Iterated Uniform Finite-State Transducers*

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

[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. A *deterministic iterated uniform finite-state transducer* (for short, IUFST) operates the same length-preserving transduction on several left-to-right sweeps. The first sweep occurs on the input string, while any other sweep processes the output of the previous one. 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.

Keywords: Iterated transducers; State complexity; Sweep complexity.

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 [8] to extract information from natural language texts. 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 [10, 11]. Cascades of deterministic pushdown transducers are investigated in [7] 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

* Supported by *Gruppo Nazionale per il Calcolo Scientifico (GNCS-INdAM)*. Extended version presented at the *21st Int. Conf. DCFS*, July 17–19, 2019, Košice, Slovakia, [13].

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 [6, 15], 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. It is an essential feature that the underlying finite-state transducer is not length-preserving. Several restrictions on finite-state transducers are studied in [17] with respect to the question of whether the (arbitrary) iteration of the restricted transducers is still computationally universal.

Here, we consider iterated finite-state transducers as language acceptors. Yet, to have a simple device, the underlying transducer is considered to be a Mealy machine [16], i.e., a deterministic length-preserving device where each input symbol is translated according to its transition function into an output symbol. More precisely, 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 [12] and many results on the operational state complexity are surveyed in [9]. Next, we consider the case when the number of sweeps is *not bounded by a fixed finite number*. The resulting IUFSTs can thus 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. We prove 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, we show a hierarchy depending on the number of sweeps.

2 Definitions and Preliminaries

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. 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 $T = \langle Q, \Sigma, \Delta, q_0, \lhd, \delta, F_+, F_- \rangle$, 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., no transitions $\delta(p, x) = (q, \triangleleft)$ with $x \neq \triangleleft$ exist. 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.

3 Iterated Transductions vs. DFAs

Iterated transduction may lead to dramatically decrease the number of states for regular acceptance with respect to DFAs. To see this, for any $k \geq 1$, let us consider the language $E_k = \{a, b\}^* b \{a, b\}^{k-1}$.

Theorem 1. *Any* DFA *for the language E_k needs at least 2^k states. On the other hand, there exists a k-*IUFST *T accepting E_k with 3 states only.*

However, some regular languages turn out to be so size-consuming that even iterated transduction cannot help. In fact, let p be any prime number. We define the language $L_p = \{\, a^{m \cdot p} \mid m \geq 0 \,\}$.

Theorem 2. *Let $k \geq 1$. Then p states are necessary and sufficient for a k-*IUFST *to accept L_p.*

4 Descriptional Complexity

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 3. *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 3 turns out to be optimal. To this aim, for any $n, k > 0$, let us define the unary language $L_{n,k} = \{ a^{c \cdot n^k} \mid c \geq 0 \}$.

Lemma 4. *The language $L_{n,k}$ is accepted by an n-state k-IUFST, while any equivalent* DFA *needs at least n^k states.*

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

Lemma 5. *For $n \geq 3$, every n-state* DFA *can be converted into an equivalent n-state k-IUFST, with $k \geq 1$.*

Lemma 5 holds 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 2, the construction provided by Lemma 5 is optimal.

4.1 States versus Sweeps

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., [5, 14]). Theorem 2 shows there are languages for which additional sweeps do not help to decrease the number of states at all. On the other hand, the next theorem provides a gradual reduction of the number of sweeps and the necessary states for the language $E_k = \{a, b\}^* b \{a, b\}^{k-1}$ of Theorem 1.

Theorem 6. *Let $\ell \geq 1$, $k = 2^\ell$ be integers. Then the useful sweep range for the 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.*

5 The State Cost of Language Operations on *k*-IUFSTs

As naturally done for many models of computation accepting regular languages (see, e.g., [1–4]), 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 which store the accept or reject outcome obtained in the middle of the input, while reaching the endmarker. In what follows, $\overline{L}$ and L^R denote, respectively, the complement and reversal of a language L.

Theorem 7. *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 in Theorem 7 is optimal, as proved in the following

Theorem 8. *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.*

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

Theorem 9. *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 $\overline{L(T)}$.*

The above two theorems enable us to establish the optimal cost of implementing intersection:

Theorem 10. *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 $L(T_1) \cup L(T_2)$.*

Concerning the cost of performing reversal on k-IUFST, we get

Theorem 11. *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 $L(T)^R$.*

6 Hierarchy of Non-Constant Sweep Complexities

Since any constant sweep bounded 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 contained in the following theorem (by lg, we denote the logarithm to base 2):

Proposition 12. *Let $s(n) \in o(\lg n)$. The language accepted by any $s(n)$-IUFST 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 13. *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 extend the sweep complexity hierarchy beyond the logarithm. To this end, we consider sweep complexities of order $O(\sqrt{n})$, and define the language $L_{cpsq} = \{ w\$w\#a^{m-1}\#a^{m-2} \cdots \#a^1\# \mid m \geq 0, w \in \{a,b\}^{m-1} \}$ for which we get

Theorem 14. *The language L_{cpsq} is accepted by an $s(n)$-IUFST with $s(n) \in O(\sqrt{n})$, while it cannot be accepted by any $s(n)$-IUFST with $s(n) \in o(\sqrt{n})$.*

Finally, the sweep complexity hierarchy can be further extended beyond the square root up to sweep complexities of order $O(n)$. In fact, we define the language $L_{\text{cpc}} = \{ w\$^m w \mid w \in \{a,b\}^*, m \geq 1 \}$ for which we prove

Theorem 15. *The language L_{cpc} is accepted by an $s(n)$-IUFST with $s(n) \in O(n)$, while it cannot be accepted by any $s(n)$-IUFST with $s(n) \in o(n)$.*

Acknowledgements. The authors wish to thank the anonymous referees.

References

1. Bednárová, Z., Geffert, V., Mereghetti, C., Palano, B.: The size-cost of Boolean operations on constant height deterministic pushdown automata. Th. Comp. 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.. In: Bulatov, A.A., Shur, A.M. (eds.), *Proc. 8th International Computer Science Symposium in Russia (CSR 2013)*. LNCS **7913**, 100–111, Springer, 2013.
3. Bednárová, Z., Geffert, V., Mereghetti, C., Palano, B.: Removing nondeterminism in constant height pushdown automata. Information and Computation **237**, 257–267 (2014).
4. Bertoni, A., Mereghetti, C., Palano. B.: Trace monoids with idempotent generators and measure only quantum automata. Natural Computing **9**, 383–395 (2010).
5. Bianchi, M.P., Mereghetti, C., Palano, B. Complexity of promise problems on classical and quantum automata. In: Calude, C.S., Freivalds, R., Iwama, K. (eds.), *Computing with New Resources, Essays Dedicated to Jozef Gruska on the Occasion of his 80th Birthday*. LNCS **8808**, 161–175, Springer, 2014.
6. Bordihn, H., Fernau, H., Holzer, M., Manca, V., Martín-Vide, C.: Iterated sequential transducers as language generating devices. Th. Comp. Sci. **369**(1-3), 67–81 (2006)
7. Citrini, C., Crespi-Reghizzi, S., Mandrioli, D.: On deterministic multi-pass analysis. SIAM J. Comput. **15**(3), 668–693 (1986)
8. Friburger, N., Maurel, D.: Finite-state transducer cascades to extract named entities in texts. Theoret. Comput. Sci. **313**(1), 93–104 (2004)
9. Gao, Y., Moreira, N., Reis, R., Yu, S.: A survey on operational state complexity. J. Autom., Lang. Comb. **21**(4), 251–310 (2017)
10. Ginzburg, A.: Algebraic theory of automata. Academic Press (1968)
11. Hartmanis, J., Stearns, R.E.: Algebraic structure theory of sequential machines. Prentice-Hall (1966)
12. 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)
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.), *Proc. 21st International Conference on Descriptional Complexity of Formal Systems (DCFS 2019)*. LNCS **11612**, 223–234, Springer, 2019.
14. Malcher, A., Mereghetti, C., Palano, B.: Descriptional complexity of two-way pushdown automata with restricted head reversals. Th. Comp. Sci. **449**, 119–133 (2012)
15. 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)
16. Mealy, G.H.: A method for synthesizing sequential circuits. Bell System Tech. J. **34**, 1045–1079 (1955)
17. Pierce, A.: Decision Problems on Iterated Length-Preserving Transducers. Bachelor's thesis, SCS Carnegie Mellon University, Pittsburgh (2011)