

Note: This problem set is mostly a review of context free languages. You may assume everything in the slides from Lecture 3, plus the following basic fact about CFL's:

Closure Theorem for Context Free Languages: Let $A, B \subseteq \Sigma^*$ be context-free languages, let $R \subseteq \Sigma^*$ be a regular language, and let $h : \Sigma^* \rightarrow \Gamma^*$ and $g : \Gamma^* \rightarrow \Sigma^*$ be homomorphisms. Then the following languages are context-free:

- | | | |
|---------------|----------------|---------------|
| 1. $A \cup B$ | 2. AB | 3. $A \cap R$ |
| 4. $h(A)$ | 5. $g^{-1}(A)$ | |

Exercise: (not to be handed in). Prove the Closure Theorem for CFLs. [The most interesting case is number 5. A hint for this case is to consider a PDA, P , s.t. $\mathcal{L}(P) = A$. You want to build a PDA, P' s.t. $\mathcal{L}(P') = g^{-1}(A)$. Note that for any string, $w \in \Gamma^*$, $w \in \mathcal{L}(P') \leftrightarrow g(w) \in P$. Suppose $w = a_1 a_2 \dots a_n$. We can test if $w \in \mathcal{L}(P')$ by simulating P on $g(w) = g(a_1)g(a_2) \dots g(a_n)$. We can thus build P' from P by adding a front end that feeds characters of $g(a_i)$ one by one to P . Since the largest number of characters we have to remember is the length of the longest string $g(\gamma)$ for $\gamma \in \Gamma$, we can create the "front end" with finitely many states and so the combined machine, P' is still a PDA.]

Problems

1. [10 pts.] Prove that the language $WW = \{ww \mid w \in \{a, b\}^*\}$ is not a CFL. I would like you to do this by using the Closure Theorem for CFLs together with the fact proved in Lecture 3 that $P_2 = \{a^n b^m a^n b^m \mid n, m \in \mathbf{N}\}$ is not a CFL.
2. [10 pts.] Prove using the pumping lemma that $\mathcal{L}_2 = \{w \in \{a, b, c\}^* \mid \#_a(w) = \#_b(w) = \#_c(w)\}$ is not a CFL. [Hint: it suffices to show that $T = \{a^n b^n c^n \mid n \in \mathbf{N}\}$ is not a CFL. Why?]
3. [10 pts.] Show using the fact from (2) that T is not a CFL, plus the Closure Theorem for CFLs, that $\mathcal{L}_3 = \{ww^R w \mid w \in \{a, b\}^*\}$ is not a CFL. [By w^R I mean the string w reversed, e.g. $(abb)^R = bba$.]
4. [10 pts.] Build a PDA for the language $\mathcal{L}_4 = \{a^i b^j \mid i < j\}$ and argue briefly why your construction is correct.
5. [10 pts.] Show that $\mathcal{L}_5 = \{a, b, c\}^* - \mathcal{L}_2$ is a CFL.
6. [10 pts.] Is $\mathcal{L}_6 = \{a^i b^j c^k \mid i \equiv j \pmod{5} \text{ and } j = k\}$ a CFL? Prove your answer.
7. [20 pts.] Describe a very efficient algorithm for the following decision problem:

$$\text{EMPTY-CFL} = \{G \mid G \text{ a CFG, } \mathcal{L}(G) = \emptyset\}$$

Analyze the time complexity of your algorithm and give a clear explanation of why it works. There is a linear time algorithm for this problem. A hint is that you should efficiently mark all those nonterminals that can generate some string of terminals. The size of a given context-free grammar is the sum of the lengths i.e., number of symbols, of its rules.

8. Let $\Sigma = \{0, 1\}$. Recall that a 1:1 correspondence, $f : A \rightarrow B$, is a function from A to B that is 1:1 and onto. For example, the map $g(n) = 2n$ defines a 1:1 correspondence from the natural numbers to the even natural numbers.

(a) [5 pts.] Define the map $\rho : \Sigma^* \rightarrow \mathbf{N}$ as follows: $\rho(w) = ((\text{int})1w) - 1$, i.e., add a “1” to the left of w , consider the result as a binary natural number, and subtract 1. Prove that ρ is a 1:1 correspondence.

From now on, when we talk about the natural number corresponding to binary string w , we will mean $\rho(w)$ and when we talk about the binary string corresponding to the natural number n , we will mean $\rho^{-1}(n)$.

(b) [15 pts.] Georg Cantor defined the following function $P : \mathbf{N} \times \mathbf{N} \rightarrow \mathbf{N}$ as $P(i, j) = \frac{(i+j)(i+j+1)}{2} + i$. Prove that P is a 1:1 correspondence. [It may help to remember the fact – which you do not have to prove – that $\sum_{i=1}^n i = \frac{n(n+1)}{2}$.]

[Notice by the way that there are thus corresponding functions $L, R : \mathbf{N} \rightarrow \mathbf{N}$ satisfying the following properties:

- i. $\forall n \in \mathbf{N}(P(L(n), R(n))) = n$
- ii. $\forall a, b \in \mathbf{N}(L(P(a, b)) = a \wedge R(P(a, b)) = b)$
- iii. $\forall n(L(n) \leq n \wedge R(n) \leq n)$

One of the useful things about the pairing function, P , is that we can now think about all computable functions f as functions from \mathbf{N} to \mathbf{N} . If we want to consider an input n as a pair of inputs, fine: $n = P(L(n), R(n))$. Similarly, we can consider it as a triple: $n = P(P(L(L(n)), R(L(n))), R(n))$, etc. Thus a single natural number (aka binary string) can be any finite number of arguments as desired. Thus we are dealing with the ultimate untyped programming language: there is only one type, namely the natural number. More about this later.]

Cantor was the first person to develop a formal understanding of the cardinality of infinite sets. He defined that two sets A and B have the **same cardinality** ($A \sim B$) iff there exists a 1:1 correspondence from A to B . Thus, from the above we get a few weird consequences of this definition such as the fact that $\mathbf{N} \sim \text{Even}$ where Even is the set of even natural numbers. Even more weird, the pairing function tells us that $\mathbf{N} \sim \mathbf{N} \times \mathbf{N}$, i.e., there are the same number of natural numbers as pairs of natural numbers.