Algebraic Characterization of Regular Languages

Chloe Sheen
Advised by Professor Steven Lindell

A Thesis Presented in Partial Fulfillment of
Bachelor's Degree

Department of Computer Science
Bryn Mawr College

Abstract

This paper presents the proof provided by Marcel-Paul Schützenberger, which connects regular language theory and algebraic automata theory. The two fundamental areas of study in mathematics and theoretical computer science hold a notable relationship, which motivated the discussion surrounding this topic. On a high-level perspective, Schützenberger's theorem bridges the two areas of study using languages recognized by aperiodic monoids to relate them to star-free regular expressions. By outlining the essential ideas in both automata theory and group theory, we attempt to produce a more comprehensible document for both computer scientists and mathematicians to discuss this topic.

Contents

Chapter 1

Introduction

1.1 Background

Mathematics is the foundation of computing, and computing is often a key component in mathematical problem-solving. For instance, linear algebra is heavily used in computer graphics, graph theory forms the basis of network analysis, and number theory is used for cryptography. Such relationship between the two disciplines has inevitably resulted in the shared interest and involvement of both mathematicians and computer scientists in the study of formal languages.

Formal language is composed of strings formed by letters from an alphabet along with a specific set of properties. This concept is often introduced in undergraduate computer science courses for its more practical connections to the field.

This paper will specifically focus on regular languages, which are formal languages that can be expressed using regular expressions. Regular expressions describe the lexical tokens in syntactic specifications of programming languages, describe the textual patterns that trigger processing actions in text manipulation systems, and serve as the basis of standard utilities in pattern matching. Regular language theory provides the theoretical basis of pattern matching utilities including text tools awk, a text-manipulation language, and grep, a Unix utility that searches for lines in files that match a pattern [2]. Beyond its wide range of applications, regular languages are an important part of theoretical computer science.

In theoretical computer science, formal language theory is closely tied

to the study of computability theory and computational complexity. Computability theory studies whether problems are computationally solvable using different representations of algorithms and languages, and computational complexity considers the inherent difficulty of evaluating computational problems [1].

Regular language theory also has connections with areas of mathematics in various fundamental fields including algebra and logic. The two areas of study have been bridged by Marcel-Paul Schützenberger's theorem, which essentially relates languages recognized by aperiodic monoids with star-free regular languages.

1.2 Motivations and Goals

Given the equivalence of regular languages and finite automata as Kleene's theorem [4] states along with the relationship of regular languages and finite algebra, we were motivated to embed regular languages into a pure mathematical system. This paper will explore the algebraic characterization of regular languages in order to

(1) provide computer scientists with a more comprehensible introduction to algebraic automata theory, and to

(2) introduce mathematicians to the area of formal languages.

We believe that with the provided background in automata theory and algebra in this paper, coupled with an explanation of star-free expressions, establishing Schützenberger's theorem to connect the two fields of study will come naturally.

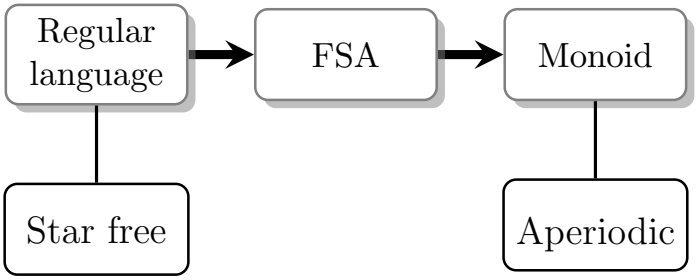
The contents of this paper will be organized into the following structure:

- In Section 2, we provide the background that is required for discussing Schützenberger's theorem.
  - In Section 2.1, we introduce standard notations, definitions, and simple examples of finite automata and regular languages.
  - In Section 2.2, we give several detailed examples of regular languages, their representation in finite automata, and their monoid representations.

  - In Section 2.3, we introduce group theory, provide several detailed examples of mapping strings to monoids, and finally prove that monoids and finite automata are equivalent.

- In Section 3, we state our definition of star-free expressions, ideals of monoids, and provide the proof of Schützenberger's theorem in a way that is more accessible to both mathematicians and computer scientists, which is our major contribution in this work.
- In Section 4, we briefly discuss further readings that may inspire further work with this subject.
- In Section 5, we conclude the work by restating the major points of this paper.

Below is the general diagram that illustrates the sequence of the main points that we will cover in order to successfully unpack Schützenberger's theorem. The idea of aperiodic monoids and star-free expressions will be introduced in section 2.3 and section 3, respectively.



Chapter 2

Notations, Definitions, and Examples

The following subsections introduce the standard notations and definitions this paper will follow to establish Schützenberger's theorem. Each of the definitions and concepts are accompanied by relevant examples.

2.1 Finite Automata and Regular Languages

Finite automata are the simplest computational model. They are used to recognize patterns within input taken from some set of characters by accepting or rejecting an input. Automata bridge the theory of formal language to computation. From a mathematical perspective, we require a more precise, notation-based definition of finite automata. In essence, a finite automaton is composed of several parts: a set of states, an input alphabet, a start state, rules that connect one state to another depending on the input symbol, and accept states. We will use the standard definition of finite automata in this paper:

Definition 1. [8] A *finite automaton* is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

(i) Q is a finite set called the *states*,

(ii) Σ is a finite set called the *alphabet*,

(iii) $\delta : \mathrm{Q} \times \Sigma \to \mathrm{Q}$ is the *transition function*,

(iv) $q_0 \in \mathrm{Q}$ is the *start state*, and

(v) $\mathrm{F} \subseteq \mathrm{Q}$ is the *set of accept states*.

From the five parts of the formal definition, we can use the notation to describe finite automata by specifying each of the components of the 5-tuple.

Example 1. Here is a finite automaton M_1.

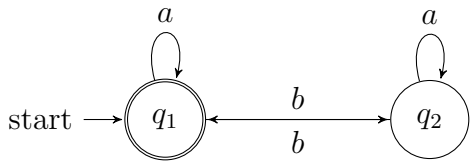


Figure 2.1: State diagram of the two-state finite automaton M_1

In the formal definition, M_1 is $(\{q_1, q_2\}, \{\mathrm{a}, \mathrm{b}\}, \delta, q_1, \{q_1\})$. The transition function δ is

	a	b
q_1	q_1	q_2
q_2	q_2	q_1.

To better understand this machine, we can analyze it with an example input string. Consider the word $w = abab$. The machine starts in the start state q_1, reads the first symbol of w, which is a, and proceeds to q_1. It then reads the next symbol b, proceeds to q_2, reads the next symbol a, stays in q_2, and reads the last symbol b, taking us back to q_1. Since $q_1 \in F$, the automaton *accepts* the word.

We will describe the languages using words in later sections.

Example 2. Figure 2.2 shows another finite automaton M_2.

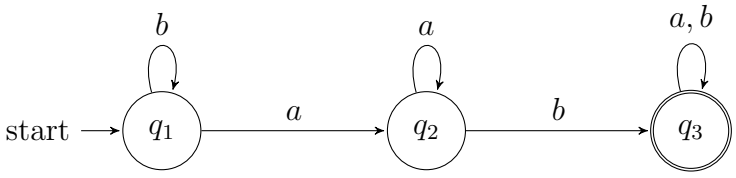


Figure 2.2: State diagram of the three-state finite automaton M_2

Machine M_2 has three states, with a formal definition of ($\{q_1, q_2, q_3\}$, {a, b}, δ, q_1, $\{q_3\}$). The transition function δ is

	a	b
q_1	q_2	q_1
q_2	q_2	q_3
q_3	q_3	q_3.

Take $w = baa$. The machine starts in the start state q_1, reads the first symbol of w, which is b, and stays in q_1. It then reads the next symbol a, proceeds to q_2, and reads the last symbol a, leaving us in q_2. Since $q_2 \notin F$, the automaton *rejects* the word.

In both **Example 1** and **Example 2**, we have given a formal definition of the automata. We can also give a formal definition of the automata's computation in the following way:

Definition 2. [8] Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton and let $w = w_1 w_2 \ ... \ w_n$ be a string where each w_i is a member of the alphabet Σ. Then M *accepts* w if a sequence of states $r_0, r_1 \ ... \ r_n$ in Q exists with three conditions:

(i) $r_0 = q_0$

(ii) $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, \ ... \ , n - 1$, and

(iii) $r_n \in F$.

M recognizes the language A if $A = \{w \mid M \text{ accepts } w\}$. The following is the standard definition of regular language.

Definition 3. Given an alphabet Σ, the collection of *regular languages* is defined by:

- The empty set ϕ and the empty string ϵ are regular.
- Each symbol $a \in \Sigma$ is regular.
- If A and B are regular languages, then $A \bullet B$ (concatenation), $A \cup B$ (union), and A^* (Kleene star) are regular languages

According to Kleene's theorem [4], a language is called a *regular language* if and only if there is some finite automaton that recognizes it. In other words, a FSA has the same expressive power as regular languages. We can revisit the previous examples to demonstrate this theorem.

In **Example 1**, we can also run the machine on an input word $w = aba$. This input leaves M_1 in q_2, so it is rejected. The machine will generally accept only strings that have an even number of b's. Thus, this automaton is simply another representation of the regular language $L(M_1) = \{w \mid w \text{ has an even number of } b\text{'s}\}$. Likewise, we can conclude that **Example 2** is a representation of the language $L(M_2) = \{w \mid w \text{ contains a substring } ab\}$.

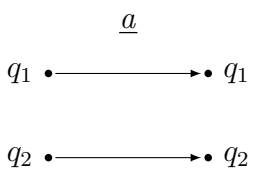
2.2 Detailed Examples

To begin our construction of monoids from regular languages, we first pose examples that illustrate each step of the process.

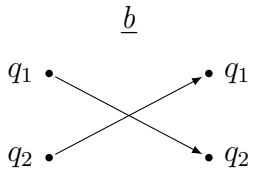
Example 3. Suppose we want to find the functional representation of the regular language from **Example 1**, $L(M_1) = \{w \mid w \text{ has an even number of } b\text{'s}\}$. Recall that the state transition table looked like this:

	a	b
q_1	q_1	q_2
q_2	q_2	q_1.

We can also sketch arrow diagrams to represent the relationship between the states:



(a) mapping given the input a



(b) mapping given the input b

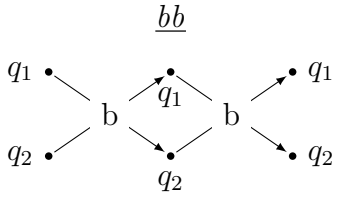


Figure 2.4: mapping given the input bb

Follow the arrows in the arrow diagram illustrating the states given input string bb. Notice that q_1 ends up back at q_1 and q_2 ends up back at q_2. This essentially holds the same relationship as taking the input string a. Thus, mathematically, we can derive the equation $b^2 = a = e$.

We now introduce a new way of representing this relationship: a *monoid multiplication table*. Multiplying each of the items in the table's rows by each of the items in the table's columns produces results that match the set of equations.

	a	b
a	a	b
b	b	a

Monoids are the simplest algebraic object used to represent a finite state machine. The following is a definition of monoids derived from its relationship to semigroups:

Definition 4. [3] A *semigroup* is a nonempty set S together with a binary operation on S which is associative: $a(bc) = (ab)c$ for a, b, $c \in G$. A *monoid* is a semigroup M which contains an identity element $e \in M$ such that $ae = ea = a$ for all $a \in M$.

Note that in the example above, a acts like the identity. Thus, we can rewrite the monoid multiplication table to the following:

	e	b
e	e	b
b	b	e.

A more detailed definition and discussion of monoids will be presented in *Section 2.3*.
The following is the only monoid of size 2:

	e	a
e	e	a
a	a	a.

Notice that there are no identities (e) in the table other than when $e \cdot e$. This is an informal definition of aperiodic monoids, which will be discussed in later sections. Here is another example of constructing a monoid from a regular language.

Example 4. Derive a functional representation of the regular language of binary modulus of 3.
This language contains all binary strings where the numerical value of the string is a multiple of 3. For instance, the word $w_1 = 1001$ is 9 in decimal, and since 9 mod 3 = 0, 1001 mod 3 = 0. We say that w_1 is in the language. The word $w_2 = 1000$ is 8 in decimal, and since 8 mod 3 = 2, 1000 mod 3 = 2. We say that w_2 is not in the language. Start with sketching the finite state automaton.

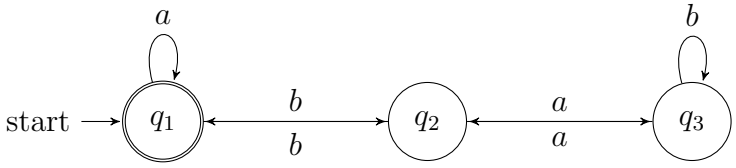
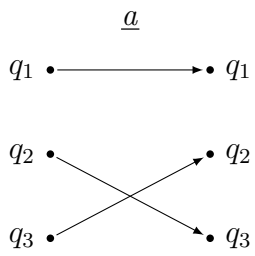
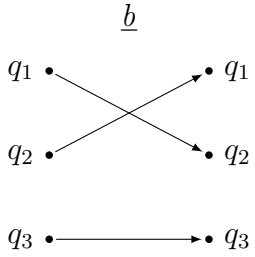


Figure 2.5: State diagram of the three-state finite automaton M_3

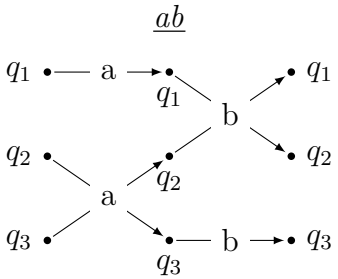
Here are some of the arrow diagrams that we can construct from this finite automaton:



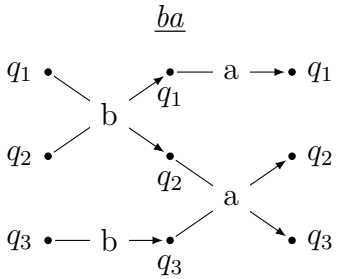
(a) mapping given the input a



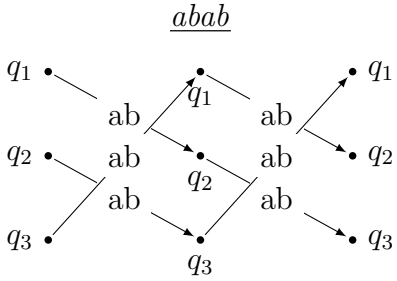
(b) mapping given the input b



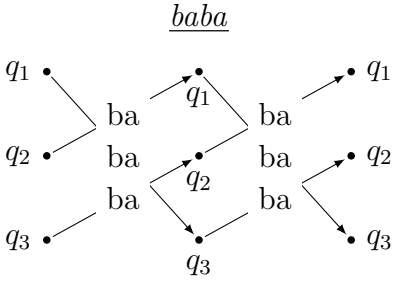
(a) mapping given the input ab



(b) mapping given the input ba



(a) mapping given the input $abab$



(b) mapping given the input $baba$

We can notice that the arrow diagram for $abab$ produces the same result as the arrow diagram for ba. Thus, we can create the equation $abab = ba$. Likewise, $baba = ab$. Clearly, creating more arrow diagrams makes deriving the set of equations much simpler. Here are more equations that represent this diagram:

$a^2 = b^2 = I = abba = baab$
$a = abb = bba = abaab$
$b = baa = aab$

$aba = bab$
$ab = baba = abaa$
$ba = abab.$

Again, we can now represent this with a monoid multiplication table and we conclude our example of creating a functional representation of the regular language of binary modulus of 3:

	e	a	b	ab	ba	aba
e	e	a	b	ab	ba	aba
a	a	e	ab	b	aba	ba
b	b	ba	e	bab	a	ab
ab	ab	aba	a	ba	e	b
ba	ba	b	bab	e	ab	a
aba	aba	ab	ba	a	b	e

Example 5. Take the regular language of strings that contain any number of a, an even number of b, and no c and provide a functional representation.

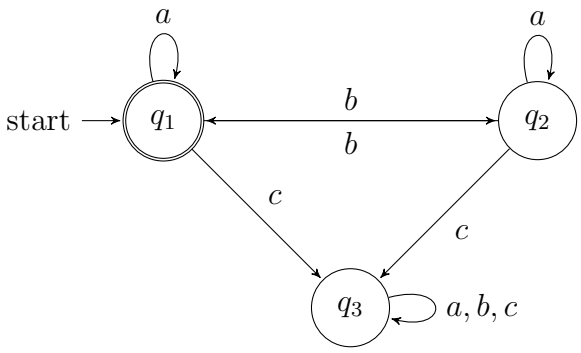
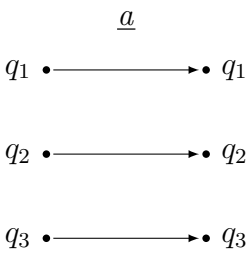
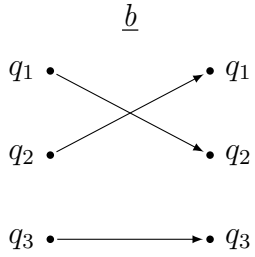


Figure 2.9: State diagram of the three-state finite automaton M_4

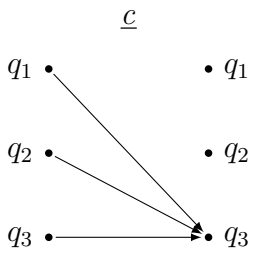
Again, here are the arrow diagrams that will help formulate the set of equations:



(a) mapping given the input a



(b) mapping given the input b



(c) mapping given the input c

Notice that taking the function a leads a state back to itself, implying that it is an identity function. The equations formed are:

$a = e = b^2$, which behaves like the identity (1).
$c = c^2$, which behaves like a zero (0).

A simple monoid multiplication table is thus formed:

	a	b	c
a	a	b	c
b	b	a	c
c	c	c	c.

Example 6. We conclude this section with another example, this time using a nondeterministic finite automaton. Take the regular language $L = \{a\}$ and provide a functional representation.

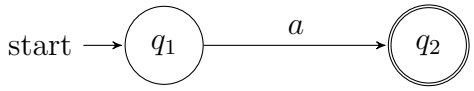
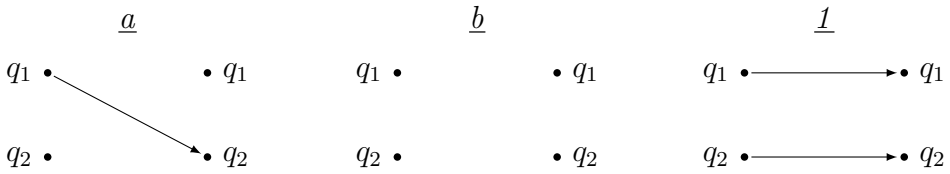


Figure 2.11: State diagram of the two-state finite automaton M_5

Construct the arrow diagrams that will help formulate the set of equations for these partial functions where $\Sigma = \{a, b\}$:



(a) mapping given the input a

(b) mapping given no input

(c) mapping given the identity function

Try aa on the arrow diagram,

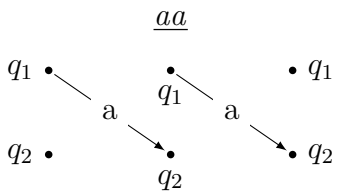


Figure 2.13: mapping given the input aa

to note that it produces the same output as the states given the input string a. Thus, we conclude that $a = a^2$.

A simple monoid multiplication table can be formed, where $1 = e$, $m = a$, and $0 = b$:

	e	a	b
e	e	a	b
a	a	b	b
b	b	b	b.

2.3 Algebra: Group Theory

The algebraic portion of this paper will focus on ideas and concepts mostly from Group Theory.

First, recall the definition of monoids from section 2.2.:

> A *semigroup* is a nonempty set S together with a binary operation on S which is associative: $a(bc) = (ab)c$ for $a,\ b,\ c \in S$. A *monoid* is a semigroup M which contains an identity element $e \in M$ such that $ae = ea = a$ for all $a \in M$ [3].

Building on this definition, we introduce the notion of groups.

Definition 5. [3] A *group* is a monoid G such that for every $a \in G$ there exists an inverse element $a^{-1} \in G$ such that $a^{-1}a = aa^{-1} = e$.

In simpler terms, a group is a set equipped with a binary operation which combines any two elements to form a third element such that four conditions are satisfied: closure, associativity, identity, and invertibility. One common example of a group is the set of integers $\mathbb{Z}$ which consists of the numbers $\{..., -4, -3, -2, -1, 0, 1, 2, 3, 4, ...\}$, with the binary operation, addition. We can show that addition does satisfy the conditions given for it to be considered a group:

(1) Closure: For any two integers $a,\ b \in \mathbb{Z}$, $a + b$ is also an integer.

(2) Associativity: For all integers a, b and c, $(a + b) + c = a + (b + c)$.

(3) Identity: If $a \in \mathbb{Z}$, then $0 + a = a + 0 = a$.

(4) Invertibility: For every $a \in \mathbb{Z}$, there exists an integer b such that $a + b = b + a = 0$. Such integer b is called the inverse element of a and is denoted $-a$.

A *finite group* is a group with a finite number of elements.

Definition 6. A *homomorphism* is a map between two structures such that the operations of one structure is preserved in the other.

Homomorphisms are primarily used in mathematics to show that two different structures behave similarly. In this context, we use homomorphisms between algebraic structures (monoids) and languages.

Definition 7.

- Given two semigroups S_1, S_2, a homomorphism between S_1 and S_2 is a function h: $S_1 \to S_2$ such that for all x, $y \in S_1$, $h(xy) = h(x)h(y)$.
- Σ^* is the finitely generated free monoid over Σ.
- Given an alphabet Σ and a monoid M, a homomorphism between Σ^* and M is a function h: $\Sigma^* \to M$ such that for all x, $y \in \Sigma^*$, $h(xy) = h(x)h(y)$. Here, xy is the concatenation of x and y in Σ^* and $h(x)h(y)$ is the product of $h(x)$ and $h(y)$ in M. M also contains the identity element, $h(\varepsilon)$.

Given this definition of a homomorphism, we can now relate regular languages and subsets of Σ^* using homomorphisms.

Definition 8. A language $L \subseteq \Sigma^*$ is *regular* if and only if there exists a finite monoid M, a subset $F \subseteq M$, and a homomorphism h: $\Sigma^* \to M$, such that $L = h^{-1}(F)$.

Definition 9. A regular language L is *aperiodic* if given a finite aperiodic monoid M, $L = h^{-1}(F)$ for $F \subseteq M$.

Example 7. Consider the empty language $L = \phi$. This can be represented by a state diagram:

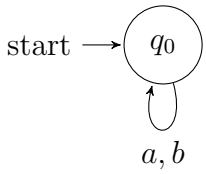


Figure 2.14: Finite state automata for the given language L

The simple arrow diagram depicting the states is produced as follows:

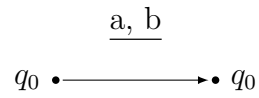


Figure 2.15: states given the input a or b

Note from the arrow diagram that the monoid is simply $M = \{e\}$, as it takes any string in Σ^* to its identity. Using the definition of homomorphisms as previously discussed, we can now map the given language to the monoid to show that the two structures are equivalent. The homomorphism is h: Σ^* $\rightarrow M = \{e\}$, where M is finite such that $L = h^{-1}(\phi)$. L is regular and is recognized by any monoid M.

Example 8. Consider this simplest nontrivial example. Recall the language from **Example 6**, $L = \{a\}$. From the monoid multiplication table, we can derive the homomorphism h: $\Sigma \rightarrow M$, where $h(\epsilon) = 0$, $h(a) = e$, and $h(b) = 0$. M is finite such that $L = h^{-1}(a)$.

With the concepts and definitions presented in this section, we can now describe the notion of aperiodicity. In Schützenberger's theorem, we will specifically focus on aperiodic monoids. We propose the following, more thorough, characterization of monoids here.

Definition 10. Given a finite monoid M, M is *aperiodic*, if

$\forall a \in M, \exists n \in \mathbb{Z}^+$ such that $a^n = a^{n+1}$.

Theorem 1. *Given a finite monoid M, the following are equivalent.*

(1) M contains no non-trivial groups, i.e., $\forall G \subseteq M$, if G is a group, then $G = \{e\}$.

(2) M is aperiodic.

(3) No non-trivial element of M is invertible, i.e., $\nexists\, m \neq e$ such that m^{-1} exists.

Proof. $\boxed{(1) \Rightarrow (2)}$ If M contains no non-trivial groups, then M is aperiodic. We will prove this by showing that $\neg(2) \Rightarrow \neg(1)$.

Suppose M is periodic. By definition, $\forall n \in M$, $\exists m \in M$ such that m^{n+1} $\neq m^n$ $(*)$. Since M is finite, by the pigeonhole principle, $m^{i+p} = m^i$ for some

smallest index $i \geq 0$, period $p \geq 1$. Because of $*$, we know that $p \neq 1$.
$\therefore p > 1$, and M does contain a non-trivial group.

Consider the collection of elements: $G = \{m^j : i \leq j < i + p\} = \{m^i, m^{i+1}, ..., m^{i+p-1}\}$, where $m^i = m^{i+p} = m^{i+2p} = ...$.

We can show that G is a group with the following conditions satisfied:

(i) Closure: First show that if $k \geq i$, then $m^k \in G$. Using the quotient remainder theorem, we can derive the following:

$$k - i = qp + r, \text{for } 0 \leq r < p.$$

Then, we can use algebra to derive:

$$\begin{aligned} m^k = m^i m^{k-i} &= m^i m^{qp+r} \\ &= m^{i+qp} m^r && \text{by associativity} \\ &= m^{i+r}, \end{aligned}$$

and since $r < p$, we have shown that m^k is in G. Now, to show closure, we have $m^{j_1} \cdot m^{j_2} = m^{j_1+j_2}$, and $j_1 + j_2 \geq i$.

(ii) Identity: Using the quotient remainder theorem again, we can let i - p - $1 = qp + r$, where if $r = 0$, $qp < i + p$, and if $r = p$ - 1, $i \geq qp$. Thus, $i \leq qp < i + p$ and we can conclude that $m^{qp} \in G$.

$$\begin{aligned} m^j m^{qp} &= (m^{j-i} m^i) m^{qp} \\ &= m^{j-i}(m^i m^{qp}) && \text{by associativity} \\ &= m^{j-i} m^{i+qp} \\ &= m^{j-i} m^i \\ &= m^j. \end{aligned}$$

(iii) Invertibility: First show that $(m^{qp+p-1})^j \cdot m^j = m^q p$.

$$\begin{aligned} (m^{qp+p-1})^j \cdot m^j &= m^{jqp+pj-j+j} \\ &= (m^{qp})^j m^{pj-j+j} \\ &= m^i m^{qp-i} \cdot m^{pj} \\ &= m^{qp}. \end{aligned}$$

Further, G is not the identity because since M is periodic, we know $p > 1$, and thus there is at least one element in G.

$\boxed{(2) \Rightarrow (3)}$ If M is aperiodic, then the only invertible element of M is the identity.

In other words, suppose $\forall\ m \in M$, $\exists n \in \mathbb{Z}^+ \parallel m^n = m^{n+1}$. Suppose an inverse element m^{-1} exists. Then,

$$\begin{aligned} m^n &= m^{n+1} \\ m^n(m^{-1})^n &= m^{n+1}(m^{-1})^n \\ e &= m. \end{aligned}$$

$\boxed{(3) \Rightarrow (1)}$ If no non-trivial element of M is invertible, then M contains no non-trivial groups.

Given a subset A of a monoid M, the submonoid of M generated by A is the smallest submonoid of M containing A. It is denoted by $<A>$ and consists of all products $a_1 \ldots a_n$ of elements of A and the identity of M. If m is invertible, then $<m, m^{-1}> = \{m^k\colon\ k \in \mathbb{Z}\}$. This is clearly a group. Further, if $G = \{e\}$, $m = e$.

$\boxed{(2) \Rightarrow (1)}$ If M is aperiodic, M contains no non-trivial groups.

Note that this direction of the proof is not necessary.

Let G be a group of M. $G \subseteq M$, take any $g \in G$. Then, by aperiodicity,

$$\exists n \in \mathbb{Z}^+ \text{ such that } g^n = g^{n+1} \qquad \text{by (2).}$$

but g has an inverse. Then,

$$\begin{aligned} (g^{-n})(g^n) &= (g^{-n})(g^{n+1}) \\ (g^{-n+1}) &= (g^{-n+n+1}) \\ e &= g. \end{aligned}$$

□

Given this explanation of aperiodicity in algebra, we can conclude that the monoid defined by the language in **Example 8** is aperiodic. Notice that $0^0 = 0^1 = 0^2$, $1^0 = 1^1 = 1^2$, and $a^0 = a^1 = a^2$.

Chapter 3

Schützenberger's theorem

The discussion of automata and algebra in Chapter 2 leads up to our ultimate goal of this paper, establishing Schützenberger's Theorem.

Definition 11. Given an alphabet Σ, a *star-free regular expression* has the following recursive definition:

- ϕ (the empty set), ϵ (the empty string), and every $a \in \Sigma$ are star-free regular expressions,
- If r_1 and r_2 are star-free regular expressions, then $(r_1 + r_2)$, $\overline{r_1}$ and $(r_1 r_2)$ are star-free regular expressions.

Each star-free regular expression r over Σ defines a language $\mathrm{L}(r) \subseteq \Sigma^*$ as follows:

- $\mathrm{L}(\phi) = \phi$.
- For every $a \in \Sigma \cup \{\epsilon\}$, L(a) = a.
- If r_1 and r_2 are star-free regular expressions, then $\mathrm{L}((r_1 + r_2)) = \mathrm{L}(r_1) \cup \mathrm{L}(r_2)$, $\mathrm{L}(\overline{r_1}) = \overline{L(r_1)} = \Sigma^*$ - $\mathrm{L}(r_1)$ and $\mathrm{L}((r_1 r_2)) = \mathrm{L}(r_1)\mathrm{L}(r_2)$. [5]

Definition 12. [5] Let M be a monoid. We say that $I \subseteq M$ is an *ideal* of M if $IM \subseteq I$ and $MI \subseteq I$. This can be re-written as $MIM \subseteq I$ or equivalently $MIM = I$, since $e \in M$. Note that ideals are always non-empty.

Definition 13. [5] Let M be a monoid and I a nonempty ideal of M. Define the structure $<M/I\ ,\cdot>$ whose elements are $(M \setminus I) \cup \{0\}$, where $0 \notin S$, and whose binary operation $\cdot$ is defined as follows:

- $x \cdot 0 = 0 \cdot x = 0$, *if* $x \in (M \setminus I) \cup \{0\}$.
- $x \cdot y = 0$, *if* $xy \in I$.
- $x \cdot y = xy$ *otherwise. (i.e.* $x, y, xy \in M \setminus I$*).*

In the construction of the proof of Schützenberger's Theorem, we would ideally obtain a homomorphism that maps the structure M to M/I. Intuitively, we will be collapsing the ideal I to *0*.

Theorem 2. *If M is an aperiodic monoid and I is an ideal of M, then M/I is an aperiodic monoid.*

Proof. We already know that *0* is the identity of M/I by definition. The only other property that it needs to satisfy is whether $\cdot$ is associative. These are the cases to be considered:

- Case 1: $x, y, z \in M \setminus I$.
  $(x \cdot y) \cdot z = (xy)z = x(yz) = x \cdot (y \cdot z)$, by definition of $\cdot$ in M.
- Case 2: $x = 0$.
  $(x \cdot y) \cdot z = 0 \cdot z = 0$. Here, if $yz \in I$, $y \cdot z = 0$ and thus $x \cdot (y \cdot z) = x \cdot 0 = 0$. Else, $y \cdot z \in M \setminus I$, and thus $x \cdot (y \cdot z) = 0$.
- Case 3: $y = 0$.
  $(x \cdot y) \cdot z = 0 \cdot z = 0 = x \cdot 0 = x \cdot (y \cdot z)$.
- Case 4: $z = 0$ follows the same proof as Case 2.

Given that M is aperiodic, the property $x^{n+1} = x^n$ also applies to $x \in M \setminus I$. We can now derive a homomorphism given an aperiodic monoid M and an ideal I of M

$$h\colon M \longrightarrow M/I.$$

which is the identity on the elements of M - I and maps every element of I to *0*. □

Definition 14. Let m be an element of a monoid M. Then, if the set

$$I_f = \{m \in M\colon m \notin MmM\}$$

is not empty, it is called the *forbidding ideal* of m. Otherwise, we say f does not have a forbidding ideal.

In other words, the forbidding ideal is a set of all elements that cannot generate m via multiplication. We will show that the forbidding ideal is indeed an ideal.

Proof.

$$M \underbrace{I_f} M \subseteq I_f$$
$$\underbrace{M\underline{m_1}}_{M}\underline{m}\underbrace{\underline{m_2}M}_{M} \subseteq I_f$$

Then, if $x \in Mm_1m\ m_2M$, then $x \in MmM$. Further, $f \notin I_f$ by definition.

□

Proof. Assume M contains a zero. We will show that $I_f = \phi$ if and only if $f = 0$.

Recall that $I_f = \{m \in M\colon m \notin MmM\}$.

If $f = 0$, $I_0 = \{m \in M\colon 0 \notin MmM\}$. Since we assume that M contains a zero, 0 is always contained in MmM. Thus, $I_0 = \phi$. If f $\neq$ 0, $0 \in I_f$. Then, by definition, $I_f = \{0\}$, and thus $I_f \neq \phi$.

□

Theorem 3. *[7] **Schützenberger's Theorem** A language is star-free if and only if it is aperiodic. (★)*

Proof. [⇒] We will show that all star-free languages are recognized by aperiodic monoids. Example 7 and 8 from Chapter 2 serve as base cases; the empty language $L = \phi$ which is recognized by the any finite monoid $M = \{e\}$ with $\phi \subseteq e$ and $h^{-1}(\phi) = \phi$, and the simplest nontrivial example language $L = \{a\}$ which is recognized by the aperiodic monoid illustrated in **Example 6**. In that example, we can further note that the base case is $F = \{m\}$, and when given $\sigma \in \Sigma^*$, $h(a) = m$ and $h(\sigma) = 0$ if $\sigma \neq a$. Further, the forbidding ideal of this language is $I_0 = \phi$, $I_1 = \{0, m\}$, and $I_m = \{0\}$.

To fully establish this direction of the proof using structural induction, we will demonstrate that the union, the complement, and the concatenation of star-free languages recognized by finite aperiodic monoids are also recognized by finite aperiodic monoids.

Union: Given two ★ languages, their union is ★.
Let L_1, $L_2 \subseteq \Sigma^*$ where,

$$L_1 = h^{-1}(F_1) \qquad L_2 = h^{-1}(F_2)$$

$$h_1 : \Sigma^* \to M_1 \supseteq F_1 \qquad h_2 : \Sigma^* \to M_2 \supseteq F_2.$$

Then, the direct product of the two monoids $M = M_1 \times M_2$ is an aperiodic monoid ($\triangle$) and the homomorphism h: $\Sigma^* \to M_1 \times M_2$, where $h(w) = (h_1(w), h_2(w))$ recognizes all combinations of L_1 and L_2 and $F = \{(f_1, f_2) : f_1 \in F_1$ **or** $f_2 \in F_2\}$. To show that the intersection of two finite aperiodic monoids is also aperiodic, we would simply use this definition and redefine $F = \{(f_1, f_2) : f_1 \in F_1$ **and** $f_2 \in F_2\}$.

We will first verify that this is in fact a homomorphism, then prove the validity of the statement $\triangle$.
Recall **Definition 7**, which essentially states that to verify whether h is a homomorphism, we need to show:

$$h[(f_1, g_1) * (g_2, h_2)] = h(f_1, g_1) * h(f_2, g_2).$$

Proposed homomorphism: h: $\Sigma^* \to M_1 \times M_2$, where $h(w) = (h_1(w), h_2(w))$
Let w = xy. Then,

$$\begin{aligned} h(xy) &= (h_1(xy) \quad , \quad h_2(xy)) && \text{definition of } h \\ &= (h_1(x)h_1(y) \ , \quad h_2(x)h_2(y)) && h_1 \text{ and } h_2 \text{ are homomorphisms} \\ &= (h_1(x), h_2(x))(h_1(y), h_2(y)) && \text{by } \textbf{Definition 7} \\ &= \quad h(x) \quad \cdot \quad h(y) && \text{definition of } h \end{aligned}$$

$\therefore$ h is a homomorphism.
Now, we will show that $M = M_1 \times M_2$ is an aperiodic monoid. First, to show that it is a monoid,

$$\mathrm{w} \in h^{-1}(F_1 \times F_2) \Leftrightarrow \mathrm{h(w)} \in F_1 \times, F_2$$

$$\begin{aligned}
&\Leftrightarrow h_1(\mathrm{w}) \in F_1 \ \&\ h_2(\mathrm{w}) \in F_2 \\
&\Leftrightarrow \mathrm{w} \in h_1^{-1}(F_1) \ \&\ \mathrm{w} \in h_2^{-1}(F_2) \\
&\Leftrightarrow \mathrm{w} \in L_1 \ \&\ \mathrm{w} \in L_2 \\
&\Leftrightarrow \mathrm{w} \in L_1 \cap L_2 \\
\therefore\ & h^{-1}(F_1 \times F_2) = L_1 \cap L_2.
\end{aligned}$$

Suppose $M_1 \times M_2$ has an invertible element. This implies that $\exists$ (m_1, m_2) $\in M$ such that (m_1, m_2) $\neq$ (e, e) and $(m_1, m_2)^{-1} \cdot (m_1,\ m_2)$ = (e, e) by **Theorem 1**. Let $(m_1, m_2)^{-1}$ = (a, b). Since $M = M_1 \times M_2$, this produces:

$$\begin{aligned}
&am_1 = e, \\
&m_1 a = e,
\end{aligned}$$

which implies that if m_1 is invertible in M_1, $m_1 = e$, and

$$\begin{aligned}
&bm_2 = e, \\
&m_2 b = e,
\end{aligned}$$

which implies that if m_2 is invertible in M_2, $m_2 = e$.
$\therefore$ M_1 and M_2 are aperiodic.

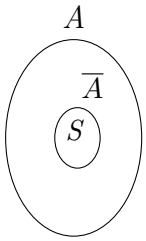
Clearly, (m_1, m_2) is the identity, and thus M is aperiodic.

Complement: Given a ★ language, its complement is ★.

Let $L \subseteq \Sigma^*$ and $F \subseteq M$ where,

$$\begin{aligned}
&L = h^{-1}(F) \\
&\overline{L} = h^{-1}(\overline{F}) \text{ where } \overline{F} \subseteq M.
\end{aligned}$$

We begin by visualizing the complement of the monoid M.



$$f : A \to \overline{A}$$

We will use the same homomorphism as used in *Union*: the direct product of the two monoids $M_1 \times M_2$ is an aperiodic monoid ($\triangle$) and the homomorphism h: $\Sigma^* \to M_1 \times M_2$, where $h(w) = (h_1(w), h_2(w))$ recognizes all combinations of L_1. Let S be some element of the monoid A. Then,

$$h^{-1}(\overline{A} - S) = A - h^{-1}(S).$$

Concatenation: Given two ★ languages, their comcatenation is ★.

This proof follows the intuition that the word w can be split into two strings x and y. For instance, given the word $w = abb$, we can have the following arrangements of the word into two split strings:

x	y
ϵ	abb
a	bb
ab	b
abb	ϵ

Let L_1, $L_2 \subseteq \Sigma^*$ be languages recognized by finite aperiodic monoids.

$$L_1 = h^{-1}(F_1) \qquad L_2 = h^{-1}(F_2)$$

$$h_1 : \Sigma^* \to M_1 \supseteq F_1 \qquad h_2 : \Sigma^* \to M_2 \supseteq F_2$$

Then, the powerset of the direct product of the two monoids $M_1 \times M_2$ is an aperiodic monoid and we have the homomorphism h: $\Sigma^* \to \mathcal{P}(M_1 \times M_2)$, where $h(w) = \{(h_1(x), h_2(y))$: $xy = w\}$ and $F = \{f : f \cap F_1 \times F_2 \neq \phi\}$.

We will verify that $\mathcal{P}(M_1$ x $M_2)$ is a finite monoid.

Let S, $T \subseteq M_1 \times M_2$. We need to show that three conditions are met: closure, associativity, and identity (refer to **Definition 5**). $S \cdot T = \{$st : s $\in S$, t $\in T\}$ shows closure as per our definition, s and t are pairs of elements in the monoids. Take R, another element in $M_1 \times M_2$. Then, $R \cdot S \cdot T = \{$(rs)·t = r·(st): s $\in S$, t $\in T$, r $\in R\}$ to show associativity. The identity element is simply $\{(e_1, e_2)\}$. Further, we know that we still have a finite monoid because the size of the monoid is at most $2^{|M_1| \cdot |M_2|}$.

[⇐] We will show that all finite aperiodic languages are star-free. Essentially, we will be taking the inverse homomorphic image of an finite aperiodic language to derive a star-free language.

$$\text{star-free } L \subseteq \Sigma^* \underset{h^{-1}}{\overset{h}{\rightleftarrows}} \text{aperiodic } M \supseteq F$$

In order to use induction on the size of the monoid for this direction of the proof, we will focus on a decrease and conquer approach by taking quotients of the monoids. In particular, we will divide the monoid by an object that is not the identity monoid. Recall **Definition 12** for the definition of an ideal; dividing a monoid by the ideal would satisfy this condition. In **Theorem 2**, we showed that dividing a monoid by the ideal produces an aperiodic monoid and that there exists a homomorphism h: $M \longrightarrow M/I$. Thus, if I_f is an ideal of a size greater than or equal to 2, any language that can be recognized by I_f can be recognized by a smaller monoid, M/I.

For any $F = f_1, f_2, ..., f_k$, $h^{-1}(F)$ is the union of the sets $h^{-1}(f_1)$, $h^{-1}(f_2)$, ... , $h^{-1}(f_k)$. Thus, we can show that $h^{-1}(f)$ can be expressed as a star-free expression with languages definable using aperiodic monoids that are smaller than M. Let $L \subseteq \Sigma^*$ be a language where given a finite aperiodic monoid M, $L = h^{-1}(\mathrm{F})$ for some $F \subseteq M$. If F is the trivial monoid ($L = \phi$ or Σ^*), both are clearly star-free languages. We can represent L as the following:

$$L = \bigcup_{f \in F} h^{-1}(f).$$

Star-free languages are closed under union, and thus there is some $f \in F$ such that $L = h^{-1}(\mathrm{f})$. Since I_f is an ideal, if I_f has two or more elements, we can use the monoid M/I_f to derive the homomorphism h: $M \rightarrow M\ /\ I_f$ where $L = h^{-1}(x)$. If I_f has no elements ($I_f = \phi$), we have shown in a previous proof that $f = 0$. The case if I_f has one element ($I_f = \{0\}$) is detailed in Mallea's sketch of the proof [5].

□

Chapter 4

Further Readings

The readings below further provide different approaches to introducing and proving Schützenberger's theorem. This paper in particular aims to cover the proof of the theorem in a more accessible way, and thus leaves out some of the algebraic concepts that are not necessary but are helpful and supplementary information for the development of the proof.

1. *A Proof of Schützenberger's Theorem*

   Alejandro Mallea's work most closely echoes the structure of this paper. His work provides further definitions and lemmas that are beyond the scope of how we presented the theorem, using more algebraic concepts in his approach to the proof. It would be helpful for one to refer to this paper if they are seeking for a more definition-oriented reading versus an example-oriented approach to introducing concepts.

2. *Mathematical Foundations of Automata Theory*

   This work by Jean-Éric Pin is essentially a comprehensive textbook covering the depths of what one would wish to learn about the algebraic foundations of automata theory. It is a rather dense read, considering that it is a textbook primarily composed of a lot of concepts presented through definitions and mathematical proofs. However, it would be a great resource for those interested in the more in-depth research on the topic.

Chapter 5

Conclusion

Coupled with the **Further Readings** presented in the previous section, this paper serves as a comprehensive yet simple guide to further exploration for mathematicians and computer scientists into the study of formal languages. This work primarily presents readers with a much more comprehensible and organized introduction to Schützenberger's theorem. Through the process, the paper presents readers with both fundamental knowledge and examples of automata theory along with the basis of group theory required to understand at least the higher-level proof of the theorem. This exploration of regular languages and its applications in algebra also poses an interesting algorithm in decidability; given a regular language L, it is now decidable to determine whether L is star-free. Further, if the language is star-free, we can now derive a star-free regular expression representation of L.

Bibliography

[1] Book, Ronald V. *Formal language theory and theoretical computer science* In: Brakhage H. (eds) Automata Theory and Formal Languages 2nd GI Conference Kaiserslautern Lecture Notes in Computer Science, vol 33. Springer, Berlin, Heidelberg, May 20–23, 1975.

[2] Brüggemann-Klein, Anne *Regular Expressions into Finite Automata* Elsevier, Theoretical Computer Science, pp. 197-213, 1993.

[3] Hungerford, Thomas W. *Algebra* Springer-Verlag New York, Inc., 1974 .

[4] Kleene, Stephen Cole *Representation of Events in Nerve Nets and Finite Automata* Princeton University Press, Princeton, N.J., 1956.

[5] Mallea, Alejandro *A Proof of Schützenberger's Theorem* Pontificia Universidad Católica de Chile, Santiago, Chile, 2012.

[6] Pippenger, Nick *Theories of Computability* Cambridge University Press, 1997.

[7] Schutzenberger, M.P. *"On Finite Monoids Having only Trivial Subgroups"* Information and Control 8 190-194, 1965.

[8] Sipser, Michael *Introduction to the Theory of Computation* Verlag Nicht Ermittelbar, 2013.