Theorem 2.11: If $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ is the DFA constructed from NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ by the subset construction, then $L(D) = L(N)$.

PROOF: What we actually prove first, by induction on $|w|$, is that

$$\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)$$

Notice that each of the $\hat{\delta}$ functions returns a set of states from Q_N, but $\hat{\delta}_D$ interprets this set as one of the states of Q_D (which is the power set of Q_N), while $\hat{\delta}_N$ interprets this set as a subset of Q_N.

BASIS: Let $|w| = 0$; that is, $w = \epsilon$. By the basis definitions of $\hat{\delta}$ for DFA's and NFA's, both $\hat{\delta}_D(\{q_0\}, \epsilon)$ and $\hat{\delta}_N(q_0, \epsilon)$ are $\{q_0\}$.

INDUCTION: Let w be of length $n + 1$, and assume the statement for length n. Break w up as $w = xa$, where a is the final symbol of w. By the inductive hypothesis, $\hat{\delta}_D(\{q_0\}, x) = \hat{\delta}_N(q_0, x)$. Let both these sets of N's states be $\{p_1, p_2, \ldots, p_k\}$.

The inductive part of the definition of $\hat{\delta}$ for NFA's tells us

$$\hat{\delta}_N(q_0, w) = \bigcup_{i=1}^{k} \delta_N(p_i, a) \tag{2.2}$$

The subset construction, on the other hand, tells us that

$$\delta_D(\{p_1, p_2, \ldots, p_k\}, a) = \bigcup_{i=1}^{k} \delta_N(p_i, a) \tag{2.3}$$

Now, let us use (2.3) and the fact that $\hat{\delta}_D(\{q_0\}, x) = \{p_1, p_2, \ldots, p_k\}$ in the inductive part of the definition of $\hat{\delta}$ for DFA's:

Theorem 2.12: A language L is accepted by some DFA if and only if L is accepted by some NFA.

PROOF: (If) The "if" part is the subset construction and Theorem 2.11.

(Only-if) This part is easy; we have only to convert a DFA into an identical NFA. Put intuitively, if we have the transition diagram for a DFA, we can also interpret it as the transition diagram of an NFA, which happens to have exactly one choice of transition in any situation. More formally, let $D = (Q, \Sigma, \delta_D, q_0, F)$ be a DFA. Define $N = (Q, \Sigma, \delta_N, q_0, F)$ to be the equivalent NFA, where δ_N is defined by the rule:

- If $\delta_D(q, a) = p$, then $\delta_N(q, a) = \{p\}$.

It is then easy to show by induction on $|w|$, that if $\hat{\delta}_D(q_0, w) = p$ then

$$\hat{\delta}_N(q_0, w) = \{p\}$$

We leave the proof to the reader. As a consequence, w is accepted by D if and only if it is accepted by N; i.e., $L(D) = L(N)$. $\Box$

Theorem 2.22: A language L is accepted by some ϵ-NFA if and only if L is accepted by some DFA.

PROOF: (If) This direction is easy. Suppose $L = L(D)$ for some DFA. Turn D into an ϵ-NFA E by adding transitions $\delta(q, \epsilon) = \emptyset$ for all states q of D. Technically, we must also convert the transitions of D on input symbols, e.g., $\delta_D(q, a) = p$ into an NFA-transition to the set containing only p, that is $\delta_E(q, a) = \{p\}$. Thus, the transitions of E and D are the same, but E explicitly states that there are no transitions out of any state on ϵ.

(Only-if) Let $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ be an ϵ-NFA. Apply the modified subset construction described above to produce the DFA

$$D = (Q_D, \Sigma, \delta_D, q_D, F_D)$$

We need to show that $L(D) = L(E)$, and we do so by showing that the extended transition functions of E and D are the same. Formally, we show $\hat{\delta}_E(q_0, w) = \hat{\delta}_D(q_D, w)$ by induction on the length of w.

BASIS: If $|w| = 0$, then $w = \epsilon$. We know $\hat{\delta}_E(q_0, \epsilon) = \text{ECLOSE}(q_0)$. We also know that $q_D = \text{ECLOSE}(q_0)$, because that is how the start state of D is defined. Finally, for a DFA, we know that $\hat{\delta}(p, \epsilon) = p$ for any state p, so in particular, $\hat{\delta}_D(q_D, \epsilon) = \text{ECLOSE}(q_0)$. We have thus proved that $\hat{\delta}_E(q_0, \epsilon) = \hat{\delta}_D(q_D, \epsilon)$.

INDUCTION: Suppose $w = xa$, where a is the final symbol of w, and assume that the statement holds for x. That is, $\hat{\delta}_E(q_0, x) = \hat{\delta}_D(q_D, x)$. Let both these sets of states be $\{p_1, p_2, \ldots, p_k\}$.

By the definition of $\hat{\delta}$ for ϵ-NFA's, we compute $\hat{\delta}_E(q_0, w)$ by:

1. Let $\{r_1, r_2, \ldots, r_m\}$ be $\bigcup_{i=1}^{k} \delta_E(p_i, a)$.

2. Then $\hat{\delta}_E(q_0, w) = \text{ECLOSE}(\{r_1, r_2, \ldots, r_m\})$.

If we examine the construction of DFA D in the modified subset construction above, we see that $\delta_D(\{p_1, p_2, \ldots, p_k\}, a)$ is constructed by the same two steps (1) and (2) above. Thus, $\hat{\delta}_D(q_D, w)$, which is $\delta_D(\{p_1, p_2, \ldots, p_k\}, a)$ is the same set as $\hat{\delta}_E(q_0, w)$. We have now proved that $\hat{\delta}_E(q_0, w) = \hat{\delta}_D(q_D, w)$ and completed the inductive part. $\square$

$$\hat{\delta}_D(\{q_0\}, w) = \delta_D\big(\hat{\delta}_D(\{q_0\}, x), a\big) = \delta_D(\{p_1, p_2, \ldots, p_k\}, a) = \bigcup_{i=1}^{k} \delta_N(p_i, a) \tag{2.4}$$

Thus, Equations (2.2) and (2.4) demonstrate that $\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)$. When we observe that D and N both accept w if and only if $\hat{\delta}_D(\{q_0\}, w)$ or $\hat{\delta}_N(q_0, w)$, respectively, contain a state in F_N, we have a complete proof that $L(D) = L(N)$. $\square$

Theorem 3.13: Let E be a regular expression with variables $L_1, L_2, \ldots, L_m$. Form concrete regular expression C by replacing each occurrence of L_i by the symbol a_i, for $i = 1, 2, \ldots, m$. Then for any languages $L_1, L_2, \ldots, L_m$, every string w in $L(E)$ can be written $w = w_1 w_2 \cdots w_k$, where each w_i is in one of the languages, say L_{j_i}, and the string $a_{j_1} a_{j_2} \cdots a_{j_k}$ is in the language $L(C)$. Less formally, we can construct $L(E)$ by starting with each string in $L(C)$, say $a_{j_1} a_{j_2} \cdots a_{j_k}$, and substituting for each of the a_{j_i}'s any string from the corresponding language L_{j_i}.

PROOF: The proof is a structural induction on the expression E.

BASIS: The basis cases are where E is ϵ, $\emptyset$, or a variable L. In the first two cases, there is nothing to prove, since the concrete expression C is the same as E. If E is a variable L, then $L(E) = L$. The concrete expression C is just $\mathbf{a}$, where a is the symbol corresponding to L. Thus, $L(C) = \{a\}$. If we substitute any string in L for the symbol a in this one string, we get the language L, which is also $L(E)$.

INDUCTION: There are three cases, depending on the final operator of E. First, suppose that $E = F + G$; i.e., a union is the final operator. Let C and D be the concrete expressions formed from F and G, respectively, by substituting concrete symbols for the language-variables in these expressions. Note that the same symbol must be substituted for all occurrences of the same variable, in both F and G. Then the concrete expression that we get from E is $C + D$, and $L(C + D) = L(C) + L(D)$.

Suppose that w is a string in $L(E)$, when the language variables of E are replaced by specific languages. Then w is in either $L(F)$ or $L(G)$. By the inductive hypothesis, w is obtained by starting with a concrete string in $L(C)$ or $L(D)$, respectively, and substituting for the symbols strings in the corresponding languages. Thus, in either case, the string w can be constructed by starting with a concrete string in $L(C + D)$, and making the same substitutions of strings for symbols.

We must also consider the cases where E is FG or F^*. However, the arguments are similar to the union case above, and we leave them for you to complete. $\square$

Theorem 3.14: The above test correctly identifies the true laws for regular expressions.

PROOF: We shall show that $L(E) = L(F)$ for any languages in place of the variables of E and F if and only if $L(C) = L(D)$.

(Only-if) Suppose $L(E) = L(F)$ for all choices of languages for the variables. In particular, choose for every variable L the concrete symbol a that replaces L in expressions C and D. Then for this choice, $L(C) = L(E)$, and $L(D) = L(F)$. Since $L(E) = L(F)$ is given, it follows that $L(C) = L(D)$.

(If) Suppose $L(C) = L(D)$. By Theorem 3.13, $L(E)$ and $L(F)$ are each constructed by replacing the concrete symbols of strings in $L(C)$ and $L(D)$, respectively, by strings in the languages that correspond to those symbols. If the strings of $L(C)$ and $L(D)$ are the same, then the two languages constructed in this manner will also be the same; that is, $L(E) = L(F)$. $\square$

Theorem 3.4: If $L = L(A)$ for some DFA A, then there is a regular expression R such that $L = L(R)$.

PROOF: Let us suppose that A's states are $\{1, 2, \ldots, n\}$ for some integer n. No matter what the states of A actually are, there will be n of them for some finite n, and by renaming the states, we can refer to the states in this manner, as if they were the first n positive integers. Our first, and most difficult, task is to construct a collection of regular expressions that describe progressively broader sets of paths in the transition diagram of A.

Let us use $R_{ij}^{(k)}$ as the name of a regular expression whose language is the set of strings w such that w is the label of a path from state i to state j in A, and that path has no intermediate node whose number is greater than k. Note that the beginning and end points of the path are not "intermediate," so there is no constraint that i and/or j be less than or equal to k.

Figure 3.2 suggests the requirement on the paths represented by $R_{ij}^{(k)}$. There, the vertical dimension represents the state, from 1 at the bottom to n at the top, and the horizontal dimension represents travel along the path. Notice that in this diagram we have shown both i and j to be greater than k, but either or both could be k or less. Also notice that the path passes through node k twice, but never goes through a state higher than k, except at the endpoints.

To construct the expressions $R_{ij}^{(k)}$, we use the following inductive definition, starting at $k = 0$ and finally reaching $k = n$. Notice that when $k = n$, there is no restriction at all on the paths represented, since there *are* no states greater than n.

BASIS: The basis is $k = 0$. Since all states are numbered 1 or above, the restriction on paths is that the path must have no intermediate states at all. There are only two kinds of paths that meet such a condition:

1. An arc from node (state) i to node j.

2. A path of length 0 that consists of only some node i.

If $i \neq j$, then only case (1) is possible. We must examine the DFA A and find those input symbols a such that there is a transition from state i to state j on symbol a.

a) If there is no such symbol a, then $R_{ij}^{(0)} = \emptyset$.

b) If there is exactly one such symbol a, then $R_{ij}^{(0)} = \mathbf{a}$.

c) If there are symbols $a_1, a_2, \ldots, a_k$ that label arcs from state i to state j, then $R_{ij}^{(0)} = \mathbf{a}_1 + \mathbf{a}_2 + \cdots + \mathbf{a}_k$.

However, if $i = j$, then the legal paths are the path of length 0 and all loops from i to itself. The path of length 0 is represented by the regular expression ϵ, since that path has no symbols along it. Thus, we add ϵ to the various expressions devised in (a) through (c) above. That is, in case (a) [no symbol a] the expression becomes ϵ, in case (b) [one symbol a] the expression becomes $\epsilon+\mathbf{a}$, and in case (c) [multiple symbols] the expression becomes $\epsilon + \mathbf{a}_1 + \mathbf{a}_2 + \cdots + \mathbf{a}_k$.

INDUCTION: Suppose there is a path from state i to state j that goes through no state higher than k. There are two possible cases to consider:

1. The path does not go through state k at all. In this case, the label of the path is in the language of $R_{ij}^{(k-1)}$.

2. The path goes through state k at least once. Then we can break the path into several pieces, as suggested by Fig. 3.3. The first goes from state i to state k without passing through k, the last piece goes from k to j without passing through k, and all the pieces in the middle go from k to itself, without passing through k. Note that if the path goes through state k only once, then there are no "middle" pieces, just a path from i to k and a path from k to j. The set of labels for all paths of this type is represented by the regular expression $R_{ik}^{(k-1)}(R_{kk}^{(k-1)})^*R_{kj}^{(k-1)}$. That is, the first expression represents the part of the path that gets to state k the first time, the second represents the portion that goes from k to itself, zero times, once, or more than once, and the third expression represents the part of the path that leaves k for the last time and goes to state j.

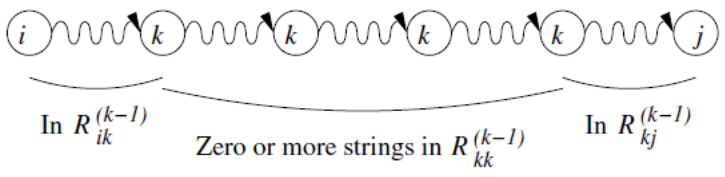

Figure 3.3: A path from i to j can be broken into segments at each point where it goes through state k

When we combine the expressions for the paths of the two types above, we have the expression

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)}(R_{kk}^{(k-1)})^*R_{kj}^{(k-1)}$$

for the labels of all paths from state i to state j that go through no state higher than k. If we construct these expressions in order of increasing superscript, then since each $R_{ij}^{(k)}$ depends only on expressions with a smaller superscript, then all expressions are available when we need them.

Eventually, we have $R_{ij}^{(n)}$ for all i and j. We may assume that state 1 is the start state, although the accepting states could be any set of the states. The regular expression for the language of the automaton is then the sum (union) of all expressions $R_{1j}^{(n)}$ such that state j is an accepting state. $\square$

Theorem 3.7: Every language defined by a regular expression is also defined by a finite automaton.

PROOF: Suppose $L = L(R)$ for a regular expression R. We show that $L = L(E)$ for some ϵ-NFA E with:

1. Exactly one accepting state.

2. No arcs into the initial state.

3. No arcs out of the accepting state.

The proof is by structural induction on R, following the recursive definition of regular expressions that we had in Section 3.1.2.

BASIS: There are three parts to the basis, shown in Fig. 3.16. In part (a) we see how to handle the expression ϵ. The language of the automaton is easily seen to be $\{\epsilon\}$, since the only path from the start state to an accepting state is labeled ϵ. Part (b) shows the construction for $\emptyset$. Clearly there are no paths from start state to accepting state, so $\emptyset$ is the language of this automaton. Finally, part (c) gives the automaton for a regular expression $\mathbf{a}$. The language of this automaton evidently consists of the one string a, which is also $L(\mathbf{a})$. It

is easy to check that these automata all satisfy conditions (1), (2), and (3) of the inductive hypothesis.

INDUCTION: The three parts of the induction are shown in Fig. 3.17. We assume that the statement of the theorem is true for the immediate subexpressions of a given regular expression; that is, the languages of these subexpressions are also the languages of ϵ-NFA's with a single accepting state. The four cases are:

1. The expression is $R + S$ for some smaller expressions R and S. Then the automaton of Fig. 3.17(a) serves. That is, starting at the new start state, we can go to the start state of either the automaton for R or the automaton for S. We then reach the accepting state of one of these automata, following a path labeled by some string in $L(R)$ or $L(S)$, respectively. Once we reach the accepting state of the automaton for R or S, we can follow one of the ϵ-arcs to the accepting state of the new automaton. Thus, the language of the automaton in Fig. 3.17(a) is $L(R) \cup L(S)$.

2. The expression is RS for some smaller expressions R and S. The automaton for the concatenation is shown in Fig. 3.17(b). Note that the start state of the first automaton becomes the start state of the whole, and the accepting state of the second automaton becomes the accepting state of the whole. The idea is that the only paths from start to accepting state go first through the automaton for R, where it must follow a path labeled by a string in $L(R)$, and then through the automaton for S, where it follows a path labeled by a string in $L(S)$. Thus, the paths in the automaton of Fig. 3.17(b) are all and only those labeled by strings in $L(R)L(S)$.

3. The expression is R^* for some smaller expression R. Then we use the

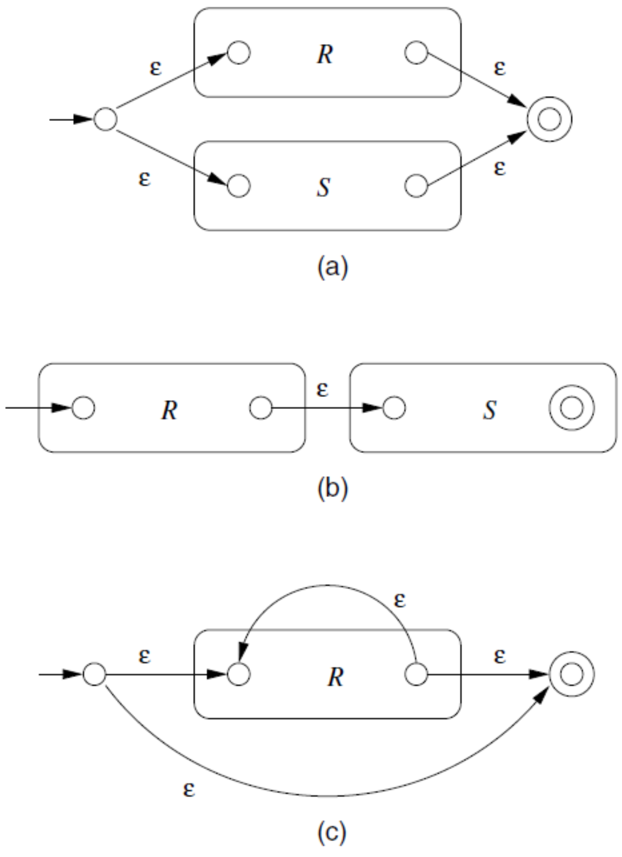

Figure 3.17: The inductive step in the regular-expression-to-ϵ-NFA construction

Figure 3.17: The inductive step in the regular-expression-to-ϵ-NFA construction

automaton of Fig. 3.17(c). That automaton allows us to go either:

(a) Directly from the start state to the accepting state along a path labeled ϵ. That path lets us accept ϵ, which is in $L(R^*)$ no matter what expression R is.

(b) To the start state of the automaton for R, through that automaton one or more times, and then to the accepting state. This set of paths allows us to accept strings in $L(R)$, $L(R)L(R)$, $L(R)L(R)L(R)$, and so on, thus covering all strings in $L(R^*)$ except perhaps ϵ, which was covered by the direct arc to the accepting state mentioned in (3a).

4. The expression is (R) for some smaller expression R. The automaton for R also serves as the automaton for (R), since the parentheses do not change the language defined by the expression.

It is a simple observation that the constructed automata satisfy the three conditions given in the inductive hypothesis — one accepting state, with no arcs into the initial state or out of the accepting state. $\square$