CS 4510 Automata and Complexity 1/29/2024

Homework 2: Context-Free Languages

Vidit D. Pokharna *Due: 2/07/2024*

You should submit a typeset or *neatly* written pdf on Gradescope. The grading TA should not have to struggle to read what you've written; if your handwriting is hard to decipher, you will be asked to typeset your future assignments. Four bonus points if you use LaTeX, and our template. You may collaborate with other students, but any written work should be your own.

1. (5 points) Give a CFG for valid regular expressions over the alphabet $\Sigma = \{a, b\}$. The alphabet of your grammar should be over $\{a, b, (,), \cup, ^*, \lambda, \emptyset\}$. You may insert more parenthesis than necessary. Here we shall use λ to represent the base-case regular expression symbolizing the set only containing the empty string, that is, $\lambda = \{\varepsilon\}$. Use the symbol ε not as part of the regular expression, but if necessary, as part of the right-hand-side of productions of the grammar to satisfy $(V \cup \Sigma)^*$. You must generate all syntactically valid regular expressions.
   $S \to (S) \mid SS \mid S^* \mid S \cup S \mid a \cup S \mid b \cup S \mid a \mid b \mid \lambda \mid \emptyset$

2. (5 points) Give a CFG for the language $\{a^i b^j \mid x, y \in \mathbb{N},\ x + y = i \text{ and } 4x + 3y = j\}$.
   $S \to aSbbb \mid aSbbbb \mid \varepsilon$

3. (10 points) Give a CFG for $\{a^{3n}b^ic^{n-i} \mid i \leq n \in \mathbb{N}\}$ Then convert it to Chomsky Normal Form. Show your work.

   $S \to aaaSc \mid T$
   $T \to aaaTb \mid \varepsilon$

   Conversion to CNF

   Step One: Add new start symbol

$$\begin{aligned} &S_0 \to S \\ &S \to aaaSc \mid T \\ &T \to aaaTb \mid \varepsilon \end{aligned}$$

   Step Two: Remove ε rules

$$\begin{aligned} &S_0 \to S \mid \varepsilon \\ &S \to aaaSc \mid T \mid aaac \\ &T \to aaaTb \mid aaab \end{aligned}$$

   Step Three: Remove unit rules

$$\begin{aligned} &S_0 \to \varepsilon \mid aaaSc \mid aaac \mid aaaTb \mid aaab \\ &S \to aaaSc \mid aaac \mid aaaTb \mid aaab \\ &T \to aaaTb \mid aaab \end{aligned}$$

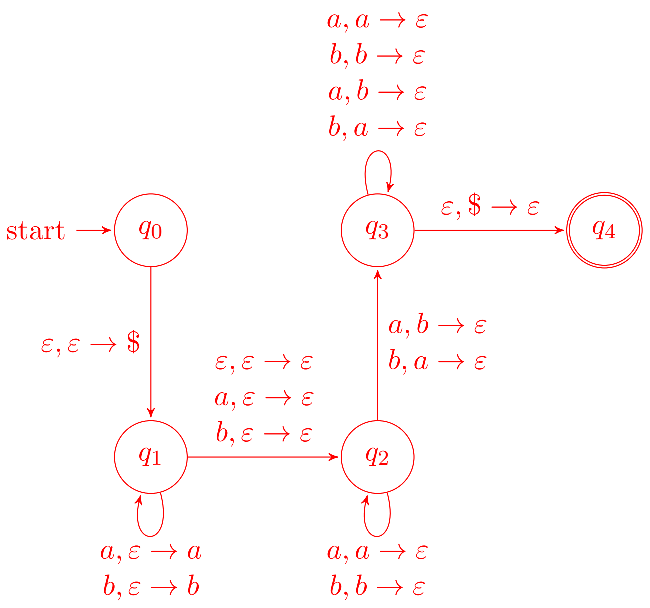
Step Four: Split rules into parts

$$
\begin{aligned}
&S_0 \to \varepsilon \mid aD \mid aF \mid aI \mid aK\\
&S \to aD \mid aF \mid aI \mid aK\\
&T \to aI \mid aK\\
&B \to Sc\\
&C \to aB\\
&D \to aC\\
&E \to ac\\
&F \to aE\\
&G \to Tb\\
&H \to aG\\
&I \to aH\\
&J \to ab\\
&K \to aJ
\end{aligned}
$$

Step Five: Replace terminals with variables

$$
\begin{aligned}
&S_0 \to \varepsilon \mid LD \mid LF \mid LI \mid LK\\
&S \to LD \mid LF \mid LI \mid LK\\
&T \to LI \mid LK\\
&B \to SM\\
&C \to LB\\
&D \to LC\\
&E \to LM\\
&F \to LE\\
&G \to TN\\
&H \to LG\\
&I \to LH\\
&J \to LN\\
&K \to LJ\\
&L \to a\\
&M \to c\\
&N \to b
\end{aligned}
$$

4. (5 points) Give a PDA for the language $\{w \in \Sigma^* \mid w \text{ is not a palindrome}\}$



5. (5 points) Prove that the intersection of a context-free language and a regular language is context-free. (Hint, Lecture 1)

Let L_1 be a context-free language accepted by a pushdown automaton (PDA) $P = (Q_P, \Sigma, \Gamma, \delta_P, q_{0P}, F_P)$, and let L_2 be a regular language accepted by a finite automaton (FA) $F = (Q_F, \Sigma, \delta_F, q_{0F}, F_F)$. We aim to show that their intersection, $L_1 \cap L_2$, is also a context-free language.

To prove this, we construct a new PDA, $P' = (Q', \Sigma, \Gamma', \delta', q_0', F')$, that recognizes $L_1 \cap L_2$ as follows:

- **State Set** Q': Each state in P' is a pair consisting of a state from P and a state from F, i.e., $Q' = Q_P \times Q_F$.
- **Stack Alphabet** Γ': The stack alphabet remains the same as in P, $\Gamma' = \Gamma$.
- **Start State** q_0': The start state is a pair of the start states of P and F, $q_0' = (q_{0P}, q_{0F})$.
- **Accept States** F': A state in P' is accepting if both components are accepting in their respective automata, $F' = \{(q_P, q_F) \mid q_P \in F_P \text{ and } q_F \in F_F\}$.
- **Transition Function** δ': For any $a \in \Sigma \cup \{\varepsilon\}$ and stack symbol $X \in \Gamma \cup \{\varepsilon\}$, the transition function δ' of P' is defined to simultaneously simulate the transitions of P and F on the same input:

$$\begin{aligned} \delta'((q_P, q_F), a, X) = \{((q_P', q_F'), \alpha) \mid (q_P', \alpha) \in \delta_P(q_P, a, X) \text{ and} \\ q_F' \in \delta_F(q_F, a), \text{ for some } \alpha \in (\Gamma \cup \{\varepsilon\})^* \end{aligned}$$

This construction ensures that P' accepts an input string if and only if it is accepted by both P (and hence in L_1) and F (and hence in L_2), while accurately simulating the stack operations as dictated by P's transitions. Therefore, the language recognized by P', $L(P')$, is exactly $L_1 \cap L_2$, proving that the intersection of a context-free language and a regular language is context-free.

6. (5 points) We defined a regular grammar to only have rules of the form $V \to \Sigma V \mid V \mid \varepsilon$. For example $A \to bC \mid aA \mid D \mid \varepsilon$.

   Prove that if a CFG only has rules of the form $V \to \Sigma^* V \mid V \mid \varepsilon$, then it decides a regular language. Rules of this form look like $A \to bcdeF \mid abaB \mid B \mid \varepsilon$.

   Consider a CFG with production rules of the form $V \to \Sigma^* V \mid V \mid \varepsilon$. The rules of this CFG are such that the right-hand side of any production either:

   - Moves to another variable V
   - Extends the current string with a sequence of terminals followed by a variable V
   - Ends the string

   I will show that this CFG is equivalent to a regular grammar (RG) by constructing an RG with the following rules for each variable V:

   (a) For each rule $V \to a_1 a_2 \ldots a_n W$ in the CFG, where $a_1, a_2, \ldots, a_n \in \Sigma$ and W is a variable, we add a chain of rules in the RG: $V \to a_1 V_1$, $V_1 \to a_2 V_2$, $\ldots$, $V_{n-1} \to a_n W$. For $n = 0$, the rule is simply $V \to W$.

   (b) For each rule $V \to \varepsilon$ in the CFG, we add the rule $V \to \varepsilon$ in the RG.

   This construction results in an RG where the right-hand side of any production is a single terminal followed by a variable, a variable, or the empty string. Such a grammar is by definition a regular grammar, as described in the question.

   Thus, we have shown that any CFG with production rules of the form $V \to \Sigma^* V \mid V \mid \varepsilon$ can be converted into an equivalent RG, and hence decides a regular language.

7. (5 points) We previously proved that all finite languages are regular. In Syntactic Structures, Chomsky argues that english cannot have regular structure. The question is never addressed if the set of syntactically valid english sentences is finite or infinite. Why is Chomsky's argument still correct, that english does not have regular structure, even if you suppose that english was finite?

   Chomsky's argument that English does not have a regular structure holds even if the set of syntactically valid English sentences were finite, due to the inherent complexity of English grammar that extends beyond the capabilities of regular grammars and deterministic finite automata. Regular grammars, defined by regular expressions and recognized by DFAs, are limited to identifying sequences of characters as valid English words. However, English grammar encompasses recursive rules, structures, and dependencies between elements that cannot be captured by the state transitions of DFAs.

   This complexity indicates that English syntax involves context-sensitive grammar, which allows for the expression of unique grammar and structure, unlike regular grammars that we currently discuss. Therefore, Chomsky's assertion about the non-regular structure of English highlights the language's reliance on context-sensitive grammar elements to convey meaning, structure, and coherence in sentences, underscoring the limitations of regular grammars in capturing the full scope of natural language syntax. Even if we suppose the set of all possible English sentences were finite, the complexities of English syntax demonstrate that English as a structured language cannot be adequately described or recognized by regular grammars or finite automata.