Context Free Gramers

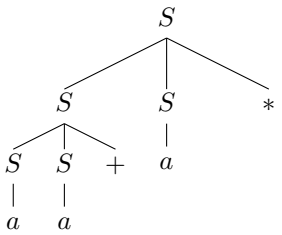
Benji Altman

April 28, 2017

4.2.1:a) $S \Rightarrow SS* \Rightarrow SS+S* \Rightarrow aS+S* \Rightarrow aa+S* \Rightarrow aa+a*$

b) $S \Rightarrow SS* \Rightarrow Sa* \Rightarrow SS+a* \Rightarrow Sa+a* \Rightarrow aa+a*$

c)



d) This is unambiguous, because you can always match with whatever character is on the right side of the string, and work your way back. That is to say if one starts at the end of the string and works your way back you may create a production for any string in the language by following these rules:

1. If you see $*$, then convert your rightmost S to $SS*$.
2. If you see $+$, then convert your rightmost S to $SS+$.
3. If you see a, then convert your rightmost S to a.

e) This language will be the subset of the language given by the regular expression $a(a|*|+)^*$ such that the following two conditions are met.

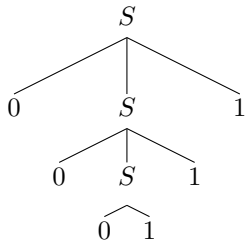
1. $a_i - +_i - *_i \geq 1$ for all $i \in [0, \mathcal{L})$.
2. $a_{\mathcal{L}} - +_{\mathcal{L}} - *_{\mathcal{L}} = 1$.

Where k_i is the number of times that a character k appears in the string in any index in the range $[0, k]$, given 0 based indexing on the string, and $\mathcal{L}$ is the number of symbols in the string. For example $+_{\mathcal{L}}$ is the number of times $+$ appears in the string in the range of indicies $[0, \mathcal{L}]$, a set that includes all indicies the given string, thus $+_{\mathcal{L}}$ is the number of times $+$ apears in the entire string.

4.2.2:a)a) $S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 000111$

b) $S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 000111$

c)



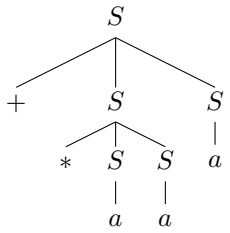
d) This language is unambigous as you must always make $n-1$ conversions from S to $0S1$ followed by a conversion from S to 01, where n is the number of 0s there are in the string. There is only ever 1 S, and thus this is trivially unambigous.

e) This language is $\{0^n 1^n | n \geq 1\}$

b)a) $S \Rightarrow +SS \Rightarrow +*SSS \Rightarrow +*aSS \Rightarrow +*aaS \Rightarrow +*aaa$

b) $S \Rightarrow +SS \Rightarrow +Sa \Rightarrow +*SSa \Rightarrow +*Saa \Rightarrow +*aaa$

c)



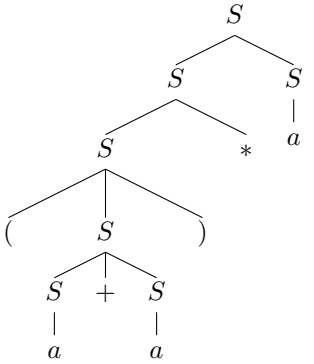
d) This is unambigious for much the same reason as in question 4.2.1, although this time we parse through the string in the opposite direction and the conversions must be changed to the corrosponding conversions for this grammer.

e) This language is the set of all strings that are in the language given by the grammer in question 4.2.1, but the ordering of all the characters in the string is flipped.

d)a) $S \Rightarrow SS \Rightarrow S*S \Rightarrow (S)*S \Rightarrow (S+S)*S \Rightarrow (a+S)*S \Rightarrow (a+a)*S \Rightarrow (a+a)*a$

b) $S \Rightarrow SS \Rightarrow Sa \Rightarrow S*a \Rightarrow (S)*a \Rightarrow (S+S)*a \Rightarrow (S+a)*a \Rightarrow (a+a)*a$

c)



d) This grammer is ambigious, due to the equivlency of these two valid parse trees:

e) Think of $*$ as a postfixed unary opperator. Think of nothing between two terms as a binary operator on them, like multiplication in normal mathematical notation. Think of $+$ as an infixed binary operation, like it normally is. Think of parentheses as one normally does. Think of a as the only allowed value, and then any well formed formula with these rules is valid in this language and this language is the set of all of these well formed formulas.

4.2.3:a)

$$S \quad \rightarrow \quad 1\ S \mid 0\ 1\ S \mid \lambda$$

b)

$$S \quad \rightarrow \quad 1\ S\ 1 \mid 0\ S\ 0 \mid \lambda$$

c)

$$S \quad \rightarrow \quad 0\ S\ 1\ S \mid 1\ S\ 0\ S \mid \lambda$$