*$, $ux \in \mathrm{dom}(f) \Leftrightarrow uy \in \mathrm{dom}(f)$
- If $ux \in \mathrm{dom}(f)$ then $\widehat{f}(u) = \overline{f}(ux) \Leftrightarrow \widehat{f}(u) = \overline{f}(uy)$ Moreover, if $\widehat{f}(u) = \overline{f}(ux)$ then $f(ux) = f(uy)$.

Observe that $\widehat{f}(u) \preceq \overline{f}(ux)$ for every $ux \in \mathrm{dom}(f)$. So the intuition behind $\widehat{f}(u) = \overline{f}(ux)$ is that no finite look-ahead on x can help to output $f(ux)$ deterministically after u. And the intuition behind $f(ux) = f(uy)$ is that the missing outputs $\widehat{f}(u)^{-1} f(ux)$ and $\widehat{f}(u)^{-1} f(uy)$ have to be equal, which is equivalent to $f(ux) = f(uy)$. Now, for a given class of $\overset{\cup}{\approx}_f$ as look-ahead, a left automaton would know the missing output and start producing it. We show in the long version that $\overset{\cup}{\approx}_f$ is a left congruence.

▶ **Example 24.** Recall the function f_{blocks} defined in Example 3. $\widehat{f_{\mathrm{blocks}}}$ maps every block to its output and stops at the last #. Hence $\widehat{f_{\mathrm{blocks}}}(u) = \overline{f_{\mathrm{blocks}}}(ux)$ if and only if x does not contain #. When $\widehat{f_{\mathrm{blocks}}}(u) = \overline{f_{\mathrm{blocks}}}(ux)$, we have $f(ux) = f(uy)$ if and only if x and y both contain an infinite number of a's, or none of them does. The congruence classes of $\overset{\cup}{\approx}_{f_{\mathrm{blocks}}}$ are thus: a) words x with an infinite number of # (yielding ux outside the domain), b) words x with a finite (non-zero) number of #, c) words without #, with an infinite number of a's, d) words without #, with a finite number of a's. This is precisely the information lacking in the look-ahead provided by $\overset{\triangle}{\approx}_{f_{\mathrm{blocks}}}$ (see Example 20) to obtain a look-ahead allowing a sequential processing of the input.

▶ **Proposition 25.** For a quasi-sequential transduction f, the ultimate congruence $\overset{\cup}{\approx}_f$ has finite index. If f is given as a bimachine, $\overset{\cup}{\approx}_f$ can be computed in 2-ExpTime. Furthermore, if f is aperiodic then $\overset{\cup}{\approx}_f$ is aperiodic.

Let $\mathcal{R}$ be a right automaton recognizing $\overset{\cup}{\approx}_f$. We define the bimachine $\mathcal{U}_f^{\mathcal{R}} = (\mathcal{A}_f, \mathcal{R}, i_f, o_{\mathcal{R}})$ with $\mathcal{A}_f$ and i_f (as in Section 2), and for $o_{\mathcal{R}}$ we take:

$$o_{\mathcal{R}}([u], a, [x]_{\mathcal{R}}) = \begin{cases} \widehat{f}(u)^{-1}\widehat{f}(ua) & \text{if } \widehat{f}(ua) \prec \overline{f}(uax) \\ \beta & \text{if } \widehat{f}(u) = \overline{f}(uax) \text{ and } \widehat{f}(u)^{-1} f(uax) = \alpha\beta^\omega \\ \alpha & \text{if } \widehat{f}(u) \prec \widehat{f}(ua) = \overline{f}(uax) \text{ and } \widehat{f}(u)^{-1} f(uax) = \alpha\beta^\omega \end{cases}$$

The following lemma states that $\mathcal{U}_f^{\mathcal{R}}$ realizes f.

▶ **Lemma 26.** *Let f be a quasi-sequential transduction, and let $\mathcal{R}$ be a right automaton recognizing the ultimate congruence $\overset{\cup}{\approx}_f$, then $\mathcal{U}_f^{\mathcal{R}}$ realizes f.*

Let $\mathcal{R}$ be the canonical right automaton of $\overset{\cup}{\approx}_f$. By the previous lemma, there exists a bimachine with $\mathcal{R}$ as right automaton realizing f. By minimizing its left automaton with respect to $\mathcal{R}$, we obtain a canonical bimachine for f.

▶ **Corollary 27.** *Let f be a quasi-sequential transduction, and let $\mathcal{R}$ be the canonical right automaton of the ultimate congruence $\overset{\cup}{\approx}_f$, then $\mathcal{B}_f^{\mathcal{R}}$ realizes f (and is finite).*

Canonical bimachine. We finally show that by composing the information given by the delay and the ultimate congruences, we obtain a canonical bimachine for any rational function. Let us make clear what we mean by composition. Let $\mathcal{R}_1 = (Q_1, \Delta_1, I_1, F_1)$ be a right automaton and let $\mathcal{R}_2 = (Q_1, \Delta_2, I_2, F_2)$ be a right automaton over $A \times Q_1$. The automaton $\mathcal{R}_1 \bowtie \mathcal{R}_2$ is defined as $(Q_1 \times Q_2, \Delta_{\{1,2\}}, I_1 \times I_2, F_1 \times F_2)$ with $F_1 \times F_2 = \{P_1 \times P_2 \mid\ P_1 \in F_1,\ P_2 \in F_2\}$ and $\Delta_{\{1,2\}} = \{((s_1, s_2), a, (r_1, r_2) \mid (s_1, a, r_1) \in \Delta_1,\ (s_2, (a, r_1), r_2) \in \Delta_2\}$, which is a right automaton.

▶ **Lemma 28.** *Let $\mathcal{R}_1 = (Q_1, \Delta, I, F)$ be a right automaton and let $\mathcal{R}_2$ be a right automaton over $A \times Q_1$. Then $[\![\ell(\mathcal{R}_2)]\!] \circ [\![\ell(\mathcal{R}_1)]\!] = [\![\ell(\mathcal{R}_1 \bowtie \mathcal{R}_2)]\!]$ (up to the isomorphism between $(A \times Q_1) \times Q_2$ and $A \times (Q_1 \times Q_2)$).*

We can now state our main result. In our construction we focused on clarity and compositionality and we obtain a several-fold exponential complexity. At the cost of greater technicality, one should obtain a tighter result.

▶ **Theorem 29** (Canonical Bimachine)**.** *Let f be a transduction given by a bimachine, let $\mathcal{R}_1$ be the canonical automaton of the delay congruence $\overset{\Delta}{\approx}_f$, and let $\mathcal{R}_2$ be the canonical automaton of the ultimate congruence $\overset{\cup}{\approx}_{(f_{\mathcal{R}_1})}$. Then the bimachine $\mathcal{B}_f^{\mathcal{R}_1 \bowtie \mathcal{R}_2}$ realizes f. Furthermore if f is aperiodic then $\mathcal{B}_f^{\mathcal{R}_1 \bowtie \mathcal{R}_2}$ is aperiodic.*

Proof. Let f be a transduction, let $\mathcal{R}_1$ be the canonical automaton of the delay congruence $\overset{\Delta}{\approx}_f$ and let $\mathcal{R}_2$ be the canonical automaton of the ultimate congruence $\overset{\cup}{\approx}_{(f_{\mathcal{R}_1})}$. Since $\mathcal{R}_1$ recognizes $\overset{\Delta}{\approx}_f$, we know according to Proposition 22 that $f_{\mathcal{R}_1}$ is quasi-sequential. Hence since $\mathcal{R}_2$ is finer than $\overset{\cup}{\approx}_{(f_{\mathcal{R}_1})}$, we know from Cor. 27 that the bimachine $\mathcal{B}_{f_{\mathcal{R}_1}}^{\mathcal{R}_2}$ realizes f. From Proposition 17 we obtain that $(f_{\mathcal{R}_1})_{\mathcal{R}_2}$, the function obtained by composing the labelings $\ell(\mathcal{R}_2)$ and $\ell(\mathcal{R}_1)$, is left-sequential. We use Lemma 28 to obtain that $f_{\mathcal{R}_1 \bowtie \mathcal{R}_2}$ is left-sequential and thus, again by Proposition 17 we know there is a bimachine with $\mathcal{R}_1 \bowtie \mathcal{R}_2$ as right automaton which realizes f. In particular, $\mathcal{B}_f^{\mathcal{R}_1 \bowtie \mathcal{R}_2}$ realizes f.

If we assume that f is aperiodic, we obtain from Proposition 22 that $\mathcal{R}_1$ is aperiodic and from Proposition 17 that $f_{\mathcal{R}_1}$ is aperiodic. Hence from Proposition 25 we have that $\mathcal{R}_2$ is aperiodic. Again from Proposition 17, we have that $(f_{\mathcal{R}_1})_{\mathcal{R}_2} = f_{\mathcal{R}_1 \bowtie \mathcal{R}_2}$ is aperiodic. A third time from Proposition 17 we have that $\mathcal{B}_f^{\mathcal{R}_1 \bowtie \mathcal{R}_2}$ is aperiodic. ◀

Note that the right automaton constructed in Proposition 1 is actually a right *Büchi* automaton. So our result would still hold for bimachines with Büchi right automata.

5 First-Order Definability Problem

In this section, we show that given a transducer $\mathcal{T}$ realizing a transduction $[\![\mathcal{T}]\!] : A^\omega \to B^\infty$, one can decide whether $[\![\mathcal{T}]\!]$ is first-order definable (FO-definable). First, let us recall the notion of FO-definability for word languages. Any word $w \in A^\infty$ is seen as a structure of domain $\{1, \dots, |w|\}$ linearly ordered by $\preceq$ and with unary predicates $a(x)$, for all $a \in A$. By FO we denote the first-order logic over these predicates, and by MSO the extension of FO with quantification over sets and membership tests $x \in X$ (see for instance [22] for a detailed definition). We write $w \models \phi$ if some word w satisfies a formula ϕ, and $\phi(x_1, \dots, x_n)$ any formula ϕ with n free first-order variables $x_1, \dots, x_n$. Interpreted over words in A^ω (resp. A^∞), any sentence ϕ defines a language $[\![\phi]\!] \subseteq A^\omega$ (resp. $[\![\phi]\!] \subseteq A^\infty$) defined as the set of words satisfying ϕ. *E.g.* the sentence $\phi_0 = \forall x, y \cdot a(x) \wedge b(y) \to x \preceq y$, interpreted on A^ω, defines the language $a^\omega \cup a^*b^\omega$. Interpreted on A^∞, it defines the language $a^\omega \cup a^*b^\omega \cup a^*b^*$. A language L is said to be FO-definable (resp. MSO-definable) if $L = [\![\phi]\!]$ for some sentence $\phi \in$ FO (resp. $\phi \in$ MSO).

Definability of transductions. An MSO-transducer is a tuple $\mathcal{F} = (A, B, \phi_{dom}, V, \mu)$ where ϕ_{dom} is an MSO-sentence, V is a finite subset of B^* and μ a function mapping any word $v \in V$ to some MSO-formula (over alphabet A) denoted $\phi_v(x)$, with one-free variable. An *FO-transducer* is an MSO-transducer which uses only FO-formulas. Any MSO-transducer defines a transduction denoted $[\![\mathcal{F}]\!] \subseteq A^\omega \times B^\infty$ such that $(u, v) \in [\![\mathcal{F}]\!]$ if $u \models \phi_{dom}$ and there exists $(v_i)_{i\geq 1}$ such that $v = v_1 v_2 v_3 \dots$ and for all $i \geq 1$, $v_i \in V$ and $u \models \phi_{v_i}(i)$. We say that $f : A^\omega \to B^\infty$ is MSO- (resp. FO-) definable if there exists some MSO- (resp. FO-) transducer $\mathcal{F}$ such that $[\![\mathcal{F}]\!] = f$.

For example the functional transduction which erases all a's of any input ω-word over $\{a, b\}$ is defined by $\phi_{dom} = \top$ and the two formulas $\phi_\epsilon(x) = a(x)$ and $\phi_b(x) = b(x)$. The functional transduction mapping any word of the form $a^n b^\omega$ to $a^{\lfloor n/2 \rfloor} b^\omega$ is not FO-definable, even though its domain is. Intuitively, the formula $\phi_a(x)$ would have to decide whether x is an odd or even position, which is a typical non FO-definable property. It is one of the goal of this paper to automatically verify that such a property is indeed not FO. It is however MSO-definable with $\phi_{dom} = \phi_0 \wedge \exists x \cdot b(x)$, where ϕ_0 has been defined before, and the three formulas $\phi_\epsilon(x) = a(x) \wedge odd(x)$, $\phi_a(x) = a(x) \wedge even(x)$ (properties which are MSO-definable) and $\phi_b(x) = b(x)$.

As a remark, Courcelle has defined in the context of graph transductions the notion of MSO-transducers [9], which can also be restricted to FO-transducers. Cast to infinite words, Courcelle's formalism is strictly more expressive than rational functions, as they allow to mirror factors of the input word for instance. Restricted to the so called *order-preserving* Courcelle transducers [4, 12], they are equivalent to our MSO- and FO-transducers, however with a more complicated definition. This equivalence can be seen, for finite words, in the proof of Theorem 4 in [12]. The same proof works for infinite words as well.

We first exhibit a correspondence between logics and transducers, the proof of which is similar to the finite case [12], but requires some additional results on aperiodic automata on ω-words.

▶ **Theorem 30** (Logic-transducer correspondences). *Let $f : A^\omega \to B^\infty$. Then:*

- *f is MSO-definable if and only if it is realizable by some transducer.*
- *f is FO-definable if and only if it is realizable by some aperiodic transducer.*

We obtain the following decidability result (in elementary complexity if the input is a transducer).

▶ **Theorem 31.** *It is decidable whether a rational function $f : A^\omega \to B^\infty$, given as a transducer or equivalently as an MSO-transducer, is definable in FO.*

Proof. By Theorem 30, it suffices to show that f is aperiodic, *i.e.* definable by some aperiodic transducer. By Theorem 16, one can construct a bimachine which is aperiodic if and only if f is. So, it suffices to construct this bimachine and to test its aperiodicity, *i.e.*, whether its left and right automata are both aperiodic, a property which is decidable [10]. ◀

References

1 André Arnold. A Syntactic Congruence for Rational ω-Languages. *Theoretical Computer Science*, 39(2–3):333–335, August 1985. Note.

2 Marie-Pierre Béal and Olivier Carton. Determinization of Transducers over Infinite Words: The General Case. *Theory Comput. Syst.*, 37(4):483–502, 2004.

3 Adrien Boiret, Aurélien Lemay, and Joachim Niehren. Learning Rational Functions. In *Developments in Language Theory - 16th International Conference, DLT 2012, Taipei, Taiwan, August 14-17, 2012. Proceedings*, pages 273–283, 2012.

4 Mikolaj Bojanczyk. Transducers with Origin Information. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 26–37. Springer, 2014.

5 Olivier Carton. Right-Sequential Functions on Infinite Words. In *Computer Science - Theory and Applications, 5th International Computer Science Symposium in Russia, CSR 2010, Kazan, Russia, June 16-20, 2010. Proceedings*, pages 96–106, 2010.

6 Olivier Carton and Max Michel. Unambiguous Büchi automata. *Theor. Comput. Sci.*, 297(1-3):37–81, 2003.

7 Olivier Carton, Dominique Perrin, and Jean-Eric Pin. Automata and semigroups recognizing infinite words. In *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*., pages 133–168, 2008.

8 Christian Choffrut. Minimizing subsequential transducers: a survey. *Theor. Comput. Sci.*, 292(1):131–143, 2003.

9 Bruno Courcelle. Monadic Second-Order Definable Graph Transductions: A Survey. *Theor. Comput. Sci.*, 126(1):53–75, 1994.

10 Volker Diekert and Paul Gastin. First-order definable languages. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata*, volume 2 of *Texts in Logic and Games*, pages 261–306. Amsterdam University Press, 2008.

11 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.

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.

13 Emmanuel Filiot, Olivier Gauwin, and Nathan Lhote. Aperiodicity of Rational Functions Is PSPACE-Complete. In *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2016, December 13-15, 2016, Chennai, India*, pages 13:1–13:15, 2016.

14 Emmanuel Filiot, Olivier Gauwin, and Nathan Lhote. First-order definability of rational transductions: An algebraic approach. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS*, pages 387–396. ACM, 2016.

15 Emmanuel Filiot, Olivier Gauwin, and Nathan Lhote. Logical and Algebraic Characterizations of Rational Transductions. *CoRR*, abs/1705.03726, 2017. Available from: `http://arxiv.org/abs/1705.03726`.

16 Françoise Gire. Two Decidability Problems for Infinite Words. *Inf. Process. Lett.*, 22(3):135–140, 1986.

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

18 D. Perrin. Recent results on automata and infinite words. In M. P. Chytil and V. Koubek, editors, *Proceedings of the 11th Symposium on Mathematical Foundations of Computer Science*, volume 176 of *LNCS*, pages 134–148, Praha, Czechoslovakia, September 1984. Springer.

19 Christophe Prieur. How to decide continuity of rational functions on infinite words. *Theor. Comput. Sci.*, 276(1-2):445–447, 2002.

20 Christophe Reutenauer and Marcel Paul Schützenberger. Minimization of Rational Word Functions. *SIAM J. Comput.*, 20(4):669–685, 1991.

21 Marcel-Paul Schützenberger. On Finite Monoids Having Only Trivial Subgroups. *Information and Control*, 8(2):190–194, 1965.

22 W. Thomas. Languages, Automata and Logic. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 3, Beyond Words. Springer, Berlin, 1997.

23 Wolfgang Thomas. Star-Free Regular Sets of omega-Sequences. *Information and Control*, 42(2):148–156, August 1979.

24 Wolfgang Thomas. A Combinatorial Approach to the Theory of omega-Automata. *Information and Control*, 48(3):261–283, 1981.

25 Thomas Wilke. Past, Present, and Infinite Future. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 95:1–95:14, 2016.