4CS4-6: Theory of Computation (Minimization of Finite Automata)

Lecture 10: Feb. 12, 2019

Prof. K.R. Chowdhary *: Professor of CS*

10.1 Introduction

The deterministic and nondeterministic finite automata are abstract models of machines used for capturing behavior of systems. Because both these approaches have different time and space complexity, therefore each has advantages over the other.

Each Deterministic Finite Automata defines a unique language. However the reverse is not true. For a language, there may be more than one representation of finite automata having varying number of states. Higher is number of states in a finite automaton, higher is the memory requirements, as well as more complexity in its implementation. Therefore, it is desirable that a DFA has a minimum possible number of states.

The regular expression is usually converted into a FA; in most cases it is deterministic finite automaton. This is usually achieved by first constructing a non-deterministic automaton accepting the language denoted by the expression and then applying the *subset construction*[1] to it, possibly reducing the resulting deterministic automaton afterwards. A closer look at the algorithms involved in this process reveal that as far as its computational complexity is concerned, the most critical step is the construction of a nondeterministic automaton. If we can construct a nondeterministic automaton with $|Q| = n$ states corresponding to a given regular expression, then the equivalent deterministic automaton may have up to 2^n states. Since, reducing of an m-state deterministic automaton is of time complexity[2] $O(m \ \log m)$, the complete process may therefore be of time complexity $O(n2^n)$. However, suppose that in the first step we obtain a nondeterministic with $2n$ states for the same expression, then the complexity of the complete process can be as bad as $O(2 * 2^{2n})$, which is almost square of the previous complexity. Thus, it is very important that to construct a nondeterminsitc automaton as small as possible for any given regular expression.

Suppose that in the first step we obtain an NFA with $2n$ states for the same expression, then, the complexity of the complete process can be as bad as $O(n*2^{2n})$, which is almost the square of the previous complexity. Therefore, it is highly desirable that NFA be constructed with as small number of states as possible.

The state minimization problem for an NFA is *computationally-hard* ($PSPACE$-complete). The worst case complexity for the same problem for DFA is $O(|Q||\Sigma| \log |Q|)$. The Minimiza-

[1] For n states, all possible subsets can be as many as 2^n.

[2] If there is m state m-state DFA, there can be at most 2^m subsets of these states. Hence, if there are m-subsets, of certain for a given number of states, those given states can be at the most $m \ \log m$.

tion (or optimization) of a deterministic finite automaton refers to detecting those states whose absence in the DFA does not affect the language acceptability of the automata. Hence, these states can be eliminated. A reduced automaton should consume lesser memory and complexity of the algorithm implementing this automaton is reduced.

10.2 Formalism for minimization

Based on the concept of minimization applied in the above example, we arrive at some definitions, which will be useful in analyzing the DFAs in point of view of their minimization.

Definition 10.1 Unreachable states. *These are the states, which are not reachable for any sequence of input. In general, if q_0 is start state then state q' is an unreachable state if there does not exist any $w \in \Sigma^*$ such that $\delta^*(q_0, w) = q'$.*

Definition 10.2 Dead states. *A dead state is non-final state from which transition terminates to itself on every input. That is, for every $a \in \Sigma$, q is dead state if $\delta(q, a) = q$, and $q \in Q - F$.*

Definition 10.3 Reachability Relation. *A state $q \in Q$ of an automaton M is accessible if there exists some string $w \in \Sigma^*$ such that when w is completely read, M is in state q. For q_0 as start state, the reachability relation is $(q_0, w) \vdash^* (q, \varepsilon)$.*

Definition 10.4 Indistinguishable states. *Two states in a DFA are indistinguishable if their behavior is indistinguishable for all the inputs. Thus, p and q are indistinguishable states if for any $w \in \Sigma^*$, $\delta^*(p, w) = \delta^*(q, w) = r, r \in Q$. If p and q are not indistinguishable then they are distinguishable states.*

Definition 10.5 Distinguishable states. *A pair p, q is distinguishable if with input a, the transitions from both p, q goes to p', q', and p', q' is already marked as a pair of distinguishable states.*

From the above discussions we get intuition that the process for minimization of a DFA should consists:

1. removing of inaccessible (unreachable) states, and
2. finding and merging each set of indistinguishable states.

The reasoning in (2) above leads to the following process: Start with an automaton M (without unreachable states). If M has indistinguishable states p, q, combine them into one state. (For instance, remove q and reroute all transitions into q to go into p instead.) Repeat this process until no more indistinguishable states can be found. At this point we will not be able to reduce M further.

10.3 Table-filling Algorithm

Assume that DFA $M = (Q, \Sigma, \delta, q_0, F)$ is represented by a directed graph. First of all remove all the states that cannot be reached from q_0, along with corresponding transitions. Use the DFS (depth first search) to mark all the reachable states. The unreachable states if any are then those that are unmarked, and can be easily eliminated.

The table-filling algorithm make use of a table of rows and columns as states (p, q), with no junction as $p = q$, to mark the distinguishable states. The heart of the algorithm is as follows: the set $Q - F$ and F are disjoint, thus pairs of the states $(Q - F) \times F$ are distinguishable, hence marked as "$\times$" in the start. For the junction of pairs of states (p, q) and (r, s) in the table, where $\delta(p, a) = r$ and $\delta(q, a) = s$, if r is already distinguishable from s, then p is marked distinguishable from q. Therefore, junction of the state pair (p, q) is marked with "$\times$". The major steps of this algorithm in simple language are as follows:

1. Remove unreachable states.
2. Initialization step. For $p \in F$ and $q \in Q - F$ put a cross-mark "$\times$" in the table at the junction (p, q). This shows that p and q are distinguishable in the minimum machine M'.
3. Follow the recursion rule by applying input $a \in \Sigma$ to each state (r, s). If we have $\delta(p, a) = r$, and $\delta(q, a) = s$, and if (r, s) were earlier marked as distinguishable, mark position (p, q) as distinguishable by symbol "$\times$" in the table.
4. Group the table states into equivalent classes. Each unmarked set of states in a (row, column) of table is an equivalent class. Group them as single state in reduced automata.

The following example demonstrates the use of table-filling algorithm to minimize a DFA.

Example 10.6 *Minimize the DFA shown in the Fig. 10.1.*

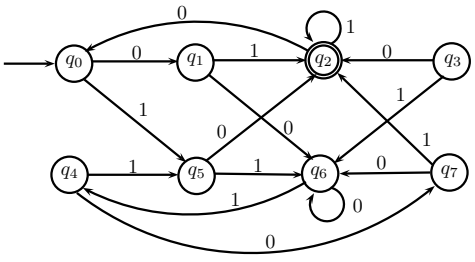

Figure 10.1: Finite automata to be minimized

Applying the table filling algorithm the DFA shown in Fig. 10.1 can be minimized through following steps.

Step 1. The only unreachable state q_3 is removed first of all.

Step 2. Next we make the pairs of final and non-final states, setting the table shown below. This shows the distinguishable state pairs.

q_1						
q_2	X	X				
q_4			X			
q_5			X			
q_6			X			
q_7			X			
	q_0	q_1	q_2	q_4	q_5	q_6

Step 3. In the first iteration we examine all unmarked pairs. For example for pair q_0, q_1, we get $\delta(q_0, 1) = q_5$ and $\delta(q_1, 1) = q_2$, and the pair q_2, q_5 is marked distinguishable, so we mark q_0, q_1 as distinguishable. After doing it for all pairs, we get the following table.

q_1	X					
q_2	X	X				
q_4		X	X			
q_5	X	X	X	X		
q_6		X	X		X	
q_7	X		X	X	X	X
	q_0	q_1	q_2	q_4	q_5	q_6

In the next iteration we examine the remaining pairs. For example, we will mark pair q_0, q_6 because $\delta(q_0, 1) = q_5$ and $\delta(q_6, 1) = q_4$. And the pair q_4, q_5 is already marked. When this is done, the following table results.

q_1	X					
q_2	X	X				
q_4		X	X			
q_5	X	X	X	X		
q_6	X	X	X	X	X	
q_7	X		X	X	X	X
	q_0	q_1	q_2	q_4	q_5	q_6

Step 4. We now group the states into equivalent classes. Since q_0, q_4 are equivalent (not distinguishable) and q_1, q_7 are equivalent, the groups are: $\{q_0, q_4\}$, $\{q_1, q_7\}$, $\{q_2\}$, $\{q_5\}$, $\{q_6\}$. The minimal automaton obtained is shown in figure 10.2.

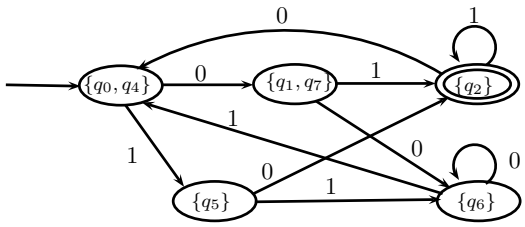

Figure 10.2: Minimized Finite automata of Fig. 10.1