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, in the Abelian case, 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

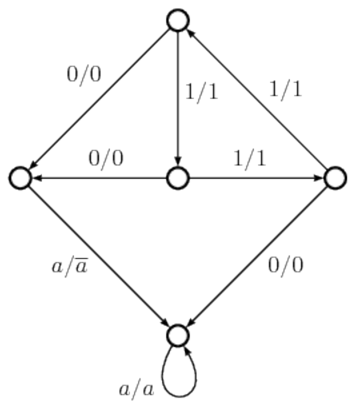


Figure 1: Grigorchuk's 5 state machine

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 [2] and [3] for background.

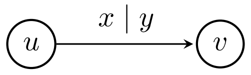
The invertible Mealy machines we consider here give rise to a class of semigroups (and sometimes groups) for which the word problem is decidable. The computability picture here is rather nuanced, however. Similarly important decision problems, among them the conjugacy problem, and the isomorphism problem are known to be undecidable - see [15] 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 $(u, x)\delta = (v, y)$.

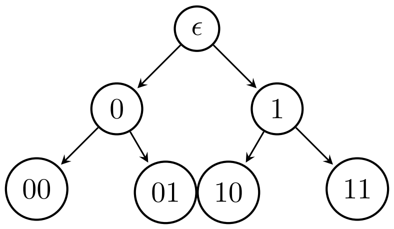


One interprets this as if A is in state u and reads symbol x, then A transitions to state v and outputs symbol y. A computation within A may then start at some state q_0, and on input $\alpha_0\alpha_1 \ldots \alpha_k$, output $\beta_0\beta_1 \ldots \beta_k$, where $(q_i, \beta_i) = (q_{i-1}, \alpha_i)\delta$ for all $i = 0 \ldots k$.

As in the above case, where δ outputs exactly one character forbe 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 $q = q_1 \dots q_n$ over Q^+ by

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

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 $\phi : Q^+ \to \operatorname{End} B^*$, where $\operatorname{End} B^*$ 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:

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

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,

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

3 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 [15], and [6], respectively).

Various other semigroup theoretic decision problems have recently been considered for small classes of semigroups by Cain in [3]. We consider a subset of his distinguished properties in the automaton semigroup case here.

3.1 IsAbelian is polynomial time

We have the following characterization from Okano: $S(A)$ is Abelian iff there exists a unique $\Theta_A \in S(A)$ such that $\partial_0 \underline{t} = \partial_1 \underline{t} \Theta_A$ for all toggle states t. So it suffices to verify that each toggle state obeys this.

Since these transducers are synchronous (output exactly one character per transition), we can build a DFA that recognizes the input-output relation of some fixed starting state. Take some s this start state. Then build a DFA over the alphabet $\Sigma \times \Sigma$ with transitions simulating the behavior of the transducer starting at s. This DFA is called the *acceptor of A at s*.

An automaton semigroup $S(A)$ has a presentation with generators corresponding to the states of A. Thinking of elements of $S(A)$ as words over this generator alphabet, we see that the state set of A is precisely the words of length 1. We can build an automaton whose state set is the set of all length 2 words as follows:

If $A = (Q, \Sigma, \delta)$, the product automaton $A \times A$ has state set $Q \times Q$ with transition function $\partial_a(s_1, s_2) = (\partial_a s_1, \partial_{as_1} s_2)$. We can see by induction that each state (s_1, s_2) corresponds to the word $s_1 s_2 \in S(A)$.

Remember that $(\partial_1 \underline{t}) = (\partial_0 \underline{t})\Theta_A$. So then we have $(\partial_1 \underline{t})(\partial_0 \underline{t})^{-1} = \Theta_A$. (This is straight from Okano's thesis, I think this expression is better written as $(\partial_0 \underline{t})^{-1}(\partial_1 \underline{t}) = \Theta_A$ - it seems more consistent with function-application-from-the-right).

Anyway, $\Theta_A = (\partial_1 \underline{t})(\partial_0 \underline{t})^{-1} = (\partial_1 \underline{t})(\partial_1 \underline{t^{-1}})$, where t^{-1} lies in the inverse automaton for A. So then build for each toggle state the following: $(A \times A^{-1})(\partial_1 \underline{t}, \partial_1 \underline{t^{-1}})$. Note that the language of this automaton is $\{x : y \mid y = x(\partial_1 \underline{t})(\partial_0 \underline{t})^{-1}\}$. It suffices to then verify that all of these constructed DFAs are equivalent, which can be done via Hopcroft's minimization algorithm.

So to summarize algorithmically: take the input automaton A. Build it's inverse automaton A^{-1}. Construct the product automaton $A \times A^{-1}$. Then for each toggle state t_i of A, take the state $s_i = (\partial_1 t_i, \partial_1 t_i^{-1})$ in $A \times A^{-1}$ and construct the DFA $(A \times A^{-1})(s_i)$. Verify all the constructed DFAs are equivalent.

This product automaton construction also gives us that the word problem for automaton semigroups is decidable.

3.2 Automorphism Membership

3.2.1 Linear algebraic background

TODO complete automata etc.

Tsutomu did a good job of summarizing a lot of the important stuff - use it.

3.2.2 Membership is decidable in the Abelian case

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.

3.2.3 Membership is open in the general case

3.3 IsGroup

3.3.1 IsGroup is decidable in the Abelian case

3.3.2 IsGroup is open in the general case

3.4 Knapsack is undecidable for automaton semigroups

We follow a proof strategy similar to [9].

We define the Knapsack Problem as follows: given as input generators $g_1 \dots g_k$ and a target semigroup element g, do there exist natural numbers $a_1 \dots a_k$ 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

[1]TODO

evaluated at this point is equal to a?" It is known that there exist polynomials for which this problem is undecidable.

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 $a_1 \ldots a_n$ such that each of the individual elements of the direct product vectors are equal.

We reduce from Hilbert's tenth problem, over natural numbers. Fix a polynomial $P(x_1, \ldots x_n)$ such that the question "is there a solution to $P(x_1, \ldots x_n) = a$" is undecidable.

Take P and separate it into P_+ and P_-, the positive and negative parts of P. Consider the equation $P_+ = P_-$.

3.5 A monoid with decidable Word Problem and undecidable IsGroup

We establish the existence of a monoid with decidable Word Problem, but undecidable IsGroup. We may take this result as an intermediate step toward the decidability of the IsGroup problem for automaton semigroups.

Preliminaries

Here we take a *Turing machine* to be a 6-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q, Σ, Γ are all finite sets. $\delta : Q \times \Sigma \to Q \times \Gamma \times \{L, S, R\}$ is the transition function, Q is the state set, Σ is the input alphabet, $\Gamma \supseteq \Sigma$ is the tape alphabet, b is some blank symbol, with $b \in \Gamma - \Sigma$, q_{accept} is the unique accepting final state, and q_{reject} the single rejecting final state.

We further define a Turing machine *configuration* to be a triple $(u, q, v) \in \Gamma^* \times Q \times \Gamma^*$. Here, u denotes the tape contents to the left of the tapehead, q is the current state, and v begins at the tapehead and extends to the right.

A configuation C for a TM M is said to *yield* configuration C' if M can step directly from C to C'.

For a Turing machine M, take C_M to be the set of all valid configurations of M. Then define CG_M to be the graph (C_M, E), where $(u, v) \in E$ if and only if u yields v in M. Define

$$\mathsf{canon}(M, w) : \mathcal{T} \times \Sigma^* \to C_M^* \cup C_M^\omega$$

to be the function that maps input w to the sequence of configurations M takes on while computing over w. $\mathsf{canon}(M, w)$ will be a finite sequence if and only if M halts on w. We call $\mathsf{canon}(M, w)$ the *canonical computation of* M *on* w.

Certainly, not every configuration in C_M will be along the sequence $\mathsf{canon}(M, w)$. Which is to say, there are unreachable configurations.

Informally, a *self-verifying Turing machine* S is one that, at every step, verifies that the current configuration lies upon the canonical computation. If S finds that this is not the case, S immediately rejects. Otherwise, the computation steps forward a single step.

In the configuration graph CG_S, there is a path extending from each valid starting configuration (ϵ, q_0, w) for $w \in \Sigma^*$. The remaining states form an infinite star graph with q_{reject} as the center.

Claim: There is a computable[2] function sv that maps Turing machines to equivalent self-verifying Turing machines.

Speaking informally, 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.

In the interest of reader intuition, we expound further upon a couple of implementation details here. For further reading, see [4], [5], and [14] for details.

[2]TODO is actually primitive recursive

TODO rough sketch of implementation details.

The Submonoid in question

Define the Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ to operate only on the blank tape; for all $s \in \Sigma$, $\delta(q, s) = (q_{reject}, b, S)$

Then take the ambient Abelian group $G_M = (C_M, \cdot)$ whose carrier set is all configurations of M. For c, c' in G_M, we have $c = c'$ if and only if c yields c'.

Claim: $G_{\mathsf{sv}(M)}$ has a decidable word problem.

For every word w in $G_{\mathsf{sv}(M)}$, there exist nonnegative integers a, r such that $w = q_{accept}^a q_{reject}^r$. Further, we may compute a and b. Recall that $\mathsf{sv}(M)$ maintains a program counter p to the left of the input. So we may simply run M for the first p steps and then check for configuration equality.

So then given two words w_1, w_2 in $G_{\mathsf{sv}(M)}$, simply compute a_1, r_1, a_2, r_2. Return true if and only if $a_1 = a_2$ and $r_1 = r_2$.

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

It is well known that the following language is undecidable

$$\textsc{halts} = \{\langle M \rangle \mid \text{TM } M \text{ halts on } \epsilon\}$$

and so we reduce from HALTS. Given as input a TM M, we use an oracle for IsGroup as follows: first, compute $\mathsf{sv}(M)$, and then consider $G_{\mathsf{sv}(M)}$. Let s be the starting configuration for M on ϵ.

If M halts, then the submonoid generated by s is the trivial group. If M hangs, then $\langle s \rangle$ is the free monoid of rank one. So then $\langle s \rangle$ is a group if and only if M halts. Since $\mathsf{sv}(M)$ and M are equivalent, we are done.

4 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

- Isomorphism problem
- Bounded automata, etc

References

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

[2] A. J. Cain. Automaton semigroups. *TCS*, 410(47–49):5022–5038, 2009.

[3] A. J. Cain and V. Maltcev. Decision problems for finitely presented and one-relation semigroups and monoids. *International Journal of Algebra and Computation*, 19(6):747–770, 2009.

[4] M. Davis. A note on universal turing machines. *Annals of Mathematics studies*, 34:167–175, 1956.

[5] M. Davis. The definition of universal turing machine. *Proceedings of the American Mathematical Society*, 8:1125–1126, 1957.

[6] Pierre Gillibert. The finiteness problem for automaton semigroups is undecidable. *CoRR*, abs/1304.2295, 2013.

[7] T. Godin. Knapsack problem for automaton groups. *HAL*, 2016.

[8] O. Kharlampovich, B. Khoussainov, and A. Miasnikov. From automatic structures to automatic groups. *ArXiv e-prints*, July 2011.

[9] D. Konig, M. Lohrey, and G. Zetzsche. Knapsack and subset sum problems in nilpotent, polycyclic, and co-context-free groups. *Contemporary Mathematics*, 2015.

[10] Y. Muntyan. *Automata Groups.* PhD thesis, Texas A&M University, May 2009.

[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] T. Okano. *Invertible Binary Transducers and Automorphisms of the Binary Tree.* PhD thesis, Carnegie Mellon University, May 2015.

[14] J. C. Shepherdson. Machine configuration and word problems of given degree of unsolvability. *Z. f. Math. Logik u. Grundlagen d. Mathematik*, 11:149–175, 1965.

[15] Z. Sunic and E. Ventura. The conjugacy problem in automaton groups is not solvable. *Journal of Algebra*, 364(148–154), 2012.

[16] K. Sutner. Invertible transducers, iterations and coordinates. *TODO*, 2013.

[17] K. Sutner and K. Lewi. Iterating inverse binary transducers. *Journal of Automata, Languages, and Combinators*, 17(2–4):293–313, 2012.