CPSC 229, Fall 2008 — Third Test Info

The third test for this course will be given in class on Monday, November 24. It will cover Chapter 3, Sections 2, 4, 5, 6, and 7 and Chapter 4, Sections 1, 2, and 3. Note that Section 3.2, which covers regular expressions on the computer, will not be on the test. Also, the test will only cover as much of Section 4.3 as we manage to get through on Wednesday. (This will certainly ***not*** *include LR(1) parsing and will probably not include LL(1) parsing.) You can expect to be asked to do a proof using the Pumping Lemma for Regular Languages. You should also be ready to apply the algorithms for converting NFAs to DFAs and regular expressions to NFAs. The format of the test will be similar to the first two tests.*

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

regular expression (over an alphabet Σ)

the language generated by a regular expression

regular language (over an alphabet Σ)

finding regular expressions for given languages

finite state automaton

start state

accepting state

transition function

DFA (Deterministic Finite-State Automaton)

Definition of a DFA M as a 5-tuple $M = (Q, \Sigma, \delta, s_o, F)$

state diagram for a DFA

relation of the state diagram to the definition $M = (Q, \Sigma, \delta, s_o, F)$

how a DFA processes an input string

what it means for a DFA to accept an input string

the extended transition function $\delta^*(q, w)$ for a DFA

the language accepted by a DFA, $L(M) = \{w \in \Sigma^* \mid \delta^*(s_o, w) \in F\}$

finding a DFA for a given language, and *vice versa*

nondeterminism

NFA (Nondeterministic Finite-State Automaton)

ε-transitions in an NFA

how an NFA processes an input string

what it means for an NFA to accept an input string

the language $L(N)$ accepted by an NFA N

the algorithm for converting an NFA to a DFA (and *why* it works)

Kleene's algorithm for converting an regular expression to a DFA

the language accepted by a DFA (or NFA) is regular

essential equivalence of regular expressions, NFAs, and DFAs

not every language is regular

Pumping Lemma for Regular Languages

the essential idea in the proof of the Pumping Lemma

using the Pumping Lemma to show that particular languages are not regular

common examples of languages that are and 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

the language generated by a CFG, $L(G) = \{\, w \in \Sigma^* \mid S \Longrightarrow^* w \,\}$

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

BNF (Bacus-Naur Form)

BNF notations: $\langle item \rangle$, ::=, [$\langle items \rangle$], and [$\langle items \rangle$]...

parsing

parse tree

left derivation

ambiguous grammar

possibly the basic idea of LL(1) parsing