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

PSET 4 — 02/06/2023

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

Credit Statement

I discussed ideas for this homework assignment with Paul Shin.

I also referred to the following books:

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

(b) **A Mathematical Introduction to Logic** by **Herbert Enderton**.

Problem 1.

Draw a PDA that recognizes the language $L = \left\{x \in \{0,1\}^* : N_1(x) \geq 2N_0(x)\right\}$. Give a high-level proof that your that your PDA works correctly.

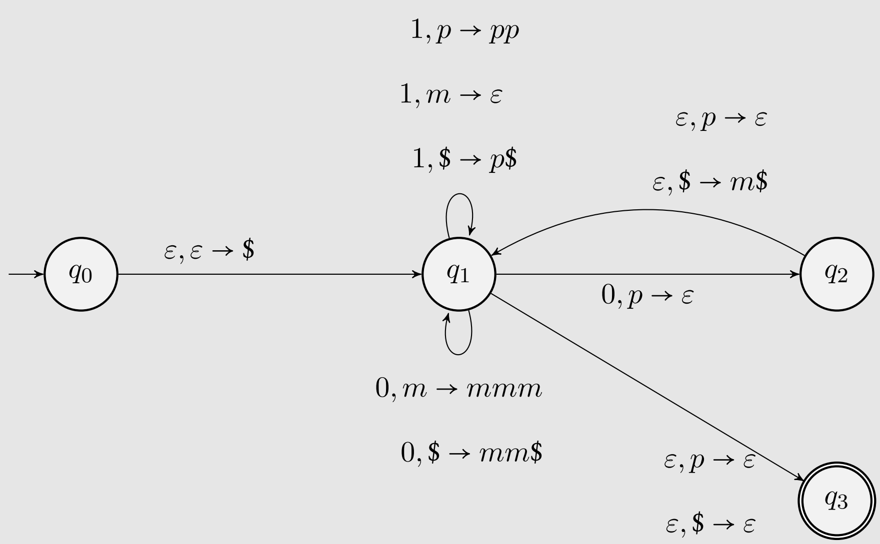

FIGURE 1. A PDA recognizing L

High-Level Idea and Proof of Correctness

We use have stack variables: p, m, and \$. Using these, we track the value of $N_1(x) - 2N_0(x)$ as we read the string. A p corresponds to a '+1', an m corresponds to a '-1', and \$ corresponds to a 0. This is how the PDA works:

- First, we enforce that no stack state can contain both p's and m's at the same time. We do this by only starting to push p's (or m's in the alternate case) if the symbol at the top of the stack is \$, symbolizing a 0.
- We start by pushing a \$ onto the stack, signifying a state of 0.
- Whenever we read a 0, we decrease the stack state by 2. This takes three forms:
  - We can remove two p's from the stack.
  - If we only have a single p at the top of the tack, we remove it and push a single m.
  - If we have an m or the zero marker (\$) at the top of the stack, we return it and push two more m's.
- Whenever we read a 1, we increase the stack state by 1. We do this by:
  - If we have a p or a \$ at the top of the stack, return it and push another p.
  - If we have an m at the top of the stack, remove it.
- Consequently, when we reach the end of the string:
  - If we have a \$ at the top of the stack, that means we have encountered *exactly* twice as many 1's as 0's, so we accept the string.
  - If we have a p at the top of the stack, that means the number of 0's we have encountered is more than twice the number of 1's we have encountered, so we accept the string.
  - Otherwise, the number of 1's was less than twice the number 0's in the string, so we do not generate a transition to the accepting state.

Problem 2.

In class, we wrote a formal construction of a PDA that proves that context-free languages are closed under union. Give similar constructions for PDAs to prove closure under:

(a) concatenation.

Let L_1 and L_2 be context-free languages. Take $M_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_{01}, F_1)$ to be a PDA that recognizes L_1 and $M_2 = (Q_2, \Sigma_2, \Gamma_2, \delta_2, q_{02}, F_2)$ to be a PDA that recognizes L_2.

Construct a new PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ as follows:

- $Q = Q_1 \cup Q_2$ (after enforcing that $Q_1 \cap Q_2 = \varnothing$)
- $\Sigma = \Sigma_1 \cup \Sigma_2$ (we don't particularly care about equality of Σ_1 and Σ_2)
- $\Gamma = \Gamma_1 \cup \Gamma_2$ (we don't particularly care about equality of Γ_1 and Γ_2)
- $q_0 = q_{01}$
- $F = F_2$
- δ is defined as follows:

$$\delta(q, a, \gamma) = \begin{cases} \delta_1(q, a, \gamma) \cup \{q_{02}, \varnothing\} & \text{if } q \in F_1 \text{ and } a = \gamma = \varepsilon. \\ \delta_1(q, a, \gamma) & \text{if } q \in Q_1. \\ \delta_2(q, a, \gamma) & \text{if } q \in Q_2. \end{cases}$$

Claim 2.1. *M recognizes $L_1 \cup L_2$.*

Proof. Note that the starting state of M is q_{01}, while the accepting states of M are in F_2. The only transition that takes M from a state formerly in Q_1 to a state formerly in Q_2 is when (1) we are at a state $q \in F_1$ (an accepting state of M_1), (2) we read no input (epsilon transition), and (3) we clear the stack.

Since M_1 recognizes L_1, we know that the computational path of M_1 on all strings in L_1 ends in an accepting state $q_{f1} \in F_1$. Likewise, since M_2 recognizes L_2, we know that the computational path of M_2 on all strings in L_2 ends in an accepting state $q_{f2} \in F_2$.

- **Completeness:** If a string s is in $L_1 L_2$, then we can write it as $s = xy$ for some $x \in L_1$ and $y \in L_2$. Then, the computational path of M on x mimics that of M_1 (since it starts at q_{01} and we use δ_1 for all states $q \in Q_1$). Therefore, M has a computational path on x that ends in an accepting state $q_{f1} \in F_1$. Then, M takes the epsilon transition to q_{02}. In processing y, M starts at q_{02} and uses δ_2 for all states $q \in Q_2$, so it has some computational path from q_{02} to $q_{f2} \in F_2$. Putting these two paths together and the middle epsilon transition, we get a computational path of M from q_{01} to $q_{f2} \in F_2$, which is an accepting state of M. Therefore, M accepts the string.

- **Soundness:** Let s be a string accepted by M. Then there must exist some computational path p_1 of M on s, taking M from q_{01} to a state in F_1, followed by the epsilon transition to q_{02}, and some computational path p_2 of M from q_{02} to a state in F_2. By definition of M, p_1 corresponds to an accepting computational path of M_1 and p_2 corresponds to an accepting computational path of M_2, Meaning that M_1 accepts some prefix x of s and M_2 accepts some suffix y of s, and x and y form the entire string s, so $s = xy, x \in L_1, y \in L_2$. Therefore, any such s accepted by M is in $L_1 L_2$.

□

(b) Kleene star.

Let L be a context-free language. Take $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ to be a PDA that recognizes L.

Construct a new PDA $M^2 = (Q \cup q_{start}, \Sigma, \Gamma, \Delta, q_0, F \cup Q_{start})$ where:

- $q_{start} \notin Q$
- δ is defined as follows:

$$\delta(q, a, \gamma) = \begin{cases} \delta(q, a, \gamma) \cup \{(q_{start}, \varnothing)\} & \text{if } q \in F \text{ and } a = \gamma = \varepsilon. \\ \{(q_0, \varnothing)\} & \text{if } q = q_{start} \text{ and } a = \gamma = \varepsilon. \\ \delta(q, a, \gamma) & \text{otherwise.} \end{cases}$$

Claim 2.2. *M_2 recognizes L^*.*

Proof. Note that:

Completeness: If a string s is in L^*, then, either:

(i) $s = \varepsilon$. Since q_{start} is an accepting state of M_2, M_2 accepts s.

(ii) $s = x_1 x_2 \cdots x_n$, with all $x_i \in L$. Then, there exists some computational path that takes M from q_0 to some state $q_i \in F$ for each x_i.

Then each x_i takes M from q_0 to some state $q_i \in F$, since q_0 recognizes L. $x_i \in L$. Then, the computational path of M_2 on s mimics that of M on x_1, then M on x_2, then M on x_3, and so on. □

Problem 3.

Give an alternate proof, using CFGs alone (no PDAs), to prove that context-free grammars are closed under:

(a) union.

Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and $G_2 = (V_2, \Sigma_2, R_2, S_2)$ such that G_1 and G_2 generate L_1 and L_2, respectively. Define $G = (V, \Sigma, R, S)$ as follows:

- $V = V_1 \cup V_2 \cup \{S\}$, where $S \notin V_1 \cup V_2$ and $V_1 \cap V_2 = \varnothing$.
- $\Sigma = \Sigma_1 \cup \Sigma_2$
- $R = R_1 \cup R_2 \cup \{(S, S_1), (S, S_2)\}$

(b) concatenation.

Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and $G_2 = (V_2, \Sigma_2, R_2, S_2)$ such that G_1 and G_2 generate L_1 and L_2, respectively. Define $G = (V, \Sigma, R, S)$ as follows:

- $V = V_1 \cup V_2 \cup \{S\}$, where $S \notin V_1 \cup V_2$ and $V_1 \cap V_2 = \varnothing$.
- $\Sigma = \Sigma_1 \cup \Sigma_2$
- $R = R_1 \cup R_2 \cup \{(S, S_1 S_2)\}$

(c) Kleene star.

Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$ be a CFG that generates L. Define $G = (V, \Sigma, R, S)$ as follows:

- $V = V_1 \cup \{S\}$, where $S \notin V_1$.
- $\Sigma = \Sigma_1$
- $R = R_1 \cup \{(S, S_1 S), (S, S S_1)\}$

Problem 4.

A string $x \in \Sigma^*$ is called a *square* if $x = w^2$ for some $w \in \Sigma^*$. Let $L_{sq} = \{w^2 : w \in \{0,1\}^*\}$. Consider its complement:

$$\overline{L}_{sq} = \left\{x \in \{0,1\}^* : x \text{ is not of the form } w^2 \text{ for any } w \in \{0,1\}^* \right\}.$$

(a) Prove that every even-length string is in $\overline{L}_{sq}$ can be decomposed as $x = uv$ where the middle symbol of u differs from the middle symbol of v.

(b) Using this property, design a context-free grammar that generates $\overline{L}_{sq}$.

$$S \implies AB \mid BA \mid X$$

$$A \implies 0A0 \mid 0A1 \mid 1A0 \mid 1A1 \mid 0$$

$$B \implies 0B0 \mid 0B1 \mid 1B0 \mid 1B1 \mid 1X \qquad \implies 0X0 \mid 0X1 \mid 1X0 \mid 1X1 \mid 0 \mid 1$$

Problem 5.

Let Σ be an alphabet, $L \subseteq \Sigma^*$, and $\# \notin \Sigma$. Define the language

$$\textsc{Intersperse}(\#, L) := \{a_1\#a_2\# \dots \#a_n\}, \text{ each } a_i \in \Sigma \text{ and } a_1a_2 \dots a_n \in L.$$

Let $M_1 = (Q, \Sigma, \Gamma, \delta, q_0, F)$ be a PDA that recognizes L. Formally describe a PDA that recognizes $\textsc{Intersperse}(\#, L)$. Also give a high-level proof that your PDA works correctly.

Let $M_2 = (Q \times \{1, \#, \varepsilon\}, \Sigma \cup \{\#\}, \Gamma, \delta_2, (q_0, 1), F_2)$ be a PDA such that:

- $F_2 = \{(q, \varepsilon) : q \in F\} \cup \{(q, \#) : q \in F\}$
- δ_2 is defined as follows:

$$\delta_2((q,x),a,\gamma) = \begin{cases} \{(q,1)\} & \text{if } x = \varepsilon, \text{ and } a = \gamma = \varepsilon. \\ \{(q,1)\} & \text{if } x = \#, a = \#, \text{ and } \gamma = \varepsilon. \\ \{\delta(q,a,\gamma), \#\} & \text{if } a \in \Sigma. \\ \{\delta(q,a,\gamma), \varepsilon\} & \text{if } a = \varepsilon. \end{cases}$$

Claim 5.1. *M_2 recognizes $\textsc{Intersperse}(\#, L)$.*

Proof. M_2 is a modification of M where at every non-epsilon transition, we require that a $\#$ symbol be read before proceeding to read the next symbol. □

Problem 6.

Consider the following CFG:

$$S \to 1S00 \mid 00S1 \mid SS \mid 0S1S0 \mid \varepsilon$$

(a) Give a simple description of the language it generates using set-builder notation.

$$L = \left\{x \in \{0,1\}^* : N_1(x) = 2N_0(x)\right\}$$

(b) Now for the hard and fun part: prove the correctness of your answer.