Regular Expressions

COMP 455 – 002, Spring 2019

Regular Expressions

- Regular expressions are simply *algebraic notation* for defining languages.
- A regular expression defines a language.
- In practice, regular expressions can usually define languages in a concise "user-friendly" manner.
  - At least compared to describing a finite automaton…

Definition of Regular Expressions

- Regular expressions are defined by taking the *union*, *concatenation*, and *closure* of languages.
  - **Union**: $L_1 \cup L_2 \equiv \{x \mid x \in L_1 \text{or } x \in L_2\}$
  - **Concatenation**: $L_1 L_2 \equiv \{xy \mid x \in L_1 \text{ and } y \in L_2\}$
  - **Closure** ("Kleene closure"): $L^* \equiv \cup_{i \geq 0} L^i$, where:
    - $L^0 = \{\varepsilon\}$
    - $L^1 = L$
    - $L^i =$ the concatenation of i copies of L

Kleene Closure: Example

- Let language $L = \{00, 11\}$.
- (From the previous definition) L^i represents i strings from L concatenated together.
  - $L^2 = \{0000, 0011, 1100, 1111\}$
  - $L^3 = \{000000, 000011, 001100, \ldots\}$
- L^* is *any* number of strings from L concatenated together.
  - $L^* = \{\varepsilon, 00, 11, 0000, 0011, 1111, 000000, \ldots\}$

Definition of Regular Expressions

Regular expressions are formally recursively defined (for an alphabet Σ):

1. $\emptyset$ is a regular expression denoting the empty set.
2. ε is a regular expression denoting $\{\varepsilon\}$.
3. For each $a \in \Sigma$, $\boldsymbol{a}$ is a regular expression denoting $\{a\}$.
4. If E and F are regular expressions denoting $L(E)$ and $L(F)$ (respectively), then:
   - $(E + F)$ is a regular expression denoting $L(E) \cup L(F)$
   - (EF) is a regular expression denoting $L(E)L(F)$
   - (E^*) is a regular expression denoting $L(E)^*$.

Note that we will use **bold** characters for literal symbols in regular expressions.

Example Regular Expressions

(The following examples assume $\Sigma = \{0, 1\}$.)

- **01**
  - The language consisting only of the string 01.
- **0 + 1**
  - The language consisting of the strings $\{0, 1\}$.
- **01 + 10**
  - The language consisting of the strings $\{01, 10\}$.

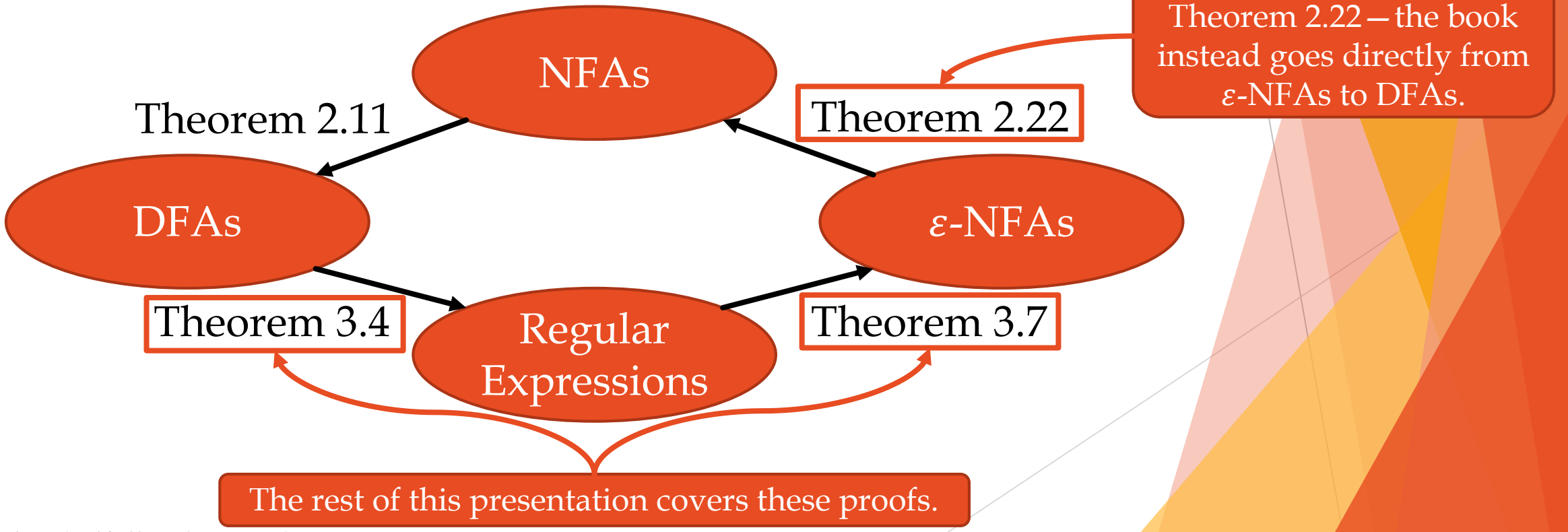
Example Regular Expressions

(The following examples assume $\Sigma = \{0, 1\}$.)

- $\mathbf{10(0 + 1)}$
  - The language consisting of $\{100, 101\}$.
- $\mathbf{10(0 + 1)}^*$
  - The language consisting of all strings that start with 10 and are followed by any number of 0s or 1s.
- $\mathbf{10(0 + 1)}^* + \varepsilon$
  - The same language as above, but also containing the empty string.

Finite Automata and Regular Expressions

Regular expressions and finite automata define the same class of languages.



From DFAs to Regular Expressions

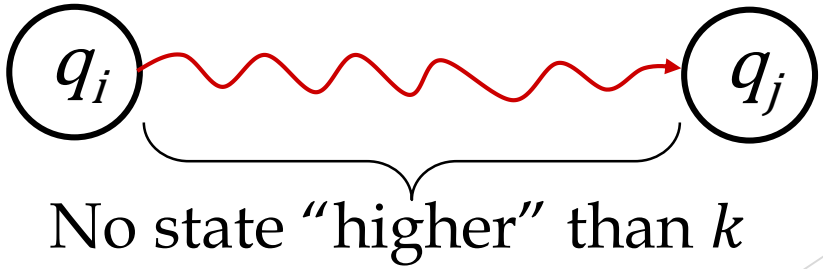
Theorem 3.4: If $L = L(M)$ for some DFA M, then a regular expression R exists for which $L = L(R)$.

We can re-label any set of states to start with q_1.

Proof:

- Let $M = \{\{q_1, q_2, \ldots, q_n\}, \Sigma, \delta, q_1, F\}$
- Let $R_{ij}^k \equiv$ all strings that take us from q_i to q_j without entering a state higher than q_k.

Note: $R_{ij}^n = all$ strings that go from state q_i to q_j.



Proof of Theorem 3.4, continued

- Let $R_{ij}^k \equiv$ all strings that take us from q_i to q_j without entering a state higher than q_k.

Define R_{ij}^k as follows:

$$R_{ij}^k = R_{ik}^{k-1}\left(R_{kk}^{k-1}\right)^* R_{kj}^{k-1} \cup R_{ij}^{k-1}$$

$$R_{ij}^0 = \{a \mid \delta(q_i, a) = q_j\}, \text{ if } i \neq j$$

$$R_{ii}^0 = \{a \mid \delta(q_i, a) = q_i\} \cup \{\varepsilon\}$$

Proof of Theorem 3.4, continued

$R_{ij}^k \equiv$ all strings that take us from q_i to q_j without entering a state higher than q_k.

Define R_{ij}^k as follows:

To get from q_i to q_j, potentially through q_k…

Start with strings that get from q_i to q_k through states lower than k….

Followed by any number of strings that cycle back to state q_k…

Followed by strings that get from state q_k to q_j.

Also include strings that go from q_i to q_j without going through q_k.

$$R_{ij}^k = R_{ik}^{k-1}\left(R_{kk}^{k-1}\right)^* R_{kj}^{k-1} \cup R_{ij}^{k-1}$$

$$R_{ij}^0 = \{a \mid \delta(q_i, a) = q_j\}, \text{ if } i \neq j$$

$$R_{ii}^0 = \{a \mid \delta(q_i, a) = q_i\} \cup \{\varepsilon\}$$

Proof of Theorem 3.4, continued

- Let $R_{ij}^k \equiv$ all strings that take us from q_i to q_j without entering a state higher than q_k.

Define R_{ij}^k as follows:

Strings that go directly from q_i to q_j.

These are the symbols that cause transitions from q_i to q_j.

$$R_{ij}^k = R_{ik}^{k-1}\left(R_{kk}^{k-1}\right)^* R_{kj}^{k-1} \cup R_{ij}^{k-1}$$

$$R_{ij}^0 = \{a \mid \delta(q_i, a) = q_j\}, \text{ if } i \neq j$$

$$R_{ii}^0 = \{a \mid \delta(q_i, a) = q_i\} \cup \{\varepsilon\}$$

Proof of Theorem 3.4, continued

- Let $R_{ij}^k \equiv$ all strings that take us from q_i to q_j without entering a state higher than q_k.

Define R_{ij}^k as follows:

Strings that loop from q_i back to q_i.

$$R_{ij}^k = R_{ik}^{k-1}\left(R_{kk}^{k-1}\right)^* R_{kj}^{k-1} \cup R_{ij}^{k-1}$$

$$R_{ij}^0 = \{a \mid \delta(q_i, a) = q_j\}, \text{ if } i \neq j$$

$$R_{ii}^0 = \{a \mid \delta(q_i, a) = q_i\} \cup \{\varepsilon\}$$

Proof of Theorem 3.4, continued

Claim: There exists a regular expression r_{ij}^k denoting R_{ij}^k.

We will prove this by induction on k.

Basis: $k = 0$

Let $A \equiv \{a_1, a_2, \ldots, a_p\}$ denote the symbols that take q_i to q_j. Then,

$$r_{ij}^0 = \begin{cases} \boldsymbol{a_1 + a_2 + \cdots + a_p} & \text{, if } A \neq \emptyset \text{ and } i \neq j \\ \emptyset & \text{, if } A = \emptyset \text{ and } i \neq j \\ \boldsymbol{a_1 + a_2 + \cdots + a_p} + \varepsilon & \text{, if } A \neq \emptyset \text{ and } i = j \\ \varepsilon & \text{, if } A = \emptyset \text{ and } i = j \end{cases}$$

Proof of Theorem 3.4, continued

Claim: There exists a regular expression r_{ij}^k denoting R_{ij}^k.

We will prove this by induction on k.

Inductive step: $k > 0$.

In this case,

$$r_{ij}^k = r_{ik}^{k-1}\left(r_{kk}^{k-1}\right)^* r_{kj}^{k-1} + r_{ij}^{k-1}$$

Proof of Theorem 3.4, continued

Claim: There exists a regular expression r_{ij}^k denoting R_{ij}^k.

We will prove this by induction on k.

Inductive step: $k > 0$.

In this case,

$$r_{ij}^k = r_{ik}^{k-1}\left(r_{kk}^{k-1}\right)^* r_{kj}^{k-1} + r_{ij}^{k-1}$$

The inductive hypothesis lets us assume that we already have these regular expressions!

Proof of Theorem 3.4, continued

Finishing up:

We still need to show that we can construct a regular expression that matches $L(M)$.

We note that $L(M) = \bigcup_{q_j \in F} R_{1j}^n$

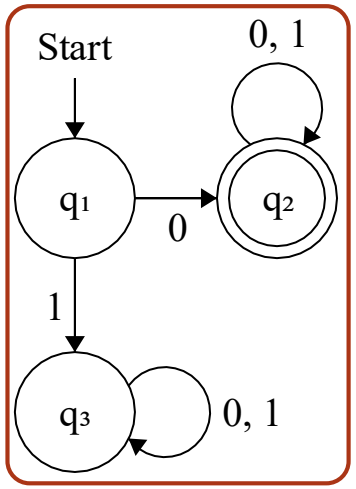
Any string that takes us from state q_1 to q_j, where q_j is a final state.

Therefore, $L(M)$ is denoted by the regular expression:

$r_{1j_1}^n + r_{1j_2}^n + \cdots + r_{1j_p}^n$, where $F = \{q_{j_1}, q_{j_2}, \ldots, q_{j_p}\}$.

Example: DFA to Regular Expression

	$k = 0$	$k = 1$	$k = 2$	$k = 3$
r_{11}^k	ε			
r_{12}^k	$\mathbf{0}$			
r_{13}^k	$\mathbf{1}$			
r_{21}^k	$\emptyset$			
r_{22}^k	$\mathbf{0} + \mathbf{1} + \varepsilon$			
r_{23}^k	$\emptyset$			
r_{31}^k	$\emptyset$			
r_{32}^k	$\emptyset$			
r_{33}^k	$\mathbf{0} + \mathbf{1} + \varepsilon$			

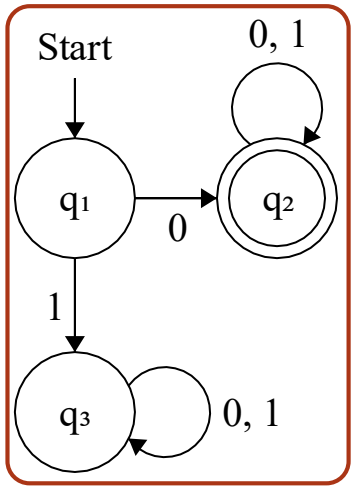


If A is the set of symbols going from q_i to q_j:

$$r_{ij}^0 = \begin{cases} \boldsymbol{a_1} + \boldsymbol{a_2} + \cdots + \boldsymbol{a_p} & \text{, if } A \neq \emptyset \text{ and } i \neq j \\ \emptyset & \text{, if } A = \emptyset \text{ and } i \neq j \\ \boldsymbol{a_1} + \boldsymbol{a_2} + \cdots + \boldsymbol{a_p} + \varepsilon & \text{, if } A \neq \emptyset \text{ and } i = j \\ \varepsilon & \text{, if } A = \emptyset \text{ and } i = j \end{cases}$$

Example: DFA to Regular Expression

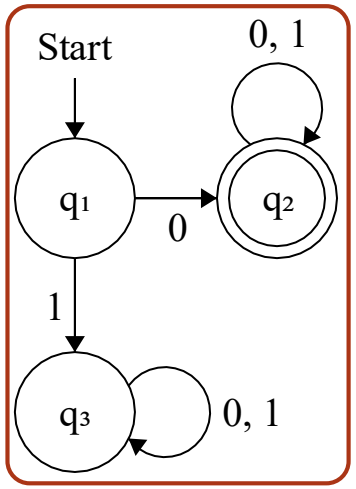
	$k = 0$	$k = 1$	$k = 2$	$k = 3$
r_{11}^k	ε	ε		
r_{12}^k	$\mathbf{0}$	$\mathbf{0}$		
r_{13}^k	$\mathbf{1}$	$\mathbf{1}$		
r_{21}^k	$\emptyset$	$\emptyset$		
r_{22}^k	$\mathbf{0} + \mathbf{1} + \varepsilon$	$\mathbf{0} + \mathbf{1} + \varepsilon$		
r_{23}^k	$\emptyset$	$\emptyset$		
r_{31}^k	$\emptyset$	$\emptyset$		
r_{32}^k	$\emptyset$	$\emptyset$		
r_{33}^k	$\mathbf{0} + \mathbf{1} + \varepsilon$	$\mathbf{0} + \mathbf{1} + \varepsilon$		



$$r_{ij}^k = r_{ik}^{k-1}\left(r_{kk}^{k-1}\right)^* r_{kj}^{k-1} + r_{ij}^{k-1}$$

Example: DFA to Regular Expression

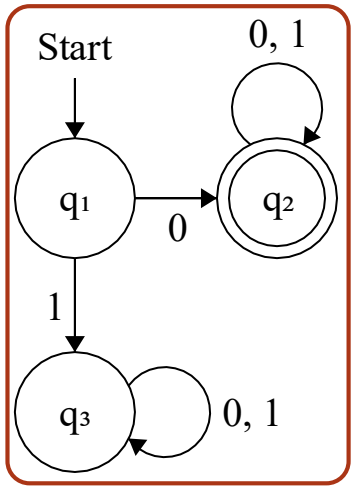
	$k = 0$	$k = 1$	$k = 2$	$k = 3$
r_{11}^k	ε	ε	ε	
r_{12}^k	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0(0+1)^*}$	
r_{13}^k	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	
r_{21}^k	$\emptyset$	$\emptyset$	$\emptyset$	
r_{22}^k	$\mathbf{0+1+\varepsilon}$	$\mathbf{0+1+\varepsilon}$	$\mathbf{(0+1)^*}$	
r_{23}^k	$\emptyset$	$\emptyset$	$\emptyset$	
r_{31}^k	$\emptyset$	$\emptyset$	$\emptyset$	
r_{32}^k	$\emptyset$	$\emptyset$	$\emptyset$	
r_{33}^k	$\mathbf{0+1+\varepsilon}$	$\mathbf{0+1+\varepsilon}$	$\mathbf{0+1+\varepsilon}$	



$$r_{ij}^k = r_{ik}^{k-1}\left(r_{kk}^{k-1}\right)^* r_{kj}^{k-1} + r_{ij}^{k-1}$$

Example: DFA to Regular Expression

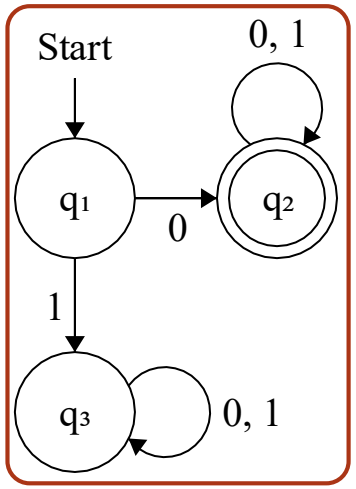
	$k = 0$	$k = 1$	$k = 2$	$k = 3$
r_{11}^k	ε	ε	ε	ε
r_{12}^k	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0(0+1)^*}$	$\mathbf{0(0+1)^*}$
r_{13}^k	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1(0+1)^*}$
r_{21}^k	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
r_{22}^k	$\mathbf{0+1+\varepsilon}$	$\mathbf{0+1+\varepsilon}$	$\mathbf{(0+1)^*}$	$\mathbf{(0+1)^*}$
r_{23}^k	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
r_{31}^k	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
r_{32}^k	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
r_{33}^k	$\mathbf{0+1+\varepsilon}$	$\mathbf{0+1+\varepsilon}$	$\mathbf{0+1+\varepsilon}$	$\mathbf{(0+1)^*}$



$$r_{ij}^k = r_{ik}^{k-1}\left(r_{kk}^{k-1}\right)^* r_{kj}^{k-1} + r_{ij}^{k-1}$$

Example: DFA to Regular Expression

	$k = 0$	$k = 1$	$k = 2$	$k = 3$
r_{11}^k	ε	ε	ε	ε
r_{12}^k	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0(0+1)^*}$	$\mathbf{0(0+1)^*}$
r_{13}^k	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1(0+1)^*}$
r_{21}^k	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
r_{22}^k	$\mathbf{0+1+\varepsilon}$	$\mathbf{0+1+\varepsilon}$	$\mathbf{(0+1)^*}$	$\mathbf{(0+1)^*}$
r_{23}^k	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
r_{31}^k	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
r_{32}^k	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
r_{33}^k	$\mathbf{0+1+\varepsilon}$	$\mathbf{0+1+\varepsilon}$	$\mathbf{0+1+\varepsilon}$	$\mathbf{(0+1)^*}$



The expression taking us from q_1 to q_2 through states up to q_3 is r_{12}^3: $\mathbf{0(0+1)^*}$

Comments on this Approach

- This approach is an example of *dynamic programming*, a COMP 550 topic.
- For an n-state DFA, this approach may produce a regular expression with 4^n symbols!
- Section 3.2.2 of the book presents a more efficient method.

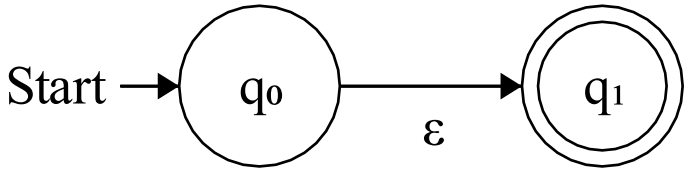
From Regular Expressions to ε-NFAs

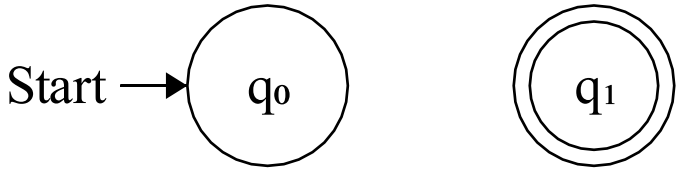
Theorem 3.7 (reworded): If R is a regular expression, $L(R) = L(E)$ for some NFA with ε-transitions E.

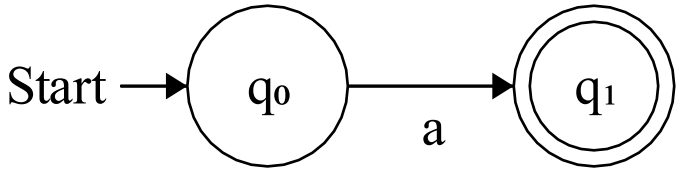
- We will construct E with one final state and *no transitions out of that state*.
- We will do this by induction on the number of operators in R.

Proof of Theorem 3.7

Base case: R has no operators. There are three possibilities here:

- $R = \varepsilon$ E: 

- $R = \emptyset$ E: 

- $R = \boldsymbol{a}$ E: 

Proof of Theorem 3.7, continued

Inductive step:

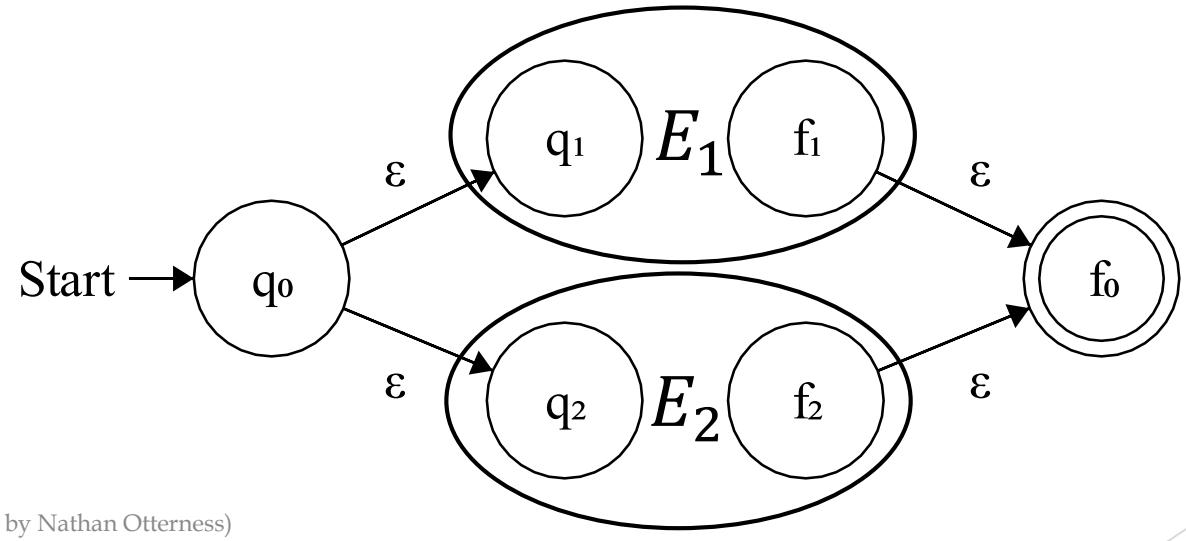
We now need to handle the three regular-expression operators:

- Union $(R_1 + R_2)$
- Concatenation $(R_1 R_2)$
- Closure (R_1^*)

Proof of Theorem 3.7, continued

Inductive step: Union: $R = R_1 + R_2$

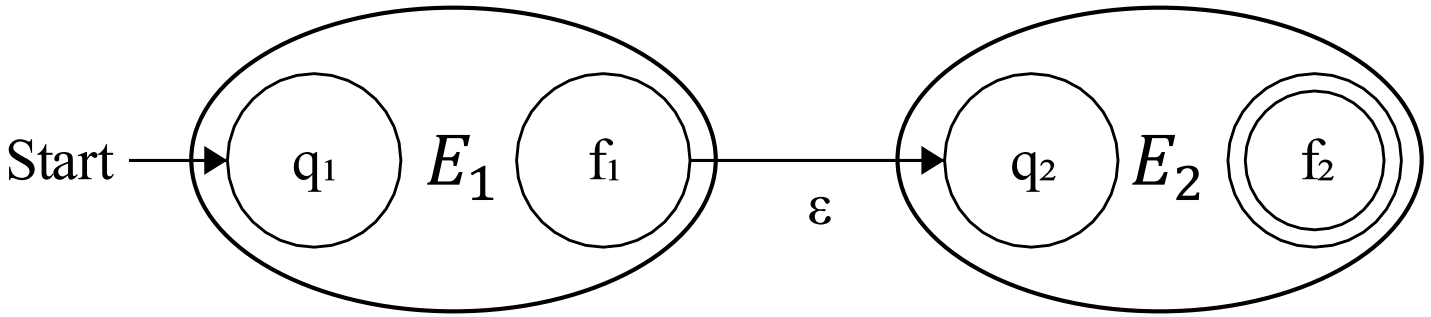
We can assume by the inductive hypothesis that we already have ε-NFAs E_1 and E_2 accepting the same languages as R_1 and R_2. Construct the ε-NFA E to accept the same language as R like this:



Proof of Theorem 3.7, continued

Inductive step: Concatenation: $R = R_1 R_2$

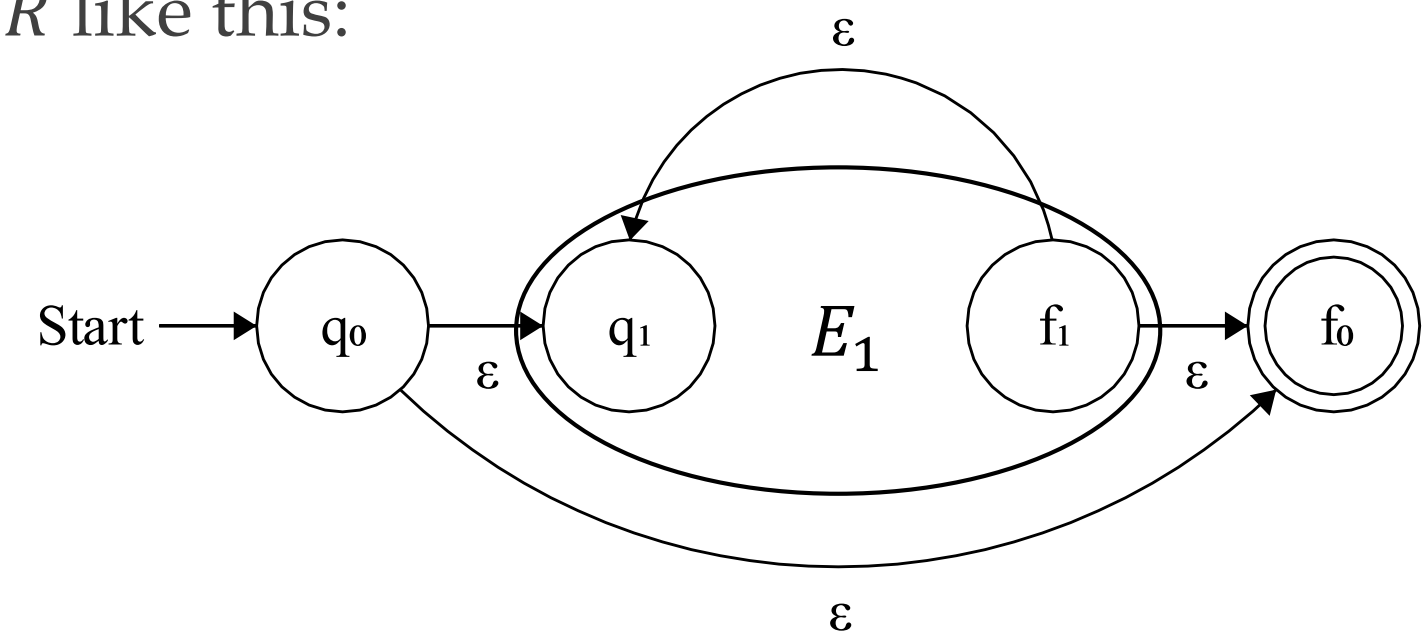
Once again, assume E_1 and E_2 are ε-NFAs accepting the same languages as R_1 and R_2. Construct the ε-NFA E to accept the same language as R like this:



Proof of Theorem 3.7, continued

Inductive step: Closure: $R = R_1^*$

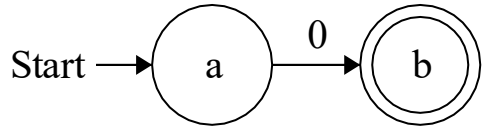
Assume E_1 is an ε-NFA accepting the same language as R_1. Construct the ε-NFA E to accept the same language as R like this:



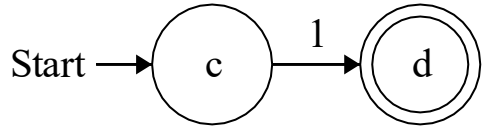
Example Regular Expression Conversion

We'll convert the regular expression $\mathbf{0(0+1)^*}$ to an NFA with ε-transitions.

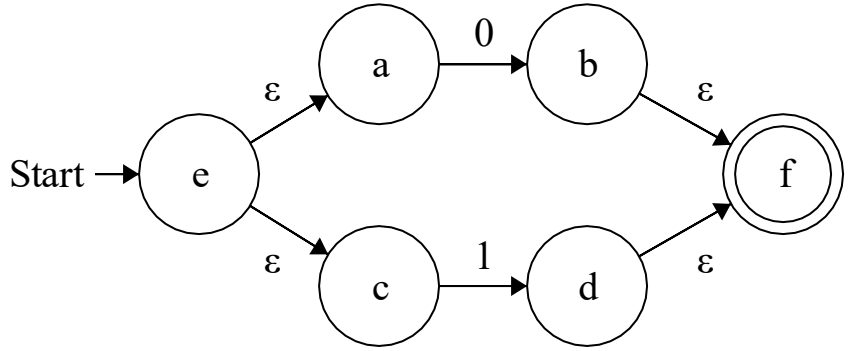
0:



1:

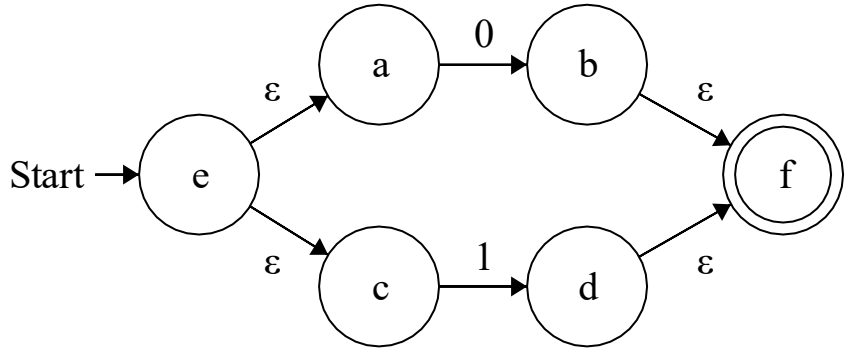


0 + 1:

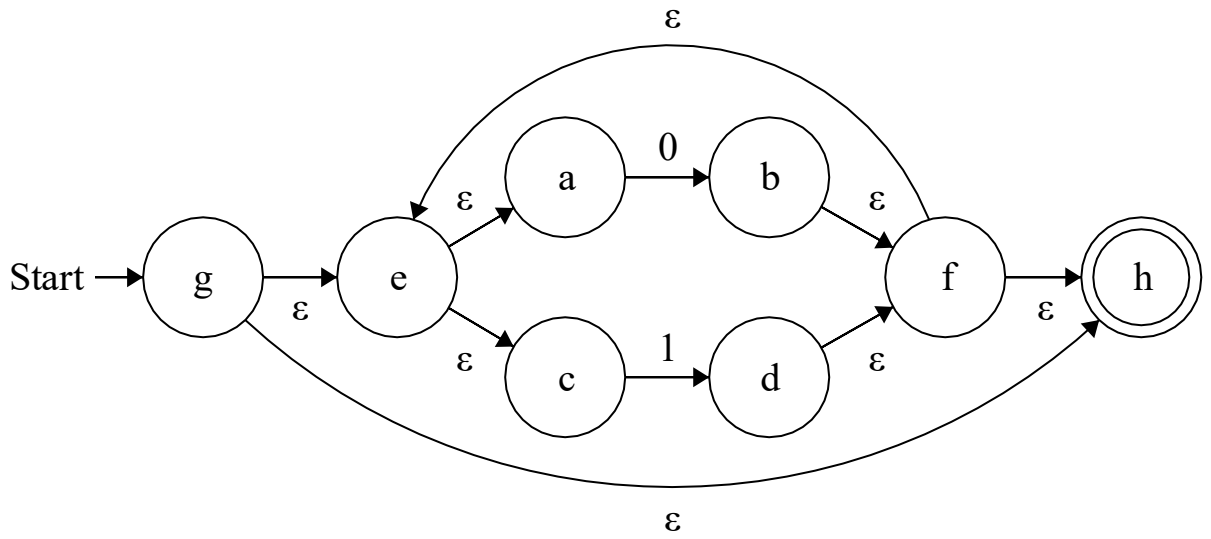


Example Regular Expression Conversion

$\mathbf{0 + 1}$:

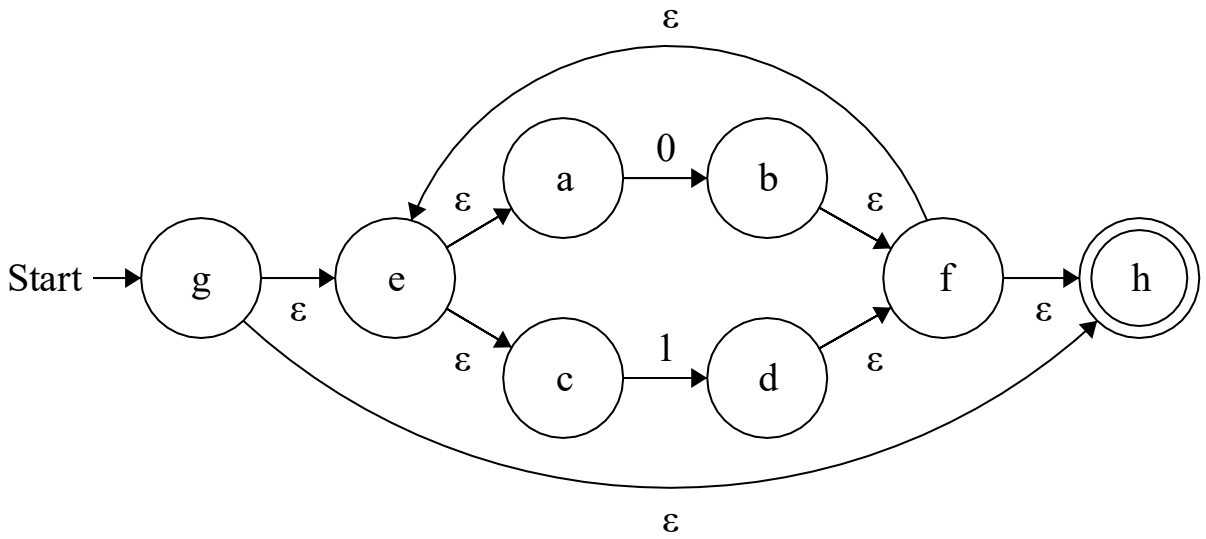


$(\mathbf{0 + 1})^*$:

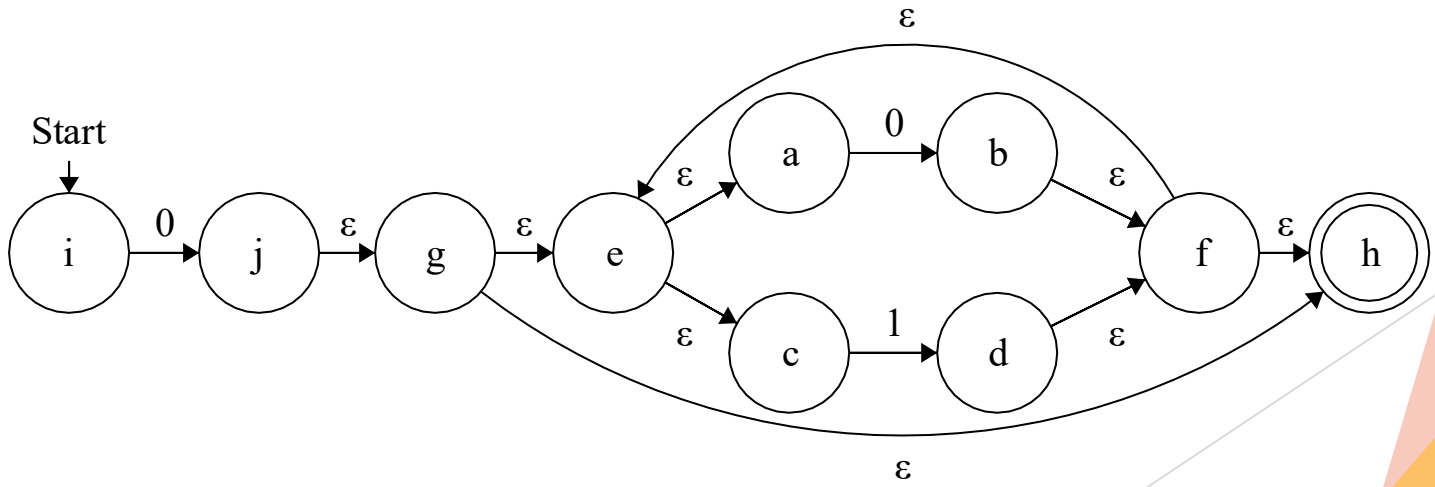


Example Regular Expression Conversion

$(\mathbf{0}+\mathbf{1})^*$:

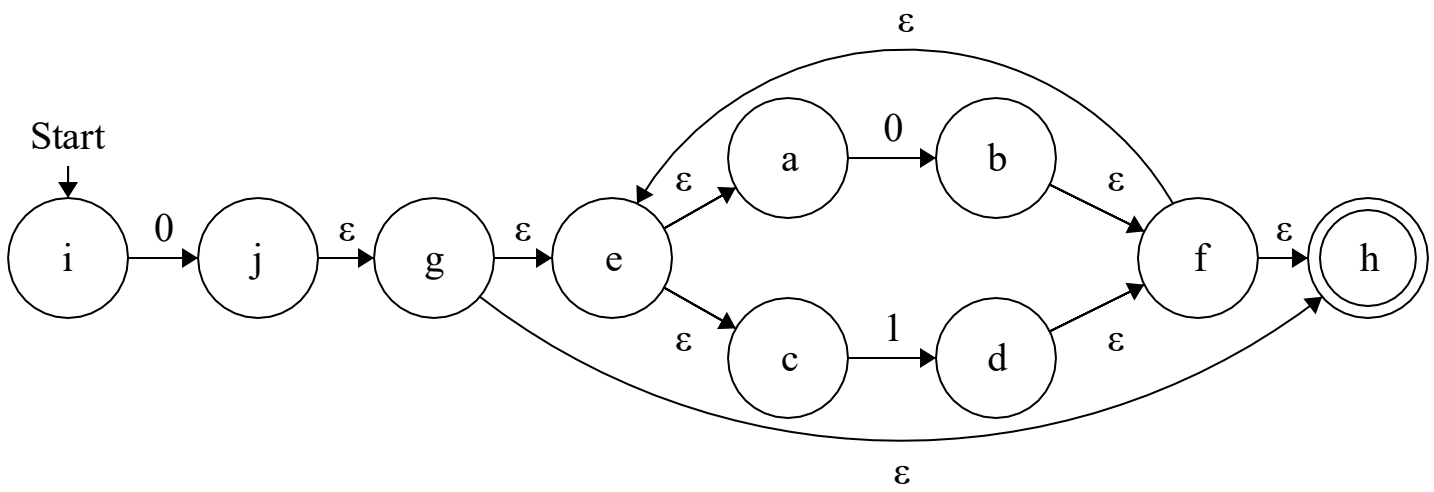


$\mathbf{0}(\mathbf{0}+\mathbf{1})^*$:



Example Regular Expression Conversion

$\mathbf{0(0+1)^*}$:



Note: A *much* simpler machine exists.