# Automata Theory, Languages, and Computation

Name: ______________________ Date: ___________

_Note_: *The purpose of the following questions is:*

| | | |
|---|---|---|
| • *Enhance learning* | • *Summarized points* | • *Analyze abstract ideas* |

## Class 12: Deterministic Pushdown Automata, Properties of Context-Free Languages, Yacc

### Positive Properties of Context-Free Languages:

1. Show that the family of context-free languages is closed under Union.
2. Show that the family of context-free languages is closed under Concatenation.
3. Show that the family of context-free languages is closed under Star-operation.

### Negative Properties of Context-Free Languages:

4. Show that the family of context-free languages is not closed under Intersection.
5. Show that the family of context-free languages is not closed under Complement.

### Intersection of Context-free Languages and Regular Languages

6. Show that the intersection of a context-free language and a regular language is a context-free language, i.e. prove the following theorem:
   *_Theorem:_ Let $L_1$ be a context-free language and $L_2$ be a regular language. Then $L_1 \cap L_2$ is context-free*

### Applications of Regular Closure

In *_class 4_* we looked at closure under certain operations and algorithms to decide on the properties of the family of regular languages. On the whole, the questions raised there had easy answers. When we ask the same questions about context-free languages, we encounter more difficulties. First, closure properties that hold for regular languages do not always hold for context-free languages. When they do, the arguments needed to prove them are often quite complicated. Second, many intuitively simple and important questions about context-free languages connot be answered. This statement may seem at first surprising and will need to be elaborated as we proceed.

7. Show that the language

$$L=\{a^nb^n: n\geq 0, n\neq 100\}$$

   *is context-free*

   ***_Note:_*** *It is possible to prove this claim by constructing a pda or a context-free grammar for the language, but the process is tedious. We can get a much neater argument with the _theorem:_ Let $L_1$ be a context-free language and $L_2$ be a regular language. Then $L_1 \cap L_2$ is context-free*

8. Show that the language

$$L = \{w: n_a = n_b = n_c\}$$

is *not* context-free.

**<u>Note:</u>** *The pumping lemma can be used for this, but again we can get a much shorter argument using closure under regular intersection.*

## Decidable Properties of Context-Free Languages

9. Prove, <u>Theorem:</u> Given a context-free grammar $G = (V,T,S,P)$, there exists an algorithm for deciding whether or not $L(G)$ is empty.
10. Prove, <u>Theorem:</u> Given a context-free grammar $G = (V,T,S,P)$, there exists an algorithm for determining whether or not $L(G)$ is infinite.
11. For the following context-free grammar G find if L(G) is infinite

    $S \rightarrow AB$

    $A \rightarrow aCb|a$

    $B \rightarrow bB|bb$

    $C \rightarrow cBS$

    <u>*Hint:*</u> *Create dependency graph for variables and get the leftmost derivation.*

## YACC

The computer program **Yacc** is a parser generator developed by Stephen C. Johnson at AT&T Corporation for the Unix operating system in 1970. The name is an acronym for "Yet Another Compiler Compiler". It generates a parser (the part of a compiler that tries to make syntactic sense of the source code) based on an analytic grammar written in a notation similar to BNF.

Yacc used to be available as the default parser generator on most Unix systems. It has since been supplanted as the default by more recent, largely compatible, programs such as Berkeley Yacc, GNU bison, MKS Yacc and Abraxas PCYACC. An updated version of the original AT&T version is included as part of Sun's OpenSolaris project. Each offers slight improvements and additional features over the original Yacc, but the concept has remained the same. Yacc has also been rewritten for other languages, including Ratfor, ML, Ada, Pascal, Java, Python, Ruby, Go and Common Lisp.

The parser generated by Yacc is a LALR parser. In order to run, it requires an external lexical analyzer. Lexical analyzer generators, such as Lex or Flex are widely available. The IEEE POSIX P1003.2 standard defines the functionality and requirements for both Lex and Yacc.

Some versions of AT&T Yacc have become open source. For example, source code (for different implementations) is available with the standard distributions of Plan 9 and OpenSolaris.