Decision Problems in Invertible Automata

Evan Bergeron & Klaus Sutner

May 5, 2017

Abstract

We consider a variety of decision problems in groups and semigroups induced by invertible Mealy machines. Notably, we present proof that the automorphism membership problem in decidable in these semigroups. In addition, we prove undecidability of a Knapsack variant. A discussion of iteration and orbit rationality follows.

Contents

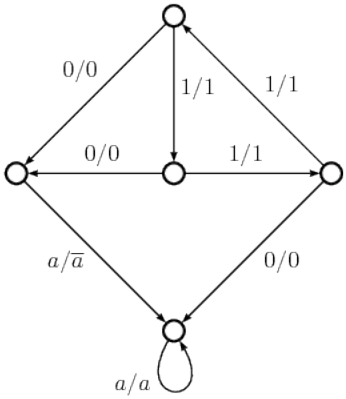


Figure 1: Grigorchuk's 5 state machine

1 Introduction

The word problem is a classic group-theoretic decision problem. Given a finitely generated group G, and a word w over the generators (and their inverses), the word problem asks "is $w \in G$." The word problem is known to be undecidable in surprisingly small classes of groups - see [**?**] and [**?**] for background.

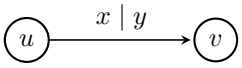
The invertible Mealy machines we consider here give rise to a class of semigroups (and sometimes groups) for which the word problem is decidable. The computability picture here is rather nuanced, however. Similarly important decision problems, among them the conjugacy problem, and the isomorphism problem are known to be undecidable - see [**?**] and TODO for details.

We present proof that, for the Abelian case, automorphism membership testing is decidable in this class of semigroups.

Invertible automata have recently been usefully applied the group theory. A classic result here is Grigorchuk's group of intermediate growth, generated by the 5 state invertible machine shown in figure 1.

2 Background

An *automaton* is a formally a triple (Q, Σ, δ), where Q is some finite state set, Σ is a finite alphabet of *symbols*, and δ is a transformation on $Q \times \Sigma$. Automata are typically viewed as directed graphs with vertex set Q and an edge labeled $x \mid y$ between u, v if $(u, x)\delta = (v, y)$.



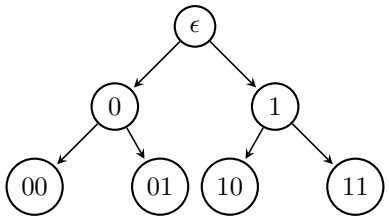
One interprets this as if A is in state u and reads symbol x, then A transitions to state v and outputs

symbol y. A computation within A may then start at some state q_0, and on input $\alpha_0\alpha_1\ldots\alpha_k$, output $\beta_0\beta_1\ldots\beta_k$, where $(q_i,\beta_i) = (q_{i-1},\alpha_i)\delta$ for all $i = 0\ldots k$.

As in the above case, where δ outputs exactly one character for every transition, we call the automaton A *synchronous*. An automaton is called *invertible* when every state in Q has some bijection π on Σ such that $(u,x)\delta = (v,\pi(x))$. A state in A is a *copy state* if π is the identity permutation and is a *toggle state* otherwise. The present paper is concerned only with invertible, synchronous automata.

2.1 Actions on the infinite tree

We may identify the set Σ^* with an infinite, regular tree of degree $|\Sigma|$. The root is labelled with the empty string ϵ, and a vertex labelled w has the child wa for each $a \in \Sigma$. Out of convenience, we will frequently conflate a vertex with its label.



Each state $q \in Q$ acts on the corresponding tree, sending vertex w to wq. Moreover, if $\alpha\alpha'q = \beta\beta'$, then $\alpha q = \beta$, for any $\alpha,\alpha',\beta,\beta' \in \Sigma^*$. Which is to say, q's action on the tree is an adjacency-preserving map and is thus an endomorphism on the tree. Since A is synchronous, q is length-preserving, and thus preserves levels of the tree (and thus is an automorphism of the tree).

We extend the action of Q on Σ^* to words $q = q_1\ldots q_n$ over Q^+ by

$$wq = (\ldots((wq_1)q_2)\ldots q_n).$$

This computation corresponds with running A starting at state q_1, then taking that output and running it through the machine starting at state q_2, and so on. We adopt the convention of applying functions from the right here. In this way, function composition corresponds naturally with string concatenation.

So there is a natural homomorphism $\phi : Q^+ \to \mathsf{End}(B^*)$, where $\mathsf{End}(B^*)$ denotes the semigroup of endomorphisms of the tree B^*. We denote the image of ϕ by $\Sigma(A)$.

Semigroup theory

A *semigroup* is a set S paired with a binary operation $f : S \times S \to S$ such that S is closed under f and f is associative over S. Any set of endofunctions forms a semigroup under composition.

A semigroup is called *Abelian* when its corresponding binary operation is commutative.

For an automaton A, we denote by $S(A)$ the semigroup generated by Q under composition. A is said to be *commutative* or *Abelian* when $S(A)$ is Abelian. We write $G(A)$ for the group generated by the elements of Q and their inverses.

One may also speak about $S(A)$ and $G(A)$ without explicit reference to an automaton A. As such, we call a semigroup S an *automaton semigroup* if there is some automaton A with $S \simeq \Sigma(A)$. *Automaton groups* G are similarly defined.

Wreath Recursions

Any automorphism f of Σ^* can be written in the recursive form:

$$f = (f_{\alpha_1}, f_{\alpha_2}, \ldots, f_{\alpha_n})\tau$$

where $n = |\Sigma|$ and each f_α is an automorphism of a subtree of the root. Here, τ is some permutation on Σ. In the case where $\Sigma = \{0, 1\}$, we have $f = (f_0, f_1)\sigma$ where σ denotes transposition. If $f = (f_0, f_1)\sigma$, f is said to be *odd*. If $f = (f_0, f_1)$, f is said to be *even*. That is to say, automorphisms may be classified as even or odd depending on their action on the first level of the tree.

The endomorphism semigroup of Σ^* decomposes into a recursive wreath product

$$\operatorname{End} B^* = \operatorname{End} B^* \wr \tau_\Sigma$$

where τ_Σ is the tranformation semigroup on Σ. Which is to say,

$$\operatorname{End} B^* = (\operatorname{End} B^* \times \ldots \times \operatorname{End} B^*) \rtimes \tau_\Sigma$$

Decision Problems

Automaton semigroups exhibit many interesting and nuanced computability properties. While it is an easy result that the WORD PROBLEM is solvable in such semigroups, similar group-theoretic problems such as the CONJUGACY PROBLEM and FINITENESS PROBLEM have been shown to be undecidable (see [**?**], and [**?**], respectively).

Various other semigroup theoretic decision problems have recently been considered for small classes of semigroups by Cain in [**?**]. We consider a subset of his distinguished properties in the automaton semigroup case here.

3 Decidable Abelian Automorphism Membership

Given an invertible automaton A and a principal Abelian automaton B, we determine if $f = A(p)$ is in the semigroup generated by B.

Thus one needs to check if there is some product automaton

$$D = B_{p_1} \times B_{p_2} \times \ldots \times B_{p_n}$$

that implements f. We have no computable bound on n, so a priori this merely semidecidable.

Now consider the complete automaton C for B, and let g be the automorphism defined by D. After minimization, D produces a subautomaton of C tat consists of a "transient part" and a copy of B (there may be SCCs in the transient part, but they are not subautomata). Hence, there is some word w such that $\partial_w g$ is just a single state in the copy of B; also, w can be found effectively[1]. In fact, for all u, there is a w such that $\partial_{ww} g$ is atomic. Essentially this just means that g is strongly tame.

We may safely assume that A is minimal. Then is looks like A has to have B as a subautomaton to satisfy this ultimate atomicity condition, plus a transient part sitting on of B. It should be decidable if things match up.

If B is not principal, there are multiple subautomata of C to content with, but that should not make a major difference. Ditto if B is just a random subautomaton of C.

4 Knapsack is Undecidable for Automaton Semigroups

We follow a proof strategy similar to [**?**].

We define the KNAPSACK PROBLEM as follows: given as input generators $g_1 \ldots g_k$ and a target semigroup element g, do there exist natural numbers $a_1 \ldots a_k$ such that

$$g_1^{a_1} \cdots g_k^{a_K} = g$$

We prove that this problem is undecidable for automaton semigroups by reducing from Hilbert's tenth problem.

We define the decision problem HILBERT as following: "given a polynomial over the integers and an integer a, do there exist values of the arguments to the polynomial such that the polynomial evaluated at this point is equal to a?" It is known that there exist polynomials for which this problem is undecidable.

[1] TODO

Claim: KNAPSACK *is undecidable in the class of automaton semigroups.*

We can expand this polynomial into a system of equations - think of a codegen step in a compiler. Each step is either an addition or a multiplication. We can take the terms with negative coefficients and move them to the other side of the equation, so we now have the equality of two different polynomials, each with positive coefficients. We can also choose to only substitute in natural numbers as arguments, by some trick that I don't know. So then we have systems of equations over the natural numbers.

We can turn each equation into a formulation of the Knapsack problem for automaton semigroups. It's known that the Heisenberg semigroup is an automaton group, and there's some equation over the elements of H_3 for multiplication. Same for addition in the natural numbers. Then we just take the direct product of these groups (the class of automaton semigroups is closed under direct product). So this polynomial is equal to a if and only if there exist $a_1 \dots a_n$ such that each of the individual elements of the direct product vectors are equal.

4.1 A group with decidable word problem with submonoid for which the Is-Group question is undecidable

We define the following ambient group: elements of the group represent states of a Turing machine. This Turing machine operates on the empty tape. There is a single halting configuration. In the group, the element corresponding to the halting configuration is the identity. If two group elements represent two configurations of the Turing machine such that one of the configurations can proceed to the other in a single step, then these two group elements are considered equal.

We have the take care defining the Turing machine, however. We'll define it to be a *self-verifying* Turing machine. How does this work? This Turing machine provides some canonical computation, that is, it proceeds from the start configuration onward, perhaps halting. However, there are all sorts of configurations that the Turing machine will never reach. A self-verifying Turing machine will, at every point, verify that it is along this canonical computation path. If it finds that it is not, it will transition to some death state, where it will stay.

This means that, considering the set of all possible configurations of the Turing machine as a graph, there's a path from the start state onward (perhaps ending in a halt state) and everything else is just a star graph transitioning to this death state.

How are these self-verifying Turing machines realized? As the canonical computation proceeds, a program counter is kept (perhaps to the left of the tapehead). After every step, the Turing machine will examine what "time step" the computation is currently sitting in. It will perform the canonical computation for the

first n steps. If it does not wind up where it's configuration says it is, it transitions to the death state. Otherwise, it continues.

Claim: This group has a decidable word problem.

We can verify if one configuration transitions to another in a single step. So given a word over the generators (that represents a sequence of TM configurations), we can first go through pairwise, rewriting these pairs. So I guess chains of consequential configurations just become their first state. We can also go through and check to see if certain configurations lie upon the canonical computational path. So then everything becomes either the death state or the start state or the identity? Then we can just check for equality of exponents?

Claim: If s is the start configuration of the Turing machine, it is undecidable whether $\langle s \rangle$ is a group.

Suppose it was decidable. We can, given, a TM, transform it into a self-verifying version of itself, and then build this group around it. If the original TM halts, then the submonoid generated by s, (the start state) is just the trivial group. If the original TM hangs, then $\langle s \rangle$ is the free monoid of rank one. So then being able to answer the question "is $\langle s \rangle$ a group" would allow us to solve the halting problem.

4.2 It is decidable if an Abelian automaton semigroup generates a group

Reduces to a system of equations. Abelian automaton semigroups can be written as a system of matrix equations: residuation is a linear operation here. We can then also write down the set of matrix equations for the inverse automaton.[2] Exactly what question do we then ask to verify there is a solution? Something about asking if the space spanned by the equations for A has any intersection with $\mathbb{N}^n$.

4.3 Residuation Fixed Point is Decidable for Abelian automaton semigroups

Take the matrix representing residuation for some word $w \in \Sigma^*$ and find if it has any eigenvectors in $\mathbb{N}^n$.

5 Abelian Automata

6 Open Questions

- All automaton semigroups are recursively presented. If these presentations are regular, or context-free, does that affect the soluability of these questions?
- Having a zero

[2]Interesting to note that there's some duality here: if the semigroup of A is a group, then so is the semigroup of A^{-1} (and they are equal).

- Isomorphism problem
- Bounded automata, etc