Deterministic and Nondeterministic Iterated Uniform Finite-State Transducers: Computational and Descriptional Power

Martin Kutrib[1], Andreas Malcher[1], Carlo Mereghetti[2(✉)], and Beatrice Palano[3]

[1] Institut für Informatik, Universität Giessen, Arndtstr. 2, 35392 Giessen, Germany
{kutrib,andreas.malcher}@informatik.uni-giessen.de
[2] Dipartimento di Fisica "Aldo Pontremoli", Università degli Studi di Milano, via Celoria 16, 20133 Milan, Italy
carlo.mereghetti@unimi.it
[3] Dipartimento di Informatica "G. degli Antoni", Università degli Studi di Milano, via Celoria 18, 20133 Milan, Italy
palano@unimi.it

Abstract. An iterated uniform finite-state transducer (IUFST) operates the same length-preserving transduction, starting with a sweep on the input string and then iteratively sweeping on the output of the previous sweep. The IUFST accepts or rejects the input string by halting in an accepting or rejecting state along its sweeps. We consider both the deterministic (IUFST) and nondeterministic (NIUFST) version of this device. We show that constant sweep bounded IUFSTs and NIUFSTs accept all and only regular languages. We study the size cost of removing nondeterminism as well as sweeps on constant sweep bounded NIUFSTs, and the descriptional power of constant sweep bounded IUFSTs and NIUFSTs with respect to classical models of finite-state automata. Finally, we focus on non-constant sweep bounded devices, proving the existence of a proper infinite nonregular language hierarchy depending on the sweep complexity both in the deterministic and nondeterministic case. Also, we show that the nondeterministic devices are always more powerful than their deterministic variant if at least a logarithmic number of sweeps is given.

Keywords: Iterated transducers · State complexity · Sweep complexity · Language hierarchies

1 Introduction

The notion of an iterated uniform finite-state transducer (IUFST) has been introduced in [13] and can be described as a finite transducer that iteratively sweeps from left to right over the input tape while performing the same length-preserving transduction at each sweep. In particular, the output of the previous sweep is

M. Anselmo et al. (Eds.): CiE 2020, LNCS 12098, pp. 87–99, 2020.
https://doi.org/10.1007/978-3-030-51466-2_8

taken as input for every new transduction sweep. (Throughout the paper, the attribute "uniform" indicates that the transduction is identical at each sweep: the transduction always starts from the same initial state on the leftmost tape symbol, and operates the same transduction rules at each computation step.) This model is motivated by typical applications of transducers or cascades of transducers, where the output of one transducer is used as the input for the next transducer. For example, finite-state transducers are used for the lexical analysis of computer programs and the produced output is subsequently processed by pushdown automata for the syntactical analysis. In [7], cascades of finite-state transducers are used in natural language processing. Another example is the Krohn-Rhodes decomposition theorem which shows that every regular language is representable as the cascade of several finite-state transducers, each one having a "simple" algebraic structure [8,10]. Finally, it is shown in [6] that cascades of deterministic pushdown transducers lead to a proper infinite hierarchy in between the deterministic context-free and the deterministic context-sensitive languages with respect to the number of transducers involved.

In contrast to all these examples and other works in the literature (see, e.g., [5,16,18]), where the subsequently applied transducers are in principle different and not necessarily length-preserving, the model of IUFSTs introduced in [13] requires that the same transducer is applied in every sweep and that the transduction is deterministic and length-preserving. More precisely, an IUFST works in several sweeps on a tape which initially contains the input string concatenated with a right endmarker. In every sweep, the finite-state transducer starts in its initial state at the first tape cell, is applied to the tape, and prints its output on the tape. The input is accepted or rejected, if the transducer halts in an accepting or rejecting state. In [13], IUFSTs both with a constant number and a non-constant (in the length of the input) number of sweeps are investigated. In the former case, it is possible to characterize exactly the set of regular languages. Thus, tight upper and lower bounds for converting IUFSTs into deterministic finite automata (DFAs) and vice versa are established. Furthermore, as always done for several models (see, e.g., [1–3]), the state complexity of language operations, that is, the costs in terms of the number of states needed for union, intersection, complementation, and reversal, is investigated in depth. Finally, the usually studied decidability questions such as emptiness, finiteness, equivalence, and inclusion are proved to be NL-complete, showing that these questions have the same computational complexity as for DFAs. For the case of a non-constant number of sweeps, the situation is quite different. It is shown that a logarithmic number of sweeps is sufficient to accept unary non-semilinear languages, while with a sublogarithmic number of sweeps only regular languages can be accepted. Moreover, the existence of a finite hierarchy with respect to the number of sweeps is obtained. Finally, all usually studied decidability questions are shown to be undecidable and not even semidecidable for IUFSTs performing at least a logarithmic number of sweeps.

In this paper, we enhance the model of IUFSTs by *nondeterminism*, thus obtaining their *nondeterministic* version (NIUFSTs). As in [13], we are interest in NIUFSTs exhibiting both a constant and non-constant number of sweeps.

Constant sweep bounded NIUFSTs are proved to accept exactly regular languages. So, their ability of representing regular languages in a very succinct way turns out to be worth investigating, as well as comparing such an ability with that of other more traditional models of finite-state automata. This type of investigation, whose importance is witnessed by a well consolidated trend in the literature, focuses on the *size* of formalisms for representing languages and is usually referred to as descriptional complexity. Being able to have "small" devices representing/accepting certain languages, leads to relevant consequences either from a practical point of view (less hardware needed to construct such devices, less energy absorption, less cooling problems, *etc.*), and from a theoretical point of view (higher manageability of proofs and representations for languages, reductions of difficult problems on general computing devices to the same problems on simpler machines, *etc.*). The reader is referred to, e.g., [11], for a thoughtful survey on descriptional complexity and its consequences.

Non-constant sweep bounded NIUFSTs are then studied for their computational power, i.e., the ability of accepting language families. In particular, such an ability is related to the number of sweeps as a function of the input length.

After defining NIUFSTs in Sect. 2, we discuss in detail an example that demonstrates the size advantages of NIUFSTs with a constant number of sweeps in comparison with its deterministic variant and the classical models of deterministic and nondeterministic finite automata (NFAs). Precisely, we exhibit a language accepted by a NIUFST such that any equivalent IUFST requires exponentially more states and sweeps, while any equivalent NFA (resp., DFA) requires exponentially (resp., double-exponentially) more states.

In Sect. 3, we study size advantages of NIUFSTs with a constant number of sweeps in more generality. By evaluating the state cost of sweep removal, we show that any NIUFST featuring n states and k sweeps can be simulated by an n^k-state NFA, and hence by a 2^{n^k}-state DFA as well. Next, we exhibit a unary (resp., binary) language witnessing the obtained size blow-up for turning a constant sweep NIUFST into an equivalent NFA (resp., DFA) is unavoidable.

In the last two sections, we consider NIUFSTs with a non-constant number of sweeps. First, we establish in Sect. 4 an infinite proper hierarchy with respect to the number of sweeps. Interestingly, this result also extends the known finite hierarchy in the deterministic case to an infinite hierarchy.

Finally, we study in Sect. 5 the question of whether the nondeterministic model is more powerful than the deterministic model. We get that the question can be answered in the affirmative if at least a logarithmic number of sweeps is provided. Moreover, we show that nondeterminism cannot be matched in power by the deterministic paradigm even if a sublinear number of sweeps is given.

2 Definitions and Preliminaries

We denote the set of positive integers and zero by $\mathbb{N}$. Given a set S, we write 2^S for its power set and $|S|$ for its cardinality. Let Σ^* denote the set of all words over the finite alphabet Σ. The *empty word* is denoted by λ, and $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. The length of a word w is denoted by $|w|$.

Roughly speaking, an iterated uniform finite-state transducer is a finite-state transducer which processes the input in multiple passes (also sweeps). In the first pass it reads the input word followed by an endmarker and emits an output word. In the following passes it reads the output word of the previous pass and emits a new output word. The number of passes taken, the *sweep complexity*, is given as a function of the length of the input. Here, we are interested in weak processing devices: we will consider length-preserving finite-state transducers, also known as Mealy machines [17], to be iterated.

Formally, we define a *nondeterministic iterated uniform finite-state transducer* (NIUFST) as a system $T = \langle Q, \Sigma, \Delta, q_0, \triangleleft, \delta, F_+, F_- \rangle$, where Q is the set of *internal states*, Σ is the set of *input symbols*, Δ is the set of *output symbols*, $q_0 \in Q$ is the initial state, $\triangleleft \in \Delta \setminus \Sigma$ is the *endmarker*, $F_+ \subseteq Q$ is the set of *accepting states*, $F_- \subseteq (Q \setminus F_+)$ is the set of *rejecting states*, and $\delta \colon Q \times (\Sigma \cup \Delta) \to 2^{Q \times \Delta}$ is the *transition function*, which is total on $(Q \setminus (F_+ \cup F_-)) \times (\Sigma \cup \Delta)$ and such that the endmarker is emitted only if it is read (i.e., no transition $(q, \triangleleft) \in \delta(p, x)$ is allowed, with $x \neq \triangleleft$). The NIUFST T *halts* whenever the transition function is undefined (which may happen only for states from $F_+ \cup F_-$) or T enters an accepting or rejecting state at the end of a sweep. Since transduction is applied in multiple passes, that is, in any but the initial pass it operates on an output of the previous pass, the transition function depends on input symbols from $\Sigma \cup \Delta$. We denote by $T(w)$ the set of possible outputs produced by T in a complete sweep on input $w \in (\Sigma \cup \Delta)^*$.

A *computation* of the NIUFST T on input $w \in \Sigma^*$ consists of a sequence of words $w_1, \ldots, w_i, w_{i+1}, \ldots \in (\Sigma \cup \Delta)^*$ satisfying $w_1 \in T(w\triangleleft)$ and $w_{i+1} \in T(w_i)$ for $i \geq 1$. The computation is halting if there exists an $r \geq 1$ such that T halts on w_r, thus performing r sweeps. The input word $w \in \Sigma^*$ is *accepted* by T if *all* computations on w are halting and at least one computation halts in an accepting state. The input word $w \in \Sigma^*$ is *rejected* if *all* computations on w are halting and *none* of the computations halts in an accepting state. Indeed, the output of the last sweep is not used. The language accepted by T is the set $L(T) \subseteq \Sigma^*$ defined as $L(T) = \{\, w \in \Sigma^* \mid w \text{ is accepted by } T \,\}$.

A NIUFST is said to be *deterministic* (IUFST) if and only if $|\delta(p, x)| \leq 1$, for all $p \in Q$ and $x \in (\Sigma \cup \Delta)$. In this case, we simply write $\delta(p, x) = (q, y)$ instead of $\delta(p, x) = \{(q, y)\}$ assuming that the transition function is a mapping $\delta \colon Q \times (\Sigma \cup \Delta) \to Q \times \Delta$.

We chose to designate our transductors as "uniform" since they perform the same transduction at each sweep: they always start from the same initial state on the leftmost tape symbol, operating the same transduction rules at every computation step. Yet, we quickly observe that a NIUFST is clearly a restricted version of a linear bounded automaton (see, e.g., [12]). So, any language accepted

by a NIUFST is context-sensitive. We leave it as an open problem to exhibit context-sensitive languages which cannot be accepted by any NIUFST.

Given a function $s\colon \mathbb{N} \to \mathbb{N}$, a NIUFST is said to be of *sweep complexity* $s(n)$ whenever it accepts or rejects any word of length n in at most $s(n)$ sweeps. In this case, we use the notation $s(n)$-NIUFST. Note that sweep complexity requires that any input is either accepted or rejected, that is, the NIUFST always halts. It is easy to see that a 1-IUFST (resp., 1-NIUFST) is actually a deterministic (resp., nondeterministic) finite automaton (DFA and NFA, respectively).

Throughout the paper, two accepting devices are said to be *equivalent* if and only if they accept the same language.

2.1 Accepting Languages by Iterated Transductions: An Example

In order to clarify the notion of acceptance by iterated transduction, we propose a language and design several accepting devices for such a language. For any integer $k \geq 2$, we define the block language

$$B_k = \{\, u_1\#u_2\#\cdots\#u_m \mid u_i \in \{0,1\}^k,\, m > 1,\, \exists i < m\colon u_i = u_m \,\}.$$

To accept B_k by a DFA, 2^{2^k+1} states are necessary and sufficient. On the other hand, an exponentially smaller NFA A may accept B_k as follows:

1. In a first phase, on each block in the input string, A stores the block in its finite control and then nondeterministically decides whether to keep the block or to ignore it. Along this phase, A checks the correct block structure of the input so far scanned as well. This phase takes 2^{k+1} states.
2. Once A decides to keep a block, say u, in its finite control, a second phase starts in which A scans the rest of the input checking the correct block structure and guessing another block w to be matched symbol-by-symbol against u. If matching is successful and w is the last block, then A accepts. This phase takes $2^{k+1} \cdot (k+1)$ states.

Globally, the NFA A features $2^{k+1} + 2^{k+1} \cdot (k+1) = 2^{k+1} \cdot (k+2)$ states.

Indeed, A can also be seen as a $2^{k+1} \cdot (k+1)$-state 1-NIUFST which outputs the scanned symbol at each step. However, paying by the number of sweeps (see, e.g., [15]), we can build a k-NIUFST T for B_k with only $O(k)$ states. Informally:

1. In a first sweep, T checks the correct block structure of the input string, nondeterministically chooses two blocks to be matched symbol-by-symbol, and compares the first symbol of the two blocks by storing the first symbol of the first block in its finite control and replacing these two symbols with a blank symbol.
2. At the ith sweep, T checks the ith symbol of the two blocks chosen in the first sweep by storing and blank-replacing symbols as explained at the previous point. To distinguish the first sweep (where both nondeterministic block choices and symbol comparisons take place) from the others (where only symbol comparisons take place), a special symbol can replace the first input symbol at the beginning of the first sweep.

It is not hard to see that $O(k)$ states are needed to check input formatting along the first sweep, and that a constant number of states suffices to blank-replacing and comparing input symbols. Indeed, after k sweeps all nondeterministically chosen blocks symbols are compared so that T may correctly accept or reject. This gives the claimed state and sweep bounds for T.

We remark that: (i) a $2^k(k+4)$-state 2^k-IUFST is designed in [13] for B_k, (ii) 2^{2^k+1} states are necessary and sufficient to accept B_k by a DFA, and that (iii) n^k states are sufficient for a DFA to simulate an n-state k-IUFST [13]. These facts, together with above designed $O(k)$-states k-NIUFST, show that NIUFSTs can be exponentially more succinct than IUFSTs either in the number of states, or the number of sweeps, or possibly both. Indeed, we also have that NIUFSTs can be exponentially more succinct than NFAs and double-exponentially more succinct than DFAs.

In the next section, we approach more generally the analysis of the descriptional power of NIUFSTs with respect to their deterministic counterparts and classical finite-state models.

3 Reducing Sweeps and Removing Nondeterminism

Let us begin by showing how to reduce sweeps from NIUFSTs and evaluate the state cost of reduction. We will then use this construction to reduce to one the sweeps of constant sweep bounded NIUFST, thus obtaining equivalent NFAs whose number of states will be suitably bounded.

Theorem 1. *Let $n, k > 0$ be integers. Every n-state k-NIUFST (resp., k-IUFST) can be converted to an equivalent n^i-state $\lceil \frac{k}{i} \rceil$-NIUFST (resp., $\lceil \frac{k}{i} \rceil$-IUFST).*

Proof. Let $T = \langle Q, \Sigma, \Delta, q_0, \triangleleft, \delta, F_+, F_- \rangle$ be a k-NIUFST with $|Q| = n$. To simplify the proof, we show how to transform a k-IUFST into an equivalent $\lceil \frac{k}{2} \rceil$-NIUFST T' with n^2 states, i.e, we prove the theorem for $i = 2$. We suppose that δ is completely defined. The opposite case is briefly discussed at the end of the proof.

The idea is to simulate two consecutive sweeps of T in one sweep. To this aim, the set of states of T' is defined as Q^2, its initial state is the pair (q_0, q_0), and the set of output symbols is Δ^2. In order to define the transition function δ' of T', we remark that: (i) the simulation of the first two sweeps takes place on the input string in Σ^*, while (ii) the simulations for the other sweeps take place on output strings in $(\Delta^2)^*$ but only the second component of scanned symbols from Δ^2 is to be considered for the computation in T'. Therefore, we use σ to denote either a symbol from Σ for situation (i) and the second component of a symbol from Δ^2 for situation (ii) and, with a slight abuse of notation, we define $\delta' \colon Q^2 \times (\Sigma \cup \Delta) \to 2^{Q^2 \times \Delta^2}$ as

$$\delta'((s_1, s_2), \sigma) \ni ((t_1, t_2), (\tau_1, \tau_2)) \Leftrightarrow \delta(s_1, \sigma) \ni (t_1, \tau_1) \text{ and } \delta(s_2, \tau_1) \ni (t_2, \tau_2).$$

A state $(f_1, f_2) \in Q^2$ is accepting (resp., rejecting) in T' whenever either $f_1 \in F_+$ (resp., $f_1 \in F_-$) or $f_2 \in F_+$ but $f_1 \notin F_-$ (resp, $f_2 \in F_-$ but $f_1 \notin F_+$). The reader may verify that the n^2-state $\lceil \frac{k}{2} \rceil$-NIUFST T' is equivalent to T.

In case δ of T is not completely defined (and this may happen only on states in $F_+ \cup F_-$), the number of states of T' does not increase as well. In fact, suppose T halts in $q \in F_+ \cup F_-$ in the middle of the input string on the jth sweep. We define δ' in such a way that the simulation of the jth sweep of T remains in q at every step and after the endmarker scanning as well.

It is not hard to see that this construction can be suitably adapted to merge $2 < i \leq k$ sweeps into one, thus yielding a NIUFST equivalent to T featuring $\lceil \frac{k}{i} \rceil$ sweeps and n^i states. Yet, it is also easy to see that the construction preserves determinism. □

The sweep reduction presented in Theorem 1 can be directly used to transform constant sweep bounded NIUFSTs into equivalent NFAs:

Theorem 2. *Let $n, k > 0$ be integers. Every n-state k-NIUFST can be converted to an equivalent NFA with at most n^k states.*

Proof. Given an n-state k-NIUFST, by Theorem 1 we can obtain an equivalent n^k-state 1-NIUFST which is actually a NFA. □

We obtain the optimality of the state blow-up in Theorem 2 by establishing an optimality condition for the size cost of sweep reduction proved in Theorem 1. To this aim, for $n, k > 0$, let the unary language

$$L_{n,k} = \{ a^{c \cdot n^k} \mid c \geq 0 \}.$$

In [13], an n-state k-IUFST for $L_{n,k}$ is provided, whereas any equivalent DFA or NFA needs at least n^k states. By using $L_{n,k}$ as witness language, we can show

Theorem 3. *Let $n, k, i > 0$ be integers such that i divides k. There exists an n-state k-NIUFST T such that any equivalent $\frac{k}{i}$-NIUFST T' cannot have less than n^i states.*

Proof. Suppose by contradiction, we can always design T' with $\frac{k}{i}$ sweeps and $s < n^i$ states. By using our construction in Theorem 1, we can obtain from T' an equivalent 1-NIUFST A with $s^{\frac{k}{i}} < n^k$ states. Clearly, having a single sweep, A is a NFA. By using this approach on the n-state k-IUFST above recalled for the language $L_{n,k}$, we could obtain an equivalent NFA featuring less than n^k states, a contradiction. □

By Theorem 3, one may easily obtain

Corollary 4. *For any integers $n, k > 0$, there exists an n-state k-NIUFST which cannot be converted to an equivalent NFA with less than n^k states.*

We conclude this section by discussing the optimal size cost of turning constant sweep bounded NIUFSTs into DFAs, i.e., the cost of removing both nondeterminism and sweeps at once:

Theorem 5. *Let $n, k > 0$ be integers. Every n-state k-NIUFST can be converted to an equivalent* DFA *with at most 2^{n^k} states.*

Proof. The result follows by first converting, according to Theorem 2, the n-state k-NIUFST into an equivalent n^k-state NFA which, in turn, is converted to an equivalent 2^{n^k}-state DFA by the usual powerset construction (see, e.g., [12]) □

The optimality of the size blow-up in Theorem 5 can be proved by considering the following language for any $n, k > 1$:

$$E_{n,k} = \{\, vbw \mid v, w \in \{a,b\}^*, |w| = c \cdot n^k \text{ for } c > 0 \,\}.$$

Theorem 6. *For any integers $m > 1$ and $k > 0$, there is an m-state k-NIUFST which cannot be converted to an equivalent* DFA *with less than $2^{(m-1)^k}$ states.*

Proof. As above quoted, an n-state k-NIUFST is given in [13], for the unary language $L_{n,k} = \{\, a^{c \cdot n^k} \mid c \geq 0 \,\}$. Such a device can be trivially converted to an n-state k-NIUFST T for the binary language $\{w \in \{a,b\}^* \mid |w| = c \cdot n^k \text{ for } c > 0\}$. We use T as a module for designing an $(n+1)$-state k-NIUFST T' accepting the language $E_{n,k}$. Informally, T' uses a separate state to scan the input string during the first sweep and, upon reading a symbol b, it nondeterministically decides whether to keep on reading the input string or to call the n-state k-IUFST module T which checks whether the length of the remaining part of the input string is a multiple of n^k.

On the other hand, by suitably using pigeonhole arguments (see, e.g., [4]), we can show that any DFA for $E_{n,k}$ needs at least 2^{n^k} states. In fact, suppose by contradiction a DFA A exists, accepting $E_{n,k}$ with less than 2^{n^k} states. Clearly, by counting arguments, there exist $\alpha, \beta \in \{a,b\}^*$ such that $\alpha \neq \beta$, $|\alpha| = |\beta| = n^k$, and the computation of A on both α and β ends up in the same (non-accepting) state q. Since $\alpha \neq \beta$, without loss of generality, we can assume that $\alpha = xay$ and $\beta = vbw$, for suitable $x, y, v, w \in \{a,b\}^*$ such that $|x| = |v|$ and $|y| = |w|$.

Now, consider any string $z \in \{a,b\}^*$ satisfying $|z| = n^k - |y| = n^k - |w|$, and let the strings $\alpha' = \alpha\, z$ and $\beta' = \beta\, z$. Note that $1 \leq |z| \leq n^k$, and so $|\alpha'| = |\beta'| \leq 2 \cdot n^k$. Therefore, the acceptance/rejection by A on α' and β' is only due to the symbol at position $|x| + 1 = |v| + 1$. This clearly means that the string α' does not belong to $E_{n,k}$ while β' does. However, in the computation on both α' and β', the DFA A reaches the same state q before consuming z and, being deterministic, the same state after consuming z. Hence, either A accepts both the two strings or rejects both of them, a contradiction. □

4 An Infinite Sweep Hierarchy

We now consider $s(n)$-NIUFSTs where $s(n)$ is a non-constant function. In what follows, by $\log n$ we denote the logarithm of n to base 2. In [13] it is proved that $o(\log n)$ sweep bounded IUFSTs accept regular languages only, and that

such a logarithmic sweep lower bound is tight for nonregular acceptance. Then, a three-level proper language hierarchy is established, where $O(n)$ sweeps are better than $O(\sqrt{n})$ sweeps which, in turn, are better than $O(\log n)$ sweeps for IUFST. Here, we extend the hierarchy to infinitely many levels for both IUFSTs and NIUFSTs.

Let $f : \mathbb{N} \to \mathbb{N}$ be a non-decreasing function. Its inverse is defined as the function $f^{-1}(n) = \min\{\, m \in \mathbb{N} \mid f(m) \geq n \,\}$. To show an infinite hierarchy dependent on some resources, where the limits of the resources are given by some functions in the length of the input, it is often necessary to control the lengths of the input so that they depend on the limiting functions. Usually, this is done by requiring that the functions are *constructible* in a desired sense. The following notion of constructibility expresses the idea that the length of a word relative to the length of a prefix is determined by a function. A non-decreasing computable function $f\colon \mathbb{N} \to \mathbb{N}$ is said to be *constructible* if there exists an $s(n)$-IUFST T with $s(n) \in O(f^{-1}(n))$ and an input alphabet $\Sigma \uplus \{a\}$, such that

$$L(T) \subseteq \{\, a^m v \mid m \geq 1, v \in \Sigma^*, |v| = f(m) \,\}$$

and such that, for all $m \geq 1$, there exists a word of the form $a^m v$ in $L(T)$. The $s(n)$-IUFST T is said to be a *constructor* for f.

Since constructible functions describe the length of the whole word dependent on the length of a prefix, it is obvious that each constructible function must be greater than or equal to the identity. In order to show that the class of functions that are constructible in this sense is sufficiently rich to witness an infinite dense hierarchy, we next show that it is closed under addition and multiplication:

Proposition 7. *Let $f\colon \mathbb{N} \to \mathbb{N}$ and $g\colon \mathbb{N} \to \mathbb{N}$ be two constructible functions. Then the functions $f + g$ and $f \cdot g$ are constructible as well.*

In [13] it is shown that the unary language $L_{\text{uexpo}} = \{\, a^{2^k} \mid k \geq 0 \,\}$ is accepted by some $s(n)$-IUFST with $s(n) \in O(\log n)$. The construction can straightforwardly be extended to show that the function $f(n) = 2^n$ is constructible. Moreover, again from [13], we know that $L_{\text{eq}} = \{\, u\$v \mid u \in \Sigma_1^*, v \in \Sigma_2^*, \text{ and } |u| = |v| \,\}$ is a language accepted by some $s(n)$-IUFST with $s(n) \in O(n)$, where Σ_1 is an alphabet not containing the symbol \$ and Σ_2 is an arbitrary alphabet. Even in this case, only a tiny modification shows that the identity function is constructible. These facts together with Proposition 7 yield, in particular, that the function $f(n) = n^x$ for all $x \geq 1$ is constructible.

In what follows, we use the fact, proved in [13], that the copy language with center marker $\{\, u\$u \mid u \in \{a, b\}^* \,\}$ is accepted by some $s(n)$-IUFST satisfying $s(n) \in O(n)$. The next theorem provides some language that separates the levels of the hierarchy.

Theorem 8. *Let $f\colon \mathbb{N} \to \mathbb{N}$ be a constructible function, T_f be a constructor for f with input alphabet $\Sigma \uplus \{a\}$, and b be a new symbol not belonging to $\Sigma \uplus \{a\}$. Then language $L_f = \{\, u\$uv \mid u \in \{a, b\}^*, v \in \Sigma^*, a^{2|u|+1}v \in L(T_f) \,\}$ is accepted by some $s(n)$-IUFST with $s(n) \in O(f^{-1}(n))$.*

Proof. Since the suffix v of a word $w \in L_f$ must be the suffix of some word $a^{2|u|+1}v$ in $L(T_f)$, we have that $|v| = f(2|u|+1)$ and $|w| = 2|u|+1+f(2|u|+1)$. Since $s(n)$ is claimed to be of order $O(f^{-1}(n))$, an $s(n)$-IUFST accepting L_f may perform at least $O(2|u|+1)$ many sweeps.

An $s(n)$-IUFST T accepting L_f essentially combines in parallel the acceptors for the copy language with center marker and the language $L(T_f)$. To this end, T establishes two tracks in its output. On the first track T simulates an acceptor for the copy language $\{\, u\$u \mid u \in \{a,b\}^* \,\}$, where the first symbol of Σ (i.e., the first symbol of v) acts as endmarker. In this way, the prefix $u\$u$ is verified. The result of the computation is written to the output track. This task takes $O(2|u|+1)$ sweeps. On the second track T simulates the constructor T_f, where all symbols up to the first symbol of Σ (i.e., all symbols of the prefix $u\$u$) are treated as input symbols a. In this way, T verifies that $|v| = f(2|u|+1)$. The result of the computation is written to the output track. This task takes $O(2|u|+1)$ sweeps.

Finally, T rejects whenever one of the above simulations ends rejecting. Instead, T accepts if it detects positive simulation results of the two tasks on the tracks. □

To show that the witness language L_f of Theorem 8 is not accepted by any $s(n)$-NIUFST with $s(n) \in o(f^{-1}(n))$, we use Kolmogorov complexity and incompressibility arguments. General information on this technique can be found, for example, in the textbook [14, Ch. 7]. Let $w \in \{a,b\}^*$ be an arbitrary binary string. Its Kolmogorov complexity $C(w)$ is defined to be the minimal size of a binary program (Turing machine) describing w. The following key fact for using the incompressibility method is well known: there exist binary strings w of *any* length such that $|w| \leq C(w)$.

Theorem 9. *Let $f\colon \mathbb{N} \to \mathbb{N}$ be a constructible function, T_f be a constructor for f with input alphabet $\Sigma \cup \{a\}$, and b be a new symbol not belonging to $\Sigma \cup \{a\}$. Then language $L_f = \{\, u\$uv \mid u \in \{a,b\}^*, v \in \Sigma^*, a^{2|u|+1}v \in L(T_f) \,\}$ cannot be accepted by any $s(n)$-NIUFST with $s(n) \in o(f^{-1}(n))$.*

Proof. Contrarily, let us assume that L_f is accepted by some $s(n)$-NIUFST $T = \langle Q, \Sigma \cup \{a,b\}, \Delta, q_0, \triangleleft, \delta, F_+, F_- \rangle$ with $s(n) \in o(f^{-1}(n))$.

We choose a word $u \in \{a,b\}^*$ long enough such that $C(u) \geq |u|$. Then, we consider an accepting computation of T on $u\$uv$, and derive a contradiction by showing that u can be compressed via T. To this end, we describe a program P which reconstructs u from a description of T, the length $|u|$, and the sequence of the $o(f^{-1}(n))$ many states $q_1, q_2, \ldots, q_r$ entered along the accepting computation at that moments in which T reads the first symbol after the \$ along its $o(f^{-1}(n))$ sweeps, n being the total length of the input.

Basically, the program P takes the length $|u|$ and enumerates the finitely many words $u'v'$ with $u' \in \{a,b\}^{|u|}$, $v' \in \Sigma^*$, and $|v'| = f(2|u|+1)$. Then, for each word in the list, it simulates by dovetailing all possible computations of T on $u'v'$ and, in particular, it simulates $o(f^{-1}(n))$ successive partial sweeps of T on $u'v'$, where the ith sweep is started in state q_i for $1 \leq i \leq r$. If the simulation ends accepting, we know that $u\$u'v'$ belongs to L_f and, thus, $u' = u$.

Let us consider the size of P. Let $|T|$ denote the constant size of the description of T, and $|P|$ denote the constant size of the program P itself. The binary description of the length $|u|$ takes $O(\log(|u|))$ bits. Each state of T can be encoded by $O(\log(|Q|))$ bits. So, we have

$$C(u) \in |P|+|T|+O(\log(|u|)+o(f^{-1}(n))\cdot O(\log(|Q|)) = O(\log(|u|))+o(f^{-1}(n)).$$

Since $n = 2|u| + 1 + f(2|u| + 1)$ and $f(n) \geq n$, for all $n \geq 1$, we have $n \in \Theta(f(2|u| + 1))$. So, we can conclude that $C(u) \in O(\log(|u|)) + o(|u|) = o(|u|)$. This contradicts our initial assumption $C(u) \geq |u|$, for u long enough. Therefore, T cannot accept L_f with sweep complexity $o(f^{-1}(n))$. □

Finally, we would like to remark that, due to our observation that all functions $f(n) = n^x$ are constructible for $x \geq 1$, it is an easy application of the above theorems to obtain the following infinite hierarchies with regard to the number of sweeps both in the deterministic and the nondeterministic case. Namely: *For every $x \geq 1$ we have that the set of all languages that are accepted by $s(n)$-IUFSTs ($s(n)$-NIUFSTs) with $s(n) \in O(n^{1/(x+1)})$ is properly included in the set of all languages that are accepted by $s(n)$-IUFSTs ($s(n)$-NIUFSTs) with $s(n) \in O(n^{1/x})$.*

5 Nondeterminism Beats Determinism on All Levels

We now turn to compare the computational power of IUFSTs and NIUFSTs. Since for sweep bounds of order $o(\log n)$ both variants accept regular languages only (see [9,13]), it remains to consider sweep bounds beyond $o(\log n)$. Here, we will show that there exist witness languages that are accepted by some nondeterministic $s(n)$-NIUFST with $s(n) \in O(\log n)$, but cannot be accepted by any deterministic $s(n)$-IUFST with $s(n) \in o(n)$, thus separating determinism from nondeterminism for almost all levels of the sweep hierarchy.

For any integer $k \geq 1$, let $\mathrm{bin}_k \colon \{0, 1, 2, \ldots, 2^k-1\} \to \{0, 1\}^k$ map any integer in the range from 0 to $2^k - 1$ to its binary representation of length k, starting from the left with the least significant bit and possibly completed with zeroes to the right. E.g., $\mathrm{bin}_4(5) = 1010$ and $\mathrm{bin}_4(12) = 0011$. We consider the language

$$D = \{\, a^k b^{2^k}\, \mathrm{bin}_k(0)u_0\, \mathrm{bin}_k(1)u_1 \cdots \mathrm{bin}_k(2^k-1)u_{2^k-1}\, \mathrm{bin}_k(i)u_i \mid k \geq 2,\ 1 \leq i \leq 2^k - 1,\ u_j \in \{a, b\}^k \text{ for all } 1 \leq j \leq 2^k - 1 \,\}.$$

Theorem 10. *The language D can be accepted by an $s(n)$-NIUFST satisfying $s(n) \in O(\log n)$.*

Proof. We sketch the construction of an $s(n)$-NIUFST T that accepts D with $s(n) \in O(\log n)$. The basic idea of the construction is to use two output tracks. So, during its first sweep, T splits the input into two tracks, each one getting the original input. In addition, T verifies if the structure of the input is correct, that is, if the input is of the form $a^+b^+0^+\{a, b\}^+(\{0, 1\}^+\{a, b\}^+)^+1^+\{a, b\}^+$ with at least two leading a's. If the form is incorrect, T rejects.

In subsequent sweeps, T behaves as follows. The original input on the first track is kept but the symbols can be marked, while on the second track the input is successively shifted to the right. More precisely, in any sweep the first unmarked symbol a in the leading a-block is marked. In the following b-block, every second unmarked symbol b is marked. In the further course of the sweep, the leftmost unmarked symbol in any $\{0,1\}$-block as well as in any $\{a,b\}$-block is marked. On the second track, the input is shifted to the right by one symbol, whereby the last symbol is deleted and some blank symbol is added at the left.

Let $k \geq 2$ be the length of the leading a-block. When the last of its symbols is marked, T checks in the further course of the sweep whether in the following b-block exactly one symbol remains unmarked, and whether in all remaining blocks the last symbol is being marked. Only in this case the computation continues. In all other cases T halts rejecting.

From the construction so far, we derive that if the computation continues then all but the second block have the same length, namely, length k. Moreover, since in the second block every second unmarked symbol has been marked during a sweep and one symbol is left, the length of the block is 2^k.

Next, T continues to shift right the content of the second track until the $\{0,1\}$-blocks are aligned with their neighboring $\{0,1\}$-blocks (except for the last one). This takes other k sweeps. In the next sweep, T checks if the $\{0,1\}$-block on the second track is an integer that is one less the integer in the aligned block on the first track. This can be done by adding one on the fly and comparing the result with the content on the first track. Only if the check is successful, T continues. Otherwise, it halts rejecting. In the former case, we get that the sequence of $\{0,1\}$-blocks are the numbers from 0 to $2^k - 1$ in ascending order.

In the next sweep, T guesses the $\{0,1\}$-block that has to match the rightmost $\{0,1\}$-block and marks it appropriately. Finally, this block together with its following $\{a,b\}$-block is symbolwise compared with the last $\{0,1\}$-block together with its following $\{a,b\}$-block in another $2k$ sweeps. To this end, note that T can detect that the last block follows when it scans a $\{0,1\}$-block consisting of 1's only. For the comparison, the symbols can further be marked appropriately.

Now, T accepts only if the guessed $\{0,1\}$-block together with its following $\{a,b\}$-block match the last $\{0,1\}$- and $\{a,b\}$-block. Otherwise T rejects. The construction shows that for any word from D there is one accepting computation and that only words from D are accepted. So, T accepts D.

Altogether, T performs at most $1+k+k+1+2k \in O(k)$ sweeps. The length of the input is $k + 2^k + (2^k + 1) \cdot 2k = O(k2^k)$. Since $\log(O(k2^k)) \in O(k)$, the NIUFST T obeys the sweep bound $s(n) \in O(\log n)$. □

To see that the language D is not accepted by any $s(n)$-IUFST with $s(n) \in o(n)$, we use again Kolmogorov complexity and incompressibility

Theorem 11. *The language D cannot be accepted by any $s(n)$-IUFST satisfying $s(n) \in o(n)$.*

Acknowledgements. The authors wish to thank the anonymous referees for useful comments and remarks.

References

1. Bednárová, Z., Geffert, V., Mereghetti, C., Palano, B.: The size-cost of Boolean operations on constant height deterministic pushdown automata. Theoret. Comput. Sci. **449**, 23–36 (2012)
2. Bednárová, Z., Geffert, V., Mereghetti, C., Palano, B.: Boolean language operations on nondeterministic automata with a pushdown of constant height. J. Comput. Syst. Sci. **90**, 99–114 (2017)
3. Bertoni, A., Mereghetti, C., Palano, B.: Trace monoids with idempotent generators and measure-only quantum automata. Nat. Comput. **9**(2), 383–395 (2010)
4. Bianchi, M.P., Mereghetti, C., Palano, B.: Complexity of promise problems on classical and quantum automata. In: Calude, C.S., Freivalds, R., Kazuo, I. (eds.) Computing with New Resources. LNCS, vol. 8808, pp. 161–175. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13350-8_12
5. Bordihn, H., Fernau, H., Holzer, M., Manca, V., Martín-Vide, C.: Iterated sequential transducers as language generating devices. Theoret. Comput. Sci. **369**(1–3), 67–81 (2006)
6. Citrini, C., Crespi-Reghizzi, S., Mandrioli, D.: On deterministic multi-pass analysis. SIAM J. Comput. **15**(3), 668–693 (1986)
7. Friburger, N., Maurel, D.: Finite-state transducer cascades to extract named entities in texts. Theoret. Comput. Sci. **313**(1), 93–104 (2004)
8. Ginzburg, A.: Algebraic Theory of Automata. Academic Press, Cambridge (1968)
9. Hartmanis, J.: Computational complexity of one-tape turing machine computations. J. ACM **15**(2), 325–339 (1968)
10. Hartmanis, J., Stearns, R.E.: Algebraic Structure Theory of Sequential Machines. Prentice-Hall, Upper Saddle River (1966)
11. Holzer, M., Kutrib, M.: Descriptional complexity - an introductory survey. In: Martín-Vide, C. (ed.) Scientific Applications of Language Methods, pp. 1–58. Imperial College Press (2010)
12. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Boston (1979)
13. Kutrib, M., Malcher, A., Mereghetti, C., Palano, B.: Descriptional complexity of iterated uniform finite-state transducers. In: Hospodár, M., Jirásková, G., Konstantinidis, S. (eds.) DCFS 2019. LNCS, vol. 11612, pp. 223–234. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23247-4_17
14. Li, M., Vitányi, P.M.B.: An Introduction to Kolmogorov Complexity and Its Applications, 3rd edn. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11298-1
15. Malcher, A., Mereghetti, C., Palano, B.: Descriptional complexity of two-way pushdown automata with restricted head reversals. Theoret. Comput. Sci. **449**, 119–133 (2012)
16. Manca, V.: On the generative power of iterated transductions. In: Words, Semigroups, and Transductions - Festschrift in Honor of G. Thierrin, pp. 315–327. World Scientific (2001)
17. Mealy, G.H.: A method for synthesizing sequential circuits. Bell Syst. Tech. J. **34**, 1045–1079 (1955)
18. Pierce, A.: Decision problems on iterated length-preserving transducers. Bachelor's thesis, SCS Carnegie Mellon University, Pittsburgh (2011)