Studying for Exam 2 COMP 455

Pushdown automata:

K: finite set of states
Sig: input alphabet
Γ : stack alphabet
s in K is an initial state
F is set of final states

are a finite automata but they also have finite memory.

((cur state, input read, cur stack top), (state to go to, new stack top))
((p, a, B), (q, theta))

Configurations are current states of the automata, which pretty much are just the condition part of the above statement. You have (cur state, remaining inputs to be read, current entire stack).

$\vdash_M$, means it transitions from one configuration to the next in one step

$\vdash_M^*$ means it transitions from one configuration to the next after zero or more "yields in one step" transitions. That is, essentially, it does a star operation on the yields in one step.

Note: in order for a string to be accepted, it has to start at a start state (s) and end at a final state (f \in F) with an EMPTY STACK. That is, given f \in F, you have to get to the configuration (f, e, e).

Write transitions like so:

$$(s, abbbba, e) \vdash_M (s, bbbba, a) \vdash_M (s, bbba, ba) \vdash_M (s, bba, bba) \vdash_M (f, bba, bba) \vdash_M (f, ba, ba) \vdash_M (f, a, a) \vdash_M (f, e, e).$$

leftmost derivation is when the leftmost nonterminal is replaced. Rightmost is similarly defined. So leftmost would be S => SS => (S)S => ()S = ()(). The left nonterminal is replaced first. A context-free grammar G = (V, Σ, R, S) is ambiguous if there is some string w ∈ Σ such that there are two distinct parse trees T1 and T2 having S at the root and having yield w. Equivalently, w has two or more leftmost derivations, or two or more rightmost derivations.

Minimizing finite automata:

Define equivalence with respect to the language you're dealing with. $x \approx_L y$

Define equivalence with respect to the automaton. $x \sim_M y$.

Iff $\approx_L$ has finitely many equivalence classes, then L is a regular language.

Two states are equivalent in M if the language that each of the states is equal?

Remove equivalent states.

Equivalence: in trying to determine if x and y are equivalent, if you can append something to either of them and only one of them is not in the language anymore, then they are not equivalent.
Equivalence with respect to language: for each char in L, next to it what you can add to the end of it (as a set) and still get something into L. Then read off what's equivalent.

x	$\{z : xz \in L\}$
a	$\{b, c\}$
b	$\{b, c\}$
c	ϕ
ab	$\{\epsilon\}$
ac	$\{\epsilon\}$
ϵ	L

any deterministic finite automaton M recognizing L has to have at least as many states as the number of equivalence classes of the relation $\approx_L$. Also, there's a theorem that says: there is a minimal deterministic finite automaton where the number of the states in the automaton is equal to the number of equivalence classes

A deterministic finite automaton M is minimal if for each pair p, q of states of M, there is a string w ∈ Σ* such that M accepts w starting from p, but M does not accept w starting from q, or vice versa. Also, all states must be reachable from the start state.

To minimize a finite automaton M, we do the following:

1. Compute the equivalence relation $\equiv$ on states of M.
2. Collapse all pairs p, q of states of M such that $p \equiv q$.
3. Delete all states of M that are not reachable from the start state.

Understand Regular, Context-free but not regular, not context free, finite.
Context-free languages > regular languages
Great resource: http://www.quora.com/What-is-an-intuitive-way-to-describe-the-differences-between-context-sensitive-context-free-and-regular-grammars

Regular languages accept only dependent on the state that the automaton is in, nothing else (pointing at the fact that context free/push down automaton require an empty stack to accept). Note that you can get a sense for context free because they may have more conditions than regular languages. For instance, it may have something like “even length string”, which you can kind of see requires some sort of memory for determining whether you should accept at a certain point. Accepting requires some “context”.

More things to look over:

Is a given context free grammar ambiguous? Done.

Context free grammars and closure under certain operations. Need to do.

Things to remember:
How to minimize a finite automata: He said “Figure out what language it recognizes and then figure out a simpler way to write it and then that’ll be minimal.”
Context free grammar is closed under: concatenation, union, klene star , union with regular, intersection with regular.

Production in push down automaton? Um. Stupid easy. It just means what’s pushed/popped in a given state transition.

Pumping Lemma for regex (proof problem):

http://www.cs.unc.edu/~plaisted/comp455/hndout4.pdf
This is to say: If you can choose a string in L of arbitrary length (any n choice) such that if you were to split it up and copy the middle portion a certain number of times, the resulting string would NOT be in L, then you have won and shown that the language is not regular. If the opponent wins, that doesn't necessarily mean the language is regular. You can't use the game to show regularity, just show irregularity.

How to do a parse tree: