Introduction to Automata and Theory of Computation
COL352 - Assignment 1

Tanishq Dubey (2019CS51077), Satyam Kumar Modi (2019CS50448), Rupanshu Shah (2019CS10395)

February 2022

Contents

1 Binary languages of primes

$L_1 = \{bin(p) : p \text{ is a prime number}\}$

To prove L_1 is not regular, we use pumping Lemma. Let the language L_1 be regular such that its DFA has n states. Let p be any prime number $> 2^n$, i.e. binary representation of p contains more than n-bits. Now, since the length of $bin(p) > n$, there must exist x, y, z such that $bin(p) = xyz$ such that $y \neq \epsilon$ and $xy^kz \in L_1 \; \forall k \geq 1$.

Let the length of x, y, z be l, m, n respectively. Then, $p = 2^{m+n}dec(x) + 2^n dec(y) + dec(z)$ where $dec(x)$ represents the decimal value of x.
Now, let us assume $xy^kz \in L_1$ for some k,i.e. we have $dec(xy^kz) = p'$ for some prime p' where $p' > p$. We can also say,

$p' = 2^{km+n}dec(x) + (2^n + 2^{m+n} + ... + 2^{(k-1)m+n})dec(y) + dec(z)$.
$\Rightarrow p' = 2^{km+n}dec(x) + (2^n + 2^{m+n} + ... + 2^{(k-1)m+n})dec(y) + p - 2^{m+n}dec(x) - 2^n dec(y)$
$\Rightarrow p' = p + (2^{km+n} - 2^{m+n})dec(x) + 2^n(2^m + 2^{2m} + .. + 2^{(k-1)m})dec(y)$
$\Rightarrow p' = p + 2^{m+n}(2^{(k-1)m} - 1)dec(x) + 2^{m+n}(1 + 2^m + 2^{2m} + .. + 2^{(k-2)m})dec(y)$
$\Rightarrow p' = p + 2^{m+n}(2^m - 1)(1 + 2^m + 2^{2m} + .. + 2^{(k-2)m})dec(x) + 2^{m+n}(1 + 2^m + 2^{2m} + .. + 2^{(k-2)m})dec(y)$

Let $2^m \equiv q \pmod{p}$ for some $q < p$.
$(1 + 2^m + .. + 2^{(k-2)m)}) \equiv (1 + q + q^2 + ..q^{k-2}) \pmod{p}$.
From Fermat's little theorem, we know $q^{p-1} - 1 \equiv 0 \pmod{p}$.
$\Rightarrow (q-1)(1 + q + q^2 + .. + q^{p-2}) \equiv 0 \pmod{p}$
Since $q < p$, we have $p \not| (q-1)$. This implies $p|(1 + q + q^2 + .. + q^{p-2})$. So, for $k = p$, we have $p|(1 + 2^m + .. + 2^{(k-2)m)})$
This implies $(p' - p) \equiv 0 \pmod{p} \Rightarrow p' \equiv 0 \pmod{p}$. Since, $p' > p$, only possibility is $p|p'$ which is a contradiction to the fact that p' is prime. This implies our assumption that L_1 is regular is false. Hence, L_1 is non-regular language.

2 Language of fibonacci

$\Sigma = \{a\}$, $L_2 = \{a^m \mid m = F_n \text{ for some } n\}$ where F_n is the nth fibonacci number.

We will be using pumping lemma to prove that the language L_2 is non regular. Let us assume that the language is regular such that the DFA of the language has k states. Let us choose n such that $F_{n-1} > k$. Clearly, $F_n > k$, so $x = a^{F_n}$ must have length $> k$. Thus, there exist u, v, w such that $x = uvw$ and $uv^iw \in L_2 \; \forall i \geq 1$.The length of v must be less than Or equal to k. Length of uv^2w must be $F_n + len(v)$. Since, $uv^2w \in L_2$, $F_n + len(v) \geq F_{n+1}$ which implies $len(v) \geq F_{n-1}$. But, this is a contradiction to fact that $len(v) \leq k < F_{n-1}$. So, the assumption that the language L_2 is regular is wrong. Hence, the language L_2 is non-regular.

3 half(L)

A is a regular language. We define $A_{\frac{1}{2}-} = \{x|$ for some $y, |x| = |y|$ and $xy \in A\}$. To prove that $A_{\frac{1}{2}-}$, we construct a DFA for it,then prove that that language accepted by the DFA is same as $A_{\frac{1}{2}-}$.

Let $M = (q_{0m}, F_m, \delta_m, \Sigma, Q_m)$ be the DFA for language A. We define the DFA N as follows for $A_{\frac{1}{2}-}$ as follows:

- States in Q_n are of the form (q, S) where $q \in Q$ and $S \subseteq Q$.
- $q_{0n} = (q_{0m}, F_m)$
- $\delta_n((q, S), a) = (\delta(q, a), T)$ where $T = \{q' \in Q_m|$ for some $b \in \Sigma$, $\exists p \in S \ni \delta_m(q', b) = p\}$
- $F_n = \{(q, S)|q \in S\}$

The state $(q, S) \in Q_n$ is defined as after reading some input string x from the start state (q_0, F_m), q is the state in which DFA M would be after reading string x and S maintains the set of those states such that there is a path from these states to an accepting state in M that has the same length as the string x.

We now prove that $L(N) = A_{\frac{1}{2}-}$.

- **Part-I** First we prove that $L(N) \subset A_{\frac{1}{2}-}$. Let x be a string accepted by the DFA N. We prove that $x \in A_{\frac{1}{2}-}$. After reading input x, let's say the DFA N is at state (q, S) where $q \in Q_m$ and $S \subseteq Q_m$. The state (q, S) must be a final state in N. This means $q \in S$(by our definition of N). Since, $q \in S$, there exists a string w such that $|w| = |x|$ and $\delta_m(q, w) \in F_m$. Hence, the string $xw \in A$, as the state after reading the string xw is a final state in M. Thus, $x \in A_{\frac{1}{2}-}$. This implies $L(N) \subset A_{\frac{1}{2}-}$.

- **Part-II** Now, we prove that $A_{\frac{1}{2}-} \subset L(N)$. Let $x \in A_{\frac{1}{2}-}$. We prove that $x \in L(N)$. Since, $x \in A_{\frac{1}{2}-}$, there exists a string w such that $|w| = |x|$ and $xw \in A$. Since, $xw \in A$, we have $\delta_m(q_0, xw) \in F_m$. Let (q, S) be the state after reading the string x through the DFA N, i.e. $\delta_n((q_0, F_m), x) = (q, S)$ where $\delta_m(q_0, x) = q$ and S is the set of states such that there is a path from states in S to a final state in F_m of the same length as x. Since, $\delta_m(q_0, xw) = \delta_m(\delta_m(q_0, x), w) = \delta_m(q, w) \in F_m$. Thus, we have a string w of the same length as x such that $\delta_m(q, w) \in F_m$. Therefore, q must belong to S. Hence, $x \in L(N)$ and this implies that $A_{\frac{1}{2}-} \subset L(N)$.

  From Part-I and Part-II, we have $L(N) = A_{\frac{1}{2}-}$. Since, there exists a DFA N for the language $A_{\frac{1}{2}-}$, the language $A_{\frac{1}{2}-}$ is regular.

4 Middle String Removed

Claim 4.1: Let us define a regular language $L = a^*bc^*$ over the alphabet $\Sigma = \{a, b, c\}$, then $M = L_{1/3-1/3} \cap \{a^*c^*\}$ is equivalent to $\{a^nc^n|n \geq 0\}$.

Proof by Deduction:

- **Part I:** We show $M \subseteq \{a^nc^n|n \geq 0\}$, let string $xz \in M$, such that there exists $xyz \in L$ with $|x| = |y| = |z|$. Note that since $xz \in a^*c^*$, it does not contain b, and hence y is of the form a^*bc^*. Clearly, now b does not exist in the string x or z. Therefore, x is of the form a^* and z of c^*. Since, xz is a string with $|x| = |z|$, we conclude that $xz \in \{a^nc^n|n \geq 0\}$. This proves what was required.
- **Part II:** We show $\{a^nc^n|n \geq 0\} \subseteq M$, take any string a^kc^k for fixed $k \geq 0$. Now, the string $s = a^{2k-1}bc^k \in L$. Clearly for $xyz = s$, such that $|x| = |y| = |z|$, we have $xz = a^kc^k$. Thus, we have shown string a^kc^k for fixed $k \geq 0$ belongs to M. This proves what was required.

Using the above two claims, we can conclude that M is equivalent to the language $\{a^nc^n|n \geq 0\}$. ■

Claim 4.2: If a language L is regular, then $L_{1/3-1/3}$ may not be regular.

Proof by Deduction:

Simply take the language $L = a^*bc^*$ as defined above, let if possible $L_{1/3-1/3}$ be regular. Since regular languages are closed under intersection, then the language $L \cap \{a^*b^*\} = M$ must also be regular. Using Claim 4.1, we know that this is not possible since $M = \{a^nc^n|n \geq 0\}$ which is non regular as shown using pumping lemma in the lecture. Hence, we have a contradiction and we conclude $L = a^*bc^*$ is regular but not $L_{1/3-1/3}$. ■

5 Language of 2-NFA

a) We prove both the directions.

Suppose x is not accepted by A.

$\forall 0 \leq i \leq n+1$ define W_i as the set of all possible states q such that configuration (q, i) is reached by some sequence of transitions while reading string x by A. We claim that these W_i's are the required sets.

All start states have pointer at index 0 initially i.e., we have configuration $(s, 0) \forall s \in S$. Thus, $S \subseteq W_0$

Now if $u \in W_i,\ 0 \leq i \leq n$ and $(v, R) \in \Delta(u, a_i)$, then there is a sequence of (not necessarily distinct) states $p_1, p_2, \ldots, p_r = u$ from configuration $(s, 0)$ to (u, i). This implies we have a sequence of states $p_1, p_2, \ldots, p_r = u, p_{r+1} = v$ from configuration $(s, 0)$ to $(v, i+1)$ where the last transition is as per $(v, R) \in \Delta(u, a_i)$. This implies $v \in W_{i+1}$.

Similarly, if $u \in W_i,\ 1 \leq i \leq n+1$ and $(v, L) \in \Delta(u, a_i)$, then there is a sequence of (not necessarily distinct) states $p_1, p_2, \ldots, p_r = u$ from configuration $(s, 0)$ to (u, i). This implies we have a sequence of states $p_1, p_2, \ldots, p_r = u, p_{r+1} = v$ from configuration $(s, 0)$ to $(v, i-1)$ where the last transition is as per $(v, L) \in \Delta(u, a_i)$. This implies $v \in W_{i-1}$.

Since, string x is not accepted by A, therefore configuration $(t, n+1)$ is not attainable when reading x by A. Since configuration $(t, n+1)$ is not attainable, therefore $t \notin W_{n+1}$

Since, the aforementioned set of states satisfies the conditions, therefore *"x not accepted by A"* implies *"There exist sets of states $W_i \subseteq Q, 0 \leq i \leq n+1$ satisfying the given conditions."*

Suppose x is accepted by A.

Since, x is accepted, therefore configuration $(t, n+1)$ is reached starting from $(s, 0)$ through some sequence of states when reading x by A. Let the sets of states $W_i, 0 \leq i \leq n+1$ be defined as per the given 4 conditions. Let $P = (p_0, l_0), (p_1, l_1), (p_2, l_2), \ldots, (p_r, l_r)$ be a sequence of configurations corresponding to string x read by A that accepts the string. Since, the string is accepted, therefore $p_r = t, l_r = n+1$. Since final configuration is $(t, n+1)$, therefore $t \in W_{n+1}$ Therefore, *x is not rejected* implies *There exist no sets of states $W_i \subseteq Q, 0 \leq i \leq n+1$ satisfying all the 4 conditions.*

Hence Proved.

b) We proceed by proving that Language rejected by A is a regular language. Then by closure of Regular Languages under complementation, we conclude $L(A)$ is a Regular Language.

First we remove ϵ transitions from the $2-NFA$. Then we use subset construction to convert $2-NFA$ to $2-DFA$. Since, the language accepted by $2-DFA$ is Regular Language, therefore $\overline{L(A)}$ is a Regular Language. By closure of Regular languages under complementation, $L(A)$ is a Regular Language.

Hence Proved.

6 Synchronizing Sequence

Claim 6.1: Let us consider a synchronizable DFA M with n states, then for any two distinct states, a, b, there exists a word w of states such that $\delta(a, w) = \delta(b, w)$ and the length of w is at most $n(n-1)/2$.

Proof by Contradiction:

- **Part I:** Let if possible no such w exist, then we can conclude that there is no word that ever takes a and b to the same state which is a contradiction to the synchronizability of M.
- **Part II:** Let us say that the minimum length possible of such a w be $|w| = m$ and w takes a and b to c using the paths $a_0, a_1, a_2, ...a_m$ and $b_0, b_1, b_2, ...b_m$, where $a_0 = a$, $b_0 = b$ and $a_m = b_m = c$. After the i^{th} step, we are at the pair (a_i, b_i), where $a_i \neq b_i$ if $i < m$. Let if possible, $m > n(n-1)/2$. Clearly, since there are n states, then we have a total of ${}^nC_2$ ways of choosing an ordered pair of states. Therefore, by this claim we conclude that there must exist a pair of steps i, j such that WLOG $i < j$ and we have either $(a_i, b_i) = (a_j, b_j)$ or $(a_i, b_i) = (b_j, a_j)$. We will show a contradiction on minimality of w in both these cases.
  - **Case I:** We have $(a_i, b_i) = (a_j, b_j)$, then the steps traversed by the run of w will be of the form
    $(a_0, b_0), (a_1, b_1), ...(a_i, b_i), (a_{i+1}, b_{i+1})...(a_i, b_i), (a_{j+1}, b_{j+1}), ...(a_m, b_m)$
    where $a_m = b_m = c$. Clearly, we can shorten this path in the following way, $(a_0, b_0), (a_1, b_1), ...(a_i, b_i), (a_{j+1}, b_{j+1}), ...(a_m, b_m)$, i.e we remove the steps from $i+1$ to j. Note that since $j > i$, at least one step is removed.
  - **Case II:** We have $(a_i, b_i) = (b_j, a_j)$, then the steps traversed by the run of w will be of the form
    $(a_0, b_0), (a_1, b_1), ...(a_i, b_i), (a_{i+1}, b_{i+1})...(b_i, a_i), (a_{j+1}, b_{j+1}), ...(a_m, b_m)$
    where $a_m = b_m = c$. Clearly, we can shorten this path in the following way, $(a_0, b_0), (a_1, b_1), ...(a_i, b_i), (b_{j+1}, a_{j+1}), (b_{j+2}, a_{j+2}), ...(b_m, a_m)$, i.e, we remove the steps from $i+1$ to j and invert (a_k, b_k) to (b_k, a_k) for all the $k > j$. Note that since $j > i$, at least one step is removed.
- Thus we arrive at a contradiction on minimality of w and conclude that $|w| \leq n(n-1)/2$ as required. ■

Definition: Let us consider a synchronizable DFA, M with set of states Q with size n, then we define the transition function $\Delta : S \times w \to R$, where $S, R \subseteq Q$ and w is an input string and $R = \{\delta(q, w) | q \in S\}$.

Algorithm: We define the following algorithm to get the synchronizing sequence of M.

> **Step 1:** Start with $Q_0 \leftarrow Q$ and pick any two states $a, b \in Q_0$ and find a word w_1 such that $\delta(a, w_1) = \delta(b, w_1)$. Such a word always exists by Claim 6.1.
>
> **Step 2:** Obtain $Q_1 \leftarrow \Delta(Q_0, w_1)$. Note that $|Q_1| \leq |Q_0| - 1$.
>
> **Step 3:** Repeat Steps 1 and 2 for the new set Q_1 to obtain Q_2 and w_2. Similarly, keep repeating process until we reach Q_t such that $|Q_t| = 1$. Note that $t \leq n - 1$ as $|Q_{i+1}| \leq |Q_i| - 1$.
>
> **Step 3:** Return $w_1 w_2 w_3 ... w_t$ as the synchronizing sequence.

Claim 6.2: For a synchronizable DFA M, the above algorithm always terminates and return returns a synchronizing sequence.

Proof by Deduction:

> **Part I:** Note that at every step we can guarantee that $|Q_{i+1}| \leq |Q_i| - 1$ because by definition, w_{i+1} takes at least two states in Q_i to the same state in Q_{i+1}. Thus, in every step the size of Q_j reduces by at least 1, thus we can conclude that the algorithm terminates in t steps where $t \leq n - 1$, because $|Q_0| = n$ and the termination condition is $Q_j = 1$.
>
> **Part II:** Now, we will show that any state $q \in Q = Q_0$ lands in the same state $q_t \in |Q_t|$ for the string $w_1 w_2 ... w_t$. Observe that after running the string upto w_i, by definition of the algorithm, $\delta(q_t, w_1 w_2 ... w_i) \in Q_i$. Thus, for $i = t$, we have that any state q ends up at q_t as it is the only state in Q_t, thereby proving that $w_1 w_2 ... w_t$ is in fact the synchronizing sequence.
>
> We have completed the proof for both parts as required and hence proved the correctness of the algorithm. ■

Claim 6.3: The synchronizing sequence of the DFA M with n states is at most of length $n(n-1)^2/2$.

Proof by Deduction:

Clearly, we can observe that $|w_i| \leq n(n-1)/2$ by Claim 6.1 and hence, we conclude that $|w_1 w_2 .. w_t| \leq t(n)(n-1)/2$. We also know that $t \leq n - 1$, thus we arrive at the upper bound as $n(n-1)^2/2 \leq n^3$. ■

7 Language of strings less than θ

Claim 1: We claim that rational numbers have terminating, or non-terminating recurring representation in binary.

Proof:

Let $x = a/b$ be the rational number.
If x has a terminating decimal representation, then it has a terminating binary representation as well.
Let us suppose x has non-terminating decimal representation. If b is even, split $x = c/d + e/f$ where $d = 2^k, k > 0$ and f is *odd*. Since d is a power of 2, therefore c/d has a terminating binary representation. This implies e/f has a non-terminating binary representation. This implies $1/f$ has a non-terminating binary representation, where f is odd.
Thus, we consider the case where $x = 1/q$ where q has a non-terminating decimal(and non-terminating binary) representation. If $1/q$ has a recurring binary representation, then so does p/q and $p/q + c/d$ where $d = 2^k, k > 0$.

So we focus on binary representation of $1/q$. Since q and 2 are co-prime , therefore by Euler's theorem there exists positive integer m such that $2^m \equiv 1 (mod\ q)$.
This implies $2^m - 1 = \lambda \cdot q$.
$\implies \frac{1}{q} = \frac{\lambda}{2^m - 1}$. Now $\frac{\lambda}{2^m-1}$ has a recurring non-terminating binary representation if ond only if $\frac{1}{2^m-1}$ has a recurring non-terminating binary representation.

Consider $z = (0.\overline{00\ldots1})_2$, where the recurring length in m.
$2^m \cdot z = (1.\overline{00\ldots1})_2$. This implies $(2^m - 1) \cdot z = 1$.
This implies $z = \frac{1}{2^m-1}$. Therefore, $\frac{1}{2^m-1}$ has a recurring non-terminating binary representation.

Therefore, rational numbers have either terminating or non-terminating recurring binary representation.
Hence proved

Now we use *Claim* 1 to prove that L_θ is regular language if and only if θ is rational.

"θ is rational $\Rightarrow L_\theta$ is Regular Language"

Proof:

θ is rational implies θ has either terminating or non-terminating recurring. Let w be the binary representation of θ. with length n.

Case 1: w is terminating

Length of string w is n. The Language accepts all strings whose prefixes are less than w and those strings which are of the form $w \cdot (0^*)$.
Let p be a prefix less than w. We have Regular expression $R_p = p(0+1)^*$ corresponding to this prefix.
Since, there are finitely many prefixes less than w, we have finite union of Regular Expressions as:
$RegularExpression(L_\theta) = \left(\bigcup_{p=prefix<w} RegularExpression\ R_p\right) \bigcup w \cdot (0^*)$.
Therefore, L_θ is a Regular Language.

Case 2: w is non-terminating recurring

Regular expression for w is $w = (x)(y^*)$ where y is the repeating part.
Let p be a prefix. We have Regular expression $R_p = p(0+1)^*$ corresponding to this prefix.
Let $R_x = \left(\bigcup_{p=prefix<x} RegularExpression\ R_p\right)$
Let $R_y = \left(\bigcup_{p=prefix<y} RegularExpression\ R_p\right)$
The Language accepts all strings which are of the form $x \cdot (0^*)$, and those strings s in which at the first point of difference from w, the symbol in w is 1 and the symbol in s is 0.
In other words, $RegularExpression(L_\theta) = (R_x)\ +\ (x \cdot (y^*) \cdot R_y)\ +\ (x \cdot (y^*))$.
Since this is a finite union, Therefore, L_θ is a Regular Language.

"θ is irrational $\Rightarrow L_\theta$ is not a Regular Language"

Proof:

We use the pumping lemma to prove that L_θ is not a regular language.
Binary Representation of an irrational number is non-terminating, non-recurring.

Pumping Lemma:
For given $n \geq 1$, let $v =$ first n symbols of $binary(\theta)$
Choose $w \in L$ as $w =$"first r symbols of $binary(\theta)$" where r is such that . $w \neq ab^k$ for any partition of v into a, b where $|b| > 0$
We can do this with r finite because $binary(\theta)$ is non-recurring. Let x, y, z be some break-up of w such that $|y| > 0, |xy| \leq n$.
We consider the following cases:

Case 1: y is greater than z.
Choose $i = 2$.
$w' = xy^iz = xyyz$.
$y > z \Rightarrow xyyz > xyz$.
$\Rightarrow w' = xyyz > w$.
$\Rightarrow w' \notin L_\theta$.

Case 2: y is less than z.
If y is not prefix of z then:
Choose $i = 0$.
$w' = xy^i z = xz$.
$z > y \Rightarrow w' = xz > xy(1+0)^*$
$\Rightarrow w' = xz > w$.
$\Rightarrow w' \notin L_\theta$.

If y is prefix of z then Let λ be the number such that $binary(\theta) = xy^\lambda q$ where y is not prefix of q. Since, the binary representation of θ is non-recurring, therefore λ is finite. Let m be the length of y. Let q' be the string formed by first m letters of q.
$q' \neq y$.
If $q' > y$ then: Choose $i = 0$.
$w' = xy^i z = xz$.
$q' > y \Rightarrow z > yz \Rightarrow xz > xyz$
$\Rightarrow w' = xz > w$
$\Rightarrow w' \notin L_\theta$.

If $q' < y$ then: Choose $i = 2$.
$q' < y \Rightarrow yyz > yz \Rightarrow xyyz > xyz$
$\Rightarrow w' = xyyz > w$
$\Rightarrow w' \notin L_\theta$.

Therefore, by pumping lemma, we have L_θ is not a Regular Language.

Since, *"θ is rational $\Rightarrow L_\theta$ is Regular Language"* and *"θ is irrational $\Rightarrow L_\theta$ is not a Regular Language"*
Therefore L_θ is Regular Language if and only if θ is Rational.

Hence proved.