

# CS385-Assignment 3

Due Tuesday, September 25

I've organized these problems according to theme, rather than according to the order in which they occur in the book or their difficulty. Problems 1.43 and 1.61 are more difficult, since they ask you to devise general proofs.

## 1 Understanding Nondeterminism

Exercise 1.6(b,d,e,g,h). Just draw the state diagrams.

Problem 1.60. We used the example  $C_3$  in class, so you should already know how to write down the state-transition diagram. Pay attention here to writing a correct formal description of the NFA: That is, write out exactly what  $Q$ ,  $\Sigma$ ,  $\delta$  and  $F$  are for a  $(k + 1)$ -state NFA that recognizes  $C_k$ .

## 2 Understanding the Relation Between Determinism and Nondeterminism

Exercise 1.16, both parts. This, of course, is an exercise in applying the algorithm for converting an NFA into an equivalent DFA.

Problem 1.61. Here is the point that this problem is trying to make: When you convert an NFA into an equivalent DFA, there is the risk of an exponential blowup in the number of states, since each state of the DFA is a *set* of states of the original NFA, and there are  $2^n$  subsets of an  $n$ -element set. In practice, you usually find that the DFA you construct using this algorithm has many fewer states than the maximum possible, and that the resulting DFA can usually be simplified to one with still fewer states. But that is not always the case.

So how do you prove that any DFA for  $C_k$  has at least  $2^k$  states? Suppose that  $v_1$  and  $v_2$  are two different strings of length  $k$ , and let  $q_1$  and  $q_2$  be the states that you arrive at beginning at the initial state after reading  $v_1$  and  $v_2$  respectively. Show that  $q_1$  and  $q_2$  must be different states. How does this imply the desired result?

### 3 Understanding Closure Operations

Problem 1.43. Comments: Problems 1.40-1.43 are all designed to show that the class of regular languages is closed under various unary and binary operations on languages. (A *binary operation* is one with two operands, like union, intersection, and concatenation; a *unary operation* has only one operand, like complementation and star). The basic structure for such proofs is fairly uniform: start with an automata for the operands, and use it to construct an automaton for the language formed by application of the operations. Sometimes the original automata and the new automaton have to be deterministic, as with the constructions for the intersection and complement, but often the construction of the new automaton is greatly simplified by the use of nondeterminism, as in the picture proofs of Theorems 1.45, 1.47 and 1.49. For Problem 1.43, use an NFA that guesses if the current position is where the dropout letter lives.

### 4 Understanding Regular Expressions

Exercises 1.17, 1.18 and 1.20.