Formal Power Series Representation of Regular Languages: How Infinite Sums Make Languages Ring

Melanie Jensenworth
Math 336
June 3, 2010

Contents:

1. Abstract

At first glance, one might not think that the mathematical field of algebra and computer science are particularly related. After all, algebra deals with details of sets and their operations and computer science deals with writing programs, right? If you focus on the basis of computer science, however, you will see that it involves a lot of algebra. Computer programs have to be computable to work properly, and that requires that you write in the proper language - syntax, etc. The theoretical abstractions make languages easier to manipulate with mathematics and - guess what - are basically *sets with operators*. In the first two sections we will cover the necessary algebra and apply it to the set of languages to form a semiring. In the third section we introduce formal power series and how they become a semiring under addition and Cauchy product. In the fourth section, we prove that the semiring of languages is isomorphic to the semiring of formal power series. Hopefully this introduction will fascinate people enough that they pursue the topic further, into the (harder) textbooks on the subject. It certainly did so for me.

2. Background

Unfortunately, as is typical, each field has developed its own distinct notation. For the purposes of this paper, I will use a consistent notation so that readers may more easily see the parallels between the two approaches.

2.1 Algebra

Algebra is typically used to describe systems of operators and the sets they operate on. Because of this, algebra forms the backbone of this topic. For this paper, we will need the following basic definitions.

Definition 2.1.1 A *monoid* $< M, \bullet, \varepsilon >$ is a set M, a product operator "$\bullet$", and an identity element ε, such that:

for any $a, b, c \in M$,

(1) $a \bullet b \in M$ (closure)

(2) $a \bullet (b \bullet c) = (a \bullet b) \bullet c$ (associativity)

(3) $\varepsilon \bullet a = a \bullet \varepsilon = a$ (identity element)

Note: Since the operator "$\bullet$" looks like multiplication and can be quite similar, the symbol is often dropped and the elements of M are simply written next to each other. That is, $a \bullet b = ab$ and $a \bullet (b \bullet c) = a(bc)$.

Definition 2.1.2 A monoid is *commutative* if $ab = ba$ for all $a, b \in M$.

For example, $\langle \mathbb{N},\ \text{multiplication}, 1 \rangle$ is a commutative monoid because the natural numbers are closed under multiplication, multiplication is associative and commutative, and 1 times any number is the original number.

Definition 2.1.3 The *star* of set X, denoted X^*, is the set of all possible finite length sequences of zero or more elements of X, where the finite length sequence is constructed using the product operator defined by a monoid, and the new monoid is called a *free monoid*.

Definition 2.1.4 A *semiring* $< A, +, \bullet, 0, 1 >$ is a set A, two operators, and their respective identity elements such that for all $a, b, c \in A$

(i) $< A, +, 0 >$ is a commutative monoid

(1) $a + b \in A$ (closure)
(2) $a + (b + c) = (a + b) + c$ (associativity)
(3) $0 + a = a + 0 = a$ (identity element)
(4) $a + b = b + a$ (commutative)

(ii) $< A, \bullet, 1 >$ is a monoid *(again the dot is usually represented by placing elements of A side-by-side)*

(1) $ab \in A$ (closure)
(2) $a(bc) = (ab)c$ (associativity)
(3) $1a = a1 = a$ (identity element)

(iii) $a(b + c) = ab + ac$ (distributivity)
(iv) $(a + b)c = ac + bc$ (distributivity)
(v) $0a = a0 = 0$ (zero element)

2.2 Language Theory

Computer science has developed complex methods to describe the languages that can be recognized by various types of machines. Of course, every machine has a mathematical basis, so the basic ideas are similar to familiar mathematical concepts. For this paper, we will need the following definitions.

Definition 2.2.1 An *alphabet* Σ is a nonempty, finite set, whose elements are the letters of the alphabet.

Normally, each letter of an alphabet is written as one character. Two particularly common alphabets are $\Sigma = \{a, b, c\}$ and $\Sigma = \{0, 1\}$.

Definition 2.2.2 A *string* is a finite sequence of letters. The *empty string* ε is a string containing zero characters.

Definition 2.2.3 The *concatenation* of two strings $a, b \in \Sigma$, denoted by *ab*, is an associative operator that converts the two strings into one string by making the first letter of *b* follow the last letters of *a*.

Note that because ε is a string, $\varepsilon \notin \Sigma$ for any Σ.

Definition 2.2.4 The length of a string *w*, denoted $|w|$, is the number of letters in *w*. Each letter is counted as many times as it occurs.
Ex. $|\varepsilon| = 0$. $|abaac| = 5$. $|aaaa| = 4$.

This means letters are essentially strings of length one.

Lemma 2.2.5 The empty string is the identity element for the operation of concatenation.
Proof: To show that ε is the identity for concatenation for an alphabet Σ, we must show that $\varepsilon a = a\varepsilon = a$ for any string *a* over Σ. However, this is true because ε contains no letters and has zero length, so it changes neither defining characteristic of the string *a*, whichever side it was added to, so $\varepsilon a = a\varepsilon = a$.
□

As in definition 2.1.3, the starred set Σ^* is the set of all finite length strings that can be constructed by concatenating a finite number of elements of Σ. The resultant set is the infinite set Σ^*.

Since Σ^* is closed under concatenation, concatenation is associative, and ε is the identity element for concatenation of strings, $\langle \Sigma^*, \text{ concatenation}, \varepsilon \rangle$ is a free monoid.

Definition 2.2.5 A *language* L over the alphabet Σ is a subset of the set Σ^*.

For example, one language containing a finite number of elements over the alphabet $\Sigma = \{a, b\}$ is $L_1 = \{a, aa, b, aba\}$. An example of a language containing an infinite number of elements over the alphabet $\Sigma = \{0,1\}$ is $L_2 = \{w \in \Sigma^* \mid \text{the letter 1 occurs in } w \text{ an even number of times}\}$. Notice that the empty language $\varnothing$ is distinct from the language containing only the empty string, $\{\varepsilon\}$.

Since a language is a set, we can use the normal set definitions of union, intersection, and complementation.

The set of all possible languages over an alphabet Σ is the power set of Σ^*, denoted $\mathcal{P}(\Sigma^*)$.
Notice that since the power set contains all possible subsets of Σ^*, $\mathcal{P}(\Sigma^*)$ is closed under union of languages over the same alphabet.

Further, $\varnothing$ is an identity element for the operation of union over $\mathcal{P}(\Sigma^*)$ since $L \cup \varnothing = \varnothing \cup L = L$. From the normal definition of sets, union is associative and commutative. Thus, we have shown that $< \mathcal{P}(\Sigma^*), \cup, \varnothing >$ satisfies all the conditions to be a commutative monoid.

Definition 2.2.6 The *concatenation* of two languages L_1, L_2, denoted $L_1 \circ L_2$, is defined as $L_1 \circ L_2 = \{w_1 w_2 \mid w_1 \in L_1 \text{ and } w_2 \in L_2\}$.

For example, let $\Sigma = \{a, b, c, \ldots, y, z, 0, 1, \ldots, 9\}$, $L_1 = \{math\}, L_2 = \{334, 335, 336\}$. Then $L_1 \circ L_2 = \{math334, math335, math336\}$.

Notice that since the power set contains all possible subsets of Σ^*, $\mathcal{P}(\Sigma^*)$ is closed under concatenation of languages over the same alphabet. Further, since concatenation is simply linking strings together in order, it does not matter which strings are linked first, so concatenation of languages is associative.

Also, $\{\varepsilon\}$ is an identity element of concatenation of languages because $L \circ \{\varepsilon\} = \{\varepsilon\} \circ L = L$.

Thus, $< \mathcal{P}(\Sigma^*), \circ, \{\varepsilon\} >$ is a monoid.

Lemma 2.2.7 Concatenation of languages is distributive over union of languages
Proof: First, because concatenation is not commutative, we must show both that $L \circ (X \cup Y) = (L \circ X) \cup (L \circ Y)$ and that $(X \cup Y) \circ L = (X \circ L) \cup (Y \circ L)$. However, the proofs are nearly identical, and we will leave the second to the reader. Note that in either case if one side equaled $\varnothing$, then either $L = \varnothing$ or $X = \varnothing = Y$, which causes the other side of the equation to be $\varnothing$ as well. Thus, suppose that both sides contain at least one element.

First, we will show that $L \circ (X \cup Y) \Rightarrow (L \circ X) \cup (L \circ Y)$. Let $w \in (L \circ (X \cup Y))$. Because *w* is in a set that is a concatenation of two languages, we know that for some $w_1 \in L, w_2 \in X \cup Y$, $w = w_1 w_2$. Since $w_2 \in X \cup Y$, $w_2 \in X$ or $w_2 \in Y$. Then, either $w \in (L \circ X)$ or $w \in (L \circ Y)$, but then by the definition of union, $w \in ((L \circ X) \cup (L \circ Y))$.

Now we will show that $L \circ (X \cup Y) \Leftarrow (L \circ X) \cup (L \circ Y)$. Let $w \in ((L \circ X) \cup (L \circ Y))$. Then by definition of union of languages, $w \in (L \circ X)$ or $w \in (L \circ Y)$. Then by the definition of concatenation, either there exists $w_1 w_2 = w$ such that $w_1 \in L$ and $w_2 \in X$ or $w_1 \in L$ and $w_2 \in Y$. However, that is the same as saying that there exists $w_1 w_2 = w$ such that $w_1 \in L$ and $w_2 \in X \cup Y$. That is, $w \in (L \circ (X \cup Y))$.

□

Lemma 2.2.8 For any language *L*, $L \circ \varnothing = \varnothing \circ L = \varnothing$
Proof: We will use proof by contradiction. Suppose $L \circ \varnothing \neq \varnothing$ (the other case is nearly identical). Then by the definition of concatenation, there exists some $w \in (L \circ \varnothing)$ such that we can split *w* into two strings w_1, w_2 such that $w = w_1 w_2$, $w_1 \in L$, and $w_2 \in \varnothing$. However, the last result is clearly incorrect because there are no strings in the empty set. Thus, it must be that there are no strings in the concatenation of the empty set and a language L. That is, $L \circ \varnothing = \varnothing \circ L = \varnothing$.
□

Thus, $< \mathcal{P}(\Sigma^*), \cup, \circ, \varnothing, \{\varepsilon\} >$ is a semiring.

Definition 2.2.8 The *star* of a language, denoted L^*, is the union of L concatenated with itself a finite number of times. That is, $L^* = \bigcup_{i \geq 0} L^i$, where L^i represents L concatenated with itself *i* times. i.e. $L^i = \underbrace{L \circ L \circ L \circ \cdots \circ L \circ L}_{i \text{ times}}$. We define for $i = 0$, $L^i = \{\varepsilon\}$ and $L^1 = L$

3 Isomorphism Connects Algebra and Language Theory

3.1 Formal Power Series

Because Σ^* consists of all finite length strings of elements of Σ, we can list those strings in order, sorting first by length of string then by "alphabetical" order. In fact, any specific ordering of Σ^* works for this purpose, but "alphabetical" order is simple to picture. For example, in the alphabet $\Sigma = \{0,1\}$, we can order the elements first by string length and then by increasing binary number:

Σ^*:	ε	0	1	00	01	10	11	000	001	010	011	…

Now, consider the language $L = \{w \in \Sigma^* \mid \text{there is an even number of ones in } w\}$. Because L is a subset of Σ^*, a way to describe L would be to say, for each element of Σ^*, whether that element is in L or not. We will use the convention that if an element of Σ^* is in *L*, then it is described by a 1, and if it is not in *L*, then it is described by a 0:

Σ^*:	ε	0	1	00	01	10	11	000	001	010	011	…
L:	1	1	0	1	0	0	1	1	0	0	1	…

Some other simple languages become:

Σ^* :	ε	0	1	00	01	10	11	000	001	010	011	…
ε :	1	0	0	0	0	0	0	0	0	0	0	…
$\varnothing$:	0	0	0	0	0	0	0	0	0	0	0	…

This sequence of 0's and 1's is called the *characteristic sequence* for L. When the characteristic sequence is considered in its own right, it is denoted *r*, and L is called the *support* of r, denoted $supp(r)$. It would be convenient if we could represent this more concisely, and formal power series is the answer. A formal power series is written as an infinite sum and looks very similar to a standard power series; however, we are not actually computing the sum of the terms. The sum notation is used because of the amazing similarities in how operators are applied. Instead of indexing the term in the formal power series by the power of the variable, it is indexed by the variable *w* as *w* goes over the (countably infinite) elements of Σ^*. Thus, we can write the characteristic sequence r as

$$r = \sum_{w \in \Sigma^*} \chi_L(w) w$$

where $\chi_L(w)$ is the coefficient for the *w*'th term and is a function depending both on the element $w \in \Sigma^*$ and the language $L \in \mathcal{P}(\Sigma^*)$ Notice that $\chi_L(w)$ is only ever zero or one. Since the coefficient $\chi_L(w)$ is a simple, two-valued function, it is a Boolean function. Thus, the set of all possible characteristic sequences is written $\mathbb{B}\langle\langle \Sigma^* \rangle\rangle$, where the $\mathbb{B}$ is for the use of the Boolean numbers.

We now have another notation for a language. That is, $L = \{w \mid \chi_L(w) \neq 0\}$. Since we are only considering formal power series with Boolean coefficients, this is an identical definition to $L = \{w \mid \chi_L(w) = 1\}$.

Definition 3.1.1 Two formal power series r_1, r_2 are *equal* if and only if $r_1 = \sum_{w \in \Sigma^*} \chi_{L_1}(w) w$ and $r_2 = \sum_{w \in \Sigma^*} \chi_{L_2}(w) w$, where $\chi_{L_1}(w) = \chi_{L_2}(w)$ for all $w \in \Sigma^*$.

Now, we will define some operators for formal power series. Let r_1 be the formal power series representation of L_1 and similarly for r_2. Like we do with power series, we define the *sum* $r_1 + r_2 = \sum_{w \in \Sigma^*} \chi_{L_1 + L_2}(w) w$ with $\chi_{L_1 + L_2}(w) = \chi_{L_1}(w) + \chi_{L_2}(w)$, where 1+1=1, but the other numbers add as expected.

Define $0 = \sum_{w \in \Sigma^*} 0w$. Then, because this changes none of the entries of another formal power series when they are added together, 0 is the additive identity in $\mathbb{B}\langle\langle \Sigma^* \rangle\rangle$. Further, since in

Boolean arithmetic $1+0=0+1=1$, etc. , the sum of formal power series is a commutative operator. That is, $\left\langle \mathbb{B}\left\langle\left\langle \Sigma^{*}\right\rangle\right\rangle,+,0\right\rangle$ is a commutative monoid.

Now, because each element of Σ^{*} is a finite length string, there is a finite number of ways to split an element *w* into two parts. That is, a finite number of distinct possible strings w_1, w_2 such that $w = w_1 w_2$. For example, there are three ways to split the two character string $w = ab$: $\varepsilon / ab; a / b; ab / \varepsilon$.

Define the (Cauchy) product of two formal power series

$$r_1 \cdot r_2 = \sum_{w\in\Sigma^{*}} \chi_{L_1 \cdot L_2}(w)\, w, \text{ where } \chi_{L_1 \cdot L_2}(w) = \sum_{w_1 w_2 = w} \left(\chi_{L_1}(w_1)\chi_{L_2}(w_2)\right)$$

where the multiplication $\chi_{L_1}\chi_{L_2}$ follows the normal rules for multiplication of natural numbers.

Lemma 3.1.2 The formal power series for $L = \{\varepsilon\}$ is the multiplicative identity for formal power series.

Proof: For all $w \in \Sigma^{*}$, $w = \varepsilon w$, so we can split *w* into two parts, ε and *w*. If $\chi_{L_2}(\varepsilon) = 1$, then all the coefficients in r_1 that were 1 will still be 1 because there will be at least one term in the sum in $\chi_{L_1 L_2}$ that is 1. If $\chi_{L_2}(w) = 1$ only for ε, then there will never be a time when a coefficient in r_1 is zero but the relevant term in $r_1 r_2$ is 1. Thus, the formal power series for $L = \{\varepsilon\}$ is the identity for multiplication in formal power series. For simplicity, we will represent this power series by ε.

□

We can similarly quickly show that the distributive property holds, and it is clear that the Cauchy product of any series with 0 is 0. Thus, $\left\langle \mathbb{B}\left\langle\left\langle \Sigma^{*}\right\rangle\right\rangle,+,\cdot,0,\varepsilon\right\rangle$ is a semiring.

3.2 Semiring Homomorphisms

At least in the texts I found, semirings and their properties were used nonchalantly without clearly defining what they were. Because of this lack, I constructed definitions to produce the results.

Definition 3.2.1 A *semiring homomorphism* between two semirings $A = \langle S_A, +_A, \times_A, 0_A, 1_A\rangle$ and $B = \langle S_B, +_B, \times_B, 0_B, 1_B\rangle$ is a function $f : S_A \to S_B$ where for $a, b \in S_A$

(i) $f(a +_A b) = f(a) +_B f(b)$

(ii) $f(a \times_A b) = f(a) \times_B f(b)$
(iii) $f(0_A) = 0_B$

Definition 3.2.2 A *semiring isomorphism* between two semirings *A* and *B* is a semiring homomorphism where the function is also bijective.

Theorem 3.2.3 If there is a semiring isomorphism *f* from $A = \langle S_A, +_A, \times_A, 0_A, 1_A \rangle$ to $B = \langle S_B, +_B, \times_B, 0_B, 1_B \rangle$, then there is an inverse semiring isomorphism *g* from *B* to *A*.

Proof: Since *f* is a bijection, we immediately have that we can define $g(b) = f^{-1}(b) = a$ for all $b \in S_B$ and $a \in S_A$. Further, that is true for the zero element of both rings.

Now let $x, y \in B$. Since *f* is bijective, there exist $u, v \in A$ such that $f(u) = a$ and $f(v) = b$. Further, since *f* is an isomorphism, $f(u+v) = a+b$. Taking *f*-inverse of both sides, we have $u+v = f^{-1}(a+b)$. However, $g(a)+(b) = u+v = f^{-1}(a+b) = g(a+b)$. The process is similar for multiplication.
□

Theorem 3.2.4 $\left\langle \mathbb{B}\left\langle\left\langle \Sigma^* \right\rangle\right\rangle, +, \cdot, 0, \varepsilon \right\rangle$ and $< \mathcal{P}\left(\Sigma^*\right), \cup, \circ, \varnothing, \{\varepsilon\} >$ are isomorphic.

For simplicity, we will call the first semiring *B* and the second semiring *P*. We will show they are isomorphic by finding the necessary function from *B* to *P*. In fact, we will show that the support of a formal power series, as is defined in the formal power series section, is the necessary function. First we will show that $supp : B \to P$ satisfies the requirements for a ring homomorphism, then show that it is one-to-one and finally onto.

Recall that the support of a power series r can be described as $\left\{ w \in \Sigma^* \mid \chi_L(w) \neq 0 \right\}$.

(i) $supp(r_a + r_b) = supp(r_a) \cup supp(r_b)$
When adding two formal power series together, $\chi_{a+b}(w) = 1$ if and only if $\chi_a(w) = 1$ or $\chi_b(w) = 1$. Then, by taking the support, $w \in supp(r_{a+b})$ if and only $w \in supp(r_a)$ or $w \in supp(r_b)$. However, that is identical to saying $w \in supp(r_{a+b})$ if and only $w \in \left(supp(r_a) \cup supp(r_b)\right)$. Thus, we have that $supp(r_a + r_b) = supp(r_a) \cup supp(r_b)$.

(ii) $supp(r_a \cdot r_b) = supp(r_a) \circ supp(r_b)$
When multiplying two formal power series together, $\chi_{ab}(w) = 1$ if and only if there was at least one way to split w such that $w = w_1 w_2$ where $\chi_a(w_1) = 1$ and $\chi_b(w_2) = 1$. Then, taking the support, $w \in supp(r_{ab})$ if and only there exists a split $w = w_1 w_2$ such that $w_1 \in supp(r_a)$ and

$w_2 \in supp(r_b)$. However, $w \in (supp(r_a) \circ supp(r_b))$ if and only if there exists such a split. Thus, $supp(r_a \cdot r_b) = supp(r_a) \circ supp(r_b)$.

(iii) $supp(0) = \varnothing$
This is obvious since the power series contains no nonzero coefficients, and thus the support can contain no strings.
Thus, support is a semiring homomorphism. Now, there are two conditions left to check to show support is a semiring isomorphism.

Support is 1-1
If it was not 1-1, then there exists at least two distinct formal power series r_1, r_2 where $supp(r_1) = supp(r_2)$. However, that would imply that $\{w \in \Sigma^* \mid \chi_1(w) = 1\} = \{w \in \Sigma^* \mid \chi_2(w) = 1\}$. Since the only other option for $\chi(w)$ is 0, we also have that $\{w \in \Sigma^* \mid \chi_1(w) = 0\} = \{w \in \Sigma^* \mid \chi_2(w) = 0\}$, so each *w* has a coefficient of either 0 or 1, and for all $w \in \Sigma^*, \chi_1(w) = \chi_2(w)$, so $r_1 = r_2$. However, this contradicts the assumption that r_1 and r_2 are distinct. Thus, the support function must be one-to-one.

Support is onto
Choose some $L \in \mathcal{P}(\Sigma^*)$. Then we can determine the characteristic sequence for *L*, and thus we can define $r_L = \sum_{w \in \Sigma^*} \chi_L(w) w$ for any $L \in \mathcal{P}(\Sigma^*)$. That is, the support function maps the Boolean formal power series onto the set of languages over the same alphabet.
□

4 Conclusion and Beyond

This is just one tiny start on the connection between formal power series and languages. It is typical to continue on by discussing the various closed subsets of languages and their power series definitions. Further, it is common to begin to mathematically define machines - roughly directed graphs that start and stop in the right places with linear algebra and the such. However, the next step from this paper requires only a few more definitions to describe.

Definition 4.1 A formal power series represents a *polynomial* if only a finite number of terms have nonzero coefficient $\chi_L(w)$.

Definition 4.2 A power series *r* belonging to $\mathbb{B}\langle\langle\Sigma^*\rangle\rangle$ is called *rational* (over Σ) if *r* can be obtained from polynomial elements of $\mathbb{B}\langle\langle\Sigma^*\rangle\rangle$ by finitely many applications of the sum, product, and star formal power series operations. The family of such rational power series is denoted by $\mathbb{B}^{rat}\langle\langle\Sigma^*\rangle\rangle$.

In fact, $\mathbb{B}^{rat}\langle\langle\Sigma^*\rangle\rangle$ is the smallest closed subsemiring of $\mathbb{B}\langle\langle\Sigma^*\rangle\rangle$ containing all the polynomials over Σ, although the proof is beyond the level this paper. However, that implies that $\mathbb{B}^{rat}\langle\langle\Sigma^*\rangle\rangle$ is generated by applying the operations of sum, product, and star in sequence to polynomials. In fact, we merely need the polynomials representing each individual letter in Σ and the polynomial representing ε, and the power series 0.

Now we turn to the computer science side of things.

Definition 4.3 R is a *regular expression* over the alphabet Σ if R is one of the following or generated from the following by a finite number of language operations of union, concatenation, and star.

1. $\{a\}$, for some $a \in \Sigma$
2. $\{\varepsilon\}$
3. $\varnothing$

Definition 4.4 The set of possible strings generated by a regular expression is called a *regular language*.

The next step is to show that regular languages represented by regular expressions and by rational power series represent the same set. That is, there is an isomorphism between the two subsemirings. This identity is usually proved by way of a set of machines that also deal with the same set of languages, but mathematically it appears this direct approach of algebra might be simpler.

4. References

1. Clifford, A. H. and Preston, G. B., The Algebraic Theory of Semirgroups Volume I. Mathematical Surveys - Number 7. American Mathematical Society. 1961.

2. Conway, J. H., Regular Algebra and Finite Machines. Chapman and Hall Mathematics Series. Chapman and Hall LTD. 1971.

3. Eilenberg, Samuel. Automata, Languages, and Machines volume A. Pure and Applied Mathematics A Series of Monographs and Textbooks. Academic Press. 1974.

4. Ginzburg, Abraham. Algebraic Theory of Automata. ACM Monograph Series. Academic Press. 1968.

5. Kuich, Werner and Salomaa, Arto. Semirings, Automata, Languages. EATCS Monographs on Theoretical Computer Science. Springer-Verlag. 1986.

6. Salomaa, Arto and Soittola, Matti. Automata -- Theoretic Aspects of Formal Power Series. Texts and Monographs in Computer Science. Springer-Verlag. 1978

7. Sipser, Michael. Introduction to the Theory of Computation second edition. Course Technology, Cengage Learning. 2006.