Exam 1 Topics

The following provides a list of the knowledge and skills you should have in order to succeed on the first exam. To prepare for the exam, make sure that you have done all assigned reading, attempted all the homework problems and understood the released solutions, and looked at all lecture notes.

1 Alphabets, Strings, Languages

- Know the definitions of alphabet, string, and language.
- Be able to describe various languages of strings in set-builder notation.
- Understand the effect of concatenation, Kleene star, and union on strings and as operators on languages.
- Be clear about the meanings of $\emptyset$ and $\varepsilon$.

2 Deterministic Finite Automata

- Know the definition of a DFA as a transition diagram or 5-tuple.
- Understand the transition function $\delta$, which needs to be defined for every input symbol in every state.¹
- Understand the essence of DFAs: Finite memory. States represent what needs to be remembered to allow the computation to proceed.²
- Be able to design a DFA to meet a given specification.
- Be able to describe the language recognized by a given DFA.

3 Nondeterministic Finite Automata

- Understand the basic definition of NFAs, both as a transition diagram and a tuple.
- Understand that an NFA accepts whenever there is at least one accepting computation.
- Be able to take advantage of nondeterminism by designing a simple NFA where designing a DFA would be much more difficult.
- Understand use of $\varepsilon$-transitions and the convenience they offer.
- Be able to create a DFA that is equivalent to a given NFA by computing the $\varepsilon$ closure and using the subset-of-states construction.³

4 Regular Expressions

- Understand the recursive definition of regular expressions.
- Know the precedence of the operators in a regular expression (*, concatenation, $\cup$)

¹ Advanced: Understand the extended transition function $\tilde{\delta}$, where $\tilde{\delta}(q,xa) = \delta(\tilde{\delta}(q,x), a)$ for any state $q$, string $x$, and symbol $a$.

² Intuition: Counting for some fixed constant $n$ is possible, but counting and remembering an arbitrary value for later comparison is not.

³ Make sure you understand why these constructions work. We’re making a DFA whose states correspond to sets of possibilities.
• Given a regular expression \( r \), be able to describe the language \( L(r) \) it denotes.
• Given a DFA \( M \), be able to produce a regular expression \( r \) such that \( L(r) = L(M) \), using the state elimination method.
• Given a regular expression \( r \), be able to produce an NFA \( M \) such that \( L(M) = L(r) \).

5 Pumping Lemma

• Understand how (and why) the Pumping Lemma works.\(^4\)
• Be able to use the Pumping Lemma to prove that a given language is not regular.\(^5\)
• Know that the Pumping Lemma is a necessary but not sufficient condition for regular languages. You cannot use the Pumping Lemma to prove a language is regular.

6 Closure Properties of Regular Languages

• Be able to show that regular languages are closed under the following operations: union, concatenation, Kleene star, complement, intersection, reversal.
• Be able to use closure properties to prove that a language is regular.\(^6\)
• Be able to use closure properties to prove that a language \( L \) is not regular, for example, by showing that if \( L \) were regular, by applying closure properties, we could obtain a known non-regular language \( L' \) from \( L \) (and perhaps other regular languages), and therefore we can conclude that since \( L' \) is known to be non-regular, \( L \) is not regular.

\(^4\) This involves understanding that if a string \( w \) is accepted by some \( M \) and \( w \) is as long or longer than the number of states, then \( w \) must "loop" back to some state \( M \) en route to accept. Therefore, we can eliminate (pump 0 times) or repeat (pump 1 or more times, up to infinity) the segment of \( w \) that labels the loop to obtain new accepted strings.

\(^5\) Make sure that your argument does not assume some particular \( n \), or some particular \( x \), \( y \), and \( z \). That is, know that you do not get to choose \( n \) or \( x \), \( y \), \( z \), but that you do get to choose \( w \) and the \( i \) in \( xy^iz \).

\(^6\) E.g., \( L \) is regular if you can find known regular languages \( L_1 \) and \( L_2 \) such that \( L = L_1 - L_2 \) because regular languages are closed under difference.