Flipping Pebbles

K. Sutner
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

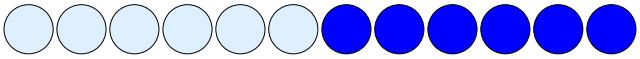
We study a simple, 3-state Mealy automaton and analyze the group of transductions defined by this automaton. In particular it is demonstrated that iterating the transductions produces rational equivalence relations. A critical ingredient in this study is a special type of normal form for transductions suggested by Knuth.

1 A Pebble Game

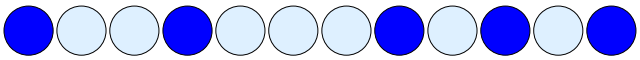
Suppose you have a sequence of pebbles, blue on one side and white on the other. Starting at the leftmost pebble, flip (some of) them over according to the following rule:

> *Flip the current pebble. If it is now white-side up, skip over the next one. Otherwise, skip over the next two pebbles. Repeat till you fall off the end.*

Let us call this the *toggle operation*. Thus, the toggle operation turns the following pebble sequence



into



Admittedly, it is a bit of a stretch to refer to this operation as a game. But, as in Conway's Game-of-Life [5], interesting questions arise when we iterate. Figure 1 shows the result of applying our rule repeatedly to a sequence of 20 pebbles. Note that the behavior of the first few pebbles is simply periodic, but more complicated patterns emerge further down the line.

We are trying to understand exactly how complicated these orbits are. A moment's thought reveals that the toggle operation is reversible, so the orbit of any sequence must be a cycle. A few computational experiments suggest that the length of the cycle associated with a

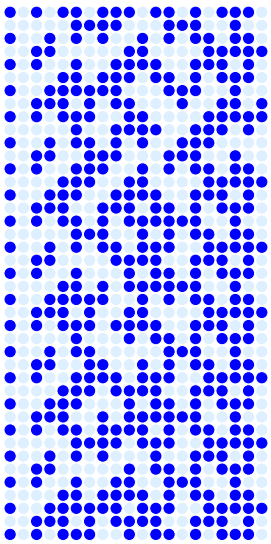


Figure 1: Repeatedly flipping a sequence of 20 pebbles.

sequence of length n is about $2^{n/2}$ and thus has exponential length. This leads to a natural computational question that we will refer to as the *Orbit Problem*: given two sequences of pebbles, how does one check whether one appears in the orbit of the other? Clearly this question can be answered by brute-force enumeration, but we are interested in better solutions, preferably ones whose running times are polynomial in the length of the sequences in question.

A closely related question is the *Timestamp Problem*: given two sequences in the same orbit, how many applications of the toggle operation are needed to get from one to the other? Going in the opposite direction we can ask which sequence is generated by applying our operation t times to a given sequence; we refer to this question as the *Iteration Problem*. Again, we do not wish to resort to brute-force enumeration, we are looking for fast algorithms.

First, let us formalize the problem slightly. The toggle operation is best described in terms of a *Mealy automaton*, a finite state machine that translates input words over the binary alphabet into output words. The machine $\mathcal{A}$ has 3 states and is shown in figure 2.

Here $a \in \mathbf{2} = \{0, 1\}$. There is no fixed initial state; we can define three maps $\underline{0}$, $\underline{1}$ and $\underline{2}$ on binary words, the *basic transductions*, by selecting the corresponding state to be initial. These devices are called output modules in Eilenberg [3]. See also [13, 1] and the classic [4] for background. Thus, in state 0 the transducer outputs the complement of the next input bit, in states 1 and 2 it simply copies the next input bit to output. Naming the states 0, 1 and 2 may seem a bit uninspired, but will turn out to be very helpful in a moment. Note that our transductions are length-preserving bijections on finite words.

We write $\mathsf{orb}(u; f)$ for the orbit of word u under transduction f. As a first step towards understanding the structure of these orbits, consider an arbitrary cycle $u_0, u_1, \ldots, u_{n-1}$ under some transduction f. Extending u_0 by one bit on the right either yields two disjoint cycles $u_0 0, u_1 b_1, \ldots, u_{n-1} b_{n-1}$ and $u_0 1, u_1 \overline{b_1}, \ldots, u_{n-1} \overline{b_{n-1}}$, or a single cycle of the form $u_0 0, u_1 b_1 \ldots, u_{n-1} b_{n-1}, u_0 1, u_1 \overline{b_1} \ldots, u_{n-1} \overline{b_{n-1}}$. We will call u_0 *splitting* or *doubling*, respec-

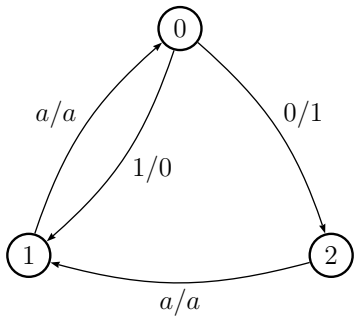


Figure 2: A 3-state invertible transducer.

tively. The next observation follows immediately.

Proposition 1.1 *For any transduction f, all orbits under f have length a power of 2.*

The collection of all orbits can be nicely represented by a tree. To this end, define the *root* of an orbit to be the lexicographically least element of the orbit, which element we will denote $\mathsf{root}(u)$. Just like our transductions, the root function is length-preserving and prefix-preserving: $\mathsf{root}(x)$ is a prefix of $\mathsf{root}(xy)$ for all words x and y. In the terminology of [8], root is the first canonical form of the equivalence relation $f^\star$. We refer to $\mathsf{root}(\mathbf{2}^\star)$ as the *root language* of f. We can organize all cycles under f into an ordered binary tree $\mathcal{T}_f$ whose nodes are the words of the root language. It is convenient to start at the fixed point ε, the empty word. Node u has a left child $u0$ and a right child $u1$ whenever u splits, and a single child $u0$ otherwise. The initial part of the orbit tree for $\underline{0}$ is shown in figure 3.

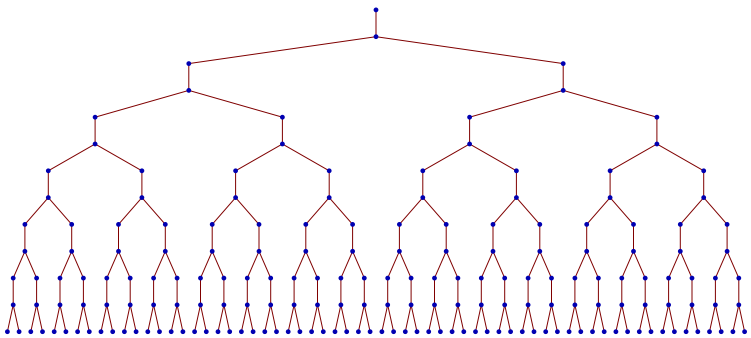


Figure 3: The first 12 levels of the orbit tree of transduction $\underline{0}$.

The picture immediately suggests that the orbit tree is *homogeneous* in the sense that all nodes at level n either double or split, for all n. All homogeneous trees can be represented by the *type* of the tree, an infinite sequence over $\{1, 2\}$ that indicates the out-degree at all levels. In this case, the sequence appears to be periodic.

Conjecture 1.1 *The orbit tree of* $\underline{0}$ *is homogeneous and has type* $(12)^\omega$.

It is a good exercise to determine the orbit trees for the transductions $\underline{1}$ and $\underline{2}$. If true, our conjecture has lots of consequences for the orbits of $\underline{0}$. All words of length $2n$ and $2n-1$ will have orbits of length 2^n. The root language will consist of all prefixes of $(0\mathbf{2})^\star$. Recall that for two words $a = a_1, \ldots, a_n$ and $b = b_1, \ldots, b_n$ the *shuffle product* $a \parallel b$ is defined to be the word $a_1 b_1 a_2 b_2 \ldots a_n b_n$ obtained by interleaving the letters from a and b in a strictly alternating fashion. Then the root of any orbit of an even length word will be the uniquely determined word of the form $0^n \parallel b$ in that orbit. In fact, for every word a of length n there will be exactly one word b of length n such that $a \parallel b$ lies in any chosen orbit in $\mathbf{2}^{2n}$. In other words, every orbit of transduction $\underline{0}$ of an even-length word would have the form $\{\, u \parallel s(u) \mid u \in \mathbf{2}^n \,\}$ for some word map s.

Since we are interested in iteration, it seems natural to consider the semigroup $\mathcal{S}$ generated by $\underline{0}$, $\underline{1}$ and $\underline{2}$. We will refer to the elements of this semigroup as *(composite) transductions*. Naturally, there are generalized versions of the decision problems we mentioned previously, where $\underline{0}$ is replaced by an arbitrary composite transduction $f \in \mathcal{S}$. Note that we can build transducers for all the elements of $\mathcal{S}$ using a standard product construction. As we will see shortly, $\mathcal{S}$ turns out to be a commutative group. This can be proven in a purely computational fashion by constructing the corresponding machines and checking for isomorphisms.

A more attractive alternative is to use algebra: we can think of our transductions as automorphisms of the rooted, infinite, binary tree $\mathbf{2}^\star$: this tree has all binary words as nodes, the empty word ε is the root, and there are edges $x \to x0$ and $x \to x1$ for every word x. It is easy to see that our transductions are indeed automorphisms of this tree and we can think of $\mathcal{S}$ as being contained in the full automorphism group $\mathsf{Aut}(\mathbf{2}^\star)$, see [14] for a detailed discussion.

Note that an automorphism of $\mathbf{2}^\star$ can be represented by labeling the nodes of the tree with permutations from $\mathfrak{S}_2$, the symmetric group on two letters. Hence we can write a transduction f in the form $f = (f_0, f_1)s$ where $s \in \mathfrak{S}_2$: s acts on the two subtrees of the root of the tree, and f_0 and f_1 are automorphisms on these subtrees. The group operation is given by

$$(f_0, f_1)s\,(g_0, g_1)t = (f_0 g_{s(0)}, f_1 g_{s(1)})\,st$$

We write function composition in diagrammatic order, $(fg)(x) = g(f(x))$ to be compatible wit relational composition. The components f_i arise naturally as the *residuals* of f. More precisely, for any word x, define the function $\partial_x f$ by $f(x)\,\partial_x f(z) = f(xz)$. Note that $\partial_x fg = \partial_x f\,\partial_{f(x)} g$. It follows that the transduction semigroup $\mathcal{S}$ is closed under residuals. We write σ for the transposition in $\mathfrak{S}_2$ and omit the identity. Call f *even* if $f(a) = a$ and *odd* otherwise. Then any odd transduction has the wreath form $f = (f_0, f_1)\,\sigma$ and $f(ax) = \overline{a}\, f_a(x)$. For even f we have $f = (f_0, f_1)$ and $f(ax) = a\, f_a(x)$. In terms of wreath products, the full automorphism group of $\mathbf{2}^\star$ can be written as

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

see [11] for details. We are particularly interested in the sub-semigroup generated by the basic transductions in $\mathcal{A}$:

$$\underline{0} = (\underline{2}, \underline{1})\,\sigma \qquad \underline{1} = (\underline{0}, \underline{0}) \qquad \underline{2} = (\underline{1}, \underline{1})$$

Lemma 1.1 *The transduction semigroup of $\mathcal{A}$ is a commutative group.*

Proof. We have

$$\begin{aligned}
\underline{0}\,\underline{i+1} &= (\underline{2},\underline{1})\,\sigma\,(\underline{i},\underline{i}) = (\underline{2}\,\underline{i},\underline{1}\,\underline{i})\,\sigma \\
\underline{i+1}\,\underline{0} &= (\underline{i},\underline{i})\,(\underline{2},\underline{1})\,\sigma = (\underline{i}\,\underline{2},\underline{i}\,\underline{1})\,\sigma \\
\underline{1}\,\underline{2} &= (\underline{0},\underline{0})\,(\underline{1},\underline{1}) \;\; = (\underline{0}\,\underline{1},\underline{0}\,\underline{1}) \\
\underline{2}\,\underline{1} &= (\underline{1},\underline{1})\,(\underline{0},\underline{0}) \;\; = (\underline{1}\,\underline{0},\underline{1}\,\underline{0})
\end{aligned}$$

where $i = 0, 1$; done by induction. A similar computation shows that

$$\underline{0}^2\underline{1}^2\underline{2} = (\underline{0}^2\underline{1}^2\underline{2}, \underline{0}^2\underline{1}^2\underline{2})$$

so that $\underline{0}^2\underline{1}^2\underline{2}$ is the identity. Thus the inverses of the elements of $\mathcal{S}$ are already contained in $\mathcal{S}$, there is no need to change the transducer. It follows that $\mathcal{S}$ is a commutative subgroup of $\mathsf{Aut}(\mathbf{2}^\star)$. □

2 Knuth Normal Form

The collection of all even transductions forms a subgroup H of index 2. From the identity $\underline{0}^2\underline{1}^2\underline{2} = I$ it follows that $\mathcal{S}$ must be a quotient of $\mathbb{Z}^2$, the free Abelian group of rank 2. It seems difficult to find other identities, so one is lead to conjecture that $\mathcal{S}$ is in fact isomorphic to $\mathbb{Z}^2$. Here is a proof suggested by D. Knuth [10] that uses a clever infinite extension of $\mathcal{A}$. The wreath forms $\underline{0} = (\underline{2},\underline{1})\,\sigma$, $\underline{1} = (\underline{0},\underline{0})$, $\underline{2} = (\underline{1},\underline{1})$ of our transductions make it tempting to continue the list with $\underline{3} = (\underline{2},\underline{2})$. Of course, there is no state 3 in the transducer–but there is a corresponding group element $\underline{3} = (\underline{2},\underline{2})$ in the ambient group $\mathsf{Aut}(\mathbf{2}^\star)$. To wit, in $\mathcal{S}$ we have $\underline{0}^2\underline{2}^{-1} = \underline{0}^4\underline{1}^2 = (\underline{2},\underline{2})$. More generally, define $\Delta : H \to \mathcal{S}$ by $H(f) = \partial_0 f$. For all $f \in H$ we have $\partial_1 f = \partial_0 f = \Delta(f)$; moreover, $\underline{3} = \Delta^{-1}(\underline{2})$. It is easy to check that Δ is a group isomorphism that can be used to continue this extension process indefinitely and we obtain the extended machine $\mathcal{K}$ shown in figure 4.

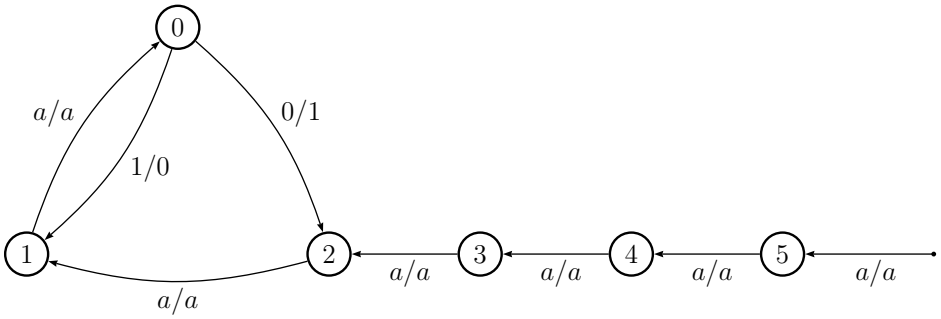


Figure 4: The infinite Knuth extension $\mathcal{K}$ of the transducer $\mathcal{A}$.

It follows from the construction that $\underline{k}(a_1a_2\ldots a_kx) = a_1\ldots a_k\,\underline{0}(x)$ for all $k \geq 0$. Hence there is a useful "lifting principle" for identities: in particular, $\underline{0}^2\,\underline{1}^2\,\underline{2} = I$ implies the *cancellation identities* $\underline{k}^2\,\underline{k+1}^2\,\underline{k+2} = I$. Here is another group of identities obtained from the extension.

Proposition 2.1 (Shift Identities) *For all $k \geq 0$:* $\underline{k}^2 = \underline{k+2}\,\underline{k+3}$,

While $\mathcal{K}$ and $\mathcal{A}$ generate the same transduction group, the former automaton affords a better notation system. For any vector we refer to the sum of its components as its *weight*.

Theorem 2.1 (Knuth Normal Form [10]) *Every transduction f in $\mathcal{S}$ has a unique normal form $f = \underline{k_1}\,\underline{k_2}\,\ldots\,\underline{k_n}$ where $k_i < k_{i+1}$.*

Proof. Using $\mathcal{K}$, we can think of a transduction f as an element $(e_i)_{i\geq 0}$ in the coproduct $\coprod_{\mathbb{N}} \mathbb{N}$. If $e_i \leq 1$ for all i we have the desired normal form. Let us say that position i requires attention if $e_i \geq 2$. So assume there is at least one position requiring attention and apply the following rewrite operation. Let k be the minimal position that requires attention. If in addition $e_{k+1} \geq 2$ and $e_{k+2} \geq 1$, apply a cancellation identity; otherwise, apply a shift identity. Note that this guarantees that ultimately k will not require attention; moreover, k can never require attention again.

We claim that this rewrite operation must terminate. Assume otherwise. Application of a cancellation identity reduces weight, so we may safely assume that only shift operations are used. Furthermore, we may assume that $e_k = 0$ when k ceases to require attention. Let ℓ be minimal such that in the original presentation of f we have $i \geq \ell$ implies $e_i = 0$. From our preceding observation, at some point none of the positions less than ℓ will require attention. At that moment, only positions ℓ, $\ell+1$ and $\ell+2$ can require attention. Indeed, from then on, there is a window of size 3 that contains all positions that require attention. Since we assume non-termination, the rewrite operation on this length 3 vector can now be described by the the rational matrix

$$\begin{pmatrix} 0 & 1 & 0 \\ \frac{1}{2} & 0 & 1 \\ \frac{1}{2} & 0 & 0 \end{pmatrix}.$$

The dominant eigenvalue of this matrix is 1, with corresponding eigenvector $(2, 2, 1)$ and we have a contradiction. □

As a consequence of Knuth's theorem we can see that the map $\mathbb{Z}^2 \to \mathcal{S}$, $(a, b) \mapsto \underline{0}^a\underline{1}^b$ is a group isomorphism.

Corollary 2.1 *$\mathcal{S}$, the transduction semigroup of $\mathcal{A}$, is isomorphic to $\mathbb{Z}^2$.*

We will write $\mathsf{KNF}(f)$ for the Knuth normal form of a transduction f. As Knuth points out, there is an elegant description of the group isomorphism from the corollary in terms of a numeration system with base $(i-1) \in \mathbb{Z}[i]$ and digits $\{0, 1\}$. More precisely, there is a group isomorphism $\Phi : \mathcal{S} \to \mathbb{Z}[i]$ from $\mathcal{S}$ to the Gaussian integers defined as follows. Let $\mathsf{KNF}(f) = \underline{k_1}\,\underline{k_2}\,\ldots\,\underline{k_n}$ and set $\Phi(f) = \sum_j (i-1)^{k_j}$. Incidentally, letting $(i-1)^k = a_k + ib_k \in \mathbb{Z}[i]$ we have $\mathsf{KNF}(\underline{0}^{a_k+b_k}\underline{1}^{b_k}) = \underline{k}$. To see this, note the linear recurrence $(a_k, b_k) = -2(a_{k-1}, b_{k-1}) - 2(a_{k-2}, b_{k-2})$. Hence, by induction,

$$\begin{aligned} \underline{0}^{a_k+b_k}\underline{1}^{b_k} &= (\underline{0}^{(a_{k-1}+b_{k-1})}\underline{1}^{b_{k-1}})^{-2}\,(\underline{0}^{(a_{k-2}+b_{k-2})}\underline{1}^{b_{k-2}})^{-2} \\ &= \underline{k-1}^{-2}\underline{k-2}^{-2} = \underline{k} \end{aligned}$$

where the last step follows from our cancellation identities.

The rewrite system in the proof of theorem 2.1 provides an algorithm to compute $\mathsf{KNF}(f)$. For example, $\mathsf{KNF}(\underline{0}^{50}) = \underline{3}\,\underline{4}\,\underline{9}\,\underline{11}\,\underline{12}$. Figure 5 shows the KNF of all $\underline{0}^a$, $0 \leq a < 2^{10}$. The height of the column indicates the number of terms and the actual terms are color coded. There are obvious patterns, but it seems difficult to give a simple description of this sequence.

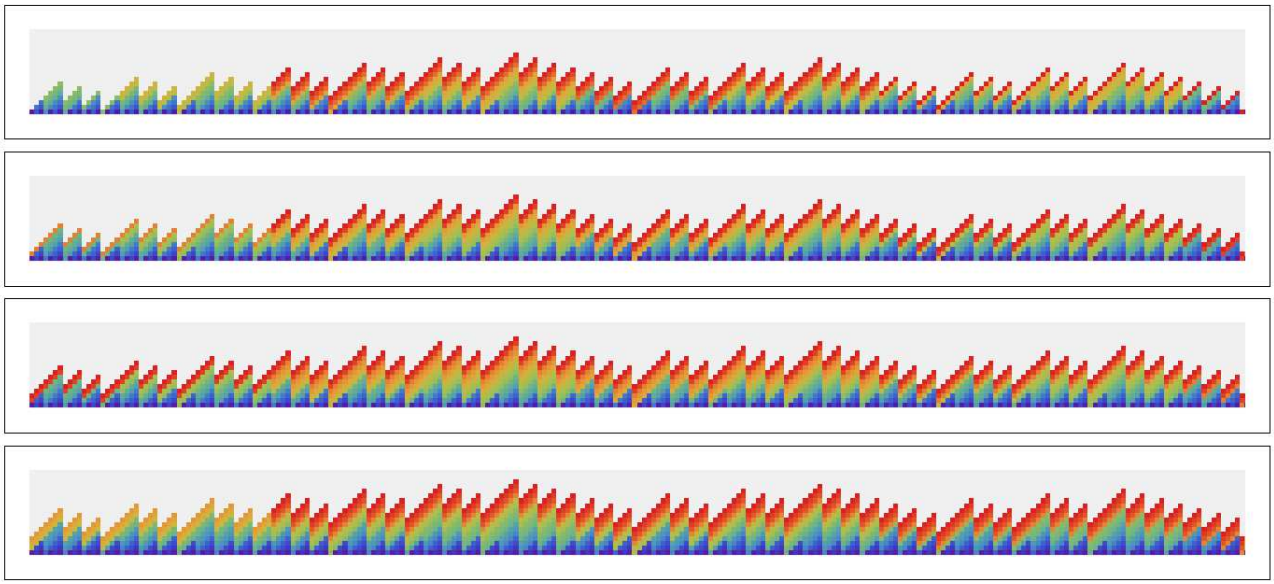


Figure 5: KNF of $\underline{0}^a$ for $0 \leq a < 2^{10}$.

Things become even more interesting if we plot the KNF for transductions $\underline{0}^a\ \underline{1}^b$. Figure 6 shows the number of terms in the KNF of all transductions of this form when $|a|, |b| \leq 2^8$.

Knuth normal form has a number of interesting properties that will be useful later. We need a bit more notation: let $\mathsf{KNF}_1(f)$ denote the first term in the normal form of $f \neq I$; for any transduction f, write $\mathsf{shift}_s(f)$ for the transduction obtained by replacing any term $\underline{k}$ in the KNF of f by $\underline{k+s}$. 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 $\gamma_i' = \mathsf{shift}_1(\gamma_i)$.

Lemma 2.1 *Let $0 \leq k$ and $0 \leq i < 4$. Then $\mathsf{KNF}(\underline{0}^{2^{4k+i}}) = \mathsf{shift}_{8k+2i}(\gamma_i)$.*

More generally, for $f = \underline{0}^a\underline{1}^b$, we have $\mathsf{KNF}(f^{2^{4k+i}}) = \mathsf{shift}_{8k+2i}(\mathsf{KNF}(\gamma_i^a(\gamma_i')^b))$.

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 $\mathsf{KNF}(\underline{0}^{2^{4k+1}}) = \mathsf{KNF}(\underline{8k}^2) = \underline{8k+2}\,\underline{8k+3} = \mathsf{shift}_{8k+2}(\gamma_1)$. The cases $i = 2, 3$ are entirely similar. The second claim follows immediately from the first. □

Observe that the first term in the Knuth normal form of $\underline{0}^{2^k}$ is $\underline{2k}$. But then $\underline{0}^{2^k}(uab) = u\bar{a}b$ for all words u of length $2k$ and we can verify our conjecture 1.1: all even levels in the orbit tree of $\underline{0}$ double, all odd levels split.

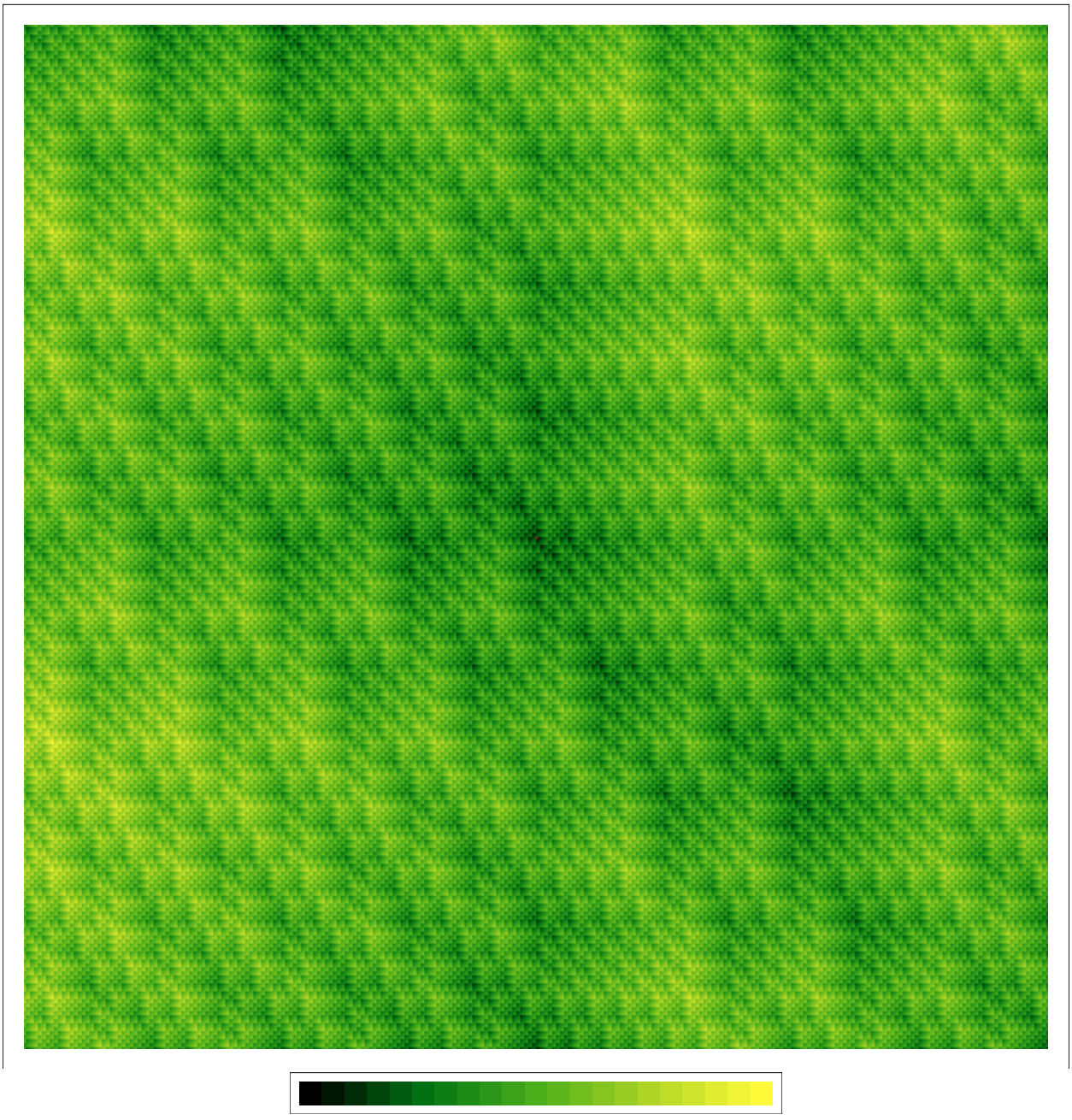


Figure 6: The weight of the KNF of group elements in the range $-2^8 \leq a, b \leq 2^8$. The origin is the red dot in the middle. The color patch corresponds to counts from 0 to 20.

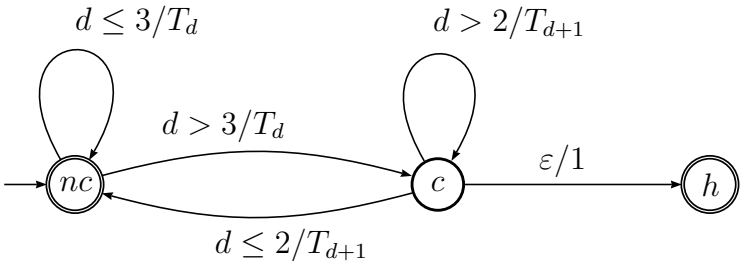
Corollary 2.2 *For every transduction f in $\mathcal{S}$ other than the identity, the orbit tree of f is homogeneous and has type $22\ldots2(12)^\omega$.*

Proof. Let $\underline{k_1}\,\underline{k_2}\ldots\underline{k_n}$ be the normal form of f. We claim that the type of the orbit tree of f is $2^{k_1}(12)^\omega$. Clearly, the first k_1 levels in the tree split. For longer words write $x = u\,v$ where u has length k_1 and let $g = \partial_{0^{k_1}} f$. Then $f(uv) = ug(v)$ and it suffices to show that all odd transductions have orbit tree of type $(12)^\omega$. But g has normal form $\underline{0}\,\underline{\ell_2}\ldots\underline{\ell_n}$ and all the transductions $\underline{\ell_i}$ have orbit lengths dividing the orbit lengths of $\underline{0}$; our claim follows. □

One might wonder how hard it is to compute KNF. Somewhat surprisingly, a finite state machine suffices. To see why, first pre-compute the KNFs of $\underline{0}^a$ for $0 \le a < 16$, written as bit-vectors and padded out to 8 bits when 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. 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 hexdigit d the correct output is T_d, but with a carry it's $T_{d+1 \bmod 16}$. Here is a sketch of the appropriate transducer; input is hexadecimal, output is binary.



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:

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

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 a bit more complicated.

3 Iteration and Orbit Equivalence

In order to evaluate a transduction f on a particular word, we can use the obvious recursive approach: $f(ax) = f(a)\,\partial_a f(x)$. We can describe this computation by a transition system $\mathcal{G}$, much the way $\mathcal{A}$ describes the basic transductions $\underline{0}, \underline{1}, \underline{2}$: the states are all transductions in the group and the transitions are $f \xrightarrow{s/f(s)} \partial_s f$. Of course, this system is infinite; it is referred to as the *complete automaton* in [11]. Using the group representation $\boldsymbol{u} \in \mathbb{Z}^2$ to denote the elements in $\mathcal{S}$, we can compute residuals as follows.

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

where

$$A = \begin{pmatrix} -1 & 1 \\ -1/2 & 0 \end{pmatrix} \quad \text{and} \quad \boldsymbol{a} = (1, 3/2)$$

A has complex eigenvalues of norm $1/\sqrt{2} < 1$. As a consequence, the set $\{\, \partial_x \boldsymbol{u} \mid x \in \mathbf{2}^\star \,\}$ is bounded and thus finite. In fact, $\mathcal{G}$ has exactly 8 nontrivial strongly connected components. Note that $\mathcal{G}$ admits an involution that sends $f \xrightarrow{s/t} g$ to $f^{-1} \xrightarrow{\bar{s}/\bar{t}} g^{-1}$. Omitting the component of the identity, the strong components modulo this involution are listed in figure 7. Note that the operations ∂_s preserve weight modulo 5; thus, for example, all transitions of weight 2 modulo 5 are bound to wind up in the component on the top right in figure 7 under repeated application of ∂_s.

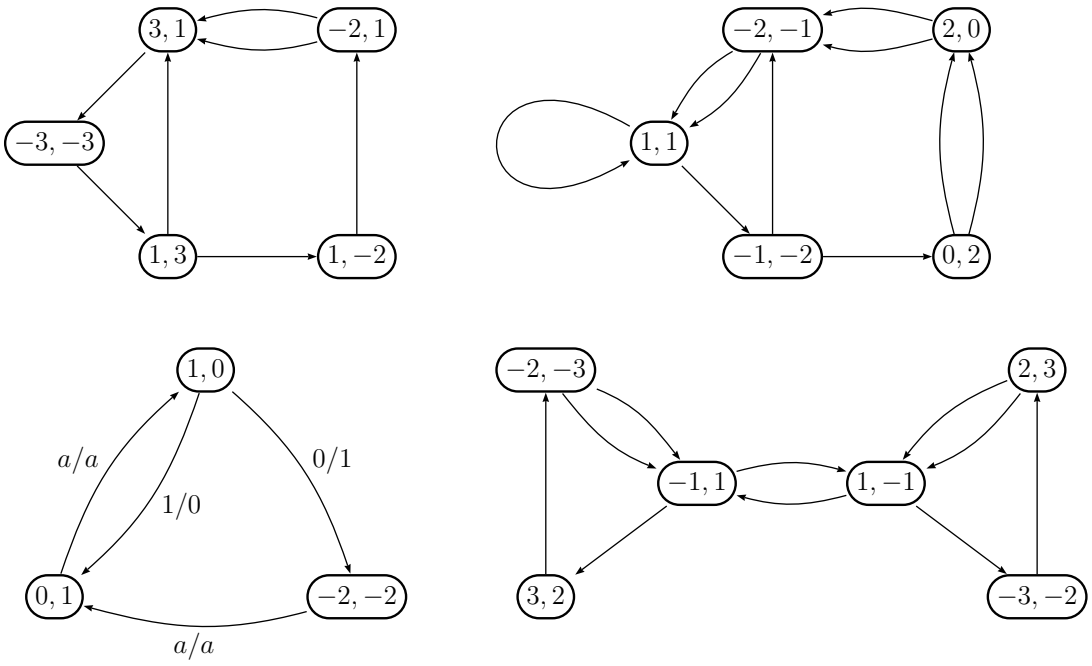


Figure 7: Strong components in the complete automaton $\mathcal{G}$. The last component is invariant under the involution.

Thus evaluating $f(x)$ comes down to tracing a path in the complete automaton $\mathcal{G}$ and it follows that the residuals along this path stay bounded in norm. If the input word is long enough, we ultimately wind up in one of the strongly connected components. Hence we can calculate f quickly.

Proposition 3.1 *The Iteration Problem for any transduction f in $\mathcal{S}$ of norm w can be solved in time $O(|x| \log^2 w)$.*

Proof. Because of the contraction property of residuals it takes $O(\log w)$ steps before one of the strongly connected components is reached. Beyond this point, the computation is linear in the length of x and our claim follows. □

The decision problems from section 1 clearly generalize to the whole group $\mathcal{S}$. For example, the general Orbit Problem can be phrased as follows: given a transduction $f \in \mathcal{S}$ and two words x and y, determine whether y lies in the orbit of x under f. Correspondingly, let us say that two words x and y are *orbit equivalent* (wrto. transduction f) if they lie on the same orbit under f. How hard is it to check orbit equivalence for, say, $\underline{0}$? Clearly, we may assume that x and y have the same length n, so a brute-force approach will take $O(n\, 2^{n/2})$ steps. As a first step towards a better algorithm, we will provide a geometric description of our orbits. To this end, let $\mathcal{S}_0$ be the semigroup generated by $\underline{0}$ and $\underline{1}$. For any k, let H_k be the quotient group of $\mathcal{S}_0$ obtained by factoring with respect to the congruence $x^{2^k} = I$. It is customary to refer to $\mathbf{2}^n$ as a subset of the full binary tree as the nth level set.

Lemma 3.1 *The semigroup $\mathcal{S}_0$ acts transitively on all level sets $\mathbf{2}^n$. The group H_k acts simply transitively on $\mathbf{2}^{2k}$.*

Proof. As we have seen, $\mathsf{KNF}_1(\underline{0}^{2^k}) = \underline{2k}$ and $\mathsf{KNF}_1(\underline{1}^{2^k}) = \underline{2k+1}$. But then the semigroup $\mathcal{S}_0$ must act transitively on $\mathbf{2}^n$. To see this, consider two words $x = x_0 x_1 \dots x_{n-1}$ and $y = y_0 y_1 \dots y_{n-1}$ of length n. Assume by induction that for some $f \in \mathcal{S}_0$ we have $f(x_{<i}) = y_{<i}$. If $f(x_{\le i}) = y_{\le i}$ there is nothing to do. If $i = 2k$ is even then $f' = f\,\underline{0}^{2^k}$ maps $x_{\le i}$ to $y_{\le i}$; for odd i, $f' = f\,\underline{1}^{2^k}$ works. Clearly $f' \in \mathcal{S}_0$.

If we take the quotient of $\mathcal{S}_0$ with respect to the congruence induced by $\underline{0}^{2^k} = \underline{1}^{2^k} = I$ we obtain a group. Closer inspection of the previous argument shows that this group still acts transitively on $\mathbf{2}^{2k}$. Since the cardinality of H_k is 2^{2k} is must in fact act simply transitively. □

To obtain a more geometric interpretation of the last lemma let us say that two transductions f and g in $\mathcal{S}$ are *orthogonal* if their orbits overlap as little as possible: $\mathsf{orb}(u; f) \cap \mathsf{orb}(u; g) = \{u\}$ for all words u. Then the lemma shows that $\underline{0}$ and $\underline{1}$ are orthogonal. Hence there is a natural, two-dimensional coordinate system for words based on the transducer $\mathcal{A}$ rather than some external ordering principle such as lexicographic order given by the bijection

$$\mathbf{2}^{2n} \longleftrightarrow \mathbb{Z}/(2^n) \times \mathbb{Z}/(2^n)$$

that associates a word w of length $2n$ with the unique coordinates $0 \le w_0, w_1 < 2^n$ such that $w = \underline{0}^{w_0}\underline{1}^{w_1}(0^{2n})$. We will write $\langle\, w \,\rangle$ for the coordinates of w. Not that this observation gives

rise to yet another computational problem, the *Coordinate Problem*: given a word $x \in \mathbf{2}^{2n}$, determine its coordinates. Surprisingly, we will show that this problem can be solved by a finite state machine in section 4.

At any rate, we can express transductions in terms of coordinates: let $f = (u_1, u_2) \in \mathbb{Z}^2$ in group representation and let $\langle\, w \,\rangle = (w_1, w_2)$. By commutativity, $f(w) = \underline{0}^{u_1+w_1}\underline{1}^{u_2+w_2}(0^{2n}))$ so that $\langle\, f(w) \,\rangle = (u_1 + w_1, u_2 + w_2)$. Hence

$$\langle\, \mathsf{orb}(w; f) \,\rangle = (w_1, w_2) + \mathbb{N} \cdot (u_1, u_2) \pmod{2^n}$$

Thus, in our coordinate system, orbits are affine subspaces of $\mathbb{Z}/(2^n) \times \mathbb{Z}/(2^n)$; in fact, they are all translations of the basic linear subspace $\mathsf{orb}(0^{2n}; f)$. This provides another proof of the homogeneity of the orbit tree. Since orbits have a simple structure, it should not be difficult to determine whether two transductions have the same orbits (as sets). Let us say that two transductions $f, g \in \mathcal{S}$ are *star equivalent*, in symbols $f \approx g$, if their orbits agree: $\mathsf{orb}(x; f) = \mathsf{orb}(x; g)$ for all words x. For our decision algorithm we will need a slightly stronger notion: define the *generalized orbit* of a word x under (f, h) to be

$$\mathsf{orb}(x; f, h) = h(\mathsf{orb}(x; f)) = \{\, h(f^i(x)) \mid i \geq 0 \,\}.$$

By a *star pair* we mean any pair (f, h) of transductions; two star pairs are *star equivalent* if their generalized orbits coincide, in symbols $(f, h) \approx (g, h')$. Note the following simple sufficient condition for star equivalence.

Proposition 3.2 *For any odd integer k and any integer ℓ we have $(f, h) \approx (f^k, f^\ell h)$.*

Lastly, we write $\nu_2(n)$ for the dyadic valuation of n, so that $n = n_0\, 2^{\nu_2(n)}$ with n_0 odd.

Theorem 3.1 *Star equivalence in $\mathcal{S}$ is decidable in polynomial time.*

Proof. First consider two transductions $f = (u_1, u_2)$ and $g = (v_1, v_2)$. We claim that $f \approx g$ if, and only if, $\nu_2(u_i) = \nu_2(v_i)$ and $u_1 v_2 = u_2 v_1$. To see this, note that as in the preceding paragraph, one can show that f and g are star equivalent if, and only if, for each $n \geq 1$ there is a unit $z = z_n$ such that

$$u_i = z \cdot v_i \pmod{2^n}$$

for $i = 1, 2$. The sequence (z_n) defines a dyadic rational, essentially the slope of the line representing the orbit. Our claim follows.

Now consider two star pairs (f, h_1) and (g, h_2). Letting $h = h_1^{-1} h_2$ we have $(f, h_1) \approx (g, h_2)$ if, and only if, $\mathsf{orb}(x; f) = \mathsf{orb}(x; g; h)$ for all words x. The last condition means that the linear subspace $\mathbb{N}(u_1, u_2) \pmod{2^n}$ coincides with the affine subspace $\mathbb{N}(v_1, v_2) + (c_1, c_2) \pmod{2^n}$, for all $n \geq 1$, where $h = (c_1, c_2)$. If follows that f and g must be star equivalent and that (c_1, c_2) lies in the linear subspace. Hence we have to check the solvability of the equations $c_i = z \cdot u_i \pmod{2^n}$, $i = 1, 2$, for all $n \geq 1$. It is easy to see that these equations are solvable if, and only if, $\nu_2(u_i) \leq \nu_2(c_i)$ and $u_1 c_2 = u_2 c_1$.

It is clear from our discussion that the arithmetic operations required to test star equivalence are all polynomial in the size of the input. □

The position of point (c_1, c_2) in the orbit of f may be fractional in the sense that the solutions z_n define a dyadic rational. For example, for $(c_1, c_2) = (1, 3)$ and $f = (3, 9)$ the positions are given by $1, 3, 3, 11, 11, 43, 43, 171, 171, 683, 683, 2731, \ldots$ which is the standard sequence representation of $1/3$ in $\mathbb{Z}_2$. In a similar vein, given odd integers r and s, we can define the *fractional power* $f^{r/s}$ of $f = (a_1, a_2)$, written in terms of coordinates, as follows:

$$\left\langle\, f^{r/s}((w_1, w_2))\,\right\rangle = (w_1, w_2) + rs_n\,(a_1, a_2) \pmod{2^n}$$

where (s_n) is the standard sequence representation of $1/s$ in the dyadic numbers $\mathbb{Z}_2$, see [6].

We can now pin down the computational complexity of orbit equivalence. For the time being, let us only consider $f = \underline{0}$. The orbit of x has length exponential in the length of x, so the question arises whether there is a polynomial time shortcut. As we will see, there is a linear time algorithm; in fact, orbit equivalence can be decided by a finite state machine. The appropriate type of machine in this context uses two input tapes and reads the given words in a synchronous fashion, one symbol from each word at a time. We can think of these machines as being ordinary language acceptors. To this end, define the the *convolution* $x{:}y$ of two words x and y of the same length to be the word

$$x{:}y = \begin{array}{|c|c|c|c|} \hline x_1 & x_2 & \ldots & x_n \\ \hline y_1 & y_2 & \ldots & y_n \\ \hline \end{array}$$

over the alphabet $\mathbf{2}\times\mathbf{2}$. Then a finite state machine over $\mathbf{2}\times\mathbf{2}$ defines a relation $R \subseteq (\mathbf{2}\times\mathbf{2})^\star$ by setting $(x, y) \in R$ iff $x{:}y$ is accepted by the machine. These relations are known as *automatic* or *synchronous*, and we can think of our transducer as establishing automaticity of the relations $\underline{0}$, $\underline{1}$ and $\underline{2}$. Automatic relations are important since they form a Boolean algebra, which fact can be exploited to decide first-order logic over structures consisting of automatic relations, see [13, 9] for background.

In order to construct a finite state machine that decides orbit equivalence for $\underline{0}$, we use Brzozowski's quotient method [2]. Since orbit equivalence is length-preserving, we can think of this relation as a formal language $R \subseteq (\mathbf{2}\times\mathbf{2})^\star$. Regularity of R as a language is equivalent to having finitely many (left) quotients

$$u^{-1}R = \{\, x \in (\mathbf{2}\times\mathbf{2})^\star \mid ux \in R \,\}.$$

To calculate these quotients we use the generalized orbits from above. Define the generalized orbit relation $\mathbf{R}(f, h)$ as follows: $\mathbf{R}(f, h)$ holds on u and v if $v \in \mathsf{orb}(u; f, h)$. When h is the identity then $\mathbf{R}(f, I)$ is simply the orbit relation of f. However, in general $\mathbf{R}(f, h)$ fails to be an equivalence relation (and even to be reflexive). But, we still can use these relations to compute quotients. It is easy to verify the next lemma.

Lemma 3.2 *Quotient Lemma*

Let f and h be two transductions and set $b = h(a)$, $h = (h_0, h_1)$ s. If $f = (f_0, f_1)$ in wreath notation, then

$$(a{:}b)^{-1}\,\mathbf{R}(f, h) = \mathbf{R}(f_a, h_a)$$

The quotient with $a{:}\overline{b}$ is empty. Otherwise, for $f = (f_0, f_1)\,\sigma$ we have

$$(a{:}b)^{-1}\,\mathbf{R}(f,h) = \mathbf{R}(f_a f_{\overline{a}}, h_a)$$
$$(a{:}\overline{b})^{-1}\,\mathbf{R}(f,h) = \mathbf{R}(f_a f_{\overline{a}}, f_a h_{\overline{a}})$$

Given any transduction f, the lemma shows how do define a transition system $\mathcal{M}_f$ that decides orbit equivalence for f. For let $\mathcal{Q} \subseteq \mathcal{S}^2$ be the closure of (f, I) under the quotient operations. Call a star pair (f, h) even if f is even, odd otherwise. We can introduce transitions on $\mathcal{Q}$ in the natural way: for example, for even (f, h) we have $(f, h) \xrightarrow{a:b} (f_a, h_a)$ where $b = h(a)$, and $(f, h) \xrightarrow{a:\overline{b}} \perp$ where $\perp$ is a special sink state corresponding to the empty quotient. For odd (f, h) there are four transitions, none with target $\perp$. (f, I) is the initial state and all states other than the sink are final.

Of course, $\mathcal{M}_f$ might be infinite. To see why $\mathcal{Q}$ is actually finite let us first consider the case $f = \underline{0}$. Let $\mathcal{Q}_0$ be the projection of $\mathcal{Q}$ onto the first components. Since we are calculating modulo star equivalence, $\mathcal{Q}_0$ has only four elements:

$$\underline{0} \to \underline{1}\,\underline{2} \to \underline{0}\,\underline{1} \to \underline{1}^{-1} \approx \underline{1} \to \underline{0}$$

During the execution of the closure algorithm we cycle through these four transductions in the first component. Note that they are alternately odd and even, so, according to the last equation in the quotient lemma, at all the odd stages some of the second components pick up an additional term and it is far from clear that $\mathcal{Q}$ is also finite. A brute-force computation using only proposition 3.2 to approximate star equivalence (rather than the algorithm from theorem 3.1) produces a machine with 34 states, plus the sink. The star pairs so obtained have the form

$$(\underline{0}, \underline{1}^\ell),\ |\ell| \le 3, \quad (\underline{1}\,\underline{2}, \underline{0}^\ell),\ |\ell| \le 5, \quad (\underline{0}\,\underline{1}, \underline{0}^\ell),\ |\ell| \le 3, \quad (\underline{1}, \underline{0}^\ell),\ |\ell| \le 4.$$

and the automaton turns out to be minimal. We will now show that this is no coincidence.

Theorem 3.2 *The orbit relation $f^\star$ is automatic for all transductions f in $\mathcal{S}$.*

Proof. By construction, $\mathcal{M}_f$ decides orbit equivalence for f and it remains to show that $\mathcal{Q}$ is finite. Let us first focus on $\mathcal{Q}_0$, the projection of $\mathcal{Q}$ onto the first component. It follows from lemma 3.2 that $\mathcal{Q}_0$ is the orbit of f under the map $\pi(f) = \partial_0 f$ for f even, and $\pi(f) = \partial_0 f^2$ otherwise. Except for the fixed point I, all orbits of π end in an 8-cycle and, modulo star equivalence, even in a 4-cycle (we assume a is odd):

$$(a, b), (2b - 2a, -a), (a - 2b, a - b), (2b, 2b - a), (-a, -b) \approx (a, b)$$

Thus $\mathcal{Q}_0$ is finite.

In fact, $\mathcal{M}_f$ consists of an acyclic part that leads to quadripartite components. Denote the functions on one of the corresponding 4-cycle in $\mathcal{Q}_0$ by f_0, f_1, f_2 and f_3. It suffices to show that for any h, the closure of the star pair (f_0, h) under quotients is finite. To this

end we will over-approximate the operations required for the second components by a map $\Phi : \mathbb{Q}^2 \to \mathfrak{P}(\mathbb{Q}^2)$ defined by

$$\Phi(\boldsymbol{u}) = \{\, A \cdot \boldsymbol{u} + c\,\boldsymbol{a} + \boldsymbol{w} \mid c \in \{0, \pm 1\}\,, \boldsymbol{w} \in W\,\}.$$

Here A and $\boldsymbol{a}$ are from section 2 and W is a set of residuals obtained from the transductions in the π-cycle as required by lemma 3.2. Again, A is a contraction so that the closure of h under Φ is a bounded set in $\mathbb{Q}^2$, containing only finitely many integral points. □

In the special case $f = \underline{0}$ the π orbit has the form $(1,0),(2,1),(1,1),(0,1)$, up to star equivalence. Hence we can exploit the rules for star equivalence to rewrite the translations into a form where only one component is non-zero. We are left essentially with a one-dimensional problem and one can show that

$$\Phi(x) = \{\, -(i+x)/4 \mid -11 \le i \le 13\,\}.$$

Hence the closure under Φ starting at points $|x| \le 13/3$ will stay in the interval $[-13/3, 13/3]$. There are 9 integral points in the interval and, since there are 4 rounds in the quotient process, an upper bound for the number of states in $\mathcal{M}_{\underline{0}}$ is 36, surprisingly close to the actual value.

Returning to our basic transduction $\underline{0}$, we can extract more information from the orbit automaton $\mathcal{M}_{\underline{0}}$. For odd any odd state p there are transitions of the form

$$p \xrightarrow{s:s} q, p \xrightarrow{0:1} q', p \xrightarrow{1:0} q'' \quad \text{where} \quad q \neq q' \neq q''$$

and for even p the transitions not leading to the sink are

$$p \xrightarrow{s:s} q \quad \text{or} \quad p \xrightarrow{0:1} q', p \xrightarrow{1:0} q'' \quad \text{where} \quad q' \neq q''$$

Note that this provides another proof of conjecture 1.1 regarding the structure of the orbit tree. It is now easy to construct a transducer that computes the root function: simply remove all transitions $p \xrightarrow{s:1} q$ for all odd p from the orbit automaton. After removal of inaccessible states and the sink we obtain a transducer over $(\mathbf{2} \times \mathbf{2})^\star$ with 21 states that computes root. The root language $(0\mathbf{2})^\star(\varepsilon + 0)$ itself is much less complicated, it is accepted by a machine with two states plus a sink.

With a little more effort one can extract more information from $\mathcal{M}_{\underline{0}}$. For example, we know that "de-shuffling" the orbit of some $x \in \mathbf{2}^{2n}$ produces one component of maximal cardinality 2^n. The following proposition deals with the other component.

Proposition 3.3 *Let $u \in \mathbf{2}^{2k}$. The cardinality of $\mathsf{even}(\mathsf{orb}(u; \underline{0}))$ is $3^{(k-1)/2}$ when k is odd and $2 \cdot 3^{(k-2)/2}$ when $k > 0$ is even.*

We are only aware of a fairly messy computational proof and will spare the reader the mental anguish.

4 Time Stamps and Coordinates

In light of the results of the last section it seems plausible that the Timestamp Problem and the Coordinate Problem also yield to polynomial time algorithms. More surprising is the fact that both be solved by finite state machines.

Let us first dispense with the Coordinate Problem: we are given a word $x \in \mathbf{2}^{2n}$, and we need to compute two integers s and t, $0 \leq s, t < 2^n$, such that $\underline{0}^s\underline{1}^t(0^{2n}) = x$. Write $s = \sum s_i 2^i$ and $t = \sum t_i 2^i$. Recall the from section 3 the transductions $\gamma_0, \ldots, \gamma_3$. We calculate the binary digits of s and t in n rounds as follows.

// coordinate algorithm
    $h = (0, 0)$;
    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 3.

Theorem 4.1 *The Coordinate Problem 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 2.1.

In phase 1, if $f(\mathbf{0}0) = xa$ then $f' = f$ and we have $f'(\mathbf{0}0) = f(\mathbf{0}0) = xa$. 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 $f'(\mathbf{0}0) = \underline{0}^{2^{4n}}(f(\mathbf{0}0)) = \underline{0}^{2^{4n}}(x\overline{a}) = x\,\underline{0}(\overline{a}) = xa$. 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 2.1.

For phase 2 first consider the case $f'(\mathbf{0}00) = xab$. Then $t'' = t$ and we have $f''(\mathbf{0}00) = f'(\mathbf{0}00) = xab$. 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 $f''(\mathbf{0}00) = \underline{1}^{2^{4n}}(f'(\mathbf{0}00)) = \underline{1}^{2^{4n}}(xa\overline{b}) = x\,\underline{1}(a\overline{b}) = xa\underline{0}(\overline{b}) = xab$. 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. □

Note that one can reduce the computation of timestamps to a coordinate computation using the description of orbits as affine subspaces in section 3. For $f = (f_1, f_2)$ and $x, y \in \mathbf{2}^{2n}$ with coordinates (x_1, x_2) and (y_1, y_2) we can solve the linear equations $x_i + tf_i = y_i \pmod{2^n}$.

However, there is a more direct approach: a finite state machine can compute the appropriate timestamp, or determine that none exists.

Theorem 4.2 *The Timestamp Problem for any transduction f in $\mathcal{S}$ can be solved by a transducer that computes the timestamp in reverse binary.*

Proof. Consider words $x, y \in 2^{2n}$. We need to compute an exponent $t = \sum t_{i<n} 2^i$ such that $f^t(x) = y$, or determine that no such t exists. Let $\mathsf{KNF}_1(f) = \underline{k}$. If $2n \leq k$ then $t = 0$ provided that $x = y$; otherwise y is not in the f-orbit of x. So suppose $2n > k$ and assume without loss of generality that the prefixes of length k of x and y agree. Write $x = ux'$ and $y = uy'$ accordingly where $|u| = k$. Since $\mathsf{KNF}_1(\partial_u f) = \underline{0}$ we may safely assume that f is odd and $x, y \in 2^{2n}$.

Let $f = \underline{0}^a \underline{1}^b$ where a is odd. Then by lemma 2.1

$$\mathsf{KNF}(f^{2^{4r+i}}) = \mathsf{shift}_{8n+2i}(\mathsf{KNF}(\gamma_i^a\,(\gamma_i')^b))$$

where $0 \leq i < 4$ and $\gamma_i' = \mathsf{shift}(\gamma_i)$. But then $\mathsf{KNFS}(f^{2^j})$ is periodic with period at most 4. Let us refer to this sequence as Γ_j. Then the following algorithm computes the requisite exponent t:

// timestamp algorithm
    $h = (0, 0)$;
    for $r = 0, \ldots, n-1$ **do**
        $t_r = h_1 + x_{2r} + y_{2r} \bmod 2$; // phase 1: bind t_r
        $h = \partial_{x_{2r}}(h + t_r \cdot \Gamma_d)$;
        if $h_1 + x_{2r+1} + y_{2r+1} = 0 \bmod 2$ // phase 2: check
        then $h = \partial_{x_{2r+1}}(h)$;
        else return No;
    return t;

The correctness proof is essentially the same as for the coordinate algorithm from above. Moreover, because of the contraction property from section 3, the algorithm can be implemented by a finite state machine and we are done. □

5 Some Questions

Our transducer $\mathcal{A}$ is based on a stride of 2 or 3, depending on whether the currently read letter is a 0 or a 1. Of course, we could similarly consider a stride of n and m. This leads to a simple class of transducers which generalize $\mathcal{A}$: *cycle-cum-chord* transducers. In a cycle-cum-chord transducer $\mathcal{A}_m^n$, $1 \leq m \leq n$, the transition diagram consists of a cycle of length n plus one chord and the source node of the chord is the only toggle state in the transducer. More precisely, the machine has state set $\{0, 1, \ldots, n-1\}$, and the transitions are given by $\underline{0} = (\underline{n-1}, \underline{m-1})\sigma$, $\underline{k} = (\underline{k-1}, \underline{k-1})$. Thus, our old transducer $\mathcal{A}$ is none other than $\mathcal{A}_2^3$.

The diagram of $\mathcal{A}_2^5$ is shown in figure 8. These devices share many of the properties of $\mathcal{A}$. For example, the transition semigroup of $\mathcal{A}_m^n$ is a commutative group. In the degenerate case $n = m$ we get the Boolean group $\mathbf{2}^n$, in all other cases the group is isomorphic to $\mathbb{Z}^d$ where $d = n - \gcd(n, m)$, see [15].

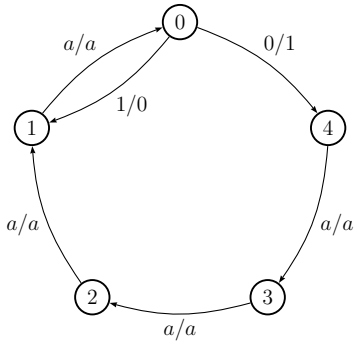


Figure 8: $\mathcal{A}_2^5$, an invertible transducer on 5 states that generalizes $\mathcal{A}$.

More generally, cycle-cum-chord transducers are a particular example of so-called *invertible transducers* over a binary alphabet, see [7]. The transitions in a binary invertible transducer are required to be of the form $p \xrightarrow{a/\sigma(a)} q$ where σ is in $\mathfrak{S}_2$. The transductions determined by any invertible transducer are bijections and can be represented in the wreath form of section 2. Of course, the transduction semigroup need not be commutative and it need not be a group.

In general it is quite difficult to understand the semigroups and groups generated by invertible transducers. For example, Grigorchuk's well-known example of a group of intermediate growth has an elegant description in terms of a 5-state invertible transducer with a single toggle state. Even two states can already produce a transduction group of significant complexity: the two-state automaton in figure 9 generates the lamplighter group $\mathbf{2} \wr \mathbb{Z}$. The orbit tree of this transducer is not regular, so the orbit relation cannot be rational. A complete classification of all the groups associated with 2-state automata over a binary alphabet can be found in [7] and [12] contains a detailed discussion of automata with transduction groups that are free Abelian.

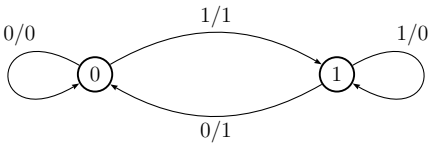


Figure 9: A 2-state transducer generating the lamplighter group.

Surprisingly, even cycle-cum-chord transducers turn out to be rather complicated. It is not hard to show that $\mathcal{A}_1^n$ and $\mathcal{A}_m^n$ both have rational orbit relations, and one can identify a larger class of such cycle-cum-chord transducers for which this holds, see [15]. One can show that in $\mathcal{A}_3^4$ the orbit relation of $\underline{0}$ fails to be rational, but our proof uses field theory in combination with symbolic computation and does not appear to generalize to any other situation.

References

[1] J. Berstel. Transductions and context-free languages. `http://www-igm.univ-mlv.fr/~berstel/LivreTransductions/LivreTransductions.html`, 2009.

[2] J. A. Brzozowski. Derivatives of regular expressions. *Journal Assoc. for Comp. Machinery*, 11, 1964.

[3] S. Eilenberg. *Automata, Languages and Machines*, volume A. Academic Press, 1974.

[4] C. C. Elgot and J. E. Mezei. On relations defined by generalized finite automata. *IBM J. Res. Dev.*, 9:47–68, January 1965.

[5] M. Gardner. Mathematical games: The fantastic combinations of John Conway's new solitaire game 'Life'. *Sci. American*, 223(4):120–123, 1970.

[6] F. Q. Gouvêa. *p-Adic Numbers: An Introduction*. Springer Verlag, 2nd edition, 1997.

[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. *Automata Theory and its Applications*. Birkhäuser, 2001.

[10] D. Knuth. Private communication, 2010.

[11] V. Nekrashevych. *Self-Similar Groups*, volume 117 of *Math. Surveys and Monographs*. AMS, 2005.

[12] V. Nekrashevych and S. Sidki. *Automorphisms of the binary tree: state-closed subgroups and dynamics of 1/2-endomorphisms*. Cambridge University Press, 2004.

[13] J. Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

[14] J.-P. Serre. *Arbres, Amalgames, SL_2*. Number 46 in Astérisque. Société Mathématique de France, Paris, 1977.

[15] 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.