Introduction to Context Free Grammar
Additional Notes on the CYK algorithm

Jerry Wang
CPSC 421/501

This document contains some sample execution traces of the standard CYK algorithm. Recall that the context free grammar for PALINDROME over $\{a, b\}$ is $S \rightarrow aSa \mid bSb \mid \epsilon$. The equivalent grammar rendered in Chomsky Normal Form is… (P is the start variable)

$$P \rightarrow AX \mid BY \mid \epsilon$$
$$S \rightarrow AX \mid BY$$
$$X \rightarrow SA \mid a$$
$$Y \rightarrow SB \mid b$$
$$A \rightarrow a$$
$$B \rightarrow b$$

Note that the language of this grammar does not contain palindromes of an odd length, such as “aba”.

Trace 1: Input = “aab”

```
root@localhost ContextFreeGrammar % ./palindrome aab
>> Checking whether the input string "aab" is in this context free language...
>> Checking each substring of length 1...
     Checking the substring "aab"...
          The rule "A -> a" can generate the substring "a".
          The rule "X -> a" can generate the substring "a".
     Checking the substring "aab"...
          The rule "A -> a" can generate the substring "a".
          The rule "X -> a" can generate the substring "a".
     Checking the substring "aab"...
          The rule "Y -> b" can generate the substring "b".
          The rule "B -> b" can generate the substring "b".
>> Checking each substring of length 2...
     Checking the substring "aab"...
          Split the substring into "a" and "a".
               The rule P -> AX can generate the substring "aab".
               The rule S -> AX can generate the substring "aab".
     Checking the substring "aab"...
          Split the substring into "a" and "b".
>> Checking each substring of length 3...
     Checking the substring "aab"...
          Split the substring into "a" and "ab".
          Split the substring into "aa" and "b".
               The rule Y -> SB can generate the substring "aab".
"aab" is a palindrome: false
```

Trace 2: Input = “abba”

```
root@localhost ContextFreeGrammar % ./palindrome abba
>> Checking whether the input string "abba" is in this context free
language...
>> Checking each substring of length 1...
     Checking the substring "abba"...
          The rule "A -> a" can generate the substring "a".
          The rule "X -> a" can generate the substring "a".
     Checking the substring "abba"...
          The rule "Y -> b" can generate the substring "b".
          The rule "B -> b" can generate the substring "b".
     Checking the substring "abba"...
          The rule "Y -> b" can generate the substring "b".
          The rule "B -> b" can generate the substring "b".
     Checking the substring "abba"...
          The rule "A -> a" can generate the substring "a".
          The rule "X -> a" can generate the substring "a".
>> Checking each substring of length 2...
     Checking the substring "abba"...
          Split the substring into "a" and "b".
     Checking the substring "abba"...
          Split the substring into "b" and "b".
               The rule S -> BY can generate the substring "abba".
               The rule P -> BY can generate the substring "abba".
     Checking the substring "abba"...
          Split the substring into "b" and "a".
>> Checking each substring of length 3...
     Checking the substring "abba"...
          Split the substring into "a" and "bb".
          Split the substring into "ab" and "b".
     Checking the substring "abba"...
          Split the substring into "b" and "ba".
          Split the substring into "bb" and "a".
               The rule X -> SA can generate the substring "abba".
>> Checking each substring of length 4...
     Checking the substring "abba"...
          Split the substring into "a" and "bba".
               The rule S -> AX can generate the substring "abba".
               The rule P -> AX can generate the substring "abba".
          Split the substring into "ab" and "ba".
          Split the substring into "abb" and "a".
"abba" is a palindrome: true
```

Trace 3: Input = “abaaba”

```
root@localhost ContextFreeGrammar % ./palindrome abaaba
>> Checking whether the input string "abaaba" is in this context free
language...
>> Checking each substring of length 1...
     Checking the substring "abaaba"...
          The rule "X -> a" can generate the substring "a".
          The rule "A -> a" can generate the substring "a".
     Checking the substring "abaaba"...
          The rule "B -> b" can generate the substring "b".
          The rule "Y -> b" can generate the substring "b".
     Checking the substring "abaaba"...
          The rule "X -> a" can generate the substring "a".
          The rule "A -> a" can generate the substring "a".
     Checking the substring "abaaba"...
          The rule "X -> a" can generate the substring "a".
          The rule "A -> a" can generate the substring "a".
     Checking the substring "abaaba"...
          The rule "B -> b" can generate the substring "b".
          The rule "Y -> b" can generate the substring "b".
     Checking the substring "abaaba"...
          The rule "X -> a" can generate the substring "a".
          The rule "A -> a" can generate the substring "a".
>> Checking each substring of length 2...
     Checking the substring "abaaba"...
          Split the substring into "a" and "b".
     Checking the substring "abaaba"...
          Split the substring into "b" and "a".
     Checking the substring "abaaba"...
          Split the substring into "a" and "a".
               The rule S -> AX can generate the substring "abaaba".
               The rule P -> AX can generate the substring "abaaba".
     Checking the substring "abaaba"...
          Split the substring into "a" and "b".
     Checking the substring "abaaba"...
          Split the substring into "b" and "a".
>> Checking each substring of length 3...
     Checking the substring "abaaba"...
          Split the substring into "a" and "ba".
          Split the substring into "ab" and "a".
     Checking the substring "abaaba"...
          Split the substring into "b" and "aa".
          Split the substring into "ba" and "a".
     Checking the substring "abaaba"...
          Split the substring into "a" and "ab".
          Split the substring into "aa" and "b".
               The rule Y -> SB can generate the substring "abaaba".
     Checking the substring "abaaba"...
          Split the substring into "a" and "ba".
          Split the substring into "ab" and "a".
>> Checking each substring of length 4...
     Checking the substring "abaaba"...
```

```
          Split the substring into "a" and "baa".
          Split the substring into "ab" and "aa".
          Split the substring into "aba" and "a".
     Checking the substring "abaaba"...
          Split the substring into "b" and "aab".
               The rule S -> BY can generate the substring "abaaba".
               The rule P -> BY can generate the substring "abaaba".
          Split the substring into "ba" and "ab".
          Split the substring into "baa" and "b".
     Checking the substring "abaaba"...
          Split the substring into "a" and "aba".
          Split the substring into "aa" and "ba".
          Split the substring into "aab" and "a".
>> Checking each substring of length 5...
     Checking the substring "abaaba"...
          Split the substring into "a" and "baab".
          Split the substring into "ab" and "aab".
          Split the substring into "aba" and "ab".
          Split the substring into "abaa" and "b".
     Checking the substring "abaaba"...
          Split the substring into "b" and "aaba".
          Split the substring into "ba" and "aba".
          Split the substring into "baa" and "ba".
          Split the substring into "baab" and "a".
               The rule X -> SA can generate the substring "abaaba".
>> Checking each substring of length 6...
     Checking the substring "abaaba"...
          Split the substring into "a" and "baaba".
               The rule S -> AX can generate the substring "abaaba".
               The rule P -> AX can generate the substring "abaaba".
          Split the substring into "ab" and "aaba".
          Split the substring into "aba" and "aba".
          Split the substring into "abaa" and "ba".
          Split the substring into "abaab" and "a".
"abaaba" is a palindrome: true
```