# An algebraic view of topological $\epsilon$-machines

Luke Grecki
Graduate Group in Applied Mathematics
lgrecki@math.ucdavis.edu

June 8, 2010

# Contents



**Abstract**

In algebraic automata theory(AAT), automata are represented as algebraic objects, and algebraic methods are used to characterize them. In this paper we use the basic ideas of AAT to construct a new perspective on topological $\epsilon$-machines. In our first section we define the algebraic structure of a semigroup and briefly review some ideas used in the study of semigroups. We then introduce the basic concept of AAT, the semigroup of an automaton, and give some examples to illustrate this idea. In our third section we make a correspondence between special sets of $\epsilon$-machine words and algebraic structures in its semigroup, which allows us to form an algebraic picture of the $\epsilon$-machine. In our fourth section we extend the picture of the previous section by introducing hierarchies of words, which allows us to represent the process of synchronization in a new way. In our final section we summarize and point out directions for future work.

# 1 Semigroups

Semigroups are among the simplest structures studied in algebra. They arise frequently in modern mathematics and are fundamental objects in algebraic automata theory.

**Definition 1.** *A* semigroup *is a tuple $(S, \cdot)$ where $S$ is a set and $\cdot$ is a binary operation on $S$. This operation is closed, so for any two elements $s$ and $t$ in $S$ we have $s \cdot t \in S$. It is also associative, so the identity $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ holds.*

To keep the notation simpler we will sometimes write $s \cdot t$ as $st$, and refer to $S$ as the semigroup when the operation is understood.

An element $e \in S$ is called an *identity element* if $se = es = s$ for all $s \in S$. If an identity element exists it must be unique. For if $e$ and $e'$ are two identity elements we have $e = ee' = e'$. By requiring that a semigroup contain an identity element we get a slightly more complex algebraic structure.

**Definition 2.** *A* monoid *is a semigroup that contains an identity element.*

We will now consider some important examples of these two structures.

**Example 1.** *Let $\Sigma$ be a non-empty set. A string or a word over $\Sigma$ is any finite sequence of elements from $\Sigma$. Denote the set of all words by $\Sigma^+$. We can define a binary operation on $\Sigma^+$ by concatenating words. That is, if $\sigma_0 \dots \sigma_m$ and $\sigma'_0 \dots \sigma'_n$ are in $\Sigma^+$, we define*

$$\sigma_0 \dots \sigma_m \cdot \sigma'_0 \dots \sigma'_n \equiv \sigma_0 \dots \sigma_m \sigma'_0 \dots \sigma'_n$$

*One can check that $\Sigma^+$ is a semigroup with respect to this operation. By adding the empty word $\lambda$ to $\Sigma^+$ we get $\Sigma^*$, where concatenation with $\lambda$ is defined by $w \cdot \lambda = \lambda \cdot w = w$ for $w \in \Sigma^*$. Then $\lambda$ is an identity element and $\Sigma^*$ is a monoid. $\Sigma^+$ and $\Sigma^*$ are called the free semigroup and free monoid on $\Sigma$.*

**Example 2.** *Let $X$ be a non-empty set and denote the set of all partial functions from $X$ to $X$ by $\mathbf{PF}(X)$. We can define a binary operation on $\mathbf{PF}(X)$ by restricted function composition:*

$$(f \cdot g)(x) = y \iff f(g(x)) = y$$

*Under this definition $f \cdot g$ is undefined if $g(x)$ is undefined or if $f(g(x))$ is undefined, so $f \cdot g$ is a partial function. One can check that the operation is associative, and hence $\mathbf{PF}(X)$ is a semigroup with this operation. In addition, $\mathbf{PF}(X)$ contains the identity function $1_X$, which satisfies $f \cdot 1_X = 1_X \cdot f = f$ for all $f \in \mathbf{PF}(X)$. Therefore $\mathbf{PF}(X)$ is also a monoid.*

In group theory Cayley's theorem states that every group can be realized as a permutation group. In other words, every group can be realized as a closed collection of invertible functions from a set to itself. Similarly, any semigroup can be realized as a closed collection of functions from a set to itself. The latter structure is known as a transformation semigroup.

**Definition 3.** *A* transformation semigroup *is a pair $(X, S)$ where $X$ is a set and $S$ is a semigroup. There is an action of $S$ on $X$, which is a binary function $X \times S \to X$ denoted by $x \cdot s$. This action satisfies the following two conditions:*

1. $(x \cdot s) \cdot t = x \cdot (st)$

2. $x \cdot s = x \cdot t$ *for all* $x \in X$ *implies* $s = t$

*The first condition requires that the action be compatible with the semigroup structure of* $S$*. The second condition is that the action is faithful, so that each distinct element in* $S$ *defines a unique transformation of* $X$*. From now on we will denote* $x \cdot s$ *by* $xs$*, relying on context to distinguish the action of* $S$ *on* $X$ *from the semigroup operation on* $S$*.*

To finish this section we define ideals, congruence relations, and quotient semigroups.

**Definition 4.** *A subset* $I$ *of a semigroup* $S$ *is called an* ideal *if* $sI \subseteq I$ *and* $Is \subseteq I$ *for all* $s \in S$*.*

**Definition 5.** *Let* $S$ *be a semigroup. A* congruence relation *on* $S$ *is an equivalence relation* $\sim$ *that is compatible with multiplication in* $S$*:*

$$s \sim t \Rightarrow us \sim ut \text{ and } su \sim tu \quad \forall u \in S$$

**Definition 6.** *Let* $S$ *be a semigroup and* $\sim$ *be a congruence relation on* $S$*. Denote the set of all equivalence classes of* $S$ *by* $S/\sim$*. Define a multiplication on* $S/\sim$ *by:*

$$[s] * [t] = [st]$$

*We call* $(S/\sim, *)$ *the* quotient semigroup *of* $S$ *with respect to* $\sim$*.*

## 2 Semigroups of automata

In this section we will see how to associate semigroups to automata. In fact we will associate semigroups to *semi*automata, but informally we will just talk about automata. Before introducing these semigroups we review the definition of a semiautomaton.

**Definition 7.** *A* semiautomaton *is a triple* $(Q, \Sigma, T)$ *where* $Q$ *is a non-empty set called the set of states,* $\Sigma$ *is a non-empty set called the input alphabet, and* $T$ *is a partial function* $T : Q \times \Sigma \to Q$ *called the transition function.*

It will be useful to consider the state mapping $T_\sigma : Q \to Q$ induced by a symbol $\sigma \in \Sigma$. This mapping is defined by

$$qT_\sigma = T(q, \sigma)$$

for all $q \in Q$. We can straightforwardly extend this to consider the state mapping $T_w$ induced by a word $w = \sigma_1 \dots \sigma_k \in \Sigma^+$:

$$qT_w = qT_{\sigma_1} \dots T_{\sigma_k}$$

To obtain the semigroup of an automaton we group words that induce equivalent state transformations. Let $u$ and $w$ be in $\Sigma^+$. Define an equivalence relation $\sim$ on $\Sigma^+$ by:

$$u \sim w \iff T_u = T_w \tag{2.1}$$

One can verify that $\sim$ is a congruence relation on $\Sigma^+$. By taking the quotient of $\Sigma^+$ by $\sim$ we get the semigroup of the automaton.

**Definition 8.** *Let $\mathcal{M} = (Q, \Sigma, T)$ be a semiautomaton. The* semigroup of $\mathcal{M}$ *is* $\mathbf{S}(\mathcal{M}) \equiv \Sigma^+/\sim$, *where $\sim$ is defined by* (2.1). *By including the empty word $\lambda$ and defining its induced state transformation as $T_\lambda = 1_Q$ we can define*[1] *the* monoid of $\mathcal{M}$ *as* $\mathbf{M}(\mathcal{M}) \equiv \Sigma^*/\sim$, *where $\sim$ is extended to $\Sigma^*$ in the natural way.*

By construction each element of $\mathbf{S}(\mathcal{M})$ corresponds to a unique state transformation. Because of this we can define $\mathbf{S}(\mathcal{M})$ in an alternate (but equivalent) way by considering the state transformations directly.

**Definition 9.** *Let $\mathcal{M} = (Q, \Sigma, T)$ be a semiautomaton. The* semigroup of $\mathcal{M}$ *is* $\mathbf{S}(\mathcal{M}) \equiv \{T_w \mid w \in \Sigma^+\}$ *with the binary operation given by function composition:*

$$T_u * T_w = T_{uw}$$

The mapping $[w] \mapsto T_w$ is an isomorphism between the two different representations of $\mathbf{S}(\mathcal{M})$. Thus we can consider elements of $\mathbf{S}(\mathcal{M})$ either as equivalence classes of words or as state transformations, and in the following we will freely switch between these two representations.

Intuitively, $\mathbf{S}(\mathcal{M})$ is a representation of the computation of $\mathcal{M}$. It encodes how the state transformations of $\mathcal{M}$ depend on its input string. It does so abstractly, without any reference to the states of $\mathcal{M}$. To illustrate these concepts we now consider some examples.

**Example 3.** *Let $Q = \{A\}$, $\Sigma = \{0, 1\}$, and $T(A, 0) = T(A, 1) = A$. The semigroup $\mathbf{S}(\mathcal{M})$ is trivial, and contains just a single element corresponding to the identity transformation.*

**Example 4.** *Let $Q = \{q_0, q_1\}$, $\Sigma = \{0, 1\}$, with the transition function specified by Figure 1.*

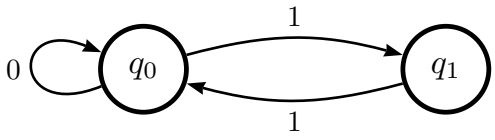


Figure 1: A two state automaton with a two letter alphabet.

*The monoid $\mathbf{M}(\mathcal{M})$ has 7 elements, and we can describe it using Table 1. We notice the element* [010] *has the special property that* $[010] * [w] = [w] * [010] = [010]$ *for all* $[w] \in \mathbf{M}(\mathcal{M})$, *and we say that it is a zero element. In general, an element $z$ in a semigroup $S$ is a* zero element, *or simply a* zero, *of $S$ if $z \cdot s = s \cdot z = z$ for all $s \in S$. Note that if a semigroup has a zero element then, just as in the case of an identity element, this element must be unique.*

*The word* 010 *does not induce any valid state transformation, so $T_{010}$ is just the empty function $0_Q : \emptyset \to Q$. Since we always have $0_Q \cdot T_w = T_w \cdot 0_Q = 0_Q$ we see that* any *forbidden input word must be a zero of $\mathbf{M}(\mathcal{M})$. We explore this and similar observations in the next section.*

## 3 $\epsilon$-machine words and monoid structures

In this section we will be considering *topological* $\epsilon$-machines, and treating them as automata in the sense above. This allows us to associate a monoid[2] to an $\epsilon$-machine. We will show that certain

[1] In some cases $\mathbf{S}(\mathcal{M})$ will already be a monoid, so one does not need to add the empty word to give it this structure.

| * | $\lambda$ | 1 | 0 | 01 | 10 | 010 | 101 |
|---|---|---|---|---|---|---|---|
| $\lambda$ | $\lambda$ | 1 | 0 | 01 | 10 | 010 | 101 |
| 1 | 1 | $\lambda$ | 10 | 101 | 0 | 010 | 01 |
| 0 | 0 | 01 | 0 | 01 | 010 | 010 | 010 |
| 01 | 01 | 0 | 010 | 010 | 0 | 010 | 01 |
| 10 | 10 | 101 | 10 | 101 | 010 | 010 | 010 |
| 010 | 010 | 010 | 010 | 010 | 010 | 010 | 010 |
| 101 | 101 | 10 | 010 | 010 | 10 | 010 | 101 |

Table 1: Multiplication table of $\mathbf{M}(\mathcal{M})$ for Example 4

special sets of $\epsilon$-machine words correspond to algebraic structures in the monoids $\mathcal{A}^*$ and $\mathbf{M}(\mathcal{M})$. Together, these correspondences form a new picture of an $\epsilon$-machine.

It is important to note that what we called input words above are now to be interpreted as output words of the $\epsilon$-machine. To be consistent with the computational mechanics literature we will now use $\boldsymbol{\mathcal{S}}$ to denote the set of states, $\mathcal{S}$ to denote a particular state, and $\mathcal{A}$ to denote the output alphabet. We start by clarifying what we mean by a topological $\epsilon$-machine.

**Definition 10.** *A* topological $\epsilon$-machine *is a semiautomaton* $(\boldsymbol{\mathcal{S}}, \mathcal{A}, T)$ *that has the following properties:*

1. *(***Connected***) For any two states* $\mathcal{S}$ *and* $\mathcal{S}'$ *there exists a word* $w \in \mathcal{A}^*$ *such that* $\mathcal{S}T_w = \mathcal{S}'$ *or* $\mathcal{S}'T_w = \mathcal{S}$*.*

2. *(***Non-halting***) For any state* $\mathcal{S}$ *there exists a letter* $x \in \mathcal{A}$ *such that* $\mathcal{S}T_x$ *is defined.*

Note that unifilarity is automatically guaranteed by defining the topological $\epsilon$-machine as a semiautomaton.

### 3.1 Forbidden words and zeros

A forbidden word is a word that can never be generated by the $\epsilon$-machine. Alternatively, we can say that a forbidden word does not induce any state transformation. Focusing on this latter perspective, we say that a word $w \in \mathcal{A}^*$ is a *forbidden word* if $T_w = 0_{\boldsymbol{\mathcal{S}}}$. It is clear that forbidden words can be extended and the result is still a forbidden word. Using this property we can characterize the forbidden words in terms of $\mathbf{M}(\mathcal{M})$. Before we state this result we need one definition.

**Definition 11.** *Let* $\mathcal{M}$ *be a topological* $\epsilon$*-machine. A state* $\mathcal{S}$ *is called a* sink state *if* $\mathcal{S}T_w = \mathcal{S}$ *for all* $T_w$*, i.e. the only transition from* $\mathcal{S}$ *is a self-loop.*

**Proposition 1.** *Let* $\mathcal{M}$ *be a topological* $\epsilon$*-machine with no sink states. If the set of forbidden words is nonempty it is equal to the unique zero of* $\mathbf{M}(\mathcal{M})$*. If there are no forbidden words then* $\mathbf{M}(\mathcal{M})$ *does not contain a zero.*

[2]When one has a finite $\epsilon$-machine its monoid can also be realized as the monoid generated by its topological symbol transition matrices, with the empty word $\lambda$ corresponding to the identity matrix. Whenever possible we will not use these matrices, so that our results apply to infinite $\epsilon$-machines as well.

*Proof.* First we assume that the set of forbidden words is nonempty and so contains at least one word $z$. We mentioned above that if a zero exists it must be unique, so we just have to show that the set of forbidden words is equal to a zero of $\mathbf{M}(\mathcal{M})$. Since $z$ is forbidden we know $T_z = 0_{\boldsymbol{\mathcal{S}}}$. By definition, the equivalence class $[z]$ contains exactly those words $z'$ for which $T_{z'} = T_z = 0_{\boldsymbol{\mathcal{S}}}$, so $[z]$ must be equal to the set of forbidden words. The unique transformation $0_{\boldsymbol{\mathcal{S}}}$ specified by $[z]$ satisfies $0_{\boldsymbol{\mathcal{S}}} \cdot T_w = T_w \cdot 0_{\boldsymbol{\mathcal{S}}} = 0_{\boldsymbol{\mathcal{S}}}$ for all $T_w$, which implies that $[z]$ is a zero of $\mathbf{M}(\mathcal{M})$.

Now suppose there are no forbidden words. We assume for a contradiction that $[w]$ is a zero of $\mathbf{M}(\mathcal{M})$. Since $w$ is not a forbidden word $T_w$ is not the empty function, and there exists a state $\mathcal{S}$ in the domain of $T_w$. Let $\mathcal{S}' = \mathcal{S}T_w$. Since $\mathcal{M}$ is a topological $\epsilon$-machine it is non-halting, so there exists a $T_u$ such that $\mathcal{S}'T_u$ is defined. Furthermore, the state $\mathcal{S}'$ cannot be a sink state, so we can assume that $\mathcal{S}'T_u = \mathcal{S}'' \neq \mathcal{S}'$. But because $[w]$ is a zero we must have $\mathcal{S}'' = \mathcal{S}T_wT_u = \mathcal{S}T_w = \mathcal{S}'$, which is a contradiction. □

### 3.2 Idle words and identity elements

Idle words complement forbidden words. Whereas no state transformations correspond to forbidden words, only the identity state transformation is induced by idle words. More concisely, we say that a word $w \in \mathcal{A}^*$ is an *idle word* if $T_w = 1_{\boldsymbol{\mathcal{S}}}$. Note that by definition $\lambda$ is an idle word. The following proposition about idle words follows directly from our definition.

**Proposition 2.** *Let $\mathcal{M}$ be a topological $\epsilon$-machine. The set of idle words is equal to the unique identity element of $\mathbf{M}(\mathcal{M})$.*

### 3.3 Synchronizing words and ideals

Seeing a synchronizing word allows one to know exactly the current state of the $\epsilon$-machine. In other words, every synchronizing word corresponds to a state transformation that sends all states to a single state. We call state transformations of this form reset transformations.

**Definition 12.** *A state transformation $T_w : \boldsymbol{\mathcal{S}} \to \boldsymbol{\mathcal{S}}$ is called a* reset transformation *if $|\boldsymbol{\mathcal{S}}T_w| = 1$.*

Using this definition we say that a word $w \in \mathcal{A}^*$ is a *synchronizing word* if $T_w$ is a reset transformation, and denote the set of synchronizing words of a machine $\mathcal{M}$ by $Sync(\mathcal{M})$. The transformation corresponding to a synchronizing word is depicted in Figure 2. Just like for forbidden words, one can extend a synchronizing word and the result (if not forbidden) is still a synchronizing word. We state this algebraically in the following proposition.

**Proposition 3.** *Let $\mathcal{M}$ be a topological $\epsilon$-machine (or any semiautomaton). Define*

$$I_{\mathcal{M}} \equiv \begin{cases} Sync(\mathcal{M}) \bigcup [0_{\boldsymbol{\mathcal{S}}}] & \text{if } 0_{\boldsymbol{\mathcal{S}}} \in \mathbf{M}(\mathcal{M}) \\ Sync(\mathcal{M}) & \text{otherwise} \end{cases}$$

$$I_{\mathbf{M}(\mathcal{M})} \equiv \pi(I_{\mathcal{M}})$$

*where $\pi : \mathcal{A}^* \to \mathbf{M}(\mathcal{M})$ is the canonical projection map given by $\pi(w) = [w]$. Then $I_{\mathcal{M}}$ is an ideal of $\mathcal{A}^*$ and $I_{\mathbf{M}(\mathcal{M})}$ is an ideal of $\mathbf{M}(\mathcal{M})$.*

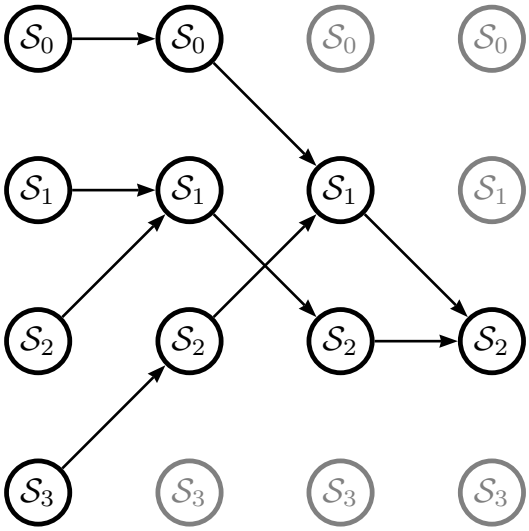


Figure 2: The process of synchronization. Let $w = x_0x_1x_2$ be the synchronizing word. The transitions between the $i$th and $(i+1)$th columns correspond to seeing the $i$th letter $x_i$ of the synchronizing word. The total transformation $T_w$ is a reset transformation sending every state to $\mathcal{S}_2$. Note that one could extend the word in either direction and obtain the same structure as long as the new word is not forbidden.

*Proof.* We start by proving that $I_{\mathcal{M}}$ is an ideal of $\mathcal{A}^*$. Let $w \in I_{\mathcal{M}}$ and $u \in \mathcal{A}^*$. We must show that $uw$ and $wu$ are also in $I_{\mathcal{M}}$. If $w \in [0_{\boldsymbol{\mathcal{S}}}]$ then $uw, wu \in [0_{\boldsymbol{\mathcal{S}}}]$ because $[0_{\boldsymbol{\mathcal{S}}}]$ is a zero of $\mathbf{M}(\mathcal{M})$. Now assume $w \in Sync(\mathcal{M})$. Examining the cardinalities of the relevant transformations we see

$$|\boldsymbol{\mathcal{S}}T_{uw}| = |(\boldsymbol{\mathcal{S}}T_u)T_w| \leq |\boldsymbol{\mathcal{S}}T_w| = 1$$

which implies that $T_{uw}$ is either a reset transformation or $0_{\boldsymbol{\mathcal{S}}}$, and hence shows $uw \in I_{\mathcal{M}}$. For the word $wu$ we can again examine cardinalities

$$|T_{wu}(\boldsymbol{\mathcal{S}})| = |(\boldsymbol{\mathcal{S}}T_w)T_u| \leq |\boldsymbol{\mathcal{S}}T_w| = 1$$

The inequality follows from the fact that the cardinality of the image of a function can never be greater than the cardinality of its domain. Therefore we have $wu \in I_{\mathcal{M}}$.

To prove that $I_{\mathbf{M}(\mathcal{M})}$ is an ideal of $\mathbf{M}(\mathcal{M})$ we note that $\pi$ is a surjective monoid homomorphism. These maps send ideals to ideals, which immediately implies that $I_{\mathbf{M}(\mathcal{M})}$ is an ideal of $\mathbf{M}(\mathcal{M})$. □

At the lowest level, the sets $I_{\mathcal{M}}$ and $I_{\mathbf{M}(\mathcal{M})}$ are made up of the same words. The difference being that in $I_{\mathbf{M}(\mathcal{M})}$ the synchronizing words are grouped according to the state they synch to (and the forbidden words are grouped together as well).

### 3.4 Inert words and filters

In contrast to synchronizing words, inert words preserve uncertainty instead of eliminating it. Starting with a uniform probability distribution over states, seeing an inert word will leave one

with a uniform distribution over states. In order to avoid an overlap of probability mass these transformations must send each state to a distinct state. In light of this, we say that a word $w \in \mathcal{A}^*$ is an *inert word* if $T_w$ is a permutation, and denote the set of all inert words of $\mathcal{M}$ by $Inert(\mathcal{M})$. We depict a transformation corresponding to an inert word in Figure 3.

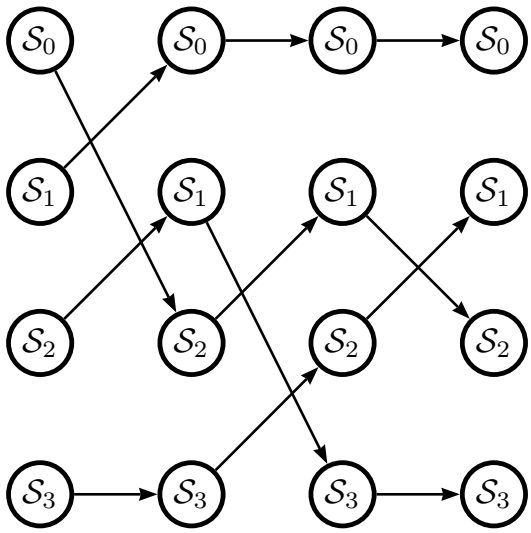


Figure 3: An inert transformation. The transitions are to be interpreted in the sense of Figure 2. We see that the total transformation $T_w$ is a permutation of the states, which implies it will preserve a uniform probability distribution over states. Note that each sub-transformation is a permutation as well.

We observe that an inert word can be extended by another inert word, and the result is again an inert word. In the case of finite machines we additionally notice that $T_w$ can only be a permutation if every sub-transformation is a permuation as well. This is the only way to guarantee that there is no collapse of states that would result in an overlap of probability mass. We describe these facts algebraically below.

**Definition 13.** *A subset $F$ of a language $\mathcal{L}$ is called a* filter *if $F$ is closed under taking subwords.*

**Proposition 4.** *Let $\mathcal{M}$ be a topological $\epsilon$-machine (or any semiautomaton). Then $Inert(\mathcal{M})$ is a submonoid of $\mathcal{A}^*$ and $\pi(Inert(\mathcal{M}))$ is a submonoid of $\mathbf{M}(\mathcal{M})$. If $\mathcal{M}$ is* finite *then $Inert(\mathcal{M})$ is a filter of $\mathcal{A}^*$.*

*Proof.* Let $u, w \in Inert(\mathcal{M})$. Then by definition $T_u$ and $T_w$ are permutations. Since the composition of two permutations is a permutation, $T_{uw}$ must also be a permutation. Hence $uw \in Inert(\mathcal{M})$. Furthermore, $T_\lambda = 1_{\boldsymbol{\mathcal{S}}}$ is a permutation, which implies that $Inert(\mathcal{M})$ is indeed a submonoid of $\mathcal{A}^*$.

To see that $\pi(Inert(\mathcal{M}))$ is a submonoid of $\mathbf{M}(\mathcal{M})$ we recall that the projection $\pi$ is a monoid homomorphism. Since monoid homomorphisms send submonoids to submonoids, we can immediately conclude the desired result.

Finally, we show that if $\mathcal{M}$ is finite then $Inert(\mathcal{M})$ is a filter of $\mathcal{A}^*$. We prove the result in cases.

Case 1: $w = xy$

By definition $T_w = T_x T_y$ is a permutation, meaning that it is a bijection. This implies that $T_x$ must be injective and $T_y$ must be surjective. The domain and codomain of all these functions is the same finite set $\boldsymbol{\mathcal{S}}$, and in this situation being injective or surjective is equivalent to being bijective. Therefore $T_x$ and $T_y$ are also permutations and the subwords $x$ and $y$ are in $Inert(\mathcal{M})$.

Case 2: $w = xyz$

By factoring $w$ as $w = (xy)z$ the above implies that $xy, z \in Inert(\mathcal{M})$. Applying the same result to $xy$ we get that $x, y \in Inert(\mathcal{M})$. Since $y$ is a general subword the proof is complete. □

In some setups the set of inert words is generically trivial. If $w \in Inert(\mathcal{M})$ and $\mathcal{M}$ is finite then Proposition 4 says all subwords of $w$ are also in $Inert(\mathcal{M})$. In particular, this implies that all the letters of $w$ are inert words. If $\mathcal{A}$ is a binary alphabet and $w$ contains a 0 and a 1, then all the letters of $\mathcal{A}$ are inert words. Since the inert words are closed under composition this implies that every word is an inert word. Thus the set of inert words is either mostly uninteresting or is all of $\mathcal{A}^*$. One has to go to larger alphabets for $Inert(\mathcal{M})$ to have more complex structure[3].

Considering Proposition 3 and Proposition 4 together we notice that the synchronizing words and inert words are dual, with $Sync(\mathcal{M})$ being closed under word extension and $Inert(\mathcal{M})$ being closed under word reduction. We return to this type of symmetry in Section 5.

### 3.5 An algebraic picture

Our results in this section allow us to form a new picture of the $\epsilon$-machine in terms of the monoids $\mathcal{A}^*$ and $\mathbf{M}(\mathcal{M})$, which we give below in Figure 4.

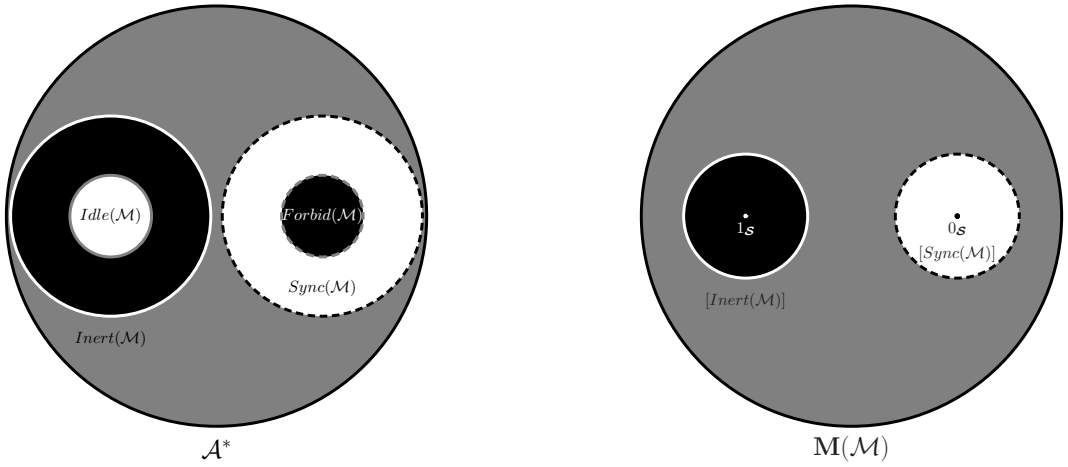


Figure 4: The monoids $\mathcal{A}^*$ and $\mathbf{M}(\mathcal{M})$. A solid boundary indicates that the set is a submonoid while a dashed boundary means that it is an ideal. In the diagram for $\mathbf{M}(\mathcal{M})$ the notation $[W]$ is short for $\pi(W)$.

[3]The fact that $Inert(\mathcal{M})$ claims all words composed of certain letters also creates a tension between the size of $Sync(\mathcal{M})$ and the size of $Inert(\mathcal{M})$.

### 3.6 An example

Consider the machine depicted in Figure 5.

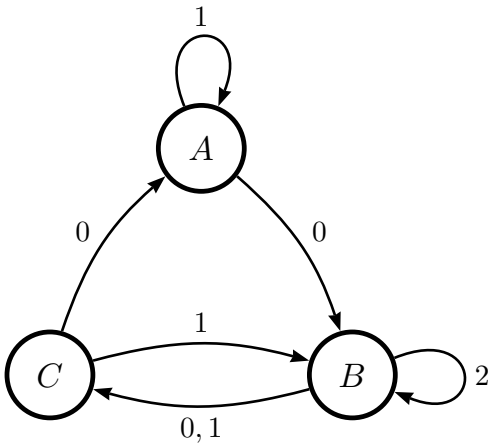


Figure 5: A 3 state topological $\epsilon$-machine with a 3 letter alphabet.

By making some simple observations we can find some of the words in the sets shown in Figure 4. First of all, we notice that $T_0$ and $T_1$ are permutations, which means that 0 and 1 are inert words. By Proposition 4 we then know that all words containing only 0s and 1s must be inert words as well. Next, we observe that seeing a 2 allows us to know with certainty that we are in state $B$. Using Proposition 3, we can conclude that any word containing a 2 is either a forbidden word or a synchronizing word. Lastly, we note that both the set of forbidden and idle words is nonempty. For example, any word containing 202 is forbidden and the word 000 is idle. Therefore every set in Figure 4 is nonempty. We summarize these observations below in Table 2.

| word set | contains |
|---|---|
| $Forbid(\mathcal{M})$ | $\{*202*\}$ |
| $Sync(\mathcal{M})$ | $\{*2*\}$ |
| $Idle(\mathcal{M})$ | $\{(000)^n\}$ |
| $Inert(\mathcal{M})$ | $\{0,1\}^*$ |

Table 2: Distinguished words of the machine in Figure 5.

## 4 Algebraic levels and possibility reduction

In this section we extend the algebraic picture of Figure 4, filling in the mysterious grey area between the sets shown there. We consider two hierarchies of sets of words, each of which is characterized by the state collapse induced by its transformations. The sets we have previously considered appear as levels in these hierarchies, and the levels between them fill in our previous algebraic picture. The finished product is a new representation of $\mathbf{M}(\mathcal{M})$ and the process of synchronization, giving us one way to visualize the reduction of possibility that occurs as observations are made.

### 4.1 Word hierarchies

Noticing the pattern in our definitions of forbidden, synchronizing, and inert words, we procede to group words according to the size of the image of their transformations. The importance of this size is that it is the number of possible states one could be in after seeing the word.

**Definition 14.** *Let $\mathcal{M}$ be a topological $\epsilon$-machine with $n$ states. The* forward word hierarchy $\overrightarrow{\Delta}$ *and the* reverse word hierarchy $\overleftarrow{\Delta}$ *are defined by*

$$\overrightarrow{\Delta} \equiv \{L_k^{\rightarrow} \mid k = 0, \cdots, n\}$$

$$\overleftarrow{\Delta} \equiv \{L_k^{\leftarrow} \mid k = 0, \cdots, n\}$$

*where the levels $L_k^{\rightarrow}$ and $L_k^{\leftarrow}$ are given by*

$$L_k^{\rightarrow} \equiv \{w \in \mathcal{A}^* \,:\, |\boldsymbol{\mathcal{S}}T_w| \leq k\}$$

$$L_k^{\leftarrow} \equiv \{w \in \mathcal{A}^* \,:\, |\boldsymbol{\mathcal{S}}T_w| \geq k\}$$

Explicitly, each level $L_k^{\rightarrow}$ consists of words whose transformations map to at most $k$ states, and each level $L_k^{\leftarrow}$ contains the words whose transformations map to at least $k$ states. The hierarchies are just the collections of these levels. We note that $\mathcal{A}^* = \bigcup_{k=0}^{n} L_k^{\rightarrow} = \bigcup_{k=0}^{n} L_k^{\leftarrow}$ so every word is in a level of the forward hierarchy and a level of the reverse hierarchy. Observe that the levels are nested:

$$\cdots \subseteq L_{k-1}^{\rightarrow} \subseteq L_k^{\rightarrow} \subseteq L_{k+1}^{\rightarrow} \subseteq \cdots$$

$$\cdots \supseteq L_{k-1}^{\leftarrow} \supseteq L_k^{\leftarrow} \supseteq L_{k+1}^{\leftarrow} \supseteq \cdots$$

The sets we have previously considered appear as levels (or sublevels):

$$L_0^{\rightarrow} = Forbid(\mathcal{M})$$
$$L_1^{\rightarrow} = Forbid(\mathcal{M}) \cup Sync(\mathcal{M})$$
$$L_n^{\leftarrow} = Inert(\mathcal{M}) \supset Idle(\mathcal{M})$$

We represent these hierarchies visually in Figure 6. Earlier, we observed that the synchronizing words and the inert words are algebraically dual, having the structure of an ideal and a filter, respectively. It turns out that the levels $L_k^{\rightarrow}$ and $L_k^{\leftarrow}$ share the same relationship.

**Proposition 5.** *Let $\mathcal{M}$ be a topological $\epsilon$-machine. Each forward level $L_k^{\rightarrow}$ is an ideal of $\mathcal{A}^*$, and $\pi(L_k^{\rightarrow})$ is an ideal of $\mathbf{M}(\mathcal{M})$. Each reverse level $L_k^{\leftarrow}$ is a filter of $\mathcal{A}^*$.*

*Proof.* First we show that $L_k^{\rightarrow}$ is an ideal of $\mathcal{A}^*$. Let $w \in L_k^{\rightarrow}$ and $u \in \mathcal{A}^*$. By definition $|\boldsymbol{\mathcal{S}}T_w| \leq k$. Examining cardinalities we see that

$$|\boldsymbol{\mathcal{S}}T_{uw}| = |(\boldsymbol{\mathcal{S}}T_u)T_w| \leq |\boldsymbol{\mathcal{S}}T_w| \leq k$$

which shows that $uw \in L_k^{\rightarrow}$. Similarly we observe

$$|\boldsymbol{\mathcal{S}}T_{wu}| = |(\boldsymbol{\mathcal{S}}T_w)T_u| \leq |\boldsymbol{\mathcal{S}}T_w| \leq k$$

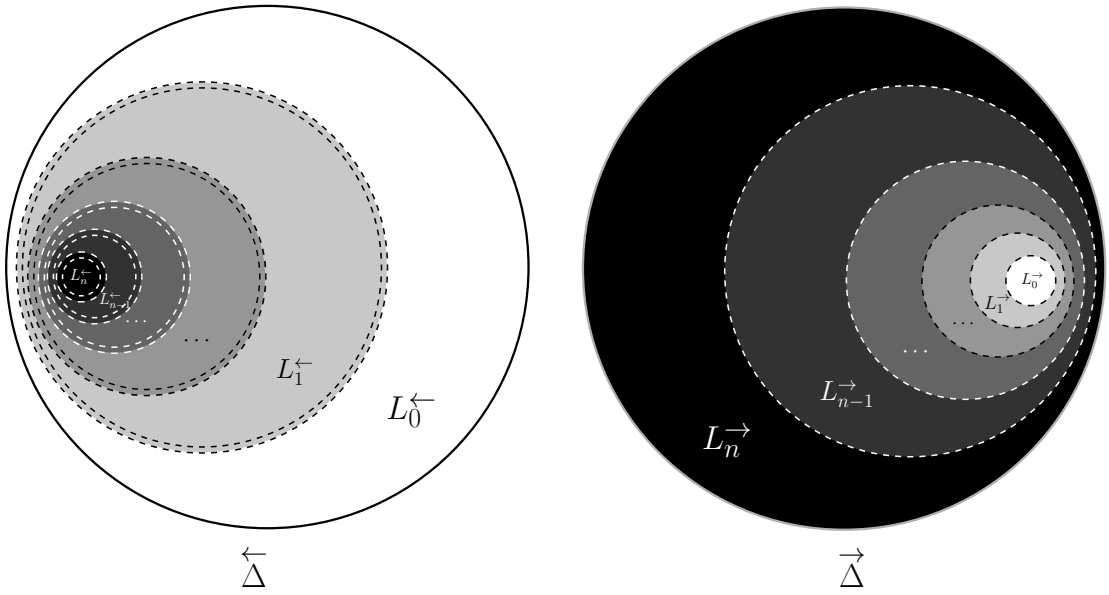


Figure 6: The word hierarchies. A dashed boundary indicates that the set is an ideal while a double dashed boundary means that it is a filter. Each of $\overleftarrow{\Delta}$ and $\overrightarrow{\Delta}$ cover $\mathcal{A}^*$, and the levels in them are nested.

where the first inequality holds since the cardinality of the domain of a function is never smaller than the cardinality of its image. This shows that $wu \in L_k^{\rightarrow}$, and therefore $L_k^{\rightarrow}$ is an ideal of $\mathcal{A}^*$. We noted previously that all monoid homomorphisms send ideals to ideals. Since $\pi$ is a monoid homomorphism we can immediately conclude that $\pi(L_k^{\rightarrow})$ is an ideal of $\mathbf{M}(\mathcal{M})$.

Now consider a word $xy \in L_k^{\leftarrow}$. By definition $|\boldsymbol{\mathcal{S}}T_{xy}| \geq k$ so

$$|\boldsymbol{\mathcal{S}}T_x| \geq |(\boldsymbol{\mathcal{S}}T_x)T_y| = |\boldsymbol{\mathcal{S}}T_{xy}| \geq k$$

so $x \in L_k^{\leftarrow}$. Identically, we see that $y \in L_k^{\leftarrow}$. Finally let $xyz \in L_k^{\leftarrow}$. We can factor this word as $(xy)z$ and by applying the previous result twice we find that $y \in L_k^{\leftarrow}$. Since $y$ is a generic subword we have shown that $L_k^{\leftarrow}$ is closed under subwords, and is therefore a filter. □

The above allows us to understand the $\epsilon$-machine (via both $\mathcal{A}^*$ and $\mathbf{M}(\mathcal{M})$) as a nested collection of algebraic objects. This representation naturally provides us with another perspective on synchronization.

### 4.2 Synchronization

By observing a word letter by letter one can (hopefully) synchronize to the process, sequentially reducing one's uncertainty in the current state of the $\epsilon$-machine. We can understand this procedure as a progression through the levels of the forward hierarchy $\overrightarrow{\Delta}$ that converges to the smallest non-trivial level, that of the synchronizing words. This results in an intuitive picture of the successive reduction in possibility that occurs as observations are made.

For each word $w = x_0 x_1 x_2 \cdots$ we have the sequence of partial words we observe at each time step:

$$x_0 \to x_0 x_1 \to x_0 x_1 x_2 \to \cdots$$

To each of these partial words $w'$ we can assign the smallest level of $\overrightarrow{\Delta}$ that contains it. Explicitly, we map $w'$ into $\overrightarrow{\Delta}$ by

$$L(w') \equiv \bigcap_{w' \in L_k^{\to}} L_k^{\to}$$

which gives us what we want because the levels $L_k^{\to}$ are nested. With this map we get a sequence of levels corresponding to the sequence of observed partial words:

$$L(x_0) \to L(x_0 x_1) \to L(x_0 x_1 x_2) \to \cdots$$

Since the size of the image of $T_{w'x}$ is at most the size of the image of $T_{w'}$, this sequence is monotonically decreasing, i.e. $L(w') \supseteq L(w'x)$. If synchronization is possible the levels will converge to the smallest level of possible words, the level $L_1^{\to}$ of synchronizing words. We illustrate this convergence in Figure 7.

# 5 Conclusion

We started by introducing the algebraic structure of a semigroup, and showed how one could associate a semigroup to an automaton using algebraic automata theory. We then applied these ideas to topological $\epsilon$-machines, and found that certain sets of epistemically important words (the forbidden, synchronizing, idle, and inert words) corresponded to algebraic structures in the monoid of the $\epsilon$-machine. This resulted in our image of the $\epsilon$-machine in Figure 4. In our final section we filled in the previous picture by introducing hierarchies of words, characterized by the degree to which they reduce uncertainty in the state of the $\epsilon$-machine. In this framework we saw that the process of synchronization is naturally represented as a monotonic movement through nested levels of decreasing size.

From here one could pursue a few different research directions. First, one could consider the full probabilistic $\epsilon$-machine. If it is finite its transformations are given by finite stochastic matrices, which form a monoid under multiplication. To avoid matrices with arbitrarily small entries one should normalize after multiplication. The concepts introduced here should easily translate. Instead of considering the size of the image $|\boldsymbol{\mathcal{S}} T_w|$ we can use the conditional entropy $H[\mathcal{S}|w]$ as a measure of state uncertainty given the word $w$. We believe the results here will hold in this new setup. The algebraic picture of synchronization in the probabilistic case should give another perspective on familiar entropy convergence curves.

For the most part the propositions here did not depend on the special properties of topological $\epsilon$-machines and held for arbitrary semiautomata. It would be interesting to see if one can make stronger statements using these properties, and perhaps characterize the optimality of the probabilistic $\epsilon$-machine from this algebraic perspective.

Finally, the ideas introduced here might also be used in studying hierarchical $\epsilon$-machine reconstruction. In the latter procedure there are three processes occuring simulataneously: synchronization, $\epsilon$-machine construction in a model class, and movement from one model class to another. We think our picture of synchronization might inspire a unified way to look at these three processes, and perhaps unite them into one inferential procedure.

## References


[1] M.A. Arbib, K. Krohn, and J.L. Rhodes. *Algebraic theory of machines, languages, and semigroups*. Academic Press, 1968.

[2] S. Eilenberg. *Automata, languages, and machines*. Academic Press, 1976.

[3] M. Holcombe. *Algebraic automata theory*. Cambridge University Press, 2004.

[4] J. Rhodes. *Applications of Automata Theory and Algebra: Via the Mathematical Theory of Complexity to Biology, Physics, Psychology, Philosophy, Games, Codes*. Rhodes, 1971.

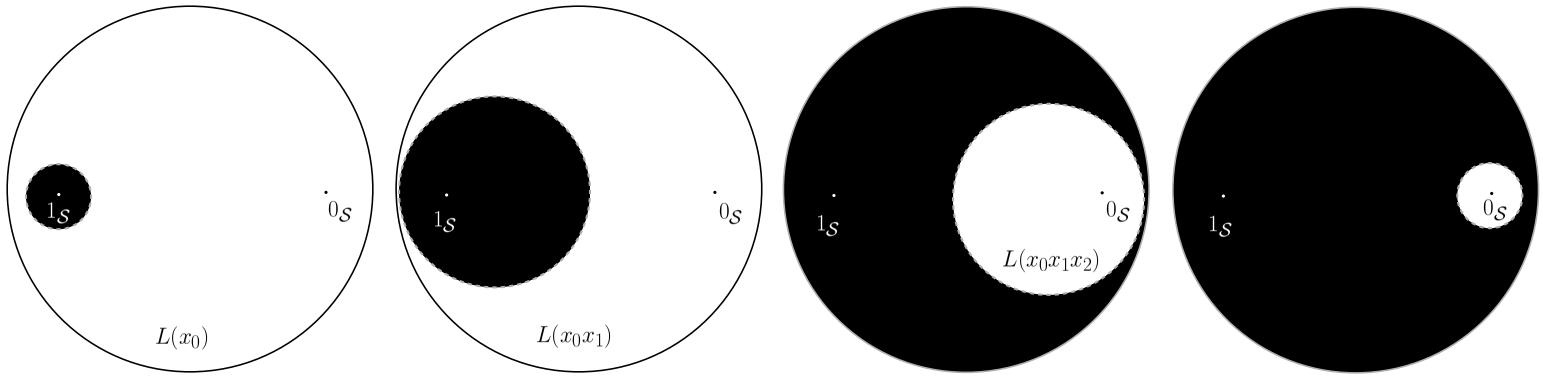


Figure 7: Synchronization as seen in the monoid $\mathbf{M}(\mathcal{M})$. At each time step a letter $x_j$ is observed and the new partial word $w'$ enters a smaller level $L(w')$. Here the white area represents the set $L(w')$ while the black area is *not* contained in the set. At the end of the procedure the level $L(x_0x_1x_2x_3)$ is equal to the set of synchronizing words, and the observer is synchronized to the process. Compare this picture with our static image of $\mathbf{M}(\mathcal{M})$ in Figure 4.