Multitape automata and finite state transducers with lexicographic weights

Aleksander Mendoza-Drosik

***Abstract*—Finite state transducers, multitape automata and weighted probabilistic automata have a lot in common. By studying their universal foundations, one can discover some new insights into all of them. The main result presented here is the introduction of lexicographic finite state transducers, that could be seen as intermediate model between multitape automata and weighted transducers. Their most significant advantage is being equivalent, but often exponentially smaller than non-weighted transducers.**

***Index Terms*—Mealy machines, transducers, sequential machines, computability, complexity**

I. Introduction

THE goal of this paper is present a new model of weighted finite state transducers and show its special properties. Their definitions, inspired by the ones found in other papers [1] [2], could be stated as follows.

Definition 1. *Non-weighted finite state transducers (FST) are defined here as a tuple $(Q, q_0, \Sigma, \Gamma, \delta, F)$ with finite set of states Q, initial state q_0, input alphabet Σ, output alphabet Γ, transition function $\delta \subset Q \times \Sigma \times \Gamma^* \times Q$ and set of accepting states $F \subset Q$.*

Definition 2. *Weighted finite state transducers (WFST) are defined similarly to FST as $(Q, q_0, \Sigma, \Gamma, W, \delta, F)$ where W is the set of weights and δ is the transition function is of the form $\delta \subset Q \times \Sigma \times \Gamma^* \times W \times Q$. Weights should form a complete semiring $(W, +, \cdot)$*

Definition 3. *Lexicographic weighted finite state transducers (LFST) are a special case of WFST. Let Ω be some finite set of weight digits with total order relation $\leq$. Then $W = \Omega*$ produces a free semiring $(W, +, \cdot)$ with reverse lexicographic order $\leq_{lex}$ (the strings are reversed before performing lexicographic comparison, or in other words, they are compared from right to left), concatenation as the multiplicative $\cdot$ and max function as additive $+$. Empty set $\emptyset$ is designated as multiplicative zero. Lastly, there is a constraint on δ, which requires it to be of the form $\delta : Q \times \Sigma \times \Gamma^* \times \Omega \times Q$ (only single-character weights are allowed on transitions).*

In the rest of this paper, we shall reinvent those definition starting from more universal foundations, that will give us new insights.

II. Universal foundations

A. Algebraic foundations

Every transducer can, in principle, be treated like a multitape finite state automaton, simultaneously reading input and output tape. This is made more precise if we consider some monoid A, set of states Q and set of edges $\delta \subset Q \times A \times Q$. Let's define **path** to be a finite sequence of edges $(q_{k_1}, x_1, q_{k_2}), (q_{k_2}, x_2, q_{k_3}), ...(q_{k_m}, x_m, q_{k_{m+1}})$ where $q_{k_i}, q_{k_{i+1}} \in Q$, $x_i \in A$ and $(q_{k_i}, x_i, q_{k_{i+1}}) \in \delta$ for every index i. **Signature** [3] of a path is defined as $x_1 x_2 ... x_m \in A$. **Automaton** [4] is defined as tuple (Q, I, A, δ, F) where Q and δ are finite and A is finitely generated. Path is **accepting** if it starts in $q_{k_1} \in I$ and ends in $q_{k_{m+1}} \in F$. An automaton accepts element $x \in A$ if it is a signature of some accepting path. Given $A = B \times C$ (notice that associativity holds for $\times$), we call δ **atomic up to** B if $b \neq 1_B, b' \neq 1_B$ implies $(q, (bb', c), q') \notin \delta$ (where 1_B stands for neutral element of B). Automaton is **atomic** if its δ is atomic up to entire A. Automaton with $A = B \times C$ is **deterministic up to** B if it is atomic up to B and $|I| = 1$ and $\delta \subset Q \times (B \backslash \{1_B\}) \to C \times Q$ (when the function is partial, then automaton is called **partial**). Automaton is **deterministic** when it is deterministic up to entire A, that is, $|I| = 1$ and has atomic $\delta \subset Q \times (A \backslash \{1_A\}) \to Q$. Given $A = B \times C$ we call δ **ϵ-free up to** B if it is of the form $\delta \subset Q \times (B \backslash 1_B) \times C \times Q$. Automaton is **$\epsilon$-free** if its δ is ϵ-free up to entire A. Subset $L \subset A$ is **rational** if and only if there exists some automaton accepting all $x \in L$ and rejecting $x \notin L$. Moreover, if A is a direct product of several monoids, then we call L a **rational relation**. If rational relation is a function, then automaton recognizing it is called **functional**. If $L \subset B \times C$ then $(L)_B = \{b \in B : \exists_{c \in C}(b, c) \in L\}$ is called the B **projection** of L. By $L(b)$ we denote output for $b \in B$ defined as $(L \cap (\{b\} \times C))_C$. If M is some automaton recognizing L then $M(b)$ denotes $L(b)$.

To give some intuition, consider that in case of nondeterministic finite state automata (FSA), A would often be defined as Σ^* and δ would be of the form $\delta \subset Q \times \Sigma \times Q$ (which is also $Q \times \Sigma \times Q \subset Q \times A \times Q$). Deterministic finite state automata (FSA) would have $\delta \subset Q \times \Sigma \to Q$. Multitape automata, could be modelled using $A = \Sigma^* \times \Gamma^*$ and $\delta \subset Q \times \Sigma \times \Gamma \times Q$. Epsilon transitions can be added with $\delta \subset Q \times (\Sigma \cup \{\epsilon\}) \times Q$. Nothing stops us from allowing entire strings on transitions $\delta \subset Q \times \Sigma^* \times Q$. Therefore one might notice that FST are multitape automata reading 2 tapes, while WFST operate on 3 tapes. The common notation $(Q, q_0, \Sigma, \Gamma, \delta, F)$ in fact stands for automaton $(Q, \{q_0\}, \Sigma^* \times \Gamma^*, \delta, F)$.

B. Free monoids and metric on A

When Σ^* and Γ^* are free monoids, then the direct product $\Sigma^* \times \Gamma^*$ is a monoid as well, but not necessarily free. In cases when A is free, it's easy to define metric (for every $x \in A$ the $|x|$ equals to the length of word/string). When A is a direct product of several free monoids, then one can study the relationship between their lengths. The most notable property is that if $A = \Sigma^* \times \Gamma^*$ and $\delta \subset Q \times \Sigma \times \Gamma \times Q$, then for every accepted $(\sigma, \gamma) \in A$ the lengths are equal $|\sigma| = |\gamma|$. This can be further generalized to $A = \Sigma_1^* \times \Sigma_2^* \times ... \Sigma_n^*$. If δ is of the form such that at least one Σ_i^* is required to be of length exactly 1 on each transition (that is $\delta \subset Q\times\Sigma_1^*\times...\times\Sigma_i\times...\Sigma_n^*$) then we can induce metric on the accepted subset of A, that is, $|(\sigma_1, ..., \sigma_i, ...\sigma_n)| = |\sigma_i|$ (which also coincides with length of accepting path). Perhaps the most notable property of such metric induced on Σ_i^* is that it allows for generalising **pumping lemma for multitape automata** (and hence, transducers as well).

If A contains some elements with inverses then the automaton cannot be atomic. In particular suppose $aa^{-1} = 1_A$ and $(q, a', q') \in \delta$ then $a' = a'1_A = a'(aa^{-1}) = (a'a)a^{-1}$ but $a'a \neq 1_A$ and $a^{-1} \neq 1_A$, hence atomicity is violated.

A may or may not contain commuting elements. If $A = B \times C$ then all the elements of the form $(1_B, c)$ and $(b, 1_C)$ commute. This phenomenon characterizes nonsequential machines which are used to encode conurrent systems (see theory of traces [5]). For this reason, all multitatpe automata with ϵ-transitions are in a sense "concurrent" machines.

C. Superpositions

Combination is defined to be a subset of Q. Given $K \subset Q$ and $x \in A$, define $\hat{\delta}$ to be transitive closure of δ, such that $\hat{\delta}(K, x)$ is the set of all states q, for which exists a path starting in K and ending in q with signature x. In cases when $A = B\times C$ we can extend the concept of combination to include C, that is, define **superposition** as a subset of $Q \times C$. Given some $S \subset Q \times C$ we define $\hat{\delta}_C$ such that $(q', yy') \in \hat{\delta}_C(S, x)$ whenever there exists $(q, y) \in S$ and path starting in q and ending in q' with signature (x, y'). If automaton is atomic up to B and $\delta \subset Q\times B\backslash\{1_B\}\times C\times Q$ then we can define image of combination

$$\delta_C(K, b) = \{q' \in Q : \exists_{q\in K}(q, (b, c'), q') \in \delta\}$$

and image of superposition

$$\delta_C(S, b) = \{(q', cc') \in Q \times C : \exists_{(q,c)\in S}(q, (b, c'), q') \in \delta\}$$

and then $\hat{\delta}_C$ becomes

$$\hat{\delta}_C(S, \epsilon) = S$$
$$\hat{\delta}_C(S, bx) = \hat{\delta}_C(\delta_C(S, b), x)$$

In all equation above b refers to element of B's smallest generator and b. Note that if M is some automaton atomic up to B then $M(x) = (\hat{\delta}_C(I \times \{1_C\}, x))_C$ for all $x \in B$.

D. Stochastic languages and weighted automata

Let $M = (Q, I, B \times C, \delta, F)$ be some automaton that isn't deterministic. We define $\delta' \subset Q \times B \times W \to C \times Q$ to be a **disambiguation up to** B for M if $(Q, I, B \times W \times C, \delta', F)$ is deterministic and $(q, (b, c), q') \in \delta \iff \exists_{w\in W}(q, (b, w, c), q') \in \delta'$. Very often W indeed is the set of weights.

Consider an automaton that reads input from randomly filled tape. The sequence of symbols in that tape might be interpreted as a sequence of random variables. This allows us to consider the probability of accepting random strings. The set of weights W can in principle represent such a random tape. For instance, automaton that returns $\delta(q_1, x) = q_2$ with 20% probability and $\delta(q_1, x) = q_3$ with 80%, can be disambiguated by adding two weights $\delta(q_1, x, w_2) = q_2$ and $\delta(q_1, x, w_3) = q_3$ such that probability of getting w_2 and w_3 is 20% and 80% respectively.

Probabilistic automaton is any automaton whose A is a measurable space with total measure equal 1. One should notice that any probabilistic transducer $(Q, I, \Sigma^* \times \Gamma^* \times W, \delta, F)$ is in fact a probabilistic automaton and probability of accepting (x, y, w) is often made conditional on $x \in \Sigma^*$. Very often one also tries to find y that maximizes w, given x.

If $A = B \times C$ and C is a complete semiring, then quotient $L\backslash C \subset B \to C$ of language $L \subset A$ is defined as $(x, \sum_{(x,c)\in L} c) \in L\backslash C$ for all $x \in (L)_B$. It's a common practice to use weights as C and then search for such x that maximizes $(L\backslash C)(x)$. In probabilistic automata, the semiring operations on C would correspond to multiplying and adding probabilities. In particular it's possible to approximate real-valued probability by splitting the interval $[0, 1]$ into equal c sized intervals, each of them becoming a symbol of alphabet C, then semiring addition works as union of intervals and concatenation of symbols c_1c_2 represents interval c_2 relative to c_1 (for instance, if $c_1 = (0, 0.5)$ and $c_2 = (0.5, 1)$ then $c_1c_2 = (0.25, 0.5)$). As the constant c approaches 0, the precision of symbols in C increases. Notice that every random sequence of real numbers falls into some string (or union of strings) in C with probability equal to the measure of that string (or union).

III. Properties

Theorem 1 (Deterministic superposition)**.** *If automaton over $A = B \times C$ is deterministic up to B then $|\hat{\delta}_C(S, x)| \leq 1$ for all $x \in B$ and all initial superpositions $|S| = 1$.*

Proof: Determinism states that $\delta \subset Q \times B \to C \times Q$, so there is at most one transition that can be taken each step. Therefore the number of elements in superposition cannot increase. ∎

As consequence one may easily show that in deterministic automata initial state and signature uniquely determine path. This leads us to introduce the following theorem.

Theorem 2 (Rule of matching prefixes)**.** *Let M be some automaton over $A = B \times C$ deterministic up to B. For all strings $x, x' \in B$ if $M(xx') = y' \neq \emptyset$ and $M(x) = y \neq \emptyset$, then y is a prefix of y'.*

Proof: It follows directly from uniqueness of path that corresponds to signature xx'. ∎

Theorem 3 (Infinite superposition)**.** *Let M be an automaton over $A = B \times C$ atomic up to B. $|M(x)| = \infty$ for some $x \in B$ only if M contains ϵ-cycle $(q_{k_1}, (1_B, y_1), q_{k_2})$,...,$(q_{k_m}, (1_B, y_m), q_{k_1})$ where $y_i \in C$ and $(1_B, y_1...y_m) \neq 1_A$.*

Proof: Every time a non-ϵ-transition from $\delta \subset Q \times (B\backslash\{1_B\}) \times C \times Q$ is taken, it increases the length of x in the corresponding signature $(x, y) \in A$. Only ϵ-transitions of the form $\delta \subset Q \times \{1_B\} \times C \times Q$ do not increase length of x. There are only finitely many elements y of specific finite length $|y|$. Therefore in order to obtain infinite subset of C it must contain strings of unbound length. The only way to have unbound y, while keeping x bound is by taking infinitely many $Q \times \{1_B\} \times C \times Q$ transitions. If there is no

ϵ-cycle then we can take only finite number of ϵ transitions, before being forced to take non-ϵ-transition. Therefore there must be an ϵ-cycle. ■

Recall that in the definition of quotient $L\backslash C$ we required C to be a complete semiring. Thanks to theorem 3, it's possible to relax this assumption and use any semiring, under the condition that L is recognized by ϵ-free automaton (or at least one that doesn't have ϵ-cycles). This way, infinite sum will never have the chance to arise.

Theorem 4 (Functional superposition)**.** *Let M be a functional automaton over $A = B \times C$, atomic up to B and whose recognized languages is of the form $L \subset B \to C$. Then there exists an equivalent automaton such that $\hat{\delta}_C(S, x) \subset Q \to C$ for all $S \subset Q \to C$ and $x \in B$.*

Proof: Suppose to the contrary that there is x and q such that $|\hat{\delta}_C(S, x)(q)| > 1$. Then there are two possibilities: either there is a path that starts in q and ends in F or there is not. If the first case is true, then M is not functional, because we might follow that path and accept with multiple C outputs. If the second case applies, then the state q is redundant and we are free to delete it. ■

Let $L \subset B \to C$ be some (partial) function and let $b_1, b_2 \in B$. Element $(b, c) \in A$ is a **distinguishing extension up to** B of b_1 and b_2 if exactly one of $(b_0 b, L(b_0)c)$ or $(b_1 b, L(b_1)c)$ belongs to L. Define an equivalence relation $=_L$ on A such that $a_0 =_L a_1$ if and only if there is no distinguishing extension up to B for a_0 and a_1 (the exact B should always be clear from context).

Theorem 5 (Generalized Myhill-Nerode theorem)**.** *Let $L \subset B \to C$. Assume that $L(bb') = c' \neq \emptyset$ and $M(b) = c \neq \emptyset$ implies c is a prefix of c' (rule of matching prefixes holds). L can be recognized by automaton deterministic up to B if and only if there are only finitely many equivalence classes induced by $=_L$.*

Proof: ($\Longleftarrow$) First assume there are finitely many equivalence classes. Let G be the smallest generator of B. Then build an automaton by treating every class as a state Q. Put a transition from class q to q' over $b' \in G\backslash 1_B$ whenever there exists $(b, c) \in q$ and $(bb', c') \in q'$. Rule of matching prefixes guarantees existence of suffix s such that $cs = c'$. This suffix shall be used as transition output. By taking b' only from $G\backslash 1_B$ we ensure that automaton is atomic up to B and has no ϵ-transitions. The class that contains 1_A is designated as the unique initial state (hence the automaton is deterministic up to B). All the classes intersecting L are accepting states (note that if $q \cap L \neq \emptyset$ then $q \subset L$).

($\Longrightarrow$) Conversely, if there is an automaton deterministic up to B and recognizing L, then there could be found a homomorphism from states of machine to equivalence classes. ■

Note that this theorem no longer works when automaton is not deterministic at least up to B.

Recall that LFST are defined as (Q, I, A, δ, F) where $A = \Sigma^* \times \Gamma^* \times W^*$. Transition function $\delta : Q \times \Sigma \times \Gamma^* \times W \times Q$ gives us the guarantee that for each $(x, y, w) \in L$, the lengths are equal $|x| = |w|$, therefore the reversed lexicographic ordering $\leq_{lex}$ never compares strings of different lengths when performing $L\backslash W$. Because every set of equally-sized strings is finite and has maximal element, the semiring $(W, \cdot, max)$ does not need to be complete.

Moreover, if at any point in computation LFST reaches a superposition with multiple $\Gamma^* \times W$ assigned to the same state Q, then we can discard almost all the elements and keep only those with maximal w.

Theorem 6 (LFST equivalent to FST)**.** *Let L be some regular relation recognized by lexicographic finite state transducer. Then there exists FST recognizing L' such that $(x, y) \in L'$ if and only if y maximizes $w = (L\backslash W)(x, y)$ for given x, as in $\underset{y}{\operatorname{argmax}}(L\backslash W)(x, y)$. (There may be multiple equally maximal y)*

Proof: Conversion can be carried out using (extended) powerset construction. Suppose Q are the states of LFST, then $Q' = 2^Q \times Q$ are states of equivalent FST. For any state $(K, q) \in Q'$, the K represents particular combination of Q and q is some representative in K (we can discard all those states of Q' where $q \notin K$). We need the representative because it will specify, which Q state's output to store in corresponding state of Q'. More formally, if S is superposition in LFST and S' is superposition in FST, then $(q, y, w) \in S$ can be converted to $((K, q), y) \in S'$ such that $K = (S)_Q$. We put transition from (K_1, q_1) to $(K_2, q_2) \in Q'$ over $x \in \Sigma$ with output $y \in \Gamma^*$ whenever $\delta(K_1, x) = K_2$ and $(q_1, x, y, w, q_2) \in \delta$. The only exception is when there are multiple conflicting states in Q' that transition to (K_1, q_1) over the same x. In such cases we pick only those with highest weight w and discard the rest. Finally a state of $(K, q) \in Q'$ is accepting iff $q \in K$ is accepting. It's worth noting that if LFST is functional, then FST must be functional too. ■

Theorem 7. *Deciding whether atomic nondeterministic automaton is functional is coNP-hard.*

Proof: Given formula $\phi = (X_{h_1} \vee X_{i_1} \vee X_{j_1}) \wedge ... \wedge (X_{h_m} \vee X_{i_m} \vee X_{j_m})$ in 3-conjunctive normal form over variables $x_1, x_2, ...x_n$ (every uppercase X is a placeholder for some variable) we can construct an automaton M over alphabet $A = \Sigma^* \times \Gamma^* = \{0, 1\} \times \{0, 1\}$ such that $\exists_{x \in \Sigma^*} |M(x)| > 1$ iff ϕ is not tautology [6]. Strings $x \in \Sigma^*$ encode partial assignments to variables $x_1, ...x_n$. In fig.1 every state q_{kl} stands for "after reading input $x_1 x_2 x_3 ... x_k \in \{0, 1\}^k$, partially assigned clause $(X_{h_l} \vee X_{i_l} \vee X_{j_l})$ still has potential to be true". We shall put edge with label $1 : \epsilon$ from state q_{kl} to $q_{(k+1)l}$ iff $\neg x_{k+1}$ is in the clause. Analogically we put edge $0 : \epsilon$ iff x_{k+1} is in the clause. The final vertices $q_{n1}, ..q_{nm}$ tell us which clauses are true for given assignment. Next to them we also designate special vertices $\neg q_{n1}, ..., \neg q_{nm}$, that collect all the failed false clauses. In the end when automaton reads all n variables, there is an epsilon edge coming from every state $q_{n1}, ..q_{nm}$ that prints $\epsilon : 1$ (meaning clause is true). Similarly from vertices $\neg q_{n1}, ..., \neg q_{nm}$ automaton prints $\epsilon : 0$ if clause is false. If ϕ is a tautology, then for all assignments we will always end up in one of the $q_{n1}, ..q_{nm}$ and never in $\neg q_{n1}, ..., \neg q_{nm}$. Therefore the output associated with input sequence $x_1 x_2 x_3 ... x_n$ will be unambiguously 1. If at least one clause fails, then ϕ is not tautology and M returns both 0 and 1. There is also a special case, when ϕ is contradiction (false for all assignments). Thanks to states $q_0, ...q_n$ we should also get ambiguous

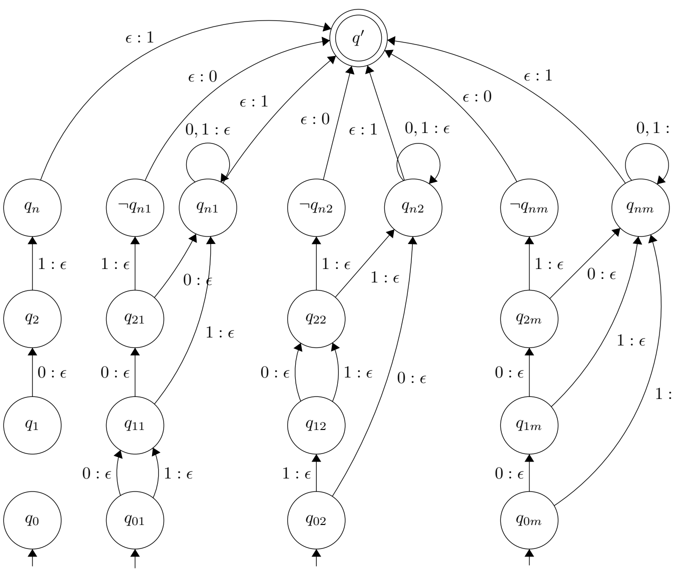

Fig. 1. Construction that uses two-tape nondeterministic automaton to represent CNF formula. Each column evaluates one clause and ends in state q_{nj} if clause is true or $\neg q_{nj}$ when it's false. State q' will accept producing output 0 if any clause was false and 1 if any clause was true. Leftmost column is a "bias" that prints 1 if all clauses are false. This graph uses ϵ outputs, but by treating ϵ as an ordinary letter of alphabet, it could be converted to ϵ-free automaton easily.

output. q_n is a special state, that can be reached only in one case - when all clauses are simultaneously false. We put edge $1 : \epsilon$ transitioning from q_k to q_{k+1} iff x_k is not present in any clause. Similarly $0 : \epsilon$ iff $\neg x_k$ is not present in any clause. Such construction guarantees us that ϕ is tautology iff M is functional. The construction itself uses only polynomially many vertices and edges. ■

Theorem 8. *There exists a family of LFST such that their corresponding minimal FST are exponentially larger.*

Proof: We define family of LFST in such a way that for every $i \geq 3$ there is LFST on i states. The number of states of minimal equivalent FST is $O(2^i)$. Figure 2 presents a way to build such automata. State q_0 is initial. Using strings from $\{0,1\}^*$ one can obtain any combination of states q_1 to q_n. Let's associate each combination with a string $z \in \{0,1\}^n$ (for instance $z = 011$ would be a combination $\{q_2, q_3\}$). That gives 2^n possible strings. State q_{n+1} is accepting and all the states $q_1 ... q_n$ are connected to it. Essentially the relation described by this automaton is a subset of $\{0,1\}^+2 \times \{y_1, ..., y_n\}$. Without loss of generality suppose that weights $w_1 ... w_n$ are in ascending order. Then the automaton maps every z determined by $x \in \{0,1\}^+$ to some (x, y_k) such that k indicates the least significant bit in z. For instance suppose $n = 4$ and $x = 0011 \sim z = 1101$ then $(x2, y_4)$,$(x02, y_3)$,$(x002, y_4)$,$(x0002, y_4)$,$(x00002, \emptyset)$. Notice that one can reconstruct z from such sequence of y's.

Every FST is just an FSA reading single tape over alphabet of tuples and in this case the alphabets on each side of tuple are different, so we can "squash" them into single alphabet $D = \{0, 1, 2, y_1, ..., y_n\}$. This way the language becomes subset of $\{0,1\}^+2\{y_1, ..., y_n\}$ and every pair (x, y_k) becomes a string xy_k. Using Myhill-Nerode theorem, it can be seen that no two x strings that map to two different z

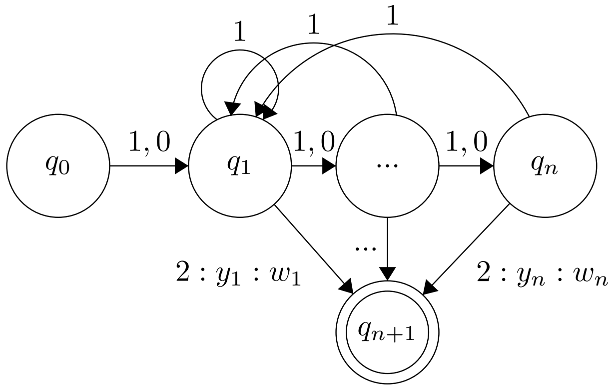

Fig. 2. In this sketch of LFST, the all the weights w_1,...,w_n are distinct. In many other transitions, weights were omitted, as they don't play any role and could be arbitrary.

are equivalent, hence the smallest deterministic FSA must have at least 2^n states. Call this minimal FSA $\mathcal{A}$. The most difficult problem is to show that no nondeterministic FSA polynomially smaller than $\mathcal{A}$ can be build. The rigorous proof can be obtained with help of Theorem 7 and Lemma 7 presented in Kameda Weiner [7]. We can build RAM using $D(\mathcal{A})$ and $D(\overleftarrow{\mathcal{A}})$ (all defined in [7]). Notice that every combination ("configuration" in [7]) of states $q_1 ... q_n$ has different succeeding event[7] and none of them is subset of the other (because there exists bijection between z and sequence of y's produced by $(x2, y_{k_0})$,$(x02, y_{k_1})$,...). Hence the minimal legitimate grid cannot be extended for any of them and the nondeterministic FSA cannot be much smaller than 2^n. ■

There is also a more intuitive (and perhaps inspiring) explanation for theorem 8. In the usual setting, nondeterminism works in a sense like logical disjunction of branching computation paths. The automaton accepts as a whole if either branch accepts individually. In LFST the weights allow us to encode logical conjunctions and different nondeterministic branches can "communicate" with each other. The only way to simulate logical "AND" gate of FSA states is by performing their cross product. For instance to build AND gate of q_1 and q_2 we need one state for each of $q_1 \wedge q_2$, $q_1 \wedge \neg q_2$, $\neg q_1 \wedge q_2$, $\neg q_1 \wedge \neg q_2$. When we want "AND" gate of n states, we need to perform cross product n times, yielding 2^n states.

One can easily notice that every FST can be treated like a LFST with all weights equal, therefore the opposite of theorem 8 doesn't hold (there is no family of FST such that LFST would be larger).

IV. Conclusions

Multitape automata form universal foundations for defining all other, more specialized types of machines. Interestingly, automata with writeable tapes could be defined in similar spirit as (Q, I, A, δ, F) with $\delta \subset Q \times A \times Q \times A$. At every transition (q, a, q', a') the symbol a is read from tape and a' is written. 1_A stands for "pass" operation and all read-only automata become the a special case of read-write automata, whose transition function is of the form $\delta \subset Q \times A \times Q \times \{1_A\}$. Pushdown automata are 2-tape automata $(Q, I, \Sigma^* \times \Gamma^*, \delta, F)$ with one tape being read-only $\delta \subset Q \times \Sigma \times \Gamma \times Q \times \{1_\Sigma\} \times \Gamma$. Transducers can be viewed as special case of pushdown automata [8] that never read the second tape $\delta \subset Q \times \Sigma \times \{1_\Gamma\} \times Q \times \{1_\Sigma\} \times \Gamma$. There is also unusual connection between writeable tapes functioning as stack and invertible elements of A. For instance, pushing onto stack is could be seen as reading some letter a and popping off of the stack is like reading a^{-1}. This way we obtain alternative definition for pushdown automata as 2-tape automata $(Q, I, B \times C, \delta, F)$ where C is a group and B a monoid with no invertible elements (transition function stays as $\delta \subset Q \times A \times Q$). Further generalisations like this deserve deeper research in the future.

Weights can be viewed as special tapes reading sequences of random variables. This more general approach give us necessary foundations for defining lexicographic transducers, which in many ways seem to be more natural than probabilistic automata (concept of lexicographic ordering is much closer to formal languages, than trying to encode real numbers in tape's alphabet). There is yet a lot to discover. Perhaps the most interesting aspect of LFST is their relation to logical AND/OR gates. Theorem 8 gives certain clues, that perhaps LFST could be inferred more efficiently, or at least generalise better. Solomonoff's theory of inductive inference [9][10][11] says that simpler and shorter automata, should be the preferred solution to inference problems. LFST are not only smaller than their non-weighted counterparts but also are better suited to express logical dependencies between states. This line of research deserves further investigation in the future.

Acknowledgment

The author would like to thank Piotr Radwan for all the great inspiration.

References

[1] F. P. Mehryar Mohri and M. Riley, "Weighted finite-state transducers in speech recognition," *AT&T Labs Research*, 2008.
[2] M. Mohri, *Weighted Finite-State Transducer Algorithms. An Overview*. Springer, 2004.
[3] J.-E. Pin, *Mathematical Foundations of Automata Theory*. American Mathematical Society, 2017.
[4] S. Eilenberg, *Automata, Languages and Machines Vol. A*. Academic Press, 1974.
[5] V. Diekert, *The Book Of Traces*. Wspc, 1995.
[6] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*. Princeton University, 2007.
[7] P. W. Tsunehiko Kameda, "On the state minimization of nondeterministic finite automata," *IEEE Transactions on Computers*, 1970.
[8] J. Berstel, *Transductions and Context-Free Languages*. Teubner Studienbucher: Stuttgart, 1979.
[9] R.J.Solomonoff, "A formal theory of inductive inference. part i," *Information and Control*, 1964.
[10] N. V. A. Shen, V. A. Uspensky, *Kolmogorov Complexity and Algorithmic Randomness*. IRIF, 2019.
[11] R.J.Solomonoff, "A formal theory of inductive inference. part ii," *Information and Control*, 1964.