CS 373: Theory of Computation
Sariel Har-Peled and Madhusudan Parthasarathy

Discussion : Context-Free Grammars

March 3, 2009

Questions on homework or exam?

Any questions? Complaints, etc?

1 Context free grammar for languages with balance or without it

1.1 Balanced strings

Let $L_{\mathtt{a}=\mathtt{b}} = \left\{ \mathtt{a}^n \mathtt{b}^n \;\middle|\; n \geq 1 \right\}$. Here is a grammar for this language

$$\mathsf{S} \rightarrow \mathtt{a}\mathsf{S}\mathtt{b} \mid \epsilon.$$

1.2 Mixed balanced strings

Let $L^{\mathrm{mix}}_{\mathtt{a}=\mathtt{b}}$ be the language of all strings over $\{\mathtt{a}, \mathtt{b}\}$, with equal number of as and bs, where the as and bs might be mixed together.

We the following grammar for generating all strings that have the same numbers of a's and b's

$$\mathsf{S} \rightarrow \; \mathtt{a}\mathsf{S}\mathtt{b} \mid \mathtt{b}\mathsf{S}\mathtt{a} \mid \mathsf{S}\mathsf{S} \mid \epsilon,$$

where S is the start variable.

Advice To TA:: State the following lemma, and sketch its proof, but do not do the proof in the discussion section. Point the interested students to the class notes. end

Lemma 1.1 *We have* $L(\mathcal{G}) = L^{\mathrm{mix}}_{\mathtt{a}=\mathtt{b}}$.

Proof: It is easy to see that every string that $\mathcal{G}$ generates has equal number of a's and b's. As such, $L(\mathcal{G}) \subseteq L^{\mathrm{mix}}_{\mathtt{a}=\mathtt{b}}$.

We will use induction on the length of string $x \in L(\mathcal{G})$, $2n = |x|$. For $n = 0$ we can generate ϵ by $\mathcal{G}$. For $n = 1$, we can generate both ab and ba by $\mathcal{G}$.

Now for $n > 1$, consider a balanced string with length $2n$, $x = x_1 x_2 x_3 \cdots x_{2n} \in L^{\mathrm{mix}}_{\mathtt{a}=\mathtt{b}}$. Let $\#_c(y)$ be the number of appearances of the character c in the string y. Let $\alpha_i = \#_{\mathtt{a}}(x_1 \cdots x_i) - \#_{\mathtt{b}}(x_1 \cdots x_i)$. Observe that $\alpha_0 = \alpha_{2n} = 0$. If $\alpha_j = 0$, for some $1 < j < 2n$,

then we can break x into two words $y = x_1 \dots x_j$ and $z = x_{j+1} \dots x_{2n}$ that are both balanced. By induction, $y, z \in L(\mathcal{G})$, and as such $\mathsf{S} \Rightarrow^* y$ and $\mathsf{S} \Rightarrow^* z$. This implies that

$$\mathsf{S} \Rightarrow \mathsf{SS} \Rightarrow^* yz = x.$$

Namely, $x \in L(\mathcal{G})$.

The remaining case is that $\alpha_j \neq 0$ for $j = 2, \dots, 2n-1$. If $x_1 = \mathtt{a}$ then $\alpha_1 = 1$. As such, for all $j = 1, \dots, 2n-1$, we must have that $\alpha_j > 0$. But then $\alpha_{2n} = 0$, which implies that $\alpha_{2n-1} = 1$. We conclude that $x_1 = \mathtt{a}$ and $x_{2n} = \mathtt{b}$. As such, $x_2 \dots x_{2n-1}$ is a balanced word, which by induction is generated by $L(\mathcal{G})$. Thus, the x can be derived via $\mathsf{S} \to \mathtt{a}\mathsf{S}\mathtt{b} \Rightarrow^* \mathtt{a}x_2x_3 \dots x_{2n-1}\mathtt{b} = x$. Thus, $x \in L(\mathcal{G})$.

The case $x_1 = \mathtt{b}$ is handled in a similar fashion, and implies that $x \in L(\mathcal{G})$ also in this case. We conclude that $L^{\text{mix}}_{\mathtt{a}=\mathtt{b}} \subseteq L(\mathcal{G})$.

Thus $L^{\text{mix}}_{\mathtt{a}=\mathtt{b}} = L(\mathcal{G})$. ∎

1.3 Unbalanced pair

Consider the following language:

$$L_{\mathtt{a}\neq\mathtt{b}} = \left\{ \mathtt{a}^n\mathtt{b}^m \;\middle|\; n \neq m \text{ and } \; n, m \geq 0 \right\}.$$

If $n \neq m$ then either $n > m$ or $m > n$, therefore we can design this grammar by first starting with the basic grammar for when $n = m$, and then transition into making more $\mathtt{a}$'s or $\mathtt{b}$'s.

Let X be the non terminal representing "choosing" to generate more $\mathtt{a}$'s than $\mathtt{b}$'s and Y be the non-terminal for the other case. One grammar that generates $L_{\mathtt{a}\neq\mathtt{b}}$ will therefore be:

$$\mathsf{S} \to \mathtt{a}\mathsf{S}\mathtt{b} \mid \mathtt{a}\mathsf{A} \mid \mathtt{b}\mathsf{B}, \qquad \mathsf{A} \to \mathtt{a}\mathsf{A} \mid \epsilon, \qquad \mathsf{B} \to \mathtt{b}\mathsf{B} \mid \epsilon.$$

1.4 Balanced pair in a triple

Consider the language

$$L_4 = \left\{ \mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \;\middle|\; i = j \text{ or } \; j = k \right\}.$$

We can essentially combine two copies of the previous grammar (with one version that works on b and c) in order to create a grammar that generates L_2:

$$\mathsf{S} \to \mathsf{S}_{\mathtt{a}=\mathtt{b}}\mathsf{C} \mid \mathsf{A}\mathsf{S}_{\mathtt{b}=\mathtt{c}}$$

$$\mathsf{S}_{\mathtt{a}=\mathtt{b}} \to \mathtt{a}\mathsf{S}_{\mathtt{a}=\mathtt{b}}\mathtt{b} \mid \epsilon. \qquad \mathsf{S}_{\mathtt{b}=\mathtt{c}} \to \mathtt{b}\mathsf{S}_{\mathtt{b}=\mathtt{c}}\mathtt{c} \mid \epsilon. \qquad \mathsf{A} \to \mathsf{A}\mathtt{a} \mid \epsilon \qquad \mathsf{C} \to \mathsf{C}\mathtt{c} \mid \epsilon.$$

Exercise 1.2 *Derive a CFG for the language* $L'_4 = \left\{ \mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \;\middle|\; i = j \text{ or } j = k \text{ or } i = k \right\}$.

1.5 Unbalanced pair in a triple

Now consider the related language

$$L_2 = \left\{ \mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \;\middle|\; i \neq j \text{ or } \; j \neq k \right\}.$$

We can essentially combine two copies of the previous grammar (with one version that works on b and c) in order to create a grammar that generates L_2:

$$\mathsf{S} \to \mathsf{S}_{\mathtt{a}\neq\mathtt{b}}\mathsf{C} \mid \mathsf{A}\mathsf{S}_{\mathtt{b}\neq\mathtt{c}} \qquad \mathsf{S}_{\mathtt{a}\neq\mathtt{b}} \to \mathtt{a}\mathsf{S}_{\mathtt{a}\neq\mathtt{b}}\mathtt{b} \mid \mathtt{a}\mathsf{A} \mid \mathtt{b}\mathsf{B}. \qquad \mathsf{S}_{\mathtt{b}\neq\mathtt{c}} \to \mathtt{b}\mathsf{S}_{\mathtt{b}\neq\mathtt{c}}\mathtt{c} \mid \mathtt{b}\mathsf{B} \mid \mathtt{c}\mathsf{C}.$$

$$\mathsf{A} \to \mathsf{A}\mathtt{a} \mid \epsilon \qquad \mathsf{B} \to \mathsf{B}\mathtt{b} \mid \epsilon \qquad \mathsf{C} \to \mathsf{C}\mathtt{c} \mid \epsilon.$$

1.6 Anything but balanced

Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, and let Let $\overline{L_{\mathtt{a}=\mathtt{b}}} = \Sigma^* \setminus \left\{ \mathtt{a}^n\mathtt{b}^n \;\middle|\; n \geq 1 \right\}$.

The idea is that lets first generate all words that contain b in them, and then later the contain a. The grammar for this language is

$$\mathsf{S}_1 \to \mathsf{Z}\mathtt{b}\mathsf{Z}\mathtt{a}\mathsf{Z} \qquad \mathsf{Z} \to \mathtt{a}\mathsf{Z} \mid \mathtt{b}\mathsf{Z} \mid \epsilon.$$

Clearly $L(\mathsf{Z}) \subseteq \overline{L_{\mathtt{a}=\mathtt{b}}}$. The only words we miss, must have all their as before their bs. But these are all words of the form $\mathtt{a}^i\mathtt{b}^j$, where $i \neq j \geq 0$. But we already saw how to generate such words in Section 1.3. Putting everything together, we get the following grammar.

$$\begin{aligned}
\Rightarrow &\mathsf{S} \to \mathsf{S}_1 \mid \mathsf{S}_{\mathtt{a}\neq\mathtt{b}} \\
&\mathsf{S}_1 \to \mathsf{Z}\mathtt{b}\mathsf{Z}\mathtt{a}\mathsf{Z} \\
&\mathsf{Z} \to \mathtt{a}\mathsf{Z} \mid \mathtt{b}\mathsf{Z} \mid \epsilon \\
&\mathsf{S}_{\mathtt{a}\neq\mathtt{b}} \to \mathtt{a}\mathsf{S}_{\mathtt{a}\neq\mathtt{b}}\mathtt{b} \mid \mathtt{a}\mathsf{A} \mid \mathtt{b}\mathsf{B}, \\
&\mathsf{A} \to \mathtt{a}\mathsf{A} \mid \epsilon, \\
&\mathsf{B} \to \mathtt{b}\mathsf{B} \mid \epsilon.
\end{aligned}$$

2 Similar count

Consider the language

$$L = \left\{ w\mathtt{0}^n \;\middle|\; w \in \{\mathtt{a}, \mathtt{b}\}^* \text{ and } \#_{\mathtt{a}}(w) = n \right\},$$

where $\#_{\mathtt{a}}(w)$ is the number of appearances of the character a in w. The grammar for this language is

$$\mathsf{S} \to \epsilon \mid \mathtt{b}\mathsf{S} \mid \mathtt{a}\mathsf{S}\mathtt{0}.$$

3 Inherent Ambiguity

In lecture, the following ambiguous grammar representing basic mathematical statements was discussed:

$$\mathsf{E} \to \mathsf{E} * \mathsf{E} \mid \mathsf{E} + \mathsf{E} \quad \mathsf{N} \to \mathtt{0}\mathsf{N} \mid \mathtt{1}\mathsf{N} \mid \mathtt{0} \mid \mathtt{1}.$$

The ambiguity caused because there is no inherent preference from combining expressions with $*$ over $+$ or vise versa. It was then fixed by introducing a preference :

$$\mathsf{E} \to \mathsf{E} * \mathsf{E} \mid \mathsf{T}, \qquad \mathsf{T} \to N * T \mid \mathsf{N} \qquad \mathsf{N} \to \mathtt{0}\mathsf{N} \mid \mathtt{1}\mathsf{N} \mid \mathtt{0} \mid \mathtt{1}.$$

However some languages are inherently ambiguous, no context free grammar without ambiguity can generate it.

Consider the following language:

$$L = \left\{ \mathtt{a}^n\mathtt{b}^n\mathtt{c}^k\mathtt{d}^k \;\middle|\; n, k \geq 1 \right\} \cup \left\{ \mathtt{a}^n\mathtt{b}^k\mathtt{c}^k\mathtt{d}^n \;\middle|\; n, k \geq 1 \right\}.$$

In other words, it is the language of $a^+b^+c^+d^+$ where either:

1. the number of `a`'s equals the number of `b`'s and the number `c`'s equals the number of `d`'s
2. the number of `a`'s equals the number of `d`'s and the number of `b`'s equals the number of `c`'s

One ambiguous grammar that generates it

$$\mathsf{S} \to \mathsf{XY} \mid \mathsf{Z}, \qquad X \to \mathtt{a}\mathsf{X}\mathtt{b} \mid \epsilon, \qquad \mathsf{Y} \to \mathtt{c}\mathsf{Y}\mathtt{d} \mid \epsilon, \qquad \mathsf{Z} \to \mathtt{a}\mathsf{Z}\mathtt{d} \mid \mathsf{T}, \qquad \mathsf{T} \to \mathtt{b}\mathsf{T}\mathtt{c} \mid \epsilon.$$

The reason why all grammars for this language must be ambiguous can be seen in strings of the form $\left\{ \mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n \;\middle|\; n \geq 1 \right\}$. Any grammar needs some way of generating the string in a way that either the `a`'s and `b`'s are equal and the `c` and `d`'s are equal or the `a`'s and `d`'s are equal and the `b`'s and `c`'s are equal. When generating equal `a`'s and `b`'s, it must be still possible to have the same number of c's and d's. When generating equal `a`'s and d's , it must still be possible to have the same number of `b`'s and `c`'s. No matter what grammar is designed, any string of the form $\left\{ a^nb^nc^nd^n \;\middle|\; n \geq 1 \right\}$ must have at least two possible parse trees.

(This is of course only an intuitive explanation. A formal proof that any grammar for this language must be ambiguous is considerably more tedious and harder.)

4 A harder example

Consider the following language:

$$L = \left\{ xy \;\middle|\; x, y \in \{0,1\}^* \text{ where } |x| = |y|, \text{ and } x \neq y \right\}.$$

It should be clear that this language cannot be regular. However, it may not be obvious that we can in fact design a context free grammar for it. x and y are guaranteed to be different if, for some k, the kth character is 0 in x and 1 in y (or vise versa). It is important to notice that we should not try to build x and y separately as, in a CFG, we would have no way to enforce them being of the same length. Instead, we just remember that if the string is of length $2n$, the first n characters are considered x and the second n characters are y. Similarly, notice that we cannot choose k ahead of time for similar reasons.

So, consider the following string

$$w = x_1 x_2 \dots x_{k-1} \boxed{1} x_{k+1} \dots x_n y_1 y_2 \dots y_{k-1} \boxed{0} y_{k+1} \dots y_n \in L.$$

Then we can rewrite this string as follows

$$w = \overbrace{x_1 x_2 \dots x_{k-1}}^{k-1 \text{ chars}} \boxed{1} \overbrace{x_{k+1} \dots x_n}^{n-k \text{ chars}} \overbrace{y_1 y_2 \dots y_{k-1}}^{k-1 \text{ chars}} \boxed{0} \overbrace{y_{k+1} \dots y_n}^{n-k \text{ chars}}.$$

In particular, let $z_1 z_2 \dots z_{n-1} = x_{k+1} \dots x_n y_1 y_2 \dots y_{k-1}$. Then,

$$\begin{aligned} w &= \overbrace{x_1 x_2 \dots x_{k-1}}^{k-1 \text{ chars}} 1 z_1 z_2 \dots z_{n-1} 0 \overbrace{y_{k+1} \dots y_n}^{n-k \text{ chars}} \\ &= \underbrace{\overbrace{x_1 x_2 \dots x_{k-1}}^{k-1 \text{ chars}} 1 \overbrace{z_1 \dots z_{k-1}}^{k-1 \text{ chars}}}_{=X} \underbrace{\overbrace{z_k \dots z_{n-1}}^{n-1-k+1 \text{ chars}} 0 \overbrace{y_{k+1} \dots y_n}^{n-k \text{ chars}}}_{=Y}. \end{aligned}$$

Now, X is a word of odd length with 1 in the middle (and we definitely know how to generate this kind of words using context free grammars). And Y is a word of odd length, with 0 in the middle. In particular, any word of L can be written as either XY or YX, where X and Y are as above. We conclude, that the grammar for this language is

$$\mathsf{S} \to \mathsf{XY} \mid \mathsf{YX} \qquad \mathsf{X} \to \mathsf{DXD} \mid 1 \qquad \mathsf{Y} \to \mathsf{DYD} \mid 0 \qquad \mathsf{D} \to 0 \mid 1.$$