FIRST-ORDER LOGIC NOTES

JASON BELL

Contents

1. Formal languages 1
2. Regular languages 2
3. Exercises 3
4. The Pumping Lemma 3
5. Exercises 4
6. Enumeration and Linear Recurrence 5
7. Exercises 9

1. Formal languages

We will eventually explore first-order logic from a syntactic point of view, but before we can do so, we must first understand the machinery of formal languages. To build a language, we begin with an *alphabet* Σ, which is a finite, non-empty set of symbols. From this alphabet, we form *words*, which are finite sequences obtained by concatenating symbols from Σ. For instance, if $\Sigma = \{0, 1\}$, then 0110 and 110001 are examples of words over Σ. The collection of all words over Σ is denoted Σ^*; it includes the empty word ϵ, and it carries a natural associative binary operation given by concatenation of words, which we think of as a form of multiplication. For this reason, given a word u over the alphabet Σ, we write u^k to denote the k-fold concatenation $uu \cdots u$. We let $|u|$ denote the length of a word u; for example, the binary word 01101 has length 5 and so $|01101| = 5$, and we let Σ^n denote the set of length-n words over Σ.

A *formal language* over Σ is simply a set of such words. Given an alphabet, there are uncountably many formal languages one can define over it, but most of these are neither useful nor describable in an effective way. In practice, we are interested only in those languages that we can *construct*; that is, languages for which we can provide a rule or procedure that specifies which words belong.

For example, the set of binary strings that begin with 1, or the set of strings whose length is a prime number, are both formal languages over $\{0, 1\}$ that can be defined by a clear computational procedure. We will use the word *procedure* informally for now, though we will return to this notion more formally when we introduce Turing machines. Our goal is to understand formal languages in terms of their *computational complexity.* With that in mind, we begin our study with the simplest and most well-behaved class: *regular languages.*

2. Regular languages

In the 17th century, Descartes proposed that (non-human) animals could be understood as intricate biological machines, which were capable of responding to environmental stimuli, but which lacked anything resembling a complex inner life. While Descartes' guiding paradigm was biologically flawed, the idea of such a machine turns out to be surprisingly fruitful in theoretical computer science. When we strip Descartes' model down to its key components, we find a finite number of internal states and a set of rules that govern transitions between these states in response to input. A machine of this type is called a *finite-state automaton*, and a language is said to be *regular* if it can be recognized by such a machine. Finite-state automata are, in this sense, the computational counterparts of Descartes' animals and regular languages occupy the lowest tier in the hierarchy of formal languages in terms of their expressive power.

Definition 2.1. A (*deterministic*) *finite-state automaton* is a 5-tuple

$$M = (Q, \Sigma, \delta, q_0, F),$$

where:

- Q is a finite set of states,
- Σ is a finite input alphabet,
- $\delta : Q \times \Sigma \to Q$ is the transition function,
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is the set of accepting (or final) states.

The automaton M accepts a *regular language* $L \subseteq \Sigma^*$ as follows. Given a finite word $w = x_1 x_2 \cdots x_n$, with each $x_i \in \Sigma$, we begin in the initial state q_0. We then define a sequence of states $q_1, \ldots, q_n$ inductively by

$$q_i = \delta(q_{i-1}, x_i), \quad \text{for } i = 1, \ldots, n.$$

If the final state q_n lies in F, we say that M *accepts* the word w, and so $w \in L$. Otherwise, w is consigned to a truly tenebrous fate: rejection—or more precisely, it is not in the language L.

This inductive procedure given above allows us to naturally extend the transition function δ to a map

$$\delta : Q \times \Sigma^* \to Q,$$

by setting $\delta(q_0, w) = q_n$. That is, we can apply δ directly to the entire input word.

Finite-state automata are typically visualized with states represented as labelled circles. The initial state is marked by an incoming arrow, accepting states are indicated by double circles, and transitions are drawn as arrows labelled by symbols from the input alphabet Σ. We give an example in Figure 1, which accepts the set of binary strings with an even number of 1s.

Observe that if we feed the binary string $w = 1101$ to the finite-state automaton in Figure 1 we begin at the initial state q_0. Reading the first bit, a 1, we transition to state q_1. The next bit is also a 1, which sends us back to q_0. Feeding in the third bit, a 0, we remain in q_0, and finally, reading the last bit, a 1, we move to q_1. Since q_1 is not an accepting state, we conclude that w is not in the language recognized by this automaton.

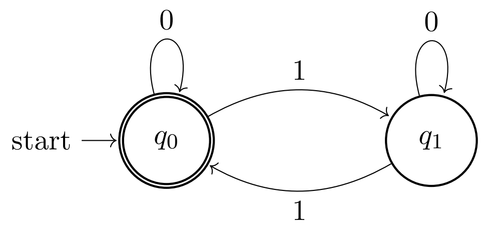

Figure 1. A finite-state automaton accepting binary strings with an even number of 1s

3. Exercises

Exercise 3.1. Given a language L over a finite alphabet Σ, we define the *Kleene star* of L (pronounced "KLEE-nee") to be the language L^* consisting of all finite concatenations (including the empty concatenation) of elements of L. That is,

$$L^* = \{w_1 \cdots w_s \colon s \geq 0,\ w_1, \ldots, w_s \in L\}.$$

Prove that if L is a regular language then so is L^*.

4. The Pumping Lemma

One of the most useful tools in proving that a formal language is not regular is a result known as the *pumping lemma*. This result initially might feel more like a trick than a tool that actually provides structural insight into regular languages, but it in fact captures an essential property of finite-state automata: given sufficiently long input, a loop must occur at some point during our transitions from state to state. This repetition means that part of the input can be "pumped" (that is, repeated any number of times)[1] without affecting membership in our language.

Formally, the pumping lemma is stated as follows. We recall that for a word u, we let u^k denote its k-fold concatenation and let $|u|$ denote its length.

Lemma 4.1. *(The Pumping Lemma) Let L be a regular language over a finite alphabet Σ. Then there exists a pumping length $p \geq 1$ such that every word w in L of length at least p can be written as $w = xyz$ such that:*

(a) $|xy| \leq p$,
(b) $|y| \geq 1$, *and*
(c) *for all $i \geq 0$, the string $xy^iz \in L$.*

Moreover, p can be taken to be the number of states in a finite-state machine that accepts the language L.

On a more intuitive level, if a regular language contains a long enough word then it must contain an infinite set of words created by "pumping" a middle section of the word. This is a consequence of the fact that a finite-state automaton cannot remember how many times it has looped through a state but instead only acts according to its current state and input.

[1]I don't know why this act is referred to as pumping, but it sounds cool.

Proof of the Pumping Lemma. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite-state automaton that accepts the regular language L. Let $p := |Q|$, and let $w = a_1a_2 \cdots a_n \in L$ be a word of length $n \geq p$.

Define a sequence of states $q_0, q_1, \ldots, q_n$ inductively by

$$q_i = \delta(q_{i-1}, a_i) \quad \text{for } i = 1, \ldots, n.$$

Since the sequence $q_0, \ldots, q_p$ has length $p + 1$ and since there are only p states in Q, by the pigeonhole principle there exist indices $0 \leq i < j \leq p$ such that $q_i = q_j$.

Now define:

$$x = a_1a_2 \cdots a_i, \quad y = a_{i+1}a_{i+2} \cdots a_j, \quad z = a_{j+1} \cdots a_n.$$

Then $w = xyz$, $|xy| \leq p$, and $|y| \geq 1$. Moreover, since $q_i = q_j$, reading y causes the machine to loop back to q_i, so for all $k \geq 0$,

$$\delta(q_0, xy^kz) = \delta(q_i, y^kz) = \delta(q_i, z).$$

But $\delta(q_i, z) = q_n \in F$, since $w \in L$. Hence, $xy^kz \in L$ for all $k \geq 0$, and we obtain the desired result. □

The true utility of the pumping lemma comes from invoking its contrapositive in order to show that a given language is not regular. The strategy is then to assume that the language is regular and hence satisfies the conditions of the pumping lemma. You then find a word $w \in L$ with the property that for every possible decomposition $w = xyz$ as given in the statement of the pumping lemma one necessarily has $xy^iz \notin L$ for some $i \geq 2$.

We note that the pumping lemma is not a characterization of regular languages: it gives a necessary but not sufficient condition for a language to be regular (see Exercises 5.8 and 5.9).

5. Exercises

Exercise 5.1. Let $L = \{0^n1^n \mid n \geq 0\}$. Use the pumping lemma to show that L is not a regular language.

Exercise 5.2. Given a natural number $k \geq 2$, we let $(n)_k$ denote the base-k expansion of n. This is the unique word $d_md_{m-1} \cdots d_0$ over the alphabet $\{0, 1, \ldots, k-1\}$ such that $d_m \neq 0$ and

$$n = \sum_{i=0}^{m} d_ik^i,$$

with the convention that $(0)_k$ is the empty word.

Show that the set

$$\{(n)_k \mid n \geq 0\}$$

is a regular language by constructing a finite-state automaton that accepts this set.

Exercise 5.3. Given a natural number $k \geq 2$ and a word $w = d_md_{m-1} \cdots d_0$ over the alphabet $\{0, 1, \ldots, k-1\}$, we define

$$[w]_k := d_mk^m + d_{m-1}k^{m-1} + \cdots + d_1k + d_0,$$

the nonnegative integer represented by w in base k. Show that if $u, v, w \in \{0, 1, \ldots, k-1\}^*$ with $|v| \geq 1$, then there exist rational numbers $\alpha \neq 0$ and β such that

$$[uv^nw]_k = \alpha k^n + \beta \quad \text{for all } n \geq 0.$$

Exercise 5.4. Use the pumping lemma to prove that if S is an infinite subset of the prime numbers and $k \geq 2$, then the set of words over the alphabet $\{0, 1, \ldots, k-1\}$ that represent the base-k expansions of elements of S does not form a regular language. (Hint: see if you can use Exercise 5.3 somehow.)

Exercise 5.5. Let Σ be an alphabet of size at least two. Show that the set of palindromes over Σ (that is, words that are equal to their reversal) is not a regular language.

Exercise 5.6. Two nonzero rational numbers α and β are said to be multiplicatively independent if the only solution to $\alpha^n = \beta^m$ with $(n, m) \in \mathbb{Z}^2$ is $n = m = 0$. Show using the pumping lemma that if k and ℓ are multiplicatively independent integers ≥ 2 then the set of words over the alphabet $\{0, 1, \ldots, k-1\}$ that represent the base-k expansions of elements of $\{\ell^n : n \geq 0\}$ does not form a regular language. (Hint: use Exercise 5.3.)

Exercise 5.7. Let Σ be a finite alphabet and let L denote the set of words over Σ of prime length. Use the pumping lemma to prove that L is not regular.

The k-th Dyck language D_k is the set of words over the alphabet

$$\{\, (_1, \ldots, (_k,)_1, \ldots,)_k \,\}$$

that represent valid parenthesizations using k distinct types of matched parentheses. For example, $(_1(_2(_1)_1)_2)_1$ is a Dyck word, but $(_1)_1)_1(_2$ and $(_1(_2)_1)_2$ are not. Show that D_k is not a regular language for any $k \geq 1$.

Exercise 5.8. Given an alphabet Σ, a word w over Σ is said to be *primitive* if it cannot be written in the form u^k for some word $u \in \Sigma^*$ and integer $k \geq 2$, where u^k denotes the word formed by concatenating k copies of u. Show that the conclusion to the statement of the pumping lemma applies to the set of primitive words over an alphabet of size at least two and that we can take the pumping length to be 2.

Exercise 5.9. (Hard!) Show that the set of primitive words is not regular.

6. Enumeration and Linear Recurrence

One of the more surprising facts about regular languages is that they aren't just simple to recognize—they're also easy to count. More precisely, given a regular language L, let L_n denote the set of words in L of length n, and let $f(n) = |L_n|$. Then there exists a natural number $d \geq 1$ and integers $c_0, c_1, \ldots, c_{d-1}$ such that

$$f(n+d) + c_{d-1} f(n+d-1) + \cdots + c_1 f(n+1) + c_0 f(n) = 0, \tag{6.1}$$

for all $n \geq 0$. That is, the number of words of length n in a regular language satisfies a linear recurrence relation with constant coefficients.

To show this, let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite-state automaton accepting the language L. To this machine we associate an adjacency matrix $A(M)$ as follows. Suppose the states of M are $q_0, q_1, \ldots, q_{d-1}$, with q_0 the initial state. Then we define an $d \times d$ matrix whose (i, j)-entry is the number of letters $a \in \Sigma$ such that

$$\delta(q_{i-1}, a) = q_{j-1}.$$

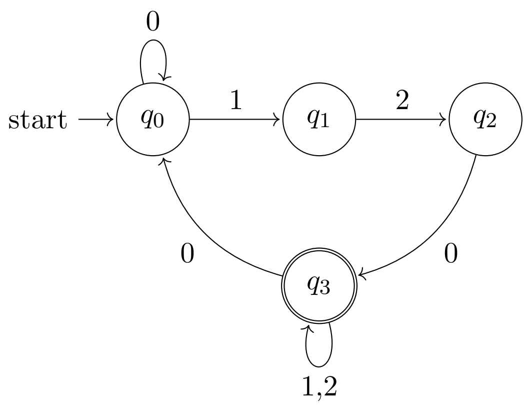

FIGURE 2. A finite-state automaton over the ternary alphabet $\{0, 1, 2\}$.

For example, if M is the finite-state automaton with input alphabet $\Sigma = \{0, 1, 2\}$ and states $Q = \{q_0, q_1, q_2, q_3\}$ as given in Figure 2, then $A(M)$ is the matrix

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 2 \end{pmatrix}.$$

We can think of the (i, j)-entry of the adjacency matrix $A(M)$ as recording the number of words $w \in \Sigma$ of length 1 such that $\delta(q_{i-1}, w) = q_{j-1}$. This idea extends naturally to words of greater length by considering powers of $A(M)$.

Notation 6.1. Given an $m \times n$ matrix B, we let $B(i, j)$ denote its (i, j)-entry. If A and B are $m \times n$ and $n \times p$ matrices respectively, then the (i, j)-entry of the product $A \cdot B$ is given by

$$(A \cdot B)(i, j) = \sum_{k=1}^{n} A(i, k)B(k, j). \tag{6.2}$$

Lemma 6.2. *Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite-state automaton with $Q = \{q_0, \ldots, q_{d-1}\}$, and let $A(M)$ be its $d \times d$ adjacency matrix. Then for all $n \geq 1$, the (i, j)-entry of $A(M)^n$ is equal to the number of words $w \in \Sigma^n$ such that $\delta(q_{i-1}, w) = q_{j-1}$.*

Proof. We prove the claim by induction on n. The base case $n = 1$ holds by the definition of $A(M)$. Now assume the statement holds for all positive integers less than m, and consider the case $n = m$.

Let $w \in \Sigma^m$ be a word of length m such that $\delta(q_{i-1}, w) = q_{j-1}$. We can write $w = au$, where $a \in \Sigma$ and $u \in \Sigma^{m-1}$. Then

$$A(M)^m = A(M) \cdot A(M)^{m-1},$$

and the (i, j)-entry of this product is given by

$$A(M)^m(i, j) = \sum_{k=1}^{d} A(M)(i, k) \cdot A(M)^{m-1}(k, j).$$

Here, $A(M)(i,k)$ is the number of letters $a \in \Sigma$ such that $\delta(q_{i-1}, a) = q_{k-1}$, and by the inductive hypothesis, $A(M)^{m-1}(k,j)$ is the number of words $u \in \Sigma^{m-1}$ such that $\delta(q_{k-1}, u) = q_{j-1}$.

So the product counts all words of the form $au \in \Sigma^m$ such that $\delta(q_{i-1}, au) = q_{j-1}$, and every length-$m$ word with this property is uniquely expressible in this form. Therefore $A(M)^m(i,j)$ gives the total number of length-m words mapping q_{i-1} to q_{j-1} under δ, completing the induction.

□

We can view our finite-state automaton M as a directed graph, with vertices given by the state set Q and edges determined by the transition function δ. In this picture, the adjacency matrix $A(M)$ records the number of label;ed edges between states, and the entry $A(M)^d(i,j)$ counts the number of directed paths of length d from state q_{i-1} to q_{j-1}.

To understand the long-term behaviour of powers of $A(M)$, we recall a fundamental fact from linear algebra.

Theorem 6.3 (Cayley-Hamilton). *Let $d \geq 1$, and let A be an $d \times d$ matrix with entries in a field $\mathbb{K}$. Then*

$$A^d + c_{d-1}A^{d-1} + \cdots + c_1 A + c_0 I = 0,$$

where I is the $d \times d$ identity matrix, and

$$x^d + c_{d-1}x^{d-1} + \cdots + c_1 x + c_0 = \det(xI - A)$$

is the characteristic polynomial of A.

Combining Lemma 6.2 with the Cayley-Hamilton theorem, we obtain the following result.

Theorem 6.4. *Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite-state automaton with $Q = \{q_0, \ldots, q_{d-1}\}$, and let $A(M)$ be its $d \times d$ adjacency matrix. Fix $i, j \in \{1, \ldots, d\}$, and let $f(n)$ denote the number of words $w \in \Sigma^n$ such that $\delta(q_{i-1}, w) = q_{j-1}$. Then*

$$f(n+d) + c_{d-1}f(n+d-1) + \cdots + c_1 f(n+1) + c_0 f(n) = 0, \tag{6.3}$$

for all $n \geq 0$, where

$$x^d + c_{d-1}x^{d-1} + \cdots + c_1 x + c_0$$

is the characteristic polynomial of $A(M)$. In particular, if L denotes the regular language accepted by M, then the number of length-n words in L satisfies the linear recurrence given in Equation (6.3).

We note that since $A(M)$ is a matrix with integer entries, the coefficients $c_{d-1}, \ldots, c_0$ of its characteristic polynomial are all integers.

Proof of Theorem 6.4. By Lemma 6.2, we have

$$f(n) = A(M)^n(i,j),$$

and by the Cayley-Hamilton theorem, the matrix $A(M)$ satisfies its characteristic polynomial:

$$A(M)^d + c_{d-1}A(M)^{d-1} + \cdots + c_1 A(M) + c_0 I = 0.$$

Multiplying both sides on the right by $A(M)^n$ yields

$$A(M)^{n+d} + c_{d-1}A(M)^{n+d-1} + \cdots + c_1 A(M)^{n+1} + c_0 A(M)^n = 0.$$

Taking the (i,j)-entry of both sides then gives

$$f(n+d)+c_{d-1}f(n+d-1)+\cdots+c_1f(n+1)+c_0f(n)=0,$$

as claimed.

Now, the number of length-n words in the language L accepted by M is the number of words $w\in\Sigma^n$ such that $\delta(q_0,w)\in F$. Since q_0 is indexed as state 1, this count is given by

$$\sum_{\substack{j=1\\ q_{j-1}\in F}}^{d} A(M)^n(1,j).$$

Each summand is of the form $f(n)=A(M)^n(i,j)$, and each such function satisfies the recurrence (6.3). Since the sum of finitely many sequences satisfying the same linear recurrence also satisfies the same recurrence, it follows that the total number of length-n words in L also satisfies the recurrence from Equation (6.3). □

Sequences that satisfy linear recurrence relations are well behaved and there are several equivalent and useful ways to characterize this property.

Theorem 6.5. *Let $f(0),f(1),f(2),\ldots$ be a sequence taking values in a field $\mathbb{K}$, and let $\overline{\mathbb{K}}$ denote the algebraic closure of $\mathbb{K}$. The following are equivalent:*

(a) *The sequence $f(n)$ satisfies a linear recurrence relation of the form*

$$f(n+d)+c_{d-1}f(n+d-1)+\cdots+c_1f(n+1)+c_0f(n)=0,$$

for all $n\geq 0$, with $c_0,\ldots,c_{d-1}\in\mathbb{K}$;

(b) *There exists a matrix $A\in\mathbb{K}^{d\times d}$, a row vector $\mathbf{v}\in\mathbb{K}^{1\times d}$, and a column vector $\mathbf{w}\in\mathbb{K}^{d\times 1}$ such that*

$$f(n)=\mathbf{v}A^n\mathbf{w}$$

for all $n\geq 0$;

(c) *The generating function $\sum_{n\geq 0}f(n)x^n$ is the power series expansion of a rational function $P(x)/Q(x)$, where $P(x),Q(x)\in\mathbb{K}[x]$ are relatively prime polynomials with $Q(x)$ of degree d, $Q(0)=1$, and the degree of $P(x)$ at most $d-1$;*

(d) *There exists a nonnegative integer $e\leq d$, distinct nonzero scalars $\lambda_1,\ldots,\lambda_e\in\overline{\mathbb{K}}$, a nonnegative integer s, and constants $c_{i,j}\in\overline{\mathbb{K}}$ for $1\leq i\leq e$ and $0\leq j\leq s$ such that*

$$f(n)=\sum_{i=1}^{e}\sum_{j=0}^{s}c_{i,j}\binom{n+j-1}{j}\lambda_i^n$$

for all $n\geq 0$. Here,

$$d-e=m_1+\cdots+m_e,$$

where m_i is the largest index $j\leq s$ such that $c_{i,j}\neq 0$ for $i=1,\ldots,e$.

Proof. See Exercises 7.1–7.3. □

7. Exercises

Exercise 7.1. Prove the equivalence $(a) \iff (b)$ in Theorem 6.5.

Hint: For the direction $(a) \implies (b)$, let A be the $d \times d$ *companion matrix* associated to the recurrence in Equation (6.1) given by

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -c_0 & -c_1 & -c_2 & \cdots & -c_{d-1} \end{bmatrix},$$

and let

$$\mathbf{w} = \begin{bmatrix} f(0) \\ f(1) \\ \vdots \\ f(d-1) \end{bmatrix}.$$

Then show for all $n \geq 0$, we have

$$\begin{bmatrix} f(n) \\ f(n+1) \\ \vdots \\ f(n+d-1) \end{bmatrix} = A^n \mathbf{w}.$$

Now show we can extract $f(n)$ by left-multiplying by an appropriate row vector. For the other direction, use the Cayley-Hamilton theorem.

Exercise 7.2. Prove the equivalence $(a) \iff (c)$.

Hint: Let $Q(x) = 1 + c_{d-1}x + \cdots + c_0 x^d$, and show using the recurrence that

$$\left(\sum_{n \geq 0} f(n) x^n \right) \cdot Q(x)$$

is a polynomial of degree at most $d-1$. For the direction $(c) \Rightarrow (a)$, show that the polynomial $Q(x)$ gives a linear recurrence satisfied by $f(n)$.

Exercise 7.3. Prove the equivalence $(c) \iff (d)$ in Theorem 6.5.

Hint: For the direction $(c) \Rightarrow (d)$, factor

$$Q(x) = (1 - \lambda_1 x)^{p_1} \cdots (1 - \lambda_e x)^{p_e}$$

with $\lambda_1, \ldots, \lambda_e$ distinct nonzero elements of $\overline{\mathbb{K}}$ and integers $p_1, \ldots, p_e \geq 1$. Use the partial fractions decomposition of $P(x)/Q(x)$ and the general form of the binomial theorem, which implies that for any $\lambda \in \overline{\mathbb{K}}$ and $j \geq 1$,

$$\frac{1}{(1 - \lambda x)^j} = \sum_{n \geq 0} \binom{n+j-1}{j-1} \lambda^n x^n.$$

For the direction $(d) \Rightarrow (c)$, reverse the partial fractions argument to show that $\sum f(n)x^n$ is the power series expansion of a rational function $P(x)/Q(x)$ with $P(x), Q(x) \in \overline{\mathbb{K}}[x]$ relatively prime, where $\deg Q(x) = d$, $Q(0) = 1$, and $\deg P(x) \leq d-1$. Finally, use the fact that the sequence $f(n)$ takes values in $\mathbb{K}$ to conclude that the coefficients of both $P(x)$ and $Q(x)$ must lie in $\mathbb{K}$.

Exercise 7.4 (Möbius inversion)**.** The Möbius function $\mu : \{1, 2, 3, \ldots\} \to \{-1, 0, 1\}$ is defined by:

$$\mu(n) = \begin{cases} 0 & \text{if } n \text{ has a square factor greater than 1,} \\ (-1)^s & \text{if } n \text{ is the product of } s \text{ distinct primes.} \end{cases}$$

Prove the Möbius inversion formula: if $f(n)$ and $g(n)$ are sequences related by

$$f(n) = \sum_{d|n} g(d),$$

then

$$g(n) = \sum_{d|n} \mu(d) f(n/d)$$

for all $n \geq 1$.

Exercise 7.5. Recall that primitive words were defined in Exercise 5.8. Use Möbius inversion to prove that if $f(n)$ denotes the number of primitive words of length n over a k-letter alphabet Σ, then

$$f(n) = \sum_{d|n} \mu(d) k^{n/d}$$

for $n \geq 1$.

Exercise 7.6. Let $k \geq 2$, and let $f(n)$ denote the number of primitive words of length n over a k-letter alphabet Σ. Use the preceding exercise to show that

$$\sum_{n \geq 0} f(n)x^n = 1 + \sum_{j \geq 1} \frac{\mu(j)x^j}{1 - kx^j},$$

where the right-hand side is expanded as a formal power series using the geometric series identity.

Exercise 7.7. Use the preceding exercise to show that the generating function

$$\sum_{n \geq 0} f(n)x^n$$

for the number of primitive words of length n over a k-letter alphabet Σ is not the expansion of a rational function when $k \geq 2$. Conclude that the set of primitive words over a k-letter alphabet is not a regular language.

University of Waterloo, Department of Pure Mathematics, Waterloo, Ontario, N2L 3G1, Canada

Email address: `jpbell@uwaterloo.ca`