COL352
Introduction to Automata and Theory of Computation
Problem Set 2

T Abishek - 2019CS10407, Gaurav Jain - 2019CS10349

---

1. Prove that $L_1 = \{bin(p) : p \text{ is a prime number}\}$ is not a regular language.

   Solution:

   Proof by Contradiction:
   Assume the language L_1 to be regular. So, it should follow the refined pumping lemma discussed in the lectures. Consider a prime p whose binary form ends with n ($n >$ pumping length) zeros followed by a 1. p can be represented as:

   $$p = (...bin(m)...)(00..n\ zeros..00)1 = m.2^{n+1} + 1$$

   Where, $bin(m)$ is the binary form of some integer m. Such a prime number p is guaranteed by theorems related to prime numbers (Dirichlet Theorem for primes).
   We use the refined version of pumping lemma where $uwv \in L_1$ and $|w| >$ pumping length. Here $u = bin(m)$ and $v = 1$ and we will pump the string w. w contains n zeros and w can be split into 3 parts x with r zeros, y with s zeros and z with t zeros. So, $r + s + t = n$ and $s \neq 0$.
   For some k,

   $$uxy^kzv = (...bin(m)...)(00..r + ks + t\ zeros..00)1$$
   $$uxy^kzv = (...bin(m)...)(00..n + (k-1)s\ zeros..00)1$$

   Using the value of p,

   $$uxy^kzv = m.2^{n+(k-1)s+1} + 1 = (p-1).2^{(k-1)s} + 1$$

   For all values of k, $q = uxy^kzv$ should be prime. Let $k = p$.
   So,

   $$q = uxy^kzv = (p-1).2^{(p-1)s} + 1 = p.2^{(p-1)s} - (2^{(p-1)s} - 1)$$

   From Fermat's little theorem,

   $$2^{(p-1)} \equiv 1 (mod\ p)$$

   Thus,

   $$q\ mod\ p \equiv 0 + 0\ mod\ p \equiv 0\ mod\ p$$

   But, q should be a prime number. This is a contradiction and thus, language L_1 is not regular.

   □

---

2. The n-th Fibonacci number is defined as $F_1 = 1, F_2 = 1$, and for all $n \geq 3, F_n = F_{n-1} + F_{n-2}$. Consider the language over $\Sigma = \{a\}$

   $$L_2 = \{a^m \mid m = F_n\}$$

   Is L_2 regular? Justify your answer.

Solution:

Consider an equivalence relation $\equiv$ on $\{a\}^*$. Suppose it is a right congruence and it refines L_2. To show L_2 is regular, the index of this relation should be finite.
Suppose the index is finite, then there exists m, n such that $n > m$ and $F_{n-1} > F_{m-1}$ and

$$[a^{F_m}] = [a^{F_n}]$$
$$[a^{F_m}][a^{F_{m-1}}] = [a^{F_n}][a^{F_{m-1}}]$$
$$[a^{F_{m+1}}] = [a^{F_n+F_{m-1}}]$$

$a^{F_{m+1}} \in L_2$ but $F_{m-1} < F_{n-1}$ and $F_n + F_{m-1} < F_n + F_{n-1}$ and thus, $F_n + F_{m-1} < F_{n+1}$.
So, $a^{F_n+F_{m-1}} \notin L_2$ which is a contradiction and hence the index of this equivalence relation can not be finite and by Myhill-Nerode Theorem, L_2 is not regular.

---

3. If A is any language, let $A_{\frac{1}{2}-}$ denote the set of all first halves of strings in A so that

$$A_{\frac{1}{2}-} = \{x \mid \text{for some } y, |x| = |y| \text{ and } xy \in A\}$$

Show that if A is regular, then so is $A_{\frac{1}{2}-}$.

Solution:

Since the language A is regular so there exists a NFA $M = (Q, \Sigma, \delta, q_0, F)$ that accepts the language A. To show the language $A_{\frac{1}{2}-}$ is regular, we need to construct a NFA or DFA which accepts this language.
We construct the NFA N for $A_{\frac{1}{2}-}$ by running the NFA M on an input symbol and running the same NFA M backwards from an accepted state on some symbol $v \in \Sigma$. The string is accepted if after running M both forward and backward we land in the same state. This indicates that the second half can be appended to the input string to produce a string which is accepted by the NFA M accepting A.
Formally, assume that the NFA M has only one accepting state (which can be constructed by merging final states). Let $N = (Q', \Sigma, \delta', q_0', F')$ where,

i. $Q' = Q \times Q$.
ii. For $q, r \in Q$, $\delta'((q, r), a) = \{(u, v) \mid u \in \delta(q, a) \text{ and } r \in \delta(v, b) \text{ for some } b \in \Sigma\}$.
iii. $q_0' = (q_0, q_{final})$ since there is only one final state in M.
iv. $F' = \{(q, q) \mid q \in Q\}$.

Thus, by constructing NFA N using NFA M which accepts A we show that the language $A_{\frac{1}{2}-}$ is regular and $A_{\frac{1}{2}-} = L(N)$.

---

4. If A is any language, let $A_{\frac{1}{3}-\frac{1}{3}}$ denote the set of strings in A with the middle-third removed so that

$$A_{\frac{1}{3}-\frac{1}{3}} = \{xz \mid \text{for some } y, |x| = |y| = |z| \text{ and } xyz \in A\}$$

Show that if A is regular, then $A_{\frac{1}{3}-\frac{1}{3}}$ is not necessarily regular.

Solution:

Consider a regular language $A = \{a^*cb^*\}$, this language can be represented by a NFA of 3 states. Consider the intersection of $A_{\frac{1}{3}-\frac{1}{3}}$ with $\{a^*b^*\}$ (which is also regular). For the intersection between $A_{\frac{1}{3}-\frac{1}{3}}$ and $\{a^*b^*\}$ to be non-empty, the symbol c in language A should be in the middle part of the string and thus, $\#a = \#b$. So,

$$A_{\frac{1}{3}-\frac{1}{3}} \cap \{a^*b^*\} = \{a^n b^n \mid n \geq 0\}$$

The intersection of two regular languages is regular. So, for $A_{\frac{1}{3}-\frac{1}{3}}$ to be regular, its intersection with $\{a^*b^*\}$ should be regular. But $\{a^nb^n\}$ is not regular.

Hence, if A is regular then $A_{\frac{1}{3}-\frac{1}{3}}$ is not necessarily regular.

---

5. A 2-NFA A is a 5-tuple $A = (Q, S, t, F, \Delta)$ where Q is the set of states, S the set of start states, t is an accept state, the transition function

$$\Delta : Q \times (\Sigma \cup \{\#, \$\}) \to 2^{Q \times (\{L,R\})}$$

Assume that whenever M accepts, it does so my moving the head (pointer) all the way to the right endmarker $\$$ and entering accept state t. In the subsequent two questions, we will try to prove that 2-NFAs accept only regular languages.

(a) Let $x = a_1 \dots a_n \in \Sigma^*, a_i \in \Sigma$, $1 \leq i \leq n$. Let $a_0 = \#, a_{n+1} = \$$. Argue that x is not accepted by A if and only if there exist sets $W_i \subseteq Q, 0 \leq i \leq n+1$ such that the following hold:

- $S \subseteq W_0$
- If $u \in W_i$, $0 \leq i \leq n$, and $(v, R) \in \Delta(u, a_i)$, then $v \in W_{i+1}$
- If $u \in W_i$, $1 \leq i \leq n+1$, and $(v, L) \in \Delta(u, a_i)$, then $v \in W_{i-1}$ and
- $t \notin W_{n+1}$.

Solution: Let $x = \#a_1 \dots a_n\$$ be not accepted by A.
To Prove: Sets $W - i$ exists such that the conditions hold
As the sets W themselves are defined by the run of x in A, we define them as, all the states , from which we read a_i, belong to W_i. Using this definition, all the required properties are satisfied as shown below:

As $\#(a_0)$ is read from the starting state s, which belongs to S, it belongs to W_0. $S \subseteq W_0$ is satisfied.

Let $u \in W_i$ and $0 \leq i \leq n$. This means that the we read the letter a_i when at the state u (according to definition), in the run.

There are 2 cases:

Case 1: The pointer moves right
This implies that $(v, R) \in \Delta(u, a_i)$. This also means that we then read a_{i+1} when at the state v. According to our definition, this means that v belongs to W_{i+1}. Satisfied.

Case 2: The pointer moves left
This implies that $(v, L) \in \Delta(u, a_i)$. This also means that we then read a_{i-1} when at the state v. According to our definition, this means that v belongs to W_{i-1}. Satisfied.

As x isn't accepted by A, this means that the state t doesn't read the $ marker. According to our definition, this means that $t \notin W_{n+1}$. Satisfied.

In the converse case, let the conditions on the sets W_i hold.
To Prove: x isn't accepted by A.

Proof by contradiction:
Assume x is accepted by A.
Let the run of x in A be, (a_0, d_0), (a_1, d_1), ... (a_m, d_m). As x is accepted by A, a_m is t and d_m is R. a_0 is s which belongs to S which belongs in W_0. In the start case, d_1 is R (as it came from # pointer). Thus, $a_1 \in W_1$ due to the conditions. By induction we can show that $a_j \in W_j$ for all m. The base case is trivial as by defintion $a_0 \in S \subseteq W_0$. Assume $a_k \in W_k$ for $k < m$. By defintion of transition function, (v,L) or (v,R) $\in \Delta(a_k, d_k)$ for some v. This implies that $v = s_{k+1} \in W_{k+1}$. This is a contradiction as by the given conditions $t \notin W_{n+1}$ but by induction $a_m = t \in W_{n+1}$.

(b) Using the previous part, show that $L(A)$ is regular.

Solution:

We will show that $L(A)$ is regular by constructing a NFA B which accepts a word w if and only if A rejects w. The NFA $B = (Q', \Sigma, Q_0, F, \delta)$ is constructed as below:

- $Q' = (2^Q \times 2^Q) \cup 2^Q$
- $\Sigma' = \Sigma \cup \{\#, \$\}$
- $Q'_0 = \{U : S \subseteq U \subseteq Q\}$ is the set of start states
- $F' = \{U : U \cap \{t\} = \phi\} \cup \{(U, V) : V \cap \{t\} = \phi\}$ where t is the accept state of 2NFA.
- $\delta' \subseteq Q' \times \Sigma \times Q'$ where δ' is defined as:
  $\delta'(T, a) = \{(T, U) : If\ s \in T\ and\ (t, R) \in \Delta(s, a)\ then\ t \in U\}$
  $\delta'(U, a) = \{(T, U) : If\ s \in T\ and\ (t, L) \in \Delta(s, a)\ then\ t \in U\}$
  $\delta'((T, U), a) = \{(U, V) : If\ s \in U\ and (t, L) \in \Delta(s, a)\ then\ t \in T\}$
  $\delta'((T, U), a) = \{(U, V) : If\ s \in U\ and (t, R) \in \Delta(s, a)\ then\ t \in V\}$

Let there exist subsets w_0, w_1 , ...w_{n+1} of Q satisfying the above given conditions. From construction we have, $\delta'(w_i, a_i) = (w_i, w_{i+1})$ for $0 \leq i \leq n$ and $\delta'(w_i, a_i) = (w_i, w_{i-1})$ for $1 \leq i \leq n+1$ and $\delta'((T, w_i), a_i) = (w_i, w_{i+1})$ for $0 \leq i \leq n$.

Since w_0 is a start state we have that on reading a_0 , we reach the state (w_0, w_1) and from there on reading a_1 we can reach (w_1, w_2) and so on till (W_n, W_{n+1}). On reading a_{n+1} we reach the states w_{n+1}, V.
If w is rejected by 2NFA A ,we have that $w_{n+1} \cap \{t\} = \phi$ and hence (w_{n+1}, w_{n+1}) will be a final state i.e., B accepts w iff A rejects w. Since regular languages are closed under complement and B accepts the complement of $L(A)$ so $L(A)$ is regular.

---

6. Let $M = (Q, \Sigma, q_0, \delta, F)$ be a DFA and let h be a state of M called its "home". A synchronizing sequence for M and h is a string $s \in \Sigma^*$ where $\hat{\delta}(q, s) = h$ for every $q \in Q$. Say that M is synchronizable if it has a synchronizing sequence for some state h. Prove that if M is a k-state synchronizable DFA, then it has a synchronizing sequence of length at most k^3. Can you improve upon this bound?

Solution:

For M to have a synchronising sequence, any pair of of states p, q has a word w such that $\delta(p, w) = \delta(q, w) = h$, where h is the home state. While traversing the word w from the states p and q, we would come across a pair of states for every letter in the word w.
For the objective of minimising the length of w, we can remove any unique pair of states that repeats in this sequence by removing the part of the string between the two repeats. This doesn't affect the ability for the word w to reach h and in the same time, reduces the length of w.
So, finally we have a word w, where all the pairs of states in the sequence are distinct. As there are a maximum of k^2 distinct pairs in a k state DFA, the the maximum length of w is k^2.

Suppose we do the same above process and run k copies of M from all k states, where each copy starts at a different state. We feed a input string s to these copies. For s to be a synchronising sequence, all the k copies have to end at state h.
During this process, when 2 copies end up at a common state l in between, the rest of the run after l would be the same for both. We can remove one of the copies for this simulation. So, at the end of the synchronising sequence we would have only 1 copy and the rest have been eliminated. There are a total of $k-1$ eliminations.
As explained above, the minimum length of a string which leads to 2 copies ending up at the same state can be a maximum of k^2. So the minimum length of s can have a maximum length of $(k-1).k^2 = k^3$.

Therefore, proved. □

---

7. For every string $x \in \{0,1\}^+$ consider the number

$$0.x = x[1] \cdot \frac{1}{2} + x[2] \cdot \frac{1}{2^2} + \cdots + x[|x|] \cdot \frac{1}{2^{|x|}}$$

where $|x|$ is the length of x. For a real number $\theta \in [0,1]$ let

$$L_\theta = \{x : 0.x \leq \theta\}$$

Prove that L_θ is regular if and only if θ is rational.

Solution: We know that if θ is rational, its binary expansion is terminating or non terminating and repeating. And if it's irrational, is is non-terminating and non-repeating.

Proof by cases:
Case 1: θ is rational

Case 1.1: The binary expansion of θ is terminating.
As the binary expansion is terminating, it can be represented as $\theta = 0.a_1a_2 \ldots a_n$ where a_n is the last bit needed to represent the number. So, in any input x of arbitrary size, only the first n bits are necessary to compare it with θ.

It can be defined as set S = 0.x : $x_1 \ldots x_n$ ¡ $0.a_1a_2 \ldots a_n \cup \theta$. It can be seen that all elements of S lie in L_θ. Conversely, take any x belonging to L_θ. By definition, it belongs to S. So S = L_θ.

As there are a maximum of atmost 2^n equivalence classes in L_θ (as there are only n bits being used for comparison), there are finite equivalence classes and thus the language is regular.

Case 1.2: THe binary expansion of θ is non terminating but repeating.

Let us assume that θ is written as $\theta = 0.a_1a_2 \ldots a_n\overline{b_1 \ldots b_m}$.

Take any x: $0.x_1 \ldots x_n$, it belongs to L_θ if $0.x_1 \ldots x_n$ ¡ $0.a_1a_2 \ldots a_n\overline{b_1 \ldots b_m}$. and doesn't belong otherwise.

This can be tested with the algorithm
1) the first n bits of x all have to be same or less than the corresponding bits in θ.
2) The next m bits of x all have to be same or less than the corresponding bits in $b_1 \ldots b_m$. 3) If x isn't exhausted, compare the next m bits again with $b_1 \ldots b_m$. 4) Repeat step 3 until x is exhausted.

If x never exceeded the corresponding bit in θ, it belongs to the language or else it doesn't. So a DFA can be constructed pretty easily in this case, with n+m states along with a reject state. The input starts at s0, reaches s(n+1) at the n+1th bit, reaches s_m state at the m'th bit and loops again to s_{n+1}. All the states are accepting states excpet the reject state. As soon as a input bit is encountered which is greater than the corresponding bit in θ, it is thrown to the reject state. This DFA satisfies the language, so it regular.

Case 2: θ is irrational.
There is no pattern in the binary expansion of the the θ and it is non-terminating. And as the input x can be of any arbitrary size, a DFA would need infinite states(or infinite equivalence classes) to compare the input with θ, and compute whether to accept it or reject it, as each bit in the input would have to be compared with the corresponding bit in θ, which needs to be stored somehow in the DFA. So, as there are no finite equivalence classes, we can say that L_θ is irregular.
