CPSC 229, Fall 2015 — Third Test Info

*The third test for this course will be given in class on Wednesday, November 18. It covers material from Sections 3.4 through Section 4.1. However, this material depends heavily on Sections 3.1 and 3.2; although there will not be specific questions on those sections, you still need to know the things that are covered in those section. Note there there will be **no** questions on regular expressions on the computer (Section 3.3) or on Backus-Naur form (Section 4.2).*

The format will be similar to previous tests, with a mix of definitions, longer essays, and problems. You can expect to apply the NFA-to-DFA conversion algorithm. There might also be a problem using the regular-expression-to-NFA algorithm. And you can expect to use the Pumping Lemma to prove that a language is not regular. There will not be any other formal proofs, but you might need to be familiar with certain results and the general idea of their proofs, such as the fact that the complement of a regular language is regular.

Here are some terms and ideas that you should be familiar with for the test:

FSA (Finite-State Automaton) — this is just a general term for NFA or DFA
DFA (Deterministic Finite Automaton)
transition diagram [the usual picture] of a DFA
state (in a finite-state automaton)
start state
accepting state (also known as final state)
definition of a DFA as a list of five things, $(Q, \Sigma, q_o, \delta, F)$ — and what each thing means
transition function, $\delta\colon Q \times \Sigma \to Q$, of a DFA
transition table for a DFA
the function $\delta^*\colon Q \times \Sigma^* \to Q$
how a DFA computes (that is, what it does when it reads and processes a string)
what it means for a DFA to accept a language
the language accepted by a DFA, $M = (Q, \Sigma, q_o, \delta, F)$: $L(M) = \{w \in \Sigma^* \mid \delta^*(q_o, w) \in F\}$
NFA (Non-deterministic Finite Automaton)
nondeterminism
the differences between NFAs and DFAs
ε-transitions
what it means for an NFA to accept a string
the language, $L(M)$, accepted by an NFA, M
algorithm for converting an NFA to an equivalent DFA
algorithm for converting a regular expression to an equivalent NFA
DFAs, NFAs, and regular expressions all define the same class of languages
operations ($L_1 \cup L_2$, $L_1 \cap L_2$, LM, L^*, L^R) on regular languages produce regular languages
how to get a DFA for the complement, $\overline{L}$, of a regular language L

how to get a DFA for the intersection of two regular languages
the Pumping Lemma for Regular Languages
using the pumping lemma to show that certain specific languages are not regular
CFGs (Context-Free Grammars)
production rules
non-terminal and terminal symbols
start symbol
definition of a CFG as a 4-tuple $G = (V, \Sigma, P, S)$
the relations $\Longrightarrow$ and $\Longrightarrow^*$
derivation (of a string from the start symbol of a CFG)
the language generated by a CFG $G = (V, \Sigma, P, S)$: $L(G) = \{\, w \in \Sigma^* \mid S \Longrightarrow^* w \,\}$
context-free language
finding a CFG for a given language and *vice versa*
if L and M are context-free languages, then so are $L \cup M$, LM, and L^*
every regular language is context-free
examples of languages that are not regular, such as:

$\{a^n b^n \mid n \in \mathbb{N}\}$
$\{a^n b^m c^k \mid k = n + m\}$
$\{w \in \{a, b\}^* \mid w = w^R\}$
$\{w \in \{a, b\}^* \mid n_a(w) < n_b(w)\}$
$\{ww \mid w \in \{a, b\}^*\}$
$\{a^n b^n c^n \mid n \in \mathbb{N}\}$
$\{a^{n^2} \mid n \in \mathbb{N}\}$
$\{www \mid w \in \{a, b\}^*\}$

examples of languages that are context-free but not regular, such as:

the **first four** languages in the previous list

examples of languages that are not context-free, such as:

the **last four** languages in the previous list

some of the tasks that you could be asked to perform:

finding a regular expression for a given language
finding a DFA or NFA for a given language
finding a regular expression for an NFA or DFA
converting an NFA into an equivalent DFA, using the algorithm
converting a regular expression into an equivalent NFA, using the algorithm
determining whether a given string is accepted by an NFA or DFA
prove that a language is not regular, using the Pumping Lemma
finding a derivation for a given string from a given CFG
finding a CFG for a given language
finding the language generated by a give CFG