

CS 6353 Compiler Construction, Homework #1

- Write regular expressions for the following informally described languages:
 - All strings of 0's and 1's with the subsequence 011.
 - All strings of 0's and 1's with the substring 00*1.
- Consider $\Sigma = \{a, b\}$. Answer the following DFA related questions. When constructing DFA, there is no need to show your construction steps, but you need to informally state how you get the DFAs.
 - Construct a DFA that accepts $(a|b)^*$ except for aabb.
 - Construct a DFA that accepts $(a|b)^*$ except for b^*a^* .
 - Based on the techniques you use in (a) and (b), can you come up with a DFA construction algorithm for the "except for" type of languages?
- Consider the regular expression $aac^* | b(a|b)c^*$ defined on $\Sigma = \{a, b, c\}$.
 - Construct the NFA for the regular expression. You can directly draw the NFA without going through the RE-to-NFA steps.
 - Convert the NFA to DFA. You need to show the conversion steps.
 - Minimize the DFA. You need to show the minimization steps.
- A token recognizer is designed to handle the following tokens, where $\Sigma = \{a, b, c, d\}$.
 - T1 = aab
 - T2 = bcc*
 - T3 = bcd*
 - T4 = aabcd*
 - Construct a minimized DFA for token recognition and the tokens are defined as follows. The DFA should specify the specific token names (T1, T2, ...) it accepts at the corresponding final states. You do not need to show the steps for the construction if you can draw the DFA directly. Note that the longest matching and first matching rules for ambiguity resolution should be used.
 - Execute your DFA to process the following string to identify tokens. List every token string and its token name. Also, describe all the backtracking actions taken during the process.
aabaabccdddbcbcccaabbccddd
 - Create the lex definition file for the token recognizer. Use lex to generate the scanner program from the definition file. After compiling the scanner, generate input strings to test it. Attach the print out for the definition file, the input string, and the output of the scanner.
- Create a lex definition file for recognizing the following sets of tokens. After compiling the scanners, generate input strings to test them. Attach the print out for the definition files, the input strings, and the output of the scanner. Please also state in your answer what you have learnt from the results.
 - Consider $\Sigma = \{a, b, c, d\}$.
 - T1 = aab /ccc
 - T2 = aab /ddd
 - T3 = c*
 - T4 = d*
 - T5 = all other strings(You need to define the lookahead strings as specified.)
 - Consider $\Sigma = \{a, b, c\}$.
 - T1 = abcc*

$$T2 = ca$$

$$T3 = cb^*$$

6. Consider the following grammar defined over $\Sigma = \{0, 1\}$.
- $$S \rightarrow 0S11$$
- $$S \rightarrow S1$$
- $$S \rightarrow \epsilon$$
- Briefly describe the language generated by this grammar.
 - Show that this grammar is ambiguous by giving a string that can be parsed in two different ways and showing the two corresponding parse trees.
 - Rewrite the grammar to eliminate the ambiguity.
7. Let L be a language defined over $\Sigma = \{a, b\}$ and L consists of all strings with the same number of a 's and b 's, Give a context free grammar for L .
8. Construct a regular grammar for the language L , where L accepts $(a|b)^*$ except for b^*a^* (check your answer for 2 for hints).
9. Construct a type-0/1 grammar for the language $\delta c \delta$, where δ can be any string of a 's and b 's. Use some examples to illustrate how your grammar would work.
10. Consider the language: All strings of 0's and 1's such that every 0 is immediately followed by at least one 1.
- Construct a DFA for the language and convert it to a grammar (following the conversion algorithm).
 - As we discussed in class, the grammar you constructed in (a) is a regular grammar (type 3 grammar). The following grammar is also for the same language. Show a parse tree for the string 1011101.

$$S \rightarrow BA$$

$$A \rightarrow 01BA \mid \epsilon$$

$$B \rightarrow 1B \mid \epsilon$$
 - Give the leftmost derivation for (b).
 - Give the rightmost derivation for (b).
 - How many different derivations can the string 1011101 has? Briefly justify your answer.