Deterministic Regular Functions of Infinite Words

Olivier Carton
Université Paris Cité, CNRS, IRIF, F-75013, France
Institut Universitaire de France, Paris, France

Gaëtan Douéneau-Tabot
Université Paris Cité, CNRS, IRIF, F-75013, France
Direction générale de l'armement – Ingénierie des projets, Paris, France

Emmanuel Filiot
Université libre de Bruxelles & F.R.S.-FNRS, Brussels, Belgium

Sarah Winter
Université libre de Bruxelles & F.R.S.-FNRS, Brussels, Belgium

Abstract

Regular functions of infinite words are (partial) functions realized by deterministic two-way transducers with *infinite* look-ahead. Equivalently, Alur et. al. have shown that they correspond to functions realized by deterministic Muller streaming string transducers, and to functions defined by MSO-transductions. Regular functions are however not computable in general (for a classical extension of Turing computability to infinite inputs), and we consider in this paper the class of *deterministic regular functions* of infinite words, realized by deterministic two-way transducers *without* look-ahead. We prove that it is a well-behaved class of functions: they are computable, closed under composition, characterized by the *guarded* fragment of MSO-transductions, by deterministic Büchi streaming string transducers, by deterministic two-way transducers with *finite* look-ahead, and by finite compositions of sequential functions and one fixed basic function called map-copy-reverse.

2012 ACM Subject Classification Theory of computation → Transducers

Keywords and phrases infinite words, streaming string transducers, two-way transducers, monadic second-order logic, look-aheads, factorization forests

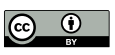
Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.121

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Funding This work was partially supported by the Fonds National de la Recherche Scientifique – F.R.S.-FNRS – under the MIS project F451019F.

1 Introduction

Transducers extend automata with output mechanisms, turning finite state machines from language acceptors to computational models for functions. Inspired by a seminal work by Engelfriet and Hoogeboom [22], the last decade has seen an increasing interest in characterizing the class of functions defined by deterministic two-way transducers over finite words (2-dT), now called the class of *regular functions of finite words*. This class admits several (effective) characterizations: it corresponds to the functions definable by MSO-transductions [22], by an MSO-based logic on origin graphs [15], by an extension of regular expressions called combinator expressions [4, 5, 20], and computed by *copyless streaming string transducers* (SST) (a deterministic one-way model which uses registers to store and update partial output words [2]). Moreover, the class of regular functions over finite words is closed under composition [11], and it has decidable equivalence problem [25].



▶ **Example 1.1.** Let Σ be an alphabet, the function map-copy-reverse : $(\Sigma \uplus \{|\})^* \to (\Sigma \uplus \{|\})^*$ takes any word of the form $u_1|\dots|u_n$ where each u_i is $|$-free, and outputs $u_1|\widetilde{u_1}|\dots|u_n|\widetilde{u_n}$, where $\widetilde{u_i}$ is the mirror image of u_i. The function map-copy-reverse is regular.

Regular functions can also be characterized as the compositions of sequential functions (functions computed by deterministic *one-way* finite transducers) and map-copy-reverse [6].

Regular functions of infinite words. The class of regular functions has been extended to infinite words in [3], and defined as the class of functions definable by MSO-transductions over infinite words. Equivalently, they have been shown to be the functions realized by deterministic two-way transducers with regular look-ahead, and by streaming string transducers with a Muller selection condition (the register holding the final output word is selected depending on the set of states seen infinitely often). As for finite words, regular functions of infinite words are closed under composition, and have decidable equivalence problem [3].

▶ **Example 1.2.** Let $\Sigma = \{a, b, c\}$ be an alphabet, and consider the function double : $\Sigma^\omega \to \Sigma^\omega$ which behaves like the identity function except that any occurrence of a is replaced by aa if there exists a b in the future of that occurence. For example, $(ab)^\omega$ is mapped to $(aab)^\omega$ and $acaab(ac)^\omega$ is mapped to $aacaaaab(ac)^\omega$. The function double is regular, as it can be realized by a one-way transducer which, when reading an a, uses regular look-aheads to determine whether there exists a b or not in the future, and produces either a or aa accordingly.

▶ **Example 1.3.** Let $\Sigma' = \{a, b, 1, 2\}$ and consider the function copy which maps:

- $u_1\sigma_1 u_2 \dots \sigma_n u \mapsto u_1^{\sigma_1}\sigma_1 \dots u_n^{\sigma_n}\sigma_n u$ where $u_1u_2\dots u_n u \in \{a,b\}^\omega$ and $\sigma_1, \dots, \sigma_n \in \{1,2\}$;
- $u_1\sigma_1 \dots u_i\sigma_i \cdots \mapsto u_1^{\sigma_1}\sigma_1 \dots u_i^{\sigma_i}\sigma_i \dots$ (if there are infinitely many $\sigma_i \in \{1,2\}$).

For example, $\mathsf{copy}(ab2a1b^\omega) = abab2a1b^\omega$ and $\mathsf{copy}((a2)^\omega) = (aa2)^\omega$. The function copy is regular, for instance realized by a deterministic two-way transducer which, using two-wayness, makes one or two passes on the blocks u_i, depending on whether they are followed by $\sigma_i = 2$. On the first pass, it always outputs what it reads, so that if no separator in $\{1,2\}$ is ever read again (which means it is reading the infinite suffix u), then it outputs u.

Despite the robustness of the class of regular functions of infinite words, witnessed by its various characterizations and algorithmic properties, they suffer from a severe downside when it comes to computability. Indeed, there are regular functions of infinite words which are *not* computable. At this point, we make clear what is meant by computability, since the input is infinite. We refer the reader to [18, 19] (and the references therein) for a formal definition of computability, and rather give intuitions here. A function f of infinite words is computable if there is a Turing machine with an infinite read-only tape which contains some infinite input word u in the domain of the function, a bidirectional working tape, and a write-only left-to-right output tape, such that by reading longer and longer input prefixes, the machine writes longer and longer prefixes of $f(u)$ on the output tape. Informally, it is an algorithm which takes the input as a stream and is able to produce the output as a stream, so that infinitely often, at least one output symbol is produced. For instance, the function double above is not computable. On reading prefixes of the form ac^n for increasing values of n, it can safely output one a symbol, but not more. Indeed, if it outputs one more a, then it is a wrong output for continuation c^ω, and if it outputs a c, then it is a wrong output for continuation b^ω, as $\mathsf{double}(ac^nc^\omega) = ac^\omega$ and $\mathsf{double}(ac^nb^\omega) = aac^nb^\omega$. Its implementation by a two-way transducer indeed requires an infinite look-ahead to check the absence of a b in the future. On the other hand, copy is realized by a deterministic two-way transducer with *no* look-ahead, so it is computable. So, deterministic two-way transducers

with (infinite) look-ahead, and their equivalent model Muller streaming string transducers, *cannot* be considered as models of computation for infinite word functions. This was observed in [19], where it is shown that the problem of deciding whether a given regular function of infinite words is computable is PSPACE-C. On the other hand, deterministic two-way transducers *without look-ahead* are a proper model of computation for functions of infinite words.

Deterministic regular functions of infinite words. Motivated by the latter observation, the class of functions computed by deterministic two-way transducers *without* look-ahead, coined the class of *deterministic regular functions*, was introduced in [9], where it is shown that they are also equivalently computed by Büchi SST (BSST). In BSST, there is one special designated register $\mathfrak{out}$ in which to write the output word, which is required to be updated with at least one new symbol infinitely often. For example, copy can be implemented by a single-state BSST with two registers $\mathfrak{out}$ and $\mathfrak{r}$, updated as follows. On reading $\sigma \in \{a, b\}$, it performs the updates $\mathfrak{out} \mapsto \mathfrak{out}.\sigma$ and $\mathfrak{r} \mapsto \mathfrak{r}.\sigma$, on reading 1, it does $\mathfrak{out} \mapsto \mathfrak{out}.1$ and $\mathfrak{r} \mapsto \varepsilon$, and on reading 2, it does $\mathfrak{out} \mapsto \mathfrak{out}.\mathfrak{r}.2$ and $\mathfrak{r} \mapsto \varepsilon$.

Several important questions remain on the class of deterministic regular functions, such as whether it is closed under composition, whether it can be logically characterized by a natural fragment of MSO-transductions, and whether they can be obtained as finite compositions of "simple" functions. In this paper, we provide positive answers to these questions.

Contributions. Concerning the class of deterministic regular functions, our main results are:

- its effective closure under composition;
- its characterization by means of finite compositions of sequential functions and an extension of map-copy-reverse to infinite words;
- a logical characterization by a natural syntactic fragment of MSO-transductions, the guarded fragment, called MSOT_g.

An MSO-transduction is defined as an MSO-interpretation, where the predicates of the output word structure, namely the successor and label relations, are defined by MSO formulas with two and one free first-order variables respectively, interpreted over a fixed number of copies of the input. The guarded fragment is defined by a classical restriction (see e.g. [24] and references therein) on the MSO formulas composing the MSO-transduction. They have to be prefixed by an existential quantifier $\exists \mathsf{g}$, where g is a word position, and all quantifiers of the formula are guarded by the guard $x \leq \mathsf{g}$ (and $\forall x \in X, x \leq \mathsf{g}$ for any set variable X). So, guarded MSO formulas on infinite words, only speak about finite prefixes. Consider again the function copy. Two copies of the input are needed to account for potential duplication of the blocks, but the presence or not of a successor edge between two nodes of the output word structure, only depends on local properties, which are definable by guarded MSO formulas. E.g., such a property may be "if position $x+1$ is labeled 2, then there is a successor between the 1st copy of x and the 2nd copy of first position of the block to which x belongs".

In general, guarded MSO formulas can test non-local properties, which is the main source of technical difficulties in the paper. It is illustrated by the next example.

▶ **Example 1.4.** The function replace : $\{0, a, b\}^\omega \multimap \{a, b\}^\omega$ of domain $\mathsf{Dom}(\mathsf{replace}) = \{u \in \{0, a, b\}^\omega : |u|_a{=}\infty \text{ or } |u|_b{=}\infty\}$ and mapping $0^{n_1}\sigma_1 0^{n_2}\sigma_2 \cdots \mapsto \sigma_1{}^{n_1+1}\sigma_2{}^{n_2+1}\cdots$ if $\sigma_i \in \{a, b\}$ and $n_i \in \mathbb{N}$, is deterministic regular. Replacing a zero at position x by a or b depends on the next non-zero symbol in the future of x, which can be arbitrarily faraway, but occurs in a finite prefix if $u \in \mathsf{Dom}(\mathsf{replace})$. This property is expressible with a guarded MSO formula, which defines the position holding this non-zero symbol as a guard.

Proof techniques and additional results. We now give an overview of the proof techniques used to show the logical characterization, along with some other interesting and useful results. We prove that deterministic two-way transducers (2-dT) are expressively equivalent to MSOT_g. The conversion of 2-dT into MSOT_g is standard and follows the same line as [22]. The converse is more involved and requires new techniques. First, we convert MSOT_g-transductions into deterministic two-way transducers with *finite look-ahead* (2-$\mathsf{dT}^{\mathsf{FLA}}$), which account for non-local, but finite, properties, as illustrated before. 2-$\mathsf{dT}^{\mathsf{FLA}}$ are equipped with regular languages of finite words on their transitions, which act as finite look-aheads in the following sense: when the reading head is at some position i of an infinite word u, in some state q, a transition from q with look-ahead L is enabled if there exists a position $j \geq i$, called witness, such that the infix $u[i{:}j]$ starting at position i and ending at position j, belongs to L. If no transition is enabled at state q, the computation fails. To ensure determinism, if several transitions are enabled, only the transition with minimal (i.e. smallest) witness j is triggered, and a disjointness requirement on the look-aheads make sure that this j is unique. The condition to consider only the transition with minimal witness j is crucial to ensure that 2-$\mathsf{dT}^{\mathsf{FLA}}$ define only computable functions. Indeed, a 2-$\mathsf{dT}^{\mathsf{FLA}}$ can be executed as follows: all finite look-aheads, supposed for instance to be finitely represented by DFA, are executed in parallel. By the minimality requirement for j and the disjointness of look-aheads, *as soon as* a prefix is accepted by one look-ahead DFA, the corresponding transition is triggered.

Adding look-aheads to two-way transducers in order to capture MSO-transductions is standard on finite words [22, 14], for example because the "moves" of the MSO-transduction depends on non-local properties. Look-aheads are then directly removed by using the closure under composition of deterministic two-way transducers [11]. Closure under composition of deterministic two-way transducers on infinite words is, to the best of our knowledge, unknown, and instead we give a direct proof of finite look-ahead removal. It is our main technical result: any 2-$\mathsf{dT}^{\mathsf{FLA}}$ is effectively equivalent to some 2-dT. To prove this result, classical techniques, such as Hopcroft-Ullman construction [1] or the tree outline construction [16] do not apply, as they heavily rely on the fact that words are finite. In our setting, we instead use a new technique, based on summarizing the computations of the look-aheads into trees which we prove to be bounded. As a side result of finite look-ahead removal, we prove that 2-dT (and so deterministic regular functions) are closed under composition. Classically, closure under composition of MSO-transductions is direct, by formula substitutions [14]. This technique however does not apply here, as the guarded MSO formulas are not syntactically closed under formula substitution, making the correspondence between MSOT_g and 2-dT crucial to obtain closure under composition of MSO_g-transductions.

Structure of the paper. In Section 2, we introduce the class of deterministic regular functions. In Section 3, we prove its closure under composition and the decomposition result. In Section 4, we introduce guarded MSO-transductions and state the logical characterization. Since its proof is based on a compilation into deterministic two-way transducers with finite look-ahead, we prove in Section 5 how to remove those look-aheads. Finally, we prove the logical characterization in Section 6. All transformations are effective in the paper. Some proofs are only sketched or simply omitted, but the proof details can be found in [10].

2 Deterministic regular functions

In this section, we introduce the class of *deterministic regular functions of infinite words* and recall that it can be described by two computation models: *deterministic two-way transducers* and *deterministic Büchi streaming string transducers*.

Notations. Letters Σ, Γ denote alphabets, i.e. finite sets of letters. The set Σ^* (resp. Σ^+, Σ^ω) denotes the set of finite words (resp. non-empty finite words, infinite words) over the alphabet Σ. Let $\Sigma^\infty := \Sigma^* \cup \Sigma^\omega$. If $u \in \Sigma^\infty$, we let $|u| \in \mathbb{N} \cup \{\infty\}$ be its length, $|u|_\sigma \in \mathbb{N} \cup \{\infty\}$ be the number of occurrences of $\sigma \in \Sigma$ and $u[i] \in \Sigma$ be the i-th letter of u for $1 \leq i \leq |u|$. If $1 \leq i \leq j \leq |u|$, $u[i{:}j]$ stands for $u[i] \cdots u[j]$. We write $u[i{:}]$ for $u[i{:}|u|]$. If $j > |u|$ we let $u[i{:}j] := u[i{:}|u|]$. If $j < i$ we let $u[i{:}j] := \varepsilon$. In this paper, functions are *by default* partial (i.e. possibly with non-total domain). A (partial) function f from S to T is denoted $f : S \rightharpoonup T$, and its domain is denoted $\mathsf{Dom}(f) \subseteq S$. A total function from S to T is denoted $f : S \to T$.

Two-way transducers. Let us recall the syntax of two-way transducers. We consider here that the machines work on infinite words, and have a Büchi acceptance condition.

▶ **Definition 2.1** (Two-way transducer). *A* deterministic two-way transducer (2-dT) *denoted* $\mathcal{T} = (\Sigma, \Gamma, Q, q_0, F, \delta, \lambda)$ *consists of:*

- *an input alphabet* Σ *and an output alphabet* Γ*;*
- *a finite set of states* Q *with an initial state* $q_0 \in Q$ *and a set of final states* $F \subseteq Q$*;*
- *a transition function* $\delta : Q \times (\Sigma \uplus \{\vdash\}) \rightharpoonup Q \times \{\triangleleft, \triangleright\}$*;*
- *an output function* $\lambda : Q \times (\Sigma \uplus \{\vdash\}) \rightharpoonup \Gamma^*$ *with same domain as* δ*.*

A *configuration* of $\mathcal{T}$ over $u \in (\Sigma \cup \{\vdash\})^\infty$ is a tuple (q, i) where $q \in Q$ is the current state and $1 \leq i \leq |u|$ is the current position of the reading head. The *transition relation* $\to$ is defined as follows. Given a configuration (q, i), let $(q', \star) := \delta(q, u[i])$. Then $(q, i) \to (q', i')$ whenever either $\star = \triangleleft$ and $i' = i - 1$ (move left), or $\star = \triangleright$ and $i' = i + 1$ (move right). A *run* over u is a (finite or infinite) sequence of consecutive configurations $(q_1, i_1) \to (q_2, i_2) \to \cdots$.

Now, we define the infinite output produced by $\mathcal{T}$ when given the infinite word $u \in \Sigma^\omega$ as input. First, we let $u[0] := \vdash$, i.e. we force the symbol $\vdash$ to be used to mark the beginning of the input. An *accepting* run is an infinite run that starts in $(q_0, 0)$, visits infinitely often configurations of the form (q, i) with $q \in F$ and such that $i_n \to \infty$ when $n \to \infty$ (without this last condition, the transducer may enter an infinite loop without reading its whole input). The partial function $f : \Sigma^\omega \rightharpoonup \Gamma^\omega$ computed by $\mathcal{T}$ is defined as follows. Let $u \in \Sigma^\omega$ be such that there exists a (unique) accepting run $(q_0^u, i_0^u) \to (q_1^u, i_1^u) \to \cdots$ labelled by $\vdash u$. Let $v := \prod_{j=1}^{\infty} \lambda(q_j^u, (\vdash u)[i_j^u]) \in \Gamma^* \cup \Gamma^\omega$ be the concatenation of the outputs produced along this run. If $v \in \Gamma^\omega$, we define $f(u) := v$. Otherwise $f(u)$ is undefined.

▶ **Definition 2.2.** *The class of* deterministic regular functions of infinite words *is the class of (partial) functions computed by deterministic two-way transducers.*

We have explained in Example 1.3 how to compute the function copy using a 2-dT (without look-aheads). Observe that the function replace from Example 1.4 can be computed in a similar fashion. Hence both functions are deterministic regular.

▶ **Example 2.3.** Let us extend the function map-copy-reverse of Example 1.1 to infinite words. Let Σ be an alphabet, we define map-copy-reverse : $(\Sigma \uplus \{|\})^\omega \to (\Sigma \uplus \{|\})^\omega$ as follows:

- map-copy-reverse$(u_1|u_2|\cdots) := u_1|\widetilde{u_1}|u_2|\widetilde{u_2}|\cdots$ with $u_i \in \Sigma^*$ for all $i \geq 0$;
- map-copy-reverse$(u_1|\cdots|u_n|u) := u_1|\widetilde{u_1}|\cdots|u_n|\widetilde{u_n}|u$ for $u_i \in \Sigma^*$ and $u \in \Sigma^\omega$.

This function is deterministic regular since we can build a 2-dT that processes twice each $|$-free factor (or only once for the last infinite one if it exists).

Büchi Streaming String Transducers. Now, we describe a model of a one-way machine with registers which captures deterministic regular functions of infinite words. Over finite words, it is well-known that deterministic two-way transducers are equivalent to *copyless streaming string transducers* [2]. A similar equivalence holds for the class of *regular functions of infinite words*, which can equivalently be described by *deterministic two-way transducers with regular look-aheads* or *copyless streaming string transducers with Muller conditions* [3]. However, Muller conditions enable to check regular properties of the infinite input, and thus describe functions which are not (Turing) computable [3]. Now, let us recall the model of *Büchi deterministic streaming string transducer* (BSST), introduced by Carton and Douéneau-Tabot in [9], that captures exactly the class of deterministic regular functions.

Formally, a Büchi deterministic streaming string transducer consists of a one-way deterministic automaton with a finite set $\mathfrak{R}$ of registers that store words from Γ^*. We use a distinguished register $\mathfrak{out}$ to store the output produced when reading an infinite word. The registers are modified when reading the input using *substitutions*, i.e. mappings $\mathfrak{R} \to (\Gamma \uplus \mathfrak{R})^*$. We denote by $\mathcal{S}_{\mathfrak{R}}^{\Gamma}$ the set of these substitutions. They can be extended morphically from $(\Gamma \uplus \mathfrak{R})^*$ to $(\Gamma \uplus \mathfrak{R})^*$ by preserving the elements of Γ.

▶ **Example 2.4** (Substitutions). Let $\mathfrak{R} = \{\mathfrak{r}, \mathfrak{s}\}$ and $\Gamma = \{b\}$. Consider $\tau_1 := \mathfrak{r} \mapsto b, \mathfrak{s} \mapsto b\mathfrak{r}\mathfrak{s}b$ and $\tau_2 := \mathfrak{r} \mapsto \mathfrak{r}b, \mathfrak{s} \mapsto \mathfrak{r}\mathfrak{s}$, then $\tau_1 \circ \tau_2(\mathfrak{r}) = \tau_1(\mathfrak{r}b) = bb$ and $\tau_1 \circ \tau_2(\mathfrak{s}) = \tau_1(\mathfrak{r}\mathfrak{s}) = bb\mathfrak{r}\mathfrak{s}b$.

▶ **Definition 2.5.** *A* Büchi deterministic streaming string transducer (BSST) *denoted by* $\mathcal{T} = (\Sigma, \Gamma, Q, F, q_0, \delta, \mathfrak{R}, \mathfrak{out}, \lambda)$ *consists of:*

- *a finite input (resp. output) alphabet* Σ *(resp.* Γ*);*
- *a finite set of states* Q *with* $q_0 \in Q$ *initial and* $F \subseteq Q$ *final;*
- *a transition function* $\delta : Q \times \Sigma \rightharpoonup Q$*;*
- *a finite set of registers* $\mathfrak{R}$ *with a distinguished output register* $\mathfrak{out} \in \mathfrak{R}$*;*
- *an update function* $\lambda : Q \times \Sigma \rightharpoonup \mathcal{S}_{\mathfrak{R}}^{\Gamma}$ *such that for all* $(q, \sigma) \in \mathsf{Dom}(\lambda) = \mathsf{Dom}(\delta)$*:*
  - $\lambda(q, \sigma)(\mathfrak{out}) = \mathfrak{out} \cdots$*;*
  - *there is no other occurrence of* $\mathfrak{out}$ *among the* $\lambda(q, \sigma)(\mathfrak{r})$ *for* $\mathfrak{r} \in \mathfrak{R}$*.*

This machine defines a partial function $f : \Sigma^\omega \rightharpoonup \Gamma^\omega$ as follows. For $i \geq 0$ let $q_i^u := \delta(q_0, u[1{:}i])$ (when defined). For $i \geq 1$, we let $\lambda_i^u := \lambda(q_{i-1}^u, u[i])$ (when defined) and $\lambda_0^u(\mathfrak{r}) = \varepsilon$ for all $\mathfrak{r} \in \mathfrak{R}$. For $i \geq 0$, let $[\![\cdot]\!]_i^u := \lambda_0^u \circ \cdots \circ \lambda_i^u$. By construction $[\![\mathfrak{out}]\!]_i^u$ is a prefix of $[\![\mathfrak{out}]\!]_{i+1}^u$ (when defined). If $[\![\mathfrak{out}]\!]_i^u$ is defined for all $i \geq 0$, q_i^u is a state of F infinitely often, and $|[\![\mathfrak{out}]\!]_i^u| \to +\infty$, then we let $f(u) := \bigvee_i [\![\mathfrak{out}]\!]_i^u$ (the symbol $\vee$ is used to denote the unique $v \in \Gamma^\omega$ such that $[\![\mathfrak{out}]\!]_i^u$ is a prefix of v for all $i \geq 0$). Otherwise $f(u)$ is undefined.

▶ **Example 2.6.** The function replace from Example 1.4 can be computed by a BSST. For all $i \geq 1$, it crosses the block 0^{n_i} and computes 1^{n_i} and 2^{n_i} in two registers. Once it sees σ_i it adds in $\mathfrak{out}$ the register storing ${\sigma_i}^{n_i}$.

▶ **Definition 2.7** (Copyless, bounded copy). *We say that a substitution* $\tau \in \mathcal{S}_{\mathfrak{R}}^{B}$ *is* copyless *(resp.* K-bounded*) if for all* $\mathfrak{r} \in \mathfrak{R}$*,* $\mathfrak{r}$ *occurs at most once in* $\{\tau(\mathfrak{s}) : \mathfrak{s} \in \mathfrak{R}\}$ *(resp. for all* $\mathfrak{r}, \mathfrak{s} \in \mathfrak{R}$*,* $\mathfrak{r}$ *occurs at most* K *times in* $\tau(\mathfrak{s})$*). We say that a* BSST $\mathcal{T} = (\Sigma, \Gamma, Q, q_0, \delta, \mathfrak{R}, \mathfrak{out}, \lambda)$ *is* copyless *(resp.* K-bounded*) if for all* $u \in \Sigma^\omega$ *and* $i \leq j$ *such that* $\lambda_i^u \circ \cdots \circ \lambda_j^u$ *is defined, this substitution is copyless (resp.* K*-bounded).*

▶ Remark 2.8. The composition of two copyless substitutions is copyless, hence a BSST is copyless as soon as $\lambda(q, \sigma)$ is copyless for all $q \in Q$ and $\sigma \in \Sigma$. However, K-boundedness is not necessarily preserved under composition.

Observe that the BSST described in Example 2.6 is copyless. Now, we recall the result of Carton and Douéneau-Tabot that proves equivalence between two-way transducers, copyless, and bounded copy Büchi deterministic streaming string transducers.

▶ **Theorem 2.9** ([9, Theorem 3.7]). *The following machines compute the same class of partial functions over infinite words:*

1. *deterministic two-way transducers* (2-dT)*;*
2. *K-bounded deterministic Büchi streaming string transducers* (K*-bounded* BSST)*;*
3. *copyless deterministic Büchi streaming string transducers* (*copyless* BSST)*.*

Furthermore, all the conversions are effective.

▶ Remark 2.10. The original proof of [9] which transforms a 2-dT into a BSST only considers machines where all states are final. Nevertheless, the proof can easily be adapted to transducers with non-final states. Furthermore, given a BSST (possibly with non-final states) one can build an equivalent BSST where all states are final by [9, Lemma D.1] (the Büchi conditions are hidden in the fact that the output must be infinite). All in all, all the models (with all states final or not) exactly capture the class of deterministic regular functions.

Finally, we recall the domains of deterministic regular functions. We say that a language is *Büchi deterministic* if it is accepted by a deterministic Büchi automaton (see e.g. [26]).

▶ **Proposition 2.11** ([9]). *If f is deterministic regular, then* Dom(f) *is Büchi deterministic.*

3 Composition and decomposition theorems

In this section, we show that deterministic regular functions are closed under composition, and that conversely they can be written as the composition of some "basic" functions.

It is known since [11] (resp. [3]) that the class of regular functions of finite (resp. infinite) words is closed under composition. We transport this result to deterministic regular functions of infinite words in Theorem 3.1. However, its proof is not an immediate extension of the regular case, and it illustrates the main difficulty of this paper: since look-aheads are not allowed, it is complex for a 2-dT to check if some property happens *after* its current position.

▶ **Theorem 3.1.** *Deterministic regular functions are (effectively) closed under composition.*

Proof idea. The approach is to compose the two transducers directly (using a product construction); the difficulty in the composition of two computations arises when one transducer is moving forward and the other backward. In that case, we need to rewind the computation of the transducer that moves backward by one computation step.

To recover the previous configuration look-ahead comes in handy. As mentioned above, (infinite) look-aheads are not permitted, but we use a weaker form of *finite* look-aheads (to be introduced in Section 5) which does not increase the expressiveness of deterministic two-way transducers over infinite words (and can be effectively removed), see Theorem 5.2. Finite look-aheads account for non-local but finite properties. The look-ahead we define basically re-traces the computation that the two-way transducer has taken so far. Note that this is indeed a finite property as only a prefix of the input has been visited by the computation of the two-way transducer. ◀

As an easy consequence of Theorem 3.1, let us observe that deterministic regular functions (effectively) preserve Büchi deterministic languages by inverse image. Analogue results hold for regular functions of finite (resp. infinite) words with regular languages.

▶ **Proposition 3.2.** *If $f : \Sigma^\omega \rightharpoonup \Gamma^\omega$ is deterministic regular and $L \subseteq \Gamma^\omega$ is Büchi deterministic, then $f^{-1}(L) \subseteq \Sigma^\omega$ is (effectively) Büchi deterministic.*

Proof. The function $f \circ \mathsf{id}_L$ (where $\mathsf{id}_L : \Gamma^\omega \rightharpoonup \Gamma^\omega$ is the identity function restricted to L) is deterministic regular. Its domain $f^{-1}(L)$ is Büchi deterministic by Proposition 2.11. ◀

Let us now focus on the converse of Theorem 3.1, i.e. showing that any deterministic regular function can be written as a composition of "basic" functions. As mentioned in introduction, regular functions of finite words can be written as compositions of map-copy-reverse (see Example 1.1) and sequential functions (computed by one-way transducers).

▶ **Theorem 3.3** ([6, Theorem 13])**.** *Over finite words, a function is regular if and only if it can (effectively) be written as a composition of* map-copy-reverse *and sequential functions.*

To state our similar result for deterministic regular functions of infinite words, we first recall formally the definition of sequential functions of infinite words.

▶ **Definition 3.4** (Sequential functions)**.** *A* deterministic one-way transducer *is a* 2-dT $(\Sigma, \Gamma, Q, q_0, F, \delta, \lambda)$ *such that for all $q \in Q$ and $\sigma \in (\Sigma \uplus \{\vdash\})$, $\delta(q, \sigma)$ has shape $(\_, \triangleright)$ (when defined). The class of (partial) functions over infinite words computed by one-way deterministic transducers is called* sequential functions of infinite words.

▶ **Example 3.5.** Any function that replaces some letter of its input by another letter is sequential. The functions replace and map-copy-reverse of Examples 1.4 and 2.3 are *not* sequential (this can be shown using a pumping argument). Observe that replace can be written as the composition of: a sequential function that replaces each $\sigma_i \in \{1, 2\}$ by $\sigma_i|$, the function map-copy-reverse, and finally a sequential function that uses the first copy of each block to determine the value of σ_i, and transforms the (mirror) second copy accordingly.

Now, we state the decomposition result, that also uses map-copy-reverse from Example 2.3. Its proof is somehow technical and it illustrates once more the main difficulty of this paper: deterministic regular functions are not able to check many properties about the "future".

▶ **Theorem 3.6.** *A function is deterministic regular if and only if it can (effectively) be written as a composition of* map-copy-reverse *and sequential functions of infinite words.*

Proof idea. In the case of finite words, the proofs of [7, 6] rely on Simon's factorization forests theorem [27]. They first build a factorization forest, and then use its structure to simulate the runs of a transducer. Furthermore, over finite words, such forests can be computed by a *rational function*, which is a composition of sequential functions and map-copy-reverse. We follow a similar proof sketch for infinite words, but the main issue is that factorization forests can no longer be computed by a composition of sequential functions and map-copy-reverse (their structure may depend on regular properties of the input). Thus we use instead a weakened version of forests, introduced by Colcombet under the name of *forward Ramseyan splits* [12]. Such splits can be computed with a sequential function. Our new techniques show how to simulate the runs of a transducer by using a forward Ramseyan split. ◀

4 Guarded MSO-transductions

In this section, we define the logic MSO over finite and infinite words, as well as MSO-transductions, and its guarded fragment. We also state the logical characterization of deterministic regular functions (Theorem 4.8).

MSO on infinite words. Infinite words over Σ are seen as structures of domain $\mathbb{N}$, over the signature $\mathcal{W}_\Sigma = \{S(x,y), (\sigma(x))_{\sigma\in\Sigma}\}$ which consists of the successor predicate $S(x,y)$, naturally interpreted as the successor over $\mathbb{N}$, and unary predicates $\sigma(x)$ for all $\sigma \in \Sigma$, interpreted as the set of positions labelled σ. Given an infinite word $u \in \Sigma^\omega$, we denote by G_u the structure it induces, and just u when it is clear that u denotes the structure G_u.

Monadic second-order formulas are defined as first-order logic formulas, which can additionally use quantifiers $\exists X, \forall X$ over sets of positions, and membership atomic formulas of the form $x \in X$, where x is a first-order variable while X is a set variable. We denote by $\mathsf{MSO}[\Sigma, S, \leq]$ (or just MSO when the predicates are clear from the context), the set of monadic second-order formulas over the word signature $\mathcal{W}_\Sigma$ extended with the order predicate $\leq$ (interpreted by the natural order on $\mathbb{N}$). It is well-known that the predicate $\leq$ is syntactic sugar. The semantics is defined as expected (details can be found in [28, 14] for instance). For a formula ϕ with sets of free first-order and set variables $\overline{x}, \overline{X}$ (we use the tuple notation which implicitly assumes an order between variables), we may write it $\phi(\overline{x}, \overline{X})$ to explicit the free variables of ϕ. We also denote by $Free(\phi)$ the free (first-order and set) variables of ϕ.Given a word w, an n-tuple of positions $\overline{p}$ of w and an m-tuple $\overline{P}$ of sets of positions of w, we write $w \models \phi(\overline{p}, \overline{P})$ to mean that the structure induced by w is a model of ϕ under assignments $\overline{p}$ and $\overline{P}$.

▶ **Example 4.1.** The formula $\mathsf{first}(x) = \forall y \cdot \neg S(y,x)$ is satisfied by any word and position x such that x is the first position to the left.

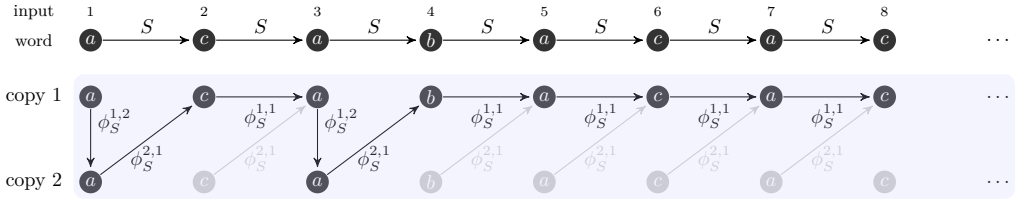
Over an alphabet Σ, any *closed* formula $\phi \in \mathsf{MSO}$ *defines* a regular language $L_\phi = \{u \in \Sigma^\omega \mid u \models \phi\}$. By Büchi-Elgot-Trakhtenbrot's theorem [29, 8, 21], it is known MSO defines precisely the class of regular languages over alphabet Σ: for any language L over Σ, L is regular if and only if $L = L_\phi$ for some $\phi \in \mathsf{MSO}$. MSO formulas can also be interpreted over finite word structures, whose domains are the (finite) set of word positions. It is also well-known that a language of finite words is regular iff it is MSO-definable.

MSO-transductions of infinite words. MSO-transductions define transformations of graph structures, and have been studied in the context of finite words by Engelfriet and Hoogeboom in [22] (see also [14] for a more recent introduction to MSO-transductions). The main result of [22] is a Büchi-like theorem: a function of finite words is MSO-definable if and only if it is regular (i.e. recognizable by a deterministic two-way transducer). This result was then lifted to functions of infinite words in [3], but deterministic two-way transducers may need infinite look-aheads to capture the full expressive power of MSO-transductions.

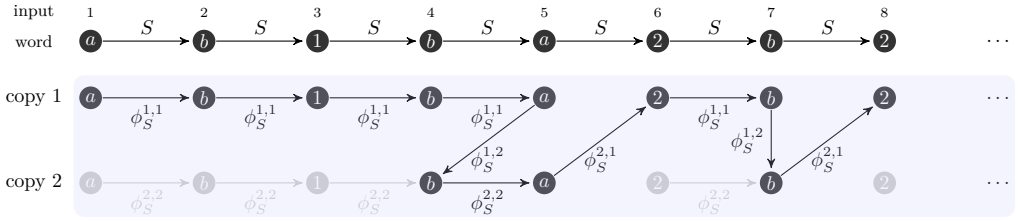
In an MSO-transduction, the output word structure is defined via an MSO interpretation over a fixed number k of copies of the input word (seen as a structure). Therefore, the nodes of the output word are copies 1 to k of the nodes of the input word. Output nodes are pairs (i,c) (often denoted i^c), for every copy c and input node i.

The output label and successor predicates are defined by MSO formulas with one and two free first-order variables respectively, interpreted over the input structure. For instance, over the output alphabet $\Gamma = \{a,b\}$, to set all the output labels to a, one just specifies the formulas $\phi_a^c(x) = \top$ and $\phi_b^c(x) = \bot$ for all copies c. The output successor predicate relates input nodes of possibly different copies, and is therefore defined by formulas of the form $\phi_S^{c,d}(x,y)$, indexed by copies $c, d \in \{1, \ldots, k\}$.

Finally, there is one distinguished copy c_0 together with a formula $\phi_{\text{fst}}^{c_0}(x)$, which must be satisfied by at most one node x. Intuitively, if the output structure is a word, this formula defines the first node of the output word. The domain of the output structure is composed of



(a) Input and output structures of the MSO-transduction for the function double of Example 1.2 on input word $acab(ac)^\omega$.



(b) Input and output structures of the MSO-transduction for the function copy of Example 1.3 on input word $ab1ba2b2\dots$.

Figure 1

all nodes that can be reached from the initial node x^{c_0} by following multiple successor edges. In general, the output structure of an input word u by an MSO-transduction $\mathcal{T}$ might not be an infinite word structure, in which case u is not in the domain of the function defined by $\mathcal{T}$.

Formally, an MSO-*transduction* over an input alphabet Σ and output alphabet Γ is a tuple $\mathcal{T} = (k, (\phi^c_\gamma(x))_{1\le c\le k, \gamma\in\Gamma}, (\phi^{c,d}_S(x,y))_{1\le c,d\le k}, c_0, \phi^{c_0}_{\text{fst}}(x))$ where $k \in \mathbb{N}\setminus\{0\}$, $1 \le c_0 \le k$ and for all input $u \in \Sigma^\omega$, there is at most one position i such that $u \models \phi^{c_0}_{\text{fst}}(i)$. We may omit c_0 in the tuple above.

We now formally define the semantics of MSO-transductions. Let $u \in \Sigma^\omega$ and $N \subseteq \mathbb{N}\times\{1,\dots,k\}$. We first define the set of output nodes that can be reached from N in zero or more steps. We let $\mathsf{Post}^0_u(N) = N$ and for all $\ell > 0$,

$$\mathsf{Post}^\ell_u(N) = \{j^d \mid \exists i^c \in \mathsf{Post}^{\ell-1}_u(N) \cdot u \models \phi^{c,d}_S(i,j)\} \text{ and } \mathsf{Post}^*_u(N) = \bigcup_{\ell\ge 0} \mathsf{Post}^\ell_u(N)$$

Given an MSO-transduction $\mathcal{T}$ as above, and input word $u \in \Sigma^\omega$, the *output structure*, denoted $\mathcal{T}(u)$, is the structure over signature $\mathcal{W}_\Gamma$ defined by the following interpretation:

- the domain is $D = \mathsf{Post}^*_u(\{i^{c_0} \mid u \models \phi^{c_0}_{\text{fst}}(i)\})$ (note that the argument of Post^*_u is either empty or a singleton)
- a node $i^c \in D$ is labelled $\gamma \in \Gamma$ if $u \models \phi^c_\gamma(i)$
- a node j^d is a successor of a node i^c if $u \models \phi^{c,d}_S(i,j)$.

The output structure $\mathcal{T}(u)$ may not be a word structure. For instance, a node might have multiple labels, $\mathcal{T}(u)$ may contain cycles, or branching. So we restrict semantically the function defined by $\mathcal{T}$ to word structures. Formally, the *function defined by* $\mathcal{T}$ is the function $[\![\mathcal{T}]\!] : \Sigma^\omega \rightharpoonup \Gamma^\omega$ whose graph is:

$$\{(u,v) \in \Sigma^\omega \times \Gamma^\omega \mid G_v \text{ (the structure associated with } v) \text{ is isomorphic to } \mathcal{T}(u)\}$$

We denote by MSOT the set of MSO-transductions and say that a function $f : \Sigma^\omega \rightharpoonup \Gamma^\omega$ is MSOT-definable if $f = [\![\mathcal{T}]\!]$ for some $\mathcal{T} \in \mathsf{MSOT}$.

▶ **Example 4.2.** We consider again the function double of Example 1.2, illustrated on Figure 1a and show how to define it with an MSO-transduction. Since some a must be duplicated, two copies are needed, so $k = 2$. Labels are preserved: $\phi^c_\sigma(x) = \sigma(x)$ for all $c \in \{1, 2\}$ and $\sigma \in \Sigma$. The first copy c_0 is 1, and $\phi^{c_0}_{\text{fst}}(x) = \mathsf{first}(x)$. The successor formulas distinguish if there is a b in the future or not. First, from the 2nd to the 1st copy, there is always a successor relation from a node to its successor in copy 1: $\phi^{2,1}_S(x, y) = S(x, y)$. There is a successor from x^1 to y^2 if $x = y$, x is labelled a and there is a b in the remaining infinite suffix starting at x: $\phi^{1,2}_S(x, y) = a(x) \wedge (x = y) \wedge \exists z \cdot x \leq z \wedge b(z)$. On the first copy, it depends on the label of the input: $\phi^{1,1}_S(x, y) = S(x, y) \wedge (a(x) \to (\forall z \geq x \cdot \neg b(z))$. On the second copy, there is never a predicate edge: $\phi^{2,2}_S = \bot$. On Figure 1a, the interpretation of those formulas is depicted, in bold if they are part of the output word, in light grey otherwise. One can see that the output structure induced by all the descendants of the first node (by the transitive closure of the successor relation) is isomorphic to the structure $G_{aacaab(ac)^\omega}$.

The function copy of Example 1.3, illustrated in Figure 1b, is definable by an MSOT with two copies ($k = 2$). Formulas $\phi^{c_0}_{\text{fst}}$ and ϕ^c_σ are the same as for double. Then:

$$\phi^{1,1}_S(x, y) = \phi^{2,2}_S(x, y) = S(x, y) \wedge \neg 2(y) \qquad \phi^{2,1}_S(x, y) = S(x, y) \wedge 2(y)$$
$$\phi^{1,2}_S(x, y) = \exists \mathsf{g} \cdot y < x \leq \mathsf{g} \wedge 2(\mathsf{g}) \wedge \forall z \leq y \cdot (S(z, y) \to (1(z) \vee 2(z))) \wedge$$
$$\forall t \cdot (y \leq t \leq x) \to (a(t) \vee b(t))$$

The class of regular functions of infinite words has been defined in [3] as the class of functions recognizable by deterministic two-way transducers extended with regular (infinite) look-ahead: to take a transition, such a transducer can query a regular oracle on the infinite current suffix (given as a deterministic parity automaton for example). Equivalently, this class corresponds to functions recognizable by (deterministic) SST: they work as BSST but are not forced to output the content of a special register infinitely often. Instead, the output of a run depends on the set of states that are seen infinitely often along that run, and can be "computed" only once the infinite input has been processed (see [3]) for more details. The following provides a logical characterization of the class of regular functions:

▶ **Theorem 4.3** ([3]). *A function $f : \Sigma^\omega \rightharpoonup \Gamma^\omega$ is regular if and only if it is* MSOT*-definable.*

The definition of MSOT in [3] is slightly different, but equivalent, to the definition we take in this paper.

Guarded MSO-transductions of infinite words. Guarded MSO formulas are a syntactical restriction of MSO formulas. This restriction requires all the free variables and quantifiers to be guarded by a first-order variable g, in the sense that quantifiers should only talk about positions which are *before* g (i.e. smaller than g). Intuitively, the satisfiability of a guarded formula on an infinite word only depends on the finite prefix up to position g. Formally, given two first-order variables x and g, we let $\mathsf{G}(x, \mathsf{g})$ be the formula $x \leq \mathsf{g}$ (x is guarded by g), and for a set variable X, we let $\mathsf{G}(X, \mathsf{g})$ be the formula $\forall x \in X, \mathsf{G}(x, \mathsf{g})$. Then, an MSO formula φ is *guarded by some variable g* if it is equal to $\psi(\mathsf{g}) \wedge \bigwedge_{\alpha \in Free(\psi)} \mathsf{G}(\alpha, \mathsf{g})$ for some $\psi(\mathsf{g})$ such that all its quantified subformulas, i.e. subformulas of the form $QX \cdot \psi'$ or $Qx \cdot \psi'$ for some $Q \in \{\exists, \forall\}$, are in one of the following forms:

$$(1)\ \forall x \cdot \mathsf{G}(x, \mathsf{g}) \to \zeta \quad (2)\ \exists x \cdot \mathsf{G}(x, \mathsf{g}) \wedge \zeta \quad (3)\ \forall X \cdot \mathsf{G}(X, \mathsf{g}) \to \zeta \quad (4)\ \exists X \cdot \mathsf{G}(X, \mathsf{g}) \wedge \zeta$$

An MSO formula is *guarded* if it is of the form $\exists \mathsf{g} \cdot \varphi$ where φ is guarded by g. We denote by MSO_g the set of guarded MSO-formulas. For conciseness, we may write $\forall x : \mathsf{g} \cdot \zeta$ instead of $\forall x \cdot \mathsf{G}(x, \mathsf{g}) \to \zeta$, and $\exists x : \mathsf{g} \cdot \zeta$ instead of $\exists x \cdot \mathsf{G}(x, \mathsf{g}) \wedge \zeta$ (and similarly for set variables).

▶ **Example 4.4.** All the formulas of the MSO-transduction of Example 4.2 defining the function double are guarded, or trivially equivalent to a guarded formula. For example, the formula $\mathsf{first}(x)$ is equivalent to the guarded formula $\exists \mathsf{g} \cdot x \leq \mathsf{g} \wedge \forall y \leq \mathsf{g} \cdot \neg S(y, x)$.

The order predicate $x \leq y$ is definable by the guarded formula $\exists \mathsf{g} \cdot x \leq \mathsf{g} \wedge y \leq \mathsf{g} \wedge y = \mathsf{g}$. Since $\neg(x \leq y)$ is equivalent to $y \leq x \wedge y \neq x$, we easily get that any MSO_g-formula ϕ is equivalent to an MSO_g-formula ψ in which the order predicate is only used to guard quantifiers, by existentially quantifying a global guard, guarding all the local guards used to define the atomic formulas of the form $z \leq t$ occurring in ϕ (assumed to occur positively).

▶ Remark 4.5. MSO_g formulas only talk about prefixes, in the following sense: If $\varphi = \exists \mathsf{g} \cdot \psi(\mathsf{g})$ is a closed guarded formula and $w \in \Sigma^\omega$, then $w \models \varphi$ if and only if there exists a finite prefix u of w such that $u \models \psi(\ell)$, where ℓ is the last position of u. This allows us to get the following immediate characterization: A language $L \subseteq \Sigma^\omega$ is MSO_g-definable if and only if there exists a regular language $F \subseteq \Sigma^*$ such that $L = F\Sigma^\omega$.

▶ **Definition 4.6** (Guarded MSO-transductions)**.** *A* guarded MSO-transduction *(*MSOT_g*) is an* MSO*-transduction all formulas of which are guarded.*

▶ **Example 4.7.** As explained in Example 4.4, all formulas of the MSO-transduction of Example 4.2 defining double are guarded, or trivially equivalent to a guarded formula.

We can now state the logical characterization of deterministic regular functions:

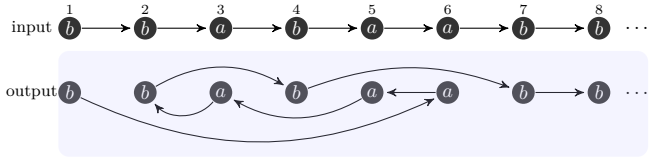
▶ **Theorem 4.8** (Logical characterization)**.** *A function $f : \Sigma^\omega \rightharpoonup \Gamma^\omega$ is deterministic regular if and only if it is* MSOT_g*-definable.*

The proof is given in Section 6. As an application of this result, since deterministic regular functions are (effectively) closed under composition by Theorem 3.1, we obtain that MSOT_g are (effectively) closed under composition as well. This is a well-known result for MSOT over finite strings [22], infinite strings [3] and more generally any structure [13], yet with purely logic-based and direct proofs, while we use here involved automata-based arguments (look-ahead removal). Indeed, composition closure of MSOT is obtained by formula substitutions. To compose two MSOT $\mathcal{T}_2 \circ \mathcal{T}_1$, the predicates occurring in $\mathcal{T}_2$ are substituted by their definition in $\mathcal{T}_1$. Such a direct proof idea does not work in the guarded fragment MSOT_g, as guarded formulas are not closed under negation.

Guarded MSO-transductions with order. We conclude this section by discussing an alternative definition of MSO_g-transductions, denoted $\mathsf{MSOT}_g[\leq]$, where instead of defining the output successor relation, it requires to define the total order $\leq$ of the output structure, with MSO_g formulas. This however allows to define uncomputable functions (in the sense of [18], see also Section 1), as stated by the following proposition:

▶ **Proposition 4.9.** *There exists an* $\mathsf{MSOT}_g[\leq]$ *which defines an uncomputable function.*

To prove this proposition, we show that the following uncomputable function h is definable with $\mathsf{MSOT}_g[\leq]$. Let $\Sigma = \Gamma = \{a, b\}$ and $\mathsf{er}_b : \Sigma^* \to \Sigma^*$ the (erasing) morphism defined by $\mathsf{er}_b(a) = a$ and $\mathsf{er}_b(b) = \varepsilon$. The function $h : \Sigma^\omega \rightharpoonup \Gamma^\omega$ is defined on inputs of the form bub^ω, for $u \in \{a, b\}^*$, by $h(bub^\omega) = b\mathsf{er}_b(u)b^\omega$. It can be shown that h is definable by a 1-copy $\mathsf{MSOT}_g[\leq]$. An example of output structure on input $bbabaab^\omega$ is given below (we depict only the successor predicate and not the order):



The output order formula for instance states that the b occurrences are ordered according to their input order, while the a occurrences are ordered in reverse. Moreover, it states that the first b occurrence is smaller than any a occurence, and that any a occurrence is smaller that any b occurrence but the first one.

Without the guarded restriction, it is known that two definitions of MSOT, with successor or with order, both define the class of regular functions of infinite words.

5 Two-way transducers with finite look-ahead

We extend deterministic two-way transducers with finite look-ahead. Transitions are additionally labelled by a regular language of *finite words*, called (finite) look-ahead. A transition with look-ahead L can only be taken if the remainder of the input sequence has a prefix that belongs to L. Such a finite prefix is called a *look-ahead witness* for L. To ensure determinism, if several look-aheads succeed, it is required that there is a unique shortest look-ahead witness. The transducer follows the transition which minimizes the length of the witness. If no look-aheads succeed the computation fails.

▶ **Definition 5.1** (Finite look-ahead). *A* deterministic two-way transducer with finite look-ahead ($2\text{-dT}^{\mathsf{FLA}}$) *is a tuple* $\mathcal{T} = (\Sigma, \Gamma, Q, q_0, F, \delta, \lambda)$ *where* $\Sigma, \Gamma, Q, q_0, F, \lambda$ *are defined as for deterministic two-way transducers w/o look-ahead,* δ *is a transition function* $Q \times (\Sigma \uplus \{\vdash\}) \times \mathcal{R}^*(\Sigma)) \rightharpoonup Q \times \{\triangleright, \triangleleft\}$ *where* $\mathcal{R}^*(\Sigma)$ *is the set of all regular languages of* finite *words over* Σ*. The function* δ *is required to have finite domain. The* look-ahead *for a transition* $(q, \sigma, L) \mapsto (q, d)$ *is* L*. Furthermore, we require that if* $\delta(q, \sigma, L)$ *and* $\delta(q, \sigma, L')$ *are defined, then* $L \cap L' = \emptyset$ *for all* $L, L' \in \mathcal{R}^*(\Sigma)$*,* $q \in Q$ *and* $\sigma \in \Sigma$*. Finally, it is assumed that the look-ahead languages are represented by deterministic finite automata.*

The semantics of a deterministic two-way transducer with finite look-ahead remains unchanged compared to the model without look-ahead. The only difference in the presence of look-ahead is when a transition is enabled: A transition with look-ahead L can only be taken if the remainder of the input sequence has a prefix that belongs to L. Formally, in a configuration (q, i) over input u, a transition of the form $\delta(q, \sigma, L)$ where $L \subseteq \Sigma^*$ is enabled if $u[i] = \sigma$ and there exists some $i < j$ such that $u[i{+}1{:}j] \in L$. The word $u[i{+}1{:}j]$ is called a *witness* for L. To ensure determinism, whenever the transducer is in a configuration (q, i), if several look-aheads $L_1, \ldots, L_k$ are enabled, the triggered transition is the unique (ensured by the disjointness requirement) transition with shortest witness.

Removing finite look-ahead. We know that infinite look-ahead is strictly more expressive than finite look-ahead. The natural question is how much expressiveness is gained by adding finite look-ahead to deterministic two-way transducers w/o look-ahead. As already explained in the introduction, any function defined by such a transducer is (Turing machine) computable: A Turing machine can memorize where it is in the input, verify which look-ahead succeeds, and continue the computation from the memorized position. A two-way transducer does not have the ability to memorize a position arbitrarily far away in the input. Hence, verifying (in the absence of some look-ahead "oracle") that some finite prefix of the remainder of

the input is a witness for some look-ahead and returning to a specific position becomes a problem to be solved. This problem is not unique to two-way transducers over infinite words, it also appears when some regular property of the remainder of a finite input word must be checked and subsequently the two-way transducer must return to the position it has been in before checking the property. On finite words, this task can be handled using the Hopcroft-Ullman [1] or the improved tree-outline construction [16]. However, these constructions rely on the fact that the input word is finite. We prove that this task can be also accomplished for infinite words using different techniques.

In the following, we show that no expressiveness is gained by allowing finite look-ahead.

▶ **Theorem 5.2** (Finite look-ahead removal). *Given a* 2-$\mathsf{dT}^{\mathsf{FLA}}$*, one can effectively construct an equivalent* 2-dT*.*

Proof sketch. The proof is divided into two parts. The main part is to translate a given 2-$\mathsf{dT}^{\mathsf{FLA}}$ into an equivalent BSST with bounded copy. We then use Theorem 2.9 to obtain an equivalent 2-dT. Given a deterministic two-way transducer *without* look-ahead, the standard approach to obtain an equivalent SST is to simulate the right-to-right runs of the deterministic two-way transducer on the so-far read prefix of the infinite input, store their outputs in registers and compose these registers in the right way (with the output of the "main" left-to-right run) to re-create the output of the two-way transducer. Since the two-way transducer is deterministic there is a global bound on the number of different right-to-right runs on any prefix of the input. The constructions presented in [3, 17, 9] are all built on this idea. In [2], equivalence between SST and two-way transducers on finite words is shown but the work exhibits no direct translation.

Our goal is to design a similar construction for deterministic two-way transducers *with* finite look-ahead. The main difficulty is that there is no global bound on the number of different runs that can occur on a prefix, if one takes additionally into account all the runs of the look-ahead automata that have been triggered so far. Alternatively, such a transducer can be seen as a non-deterministic transducer, which guesses which finite look-ahead will succeed and verifies it a posteriori, but there can be many look-ahead automata running in parallel.

Hence, we extend the standard construction to go from a deterministic two-way transducer to an SST by additionally taking all the possible look-ahead choices into account. This approach results in a tree structure representation of the possible runs (similar to a standard run-tree of a non-deterministic automaton, here the non-determinism is the look-ahead choice). A branch in such a tree corresponds to a possible run and the nodes additionally contain information to detect when look-ahead choices succeed or are doomed to fail. The size of the tree representations is kept bounded by sharing information and a relevant pruning strategy. The strategy takes care of removing branches whose look-ahead choices *cannot succeed* and (prefixes of) branches where the look-ahead choices *already have succeeded*. Applying this construction to a deterministic two-way transducer without look-ahead yields the standard translation construction. ◀

6 Logic-transducer correspondence: proof of Theorem 4.8

In this section, we give an overview of the proof of the logical characterization of Theorem 4.8. We first prove that any deterministic regular function is MSOT_g-definable. The proof is standard and uses same ideas as for regular functions of finite words [22] and infinite words [3].

▶ **Lemma 6.1.** *If a function* $f : \Sigma^\omega \multimap \Gamma^\omega$ *is deterministic regular, then it is* MSOT_g*-definable.*

Proof. The main idea is to define in MSOT_g the runs of a 2-$\mathtt{dT}$. Each copy of the MSOT_g represents a state of the 2-$\mathtt{dT}$, and there is a successor edges between node x^p to node y^q, where x, y are input positions and p, q are states, if and only if there exists a *finite* run from configuration (p, x) to configuration (q, y) which produces output symbols only in configuration (p, x) and (q, y). This property can be expressed by an MSO_g formula. ◀

Proving the converse of Lemma 6.1 is more involved. We first go to an intermediate model with MSO instructions, in the spirit of [22], called *jumping* MSO_g-transducers, proved to be equivalent to 2-$\mathtt{dT}$. It is a finite-state model which can (i) test MSO_g properties of the current position (called look-around), (ii) test safety constraints defined by MSO formulas, and (iii) jump from one position to another one with binary MSO_g formulas. Formally, it has a finite set of states (all final), and transitions are of the form $p \xrightarrow{\phi_{\mathrm{la}}(x)|w,\phi_{\mathrm{mv}}(x,y),\phi_{\mathrm{sf}}(x)} q$ where p, q are states, $\phi_{\mathrm{la}}, \phi_{\mathrm{mv}}$ are MSO_g formulas, ϕ_{sf} is an MSO formula, and w is a finite word. Look-around occurring on transitions with same source state are assumed to be pairwise disjoint (their conjunction is not satisfiable). The initial configuration is $(q_0, 0)$ where q_0 is the initial state. Whenever it is in a configuration (q, i), over an infinite word $u \in \Sigma^\omega$, it enables the transitions whose look-around $\phi_{\mathrm{la}}(i)$ holds on u, and select the transition with shortest witness. Call t this transition. It triggers t only if there exists j such that $\phi_{\mathrm{mv}}(i, j)$ holds and for all $k \geq i$, $u[{:}k] \models \phi_{\mathrm{sf}}(i)$ (otherwise the computation fails). It then outputs γ and moves to some position j such that $\phi_{\mathrm{mv}}(i, j)$ holds. Note that there could be several j, and therefore several runs on the same input in general. We thus make the following *assumption*, which can be described informally as follows: for any reachable configuration of the transducer from the initial configuration, there is always a unique j. Formally, for all infinite sequence of configurations $(q_0, i_0 = 0)(q_1, i_1)(q_2, i_2)\ldots$, for all $k \geq 0$, for any transition t triggered from configuration (q_k, i_k) to (q_{k+1}, i_{k+1}), if $\phi_{\mathrm{mv}}(x, y)$ is the jumping formula of t, then i_{k+1} is the unique position such that $\phi_{\mathrm{mv}}(i_k, i_{k+1})$ holds. As for two-way transducers, a sequence of configurations $(q_0, i_0 = 0)(q_1, i_1)\ldots$ is *accepting* if $\lim_{k\to\infty} i_k = \infty$ and it produces an infinite word.

We show that this model defines deterministic regular functions:

▶ **Lemma 6.2.** *Any jumping* MSO_g*-transducer defines a deterministic regular function.*

Sketch of proof. The proof goes in two steps. First, it is shown that jumping MSO_g-transducers are equivalent to *walking* MSO_g-transducers, i.e. MSO_g-transducers which moves (backward or forward) between successive positions. This step is standard (it appears e.g. in [22] in the non-guarded setting). Then, walking MSO_g-transducers are shown to be equivalent to an extension of 2-$\mathtt{dT}$ with finite look-around and safety constraints, then proved to be equivalent to 2-$\mathtt{dT}$ by transforming look-arounds into look-aheads, and then removing look-aheads (based on the techniques of Section 5) and safety constraints. ◀

▶ **Lemma 6.3.** *Any* MSO_g*-transduction is equivalent to a jumping* MSO_g*-transducer.*

Proof. Let $\mathcal{T} = (k, (\phi^c_\gamma)_{c\in[k],\gamma\in\Gamma}, (\phi^{c,d}_S)_{c,d\in[k]}, \phi^{c_0}_{\mathrm{fst}}(x))$ be an MSOT_g defining f. We construct a jumping MSO_g-transducer $\mathcal{T}'$ equivalent to $\mathcal{T}$. The set of states of $\mathcal{T}'$ is $\{0, 1\ldots, k\}$. In state 0, $\mathcal{T}'$ first jumps to the initial position, i.e. the position y which satisfies $\phi^{c_0}_{\mathrm{fst}}(y)$ and moves to state c_0. This is done by a transition going from state 0 to state c_0, with the trivial look-around and safety constraint $\top$, and the move $\phi_{\mathrm{mv}}(x, y) := \mathsf{first}(x) \wedge \phi^{c_0}_{\mathrm{fst}}(y)$. Then, it follows the successor relation of $\mathcal{T}$, and uses the label formulas to determine which label to output. Using safety constraints, $\mathcal{T}'$ also makes sure that the output graph structure is a word structure. In particular, they express that for any reachable node, there is exactly one label and at most one successor. There is no need to check that there is *at least* one

successor, because if there is none, then the run of $\mathcal{T}'$ stops and the input is not accepted, which is consistent with the semantics of $\mathcal{T}$ (the input is also rejected by $\mathcal{T}$ in that case). There is also no need to check that there is no cycle, because if there is some, then $\mathcal{T}'$ will never visit all input positions, and hence the input will be rejected, which is again consistent with the semantics of $\mathcal{T}$. Formally, for all copies $c, d \in \{1, \dots, k\}$ and output label γ, since $\phi_S^{c,d}(x,y)$ and $\phi_\gamma(x)$ are guarded, there are of the form $\phi_S^{c,d}(x,y) = \exists \mathsf{g} \cdot \psi_S(x,y,\mathsf{g})$ and $\phi_\gamma(x) = \exists \mathsf{g} \cdot \psi_\gamma(x,\mathsf{g})$. Then we add the following transition to $\mathcal{T}'$, from c to d:

$$c \xrightarrow{\phi_{\mathrm{la}}(x) := \exists \mathsf{g} \exists z \leq \mathsf{g} \cdot \psi_S^{c,d}(x,z,\mathsf{g}) \wedge \psi_\gamma^c(x,\mathsf{g}) \wedge \mathsf{disj}_{c,d,\gamma}(x,\mathsf{g}) \mid \gamma, \phi_{\mathrm{mv}}(x,y) := \phi_S^{c,d}(x,y), \phi_{\mathrm{sf}}(x)} d$$

in which $\mathsf{disj}_{c,d,\gamma}(x,\mathsf{g}) = \forall \mathsf{g}' \leq \mathsf{g} \cdot \bigwedge_{\gamma' \neq \gamma} \neg \psi_{\gamma'}^c(x,\mathsf{g}') \wedge \bigwedge_{d' \neq d} \forall z' \leq \mathsf{g}' \cdot \neg \psi_S^{c,d'}(x,z',\mathsf{g}')$ ensures disjointness of the look-around, and $\phi_{\mathrm{sf}}(x)$ equals

$$\begin{array}{ll} (\bigwedge_{d' \neq d} \forall y \cdot \neg \phi_S^{c,d'}(x,y)) \wedge & \text{no successor of } x \text{ in any copy } d' \neq d \\ (\forall y \forall y' \cdot (\phi_S^{c,c}(x,y) \wedge \phi_S^{c,c}(x,y')) \to y = y') \wedge & \text{at most one successor of } x \text{ in copy } c \\ (\bigwedge_{\gamma' \neq \gamma} \neg \phi_{\gamma'}^c(x) & \text{no other label for } x \end{array}$$

At this point, we remind the reader that safety constraints are not required to be defined by guarded formulas, as they are regular properties of finite words. However, the look-around and jumping formulas must be guarded, and it is indeed the case in the transition above.

Finally, note that $\mathcal{T}'$ satisfies the requirement that on infinite sequences of configurations $(q_0, i_0) \dots$, for all $k \geq 0$, i_{k+1} is the unique successor of i_k by the jumping formula. Indeed, if a sequence of configurations of $\mathcal{T}'$ is infinite, it implies that all safety constraints are satisfied, and they precisely make sure that there is no branching. ◀

As a corollary of Lemmas 6.3 and 6.2, we obtain the converse direction of Theorem 4.8:

▶ **Corollary 6.4.** *Any* MSOT_g*-definable function* f *is deterministic regular.*

7 Conclusion

In this paper, we have shown that the class of deterministic regular functions is characterized by computational models such as deterministic two-way transducers, deterministic two-way transducers with *finite* (regular) look-aheads, Büchi SST, by the logical formalism of *guarded* MSO-transductions, and by finite compositions of sequential functions and map-copy-reverse. The transformations between those models are effective. We have also shown that it is closed under composition, by extending to infinite words the known composition closure of deterministic two-way transducers, yet with new proof techniques. It is also conjectured that the class of deterministic regular functions is equal to the class of *continuous* regular functions (for the Cantor topology). It is already known that it includes the continuous letter-to-letter rational functions [23] and the strictly larger class of continuous rational functions [9]. All this, together with the fact that deterministic regular functions are computable, unlike regular functions, shows the robustness of this class.

References

1 Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. A general theory of translation. *Mathematical Systems Theory*, 3(3):193–221, 1969.

2 Rajeev Alur and Pavol Cerný. Expressiveness of streaming string transducers. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010*, volume 8 of *LIPIcs*, pages 1–12. Schloss Dagstuhl, 2010.

3 Rajeev Alur, Emmanuel Filiot, and Ashutosh Trivedi. Regular transformations of infinite strings. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012*, pages 65–74. IEEE Computer Society, 2012.

4 Rajeev Alur, Adam Freilich, and Mukund Raghothaman. Regular combinators for string transformations. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, page 9. ACM, 2014.

5 Nicolas Baudru and Pierre-Alain Reynier. From two-way transducers to regular function expressions. In *International Conference on Developments in Language Theory*, pages 96–108. Springer, 2018.

6 Mikołaj Bojańczyk and Rafał Stefański. Single-use automata and transducers for infinite alphabets. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

7 Mikołaj Bojańczyk. Polyregular Functions, 2018. doi:10.48550/arXiv.1810.08760.

8 J. R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6(1–6):66–92, 1960.

9 Olivier Carton and Gaëtan Douéneau-Tabot. Continuous rational functions are deterministic regular. In *47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022*, 2022.

10 Olivier Carton, Gaëtan Douéneau-Tabot, Emmanuel Filiot, and Sarah Winter. Deterministic regular functions of infinite words. *CoRR*, abs/2302.06672, 2023. doi:10.48550/arXiv.2302.06672.

11 Michal P. Chytil and Vojtěch Jákl. Serial composition of 2-way finite-state transducers and simple programs on strings. In *4th International Colloquium on Automata, Languages, and Programming, ICALP 1977*, pages 135–147. Springer, 1977.

12 Thomas Colcombet. A combinatorial theorem for trees. In *34th International Colloquium on Automata, Languages, and Programming, ICALP 2007*, 2007.

13 Bruno Courcelle. Monadic second-order definable graph transductions: A survey. *Theor. Comput. Sci.*, 126:53–75, 1994.

14 Bruno Courcelle and Joost Engelfriet. *Graph structure and monadic second-order logic: a language-theoretic approach*, volume 138. Cambridge University Press, 2012.

15 Luc Dartois, Emmanuel Filiot, and Nathan Lhote. Logics for word transductions with synthesis. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 295–304. ACM, 2018.

16 Luc Dartois, Paulin Fournier, Ismaël Jecker, and Nathan Lhote. On reversible transducers. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017*, volume 80 of *LIPIcs*, pages 113:1–113:12. Schloss Dagstuhl, 2017.

17 Luc Dartois, Ismaël Jecker, and Pierre-Alain Reynier. Aperiodic string transducers. *Int. J. Found. Comput. Sci.*, 29(5):801–824, 2018.

18 Vrunda Dave, Emmanuel Filiot, Shankara Narayanan Krishna, and Nathan Lhote. Synthesis of computable regular functions of infinite words. In *31st International Conference on Concurrency Theory (CONCUR 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

19 Vrunda Dave, Emmanuel Filiot, Shankara Narayanan Krishna, and Nathan Lhote. Synthesis of computable regular functions of infinite words. *Log. Methods Comput. Sci.*, 18(2), 2022.

20 Vrunda Dave, Paul Gastin, and Shankara Narayanan Krishna. Regular transducer expressions for regular transformations. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 315–324. ACM, 2018.

21 C. C. Elgot. Decision problems of finite automata design and related arithmetics. *In Transactions of the American Mathematical Society*, 98(1):21–51, 1961.

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

23 Emmanuel Filiot and Sarah Winter. Synthesizing computable functions from rational specifications over infinite words. In *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

24 Erich Grädel. On the restraining power of guards. *J. Symb. Log.*, 64(4):1719–1742, 1999.

25 Eitan M Gurari. The equivalence problem for deterministic two-way sequential transducers is decidable. *SIAM Journal on Computing*, 11(3):448–452, 1982.

26 Dominique Perrin and Jean-Éric Pin. *Infinite words: automata, semigroups, logic and games*. Academic Press, 2004.

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

28 Wolfgang Thomas. Languages, automata, and logic. In *Handbook of formal languages*, pages 389–455. Springer, 1997.

29 Boris Avraamovich Trakhtenbrot. Finite automata and logic of monadic predicates (in Russian). *Dokl. Akad. Nauk SSSR*, 140:326–329, 1961.