COMS W3261
Computer Science Theory
Homework #2

Alexander Roth

2014 – 09 – 29

Problems

1. Consider the grammar G:

$$S \to aSbS \,|\, bSaS \,|\epsilon$$

(a) Describe in English the language generated by G. Prove that G generates precisely this language.

Solution: The language generated by G returns all even strings of a's and b's such that the number of a's is equal to the number of b's.

PROOF: We shall proove that a string w which is an even length string of the same number of a's and b's (order- independent) is in $L(G)$ if and only if it is an even length string with an equal number of a's and b's.
(If) Suppose w is an even length string of equal numbers of a's and b's. We show by induction on $|w|$ that w is in $L(G)$.

BASIS: We use lengths 0 and 2 as the basis. If $|w| = 0$ or $|w| = 2$, then w is ϵ, ab, or ba. We do not use a length of 1 because w is an even length string. Since there are productions $S \to \epsilon$, $S \to aSbS$, and $S \to bSaS$, we conclude that $S \stackrel{*}{\Rightarrow} w$ in any of these basis cases.

INDUCTION: Suppose $|w| \geq 2$. Since w is an even length string of equal number of a's and b's, we know that for every a in the string, there is an matching b. That is, $w = aSbS$ or $w = bSaS$. Moreover, we know that x must be an even length string of equal numbers of a's and b's or an empty string (ϵ). Note that we need the fact that $|w| \geq 2$ to infer that there is a distinct a paired with a distinct b within w.

If $w = aSbS$, then we invoke the inductive hypothesis to claim that $G \stackrel{*}{\Rightarrow} S$. Then there is is a derivation of w from S, namely $S \Rightarrow aSbS \stackrel{*}{\Rightarrow} axbx = w$ If $w = bSaS$, the argument is the same, but

we use the production $S \to bSaS$ at the first step. In either case, we conclude that w is in $L(G)$ and complete the proof.

(Only-if) Now, we assume that w is in $L(G)$; that is, $S \stackrel{*}{\Rightarrow} w$. We must conclude that w is an even length string of equal number of a's and b's. The proof is an induction on the number of steps in a derivation of w from S.

BASIS: If the derivation is one step, then it must use one of the three productions that do not have S in the body. That is, the derivation of $S \Rightarrow \epsilon$, $S \Rightarrow ab$, or $S \Rightarrow ba$ Since ϵ, ab, and ba, are all even length strings of equal a's and b's, the basis is proven.

INDUCTION: Now, suppose that the derivation takes $n+1$ steps, where $n \geq 1$, and the statement is true for all derivations of n steps. That is, if $S \stackrel{*}{\Rightarrow} x$ in n steps, then x is an even length string of equal numbers of a's and b's.

Consider an $(n+1)$-step derivation of w, which must be of the form

$$S \Rightarrow aSbS \stackrel{*}{\Rightarrow} axbx = w$$

or $S \Rightarrow bSaS \stackrel{*}{\Rightarrow} bxax = w$, since $n+1$ steps is at least two steps, and the productions $S \to aSbS$ and $S \to bSaS$ are the only productions whose use allows additional steps of a derivation. Note that in either case, $S \stackrel{*}{\Rightarrow} x$ in n steps.

By the inductive hypothesis, we know that x is an even length string of equal number of a's and b's. But if so, then $axbx$ and $bxax$ are also even length strings with equal numbers of a's and b's. Thus, we conclude that w is an even length string of equal a's and b's, which completest the proof $\square$

(b) Is the language generated by G regular? Prove your answer.

Solution: Let us show that $L(G)$ consisting of all strings with an equal number of a's and b's is not a regular language. Suppose n is the constant that must exist if $L(G)$ is regular, according to the pumping lemma. Thus, $w = a^n b^n$, that is n a's followed by n b's. A string which is definitely in $L(G)$.

Let us break up the string w into three parts: x, y, and z. We know that $y \neq \epsilon$ and that $|xy| \leq n$, and that xy comes at the front of w. Since $|xy| \leq n$ and xy comes at the front of w, we know that x and y consist of only a's. The pumping lemma tells us that xz is in $L(G)$, if $L(G)$ is regular and if $k = 0$. However, xz has n b's because all the b's of w are in z. But xz also has fewer than n a's, because we lost the a's of y. Since $y \neq \epsilon$, we know that there can be no more than $n-1$ a's among x and z. Thus, after assuming $L(G)$ is a regular

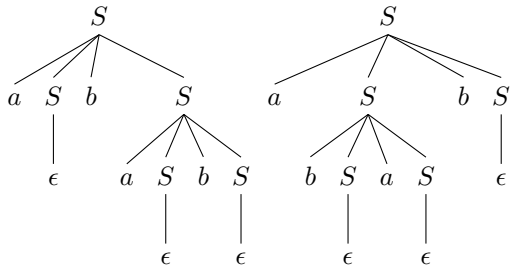
language, we have proved that xz cannot be in $L(G)$. Thus, this is a proof by contradiction that $L(G)$ cannot be a regular language.

(c) Demonstrate whether G is ambiguous or unambiguous.

Solution: Consider the sentential form $abab$ It has 2 derivations:

1. $S \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSbS \Rightarrow ababS \Rightarrow abab$
2. $S \Rightarrow aSbS \Rightarrow abSaSbS \stackrel{*}{\Rightarrow} abab$

These derivations can be represented as two distinct parse trees:



Since there are two distinct parse trees for the same yield, we regard this grammar as ambiguous.

2. Put the grammar G in question (1) into Chomsky Normal Form. Use the Cocke-Younger-Kasami algorithm to parse the sentence `abab` according to your CNF grammar. Display the CYK table constructed by the CYK algorithm and show how all parse trees for this sentence can be reconstructed from the CYK table.

Solution: Grammar G is:

$$S \to aSbS \,|\, bSaS \,|\, \epsilon$$

First, we eliminate all ϵ-productions. We need to find the nullable symbols. S is directly nullable because it has a production with ϵ as the body.

Now, let us construct the productions of grammar G_1. First, consider $S \to aSbS$. The second and fourth positions hold nullable symbols, so there are four choices of present/absent. Our four choices yield:

$$S \to aSbS \,|\, aSb \,|\, abS \,|\, ab$$

Similarly, the second production for G_1 yields:

$$S \to bSaS \,|\, bSa \,|\, baS \,|\, ba$$

Thus, we have the following productions for G_1:

$$S \to aSbS \,|aSb\,|\, abS \,|\, ab \,|bSaS \,|\, bSa \,|\, baS \,|\, ba$$

There are no unit pairs in this grammar, so we don't have to worry about those. Now, we check for useless states. All symbols are generating so

there are no useless states.

Now we can begin to put G_1 into Chomsky Normal Form. First, we give variables to the terminals a and b; that is,

$$A \to a \quad B \to b$$

If we look at grammar G_1, we now have

$$S \to ASBS \,|\, ASB \,|\, ABS \,|\, AB \,|\, BSAS \,|\, BSA \,|\, BAS \,|\, BA$$

Now, we break apart bodies of length 3 or more into a cascade of productions like so

$$C \to AS \quad D \to BS$$

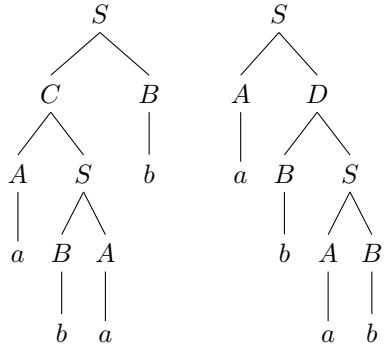
Grammar G_1 is now

$$S \to CD \,|\, CB \,|\, AD \,|\, AB \,|\, DC \,|\, DA \,|\, BC \,|BA$$

This gives us the Chomsky Normal Form for grammar G. Let us use the Cocke-Younger-Kasami algorithm to parse the sentence `abab` according to the CNF grammar.

$\{S\}$			
$\{C\}$	$\{D\}$		
$\{S\}$	$\{S\}$	$\{S\}$	
$\{A\}$	$\{B\}$	$\{A\}$	$\{B\}$
`a`	`b`	`a`	`b`

The CYK table for the CNF of G. We can reconstruct the parse trees as



By following the path given to us by the CYK algorithm.

3. From the grammar G in question (1), construct a pushdown automaton that accepts $L(G)$ by empty stack. Show all sequences of moves that your PDA can make to accept the input string `abab`.

Solution: From grammar G, we construct the pushdown automata P. Recall that the grammar G is:

$$S \to aSbS \,|\, bSaS \,|\, \epsilon$$

The set of input symbols for the PDA is $\{a, b, \epsilon\}$. These three symbols and S form the stack alphabet. The transition function for the PDA is:

(a) $\delta(q, \epsilon, S) = \{(q, aSbS), (q, bSaS), (q, \epsilon)\}$

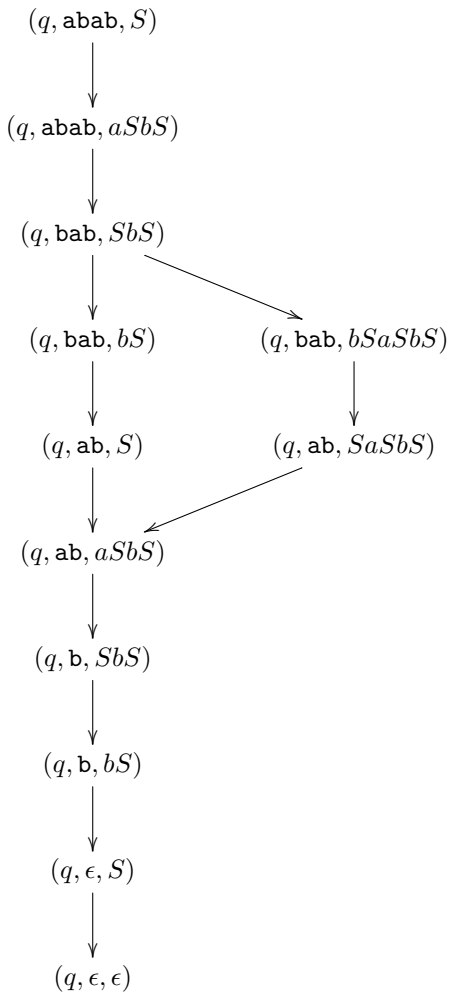
(b) $\delta(q, a, a) = \{(q, \epsilon)\}$; $\delta(q, b, b) = \{(q, \epsilon)\}$; $\delta(q, \epsilon, \epsilon) = \{(q, \epsilon)\}$.

Thus, for PDA P, we have

$$P = (\{q\}, \{a, b, \epsilon\}, \{a, b, \epsilon, S\}, \delta, q, S)$$

where δ is the transition function described above. We omit F, the set of *accepting states*, since P accepts by empty stack.

We now show all sequences of moves that P can make to accept the input string `abab`. Arrows represent the $\vdash$ relation. Note: I am showing only the two accepting paths. if I were to show all paths, the diagram would probably spread over 5 pages.



4. From your PDA in question (3), construct an equivalent context-free grammar. Show a parse tree for the input string `abab` according to your grammar.

Solution: Let us convert the PDA $P = (\{q\}, \{a, b, \epsilon\}, \{a, b, \epsilon, S\}, \delta, q, S)$ into a CFG G_2. There are only two variables in the grammar G_2:

(a) S, the start symbol.

(b) $[qSq]$, the only triple that can be assembled from the states and stack symbols of P.

The productions for grammar G_2 are as follows:

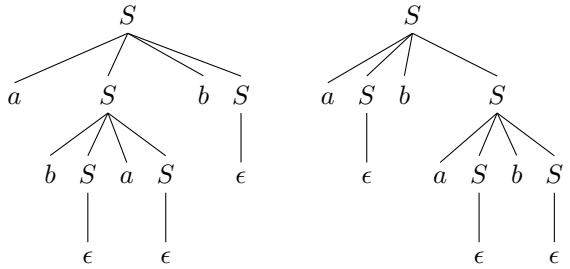
1. The only production for S is $S \to [qSq]$.
2. From the fact that $\delta(q, \epsilon, S)$ contains $(q, aSbS)$, we have $[qSq] \to a[qSq]b[qSq]$. Similarly, since $\delta(q, \epsilon, S)$ contains $(q, bSaS)$ we have $[qSq] \to b[qSq]a[qSq]$. Finally, we have $[qSq] \to \epsilon$ We may, for convenience, replace the triple, $[qSq]$, by some less complex symbol, A. Thus, our productions become

$$S \to A$$
$$A \to bAaA \,|\, aAbA \,|\, \epsilon$$

We can express these productions as

$$G_2 = (\{S\}, \{a, b, \epsilon\}, \{S \to aSbS \,|\, bSaS \,|\, \epsilon\}, S)$$

The parse tree of string `abab` according to grammar G_2 is



5. Consider the language L consisting of all strings of a's and b's that are even-length palindromes with the same number of a's as b's. If L is context free, construct a CFG for L and prove that your grammar generates precisely this language. If L is not context free, use the pumping lemma for context-free languages to prove that L is not context free.

Solution: We have a language L which consists of all strings of a's and b's which are even-length palindromes with the same number of a's and b's. Let us choose $z = a^n b^n b^n a^n$, where n is the number of a's and b's. This string is certainly in the language L. Next, z is broken up into 5 parts: $uvwxy$, with the constraints being $|vwx| \leq n$, and $vx \neq \epsilon$. Thus, we have three cases

1. vwx is the string of all a's before the b's in the palindrome.
2. vwx is the string of all b's after the set of all a's in the palindrome.
3. vwx is the string of a's and b's in z.

CASE 1: Since $|vwx| \leq n$, vwx is in the string of a's either located at the beginning or end of the palindrome. Since $vx \neq \epsilon$, if we look at the case where $i = 0$, we have v^0 and x^0; thus, we are losing all the a's within v and x. Therefore, z does not have an equal number of a's and b's. This statement causes a contradiction since z is in language L.

CASE 2: vwx is the strings of b's in the palindrome; that is, vwx is in either the first b^n or the second b^n, since $|vwx| \leq n$. Since $vx \neq \epsilon$, the case of $i = 0$ implies that there are less than $2n$ b's while there are still $2n$ a's in the string z. Therefore, there is not the same number of a's and b's. Again, this is a contradiction since we stated z is in language L, when it clearly cannot be.

CASE 3: vwx is the string of a's and b's in z such that $|vwx| \leq n$. Since $vx \neq \epsilon$, if we look at the case where $i = 0$, we see that z can no longer be a palindrome since there will be fewer than n a's and fewer than n b's on one side of string z. Thus, z cannot be in L and this provides us with a contradiction.

Whichever case holds, we conclude that L has a string we know not to be in L. This contradiction allows us to conclude that our assumption was wrong; L is *not* a CFL. $\square$