The third test of the term will be given in class on Monday, November 19. It will cover all of Chapter 3, except for Section 3.3. (That is, there will be nothing on the test specific to practical regular expressions as they are actually used on the computer.)

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

alphabet

symbol

string over an alphabet Σ

concatenation of strings, denoted xy or $x \cdot y$ for strings x and y

length of a string, denoted $|w|$ for a string w

reverse of a string, denoted w^R for a string w

empty string, denoted ε

$n_\sigma(w)$, the number of times that the symbol σ occurs in the string w

language over an alphabet Σ

the set of all strings over a an alphabet Σ, denoted Σ^*

number of strings over an alphabet Σ

number of languages over an alphabet Σ

operations on languages: $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 \smallsetminus L_2$, LM, L^R, L^*, L^2, L^3, $\ldots$

Kleene closure, L^*, of a language

Kleene star operator

regular expression over an alphabet Σ

ε as a regular expression

the operators $*$, $+$, and concatenation $(\cdot)$ in regular expressions

how $L(r)$ is constructed from r

pattern matching; what it means for a string to match a regular expression

the language generated by a regular expression, denoted $L(r)$ for a regular expression r

regular language

FSA (Finite-State Automaton)

state in an FSA

start state of an FSA

accepting state in an FSA

DFA (Deterministic Finite Automaton *or* Deterministic Finite-state Automaton) 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 $\delta^*\colon Q \times \Sigma^* \to Q$

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

transition diagram [the usual picture] of a DFA

NFA (Non-deterministic Finite Automaton *or* Non-deterministic Finite-state Automaton)

the differences between NFAs and DFAs

ε-transitions

the language, $L(M)$, accepted by an NFA, M

algorithm for converting an NFA to an equivalent DFA

the algorithm for converting a regular expression to an equivalent NFA

the fact that a DFA can be converted to a regular expression (but not an algorithm)

operations ($L_1 \cup L_2$, $L_1 \cap L_2$, LM, L^R, L^*) on regular languages produce regular languages

proof that the union, concatenation, and Kleene star of regular languages is regular [trivial]

proof that the complement of a regular language is regular [harder]

idea of the proof that the intersection of two regular languages is regular [a little harder still]

the pumping lemma

proof of the pumping lemma

using the pumping lemma to show that certain specific languages are not regular

examples of language that are not regular such as:

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

other tasks that you can be asked to perform for certain cases:

giving an English description of a language
finding a regular expression for a given language
finding a DFA for a given language
finding a regular expression for an NFA or DFA