CIS 511: Theory of Computation **Jan 17, 2017**

Lecture 2:

Professor Sampath Kannan *Zach Schutzman*

NB: *These notes are from CIS511 at Penn. The course followed Michael Sipser's Introduction to the Theory of Computation (3ed) text.*

2.1 More Finite Automata

NFAs are Equivalent to DFAs

Recall: every DFA recognizes some (regular) language, a language is regular if there exists some DFA recognizing it. NFAs also recognize regular languages (every DFA is also an NFA).

Question: are NFAs more powerful than DFAs? They are certainly a broader class of machines, given that the set of all DFAs is a propers subset of the set of all NFAs.

Theorem 2.1 *NFAs recognize exactly the class of regular languages.*

Proof: *(A rough sketch)*

Consider an NFA M without ϵ-transitions. We can imagine a string's traversal as maintaining a set of possible states you are in, and the NFA accepts if and only that set contains a final state at the conclusion of reading the string. Let's consider the set of states at each step.

Create a DFA M' with states corresponding to subsets of states of the NFA. Now, add transitions in M' corresponding to the set of possible transitions in M. Formally, let $S \subseteq Q$, then $\delta'(S, a) = \bigcup_{q \in S} \delta(q, a)$. These subsets S correspond to states in M'.

The start state of M', q_0' is the state corresponding to $\{q_0\}$. The final states of M', F' are the subsets of Q containing at least one element of F.

We can deal with ϵ-transitions by extending δ' to also include all states reachable from reading a and a following ϵ. Formally, define $E(q)$ as the set of states reachable from q without consuming an input character (i.e. 0 or more ϵ-transitions). Then make $\delta'(S, a) = \bigcup_{p \in E(q) \ \forall \ q \in S} \delta(p, a)$.

■

Closure Properties

Theorem 2.2 *If L_1 and L_2 are regular languages, then so is $L_1 \cup L_2$.*

Proof:

Let M_1 and M_2 be DFAs that recognize L_1 and L_2, respectively. Add a new start state and add an ϵ-transition to the start states of M_1 and M_2. This NFA now accepts exactly the strings in either L_1 or L_2 (or both).

Since this is an NFA recognizing it, $L_1 \cup L_2$ is regular.

■

Theorem 2.3 *If L_1 and L_2 are regular languages, then so is their concatenation, denoted L_1L_2 or $L_1 \circ L_2$.*

Proof:

Let M_1 and M_2 be DFAs that recognize L_1 and L_2, respectively. Add ϵ-transitions from the final states of M_1 to the start state of M_2. The new start state is the original start state of M_1 and the final states are those from M_2. This NFA now non-deterministically tries to split the string into its L_1 and L_2 parts.

Since this is an NFA recognizing it, $L_1 \circ L_2$ is regular.

■

Theorem 2.4 *The Kleene Star of a regular language L, denoted L^*, is regular.*

The Kleene Star is a generalization of concatenation. L^* is the set of strings that are an arbitrary (finite) number of concatenations of L with itself (including zero, i.e. $\epsilon \in L^*$ for all L).

Proof:

Let M be a DFA recognizing L. Add a new start state which is also a final state, and add an ϵ-transition to the original start state. Then, add ϵ-transitions from all of the original final states to the new start state.

Since this is an NFA recognizing it, L^* is regular.

■

Theorem 2.5 *The complement of a regular language L, denoted $\bar{L}$ or L^c is regular.*

Proof: *Given a DFA M recognizing L, make all the accept states non-final and all non-final states final. This is now a DFA recognizing L^c.* ■

Theorem 2.6 *If L_1 and L_2 are regular languages, then so is $L_1 \cap L_2$.*

Proof: *This follows directly from the proofs of closure under union and complement and applying DeMorgan's Laws.*

■

Regular Expressions

We are going to define regular expressions inductively.

$\emptyset$ is a regular expression

ϵ is a regular expression

For each $a \in \Sigma$, a is a regular expression

If r_1, r_2 are regular expressions, so are r_1r_2, $r_1 \cup r_2$, r_1^*

Theorem 2.7 *A language L is regular if and only if it is described by some regular expression.*

Proof:

The first direction is easy. We can make NFAs accepting nothing, ϵ, and any single character, and we have already shown the construction for NFAs for the three operations. Therefore, any regular expression can be converted into an NFA by this inductive construction.

To see that any DFA can be converted into a regular expression, we will use an inductive construction. We are going to transform the DFA by merging states and associating them with simpler regular expressions.

Let L be our regular language and M a DFA accepting it. Let's let $Q = [n]$, such that the states are numbered, in order to keep better track of them.

Define $R_{i,j}^{(k)} := \{x | x \text{ takes } M \text{ from } i \text{ to } j \text{ without passing through any state labelled greater than } k\}$

We know $R_{i,j}^{(0)} = \{a | \delta(i,a) = j\}$ if $i \neq j$. $R_{i,j}^{(0)} = \{a | \delta(i,a) = j\} \cup \{\epsilon\}$ if $i = j$. Let's proceed inductively. Suppose we know $R_{i,j}^{(k)}$ for all i, j. To compute $R_{i,j}^{(k+1)}$, observe that this is a superset of $R_{i,j}^{(k)}$, and it also contains those strings passing through $(k+1)$ at least once. The strings that go from (i) to $(k+1)$ are captured by $R_{i,k+1}^{(k)}$, the strings that go from $(k+1)$ to another occurence of $(k+1)$ are captured by $(R_{k+1,k+1}^{(k)})^*$ and the strings from $(k+1)$ to (j) are captured by $R_{k+1,j}^{(k)}$. So $R_{i,j}^{(k+1)}$ is $R_{i,j}^{(k)}$ cup$R_{i,k+1}^{(k)}(R_{k+1,k+1}^{(k)})^* R_{k+1,j}^{(k)}$

■