Invertible Transducers, Iteration and Coordinates

K. Sutner

Carnegie Mellon University
Pittsburgh, PA 15213, USA

Abstract. We study natural computational problems associated with iterated transductions defined by a class of invertible transducers over the binary alphabet. The transduction semigroups of these automata are known to be free Abelian groups and the orbits of words can be described as affine subspaces in a suitable geometry defined by the generators of these groups. We show how to compute the associated coordinates of words in quadratic time and how to apply coordinates to the problem of deciding whether two words generate the same orbit under a given transduction. In some special cases our algorithms can be implemented on finite state machines.

1 Invertible Transducers

We consider Mealy automata $\mathcal{A}$ where all transitions are of the form $p \xrightarrow{a/\pi(a)} q$; here $\pi = \pi_p$ is a permutation of the alphabet $\mathbf{2} = \{0, 1\}$ that depends on the source p of the transition. When π is the transposition we refer to p as a *toggle state*, and as a *copy state*, otherwise. By selecting any state p in $\mathcal{A}$ as the initial state we obtain a transduction $\mathcal{A}(p) : \mathbf{2}^\star \to \mathbf{2}^\star$. A moment's thought reveals that $\mathcal{A}(p)$ is a length-preserving permutation of $\mathbf{2}^\star$. Moreover, the corresponding inverse permutation can be obtained by interchanging 0 and 1 labels in all transitions. Correspondingly, these automata are called *binary invertible transducers*. The groups generated by the collection of all transitions $\mathcal{A}(p)$ as p ranges over the state set of $\mathcal{A}$ have attracted considerable attention in the last two decades, see [1, 6, 7, 13]. One reason invertible transducers are relevant in group theory and symbolic dynamics is that they afford very compact descriptions of surprisingly complicated groups. For example, Grigorchuk has constructed a group of intermediate growth that can be interpreted as the transduction group of a binary invertible transducer on only 5 states and with a single toggle state.

Our objective here is the study of iteration and in particular the computational complexity of problems associated with iterated transductions in an invertible transducer. Write $\mathcal{S}(\mathcal{A})$ for the semigroup generated by the basic transductions $\mathcal{A}(p)$. Each transduction f in $\mathcal{S}(\mathcal{A})$ defines its iterate $f^\star \subseteq \mathbf{2}^\star \times \mathbf{2}^\star$, a length-preserving equivalence relation on $\mathbf{2}^\star$ that we will refer to as the *orbit relation* of f: two words are related by $f^\star$ if they have the same orbits (as sets) under f.

Any equivalence relation on $\mathbf{2}^\star$ is associated with a number of natural decision problems. First, there is the recognition problem, the problem of deciding $x f^\star y$ given two words x and y. In our context we will refer to this as the *Orbit Problem*. Second, and closely related, is the first canonical form problem, the question of how hard it is to compute the *root function* $x \mapsto \min\big(z \in \mathbf{2}^\star \mid z\, f^\star\, x \big)$; here the minimum is understood to be with respect to length-lexicographical order. See [4] for a recent discussion of general complexity results relating to these questions, and [8] for results relating to rational relations. Since our equivalence relations are generated by iteration there are several other natural problems to consider. The *Iteration Problem* asks for the complexity of computing $x\, f^t$ for some transduction f, a word x and $t \geq 0$. The recognition problem has a slightly stronger variant that we refer to as the *Timestamp Problem*: given words x and y, find the least number $t \geq 0$ such that $x\, f^t = y$, or determine that no such t exists. Since f is length-preserving we only need to consider words of length k, in which case the brute-force method takes $O(k\, 2^k)$ steps for either problem; of course, we are interested in polynomial time solutions.

As Grigorchuk's example shows, even small invertible transducers with a single toggle state can produce very complicated transduction groups. In this paper we will therefore focus on a class of simple binary invertible transducers first introduced in [19] that we refer to as *cycle-cum-chord transducers*, or CCC transducers for short. These transducers have state set $\{0, 1, \ldots, n-1\}$ and transitions

$$p \xrightarrow{a/a} p-1, \quad p > 0 \qquad \text{and} \qquad 0 \xrightarrow{0/1} n-1, \quad 0 \xrightarrow{1/0} m-1$$

where $1 \leq m \leq n$. We write $\mathfrak{A}^n_m$ for this transducer; the example $\mathfrak{A}^5_3$ is shown in figure 1. It is shown in [19] that the semigroups generated by CCC transducers are in fact free Abelian groups. The argument is based on a normal form for transductions proposed by Knuth. The normal form also allows one to define a natural geometry on $\mathbf{2}^\star$ that describes the orbits of words under a transduction f as affine subspaces, see section 2 below. As a consequence, it is polynomial-time decidable whether two transductions give rise to the same equivalence relation and we can construct the minimal transition system recognizing $f^\star$ in the sense of Eilenberg [3]. The reference identifies some CCC transducers where this minimal transition system is finite, so that the orbit relation is rational and thus automatic in the sense of [9, 11].

In this paper we will show that for some CCC transducers timestamps can be computed by a finite state machine; more precisely, there is a transducer that computes the witness t in reverse binary. For arbitrary CCC transducers our methods produce a quadratic time algorithm. It is known that there is a natural coordinate system for the level sets $\mathbf{2}^k$ in the full binary tree $\mathbf{2}^\star$, based on the elementary maps defined by the transducer, see [19]. We will show how to compute coordinates in quadratic time algorithm in general. As before, some CCC transducers admit faster algorithms and coordinates can be computed by a suitable transducer.

This paper is organized as follows. In section 2 we provide background information on invertible transducers, introduce cycle-cum-chord transducers and describe their basic properties, using Knuth normal form as a central tool. In the next section, we discuss the complexity of the timestamp and coordinate problem for CCC transducers. Lastly, section 4 contains comments on related decision problems and mentions open problems.

2 Transduction Groups and Iteration

We consider Mealy machines of the form $\mathcal{A} = \langle Q, \mathbf{2}, \delta, \lambda \rangle$ where Q is a finite set, $\mathbf{2} = \{0, 1\}$ is the input and output alphabet, $\delta : Q \times \mathbf{2} \to Q$ the transition function and $\lambda : Q \times \mathbf{2} \to \mathbf{2}$ the output function. We can think of $\mathbf{2}^\star$ as acting on Q via δ, see [2, 16, 10] for background. We are here only interested in *invertible transducers* where $\lambda(p, .) : \mathbf{2} \to \mathbf{2}$ is a permutation for each state p. When this permutation is the transposition in the symmetric group $\mathfrak{S}_2$ on two letters, we refer to p as a *toggle state*, and as a *copy state*, otherwise. Fixing a state p as initial state, we obtain a transduction $\mathcal{A}(p) : \mathbf{2}^\star \to \mathbf{2}^\star$ that is easily seen to be a length-preserving permutation. If the automaton is clear from context we write $\underline{p}$ for this function; $\mathcal{S}(\mathcal{A})$ denotes the semigroup generated by all the functions $\mathcal{A}(p)$ as p ranges over Q.

If we think of $\mathbf{2}^\star$ as an infinite, complete binary tree in the spirit of [17], we can interpret our transductions as automorphisms of this tree, see [13, 18]. Any automorphism f of $\mathbf{2}^\star$ can be written in the form $f = (f_0, f_1)s$ where $s \in \mathfrak{S}_2$: s describes the action of f on $\mathbf{2}$, and f_0 and f_1 are the automorphisms induced by f on the two subtrees of the root, which subtrees are naturally isomorphic to the whole tree. Write σ for the transposition in $\mathfrak{S}_2$. The automorphisms f such that $f = (f_0, f_1)\sigma$ are *odd*, the others *even*. In terms of wreath products the whole automorphism group can be written as

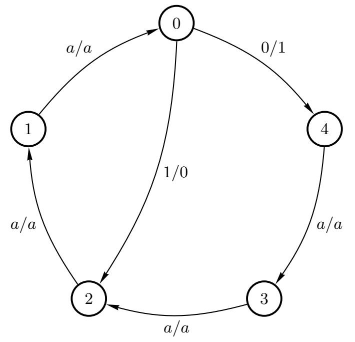
$$\mathsf{Aut}(\mathbf{2}^\star) \simeq \mathsf{Aut}(\mathbf{2}^\star) \wr \mathfrak{S}_2 = (\mathsf{Aut}(\mathbf{2}^\star) \times \mathsf{Aut}(\mathbf{2}^\star)) \rtimes \mathfrak{S}_2$$

with group operation $(f_0, f_1)s\ (g_0, g_1)t = (f_0 g_{s(0)}, f_1 g_{s(1)})\, st$, see [13, 18]. The collection $\mathfrak{I}$ of all maps defined by invertible transducers is easily seen to be closed under inverse and composition, and thus forms a subgroup $\mathfrak{I} \subseteq \mathsf{Aut}(\mathbf{2}^\star)$. For automata groups $G \subseteq \mathfrak{I}$ the wreath form naturally induces three maps ∂_0, ∂_1 and par such that $f = (\partial_0 f, \partial_1 f)\, \mathsf{par} f$. The parity is simply determined by the corresponding state being toggle or copy. The operations ∂_s are the *left residuals*, see [15, 5, 13]: for any word x, define the function $\partial_x f$ by $(x\, f)\,(z\, \partial_x f) = (xz)\, f$ for all words z (for transductions, we write function application on the right and use diagrammatic composition for consistency with relational composition). It follows that

$$\partial_{xy} f = \partial_y \partial_x f \qquad\qquad \partial_x (fg) = \partial_x f\, \partial_{xf}\, g$$

The transduction semigroup $\mathcal{S}(\mathcal{A})$ is naturally closed under residuals. In fact, we can describe the behavior of all the transductions by a transition system $\mathcal{C}$, much the way $\mathcal{A}$ describes the basic transductions: the states are all transductions in $\mathcal{S}(\mathcal{A})$ and the transitions are $f \xrightarrow{s/sf} \partial_s f$. Thus $\mathcal{C}$ contains $\mathcal{A}$ as a subautomaton. Of course, this system is infinite in general; it is referred to as the *complete automaton* in [13].

We will focus on a class of invertible transducers called *cycle-cum-chord transducers (CCC)*: their diagrams consist of a cycle plus one chord, the source of the chord is the only toggle state, all others are copy states. More precisely, a CCC transducer has state set $\{0, 1, \ldots, n-1\}$ and wreath representation $\underline{0} = (\underline{n^-}, \underline{m^-})\,\sigma$ and $\underline{k} = (\underline{k^-}, \underline{k^-})$ for $0 < k < n$. Here $1 \leq m \leq n$ and we occasionally write p^- rather than $p - 1$ to improve legibility. We will write $\mathfrak{A}^n_m$ for this transducer. The diagram of $\mathfrak{A}^5_3$ is shown in figure 1.



$$\underline{0} = (\underline{4}, \underline{2})\,\sigma$$
$$\underline{k} = (\underline{k^-}, \underline{k^-}) \quad 0 < k < 5$$

Fig. 1. The cycle-cum-chord transducer $\mathfrak{A}^5_3$ and its representation in the wreath form from section 2.

It was shown in [19] that the semigroups associated with these transducers are in fact free Abelian groups. More precisely, in the degenerate case $n = m$, $\mathfrak{A}^n_m$ generates the Boolean group $\mathbf{2}^n$. For $n > m$ let $s = \gcd(n, m)$, then $\mathfrak{A}^n_m$ generates the free Abelian group $\mathbb{Z}^{n-s}$. As explained in [19], one only needs to be concerned with the case when n and m are coprime. Hence, from now on, we assume $s = 1$. Commutativity of $\mathcal{S}(\mathfrak{A}^n_m)$ is easily seen. The reason we obtain a group is that the following *cancellation identity* holds in the semigroup:

$$\underline{0}^2\, \underline{1}^2 \ldots \underline{(m^-)}^2\ \underline{m}\ \underline{m+1} \ldots \underline{n^-} = I.$$

It follows from the cancellation identity that $\mathcal{S}(\mathfrak{A}^n_m)$ is a quotient of $\mathbb{Z}^{n-1}$. To establish isomorphism one can use the *Knuth Normal Form* of a transduction suggested in [12]. To this end, we extend $\mathfrak{A}^n_m$ to an infinite transducer with additional copy states k where $k \geq n$ and transitions $\underline{k} = (\underline{k^-}, \underline{k^-})$. This extension does not change the (semi)group generated by the machine because of the *shift identities* $\underline{k}^2 = \underline{k+m}\ \underline{k+n}$. One can then show that for every transduction f there is a unique flat representation

$$f = \underline{k_1}\, \underline{k_2}\ \ldots\ \underline{k_r},$$

where $k_1 < k_2 < \dots k_r$, the *Knuth normal form (KNF)* of f.

Thus we have two natural representations for transductions: the semigroup representation $f = \underline{0}^{e_0}\underline{1}^{e_1}\dots\underline{n-1}^{e_{n-1}}$ where $e_i \geq 0$, and the unique group representation $f = \underline{0}^{e'_0}\underline{1}^{e'_1}\dots\underline{n-2}^{e'_{n-s-1}}$ where $e'_i \in \mathbb{Z}$. Correspondingly, the *group representation* of f is the integer-valued vector $(e'_0, \dots, e'_{n-s-1})$. We will refer to $\sum |e_i|$ as the *weight* of f.

2.1 Rational Orbits

Given a transduction f we can think of the associated orbit relation $f^\star$ as a language over $(\mathbf{2}\times\mathbf{2})^\star$. One can then exploit the group representation to calculate Brzozowski quotients of this language. We obtain a generally infinite transition system that recognizes the orbits of f and whose states naturally are given by pairs of transductions, see [19] for details. Somewhat surprisingly, for some CCC transducers this transition system turns out to be finite for all the associated transductions. Thus, $f^\star$ is rational and hence automatic. For space reasons we focus here on the CCC Transducer $\mathfrak{A}^3_2$, see the reference for the following result and some generalizations.

Theorem 1. *For any transduction f in $\mathcal{S}(\mathfrak{A}^3_2)$, the orbit relation of f is rational. Accordingly, the root function can be computed by a length-preserving finite state transducer.*

This property is not shared by all CCC transducers; for example, the orbit relation of $\underline{0}$ in $\mathfrak{A}^4_3$ fails to be rational. It seems difficult to characterize CCC transducers with rational orbits.

For $\mathfrak{A}^3_2$, Knuth normal form has a number of interesting properties that will be important in section 3. Let $\mathsf{KNF}(f)$ denote the Knuth normal form of f. For any transduction f, write $\mathsf{sh}^s(f)$ for the transduction obtained by replacing any term $\underline{k}$ in the KNF of f by $\underline{k+s}$. In group representation, we have $\mathsf{sh}^1(a,b) = (-2b, a-2b)$. Lastly, let $\gamma_0 = \underline{0}$, $\gamma_1 = \underline{0}\,\underline{1}$, $\gamma_2 = \underline{0}^{-1}$ and $\gamma_3 = \underline{0}^{-1}\underline{1}^{-1}$ and set $\gamma'_i = \mathsf{sh}^1(\gamma_i)$.

Lemma 1. *Let $0 \leq k$ and $0 \leq i < 4$. Then $\mathsf{KNF}(\underline{0}^{2^{4k+i}}) = \mathsf{sh}^{8k+2i}(\gamma_i)$. More generally, for $f = \underline{0}^a\underline{1}^b$, we have $\mathsf{KNF}(f^{2^{4k+i}}) = \mathsf{sh}^{8k+2i}(\mathsf{KNF}(\gamma_i^a\,\gamma_i'^b))$.*

Proof. A straightforward computation shows that $\mathsf{KNF}(\underline{0}^{16}) = \underline{8}$ and it follows by induction that $\mathsf{KNF}(\underline{0}^{2^{4k}}) = \underline{8k}$ for all $k \geq 0$. But then $\mathsf{KNF}(\underline{0}^{2^{4k+1}}) = \mathsf{KNF}(\underline{8k}^2) = \underline{8k+2}\,\underline{8k+3} = \mathsf{sh}^{8k+2}(\gamma_1)$. The cases $i = 2,3$ are entirely similar. The second claim follows immediately from the first. $\square$

The existence proof for KNF is based on a weakly confluent rewrite system; other than a bound on the number of rewrite steps that is logarithmic in the weight of the given transduction there is no further information on the complexity computing the normal form. As it turns out, for $\mathfrak{A}_2^3$, rewriting is not required at all, a finite state transducer suffices to determine KNF in the following sense. For space reasons, let us focus on determining the KNF for $\underline{0}^t$ rather than the general group elements $\underline{0}^t\underline{1}^s$. Note that we can think of the KNF of f as an ω-sequence $\kappa \in \mathbf{2}^\omega$ where $\kappa_i = 1 \iff \underline{i}$ appears in the normal form of f. Since all but finitely many of the terms in κ are 0 we can also think of KNF as a finite bit-vector u such that $\kappa = u\,0^\omega$. We can pre-compute these finite bit-vectors of $\underline{0}^a$ for $0 \leq a < 16$ and pad to length 8 whenever necessary:

00000000	10000000	00110000	1011000	000010111	100010111
001110111	101110111	000000111	100000111	001100111	101100111
000010001	100010001	001110001	101110001		

All but the first 4 entries have length 9 and require a "carry" to the next block. According to lemma 1 we can now determine KNF of $\underline{0}^t$ as follows. Let T be a 0-indexed table whose entries are the 16 KNFs, right-padded or truncated to form blocks of length 8. If there is no carry, on input hex-digit d the correct output is T_d, but with a carry it is $T_{d+1 \bmod 16}$. Figure 2 shows a sketch of the appropriate transducer; input is hexadecimal, output is binary in blocks of 8 bits.

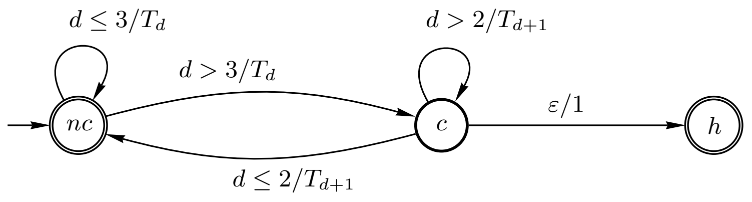


Fig. 2. A transducer that determines the Knuth normal form of a transduction $\underline{0}^a$ for CCC transducer $\mathfrak{A}_2^3$.

The state nc is the no-carry state, c is carry, and h takes care of pending carries after the last input digit. For example, for $a = 3921 = (15F)_{16r}$ we get three blocks plus one 1 because of the carry:

$$T_1T_5T_0T_1 = 10000000\ 10001011\ 00000000\ 1,$$

corresponding to KNF $\underline{0}\ \underline{8}\ \underline{12}\ \underline{14}\ \underline{15}\ \underline{24}$. Note that the KNF transducer can be converted into a recurrence equation for the length of $\mathsf{KNF}(f)$, but it seems difficult to obtain a closed form solution. Also, a similar construction works for general group elements, but the machinery becomes considerably more complicated since we now have to deal with both generators $\underline{0}$ and $\underline{1}$ of the transduction group.

2.2 Computing Iterates

Knuth normal form also suggests that $x\,f^t$ can be computed easily: we have $az\,f = a\,f(z\,\partial_a f)$ and residuation for a transduction written in KNF comes down to a left shift, except possibly for a first term $\underline{0}$. Hence, after processing an initial segment of x, the residuals of f^t will have low weight and, from then on, every single bit of x can be processed in constant time. In terms of the complete automaton $\mathcal{C}$ from section 2 this means that there are only a few non-trivial strongly connected components and every sufficiently long path winds up in one of them. For example, in the case of $\mathfrak{A}^3_2$ the complete automaton has 8 non-trivial strongly connected components the largest of which as 6 states. Technically we have the following results.

Proposition 1. *Given a transduction $f \in \mathcal{S}(\mathfrak{A}^n_m)$ of weight w we can compute its residuals in time $O(n \log w)$.*

Proof. Let $f = (e_0, \ldots, e_{n-})$ be the semigroup representation where $\sum e_i = w$. Then the semigroup representations of the residuals are

$$\partial_0 f = (e_1, e_2, \ldots, e_m + \lfloor e_0/2 \rfloor, \ldots, e_{n-1}, \lceil e_0/2 \rceil)$$
$$\partial_1 f = (e_1, e_2, \ldots, e_m + \lceil e_0/2 \rceil, \ldots, e_{n-1}, \lfloor e_0/2 \rfloor)$$

An entirely similar argument applies to group representation as well. Our claim follows. □

It follows that $x\,f$ can be computed in $O(|x|\,n \log w)$ time. However, we can do better than that.

Proposition 2. *Given a transduction $f \in \mathcal{S}(\mathfrak{A}^n_m)$ we can compute $x\,f$ in time linear in $|x|$, with coefficients depending on f.*

Proof. Suppose f has group representation $(e_0, e_1, \ldots, e_{n-2})$ so that $w = \sum |e_i|$. The first bit of $az\,f$ depends only on the parity of e_0. As we have seen in proposition 1, we can compute the residuals of f in $O(n \,\log w)$ steps, and these residuals have weight at most w. Moreover, we can express residuation as an affine operation of the form

$$\partial_s \boldsymbol{u} = \begin{cases} A \cdot \boldsymbol{u} & \text{if } \boldsymbol{u} \text{ is even,} \\ A \cdot \boldsymbol{u} - (-1)^s \boldsymbol{a} & \text{otherwise.} \end{cases}$$

where $\boldsymbol{u} \in \mathbb{Z}^{n-1}$ is the group representation of the transduction, see [14]. The spectral radius of A is less than 1, hence residuation is a contraction and after a transient part all weights are bounded by a constant depending only on n and m. Since the length of the transient is independent of x our claim follows. □

We do not know how to obtain more precise bounds on the cost of computing $x\,f$. In particular there appears to be no easy way to determine the number and size of the non-trivial strongly connected components of the complete automaton, short of actual computation.

3 Timestamps and Coordinates

One can show that for any CCC transducer $\mathfrak{A}^n_m$ the group H of transductions generated by $\underline{p}$, $0 \leq p < m$, acts transitively on $\mathbf{2}^\ell$ (which set of words is often referred to as a level set in connection with the infinite binary tree). For $\ell = km$, the quotient group H' obtained by factoring with respect to $\underline{i}^{2^k}$ acts simply transitively on the level set $\mathbf{2}^\ell$. As a consequence, there is a natural coordinate system for $\mathbf{2}^{km}$: for every $\ell = km$ there is a bijection

$$\mathbf{2}^\ell \to \mathbb{Z}/(2^k) \times \ldots \times \mathbb{Z}/(2^k)$$

where the product on the right has m terms. We will write $\langle\, w \,\rangle_\ell \in (\mathbb{Z}/(2^k))^m$ for the *coordinates* of a word w: $\langle\, w \,\rangle_\ell = (a_0, \ldots, a_{m^-})$ if, and only if, $w = 0^\ell\, \underline{0}^{a_0} \underline{1}^{a_1} \ldots \underline{m^-}^{a_{m^-}}$. We use $x \equiv y$ to express the fact that two integer vectors of length m are componentwise congruent modulo 2^k. Also, for a transduction f, define the ℓ-coordinates of f by $\langle\, f \,\rangle_\ell = \left\langle\, 0^\ell\, f \,\right\rangle_\ell$. For example, in $\mathfrak{A}^3_2$, letting $f = \underline{0}^{-1}\underline{1}^3$ we get $\langle\, f \,\rangle_{2k} = (2^k - 1, 3)$ for $k \geq 2$. By commutativity it follows that $\left\langle\, 0^\ell\, f^i \,\right\rangle_\ell \equiv i \cdot \langle\, f \,\rangle_\ell$ and $\left\langle\, 0^\ell\, f^\star \,\right\rangle_\ell \equiv \mathbb{N} \cdot \langle\, f \,\rangle_\ell$, so that the orbit of 0^ℓ is a linear subspace of $(\mathbb{Z}/(2^k))^m$. Again by commutativity general orbits can be described as affine subspaces of $(\mathbb{Z}/(2^k))^m$:

$$\langle\, w\, f^\star \,\rangle_\ell \equiv \langle\, w \,\rangle_\ell + \mathbb{N} \cdot \langle\, f \,\rangle_\ell$$

Thus, it is of interest to be able to calculate coordinates. More formally, we wish to address the following problem, assuming a CCC transducer $\mathfrak{A}^n_m$ is fixed.

Problem:	**Coordinate Problem**
Instance:	A word $x \in \mathbf{2}^\ell$ where $\ell = km$.
Output:	The coordinates $\langle\, x \,\rangle_\ell \in (2^k)^m$ of x.

Closely related is the question how many times a given transduction f must be applied to obtain a particular point in the orbit of a given word x. We refer to this as the Timestamp Problem:

Problem:	**Timestamp Problem**
Instance:	A transduction f, two words $x, y \in \mathbf{2}^k$.
Output:	The least $t \geq 0$ such that $y = x\, f^t$, if it exists; `NO` otherwise.

Clearly the Orbit Problem reduces to the Timestamp Problem, which, as we will see shortly, in turn reduces to the Coordinate Problem. We will show that all of them can be solved in quadratic time. Let us first deal with the Timestamp Problem.

Theorem 2. *The Timestamp Problem can be solved in quadratic time: given two words x and y of length $\ell = km$ and a transduction $f \in \mathcal{S}(\mathfrak{A}^n_m)$ we can find a timestamp $t \geq 0$ such that $x\, f^t = y$, or determine that no such t exists, in $O(\ell^2)$ steps.*

Proof. We may safely assume that $m < n$ and $\gcd(n, m) = 1$. Also, we only need to consider the case where f is odd, since otherwise f simply copies the first k_1 bits in the input word, where $\underline{k_1}$ is the first term in the KNF of f. Write $x = x_0 \ldots x_{\ell-1}$ and $y = y_0 \ldots y_{\ell-1}$ and let $f = (e_0, \ldots, e_d) \in \mathbb{Z}^d$ in group representation where $d = n - 2$. We will determine the bits in the binary expansion of the timestamp $t = \sum t_i 2^i$, starting with the least significant digit. Initialize a symbolic vector

$$V = \sum_{i<k} t_i 2^i \, (e_0, \ldots, e_d).$$

with entries in the polynomial ring $\mathbb{Z}[\boldsymbol{\tau}]$, $\boldsymbol{\tau} = t_0, \ldots, t_{k-1}$. We proceed in k rounds $r = 0, \ldots, k-1$, each consisting of m stages $s = 0, \ldots, m-1$. In round r, stage s, perform the following actions. If $s = 0$, bind t_r to 0 or 1 so as to satisfy $V_1 + x_{mr} = y_{mr} \pmod 2$ where V_1 is the first component of V. If $s > 0$, check that $V_1 + x_{mr+s} = y_{mr+s} \pmod 2$ and exit returning `NO` if the test fails. In either case, finish the stage by replacing V by $\partial_{x_{mr+s}} V$. If all rounds complete successfully, return the required timestamp $t = \sum t_i 2^i$.

To see that the algorithm is correct, write $x[i]$ for the prefix of x of length i. Induction shows that if the algorithm has not returned `NO` by the end of round r, state s, then, letting $p = \sum_{i \leq r} t_i \, 2^i \in \mathbb{N}$, we have

$$x[mr + s + 1] \, f^p = y[mr + s + 1] \qquad \text{and} \qquad V = \partial_{x[mr+s+1]} f^{p+\pi}$$

where π is a polynomial in variables $t_{r+1}, \ldots, t_{k-1}$. The binding of t_r at stage $s = 0$ of round r always exists since the transduction represented by V is guaranteed to be odd by lemma 14 in [19].

First assume that indeed $x \, f^t = y$ for some $0 \leq t < 2^k$. But then the algorithm correctly determines the timestamp t: In each round another binary digit of t is determined. Given the bindings for t, V initially represents the transduction f^t, and represents the appropriate quotients during execution, so that all the tests succeed. If, on the other hand, y is not in the orbit of x, consider the longest prefix x' of x such that $x' \, f^p = y'$ for the corresponding prefix y' of y and some $p \geq 0$. Writing the length of x' as $mr + s$, the algorithm will run up to round r, stage s. However, at the next stage a mismatch will be found and the algorithm returns `NO`, as required.

As to the time complexity of the algorithm, note that all numerical coefficients have at most k bits. Since the polynomials have at most k terms, a brute-force implementation will require time cubic in ℓ. However, all entries in V during round r are of the form $c + d \sum_{i \geq r} t_i 2^{i-r}$ where $c, d \in \mathbb{N}$. Thus, there is not need to maintain the full polynomials during the computation. As a consequence, quadratic time suffices. $\square$

Here is an example that shows how the computation unfolds for $\mathfrak{A}^3_2$, $f = \underline{0}$ and $x = 101110010010$, $y = 001000011000$. In this case $y = x \, f^{43}$. There are 6 rounds with 2 stages each.

r	t_i	V_1	V_2
		$t_0 + 2t_1 + 4t_2 + 8t_3 + 16t_4 + 32t_5$	0
0	$t_0 = 1$	$-2t_1 - 4t_2 - 8t_3 - 16t_4 - 32t_5$	$1 - t_1 - 2t_2 - 4t_3 - 8t_4 - 16t_5$
		$1 + t_1 + 2t_2 + 4t_3 + 8t_4 + 16t_5$	$t_1 + 2t_2 + 4t_3 + 8t_4 + 16t_5$
1	$t_1 = 1$	-1	$-1 - t_2 - 2t_3 - 4t_4 - 8t_5$
		$1 - t_2 - 2t_3 - 4t_4 - 8t_5$	2
2	$t_2 = 0$	$2 + 2t_3 + 4t_4 + 8t_5$	$1 + t_3 + 2t_4 + 4t_5$
		$-1 - t_3 - 2t_4 - 4t_5$	$-1 - t_3 - 2t_4 - 4t_5$
3	$t_3 = 1$	0	$1 + t_4 + 2t_5$
		$1 + t_4 + 2t_5$	0
4	$t_4 = 0$	$-2 - 2t_5$	$-2 - t_5$
		t_5	$1 + t_5$
5	$t_5 = 1$	2	1
		-1	-1

The technique of the last theorem can be pushed slightly to provide a fast algorithm to compute coordinates. Suppose $x \in \mathbf{2}^\ell$ where $\ell = km$. We need to compute integers $e_0, \dots, e_{m^-}$ such that

$$x = 0^\ell\, \underline{0}^{e_0} \dots \underline{m^-}^{e_{m^-}}.$$

Let us call the transduction on the right f. Then for any $r < \ell$

$$x = 0^r\, f \cdot 0^{\ell - r} \partial_{0^r} f.$$

Since the first bit of $0^{\ell-r}\partial_{0^r} f$ depends only on the parity of $\partial_{0^r} f$ we can determine the coefficients of the binary expansions of the exponents e_i.

Theorem 3. *The Coordinate Problem can be solved in quadratic time: given a word x of length $\ell = km$ we can determine its coordinates in $O(\ell^2)$ steps.*

Proof. Let $x = x_0 \dots x_{\ell 1 - 1}$ and write the coordinates of x as $e_i = \sum_{j<k} t_{i,j} 2^j$. Initialize a vector

$$V = (\sum_{j<k} t_{0,j} 2^j, \dots, \sum_{j<k} t_{m^-,j} 2^j)$$

of linear polynomials, this time over the polynomial ring $\mathbb{Z}[\boldsymbol{\tau}]$ where $\boldsymbol{\tau} = (t_{i,j} \mid i, j)$. Again we proceed in k rounds $r = 0, \dots, k-1$, each consisting of m stages $s = 0, \dots, m-1$. In round r, stage s, perform the following actions. Bind $t_{s,r}$ to 0 or 1 so as to make sure that $V_1 = x_{mr+s} \pmod 2$ where V_1 is the first component of V. Then replace V by $\partial_0 V$ where we interpret the arithmetic operations involved in ∂_0 in the obvious way on V.

Correctness and running time analysis are entirely analogous to the argument in the preceding theorem. □

Given the algorithm for the Coordinate Problem one can also tackle the Timestamp Problem via a reduction.

Proposition 3. *The Timestamp Problem reduces to the Coordinate Problem in time $O(\ell \log w + \log^2 k)$ where w is the weight of the transduction and km is the length of the words.*

Proof. We are given a transduction f and words x, y. We may safely assume that the given words have length $\ell = km$, otherwise we can simply pad and ignore some of the linear equations below. We need to find the least t such that $x\, f^t = y$. As we have seen,

$$\langle\, x\, f^t \,\rangle_\ell \equiv \langle\, x \,\rangle_\ell + t \cdot \langle\, f \,\rangle_\ell \,.$$

Thus, given the ℓ-coordinates of x, y and f we can simply solve a system of modular equations. The computation of $0^\ell f$ takes $O(\ell \log w)$ steps where w is the weight of f. The linear system has m equations and all coefficients have at most k bits. □

For some CCC transducers the quadratic bounds from the last few results can be improved upon: finite state machines sometimes suffice to calculate coordinates and timestamps. As an example, consider again $\mathfrak{A}_2^3$. The following algorithm solves the Coordinate Problem in this case. Given a word x (here assumed to be 0-indexed) we calculate its coordinates in reverse binary as follows. The γ_i are as in section 2.1.

```
// coordinate algorithm for CCC transducer $\mathfrak{A}_2^3$
    $h = (0, 0)$;
    **for** $r = 0, \ldots, n-1$ **do**
        $s_r = h_1 + x_{2r} \bmod 2$;              // phase 1: bind $s_r$
        $h = \partial_0(h + s_r \cdot \gamma_r)$;
        $t_r = h_1 + x_{2r+1} \bmod 2$;            // phase 2: bind $t_r$
        $h = \partial_0(h + t_r \cdot \gamma_r)$;
    **return** $(s, t)$;
```

As stated, the algorithm appears to require quadratic time. However, it can be implemented on a finite state machine because of the contraction property of residuals spelled out in section 2.

Theorem 4. *The Coordinate Problem for $\mathfrak{A}_2^3$ can be solved by a transducer that computes the coordinates in reverse binary.*

Proof. Given a word x of length $2n$ the algorithm determines a transduction $f = \underline{0}^s\, \underline{1}^t$ where $0 \leq s, t < 2^n$. We will show by induction on n that $f(0^{|x|}) = x$ and $\partial_{0^{|x|}} f = h$. We only present the step from length $8n$ to $8n + 2$ during one round of the algorithm, the other cases are entirely similar and will be omitted. During a particular round we denote s', f' and h' the new values of s, f and h after the first phase in the execution of the algorithm, and t'', f'' and h'' for the second phase. Write $\mathbf{0}$ for 0^{8n} and consider an extension $u = x\, ab$ of x. The following argument relies on lemma 1.

In phase 1, if $\mathbf{0}0\,f = xa$ then $f' = f$ and we have $\mathbf{0}0\,f' = \mathbf{0}0\,f = xa$. Also, $\partial_{\mathbf{0}0} f' = \partial_0 \partial_{\mathbf{0}} f = \partial_0 h = h'$. Otherwise $s' = s + 2^{4n}$ and $f' = f\,\underline{0}^{2^{4n}}$. Then $\mathbf{0}0\,f' = \mathbf{0}0\,f\underline{0}^{2^{4n}} = x\overline{a}\,\underline{0}^{2^{4n}} = x\,(\overline{a})\,\underline{0} = xa$. Furthermore, $\partial_{\mathbf{0}0} f' = \partial_0(\partial_{\mathbf{0}} f\underline{0}^{2^{4n}}) = \partial_0(h\partial_x\underline{0}^{2^{4n}}) = h'$ by lemma 1.

For phase 2 first consider the case $f'(\mathbf{0}00) = xab$. Then $t'' = t$ and we have $\mathbf{0}00\,f'' = \mathbf{0}00\,f' = xab$. Also, $\partial_{\mathbf{0}00} f'' = \partial_0(\partial_{\mathbf{0}0} f') = \partial_0 h' = h''$. In the remaining case $t'' = t + 2^{4n}$ and $f'' = f'\,\underline{1}^{2^{4n}}$. Then $\mathbf{0}00\,f'' = \mathbf{0}00\,f'\underline{1}^{2^{4n}} = xa\bar{b}\,\underline{1}^{2^{4n}} = x\,(a\bar{b})\,\underline{1} = xa(\bar{b})\underline{0} = xab$. Furthermore, $\partial_{\mathbf{0}00} f'' = \partial_0(\partial_{\mathbf{0}0} f'\underline{1}^{2^{4n}}) = \partial_0(h'\partial_{xa}\underline{1}^{2^{4n}}) = h''$, again by the lemma. $\square$

It is straightforward to modify this algorithm to deal with timestamps.

4 Open Problems

We have characterized the complexity of various computational problems associated with a class of invertible binary transducers that relate to iteration of transductions. Specifically, we have shown that for a cycle-cum-chord transducers iterates, time-stamps and coordinates can be computed quickly. The Knuth normal form of a transduction is a critical technical device in all these arguments. We do not know in general when Knuth normal form can be computed by a finite state transducer as in section 2.1. It appears that this property is quite rare but we are currently unable to characterize the corresponding CCC transducers. The situation is similar with respect to the rationality of the orbit relation; some cases are discussed in [19] but no general characterization exists. In fact, we do not even know whether orbit rationality is decidable for CCC transducers, much less for arbitrary invertible transducers, even when the number of toggle states is restricted to just one.

As already pointed out, our CCC transducers generate free Abelian groups. Of course, there are other automata that also generate these groups. One well-known example are the so-called "sausage automata" in [13], given in wreath notation by $\underline{0} = (I, \underline{n})\,\sigma$ and $\underline{k} = (\underline{k-1}, \underline{k-1})$ for $2 \leq k \leq n$. Here we ignore the identity I, as customary. In fact, [14] contains a detailed characterization of invertible automata associated with free Abelian groups. It is natural to ask whether and to what degree our results carry over to these automata.

It is straightforward to check whether $\mathcal{S}(\mathcal{A})$ is commutative, using standard automata-theoretic methods. We do not know whether it is decidable whether $\mathcal{S}(\mathcal{A})$ is a group, though this property is obviously semidecidable. Unsurprisingly, many other decidability questions regarding transduction semigroups or groups of invertible transducers are also open, see [7, chap. 7] for an extensive list.

References

1. L. Bartholdi and P. V. Silva. Groups defined by automata. *CoRR*, abs/1012.1531, 2010.

2. J. Berstel. Transductions and context-free languages. `http://www-igm.univ-mlv.fr/~berstel/LivreTransductions/LivreTransductions.html`, 2009.
3. S. Eilenberg. *Automata, Languages and Machines*, volume A. Academic Press, 1974.
4. L. Fortnow and J. A. Grochow. Complexity classes of equivalence problems revisited. *Inf. Comput.*, 209(4):748–763, April 2011.
5. V. M. Gluškov. Abstract theory of automata. *Uspehi Mat. Nauk*, 16(5(101)):3–62, 1961.
6. R. Grigorchuk and Z. Šunić. *Groups St. Andrews 2005*, volume 339 of *London Math. Soc. Lec. Notes*, chapter Self-Similarity and Branching in Group Theory. Cambridge University Press, 2007.
7. R. R. Grigorchuk, V. V. Nekrashevich, and V. I. Sushchanski. Automata, dynamical systems and groups. *Proc. Steklov Institute of Math.*, 231:128–203, 2000.
8. J. Howard Johnson. Rational equivalence relations. *Theoretical Computer Science*, 47:167–176, 1986.
9. B. Khoussainov and A. Nerode. Automatic presentations of structures. In *LCC '94: Int. Workshop on Logical and Computational Complexity*, pages 367–392, London, UK, 1995. Springer-Verlag.
10. B. Khoussainov and A. Nerode. *Automata Theory and its Applications.* Birkhäuser, 2001.
11. B. Khoussainov and S. Rubin. Automatic structures: overview and future directions. *J. Autom. Lang. Comb.*, 8(2):287–301, 2003.
12. D. Knuth. Private communication, 2010.
13. V. Nekrashevych. *Self-Similar Groups*, volume 117 of *Math. Surveys and Monographs.* AMS, 2005.
14. V. Nekrashevych and S. Sidki. *Automorphisms of the binary tree: state-closed subgroups and dynamics of 1/2-endomorphisms.* Cambridge University Press, 2004.
15. G. N. Raney. Sequential functions. *J. Assoc. Comp. Mach.*, 5(2):177–180, 1958.
16. J. Sakarovitch. *Elements of Automata Theory.* Cambridge University Press, 2009.
17. J.-P. Serre. *Arbres, Amalgames, SL_2.* Number 46 in Astérisque. Société Mathématique de France, Paris, 1977.
18. S. Sidki. Automorphisms of one-rooted trees: Growth, circuit structure, and acyclicity. *J. Math. Sciences*, 100(1):1925–1943, 2000.
19. K. Sutner and K. Lewi. Iterating invertible binary transducers. In M. Kutrib, N. Moreira, and R. Reis, editors, *Descriptional Complexity of Formal Systems*, volume 7386 of *Lecture Notes in Computer Science*, pages 294–306. Springer Berlin, 2012.