Descriptional Complexity of Iterated Uniform Finite-State Transducers

Martin Kutrib, Andreas Malcher, Carlo Mereghetti, Beatrice Palano

Descriptional Complexity of Iterated Uniform Finite-State Transducers

Martin Kutrib[1], Andreas Malcher[1], Carlo Mereghetti[2⋆[0000−0002−7778−7257]], and Beatrice Palano[3[0000−0003−3948−4658]]

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

Abstract. We introduce the deterministic computational model of an *iterated uniform finite-state transducer (*IUFST*)*. A IUFST performs the same length-preserving transduction on several left-to-right sweeps. The first sweep takes place on the input string, while any other sweep processes the output of the previous one. The IUFST accepts or rejects upon halting in an accepting or rejecting state along its sweeps. First, we focus on constant sweep bounded IUFSTs. We study their descriptional power vs. deterministic finite automata, and the state cost of implementing language operations. Then, we focus on non-constant sweep bounded IUFSTs, showing a nonregular language hierarchy depending on sweep complexity. The hardness of some classical decision problems on constant sweep bounded IUFSTs is also investigated.

Keywords: Iterated transducers; State complexity; Sweep complexity; Decidability.

1 Introduction

Finite-state transducers are finite automata with an output and they have been studied at least since 1950s. A typical application of finite-state transducers is, for example, the lexical analysis of a computer program or an XML document. Here, the correct formatting of the input is verified, comments are removed, the correct spelling of the commands is checked, and the sequence of input symbols is translated into a list of tokens. The output produced is subsequently processed by a pushdown automaton that realizes the syntactic analysis. Another example is the use of cascading finite-state transducers. Here, one has a finite number of transducers $T_1, T_2, \ldots, T_n$, where the output of T_i is the input for the next transducer T_{i+1}. Such cascades of finite-state transducers have been used, for example, in [4] to extract information from natural language texts.

⋆ Corresponding author: *Carlo Mereghetti* – Dipartimento di Fisica "Aldo Pontremoli", via Celoria 16, 20133 Milano, Italy. Email: carlo.mereghetti@unimi.it

On the other hand, the Krohn-Rhodes decomposition theorem shows that every regular language can be represented as the cascade of several finite-state transducers each of which having a "simple" algebraic structure [6, 8]. Cascades of deterministic pushdown transducers are investigated in [3] and it is shown that a proper infinite hierarchy in between the deterministic context-free and deterministic context-sensitive languages exists with respect to the number of transducers applied. All the examples considered so far have in common that the subsequently applied transducers are, at least in principal, different transducers. Another point of view is taken in [2, 13], where subsequently applied *identical* transducers are studied. Such *iterated* finite-state transducers are considered as language generating devices starting with some symbol in the initial state of the transducer, iteratively applying in multiple sweeps the transducer to the output produced so far, and eventually halting in an accepting state of the transducer after a last sweep. These iterated finite-state transducers are quite powerful, since in the nondeterministic case non-recursive languages can be generated where the underlying transducer comprises of three states. Even in the deterministic case, one state suffices to generate the class of D0L Lindenmayer systems and two states are sufficient to generate languages which are neither context-free nor in 0L. It is an essential feature that the underlying finite-state transducer is not length-preserving. Several restrictions on finite-state transducers are studied in [15] with respect to the question of whether the (arbitrary) iteration of the restricted transducers is still computationally universal.

In this paper, we will consider iterated finite-state transducers as language accepting devices. In addition, to have a simple device, the underlying transducer is considered to be a Mealy machine [14], that is, a deterministic length-preserving device where each input symbol is translated according to its transition function into an output symbol. To be more precise, an *iterated uniform finite-state transducer* (IUFST) works in several sweeps on a tape which is initially the input 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.

We start our investigations of such devices having a *fixed number* $k \geq 1$ of sweeps. Since in this case the language accepted by a k-IUFST is always a regular language, it is of interest to compare such devices with deterministic finite automata (DFA) by investigating their *descriptional complexity* and the state cost of implementing language operations. General information on descriptional complexity can be found in the survey [9] and many results on the operational state complexity are surveyed in [5]. In addition, the NL-completeness of commonly considered decision problems on k-IUFSTs is pointed out.

We will also consider the case when the number of sweeps is *not bounded by a fixed finite number*. In this case, the resulting IUFSTs can be considered as restricted variants of one-tape Turing machines that iteratively sweep from left to right, starting at the first tape cell always in their initial state. It turns out that such devices can accept non-regular languages as soon as the number

of sweeps is at least the logarithm of the length of the input. Moreover, there is a hierarchy depending on the number of sweeps and all commonly studied decidability questions turn out to be not even semidecidable.

The paper is organized as follows. In Section 2, we present the model of IUFSTs, and provide an example of a language accepted by a constant sweep bounded IUFST, with a number of states which is exponentially smaller than that of any equivalent DFA. In Section 3, we focus on the descriptional power of *constant* sweep bounded IUFSTs (k-IUFSTs), providing example of languages where iterated transduction either helps or does not help in reducing the number of states of equivalent DFAs. Section 4 is first devoted to study the optimal state cost of simulations between k-IUFSTs and DFAs. Next, the impact on the number of states is studied, of modifying the number of sweeps on k-IUFSTs. The optimal state cost of implementing Boolean language operations and reversal on k-IUFSTs is investigated, and finally the NL-completeness of typical decision problems on k-IUFSTs is proved. In Section 5, we consider *non-constant* sweep bounded IUFSTs. First, we prove that $o(\lg n)$ sweep bounded IUFSTs accept regular languages only, and show that such a logarithmic sweep lower bound is tight for nonregular acceptance. Next, we exhibit a nonregular language hierarchy with respect to sweep complexity.

2 Definitions and Preliminaries

We denote the set of positive integers and zero $\{0, 1, 2, ...\}$ by $\mathbb{N}$. The *reversal* of a word w is denoted by w^R, and for the length of w we write $|w|$. The reversal of a language $L \subseteq \Sigma^*$ is denoted by L^R, its complement by $\overline{L}$. By $\lg n$ we denote the logarithm of n to base 2. Throughout the article two devices are said to be *equivalent* if and only if they accept the same language.

An iterated uniform finite-state transducer is basically a deterministic 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. It can be seen as a restricted variant of a one-tape Turing machine. The number of passes taken, the *sweep complexity*, is given as a function of the length of the input. Since here we are interested in weak preprocessing devices, we will consider length-preserving deterministic finite-state transducers, also known as Mealy machines.

Formally, we define an *iterated uniform finite-state transducer* (IUFST) 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 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 $\delta \colon Q \times (\Sigma \cup \Delta) \to Q \times \Delta$ is the *transition function*, which is total on $(Q \setminus (F_+ \cup F_-)) \times (\Sigma \cup \Delta)$ and where the endmarker is emitted only if it is read, i.e., there are no transitions $\delta(p, x) = (q, \triangleleft)$ with $x \neq \triangleleft$. The IUFST *halts* when the transition function is undefined (which may happen only for states from $F_+ \cup F_-$) or if it enters an accepting or rejecting state at the end

of a sweep. Since the transducer is applied in multiple passes, i.e., in any but the initial pass it operates on the output of the previous pass, the transition function depends on input symbols from $\Sigma \cup \Delta$. Let $v \in \Delta^*$ be the output produced by T on input $w \in (\Sigma \cup \Delta)^*$ in a complete sweep. Then we denote v by $T(w)$. A word $w \in \Sigma^*$ is *accepted* by an IUFST T if there is an $r \geq 1$ such that $w_1 = T(w\triangleleft)$, $w_{i+1} = T(w_i)$, $1 \leq i < r$, and the transducer T halts on w_r in an accepting state. That is, the initial input is a word over the input alphabet Σ followed by the endmarker, and the output computed after $r-1$ iterations drives T in a final sweep where it halts in an accepting state. Similarly, an input $w \in \Sigma^*$ is *rejected* by T if there is an $r \geq 1$ such that T halts in the final sweep on w_{r-1} in a rejecting state. Note that the output of the last sweep is not used. The language accepted by T is $L(T) = \{\, w \in \Sigma^* \mid w \text{ is accepted by } T \,\}$.

Let $s\colon \mathbb{N} \to \mathbb{N}$ be a function. If any word of length n is accepted or rejected in at most $s(n)$ sweeps, the IUFST is said to be of *sweep complexity* $s(n)$. In this case, we use the notation $s(n)$-IUFST.

In order to clarify this notion we continue with an example.

Example 1. For $k \geq 1$, we consider the language $E_k = \{a,b\}^* b \{a,b\}^{k-1}$, whose words are characterized by having the letter b at the kth position from the right. Any deterministic finite automaton (DFA) needs at least 2^k states to accept E_k. The following k-IUFST $T = \langle Q, \Sigma, \Delta, q_a, \triangleleft_0, \delta, F_+, F_- \rangle$ accepts E_k with 3 states only. Set $Q = \{q_a, q_b, q_-\}$, $\Sigma = \{a,b\}$, $\Delta = \{a, b, \triangleleft_0, \triangleleft_1, \ldots, \triangleleft_{k-1}\}$, $F_+ = \{q_b\}$, $F_- = \{q_-\}$, and specify the transition function as:

(1) $\delta(q_a, a) = (q_a, a)$
(2) $\delta(q_a, b) = (q_b, a)$
(3) $\delta(q_a, \triangleleft_i) = (q_a, \triangleleft_{i+1})$, for $0 \leq i \leq k-2$
(4) $\delta(q_a, \triangleleft_{k-1}) = (q_-, \triangleleft_{k-1})$
(5) $\delta(q_b, a) = (q_a, b)$
(6) $\delta(q_b, b) = (q_b, b)$
(7) $\delta(q_b, \triangleleft_i) = (q_a, \triangleleft_{i+1})$, for $0 \leq i \leq k-2$
(8) $\delta(q_b, \triangleleft_{k-1}) = (q_b, \triangleleft_{k-1})$

The basic idea of the construction is to shift symbol by symbol the input word to the right (states q_a and q_b), whereby an a is inserted at the first position and the last symbol is removed. In this way, in the first $k-1$ sweeps the input is shifted $k-1$ positions to the right. Now it is sufficient to check whether the last symbol is a b which is done in the last sweep. The number of the current sweep is given by the index of the endmarker. ■

3 Iterated Transductions vs. DFAs

As seen in Example 1, iterated transduction may lead to dramatically decrease the number of states for accepting regular languages. Here, we propose another family of languages showing the high descriptional power of k-IUFST. For any integer $k \geq 2$, we define the language

$$B_k = \{\, b_1 \# b_2 \# \cdots \# b_m \mid b_i \in \{0,1\}^k,\ m > 1,\ \exists i < m\colon b_i = b_m \,\}.$$

For recognizing B_k on a DFA, 2^{2^k+1} states are necessary and sufficient. Instead:

Theorem 2. *For any $k \geq 2$, the language B_k can be recognized by a 2^k-IUFST with $2^k(k+4)$ states.*

Proof. We informally explain the behaviour of a 2^k-IUFST T for B_k. At the first sweep, T stores the first block b_1 and "erase" any other occurrence of $b_1\#$ by replacing it with $b_1\#_d$. In case $b_1 = b_m$, T accepts, otherwise it rewrites $\triangleleft$ by $\triangleleft_1$ to store the sweep number, and continues. At the ith sweep, T searches a non-erased block to store, erases the occurrences of the stored block along the string, and matches it against b_m. If the stored block matches b_m, T accepts, otherwise it rewrites $\triangleleft_{i-1}$ with $\triangleleft_i$ to record the sweep number, and continues. Clearly, within 2^k sweeps, T correctly accepts or rejects the input string. The number of states of T is as follows: $2^{k+1} - 1$ to store a block not yet checked in previous sweeps, plus $2^k(k+2)$ states to perform matching with every subsequent block along the string and to fix the outcome of matching, plus 1 state to continue the computation when, by reading the endmarker, T realizes it is not the last sweep. So, the total number of states is $2^{k+1} - 1 + 2^k(k+2) + 1 = 2^k(k+4)$. □

However, some languages are so hard that even iterated transduction cannot reduce the size for their recognition. We are going to present a family of unary languages for which using k-IUFST does not help in reducing the number of states of equivalent DFAs. Let p be any prime number. We define

$$L_p = \{\, a^{m \cdot p} \mid m \geq 0 \,\}.$$

Theorem 3. *Let $k \geq 1$. Then p states are necessary and sufficient for a k-IUFST to accept L_p.*

Proof. For the upper bound, consider the minimal DFA A for L_p, which has exactly p states. The DFA A can be turned into a p-state k-IUFST, as explained in Lemma 6 below.

For the lower bound, suppose by contradiction there exists a k-IUFST T accepting L_p with $x < p$ states. Consider an input string $a^{m \cdot p}$, for m large enough, accepted by T at the endmarker after exactly k sweeps (these acceptance requirements can be assumed without loss of generality).

Since the input is unary, in the first sweep T runs into a cycle of length, say c_1. Let β_1 be the output produced while running through one cycle. Then the input $a^{m \cdot p}$ is rewritten as $\alpha_{1,1}\beta_1^*\gamma_{1,1}$ with $|\beta_1| = c_1 = x_1 \leq x < p$, such that T enters and exits the cycle on β_1 in the same state. In the second sweep, T eventually runs into a cycle of length, say c_2, on processing β_1^*. Since $|\beta_1| = x_1$, we derive $c_2 = x_2 \cdot x_1$, for some $x_2 \leq x < p$. Let β_2 be the output produced while running through one cycle. Then T rewrites β_1^* as $\alpha_{2,1}\beta_2^*\gamma_{2,1}$ with $|\beta_2| = x_2 \cdot x_1$, and the current string after the second sweep has the form $\alpha_{1,2}\alpha_{2,1}\beta_2^*\gamma_{2,1}\gamma_{1,2}$. This process continues until the kth sweep, where T eventually runs into a cycle of length, say c_k, on processing β_{k-1}^*. Since $|\beta_{k-1}| = x_{k-1} \cdot x_{k-2} \cdots x_1$, we derive $c_k = x_k \cdot x_{k-1} \cdots x_1$, for some $x_k \leq x < p$. Let β_k be the output produced while running through one cycle. Then T rewrites β_{k-1}^* as $\alpha_{k,1}\beta_k^*\gamma_{k,1}$ with $|\beta_k| = x_k \cdot x_{k-1} \cdots x_1$ and the current string w after the kth sweep has the form

$$\alpha_{1,k}\alpha_{2,k-1} \cdots \alpha_{k,1}\beta_k^{\ell}\gamma_{k,1} \cdots \gamma_{2,k-1}\gamma_{1,k},$$

for some $\ell \geq 1$. Since the string w is accepted by T at the end of the kth sweep, also the string $\alpha_{1,k}\alpha_{2,k-1}\cdots\alpha_{k,1}\beta_k^{\ell-1}\gamma_{k,1}\cdots\gamma_{2,k-1}\gamma_{1,k}$ is accepted by T at the end of the kth sweep. But $|\alpha_{1,k}\alpha_{2,k-1}\cdots\alpha_{k,1}\beta_k^{\ell-1}\gamma_{k,1}\cdots\gamma_{2,k-1}\gamma_{1,k}| = m\cdot p - |\beta_k| = m\cdot p - c_k$. Since $c_k = x_k\cdot x_{k-1}\cdots x_1$ and the x_i are all less than p, and p is prime, $m\cdot p - c_k$ cannot be a multiple of p. This means that T accepts a string not in L_p, a contradiction. □

4 Descriptional Complexity

In this section, we approach in a more general way the study of the descriptional power of k-IUFST vs. DFAs by providing general simulations between the two models. First, we consider the cost of turning a k-IUFST into an equivalent DFA:

Lemma 4. *Let $k > 0$ be an integer. Every n-state k-IUFST can be converted to an equivalent DFA with at most n^k states.*

The result presented in Lemma 4 turns out to be optimal.

Lemma 5. *There exists a family $\{L_{n,k}\}_{n,k>0}$ of unary languages such that each language $L_{n,k}$ is accepted by an n-state k-IUFST, whereas any equivalent DFA needs at least n^k states.*

Proof. For any $n, k > 0$, we define $L_{n,k} = \{\, a^{c\cdot n^k} \mid c \geq 0 \,\}$.

The k-IUFST $T = \langle Q, \Sigma, \Delta, q_1, \lhd_0, \delta, F_+, F_-\rangle$ accepts $L_{n,k}$ with n states only. We set $Q = \{q_1, q_2, \ldots, q_n\}$, $\Sigma = \{a\}$, $\Delta = \{a, 1, 2, \ldots, n, \bar{n}, \lhd_0, \lhd_1, \ldots, \lhd_{k-1}\}$, $F_+ = \{q_n\}$, $F_- = \{q_2, \ldots, q_{n-1}\}$, and specify the transition function as:

$$(1)\quad \delta(q_i,\sigma) = \begin{cases} (q_{i+1}, i) & \text{if } \sigma \in \{a, n\} \\ (q_i, i) & \text{if } \sigma \in \{1, 2, \ldots, n-1, \bar{n}\} \end{cases} \qquad \text{for } 1 \leq i \leq n-1$$

$$(2)\quad \delta(q_n,\sigma) = \begin{cases} (q_1, n) & \text{if } \sigma \in \{a, n\} \\ (q_n, \bar{n}) & \text{if } \sigma \in \{1, 2, \ldots, n-1, \bar{n}\} \\ (q_2, \sigma) & \text{if } \sigma \in \{\lhd_0, \lhd_1, \ldots, \lhd_{k-1}\} \end{cases}$$

$$(3)\quad \delta(q_1, \lhd_i) = (q_1, \lhd_{i+1}) \qquad \text{for } 0 \leq i \leq k-2$$

$$(4)\quad \delta(q_1, \lhd_{k-1}) = (q_n, \lhd_{k-1})$$

The basic idea is that, along the first sweep, T checks whether the length of the input string is a multiple of n by rewriting it as a sequence of consecutive blocks of the form "$12\cdots n$", followed by $\lhd_1$. For the second sweep, T checks whether the length of the input string is a multiple of n^2 by rewriting n consecutive blocks "$12\cdots n$" with the block "$1^n\ 2^n\ \cdots\ (n-1)^n\ \bar{n}^{n-1}n$", followed by $\lhd_2$. For the third sweep, T checks whether the length of the input string is a multiple of n^3 by rewriting n consecutive blocks "$1^n\ 2^n\ \cdots\ (n-1)^n\ \bar{n}^{n-1}n$" with the block "$1^{n^2}\ 2^{n^2}\ \cdots\ (n-1)^{n^2}\ \bar{n}^{n^2-1}n$", followed by $\lhd_3$. Clearly, by Transition (4), the input string can be accepted after the kth sweep only upon reading $\lhd_{k-1}$. In this case, the length of the input is easily seen to be a multiple of n^k.

The fact that n^k states are necessary for a DFA to accept $L_{n,k}$ follows from a trivial pumping argument. □

Let us now focus on the opposite simulation, that is, DFAs by k-IUFST.

Lemma 6. *For $n \geq 3$, every n-state* DFA *can be converted into an equivalent n-state k-*IUFST*, with $k \geq 1$.*

Lemma 6 is stated for at least three states. For DFAs with less than three states, we can choose either to maintain the same number of states and perform one sweep only, or to add a new state and perform k sweeps.

One may expect that the size of a k-IUFST may always be decreased by increasing the number of sweeps. However by Theorem 3, the construction provided by Lemma 6 is optimal.

4.1 States versus Sweeps

Here we turn to study the impact of modifying the number of sweeps on the size of iterated transducers. The possibility of trading states for input sweeps and vice versa has been investigated in the literature for several models of computations (see, e.g., [12]). Let us start with a simple observation on the minimal number of states necessary to establish a sweep complexity of at least two.

Lemma 7. *Let $k \geq 2$ be an integer and T be some k-*IUFST *with input alphabet Σ. Unless T accepts the trivial languages $\emptyset$ or Σ^*, it has at least three states.*

Proof. Let $L(T)$ be non-trivial. Then there are inputs that have to be accepted as well as inputs that have to be rejected. To this end, two states are necessary. On the other hand, T must not be in one of these two states at the end of the first sweep. Otherwise it would halt and could not start the second sweep. So, an additional state is necessary. □

Concerning the relations between the necessary number of states and the number of sweeps we have the following situation: Theorem 3 shows that there are languages for which additional sweeps do not help to decrease the number of states at all. By Lemma 4, any n-state k-IUFST can be converted into an equivalent DFA with at most n^k states. Conversely, due to Lemma 7 we cannot reduce the number of states below three by increasing the number of sweeps. So, there is an upper bound for the number of sweeps that may help. In other words, for any regular language L we have a fixed sweep range from 1 to some k_L in which we can trade states for sweeps and vice versa. For example, the sweep range for language L_p of Theorem 3 is given by $k_{L_p} = 1$, that is, it is just one sweep. In general, the next theorem gives a lower bound for the number of necessary states when one sweep is omitted.

Theorem 8. *Let $k \geq 2$ and $n \geq 3$ be integers, and T be an n-state k-*IUFST *such that the minimal* DFA *for $L(T)$ has n^k states. Then any $(k-1)$-*IUFST *for $L(T)$ must have at least $\lceil n^{k/k-1} \rceil$ states.*

Proof. Assume a $(k-1)$-IUFST T' for $L(T)$ with strictly less than $\lceil n^{k/k-1} \rceil$ states. If $n^{k/k-1}$ is an integer, this means at most $n^{k/k-1} - 1$ states. So, by Lemma 4, we could construct from T' a DFA for $L(T)$ with at most $(n^{k/k-1} - 1)^{k-1} < n^k$ states, a contradiction. Similarly, if $n^{k/k-1}$ is not an integer, we have at most $\lfloor n^{k/k-1} \rfloor$ states. Again, by Lemma 4, we could construct a DFA accepting $L(T)$ with $(\lfloor n^{k/k-1} \rfloor)^{k-1} < n^k$ states, a contradiction. □

Example 9. For $k \geq 2$ and $n \geq 3$, let us consider the language $L_{n,k}$ of Lemma 5, where an n-state k-IUFST accepting $L_{n,k}$ is constructed. Moreover, we noticed that the minimal DFA for $L_{n,k}$ has n^k states. So, any $(k-1)$-IUFST T' accepting $L_{n,k}$ has at least $\lceil n^{k/k-1} \rceil$ states. ■

Theorem 8 shows that omitting one sweep in the sweep range may increase the number of necessary states. However, once we have this increased number of states, the next question is whether this number of states can be used to decrease the number of sweeps furthermore for free. For example, assume that we have a 3-state 16-IUFST. Then Theorem 8 says that any equivalent 15-IUFST needs at least 4 states. However, since 4^{14} is still greater than 3^{16}, four states could be enough to reduce the number of sweeps by two. The next theorem provides a gradual reduction of the number of sweeps and the necessary states exemplarily for the language $E_k = \{a,b\}^* b \{a,b\}^{k-1}$ of Example 1.

Theorem 10. *Let $\ell \geq 1$ and $k = 2^\ell$ be integers. Then the useful sweep range for language E_k is from 1 to 2^ℓ. Moreover, for any $i \in \{0, 1, \ldots, \ell - 1\}$ there is a 2^i-IUFST accepting E_k with $2^{2^{\ell-i}}$ states, and for $i = \ell$ there is a 2^i-IUFST accepting E_k with 3 states.*

Proof. For $i = 0$ the assertion follows since, in this case, we have a one-sweep IUFST which needs as many states as a DFA to accept E_k, i.e., 2^{2^ℓ} states as claimed. For $i = \ell$, the assertion follows immediately from the construction of Example 1. For the remaining cases we modify the construction of Example 1 as follows. Instead of shifting the input k times by one position to the right, now we shift it 2^i times by $2^{\ell-i}$ positions to the right. Since $2^i \cdot 2^{\ell-i} = 2^\ell = k$, altogether it is shifted again k positions to the right, so the construction does it. To shift the input by $2^{\ell-i}$ positions, $2^{2^{\ell-i}}$ states are necessary. For $i = \ell$, Lemma 7 gives the lower bound of three states, where one state is needed to start a new sweep. For $i < \ell$, there are enough states to be used to this purpose, i.e., that are neither accepting nor rejecting and can be entered upon reading the endmarker. □

4.2 The State Cost of Language Operations on k-IUFSTs

As naturally done for many models of computation accepting regular languages (see, e.g., [1]), we now analyze the state complexity of language operations on k-IUFSTs. To this aim, we asssume that their transition functions are always defined, so that acceptance/rejection takes place on the endmarker only, at the jth sweep, for some $1 \leq j \leq k$. If not, transition functions can be easily completed

by adding at most two states where we maintain the accept or reject outcome obtained in the middle of the input, while reaching the endmarker.

Let us start with Boolean operations. The first we consider is intersection:

Theorem 11. *Let $m, n, k \geq 1$ be integers, T_1 be an m-state k-IUFST and T_2 be an n-state k-IUFST. Then $m \cdot n$ states are sufficient for a k-IUFST to accept $L(T_1) \cap L(T_2)$.*

The cost of implementing intersection given in Theorem 11 is optimal, as proved in the following

Theorem 12. *Let $k \geq 1$ be an integer. There exist infinitely many integers $m, n > 1$ such that an m-state k-IUFST T_1 and an n-state k-IUFST T_2 can be built, for which $m \cdot n$ states are necessary to accept $L(T_1) \cap L(T_2)$ on a k-IUFST.*

Proof. Let m and n be co-prime, that is, $\gcd(m, n) = 1$. As explained in the proof of Lemma 5, the languages $L_{m,k}$ and $L_{n,k}$ can be accepted, respectively, by an m-state k-IUFST T_1 and n-state k-IUFST T_2, . Since m and n are co-prime, we have that $L(T_1) \cap L(T_2) = L_{m \cdot n,k}$. This latter language cannot be accepted with less than $m \cdot n$ states on any k-IUFST T. Otherwise, by applying to T the construction in Lemma 4, we would obtain a DFA for $L_{m \cdot n,k}$ with less than $(m \cdot n)^k$ states, which is a contradiction. □

Complementing languages does not increase the size of k-IUFST:

Theorem 13. *Let $k, n \geq 1$ be integers and T be an n-state k-IUFST. Then n states are sufficient and necessary in the worst case for a k-IUFST to accept $\overline{L(T)}$.*

Proof. For the size upper bound, it is enough to switch accepting and rejecting states on T, thus obtaining an n-state k-IUFST for $\overline{L(T)}$. For the size lower bound, consider the language $L_{n,k}$ in the proof of Lemma 5. As pointed out, such a language has a minimal DFA with n^k states. Suppose, by contradiction, that $\overline{L_{n,k}}$ can be accepted by a k-IUFST M with $m < n$ states. By using on M the switching technique above, an m-state k-IUFST M' is obtained for $\overline{\overline{L_{n,k}}} = L_{n,k}$. Now, by Lemma 4, M' can be turned into an equivalent DFA with $m^k < n^k$ states, a contradiction. □

Finally, we focus on union of languages. We obtain a cost similar to implementing intersection.

Theorem 14. *Let $m, n, k \geq 1$ be integers, T_1 be an m-state k-IUFST and T_2 be an n-state k-IUFST. Then $m \cdot n$ states are sufficient and necessary in the worst case for a k-IUFST to accept $L(T_1) \cup L(T_2)$.*

Proof. The size upper bound follows from the upper bounds for intersection and complementation by De Morgan's laws.

For the size lower bound, assume by contradiction that less than $m \cdot n$ states are sufficient to implement union on k-IUFST. Consider the languages $L_{m,k}$

and $L_{n,k}$ with co-prime m and n, used in Theorem 12 to show the optimality of the cost $m \cdot n$ of intersection on k-IUFST. Clearly, we have $L_{m,k} \cap L_{n,k} = \overline{\overline{L_{m,k}} \cup \overline{L_{n,k}}}$. By Theorem 13, for $\overline{L_{m,k}}$ and $\overline{L_{n,k}}$, respectively, m and n states are sufficient on a k-IUFST. Yet, by our absurdum assumption, less than $m \cdot n$ states are sufficient to accept their union. In turn, again by Theorem 13, complementing the union language can be done with less than $m \cdot n$ states on a k-IUFST. This contradicts Theorem 12. □

Let us conclude this section by discussing the cost of performing reversal on k-IUFST:

Theorem 15. *Let $k, n \geq 1$ be integers and T be an n-state k-IUFST. Then 2^{n^k} states are sufficient and at least $2^{\frac{n^k}{k2^k}}$ states are necessary in the worst case for a k-IUFST to accept $L(T)^R$.*

Proof. For the state upper bound, we transform T into an equivalent n^k-state DFA A according to Lemma 4. Next, by standard construction [16], A can be turned into a 2^{n^k}-state DFA A' for $L(A)^R = L(T)^R$. Finally, by Lemma 6, the DFA A' can be simulated by a 2^{n^k}-state k-IUFST.

For the state lower bound, let the language $C_{n,k} = \bigcup_{c>0} \{a,b\}^{c \cdot n^k - 1} b \{a,b\}^*$. The minimal DFA for ${C_{n,k}}^R$ has at least 2^{n^k} states, which means, by Lemma 4, that any k-IUFST for ${C_{n,k}}^R$ must have at least $2^{n^k/k}$ states. By suitably adapting the k-IUFST for the language $L_{n,k}$ provided in the proof of Lemma 5, we can obtain a $2n$-state k-IUFST for $C_{n,k}$. Whence, the claimed result follows. □

4.3 Decidability Questions for k-IUFSTs

In this subsection, we obtain that all commonly studied decidability questions for k-IUFSTs are NL-complete So, for k-IUFSTs the questions of testing emptiness, universality, finiteness, infiniteness, inclusion, or equivalence have the same computational complexity as for DFAs [10, 11].

Theorem 16. *Let $k \geq 1$ be an integer. Then for k-IUFSTs the problems of testing emptiness, universality, finiteness, infiniteness, inclusion, and equivalence are NL-complete.*

5 Hierarchy of Non-Constant Sweep Complexities

We turn to consider $s(n)$-IUFSTs where $s(n)$ is a non-constant function. Since for any constant $k \geq 1$ any k-IUFST accepts a regular language, the first natural question is the following: "How many sweeps are necessary to cross the edge to non-regularity?" The answer is that there is no sublogarithmic sweep complexity that gives an IUFST the power to accept a non-regular language.

Proposition 17. *Let $s(n) \in o(\lg n)$. The language accepted by any $s(n)$-IUFST is regular.*

Proof. From a given $s(n)$-IUFST T, it is not hard to construct an equivalent one-tape Turing machine M with time complexity $2n \cdot s(n)$. It is shown in [7] that any one-tape Turing machine with a time complexity of order $o(n \lg n)$ accepts a regular language. Since $s(n) \in o(\lg n)$, the time complexity of M is of order $o(n \lg n)$ and, thus, $L(M) = L(T)$ is regular. □

In fact, the gap between constant and 'useful' non-constant sweep complexities ends at a logarithmic level. The witness language given by the following lemma is even unary and non-context-free.

Lemma 18. *The non-context-free unary language* $L_{\text{uexpo}} = \{ a^{2^k} \mid k \geq 0 \}$ *is accepted by a six-state* $s(n)$-IUFST *with* $s(n) \in O(\lg n)$.

Next, we turn to extend the sweep complexity hierarchy beyond the logarithm. To this end, we consider sweep complexities of order $O(\sqrt{n})$. The goal is to show that there is a language accepted by some $s(n)$-IUFST with $s(n) \in O(\sqrt{n})$ that cannot be accepted by any $s(n)$-IUFST with $s(n) \in o(\sqrt{n})$.

Lemma 19.
(1) The copy language with center markers $L_{\text{cpc}} = \{ w\$^m w \mid w \in \{a, b\}^*, m \geq 1 \}$ *is accepted by an* $s(n)$-IUFST *with* $s(n) \in O(n)$.
(2) The copy language with single marker $\{ w\$w \mid w \in \{a, b\}^* \}$ *is accepted by an* $s(n)$-IUFST *with* $s(n) \in O(n)$.
(3) Let Σ_1 *be an alphabet not containing the symbol* \$ *and* Σ_2 *be an arbitrary alphabet. Then language* $L_{\text{eq}} = \{ u\$v \mid u \in \Sigma_1^*, v \in \Sigma_2^*, \text{ and } |u| = |v| \}$ *is accepted by some* $s(n)$-IUFST *with* $s(n) \in O(n)$.

The second ingredient to show that there is a language accepted by some $s(n)$-IUFST with $s(n) \in O(\sqrt{n})$ that cannot be accepted by any $s(n)$-IUFST with $s \in o(\sqrt{n})$ is the acceptance of a language whose word length are quadratic.

Lemma 20. *The language* $L_{\text{sqr}} = \{ \#a^{m-1}\#a^{m-2} \cdots \#a^1\# \mid m \geq 0 \}$ *is accepted by an* $s(n)$-IUFST *with* $s(n) \in O(\sqrt{n})$.

Now we are prepared to provide the witness language

$$L_{cpsq} = \{ w\$w\#a^{m-1}\#a^{m-2} \cdots \#a^1\# \mid m \geq 0, w \in \{a, b\}^{m-1} \}.$$

Theorem 21. *The language* L_{cpsq} *is accepted by an* $s(n)$-IUFST *with* $s(n) \in O(\sqrt{n})$.

To show that the witness language L_{cpsq} is not accepted by any $s(n)$-IUFST with $s(n) \in o(\sqrt{n})$, we use Kolmogorov complexity and incompressibility arguments.

Theorem 22. *The language* L_{cpsq} *cannot be accepted by any* $s(n)$-IUFST *with* $s(n) \in o(\sqrt{n})$.

Finally, the sweep complexity hierarchy can be extended beyond the square root. By Lemma 19 we know already that the copy language with center markers $L_{\mathrm{cpc}} = \{ w\$^m w \mid w \in \{a,b\}^*, m \geq 1 \}$ is accepted by an $s(n)$-IUFST with $s(n) \in O(n)$. The following theorem separates this level of the sweep complexity hierarchy from the one induced by $s(n) \in O(\sqrt{n})$:

Theorem 23. *The copy language with center markers L_{cpc} cannot be accepted by any $s(n)$-IUFST with $s(n) \in o(n)$.*

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

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. 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)
3. Citrini, C., Crespi-Reghizzi, S., Mandrioli, D.: On deterministic multi-pass analysis. SIAM J. Comput. **15**(3), 668–693 (1986)
4. Friburger, N., Maurel, D.: Finite-state transducer cascades to extract named entities in texts. Theoret. Comput. Sci. **313**(1), 93–104 (2004)
5. Gao, Y., Moreira, N., Reis, R., Yu, S.: A survey on operational state complexity. J. Autom., Lang. Comb. **21**(4), 251–310 (2017)
6. Ginzburg, A.: Algebraic theory of automata. Academic Press (1968)
7. Hartmanis, J.: Computational complexity of one-tape Turing machine computations. J. ACM **15**(2), 325–339 (1968)
8. Hartmanis, J., Stearns, R.E.: Algebraic structure theory of sequential machines. Prentice-Hall (1966)
9. 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)
10. Jones, N.D.: Space-bounded reducibility among combinatorial problems. J. Comput. System Sci. **11**, 68–85 (1975)
11. Jones, N.D., Laaser, W.T.: Complete problems for deterministic polynomial time. Theoret. Comput. Sci. **3**, 105–117 (1976)
12. Malcher, A., Mereghetti, C., Palano, B.: Descriptional complexity of two-way pushdown automata with restricted head reversals. Theoret. Comput. Sci. **449**, 119–133 (2012)
13. Manca, V.: On the generative power of iterated transductions. In: Words, Semigroups, and Transductions – Festschrift in Honor of Gabriel Thierrin. pp. 315–327. World Scientific (2001)
14. Mealy, G.H.: A method for synthesizing sequential circuits. Bell System Tech. J. **34**, 1045–1079 (1955)
15. Pierce, A.: Decision Problems on Iterated Length-Preserving Transducers. Bachelor's thesis, SCS Carnegie Mellon University, Pittsburgh (2011)
16. Salomaa, A., Wood, D., Yu, S.: On the state complexity of reversals of regular languages. Theoret. Comput. Sci. **320**, 315–329 (2004)