Lecture 2: Regular Languages

Finite Automata

A **finite automaton** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

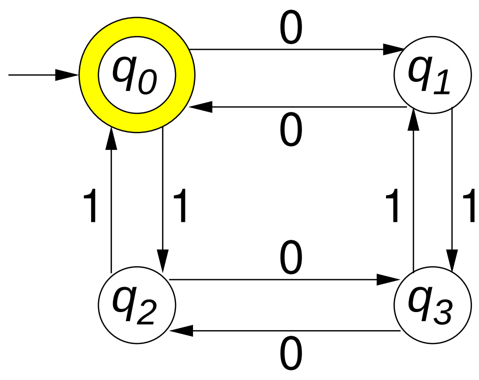
1. Q is a finite set called the **states**,
2. Σ is a finite set called the **alphabet**,
3. $\delta : Q \times \Sigma \longrightarrow Q$ is the **transition function**,
4. $q_0 \in Q$ is the **initial state**, and
5. $F \subseteq Q$ is the **set of accepting states**.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be an FA. A string $w = w_1 \cdots w_n$ is **accepted** by M if there exists a sequence $(p_0, \ldots, p_n)$ of states in Q such that $p_0 = q_0$, $p_n \in F$, and for every i, $1 \leq i \leq n$, $\delta(p_{i-1}, w_i) = p_i$.

The language **recognized** (or **accepted**) by M, notated as $L(M)$, is the language over Σ such that

(*) for every string w over Σ, $w \in L(M) \Leftrightarrow M$ accepts w.

Example: A FA that recognizes the language over $\{0, 1\}$ consisting of all the strings with an even number of 0s and an even number of 1s



Regular Languages

Let A and B be two languages. Define:

- **Union** of A and B, $A \cup B$, is $\{x \mid x \in A \text{ or } x \in B\}$,
- **Concatenation** of A and B, $A \circ B$, is $\{xy \mid x \in A \text{ and } y \in B\}$,
- **Star** of A, A^*, is $\{x_1 x_2 \cdots x_k \mid k \geq 0 \text{ and } x_1, \ldots, x_k \in A\}$.

The class of **regular languages** is the class of languages recognized by finite automata.

Nondeterministic Finite Automata

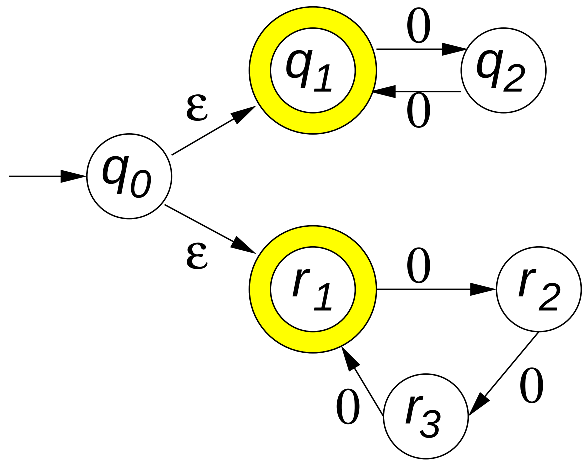
A **nondeterministic finite automaton** is a 5-tuple $N = (Q, \Sigma, \delta, q_0, F)$, where δ now is a mapping of $Q \times \Sigma_\epsilon$ to $\mathcal{P}(Q)$, the **power set of Q**, i.e., the collection of all subsets of Q, where $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$.

For each $p \in Q$ and each $a \in \Sigma_\epsilon$, "$\delta(s, a) = R$" states that **upon reading an a, N can go from s to any state in R.**

A string $w = w_1 \cdots w_n$ is **accepted** by N if there exists a sequence $(p_0, \ldots, p_m)$ of states in Q and a representation $y = y_1 \cdots y_m$ of w over Σ_ϵ such that $p_0 = q_0$, $p_m \in F$, and for every i, $1 \leq i \leq m$, $p_i \in \delta(p_{i-1}, y_i)$.

Example of NFA

An NFA that recognizes the language over $\{0\}$ that consists of all strings w such that $|w|$ is either a multiple of 2 or a multiple of 3.



FA = NFA

Theorem. *Every NFA can be converted an equivalent FA.*

Proof Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA. Define a FA $M = (S, \Sigma, \gamma, s_0, G)$ by

- $S = \mathcal{P}(Q)$,
- $s_0 = \{q_0\}$,
- $G = \{A \subseteq Q \mid A \cap F \neq \emptyset\}$, and
- for each $A \in S$ and $b \in \Sigma$, $\gamma(A, b) = \bigcup_{p \in A} \delta(p, \epsilon^* b \epsilon^*)$.

Here $\gamma(A, b)$ is the set of all states that M can go to from one of the states in A, upon receiving symbol b. So, w over Σ is accepted by N if and only if w takes M from the state s_0 to a subset of Q containing an element in F (the set of such subsets is G). ∎

Regular Expressions

An expression R is a **regular expression** if R is

1. a for some a in some alphabet Σ,
2. ϵ,
3. $\emptyset$,
4. $(R_1 \cup R_2)$ for some regular expressions R_1 and R_2,
5. $(R_1 \circ R_2)$ for some regular expressions R_1 and R_2, or
6. $(R_1)^*$ for some regular expression R_1.

$L(a) = \{a\},\ L(\epsilon) = \{\epsilon\},\ L(\emptyset) = \emptyset$,
$L(R_1 \cup R_2) = L(R_1) \cup L(R_2),\ L(R_1 \circ R_2) = L(R_1) \circ L(R_2)$,
$L((R_1)^*) = (L(R_1))^*$

Finite Automata are equivalent to Regular Expressions

This requires proofs in both directions.

Lemma (REXPR $\Rightarrow$ NFA $\Rightarrow$ FA). *Every regular expression describes a regular language.*

Proof For any $a \in \Sigma$, $L(a)$ is regular; $L(\epsilon)$ is regular; and $L(\emptyset)$ is regular. For $(R_1 \cup R_2)$, $(R_1 \circ R_2)$, $(R_1)^*$, assume for induction that there are FAs M_1, M_2 such that $L_i =_{def} L(M_i) = L(R_i)$, for $i = 1, 2$.

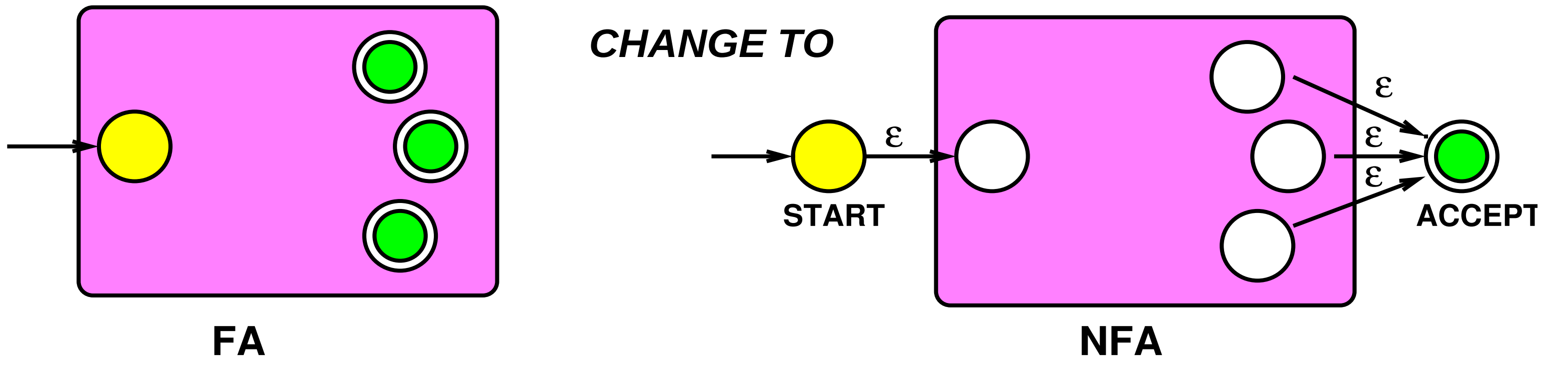
Let FA $M_i, i = 1, 2$, have initial state p_i and final states F_i. Construct an NFA with initial state p_0 and final states F_0:

Union, $(L_1 \cup L_2)$: p_0 has an ϵ-move to p_1 and to p_2; $F_0 = F_1 \cup F_2$
Concatenation, $(L_1 \circ L_2)$: $p_0 = q_1$, an ϵ-move from each $s \in F_1$ to p_2, $F_0 = F_2$
Star, $(L_1)^*$: $p_0 = p_1$, an ϵ-move from each $s \in F_1$ to p_1, $F_0 = F_1$ ∎

Lemma (FA $\Rightarrow$ NFA $\Rightarrow$ REXPR). *Every regular language is described by some regular expression.*

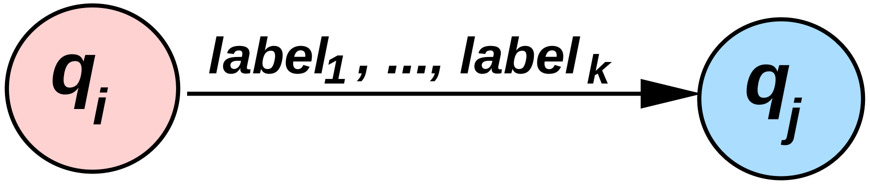
Proof (Sketch) We want to get from an FA to a regular expression.

Step 1: Change an FA for any given regular language L to an equivalent NFA with a new start state and a new, unique accepting state, using ϵ-transitions.

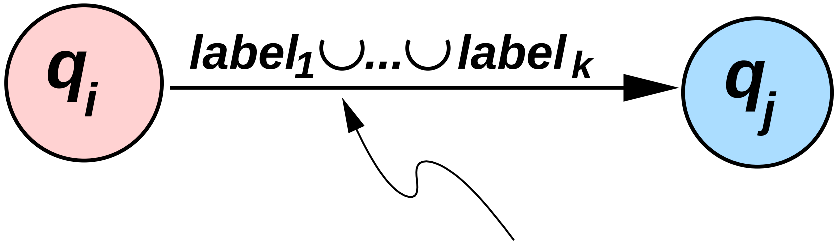


Step 2: **Combine multiple labels.**

(a) For every transition

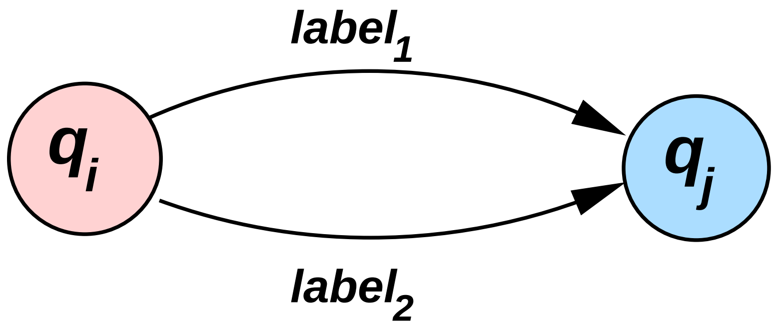


in the NFA, change it to

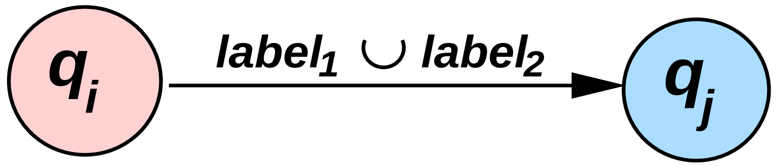


a regular expression, so we call the resulting automaton a GENERALIZED NFA (GNFA)

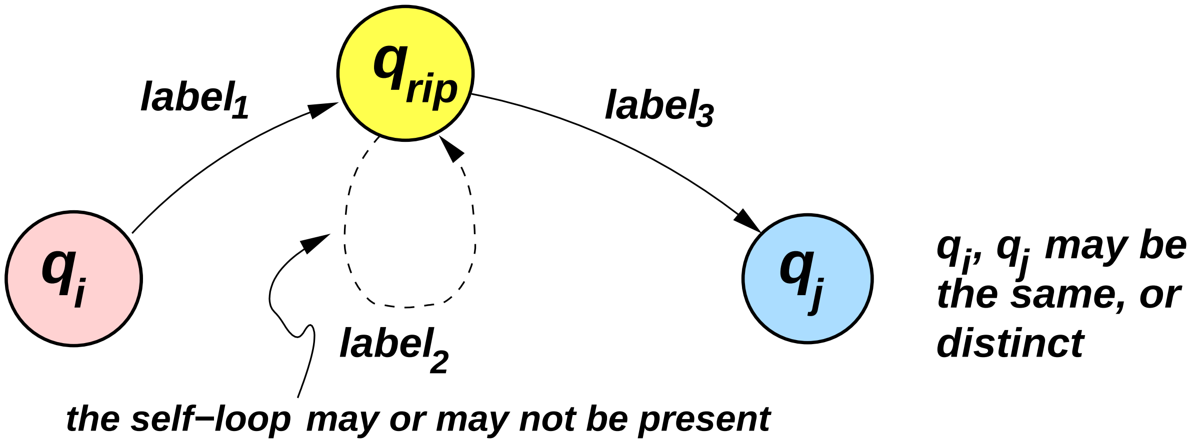
(b) For every pair of transitions (while there remain such pairs)



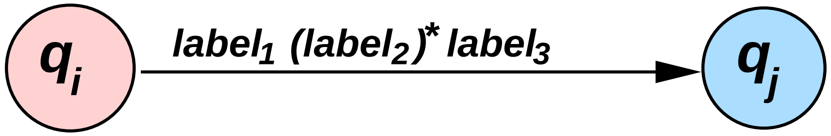
in the GNFA, change these to



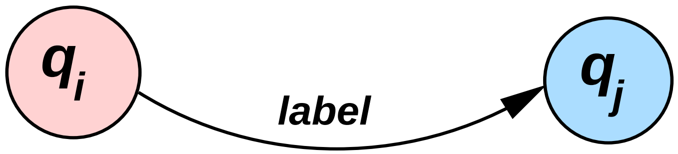
Step 2: Eliminate ("rip out") successive states of the GNFA (other than START and ACCEPT), replacing lost 2-step paths by 1-step paths. I.e., when we eliminate a state q_{rip}, we generally "lose" some local paths of the form



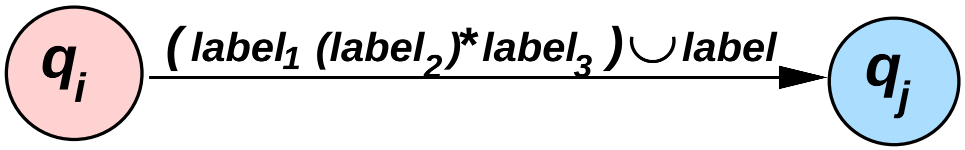
For *each* such local path,
(a) insert a single arrow labeled like this:



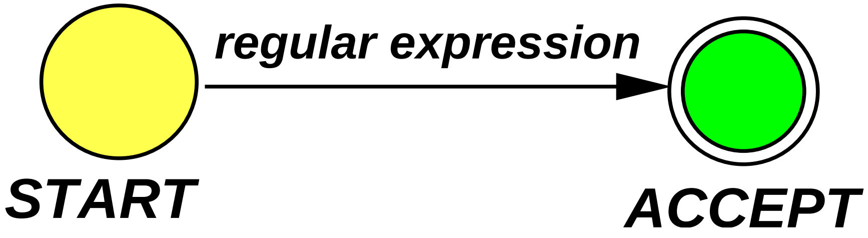
(b) If there was already a transition



from q_i to q_j, merge the two arrows into



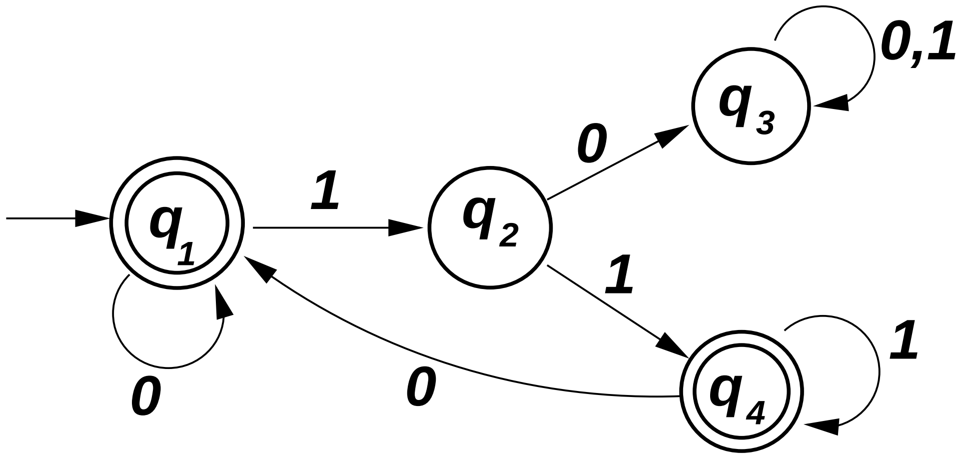
Result:



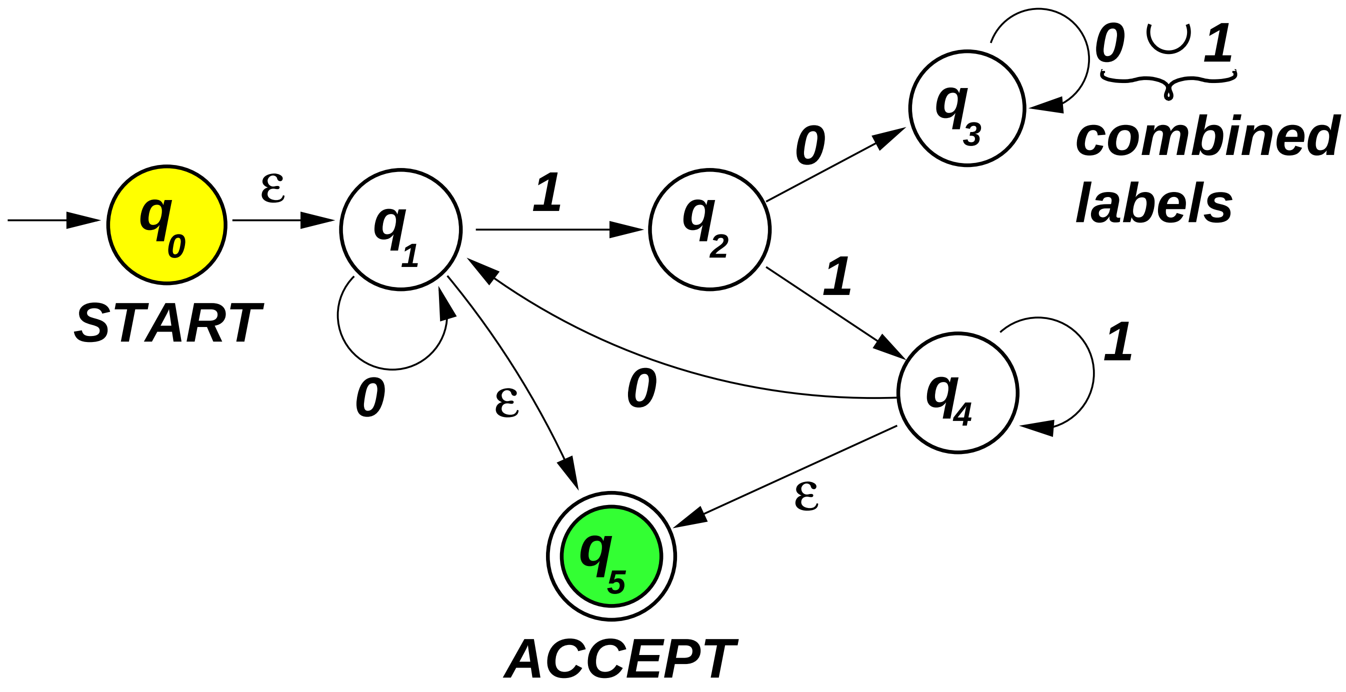
The regular expression describes L, the language recognized by the original FA. Showing this requires an inductive proof that *at each step* of the conversion, the resulting GNFA still accepts exactly the same input sequences as before. We'll omit further details.

Example of FA $\Rightarrow$ REXPR

A FA that recognizes the language over $\{0, 1\}$ that consists of all strings w with no “isolated” 1s, i.e., wherever there is a 1, there is another 1 adjacent on its left or right.



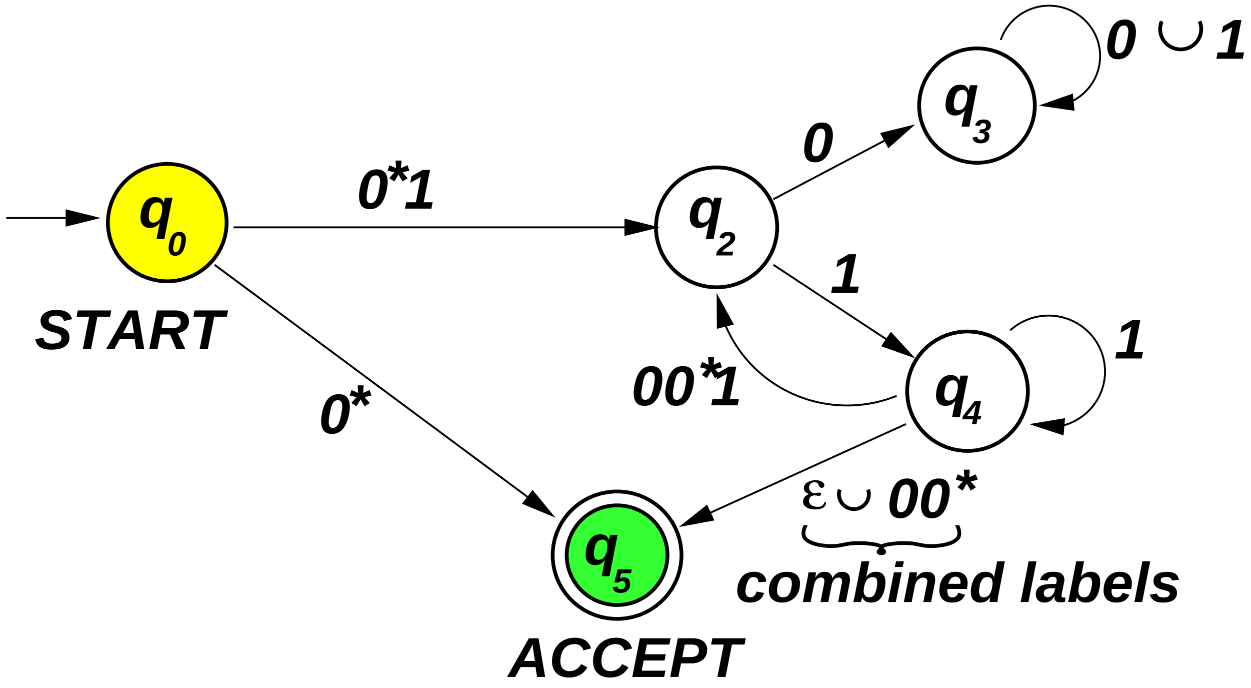
Steps 1 and 2:



q_0
START
ε
q_1
0
1
q_2
0
q_3
$0 \cup 1$
combined labels
1
q_4
1
0
ε
ε
q_5
ACCEPT

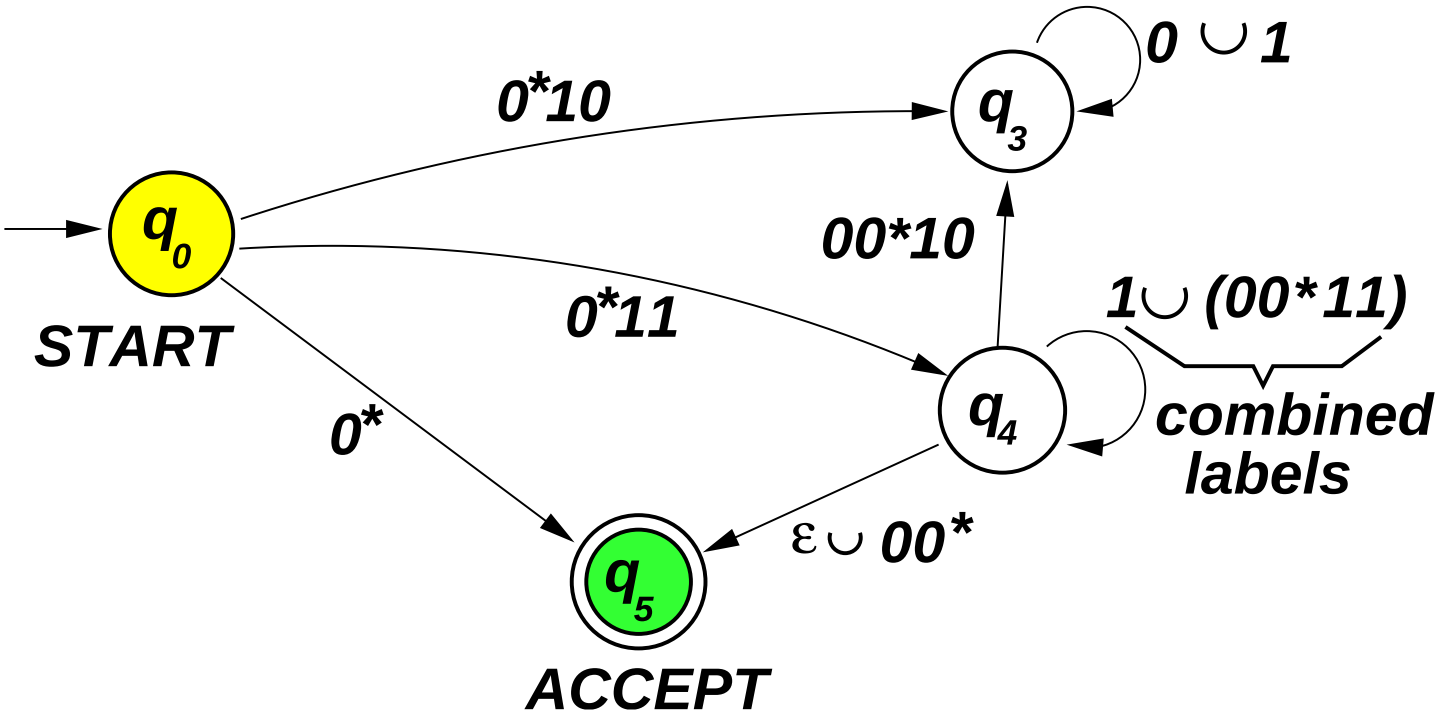
Step 3:

Rip q_1: there are 4 local paths to replace



Step 3, continued:

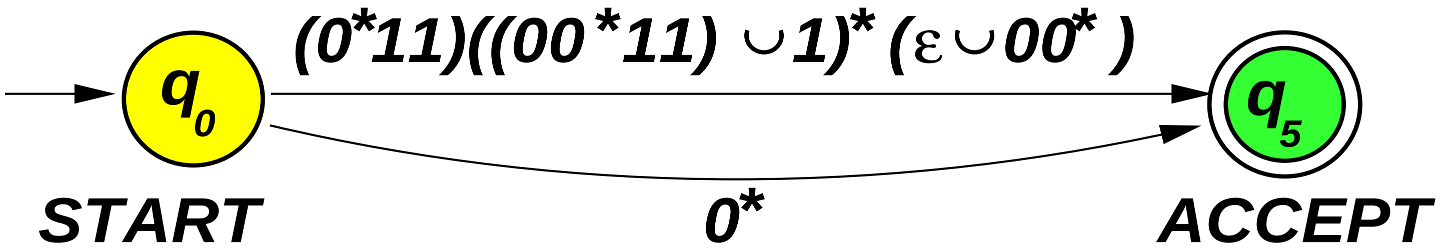
Rip q_2: there are 4 local paths to replace



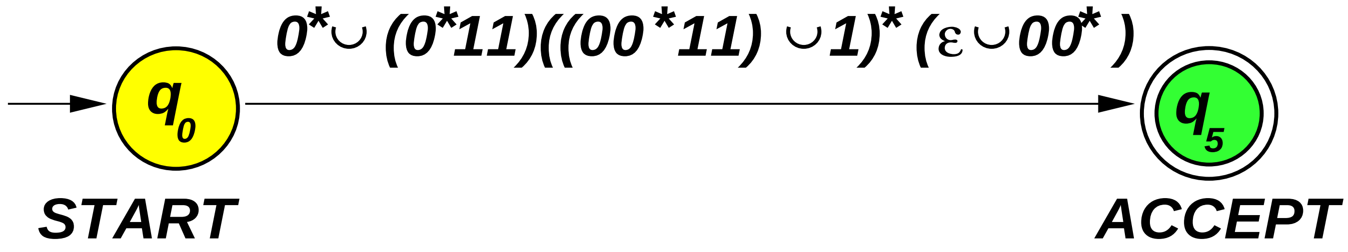
Step 3, continued:

Rip q_3: there are no local paths through q_3, so it just disappears

Rip q_4: there is 1 local path to replace



Step 3, concluded:
Combine labels:



Not as simple as it might be: $(111^* \cup 0)^*$