CS 39: Theory of Computation **Winter '23**

Mid-Term Exam 1 — 02/13/2023

Prof. Chakrabarti ***Student: Amittai Siavava***

Credit Statement

All work on the mid-term is my own. I referred to class notes and the following books:

(i) **Introduction to the Theory of Computation** by **Michael Sipser**.

Problem 1.

Let $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ be the alphabet of all decimal digits. A string $x \in D^*$ is said to be stable if, for each pair of adjacent digits in x, those two digits have a numerical difference of at most 1. For example:

$$\text{Stable:} \quad 433321001012,\ 556677654,\ 7,\ 0000,\ \varepsilon.$$

$$\text{Unstable:} \quad 6554667,\ 1213141516,\ 7890123.$$

Give a formal description of a DFA that recognizes $L_1 = \{x \in D^* : x \text{ is stable}\}$. Provide a high-level explanation of your design idea (no formal completeness and soundness proofs are required).

$$M = (Q, D, \delta, s, F), \text{ where}$$

$$Q = \{s, r\} \cup \{q_i : i \in D\}$$

$$\delta(q, x) = \begin{cases} q_x & \text{if } x \in D \text{ and } q \in \{s, q_{x-1}, q_x, q_{x+1}\}. \\ r & \text{otherwise.} \end{cases}$$

$$F = \{s\} \cup \{q_i : i \in D\} = Q \setminus \{r\}.$$

Design Idea:

We maintain a start state, a trapping reject state, and one state for each digit.

Here is how we handle transitions:

- If we are in the start state and read a digit, we transition to the corresponding digit state.
- If we are in a digit state q_x (for the digit x) and read a digit y, then:
  - if $y = x$, we loop to the same state q_x.
  - if $y = x \pm 1$, we transition to the corresponding digit state q_y.
  - if $y \neq x$ and $y \neq x \pm 1$, we transition to the reject state r.
- If we are in the reject state r and read any symbol, we stay in the reject state.

Finally, we accept if after processing a string we are still in the start state (meaning the string is empty) or in a digit state (meaning all the adjacent digits had a difference of at most 1).

Problem 2.

For each CFG G_i;

- Describe $\mathcal{L}(G_i)$ using set notation, as simply as possible.
- Either *draw* an NFA that recognizes $\mathcal{L}(G_i)$ or *prove* that $\mathcal{L}(G_i)$ is not regular.

(a) G_1

$$S \Rightarrow 0T0 \mid 1T1$$

$$T \Rightarrow 0T0 \mid 1T1 \mid X$$

$$X \Rightarrow AX \mid A$$

$$A \Rightarrow 0 \mid 1$$

(i) $\mathcal{L}(G_1) = \left\{xwx^R : x, w \in \{0,1\}^* \text{ and } |x| > 0, |w| > 0\right\}$

(ii) *Draw* an NFA that recognizes $\mathcal{L}(G_1)$.

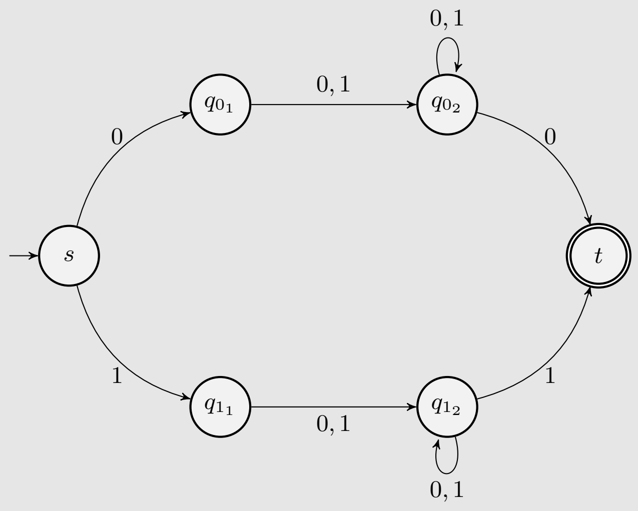

Figure 1. NFA for $\mathcal{L}(G_1)$

(b) G_2

$$S \Rightarrow 0X \mid 1Y$$
$$X \Rightarrow AXA \mid 0$$
$$Y \Rightarrow AYA \mid 1$$
$$A \Rightarrow 0 \mid 1$$

(i) $\mathcal{L}(G_2) = \left\{aw : a \in \{0,1\}, w \in \{0,1\}^* \text{ and the middle symbol of } w \text{ is } a\right\}$

(ii) *Prove* that $\mathcal{L}(G_2)$ is not regular or draw an NFA.

Claim 2.1. *$L_2 = \mathcal{L}(G_2)$ is not regular.*

Proof. Suppose L_2 is regular, and let p be the pumping length for L_2. Take $s = 10^p10^p$, then clearly $s \in L_2$. By the pumping lemma, there exists $u, v, w \in \{0,1\}^*$ such that $s = uvw$ and:

- $|uv| \leq p$
- $|v| > 0$

  This gives us two possibilities:

  - If $u = \varepsilon$, then $v = 10^a$ for some $0 \leq a \leq p-1$, and $w = 0^{p-a}10^p$.
  - If $u \neq \varepsilon$, then $u = 10^a$, $v = 0^b$, and $w = 0^{p-a-b}10^p$, where $a + b \leq p - 1$.
- $uv^kw \in L_2$ for all $k \geq 0$

  In the first case, pumping up the string tells us that $uv^2w = 10^b10^b0^{p-b}10^p \in L_2$.

  This implies that the middle symbol of $0^b10^p10^p$ is 1.

  For this to happen, either:

  - $b = 2p + 1$. This contradicts the condition that $b = |v| \leq p$ (which follows from PL1).
  - $b + p + 1 = p$. This implies that $b = -1$, which contradicts the fact that we cannot have negative-length strings.

  Therefore, in the first case L_2 must not be regular.

  In the second case, pumping down the string tells us that $uw = 10^a0^{p-a-b}10^p \in L_2$.

  This implies that the middle symbol of $0^{p-b}10^p$ is 1.

  For this to happen, we must have that $p - b = p$, meaning $b = 0$, contradicting rule PL2 which says that $b = |v| > 0$.

  Therefore, in the second case L_2 must also not be regular.

□

(c) G_3

$$S \Rightarrow AAT \mid BBT$$

$$T \Rightarrow AAT \mid BBT \mid A \mid B$$

$$A \Rightarrow 0$$

$$B \Rightarrow 1$$

(i) $\mathcal{L}(G_3) = \{0^n 1^n : n \geq 0\}$

(ii) *Draw* an NFA that recognizes $\mathcal{L}(G_3)$.

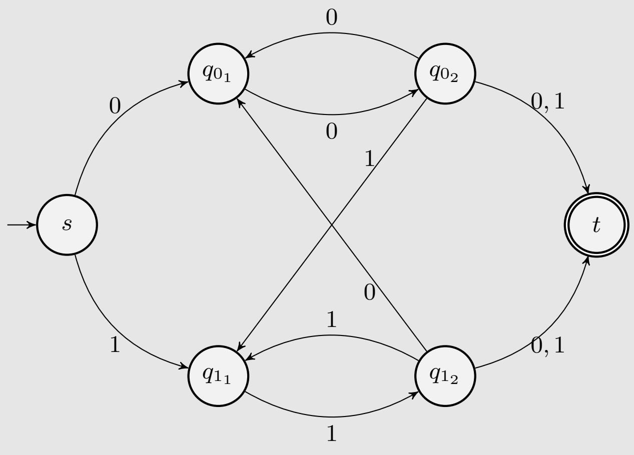

Figure 2. NFA for $\mathcal{L}(G_3)$

Problem 3.

Give a simple CFG that generates the language $L_3 = \left\{x \in \{0,1\}^* : x \neq x^R\right\}$.

Formally prove that your CFG is sound and complete.

$\underline{G_3:}$

$$S \Rightarrow 0S0 \mid 0B1 \mid 1B0 \mid 1S1$$

$$B \Rightarrow 0B0 \mid 0B1 \mid 1B0 \mid 1B1 \mid 0 \mid 1 \mid \varepsilon$$

Claim 3.1. $\mathcal{L}(G_3) = L_3$.

Proof. We need to show that G_3 is both complete $\mathcal{L}(G_3) \supseteq L_3$ and $\mathcal{L}(G_3) \subseteq L_3$.

Observations:

Let $x = x_1, x_2, \dots, x_n$ with each $x_i \in \{0,1\}$. By definition of L_3, if $x \in L_3$ then $x \neq x^R$. Therefore, $x_1x_2 \dots x_n \neq x_nx_{n-1} \dots x_1$, implying that $x_i \neq x_{n-i+1}$ for some $i \in \{1, 2, \dots, \lfloor n/2 \rfloor\}$.

Completeness:

Let $x \in L_3$, and write $x = x_1, x_2, \dots, x_n$ with each $x_i \in \{0,1\}$.

The starting symbol in G_3 is S. Let i be the smallest index such that $x_i \neq x_{n-i+1}$. Here's how we can derive x from S:

- For each $k \leq i$, we have that $x_k = x_{n-k}$, so we can write $x_kx_{k+1} \dots x_{n-k}$ as either $0S0$ or $1S1$.
- At i, we cannot write $x_ix_{i+1} \dots x_{n-i}$ as $0S0$ or $1S1$, since $x_i \neq x_{n-i+1}$. Instead, write $x_ix_{i+1} \dots x_{n-i}$ as either $0B1$ or $1B0$.
- For each $i < l \leq \lfloor n/2 \rfloor$, we do not really care whether $x_l = x_{n-l}$ since we already found an index i such that $x_i \neq x_{n-i+1}$, which guarantees that $x \neq x^R$. Follow the derivations for B to write $x_lx_{l+1} \dots x_{n-l}$ as one of $0B1, 1B0, 0B0, 1B1$.
- Finally, at index $m = \lfloor n/2 \rfloor + 1$:
  - If $m = n - m + 1$, then we have an odd-length string and we are currently at the middle element. Yield either 0 or 1 depending on the symbol in the middle of x_m.
  - Otherwise, the string has even length and we have already generated strings of length $n/2$ to the left and to the right of the current position, so we yield the empty string.

Since every string $s \in L_3$ has *at least* one index i such that $x_i \neq x_{n-i+1}$, we can always make the derivation $S \Rightarrow 0B1$ or $S \Rightarrow 1B0$ at some point in the derivation of s, and thereafter yield the full string containing only symbols from $\{0,1\}$ since since B can generate any string. Therefore, any string in L_3 can be generated by G_3.

Soundness:

Let $x = x_1, x_2, \dots, x_n$ be a string accepted by G_3. The starting symbol in G_3 is S, while only B can generate strings containing symbols exclusively from $\{0,1\}$. The only valid rules that yield B from S are $S \Rightarrow 0B1$ and $S \Rightarrow 1B0$. But x must have at least one index i such that $x_i \neq x_{n-i+1}$ for the derivation $S \Rightarrow 0B1$ or $S \Rightarrow 1B0$ to be valid. This implies that x must not be equal to x^R, since the symbol at index i in x is x_i and the symbol at index i in x^R is x_{n-i+1} and the two are not equal. Therefore, if x is accepted by G_3, then it must be the case that $x \neq x^R$ so $x \in L_3$. $\square$

Problem 4.

For each $x \in \{0,1\}^*$, define grow(x) to be the string obtained by replacing every occurrence of '0' in x with '00'. For example:

$$\text{grow}(10110) = 1001100, \quad \text{grow}(000) = 000000, \quad \text{grow}(\varepsilon) = \varepsilon, \quad \text{grow}(11) = 11.$$

Let $P = (Q, \{0,1\}, \Gamma, \delta, q_0, F)$ be a PDA. Formally describe a PDA that recognizes $\{\text{grow}(x) : x \in \mathcal{L}(P)\}$. Do not assume anything about the design of P.

Give a high-level explanation of your construction (no formal completeness and soundness proofs are required).

Define a new PDA

$$P_2 = (Q_2, \{0,1\}, \Gamma, \delta_2, q_0, F)$$

where:

$$Q_2 = \bigcup_{q \in Q} \{q, q'\}$$

$$\delta_2(q, x, \gamma) = \begin{cases} \delta(q, x, \gamma) & \text{if } q \in Q \text{ and } x \neq 0 \\ r' & \text{where } r = \delta(q, x, \gamma) \text{, if } q \in Q \text{ and } x = 0 \\ r & \text{if } q = r' \text{ for some } r \in Q \text{, and } x = 0, \gamma = \varepsilon \end{cases}$$

Main Idea:

(i) P_2 starts at the same state as P. P_2 also uses the same accepting states as P.

(ii) P_2 contains a duplicate state q' for each state $q \in Q$.

(iii) Whenever there is an incoming transition $p \to q$ with a reading $x = 0$, P_2 instead transition to q'. q' then only transitions to q if the next symbol is a 0. The transition from p to q' pushes to or pops from the stack as it would if transitioning directly from p to q in the original PDA. However, the transition from q' to q does not push to or pop from the stack, thus ensuring the stack remains consistent with that of the original PDA on the original string (single 0 transition).

(iv) All other transitions are handled the same way as in the original PDA.

Problem 5.

For a language L, define $\textsc{unique}(L) := \{x \in L : \nexists y \in L \text{ such that } |y| = |x|\}$.
In other words, $\textsc{unique}(L)$ is the set of all strings $x \in L$ such that x is the only string in L that has length $|x|$.

Is the class of regular languages closed under the operation UNIQUE? Prove your answer.

Yes, regular languages are closed under UNIQUE.

For an arbitrary language L, let

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

be a DFA that recognizes L, and let $L_2 = \textsc{unique}(L)$. Construct a new DFA as follows:

$$M_2 = (Q_2, \Sigma, \delta_2, (q_0, \varnothing), F_2) \quad \text{where}$$
$$Q_2 = \{(q, Q_0) : q \in Q, Q_0 \in \mathcal{P}(Q)\}$$
$$\delta_2((q, Q_0), x) = (\delta(q, x), Q_1) \text{ where } Q_1 = \{\delta(q, \sigma_1) : \sigma_1 \in \Sigma \setminus \{x\}\} \cup \left\{\delta(q', \sigma_2) : q' \in Q_0, \sigma_2 \in \Sigma\right\}$$
$$F_2 = \{(q_f, Q_f) : q_f \in F, Q_f \cap F = \varnothing\}$$

Main Idea:
In M_2, we keep track of the computational path of M (the original DFA) on the input string x by stepping through the transitions that M would make on x. In addition, we also track all other alternative computational paths of the same length as the current one. Each state in M_2 is a pair (q, Q_0) where $q \in Q$ is the state M would be in if it were following the current computational path, and Q_0 is the set of states M would be in if following computational paths on other strings different from the current one.

This is how we manage transitions:

- Initially, there is only a single computational path of length 0 so the start state is $(q_0, \varnothing)$.
- When we are at a state (q_i, Q_i) and we read a symbol x, we do the following:
  - Generate computational paths of M that branch from the current one — that is, $Q_j = \{\delta(q_i, \sigma_1) : \sigma_1 \in \Sigma \setminus \{x\}\}$
  - Extend existing alternative computational paths of M of the given length.
    That is, we generate the set $Q_k = \{\delta(q_k, \sigma_2) : q_k \in Q_i, \sigma_2 \in \Sigma\}$.
  - Finally, we extend the computational path of M (original DFA) on the current string to end at $\delta(q_i, x)$.

  We therefore transition to $(\delta(q_i, x), Q_j \cup Q_k)$.

Finally, if a string s is in L then the computational path of M on s ends at a state $q_f \in F$. The corresponding state in M_2 is (q_f, Q_f) where Q_f is the set of all other computational paths of the same length as the computational path that takes M from q_0 to q_f on x.

In this case, a string s is in $\textsc{unique}(L)$ if and only if none of the alternative computational paths of the same length end at an accepting state, i.e. $Q_f \cap F = \varnothing$.

We therefore define our accepting states to be $F_2 = \{(q_f, Q_f) : q_f \in F, Q_f \cap F = \varnothing\}$.

Problem 6.

Every language falls into into one of the following three categories:

(i) regular

(ii) context-free but not regular

(iii) not context-free

Which of these categories is the following language in?

$$L_6 = \left\{x \in \{0,1\}^* : \exists y, z \in \{0,1\}^* \text{ such that } x = yz, |y| = |z|, \text{ and } \beta(y) \equiv 0 \pmod 3\right\}.$$

Reminder: For a string $x \in \{0,1\}^*$, $\beta(x)$ is the string x interpreted as a binary number. For instance, $\beta(11001) = 25$ and $\beta(0011) = 3$. We also define $\beta(\varepsilon) = 0$.

L_6 is context-free *but not regular*.

It is not regular because the requirement to track length of the string and check divisibility by 3 of the first half of the string makes it impossible to reuse states in any DFA recognizing L_6. Every string is in its own equivalence class, which is uniquely determined not just by the length of the string but also by the contents of the string. Since L_6 is an infinite language, any DFA would need infinite states to recognize L_6, yet a DFA must have a finite number of states.

However, with the use of a stack, we can easily track the length of strings by pushing to the stack while checking divisibility by 3, and afterward pop from the stack while processing the second part of the string.

We can construct a PDA to recognize L_6 as follows:

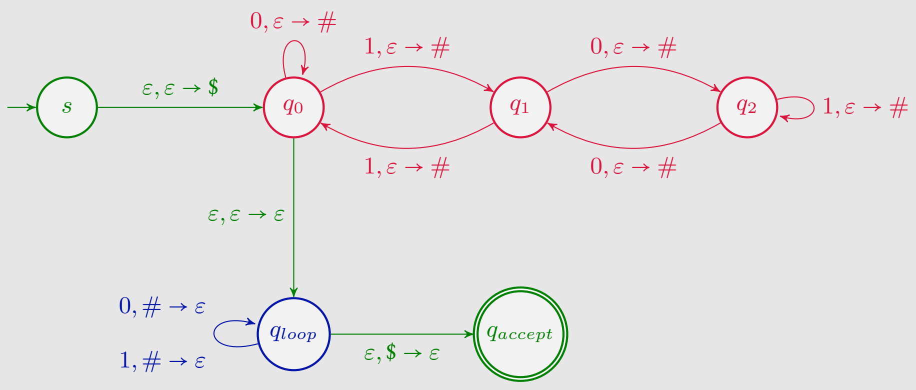

FIGURE 3. PDA for L_6

Main Idea:

By definition, all strings $s \in L_6$ can be written as $s = yz$, where $y, z \in \{0,1\}^*$, $|y| = |z|$, and $\beta(y) \equiv 0 \pmod 3$.

Our PDA works in two general phases:

- Initially, we push a \$ symbol to the stack to mark the bottom of the stack. We then take an ε-transition to q_0.
- The first phase of the PDA checks if the string processed (so far) is divisible by 3. However, on every transition we also push a $\#$ symbol to the stack to track the number of symbols processed so far.
- At any time if we are at q_0 (indicating that the string processed so far is divisible by 3), we can take an ε-transition to q_{loop} without modifying the stack.
- While in q_{loop}, we read symbols from the alphabet and pop a single $\#$ symbol from the stack to loop back to q_{loop}. This process repeats until we reach the bottom of the stack, then we can pop the bottom-of-stack marker (\$) and transition to q_{accept}.
- We claim that the PDA accepts L_6.
  - To reach q_{accept} from q_{start}, we must make the transition from q_0 to q_{loop}. This means some prefix $x_1x_2 \dots x_n$ of the input string has a computational path that ends in q_0, meaning $\beta(x_1x_2 \dots x_n)$ is divisible y 3.
  - To reach q_{accept} from q_{loop}, we must make the transition from q_{loop} to q_{accept} on reading ε and popping \$ from the stack. This means that we have to pop all the $\#$'s from the stack at q_{loop}. Since the number of $\#$'s on the stack is equal to the length of $x_1x_2 \dots x_n$, we must process some string $y_1, y_2, \dots, y_n$ such that $|y_1y_2 \dots y_n| = |x_1x_2 \dots x_n|$
  - Furthermore, q_{accept} has no outgoing or looping transitions. For a computational path to end in q_{accept}, the last transition must be from q_{loop} to q_{accept}, implying that $s = x_1x_2 \dots x_n y_1 y_2 \dots y_n$. This means $s = yz$ for some $y, z \in \{0,1\}^*$, $|y| = |z|$, and $\beta(y) \equiv 0 \pmod 3$, so $s \in L_6$.