COMS W3261 Fall 2024 Midterm Review

Yizhi Huang and Owen Terry

October 14, 2024

1 Midterm coverage

The midterm will cover all the materials shown in class up to Lecture 9 (2024-10-03), unless explicitly stated otherwise by the instructor.

- Specifically, for context-free languages (CFL), there will be no deep questions as it never appears in a quiz or homework.

You can use theorems and examples given in class without proof, but you have to explicitly state that they are given in class.

Recall that you can bring 2 double-sided letter-size sheets of paper. So you don't need to memorize things, but it is important to understand the definitions and concepts.

The solution to quizzes and homework are posted on Courseworks.

The TAs have never seen the actual midterm, and will not see it before you do.

2 Summary of the main points

See handouts on the class webpage with more detailed review and sample questions.

2.1 Basic definitions and notations

An alphabet is a *finite* non-empty set of characters.

A string (over an alphabet) is a *finite* sequence of characters in that alphabet.

A language (over an alphabet) is a set of strings in that alphabet. (A language is not necessarily finite.)

For a character or a string x, $k \geq 0$, x^k is the string $x \circ x \circ \cdots \circ x$ where x is repeated k times. In particular, x^0 is the empty string ε. (Here it does not have the meaning "the k-th power of x".)

2.2 Regular Languages

A language is regular if and only if it can be recognized by one of the following:

- A DFA $(Q, \Sigma, \delta, q_0, F)$.

- An NFA $(Q, \Sigma, \delta, q_0, F)$.
- A Regular Expression.

The above three representations are equivalent and we have seen methods of going from one to the other. (Namely, the subset construction for NFA to DFA, the GNFA method with state ripping to go from an NFA to a regular expression, and a recursive transformation of any regular expression to a NFA).

Note that being regular is a property of a language, and we cannot say a string is regular or not regular.

For any DFA/NFA/regexp, there is exactly one language recognized by it, namely the set of all strings the DFA/NFA/regexp accepts. However, for a language, there can be many DFA/NFA/regexp recognizing it.

We have also looked at how to prove a language is regular. Here are some ways to show it:

- Design a DFA/NFA/regexp recognizing the language.

  Note that we need to show not only that every string in the language is accepted by the DFA/NFA/regexp, but also that every string accepted by the DFA/NFA/regexp is in the language (or equivalently, every string not in the language is not accepted (rejected) by the DFA/NFA/regexp).

- Use closure properties: We have proved that regular languages are closed under complement, intersection, union, concatenation, and Kleene star. Thus, to prove that a language L is regular, one method is to show that L can be expressed as a combination of regular languages using these operations (e.g., show that $L = L_1 \cup L_2$ for some regular languages L_1, L_2, or show that $L = \overline{L_1}$ for some regular L_1).

- As a special case of using closure under union, every finite language is regular.

- To prove that some new operation on regular languages results in a regular language (i.e. to prove that regular languages are closed under some new operation), one approach is to start with a DFA or NFA for the original language(s), and then apply some modifications to it, resulting in a new NFA (or DFA) for the resulting language. (This is how we proved many of the closure results we have seen). Other approaches include starting from a regular expression and making modifications to it, or expressing the new operation as a combination of other operations we already know regular languages are closed under (e.g., expressing intersection of two languages as a combination of union and complement of the two languages).

- The Myhill-Nerode theorem can help you construct a minimal DFA for the language (there are also algorithmic ways to minimize any DFA for a language). This is not part of the required material for the midterm.

We have also seen non-regular languages and have shown several approaches to proving that a language is not regular:

- Pumping lemma (showing that pumping lemma does not hold).

  The pumping lemma states that if L is a regular language, then $\exists p > 0$, $\forall w \in L$ such that $|w| \geq p$, $\exists$ strings x, y, z such that $w = xyz$, $|xy| \leq p$, $|y| > 0$ and $\forall i \geq 0$, $xy^i z \in L$.

  If you find this hard to remember, it might be helpful to try to understand the proof idea in Lecture 4 (2024-09-12).

  To prove a language is not regular, we can prove pumping lemma does not hold for that language, namely, $\forall p > 0$, $\exists w \in L$ such that $|w| \geq p$, and $\forall$ strings x, y, z that satisfy $w = xyz$, $|xy| \leq p$ and $|y| > 0$, $\exists i \geq 0$ such that $xy^i z \notin L$.

  Common mistakes:

  - Not constructing a string w such that $|w| \geq p$ for all $p > 0$.
  - Not constructing a string w in L.
  - Not considering every possible way to parse $w = xyz$. (A trick here is to construct w that starts with a^p for some a in the alphabet. I'm not saying you can always do this.)
  - Not picking an i such that $xy^i z \notin L$.

- Using closure properties. To show L is not regular, assume towards contradiction that it is regular, and show how combining L with other languages known to be regular (via operations like the ones mentioned above), will give you a language already known not to be regular. This gives a contradiction, and thus L is not regular.

  For example, if you show that $L \cap L_2 = L_3$ where L_2 is known to be regular and L_3 is known to be non-regular, you can conclude that L must be non-regular.

- Myhill-Nerode theorem (which you may use but is not part of the material): You can show L is not regular by showing that there are infinitely many strings such that for every two of them, x, y, there exists a string z such that exactly one of xz, yz is in L. See e.g. the solution to homework 1.3(d) for an example of using it.

  Myhill-Nerode theorem is a sufficient and necessary condition for regular language. In contrast, pumping lemma is a necessary but not sufficient condition: namely, there exists a non-regular language that satisfies pumping lemma.

2.3 Context Free languages

A language L is context free if and only if it satisfies one of the following equivalent definitions:

- L can be generated by a context-free grammar (CFG).
- L can be recognized by a push-down automaton (PDA), which is like an NFA equipped with a stack.

We mentioned that the class of CFL is closed under the following operations:

- Union
- Concatenation
- Star

However, it is not closed under:

- Intersection
- Complement

Every regular language is a CFL, but the converse is not true.

3 Problem Set: Regular Languages

Design finite automata for the following languages. You can give the DFA/NFA by their transition diagrams.
Note: In your transition diagrams, you can use shorthand notation on the labels of the edges. For example, you can label an edge by $\Sigma \setminus \{a\}$ to indicate that the transition takes place for all input symbols except a. Make sure to specify the starting state and the accepting states in your diagrams.

1. L is the language over the alphabet $\Sigma = \{0, 1, 2\}$ consisting of all strings that:

   - Every 0 is immediately followed by a 1, every 1 is immediately followed by a 2, and every 2 is immediately followed by a 0.
   - The string starts and ends with the same symbol.
   - The string must have length at least 1.

2. The set of strings over the alphabet $\Sigma = \{a, b, \ldots, z\}$ that contain at least one m between any two a's in the string; for example $abc, john, mama, american$ are in the language, but $papa, panamerican$ are not.

For the following problems, if a language L is given, prove that L is regular or prove that L is nonregular.

3. $L = \{ww \mid w \in \{0,1\}^* \text{ and } w \text{ contains at least one 0 and at least one 1}\}$ over the alphabet $\Sigma = \{0,1\}$.

4. $L = \{a^i b^j c^k \mid i + j = k\}$ over the alphabet $\Sigma = \{a, b, c\}$.

5. $L = \{0^k u 1^k \mid k \geq 1, u \in \Sigma^*\}$ over the alphabet $\Sigma = \{0, 1\}$.

6. $L = \{0^k 1 u 1^k \mid k \geq 1, u \in \Sigma^*\}$ over the alphabet $\Sigma = \{0, 1\}$.

7. **Challenge:** L is the language consisting of all strings of a's and b's with an equal number of occurrences of ab and ba as substrings. (The string $aabbbaa$ has one occurrence of each of the substrings ab and ba.)

   There is no need to worry at all if you find this one difficult. It is harder than what we expect to see on the midterm.