# On Reversible Transducers

Luc Dartois, Paulin Fournier, Ismaël Jecker, Nathan Lhote

# On Reversible Transducers*†

**Luc Dartois[1], Paulin Fournier[2], Ismaël Jecker[3], and Nathan Lhote[4]**

1 **Université Libre de Bruxelles, Brussels, Belgium**
ldartois@ulb.ac.be
2 **Université de Bordeaux, LaBRI, Bordeaux, France**
paulin.fournier@labri.fr
3 **Université Libre de Bruxelles, Brussels, Belgium**
ijecker@ulb.ac.be
4 **Université Libre de Bruxelles, Brussels, Belgium; and Université de Bordeaux, LaBRI, Bordeaux, France**
nlhote@labri.fr

## Abstract

Deterministic two-way transducers define the robust class of regular functions which is, among other good properties, closed under composition. However, the best known algorithms for composing two-way transducers cause a double exponential blow-up in the size of the inputs. In this paper, we introduce a class of transducers for which the composition has polynomial complexity. It is the class of reversible transducers, for which the computation steps can be reversed deterministically. While in the one-way setting this class is not very expressive, we prove that any two-way transducer can be made reversible through a single exponential blow-up. As a consequence, we prove that the composition of two-way transducers can be done with a single exponential blow-up in the number of states.

A uniformization of a relation is a function with the same domain and which is included in the original relation. Our main result actually states that we can uniformize any non-deterministic two-way transducer by a reversible transducer with a single exponential blow-up, improving the known result by de Souza which has a quadruple exponential complexity. As a side result, our construction also gives a quadratic transformation from copyless streaming string transducers to two-way transducers, improving the exponential previous bound.

**1998 ACM Subject Classification** F.4.3 Formal Languages

**Keywords and phrases** Transducers, reversibility, two-way, uniformization

**Digital Object Identifier** 10.4230/LIPIcs.ICALP.2017.113

## 1 Introduction

**Automata and transducers.** Automata theory is a prominent domain of theoretical computer science, initiated in the 60s [4] and still very active nowadays. Many extensions of finite automata have been studied such as automata over more complex structures (infinite words, trees, *etc*) or transducers which can be seen as automata with an additional write-only output tape and which will be the focus of our study in the remainder of this article.

Transducers have been studied for almost as long as automata [1] and important results have been obtained, however the theory of transducers is not as advanced as automata theory.

* The full version of this article can be found at [5], http://arxiv.org/abs/1702.07157.
† This work was partially supported by the French ANR project *ExStream* (ANR-13-JS02-0010), the ARC project *Transform* (Fédération Wallonie Bruxelles) and the FNRS PDR project *Flare*. I. Jecker is an F.R.S.-FNRS Aspirant fellow.




**Figure 1** The language $A^*aA^*$ can be recognized by a deterministic (left) or codeterministic (right) automaton, but not by a reversible one.

One of the reasons for this is that many descriptions which are equivalent for automata become different in expressiveness in the case of transducers. For instance, deterministic and non-deterministic automata recognize the same class of languages, the *regular languages*. However this is not the case for transducers since in particular a deterministic transducer must realize a function while a non-deterministic one may realize a relation. Similarly, by allowing the reading head to move left and right, one gets a two-way model of automata and it is known that two-way automata are as expressive as one-way automata [12]. However two-way transducers can model relations and functions that are unobtainable in the one-way case, such as the function mirror which reverses its input. Recently, two-way transducers were also proven to be equivalent to the one-way deterministic model of *streaming string transducers* [2], which can be thought of as transducers with write-only registers.

**Reversible transducers.** A transition system is called *reversible* when for every input, the directed graph of configurations is composed of nodes of in-degree and out-degree at most one. This property is stronger than the more studied notion of determinism since it allows to navigate back and forth between the steps of a computation. In this article, we study the class of transducers that are simultaneously deterministic and codeterministic, *i.e.* reversible. The main motivation for the definition of this class is its good properties with respect to composition. When we consider one-way transducers, runs only go forward and thus determinism gives good properties for composition: the next step of a run is computed in constant time. However, when considering composition of two-way transducers, the second machine can move to the left, which corresponds to rewinding the run of the first machine. Then the stronger property of reversibility allows for this back and forth navigation over runs of transducers, and we recover the property of reaching the next (or previous here) step of a computation in a constant time. This leads to the recovery of the polynomial state complexity of composition which exists for deterministic one-way transducers.

Let us now discuss the expressiveness of reversible transducers. Regarding automata in the one-way case, it is well-known that any regular language can be recognized by a deterministic one-way automaton and symmetrically by a codeterministic one-way automaton, since the mirror of a regular language is still regular. However, the class of one-way reversible automata is very restrictive (see Figure 1 for an example or [11] for a study of its expressive power, where they are called bideterministic). It turns out however, that if we allow bidirectionality then any regular language can be recognized by a reversible automaton. In fact, a two-way reversible automaton can be constructed from either a one-way or two-way automaton using only a linear number of states (see [9] and [10], respectively). We prove, as a consequence of our main theorem, that reversible transducers are as expressive as functional two-way transducers, and exactly capture the class of regular functions. As stated earlier, regular functions are also characterized by streaming string transducers (SST). As a byproduct, we also give a quadratic construction from copyless SST to reversible transducers, improving results from [3, 6].

**Synthesis problem and uniformization of transducers.** In the bigger picture of verification, two-way transducers can be used to model transformations of programs or non-reactive systems. If we consider the synthesis problem, where the specification is given as a relation of admissible input-output pairs, an implementation is then given as a function, with the same domain, relating a unique output to a given input. The uniformization problem asks if given a relation, we can extract a function that has the same domain, and is included in the relation. We argue that the synthesis problem can be instantiated in the setting of transformations as the problem of uniformization of a non-deterministic two-way transducer by a functional transducer. Our main result states that we can uniformize any non-deterministic two-way transducer by a reversible transducer with a single exponential blow-up.

**Related work.** As stated earlier, reversible one-way automata were already considered in [11]. Two-way reversible automata were shown to capture the regular languages in [9] by a construction from a one-way deterministic automaton to a two-way reversible automaton with a linear blow-up. This construction was extended to two-way automata in [10], still with a linear complexity. However, these constructions for automata cannot be simply extended to transducers because more information is needed in order to produce the outputs at the right moment. To the best of our knowledge, reversible transducers have not been studied yet, however, since we introduce reversible transducers as a tool for the composition of transducers, our work can be linked with the construction of Hopcroft and Ullman that gives the composition of a one-way transducer and a two-way transducer, while preserving determinism. Our construction strictly improves theirs, since ours produces, with a polynomial complexity instead of an exponential one, a reversible transducer that can in turn be easily composed.

A procedure for the uniformization of a two-way non-deterministic transducer by a deterministic one has been known since [7]. The complexity of this procedure is quadruply exponential, while our construction is done in a single one, and produces a reversible transducer.

**Organization of the paper.** Preliminary definitions are given in the next Section. In Section 3, we present our main results on composability and expressiveness of reversible transducers. Section 4 is devoted to the main technical construction of the paper. Connections with streaming string transducers are discussed in Section 5 while further works are considered in Section 6.

## 2 Automata and transducers

Given a finite alphabet $A$, we denote by $A^*$ the set of finite words over $A$, and by $\varepsilon$ the empty word. We will denote by $A_{\vdash\dashv}$ the alphabet $A \uplus \{\vdash, \dashv\}$, where the new symbols $\vdash$ and $\dashv$ are called the *left* and *right endmarkers*. A *language* over $A$ is a subset $\mathcal{L}$ of $A^*$. Given two finite alphabets $A$ and $B$, a *transduction* from $A$ to $B$ is a relation $\mathcal{R} \subseteq A^* \times B^*$.

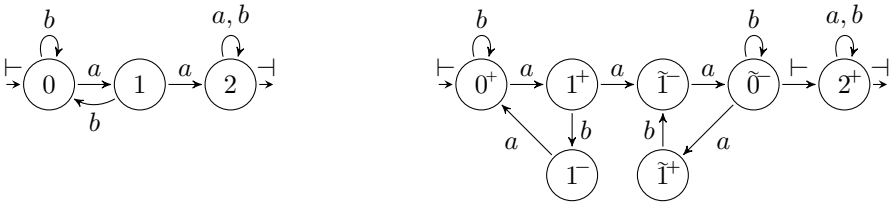
**Automata.** A *two-way finite state automaton* (2FA) is a tuple $\mathcal{A} = (A, Q, q_I, q_F, \Delta)$, where $A$ is a finite alphabet; $Q$ is a finite set of states partitioned into the set of forward states $Q^+$ and the set of backward states $Q^-$; $q_I \in Q^+$ is the initial state; $q_F \in Q^+$ is the final state; $\Delta \subseteq Q \times A_{\vdash\dashv} \times Q$ is the state transition relation. By convention, $q_I$ and $q_F$ are the only forward states verifying $(q_I, \vdash, q) \in \Delta$ and $(q, \dashv, q_F) \in \Delta$ for some $q \in Q$. However, for any backward state $p^- \in Q^-$, $\Delta$ might contain transitions $(p^-, \vdash, q)$ and $(q, \dashv, p^-)$, for some

$q \in Q$. Note that, in our figures, we do not represent explicitly the initial and final states, and rather use arrows labeled with the endmarkers to indicate the corresponding transitions. A configuration $u.p.u'$ of $\mathcal{A}$ is composed of two words $u, u' \in A^*_{\vdash\dashv}$ and a state $p \in Q$. The configuration $u.p.u'$ admits a set of successor configurations, defined as follows. If $p \in Q^+$, the input head currently reads the first letter of the suffix $u' = a'v'$. The successor of $u.p.u'$ after a transition $(p, a', q) \in \Delta$ is either $ua'.q.v'$ if $q \in Q^+$, or $u.q.u'$ if $q \in Q^-$. Conversely, if $p \in Q^-$, the input head currently reads the last letter of the prefix $u = va$. The successor of $u.p.u'$ after $(p, a', q) \in \Delta$ is $u.q.u'$ if $q \in Q^+$, or $v.q.au'$ if $q \in Q^-$. For every word $u \in A^*_{\vdash\dashv}$, a *run* of $\mathcal{A}$ on $u$ is a sequence of successive configurations $\varrho = u_0.q_0.u'_0, \dots, u_m.q_m.u'_m$ such that for every $0 \leqslant i \leqslant m$, $u_i u'_i = u$. The run $\varrho$ is called *initial* if it starts in configuration $q_I.u$, *final* if it ends in configuration $u.q_F$, *accepting* if it is both initial and final, and *end-to-end* if it starts and ends on the boundaries of $u$. More precisely, it is called *left-to-right* if $q_0, q_m \in Q^+$ and $u_0 = u'_m = \varepsilon$; *right-to-left* if $q_0, q_m \in Q^-$ and $u'_0 = u_m = \varepsilon$; *left-to-left* if $q_0 \in Q^+$, $q_m \in Q^-$ and $u_0 = u_m = \varepsilon$; *right-to-right* if $q_0 \in Q^-$, $q_m \in Q^+$ and $u'_0 = u'_m = \varepsilon$. Abusing notations, we also denote by $\Delta$ the extension of the state transition relation to a subset of $Q \times A^*_{\vdash\dashv} \times Q$ composed of the triples $(p, u, q)$ such that there exists an end-to-end run on $u$ between $p$ and $q$. For every triple $(p, u, q) \in \Delta$, we say that $q$ is a *u-successor* of $p$ and that $p$ is a *u-predecessor* of $q$. The language $\mathcal{L}_\mathcal{A}$ *recognized* by $\mathcal{A}$ is the set of words $u \in A^*$ such that $\vdash u \dashv$ admits an accepting run, i.e., $(q_I, \vdash u \dashv, q_F) \in \Delta$. The automaton $\mathcal{A}$ is called

- a *one-way finite state automaton* (1FA) if the set $Q^-$ is empty;
- *deterministic* if for all $(p, a) \in Q \times A_{\vdash\dashv}$, there is at most one $q \in Q$ verifying $(p, a, q) \in \Delta$;
- *codeterministic* if for all $(q, a) \in Q \times A_{\vdash\dashv}$, there is at most one $p \in Q$ verifying $(p, a, q) \in \Delta$;
- *reversible* if it is both deterministic and codeterministic.
- *weakly branching* if for all $a \in A$ there is at most one state $p \in Q$ and one pair of distinct states $q_1, q_2 \in Q$ such that $(p, a, q_1) \in \Delta$ and $(p, a, q_2) \in \Delta$.

An automaton with several initial and final states can be simulated by using non-determinism while reading the endmarker $\vdash$ and non-codeterminism while reading the endmarker $\dashv$, hence requiring a single initial state and a single final state does not restrict the expressiveness of our model. Let us remark that unlike in the case of most two-way machines, a reversible two-way automaton always halts on any input. Indeed, codeterminism insures that it is never the case that two transitions head to the same configuration. Hence, the unique run (due to determinism) cannot loop since no configuration can be visited twice, and the first configuration starts from the left of the initial endmarker, which is a configuration that cannot be reached later on in the run.

**Transducers.** A *two-way finite state transducer* (2FT) is a tuple $\mathcal{T} = (A, B, Q, q_I, q_F, \Delta, \mu)$, where $B$ is a finite alphabet; $\mathcal{A}_\mathcal{T} = (A, Q, q_I, q_F, \Delta)$ is a 2FA, called the *underlying automaton* of $\mathcal{T}$; and $\mu : \Delta \to B^*$ is the output function. A run of $\mathcal{T}$ is a run of its underlying automaton, and the language $\mathcal{L}_\mathcal{T}$ recognized by $\mathcal{T}$ is the language $\mathcal{L}_{\mathcal{A}_\mathcal{T}} \in A^*$ recognized by its underlying automaton. Given a run $\varrho$ of $\mathcal{T}$, we set $\mu(\varrho) \in B^*$ as the concatenation of the images by $\mu$ of the transitions of $\mathcal{T}$ occurring along $\varrho$. Note that in the deterministic (or codeterministic) case we are able to extend $\mu$ to end-to-end runs since in this case we can unambiguously associate an end-to-end run to a unique sequence of transitions $(p, u, q)$. The transduction $\mathcal{R}_\mathcal{T} \subseteq A^* \times B^*$ defined by $\mathcal{T}$ is the set of pairs $(u, v)$ such that $u \in \mathcal{L}_\mathcal{T}$ and $\mu(\varrho) = v$ for an accepting run $\varrho$ of $\mathcal{A}_\mathcal{T}$ on $\vdash u \dashv$. Two transducers are called *equivalent* if they define the same transduction. A transducer $\mathcal{T}$ is respectively called one-way (1FT), deterministic, weakly branching, codeterministic or reversible, if its underlying automaton has the corresponding property.




**(a)** A deterministic 1FA $\mathcal{A}_1$ **(b)** A reversible 2FA $\mathcal{A}_2$

**Figure 2** Two automata recognizing the language $\mathcal{L}_{aa} = A^*aaA^*$.

**Examples.** Let us consider the language $\mathcal{L}_{aa} \subseteq \{a, b\}^*$ composed of the words that contain two $a$ symbols in a row. This language is recognized by the deterministic one-way automaton $\mathcal{A}_1$, represented in Figure 2a, and by the reversible two-way automaton $\mathcal{A}_2$, represented in Figure 2b. However, it is not recognizable by a one-way reversible automaton. Let us analyze the behavior of $\mathcal{A}_2$ to see how moving back and forth through the input allows it to recognize $\mathcal{L}_{aa}$ in a reversible manner. First, $\mathcal{A}_2$ uses an intermediate step to go from $1^+$ back to $0^+$ when reading a $b$, to avoid creating non-codeterminism. Second, once $\mathcal{A}_2$ reads two consecutive $a$ symbols, it does not go directly in the final state looping on every input, since this would generate non-codeterminism. Instead, $\mathcal{A}_2$ goes in an inverse copy of the first three states, where it rewind its run until the left endmarker. It is then free to go in the looping accepting state.

# 3 Results on reversible transducers

In this section, we present the main results of our paper. In Subsection 3.1, we show the polynomial composition of reversible transducers. In the following, we give expressiveness results of the class of reversible transducers, relying on this composition procedure as well as the construction presented in Section 4.

## 3.1 Composition of reversible transducers

The nicest feature of reversible transducers has to be the low complexity (and simplicity) of their composition. Indeed the composition of two such transducers is polynomial in the number of states of the inputs, and the construction is itself quite simple. This is due to the fact that the difficult part in the composition of transducers is to be able to navigate the run easily. In the one-way case, the composition is easy since runs can only move forward. In the two-way case, one needs to advance in the run, but also rewind it. Since the former is made easy by the determinism, and the latter is symmetrically handled by the codeterministim, composition of reversible transducers is straightforward. Let us also remark that only the first transducer has to be reversible in order to obtain a polynomial complexity. However the reversible nature of the obtained transducer depends on the input transducers being both reversible.

▶ **Theorem 1.** *Let $\mathcal{T}_1$ be a reversible two-way transducers and $\mathcal{T}_2$ be two-way transducer with $n_1$ and $n_2$ states respectively, such that $\mathcal{T}_1$ can be composed with $\mathcal{T}_2$. Then one can construct a two-way transducer $\mathcal{T}_3$ with $n_1 \cdot n_2$ states realizing $\mathcal{R}_{\mathcal{T}_2} \circ \mathcal{R}_{\mathcal{T}_1}$.*
*Furthermore, if $\mathcal{T}_2$ is reversible, deterministic or codeterministic, then so is $\mathcal{T}_3$.*

**Proof.** Let $\mathcal{T}_1 = (A, B, Q, q_I, q_F, \Delta, \mu)$ and $\mathcal{T}_2 = (B, C, P, p_I, p_F, \Gamma, \nu)$. We define $\mathcal{T}_3 = (A, C, Q \times P, (q_I, p_I), (q_F, p_F), \Theta, \xi)$. The idea is that at each step, $\mathcal{T}_3$ simulates a transition

$\delta \in \Delta$, plus the behavior of $\mathcal{T}_2$ over the production $\mu(\delta) \in B^*$ of this transition. To be precise, the transducer $\mathcal{T}_3$ also detects when it simulates an initial or final configuration of $\mathcal{T}_1$ (i.e. upon reading an endmarker in the initial or final state), and accordingly adds the corresponding endmarker to the production before simulating $\mathcal{T}_2$. The partition of the set of states of $\mathcal{T}_3$ depends on the combination of the signs of both components. If $\mathcal{T}_2$ is moving to the right, we use the determinism of $\mathcal{T}_1$, we update the first component of the current state according to the unique transition $\delta$ originating from it, and we simulate $\mathcal{T}_2$ entering $\mu(\delta)$ from the left. To do so, $\mathcal{T}_3$ needs to have access to the same letter of the input tape as $\mathcal{T}_1$. Thus, we have $(Q^+ \times P^+) \subseteq (Q \times P)^+$ and $(Q^- \times P^+) \subseteq (Q \times P)^-$. If $\mathcal{T}_2$ is moving to the left, then we use the codeterminism of $\mathcal{T}_1$ to rewind the corresponding run, we update the first component of the current state according to the unique transition $\delta$ arriving in it, and we simulate $\mathcal{T}_2$ entering $\mu(\delta)$ from the right. To do so, $\mathcal{T}_3$ needs to have access to the letter on the other side of the input head (with respect to $\mathcal{T}_1$). Thus, we have $(Q^- \times P^-) \subseteq (Q \times P)^+$ and $(Q^+ \times P^-) \subseteq (Q \times P)^-$.

We now define the transition function $\Theta$ and the production function $\xi$. Let $(q, a, q') \in \Delta$ be a transition of $\mathcal{T}_1$ such that $\varrho = (p, v, p')$ is an end-to-end run of $\mathcal{T}_2$, where $v$ denotes the word $\mu(q, a, q') \in B^*$.

- If $\varrho$ is a left-to-right run of $\mathcal{T}_2$, then $((q, p), a, (q', p'))$ belongs to $\Theta$ and produces $\nu(p, v, p')$.
- If $\varrho$ is a left-to-left run of $\mathcal{T}_2$, then $((q, p), a, (q, p'))$ belongs to $\Theta$ and produces $\nu(p, v, p')$.
- If $\varrho$ is a right-to-right run of $\mathcal{T}_2$, then $((q', p), a, (q', p'))$ belongs to $\Theta$ and produces $\nu(p, v, p')$.
- If $\varrho$ is a right-to-left run of $\mathcal{T}_2$, then $((q', p), a, (q, p'))$ belongs to $\Theta$ and produces $\nu(p, v, p')$.

The behavior of the transducer $\mathcal{T}_3$ is completely determined by the combined behaviors of transducers $\mathcal{T}_1$ and $\mathcal{T}_2$. When $\mathcal{T}_3$ simulates a transition of $\mathcal{T}_1$, it also simulates the corresponding end-to-end run of $\mathcal{T}_2$ over the production of this transition. If the direction of both simulations is the same, then $\mathcal{T}_3$ moves forward. Otherwise, it moves backward. The transducer stops when it reaches a final state in both $\mathcal{T}_1$ over the input, and $\mathcal{T}_2$ over the simulated run over partial productions of the run of $\mathcal{T}_1$ over the input. Then the final output of $\mathcal{T}_3$ is the concatenation of the outputs of the partial runs of $\mathcal{T}_2$ it simulates, which corresponds to the output of $\mathcal{T}_1$. Hence, the transducer $\mathcal{T}_3$ realizes the composition $\mathcal{T}_2 \circ \mathcal{T}_1$. The possible determinism (resp. codeterminism) of $\mathcal{T}_3$ is a direct consequence of the one of $\mathcal{T}_1$ and $\mathcal{T}_2$. Indeed, a witness of non-determinism (resp. non codeterminism) of $\mathcal{T}_3$ can be traced back to a witness run of either $\mathcal{T}_1$ or $\mathcal{T}_2$ that is not deterministic (resp.codeterministic). ◀

## 3.2 One-way transducers

In the next subsections, we give some procedures to construct a reversible transducer from either a one-way or a two-way transducer. The main ingredient of the proofs is the technical construction from Lemma 6 (presented in Section 4) which constructs a reversible transducer from a *weakly branching* codeterministic one-way transducer. The proofs of this section share the same structure: in order to build a reversible transducer that defines a function $\mathcal{F}$, we express $\mathcal{F}$ as a composition of transductions definable by reversible transducers, and we conclude by using Theorem 1. The detailed constructions are presented in the full version of this article. Building on Lemma 6, we show that codeterministic one-way transducers can be expressed as the composition of weakly branching codeterministic ones.

▶ **Theorem 2.** *Given a codeterministic 1FT with $n$ states, one can effectively construct an equivalent reversible 2FT with $4n^2$ states.*

**Proof.** Let $\mathcal{T}$ be a codeterministic 1FT with $n$ states. The function $\mathcal{R}_\mathcal{T}$ can be expressed as the composition $\mathcal{R}_{\mathcal{T}'} \circ \mathcal{R}_\mathcal{M}$, where $\mathcal{M}$ and $\mathcal{T}'$ are defined as follows.

- Transducer $\mathcal{M}$ is a reversible 1FT with a single state that multiplies all the letters of the input word by $n$ while marking them with a state of $\mathcal{T}$;
- Transducer $\mathcal{T}'$ is a weakly branching and codeterministic one-way transducer that has the same set of states as $\mathcal{T}$. On input $\mathcal{R}_\mathcal{M}(u)$, $\mathcal{T}'$ mimics the behavior of $\mathcal{T}$ on $u$, while using the fact that the input word is larger to desynchronize the non-deterministic branchings that were occurring simultaneously in $\mathcal{T}$. Intuitively, a transition of $\mathcal{T}$ can only be taken by $\mathcal{T}'$ at the copy of the letter corresponding to the target state of the transition.

By Lemma 6, $\mathcal{T}'$ can be made into a reversible 2FT $\mathcal{T}''$ with $4n^2$ states. Therefore, since both $\mathcal{T}''$ and $\mathcal{M}$ are reversible, we can conclude using Theorem 1, finally obtaining a reversible 2FT with $4n^2$ states equivalent to $\mathcal{T}$. ◀

Using composition again, the statement can be extended to deterministic one-way transducers.

▶ **Theorem 3.** *Given a deterministic 1FT with $n$ states, one can effectively construct an equivalent reversible 2FT with $36n^2$ states.*

**Proof.** Let $\mathcal{T}$ be a deterministic 1FT with $n$ states. Then $\overline{\mathcal{T}}$, the transducer obtained by reversing all transitions of $\mathcal{T}$, is codeterministic. The function $\mathcal{R}_\mathcal{T}$ can be expressed as the composition $\mathcal{R}_{M_B} \circ \mathcal{R}_{\overline{\mathcal{T}}} \circ \mathcal{R}_{M_A}$, where $M_A$ and $M_B$ realize the mirror functions over the input and output alphabet of $\mathcal{T}$ respectively. Both of them are realized by a 3 states reversible transducer. Then by Theorem 2, we can construct $\overline{\mathcal{T}}'$ which has $4n^2$ states, is reversible and realizes the same function as $\overline{\mathcal{T}}$. By Theorem 1, we can compose the three transducers, finally obtaining a reversible transducer equivalent to $\mathcal{T}$ with $9 \cdot 4n^2$ states. ◀

## 3.3 Two-way transducers

We now prove our main result, which states that any two-way transducer can be uniformized by a reversible two-way transducer. Let us recall that uniformization by a deterministic transducer was done in [7]. We use similar ideas for the uniformization. The key difference is that we rely on the construction of Section 4 while in [7], the main construction is the tree-trimming construction of Hopcroft-Ullman from [8].

▶ **Theorem 4.** *Given a 2FT $\mathcal{T}$ with $n$ states, one can effectively construct a reversible 2FT $\mathcal{T}'$ whose number of states is exponential in $n$, and verifying $\mathcal{L}_{\mathcal{A}_{\mathcal{T}'}} = \mathcal{L}_{\mathcal{A}_\mathcal{T}}$ and $\mathcal{R}_{\mathcal{T}'} \subseteq \mathcal{R}_\mathcal{T}$.*

**Proof.** Let $\mathcal{T} = (A, B, Q, q_I, q_F, \Delta, \mu)$ be a 2FT with $n$ states. We define a function uniformizing $\mathcal{R}_\mathcal{T}$ as the composition $\mathcal{R}_{\mathcal{T}'} \circ \mathcal{R}_\mathcal{U} \circ \mathcal{R}_{\mathcal{D}_r}$, where $\mathcal{D}_r$, $\mathcal{U}$ and $\mathcal{T}$ are defined as follows.

- The right-oracle $\mathcal{D}_r$ is a codeterministic one-way transducer with $2^{n^2+n}$ states that enriches each letter of the input word $u \in A^*_{\vdash\dashv}$ with information concerning the behavior of $\mathcal{T}$ on the corresponding suffix, represented by the set of pairs that admit a left-to-left run, and the set of states from which $\mathcal{T}$ can reach the final state.
- The uniformizer $\mathcal{U}$ is a deterministic one-way transducer with $n!$ states. On input $u' = \mathcal{R}_{\mathcal{D}_r}(u)$, $\mathcal{U}$ uses the information provided by $\mathcal{D}_r$ to pick a run $\varrho_u$ of $\mathcal{T}$ on input $u$, and enriches each letter $a_i$ of the input word with the sequence of transitions occurring in the run $\varrho_u$ that correspond to the letter $a_i$.
- Finally, the reversible transducer $\mathcal{T}'$ has the same set of states as $\mathcal{T}$, and follows the instructions left by $\mathcal{U}$ to solve the non-determinism and the non-codeterminism.

As a consequence of Theorem 2 and Theorem 3, there exist two reversible 2FT $\mathcal{D}_r{}'$ and $\mathcal{U}'$ whose number of states are exponential in $n$, and that verify $\mathcal{R}_{\mathcal{D}_r'} = \mathcal{R}_{\mathcal{D}_r}$ and $\mathcal{R}_{\mathcal{U}'} = \mathcal{R}_{\mathcal{U}}$. Therefore, since $\mathcal{D}_r{}', \mathcal{U}'$ and $\mathcal{T}'$ are reversible, by Theorem 1 there exists a reversible transducer $\mathcal{T}''$ whose number of states is exponential in $n$, and that satisfies $\mathcal{R}_{\mathcal{T}''} = \mathcal{R}_{\mathcal{T}'} \circ \mathcal{R}_{\mathcal{U}'} \circ \mathcal{R}_{\mathcal{D}_r{}'} = \mathcal{R}_{\mathcal{T}}$. ◀

The following result is a direct corollary of Theorem 4, applied to deterministic two-way transducers.

▶ **Corollary 5.** *Reversible two-way transducers are as expressive as deterministic two-way transducers.*

## 4 The tree-outline construction

In this section lies the heart of our result. We show that any *weakly branching* and codeterministic transducer can be made reversible. These hypotheses allow us to simplify our proof, and still obtain a more general result, as a corollary.

▶ **Lemma 6.** *Let $\mathcal{T}$ be a codeterministic and weakly branching 1FT with $m$ states. Then one can effectively construct a reversible 2FT $\mathcal{T}'$ with $4m^2$ states that is equivalent to $\mathcal{T}$.*

**Proof.** The construction presented in this proof is illustrated on an example in Figure 3. Let $\mathcal{T} = (A, Q, q_I, q_F, \Delta, \mu)$ be a codeterministic 1FT, and let $\prec$ be a total order over $Q$. As an example, take the codeterministic 1FT $T$ presented in figure 3a.

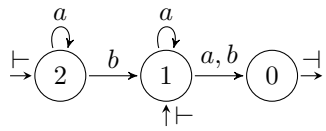
Let $\mathcal{T}' = (A, \mathcal{F}, f_I, f_F, \Delta', \mu')$ be a 2FT defined as follows:

On input $u \in \mathcal{L}_{\mathcal{T}}$, $\mathcal{T}'$ explores depth first the run-tree $T_u$ composed of the initial runs of $\mathcal{T}$ on the word $\vdash u \dashv$ (illustrated in Figure 3b). More precisely it explores the "sheath" of the run-tree (see Figure 3c for a graphical representation). To do this, the states of $\mathcal{T}'$ are composed of two states of $\mathcal{T}$ with a marker. The first state represents the upper part of the sheath, while the second state represents the lower part. Moreover the marker is used to denote whether we are above the branch ($\underline{q}$) or below the branch ($\overline{q}$).

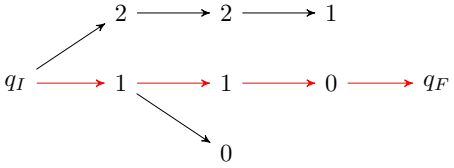
Initially we start with the state $(\underline{q_I}, \overline{q_I})$ and go forward according to the transitions of $\mathcal{T}$. While moving forward whenever a branching state $q$ is reached, if the state is marked $\underline{q}$ it moves to the maximal successor of $q$ (in order to stay above the branch) and symmetrically if the state is marked $\overline{q}$ it moves to the minimal successor of $q$ (in order to stay below the branch). Whenever one of the branches reaches a dead end we continue the sheath exploration by switching the marker (*i.e.* changing from above the branch to below or vice-versa) and start moving backward accordingly to the transitions of $\mathcal{T}$. While moving backward, if the successor of a branching state $q$ is reached, while we were inside the fork, *e.g.* in state $\overline{q_{\max}}$ (where $q_{\max}$ is the maximal successor of $q$), we continue the exploration of the sheath by going in the state $\underline{q_{\min}}$ and we start moving forward again. Whenever the upper and lower explorations of the sheath coincide, *i.e.* in states of the form $(\underline{q}, \overline{q})$ (represented in red in Figure 3d), it means we are on a prefix of the accepting run, we can thus produce the corresponding output.

Formally $\mathcal{T}' = (A, \mathcal{F}, f_I, f_F, \Delta', \mu')$ is defined as follows:

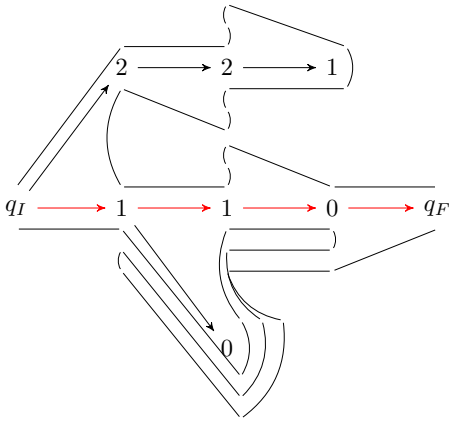
- $\mathcal{F} = \mathcal{F}^+ \cup \mathcal{F}^-$ where $\mathcal{F}^+ = \underline{Q} \times \overline{Q} \cup \overline{Q} \times \underline{Q}$ and $\mathcal{F}^- = (\overline{Q} \times \overline{Q} \cup \underline{Q} \times \underline{Q}) \backslash \{(\overline{p}, \overline{p}), (\underline{p}, \underline{p}) \mid p \in Q\}$
- $f_I = (\underline{q_I}, \overline{q_I})$
- $f_F = (\underline{q_F}, \overline{q_F})$
- We define the transition relation $\Delta'$ by differentiating several types of behavior, depending on whether we are going **f**orward, or **b**ackward, whether the **u**pper component or the



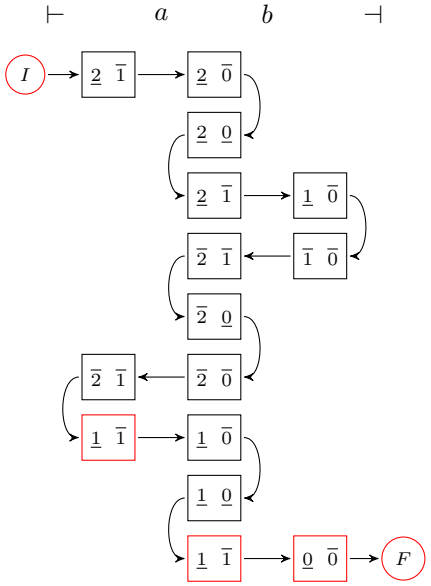

**(a)** A codeterministic transducer $\mathcal{T}$




**(b)** The run-tree of $\mathcal{T}$ on $\vdash ab \dashv$




**(c)** Graphical representation of the run of $\mathcal{T}'$




**(d)** The run of $\mathcal{T}'$

**Figure 3** Illustrations of the proof concepts of Lemma 6. For the sake of clarity, the outputs of $\mathcal{T}$ are omitted.

lower component is involved, and whether it is above or below its branch. Let $p$ and $q$ be two states in $Q$, and $a \in A$ be a letter.

If $p$ has no $a$-successor, then:

**(fua)** $\big((\underline{p}, \overline{q}), a, (\overline{p}, \overline{q})\big) \in \Delta'$, and

**(fuw)** $\big((\overline{p}, \underline{q}), a, (\underline{p}, \underline{q})\big) \in \Delta'$.

If $p$ has an $a$-successor, but not $q$, then:

**(flw)** $\big((\underline{p}, \overline{q}), a, (\underline{p}, \underline{q})\big) \in \Delta'$, and

**(fla)** $\big((\overline{p}, \underline{q}), a, (\overline{p}, \overline{q})\big) \in \Delta'$.

Otherwise, $p$ and $q$ admit an $a$-successor. We denote $p_{\max}$ (resp. $p_{\min}$) the maximal (resp. minimal) $a$-successor of $p$ (resp. $q$) with respect to $\prec$. Then:

If $p_{\min} \neq p_{\max}$, then:

**(buw)** $\big((\overline{p_{\max}}, \overline{q}), a, (\underline{p_{\min}}, \overline{q})\big) \in \Delta'$, and

**(bua)** $\big((\underline{p_{\min}}, \underline{q}), a, (\overline{p_{\max}}, \underline{q})\big) \in \Delta'$.

If $q_{\min} \neq q_{\max}$, then:

**(bla)** $\big((\underline{p}, \underline{q_{\min}}), a, (\underline{p}, \overline{q_{\max}})\big) \in \Delta'$

**(blw)** $\big((\overline{p}, \overline{q_{\max}}), a, (\overline{p}, \underline{q_{\min}})\big) \in \Delta'$

**(fualw)** $\big((\underline{p}, \overline{q}), a, (\underline{p_{\max}}, \overline{q_{\min}})\big) \in \Delta'$,

**(fuwla)** $\big((\overline{p}, \underline{q}), a, (\overline{p_{\min}}, \underline{q_{\max}})\big) \in \Delta'$,

**(bulw)** $\big((\overline{p_{\min}}, \overline{q_{\min}}), a, (\overline{p}, \overline{q})\big) \in \Delta'$, and

**(bula)** $\big((\underline{p_{\max}}, \underline{q_{\max}}), a, (\underline{p}, \underline{q})\big) \in \Delta'$.

- We define $\mu'$ as the function such that for every $(p, a, q) \in \Delta$:
  - if $q = p_{\min} = p_{\max}$ then $\mu'\big((\underline{p}, \overline{p}), a, (\underline{q}, \overline{q})\big) = \mu(p, a, q)$
  - if $q = p_{\min} \neq p_{\max}$ then $\mu'\big((\overline{p_{\max}}, \overline{q}), a, (\underline{q}, \overline{q})\big) = \mu(p, a, q)$
  - if $q = p_{\max} \neq p_{\min}$ then $\mu'\big((\underline{q}, \underline{p_{\min}}), a, (\underline{q}, \overline{q})\big) = \mu(p, a, q)$

  and $\mu'(t) = \varepsilon$ for every $t \in \Delta'$ which is not of one of theses forms.

One can see, by a case study that $\mathcal{T}'$ is deterministic. Indeed, the fact that $\mathcal{T}$ is weakly branching implies that the rules **(buw)** and **(bua)** are mutually exclusive with the rules **(bla)** and **(blw)**. Moreover these four rules are mutually exclusive with the rules **(bulw)** and **(bula)** by construction. And since $\mathcal{T}$ is codeterministic, the predecessor is unique. Finally, the rules **(fua)**, **(fuw)**, **(flw)**, **(fla)**, **(fualw)**, and **(fuwla)** are mutually exclusive by construction, since the conditions on the number of $a$-successors are incompatible.

A similar case study gives that $\mathcal{T}'$ is codeterministic. Hence $\mathcal{T}'$ is reversible.

A detailed proof of the equivalence between $\mathcal{T}$ and $\mathcal{T}'$ can be found in the full version of the article, and we give a quick intuition of the proof. It relies on two main arguments. The first one is that at any point if the transducer $\mathcal{T}'$ follows two different runs, then it will come back to the same position, where the state that leads to the shortest run has been switched. Following this, we then prove that upon any branching, $\mathcal{T}'$ comes back to the same position but since the shortest run has been switched, it is able to solve the non-determinism, take the transition of the accepting run and produce the correct output. ◀

## 5 Streaming string transducers

Streaming string transducers, which were introduced in [2], are one-way deterministic automata with additional write-only registers. Partial outputs are stored in the registers via register updates, and at the end of a run an output is produced using these registers. Thus an SST realizes a function over words, and it is known that they are as expressive as 2FT [2]. Direct transformations from SST to 2FT were already considered in [3, 6]. However, these constructions were exponential in the number of states (and linear in the number of registers). Using Theorem 3, we are able to get a construction which is quadratic in the number of states (and also linear in the number of registers). Before explaining the construction, let us formally define the SST.

**Substitutions.** Given a finite alphabet $A$ and a finite set $\mathcal{X}$ of variables. Let $\mathcal{S}_{\mathcal{X},A}$ denote the set of functions $\sigma : \mathcal{X} \to (\mathcal{X} \cup A)^*$. The elements of $\mathcal{S}_{\mathcal{X},A}$ are called *substitutions*. Any substitution $\sigma$ can be extended to range over both variables and letters of the output alphabet $\hat{\sigma} : (\mathcal{X} \cup A)^* \to (\mathcal{X} \cup A)^*$ by setting $\hat{\sigma}(a) = a$ for every $a \in A^*$ and $\hat{\sigma}(uv) = \hat{\sigma}(u)\hat{\sigma}(v)$ for $u, v \in (\mathcal{X} \cup A)^*$. This allows us to easily compose substitutions from $\mathcal{S}_{\mathcal{X},A}$ by defining $\sigma_2 \circ \sigma_1$ as the usual function composition $\hat{\sigma_2} \circ \sigma_1$. We denote by $\mathsf{Id}_{\mathcal{X}}$ the identity element of $\mathcal{S}_{\mathcal{X},A}$, which maps every variable to itself, and by $\sigma_\varepsilon$ the substitution mapping every variable to $\varepsilon$.

A substitution $\sigma$ is called *copyless* if for every $X \in \mathcal{X}$, each variable $Y \in \mathcal{X}$ appears at most once in $\sigma(X)$, and for every $Y \in \mathcal{X}$ there exists at most one $X \in \mathcal{X}$ such that $Y$ appears in $\sigma(X)$.

**Streaming string transducers.** A *streaming string transducer* (SSTfor short) is a tuple $\mathcal{Z} = (A, B, Q, q_I, q_F, \Delta, \mathcal{X}, O, \tau)$, where $B$ is the output alphabet, $\mathcal{A}_\mathcal{Z} = (A, Q, q_I, q_F, \Delta)$ is a one-way deterministic automaton, called the underlying automaton of $\mathcal{Z}$; $\mathcal{X}$ is a finite set of variables; $O \in \mathcal{X}$ is the final variable; $\tau : \Delta \to \mathcal{S}_{\mathcal{X},B}$ is the output function. A run of $\mathcal{Z}$ is a run of its underlying automaton, and the language $\mathcal{L}_\mathcal{Z}$ recognized by $\mathcal{Z}$ is the language $\mathcal{L}_{\mathcal{A}_\mathcal{Z}} \in A^*$ recognized by its underlying automaton. Given a run $\varrho$ of $\mathcal{Z}$ on $u$, we set $\tau(\varrho) \in \mathcal{S}_{\mathcal{X},B}$ as the composition of the images by $\tau$ of the transitions of $\mathcal{Z}$ occuring along $\varrho$. The transduction $\mathcal{R}_\mathcal{Z} \subseteq A^* \times B^*$ defined by $\mathcal{Z}$ is the function mapping any word $u$ of $\mathcal{L}_{\mathcal{A}_\mathcal{Z}}$ to $(\sigma_\varepsilon \circ \tau(\varrho))(O)$, where $\varrho$ is the single accepting run of $\mathcal{A}_\mathcal{Z}$ on $\vdash u \dashv$. The SST $\mathcal{Z}$ is called copyless, if for every run $\varrho$ of $\mathcal{Z}$ the substitution $\tau(\varrho)$ is copyless.

▶ **Theorem 7.** *Given a copyless SST with $n$ states and $m$ variables, one can effectively construct an equivalent reversible 2FT with $8m \cdot n^2$ states.*

**Proof.** We write $\mathcal{Z}$ as the composition of a one-way deterministic transducer $\mathcal{D}_1$ and a reversible one $\mathcal{T}$. The first transducer has the same underlying automaton as $\mathcal{Z}$, the difference being that it outputs the substitution of $\mathcal{Z}$ instead of applying it. Then $\mathcal{T}$ is a transducer that navigates the substitutions to produce the output word of $\mathcal{Z}$. This can be done in a reversible fashion thanks to the property of copylessness of $\mathcal{Z}$. Note that the transducer $\mathcal{T}$ was already defined in [6], Section 4. Formally, let $\mathcal{Z} = (A, B, Q, q_I, q_F, \Delta, \mathcal{X}, O, \tau)$ be a copyless SST with $n$ states and $m$ variables, and let $S_{\mathcal{Z}} \subset \mathcal{S}_{A,\mathcal{X}}$ be the range of $\tau$. We express $\mathcal{R}_{\mathcal{Z}}$ as the composition of $\mathcal{R}_{\mathcal{D}_1} : \mathcal{L}_{\mathcal{A}_{\mathcal{Z}}} \to S_{\mathcal{Z}}^*$ and $\mathcal{R}_{\mathcal{T}_2} : S_{\mathcal{Z}}^* \to B^*$, defined as follows.

- $\mathcal{D}_1$ is a deterministic 1FT obtained by stripping $\mathcal{Z}$ of its SST structure, i.e., $\mathcal{D}_1 = (A, S_{\mathcal{Z}}, Q, q_I, q_F, \Delta, \tau)$. It maps each word of $\mathcal{L}_{\mathcal{A}_{\mathcal{Z}}}$ to the corresponding sequence of substitutions.
- $\mathcal{T} = (S_{\mathcal{Z}}, B, P, init, fin, \Gamma, \nu)$ where $P^+ = \mathcal{X}^o \uplus \{init, fin\}$, $P^- = \mathcal{X}^i$. States labeled by $i$ (resp. $o$) are *in* (resp. *out*) states and appear when we start (resp. finish) producing a variable. We define $\Gamma$ and $\nu$ as follows:
  - $(init, \sigma, init) \in \Gamma$;
  - $(init, \vdash, O^i) \in \Gamma$;
  - $(O^o, \dashv, fin) \in \Gamma$;
  - $(X^i, \sigma, Y^i) \in \Gamma$ and $\nu((X^i, \sigma, Y^i)) = v$ if $\sigma(X) = vY...$ with $v \in B^*$;
  - $(X^i, \sigma, X^o) \in \Gamma$ and $\nu((X^i, \sigma, X^o)) = v$ if $\sigma(X) = v$;
  - $(X^o, \sigma, Y^i) \in \Gamma$ and $\nu((X^o, \sigma, Y^i)) = v$ if there exists a variable $Z$ where $\sigma(Z) = ...XvY..$;
  - $(X^o, \sigma, Y^o) \in \Gamma$ and $\nu((X^o, \sigma, Y^o)) = v$ if $\sigma(Y) = ...Xv$.

  Due to copylessness, for any $\sigma$ and any variable $X$, there is at most one variable $Y$ such that $X$ appears in $\sigma(Y)$. Plus, as the variables are ordered by their appearance in $\sigma(Y)$, the transducer $\mathcal{T}$ is reversible. It starts by reaching the end of the word, then starts producing the variable $O$. By following the substitution tree of $O$, it then produces exactly the image of the input by $\mathcal{Z}$.

By Theorem 3, there exists a reversible 2FT $\mathcal{D}_1'$ with $4n^2$ states satisfying $\mathcal{R}_{\mathcal{D}_1'} = \mathcal{R}_{\mathcal{D}_1}$. Finally, since both $\mathcal{D}_1'$ and $\mathcal{T}$ are reversible, by Theorem 1 there exists a reversible transducer $\mathcal{T}'$ with $8m \cdot n^2$ states such that $\mathcal{R}_{\mathcal{T}'} = \mathcal{R}_{\mathcal{T}} \circ \mathcal{R}_{\mathcal{D}_1'} = \mathcal{R}_{\mathcal{Z}}$. ◀

## 6 Conclusion

We argue that reversible transducers can be seen as a canonical representation of regular transductions. We believe that the polynomial complexity of composition of reversible transducers is a good tool for the verification of cascades of transformations of non-reactive systems. While preserving the expressive power of functional transducers, reversible transducers allow for easier manipulations, the best example being their polynomial composition. Thanks to the tree-outline construction that we presented, one can uniformize a non-determinsitic two-way transducer by a reversible one with a single exponential blow-up. This improves the known constructions that were used up to now, however it is still open whether this blow-up can be avoided. In [10] the authors extended the result of [9] and showed that deterministic two-way automata can be made reversible with a linear blow-up. We conjecture that our approach can also be extended to the two-way case and that deterministic two-way transducers can be made reversible using only a polynomial number of states.

We have shown that applying this construction allows for a quadratic transformation from copyless streaming string transducers to reversible two-way transducers. The converse does not hold, since even on languages deterministic two-way automata are known to be exponentially more succint than deterministic one-way automata. Beyond this, we do not reject the possibility that if one were to embed some recognition power into the variables of a SST, it may be possible to have a polynomial transformation from reversible automata to copyless SST.

## References


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

2 Rajeev Alur and Pavol Cerný. Expressiveness of streaming string transducers. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, pages 1–12, 2010.

3 Rajeev Alur, Emmanuel Filiot, and Ashutosh Trivedi. Regular transformations of infinite strings. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pages 65–74, 2012.

4 Julius R. Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundlagen Math.*, 6:66–92, 1960.

5 Luc Dartois, Paulin Fournier, Ismaël Jecker, and Nathan Lhote. On reversible transducers. *CoRR*, abs/1702.07157, 2017. URL: `http://arxiv.org/abs/1702.07157`.

6 Luc Dartois, Ismaël Jecker, and Pierre-Alain Reynier. Aperiodic string transducers. In *Developments in Language Theory – 20th International Conference, DLT 2016, Montréal, Canada, July 25-28, 2016, Proceedings*, pages 125–137, 2016.

7 Rodrigo de Souza. Uniformisation of two-way transducers. In *Language and Automata Theory and Applications – 7th International Conference, LATA 2013, Bilbao, Spain, April 2-5, 2013. Proceedings*, pages 547–558, 2013.

8 John E. Hopcroft and Jeffrey D. Ullman. An approach to a unified theory of automata. In *8th Annual Symposium on Switching and Automata Theory, Austin, Texas, USA, October 18-20, 1967*, pages 140–147, 1967.

9 Attila Kondacs and John Watrous. On the power of quantum finite state automata. In *38th Annual Symposium on Foundations of Computer Science, FOCS'97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 66–75, 1997.

10 Michal Kunc and Alexander Okhotin. Reversibility of computations in graph-walking automata. In *Mathematical Foundations of Computer Science 2013 – 38th International Symposium, MFCS 2013, Klosterneuburg, Austria, August 26-30, 2013. Proceedings*, pages 595–606, 2013.

11 Jean-Eric Pin. On reversible automata. In *LATIN'92, 1st Latin American Symposium on Theoretical Informatics, São Paulo, Brazil, April 6-10, 1992, Proceedings*, pages 401–416, 1992.

12 John C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM Journal of Research and Development*, 3(2):198–200, 1959.