Tutorial 1

September 20, 2004
1 REs and CFGs

Give regular expressions and context free grammars (using BNF notation) for the following languages or say it cannot be done

1. All strings over the alphabet \( \{0,1,2\} \) sorted in decreasing order.
   \( 2^* 1^* 0^* \)

   \[
   \begin{align*}
   <S> & ::= <\text{twos}> <\text{ones}> <\text{zeros}> | \text{epsilon} \\
   <\text{twos}> & ::= 2 <\text{twos}> | \text{epsilon} \\
   <\text{ones}> & ::= 1 <\text{ones}> | \text{epsilon} \\
   <\text{zeros}> & ::= 0 <\text{zeros}> | \text{epsilon}
   \end{align*}
   \]

   Note: it is easy to write a CFG from a RE.

2. All strings over the alphabet \( \{a,b\} \), in which every \( b \) is both immediately preceded by and followed by at least one \( a \).
   \( (a(ba)^*)^* \)

   \[
   \begin{align*}
   <S> & ::= a <BA> <S> | \text{epsilon} \\
   <BA> & ::= b a <BA> | \text{epsilon}
   \end{align*}
   \]

3. \( a^n b^n \ n \geq 0 \)
   Cannot be expressed by RE.

   \[
   <S> ::= a <S> b | \text{epsilon}
   \]

4. \( a^n b^m a^n b^m \ n, m \geq 0 \)
   Cannot be expressed by RE.
   Cannot be expressed by CFG.
2 ambiguity

a) \(<S> ::= <S> + <S> | <S> * <S> | a | b | c\)

Is this grammar ambiguous?
Yes. Two parse trees for ‘‘a + b * c’’.
Develop a grammar that generates the same strings,
but is not ambiguous.

\(<S> ::= <ADD>\)
\(<ADD> ::= <MULT> | <ADD> + <MULT>\)
\(<MULT> ::= <VAR> | <MULT> * <VAR>\)
\(<VAR> ::= a | b | c\)

b) \(<S> ::= a <S> a | b <S> b | \text{epsilon}\)

Is this grammar ambiguous?
No.
3  english descriptions

Give English descriptions of the languages described by each of the following grammars.

a) \( <S> ::= <S> \ a \ <S> \ b \ <S> \ | \ <S> \ b \ <S> \ a \ <S> \ | \ \text{epsilon} \)

Answer:
All strings with an equal number of a’s and b’s.

Example:
\[
\begin{align*}
<S> & \Rightarrow <S> \ a \ <S> \ b
\Rightarrow <S> \ b \ <S> \ a \ <S> \ a \ <S> \ b \ <S>
\Rightarrow <S> \ b \ <S> \ aa \ <S> \ b \ <S>
\Rightarrow <S> \ b \ <S> \ aa \ <S> \ b \ <S> \ a \ <S> \ b \ <S>
\Rightarrow \text{baabab}
\end{align*}
\]

b) \( <S> ::= a \ <S> \ a \ | \ b \ <S> \ b \ | \ c \ <S> \ c \ | \ \text{epsilon} \)

Answer: All even length strings over \{a,b,c\}, where the second half of the string is the reverse of the first half.

Example:
\[
\begin{align*}
<S> & \Rightarrow a \ <S> \ a
\Rightarrow a \ b \ <S> \ b \ a
\Rightarrow a \ b \ c \ <S> \ c \ b \ a
\Rightarrow a \ b \ c \ b \ <S> \ b \ c \ b \ a
\Rightarrow \text{abcb bcba}
\end{align*}
\]

c) Adjust the grammar in (b) to make it generate all palindromes.

Need to generate odd length strings.

Answer: \( <S> ::= a \ <S> \ a \ | \ b \ <S> \ b \ | \ c \ <S> \ c \ | \ a \ | \ b \ | \ c \ | \ \text{epsilon} \)