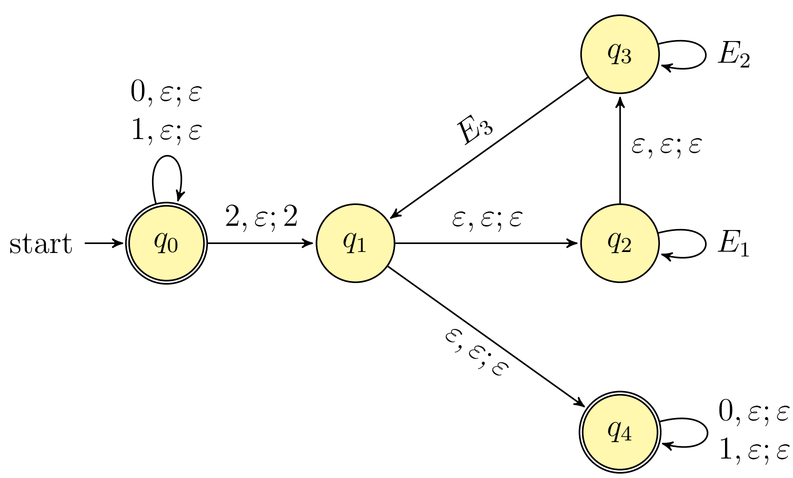
HW4 : Pushdown Automata and Context-free grammars
Sample Solutions

CSE105Sp23

Assigned questions

1. **A PDA with multiple possibilities** (22 points):
   Consider the PDA with input and stack alphabet $\Gamma = \{0, 1, 2\}$ whose "unfinished" state diagram is given below:



   There are three labels (E_1, E_2, and E_3) on the edges that are unspecified. To be precise, each E_i is of the form "$x, y; z$" where $x, y, z \in \Gamma_\varepsilon$ (recall $\Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$).

   (a) (*Graded for correctness*) [1] Prove that (no matter how the labels E_1, E_2, E_3 are specified), the language recognized by this PDA is infinite. A complete solution will include a precise description of an infinite collection of strings each of which is accepted by the PDA, with a precise and clear description of the accepting computation of the PDA on each of these strings.

[1]This means your solution will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

Solution: Consider the infinite collection of strings over Γ with at most one 2, or more formally, the language described by the regular expression: $(0 \cup 1)^* \cup (0 \cup 1)^* 2 (0 \cup 1)^*$. We will trace the computation of the PDA on each string in this set. From the start state, q_0, reading any number of 0's or 1's means the computation stays in this accept state. Reading a 2 causes a transition to q_1 (and also causes a 2 to be pushed to the stack), and from here the machine can spontaneously transition to the accepting state q_4 (without reading any input character or checking/removing/adding anything to the stack). Once in q_4, reading any number of 0s and 1s will have the computation continue to stay in q_4. This means that there is a computation on any string with at most one 2 which visits only (at most) states q_0, q_1, q_4 and reads the whole string and ends in an accept state, no matter how the labels E_1, E_2, E_3 are specified. Thus, all strings in this infinite language are accepted by the PDA. Since the language recognized by the PDA is a superset of this language, the language recognized by the PDA must be infinite too.

(b) (*Graded for completeness*) [2] Prove/Disprove: Over all the possible choices for the labels E_1, E_2, E_3, this PDA can only recognize finitely many languages. Justify your solution by referring back to the relevant definitions.

Solution: There are only finitely many possibilities for each label. Each label is in the set $\Gamma_\varepsilon \times \Gamma_\varepsilon \times \Gamma_\varepsilon$, for a total of $|\Gamma_\varepsilon|^3$ possible labels. Since there are 3 unspecified labels and $|\Gamma_\varepsilon| = 4$, we get that there are

$$(|\Gamma_\varepsilon|^3)^3 = |\Gamma_\varepsilon|^9 = 4^9 = 262144$$

total possibilities, which is finite!

(c) (*Graded for correctness*) Recall that for $L \subseteq \Sigma^*$ with $\Sigma = \{0, 1\}$, we define

$$\begin{aligned} \textsc{Rep}(L) &:= \{w \in \Gamma^* \mid \text{between every pair of successive 2's in } w \text{ is a string in } L\} \\ &= \{w \in \Gamma^* \mid \text{for all } v \in \Sigma^* \text{ if } 2v2 \in \textsc{Substring}(\{w\}), \text{ then } v \in L\} \end{aligned}$$

where for all languages $K \subseteq \Gamma^*$ we let

$$\textsc{Substring}(K) := \{w \in \Gamma^* \mid \text{there exist } a, b \in \Gamma^* \text{ such that } awb \in K\}.$$

Determine how to set the labels E_1, E_2, E_3 so that the language of the PDA is

$$\textsc{Rep}(\{0^n 1^m \mid n \geq 0, m \geq 0\})$$

In addition to specifying each E_i, a complete justification will include a precise description of why this choice of E_i means that the PDA recognizes the language indicated.

[2]This means you will get full credit so long as your submission demonstrates honest effort to answer the question. You will not be penalized for incorrect answers. To demonstrate your honest effort in answering the question, we ask that you include your attempt to answer *each* part of the question. If you get stuck with your attempt, you can still demonstrate your effort by explaining where you got stuck and what you did to try to get unstuck.

Solution: $E_1 = 0, \varepsilon; \varepsilon$, $E_2 = 1, \varepsilon; \varepsilon$, $E_3 = 2, 2; 2$.

Informally, the transitions $q_1 \to q_2 \to q_3 \to q_1$ in this PDA represent the enforcement of the rule that the string between a pair of 2s consists of any number of 0s followed by any number of 1s.

More formally, the first occurrence of a 2 causes the transition from the start state to q_1, and if no more 2's are read, the machine can transition to q_4. Alternatively, there will be at least one pair of 2's. We set E_3, the only other transition into q_1 from q_3, to be $2, 2; 2$, so that the state q_1 will more generally represent the state where the last symbol read was a 2. Now the transitions from q_1 to q_3 via q_2 must reflect any number of 0's followed by any number of 1's. Therefore, after transitioning from q_1 to q_2 on the empty string, q_2 reflects the any number of 0's portion. We set E_1 to be $0, \varepsilon; \varepsilon$ so that it stays in this state if a 0 is read and does not check or alter the stack. Similarly, q_3 reflects any number of 1's, and we set E_2 to be $1, \varepsilon; \varepsilon$ so that it stays in E_2 if a 1 is read. As the number of 0's and 1's between each pair of 2's does not have to be related, we don't need to use the stack to keep track of how many of each we have seen.

(d) (*Graded for correctness*) Determine how to set the labels E_1, E_2, E_3 so that the language of the PDA is

$$\text{Rep}(\{0^n 1^n \mid n \geq 0\})$$

In addition to specifying each E_i, a complete justification will include a precise description of why this choice of E_i means that the PDA recognizes the language indicated.

Solution: $E_1 = 0, \varepsilon; 0$, $E_2 = 1, 0; \varepsilon$, $E_3 = 2, 2; 2$. We now need to justify why the language recognized by the resulting PDA is $\text{Rep}(\{0^n 1^n \mid n \geq 0\})$.

Informally, we use the transition from $q1 \to q2 \to q3 \to q1$ to enforce the string pattern $0^n 1^n$ (for some n) between two successive 2's.

More formally: if there are no 2's in the string the machine stays in q_0, accepting the string. On the first 2, it transitions from q_0 to q_1, if there are no more 2's in the string, it could make a spontaneous transition to the accepting state q_4. Alternatively, there could be at least one pair of 2's, this can be handled by the loop $q1 \to q2 \to q3 \to q1$. We have set E_3 to $2, 2; 2$ which will allow transition $q3 \to q1$, only when another 2 is encountered in the string. The state q_1 will represents the state where the last symbol read was a 2. The transition E_1 is set to $0, \epsilon; 0$, this would store the 0's in the stack as they are encountered after the 2. E_2 is set to $1, 0; \epsilon$ which enforces that the 0's are followed by equal number of 1's, by popping a 0 from stack for every 1. The 0's pushed on the stack in state q_2 are used to check the number of following 1's in state q_3 by pop operation on the stack.

2. **Grammar practice** (12 points):
For each of the languages listed below, define a context-free grammar $G = (V, \Sigma, R, S)$

that generates the language. Instead of formally justifying your grammar, illustrate it by giving **two examples** of strings in the language and their derivations using your grammar and **one example** of a string not in the language with an explanation of why it cannot appear on the right side of any derivation in your grammar. Choose your examples so they are different enough to illustrate the role of as many of the variables in your grammar as possible.

(a) (*Graded for correctness*) $\textsc{Rep}(L)$ where $L = \{0^n 1^n \mid n \geq 0\}$

Solution: $G = (\{S, M, A, L\}, \{0, 1, 2\}, R, S)$ with rules

$$\begin{aligned} S &\to A2M \mid A \\ M &\to L2M \mid A \\ L &\to 0L1 \mid \varepsilon \\ A &\to 0A \mid 1A \mid \varepsilon \end{aligned}$$

Example 1 : 12000111210
S $\Rightarrow$ A2M $\Rightarrow$ 1A2M $\Rightarrow$ 12M $\Rightarrow$ 12L2M $\Rightarrow$ 120L12M
$\Rightarrow$ 1200L112M $\Rightarrow$ 12000L1112M $\Rightarrow$ 120001112M
$\Rightarrow$ 120001112A $\Rightarrow$ 1200011121A $\Rightarrow$ 12000111210A
$\Rightarrow$ 12000111210

Example 2 : 20122012
S $\Rightarrow$ A2M $\Rightarrow$ 2M $\Rightarrow$ 2L2M $\Rightarrow$ 20L12M $\Rightarrow$ 2012M $\Rightarrow$ 2012L2M $\Rightarrow$ 20122M
$\Rightarrow$ 20122L2M $\Rightarrow$ 201220L12M $\Rightarrow$ 20122012M $\Rightarrow$ 20122012A
$\Rightarrow$ 20122012

Example 3 : 2010102 (Not in the language)
Let's break down this claim into two steps. Let's first consider a string in the language that has two 2's in it. If we immediately use the production rule $S \to A$ in our derivation, we don't have any 2's, so we must use the rule $S \to A2M$. Once again, A cannot lead to any 2's in the string, so we must use the variable M. If we use the production rule to go to A, then we still cannot produce any 2's. This leaves only one remaining choice, we must use the production rule $M \to L2M$. Which means any derivation of a string that has at least two 2's can start as $S \implies A2L2M$. Now, we *have* to use the production rule $M \to A$ since the other production rule for M produces a 2. In other words, we must have $S \implies A2L2A$, and the string between the two 2's is a language generated by the grammar where the starting variable is L. Any derivation from variable L produces a string of form $0^m 1^m$, for some $m \geq 0$. Therefore, string 01010 cannot be derived from L. Thus, we cannot derive 2010102 from starting variable S.

(b) (*Graded for correctness*) $\{1^n = 1^a + 1^b \in \{1, =, +\}^* \mid a, b, n \geq 1 \text{ such that } a + b = n\}$

Solution: $G = (\{S, M\}, \{1, +, =\}, R, S)$ with rules

$$S \to 1S1 \mid 1M + 1$$
$$M \to 1M1 \mid 1 = 1$$

Example 1 : $11 = 1 + 1$
S ⇒ 1M+1 ⇒ 11=1+1
Example 2 : $11111 = 11 + 111$
S ⇒ 1S1 ⇒ 11S11 ⇒ 111M+111 ⇒ 1111M1+111 ⇒ 11111=11+111
Example 3 : $111 = 1 + 1$ (Not in the language)
Notice that every production rule adds exactly two 1's to the string being derived. Therefore, all strings in the language generated by the grammar G have even number of 1's. Since $111 = 1 + 1$ has an odd number of 1's, it cannot be derived in the grammar G.

3. **Substrings and regularity** (16 points):
For this problem, we fix the alphabet $\Gamma = \{0, 1, 2\}$. Recall the definition of the function Substring from Problem 1.

(a) (*Graded for correctness*) Prove that $\textsc{Substring}(\{0^n1^n \mid n \geq 0\})$ is regular. A complete solution will include a precise definition of a DFA, NFA, or regular expression that recognizes or describes it, along with a brief justification of your construction by explaining the role each state plays in the machine or referring back to relevant definitions.

Solution: To prove that the language is regular we will show that it can be described by a regular expression. Informally, this regular expression will need to describe the pattern of "some number of 0's followed by some number of 1's" because any substring of a string of the form 0^n1^n must have any of its 0's at front and any of its 1's at the back, and the numbers of 0's and 1's aren't dependent on one another.
Formally, consider the regular expression 0^*1^*. To show that it describes

$$\textsc{Substring}(\{0^n1^n \mid n \geq 0\})$$

we need to show

$$\{0^i1^j \mid i, j \geq 0\}$$
$$= \{w \in \{0,1\}^* \mid \text{there exist } a, b \in \{0,1\}^* \text{ such that } awb \in \{0^n1^n \mid n \geq 0\}\}.$$

Consider an arbitrary string in the left-hand-set, 0^i1^j (with $i, j \geq 0$). Let

$$a = 0^{\max(i,j)-i} \qquad b = 1^{\max(i,j)-j}$$

Calculating:

$$a\ 0^i1^j\ b = 0^{\max(i,j)-i}0^i1^j1^{\max(i,j)-j} = 0^{\max(i,j)}1^{\max(i,j)}$$

Since $\max(i,j)$ is a nonnegative integer, this string is in $\{0^n1^n \mid n \geq 0\}$ so a and b are the necessary witnesses for $0^i1^j \in \textsc{Substring}(\{0^n1^n \mid n \geq 0\})$.

Conversely, consider an arbitrary string w in $\textsc{Substring}(\{0^n1^n \mid n \geq 0\})$. By definition, there are strings a, b with $awb = 0^n1^n$ for some $n \geq 0$. By definition of string equality, w must have any of its 0's before all (if any) of its 1's. In other words, $w \in \{0^i1^j \mid i, j \geq 0\}$ as required.

(b) (*Graded for correctness*) Prove that $\textsc{Substring}(\{0^n1^n2^n \mid n \geq 0\})$ is not regular.

Solution: Let A denote the language $\{0^n1^n2^n \mid n \geq 0\}$.

Suppose, towards a contradiction that $\textsc{Substring}(A)$ wer regular. By the pumping lemma, let $p \geq 1$ be a pumping length of this language.

Consider the string $w = 0^p1^p2^p, |w| = 3p > p$. Clearly, $w \in \textsc{Substring}(A)$: let $a = b = \epsilon$, then $awb = 0^p1^p2^p \in \{0^n1^n2^n \mid n \geq 0\}$.

To arrive at a contradiction, we consider all decompositions of w into three substrings x, y, z, such that $w = xyz, |y| > 0, |xy| \leq p$ and show that, no matter what, we can pick $i \geq 0$ where $xy^iz \notin \textsc{Substring}(A)$

Since w starts with p zeros, we can describe all such decompositions x, y, z as:

$x = 0^{k_1}$
$y = 0^{k_2}$
$z = 0^{k_3}1^p2^p$
with $k_1 \geq 0, k_2 > 0, k_1 + k_2 \leq p, k_1 + k_2 + k_3 = p$

Choose $i = 2$. The string

$$xy^iz = 0^{k_1+ik_2+k_3}1^p2^p = 0^{p+k_2}1^p2^p$$

Since $k_2 > 0$, $p + k_2 > p$ and we will see that $0^{p+k_2}1^p2^p$ string cannot be in $\textsc{Substring}(A)$: Suppose that it were. Then, there are strings $a, b \in \Gamma^*$ such that $a \circ 0^{p+k_2}1^p2^p \circ b \in A$. However, since $p + k_2 > p$, there are already fewer 1's in the string than 0's. Since strings in A have all their characters in order: first 0's, then 1's, then 2's, a must be only 0s and b must be only 2s. Adding more 0's to the front of $0^{p+k_2}1^p2^p$ or adding more 2's to its back cannot remedy the fact that there are fewer 1's in the string than 0's, a contradiction with the assumption that there are strings a, b that witness $0^{p+k_2}1^p2^p \in \textsc{Substring}(A)$.

Since there exists some i for which the pumped string is not in $\text{SUBSTRING}(A)$, we conclude that $\text{SUBSTRING}(A)$ is not regular.

(c) (*Graded for completeness*) Is $\text{SUBSTRING}(\{0^n1^n2^n \mid n \geq 0\})$ context-free? Informally justify your answer, referring to class discussions and/or the textbook.

Solution: No. $\text{SUBSTRING}(\{0^n1^n2^n \mid n \geq 0\})$ is the collection of all and only strings of the form $0^a1^c2^b, a \leq c, b \leq c$ for c any nonnegative integer. For a PDA to recognize this set of strings, it would need to use the stack simultaneously to ensure $a \leq c$ and $b \leq c$, which it cannot do.

More formal (not required in this class because it uses the pumping lemma for CFL): We apply the pumping lemma for context-free languages. Let p be a nonnegative integer and we will prove that it is not a pumping length for this language. Choose the string $s = 0^p1^p2^p$. We consider all decompositions $s = uvxyz$ where $|uv| > 0$ and $|vxy| \leq p$. There are two cases to consider:

i. Either v or y contains 1's. Notice that since $|vxy| \leq p$, this implies that vxy can either have 0's or 2's in it, but importantly not both. Therefore, if we "pump down" by setting $i = 0$, we get a string that has fewer 1's than either 2's or 0's, so it can't be the substring of $0^n1^n2^n$ for any n.

ii. Otherwise, neither v nor y contain a 1. Therefore, when we "pump up" by setting $i > 1$, we get a string that has more 0's or 2's than 1's, which is once again not in the language.

This completes the proof.