This lecture notes are provided as a supplement to the textbook. In the textbook you have read about the pumping lemma for regular languages, a very useful tool to prove that certain languages are not regular. Here we consider a different method, called “diagonalization”, that will be very useful later on in the course. The method involves the construction of a specific language which is not regular almost by definition. The language is not particularly meaningful, I know of no application where you would want to design a finite automaton for this language. The goal of this method is just to establish the existence of some languages which are not regular.

**Encoding regular expressions** For concreteness, let us consider the set of regular languages over a specific alphabet, the binary alphabet \( \{0, 1\} \). We know that a language is regular if and only if it is the language of a regular expression \( R \). Consider the set \( R \) of all regular expressions over the set of basic symbols 0, 1. These regular expressions can be represented as string over the larger alphabet \( \Sigma = \{0, 1, +, \cdot, (,),\ast, \emptyset\} \). For example the set of all binary strings can be represented by the binary expression \( E = (0 + 1)^* \). Since the alphabet \( \Sigma \) has size 8, we may encode its symbols are triplets of bits, just like 8-bit bytes are used to represent symbols from larger alphabets. The way we map the elements of \( \Sigma \) to bits is largely arbitrary, but for concreteness let us consider a specific encoding mapping \( \phi: 0 \mapsto 000, 1 \mapsto 001, + \mapsto 010, \cdot \mapsto 011, (\mapsto 100, ) \mapsto 101, \ast \mapsto 110 \) and \( \emptyset \mapsto 111 \). Using this encoding, regular expressions can be also represented as binary strings, e.g., \( \phi(E) = 10000010001101110 \). Of course, every binary string is the representation of a regular expression, just like not every string over \( \Sigma \) is a syntactically valid regular expression. But we may consider binary languages corresponding to specific sets of regular expressions, as we do next. Notice that each regular expression \( E \in \mathcal{R} \) is represented by a single binary string \( \phi(E) \in \{0, 1\}^* \), and it also represents a language \( \mathcal{L}(E) \subseteq \{0, 1\}^* \), i.e., a set of binary strings. There is nothing special, or to be confused about. This is just the same as a computer program being represented by a string (possibly including special “new line” characters to make the string more readable), and the same program representing a set of strings, e.g., the set of input strings for which the program outputs 1.

**A nonregular language** Let \( L \) be the set of all binary strings of the form \( \phi(E) \) where \( E \in \mathcal{R} \) is a binary regular expression such that \( \phi(E) \notin \mathcal{L}(E) \). We claim that this language is not regular. In fact, the proof is very simple, as the language \( L \) was defined with the specific goal of not being regular. Here is the proof:

**Theorem:** The language \( L = \{ \phi(E) \in \mathcal{R} \mid \phi(E) \notin \mathcal{L}(E) \} \) is not regular.

**Proof:** Assume for contradiction that \( L \) is regular. Since \( L \subseteq \{0, 1\}^* \) is a binary language, there is a regular expression \( E \in \mathcal{R} \) such that \( \mathcal{L}(E) = L \). Now consider the following question: \( \phi(E) \in L \)? i.e., does the string \( w = \phi(E) \) belongs to the set \( L \). We do not know the answer to this question, but sure the answer must be either “yes” or “no”. We will show that in either case we get a contradiction: \( w \in L \) if and only if \( w \notin L \). This is proved by a chain of implications:

- By definition of \( L \), we have \( w \in L \) if and only if \( w = \phi(E') \) for some regular expression \( E' \in \mathcal{R} \) such that \( \phi(E') \notin \mathcal{L}(E') \)

- Since the function \( \phi \) is injective, and recalling that \( w = \phi(E) \), the condition \( w = \phi(E') \) is satisfied if and only if \( E = E' \).

- It follows that \( w \in L \) if and only if the string \( w = \phi(E) = \phi(E') \) is in \( \mathcal{L}(E') = \mathcal{L}(E) = L \)

This proves that \( w \in L \) if and only if \( w \notin L \). This is a contradiction. So, our contradiction hypothesis must be false and \( L \) is not regular. \( \square \)