REGULAR EXPRESSIONS

DEFINITION OF REGULAR EXPRESSIONS

- A ***regular expression*** is another tool that can describe regular languages.

- The three operations on languages described earlier are also used in the construction of regular expressions.

- For example, the regular expression 0 (01)* describes a language composed of strings that start with 0 followed by 0 or more concatenations of 01.

DEFINITION OF REGULAR EXPRESSIONS

- Formal definition

  A regular expression over a given alphabet Σ is constructed using the following rules:

  1. ε (the empty string) is a regular expression.
  2. ϕ (the empty language) is a regular expression.
  3. Every symbol *x* in Σ is a regular expression.

DEFINITION OF REGULAR EXPRESSIONS

4. If R_1 and R_2 are regular expressions:

   - $R_1 + R_2$ (the union of two regular expressions) is a regular expression.
   - R_1R_2 (the concatenation of two regular expressions) is a regular expression.
   - R_1^* (the star of a regular expression) is a regular expression.

Anything that is constructed without following any of the above mentioned rules is not a regular expression.

MORE ON REGULAR EXPRESSIONS

- Regular expressions can use parentheses to enforce precedence.

  If there are no parentheses, evaluation is done in the precedence order: star, then concatenation, then union.

  For example:

  0(01)* is a language composed of strings that start with a 0 followed by zero or more concatenations of 01.

  001* is a language composed of strings that start with a 00 followed by zero or more concatenations of 1.

MORE ON REGULAR EXPRESSIONS

- If *R* is a regular expression, the language denoted or described by *R* is written as *L*(*R*).

  Example:

  If R = (00)* then

  L(R) = {ε, 00, 0000, 000000, …}

  Take note that R^k is the notation used to denote *k* concatenations of *R*. For example, if *R* = 00, then the expression R^2 = 0000. Obviously, R^0 = ε.

MORE ON REGULAR EXPRESSIONS

- Assume Σ is an alphabet. Σ can also be considered as a regular expression that describes the language consisting of all strings of length 1 over this alphabet.

  In other words, if $\Sigma = \{0, 1\}$, then Σ as a regular expression is equivalent to (0 + 1), the language composed of strings which are either 0 or 1 only.

  Hence, the regular expression Σ^*, which is equivalent to (0 + 1)*, denotes the language composed of all possible strings of 0s or 1s.

MORE ON REGULAR EXPRESSIONS

- Recall that R^* denotes all strings that are 0 or more concatenations of strings generated by R. This of course includes the empty string ε.

  The notation R^+ will now be used to denote all strings that are one or more concatenations of strings generated by R. This excludes the empty string ε.

  R^* is then related to R^+ by the expression

$$R^* = \{\varepsilon\} \cup R^+$$

MORE ON REGULAR EXPRESSIONS

- Some examples:
  - R = 1*01*

    L(R) = {*w* | *w* contains exactly one 0}
  - R = (0 + 1)*1(0 + 1)*

    L(R) = {*w* | *w* contains at least one 1}
  - R = (0 + 1)* 111 (0 + 1)*

    L(R) = {*w* | *w* contains the substring 111}

MORE ON REGULAR EXPRESSIONS

- R = (00)*(11)*1

  L(R) = {*w*|*w* is composed of an even number of 0s followed by an odd number of 1s}

- R = ((0 + 1) (0 + 1))*

  L(R) = {*w*|*w* is a string whose length is even}

- R = ((0 + 1) (0 + 1)(0 + 1))*

  L(R) = {*w*|*w* is a string whose length is a multiple of three}

MORE ON REGULAR EXPRESSIONS

- $R = (01^+)^*$

  $L(R) = \{w \mid w$ has every 0 followed by at least one 1$\}$

- $R = (0+1)^*00(0+1)^*$

  $L(R) = \{w \mid w$ has at least one pair of consecutive 0s$\}$

- $R = (1 + 01^*)^*(0 + \varepsilon)$

  $L(R) = \{w \mid w$ has no consecutive 0s$\}$

REGULAR EXPRESSIONS AND FINITE AUTOMATA

- Regular expressions are similar to finite automata (deterministic and nondeterministic) in the sense that they are used to describe languages.

- A language is regular if there is some NFA that recognizes it.

- Do regular expressions also define regular languages in the same manner as finite automata do?

REGULAR EXPRESSIONS AND FINITE AUTOMATA

- To determine if regular expressions define regular languages, it must shown that:

1. If a language is described by a regular expression, then it is a regular language.

2. If a language is regular, then there is a regular expression that describes it.

REGULAR EXPRESSIONS AND FINITE AUTOMATA

- To show if a language that is described by a regular expression is a regular language, it must be shown that there is an NFA that recognizes the language.

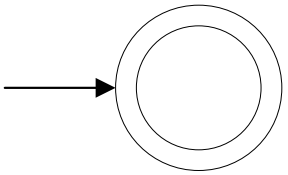
- To do this, a procedure must be outlined for converting the regular expression that describes a certain language to an NFA.

- If this is done, then the language that is being described by the regular expression is a regular language.

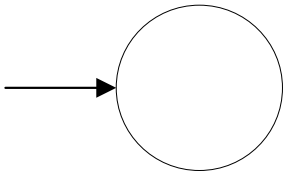
REGULAR EXPRESSIONS TO NFA

- The procedure for converting a regular expression to an NFA will be based on the definition of regular expressions.

1. The empty string ε is a regular expression. The NFA that accepts the empty string is

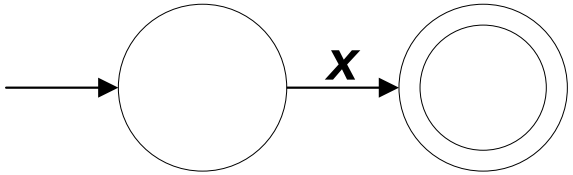


2. The empty language ϕ is a regular expression. The NFA that accepts the empty language is

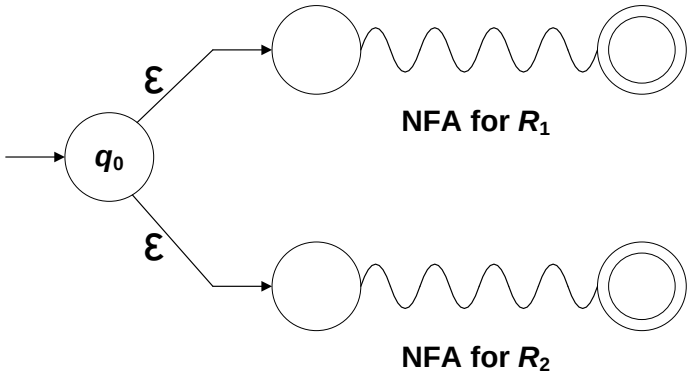


REGULAR EXPRESSIONS TO NFA

3. Every symbol *x* in Σ is a regular expression. The NFA that accepts the symbol *x* is

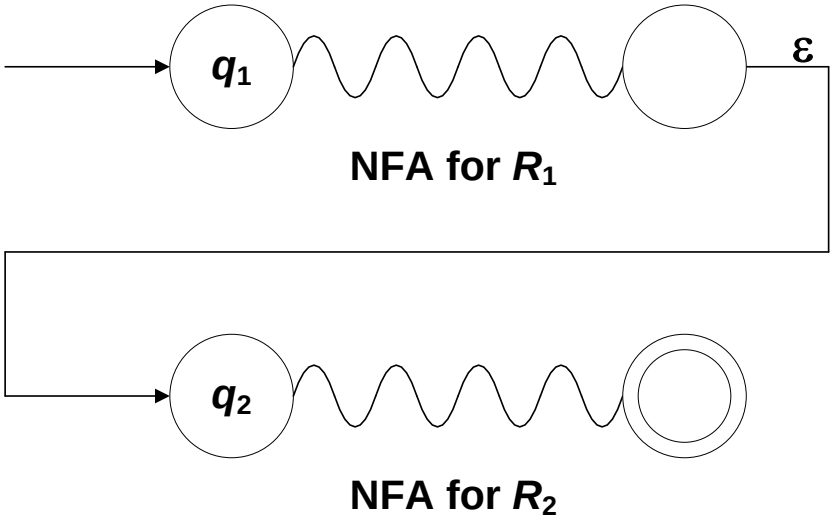


4. If R_1 and R_2 are regular expressions, then $R_1 + R_2$ is also a regular expression. The NFA that accepts $R_1 + R_2$ is



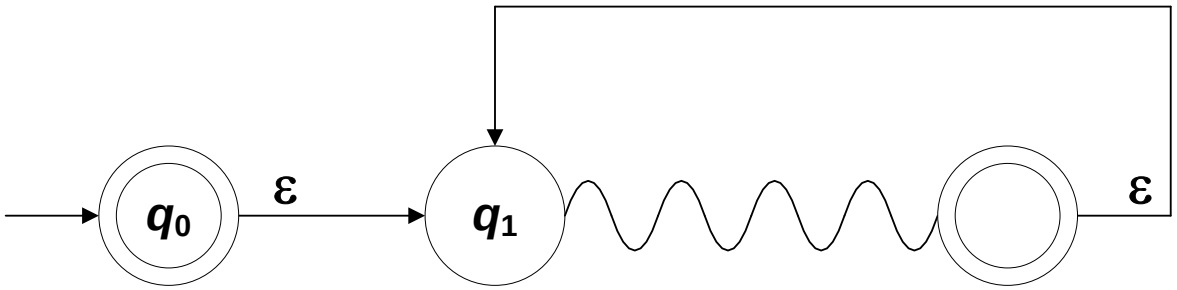
REGULAR EXPRESSIONS TO NFA

5. If R_1 and R_2 are regular expressions, then R_1R_2 is also a regular expression. The NFA that accepts R_1R_2 is



REGULAR EXPRESSIONS TO NFA

6. If R_1 is a regular expression, then R_1* is also a regular expression. The NFA that accepts R_1* is



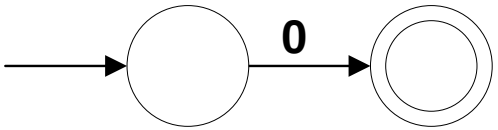
NFA for R_1

REGULAR EXPRESSIONS TO NFA

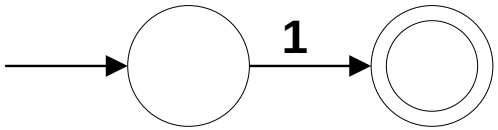
- Example:

  Convert the regular expression (0 + 1)* 01 to an NFA

  NFA for 0:

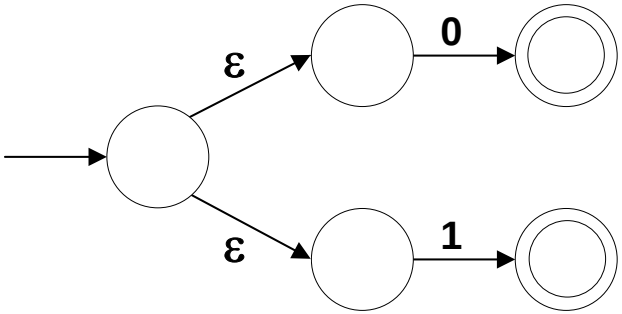


  NFA for 1:



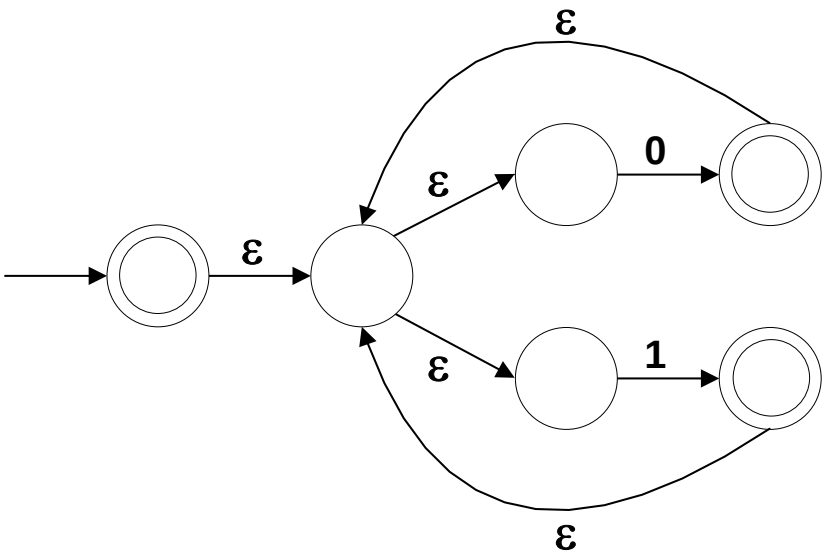
REGULAR EXPRESSIONS TO NFA

NFA for (0 + 1)



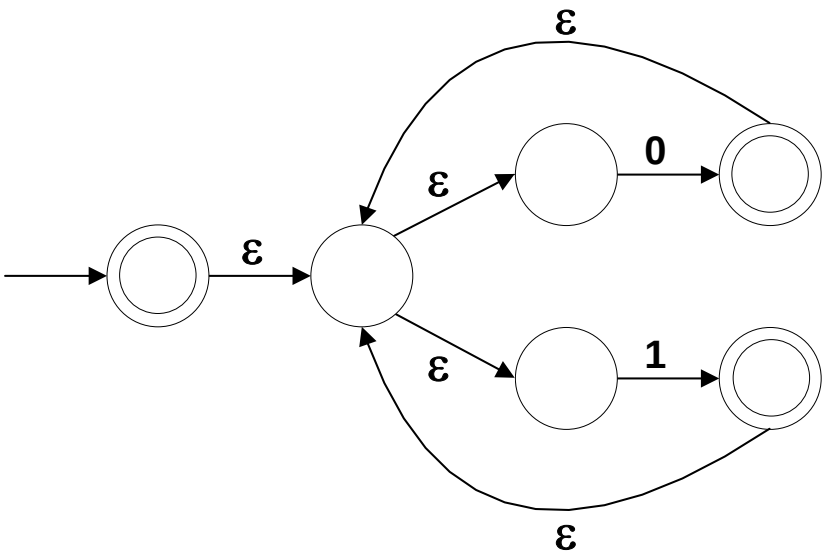
REGULAR EXPRESSIONS TO NFA

NFA for (0 + 1)*

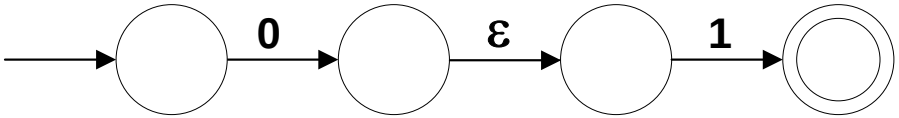


REGULAR EXPRESSIONS TO NFA

NFA for (0 + 1)*

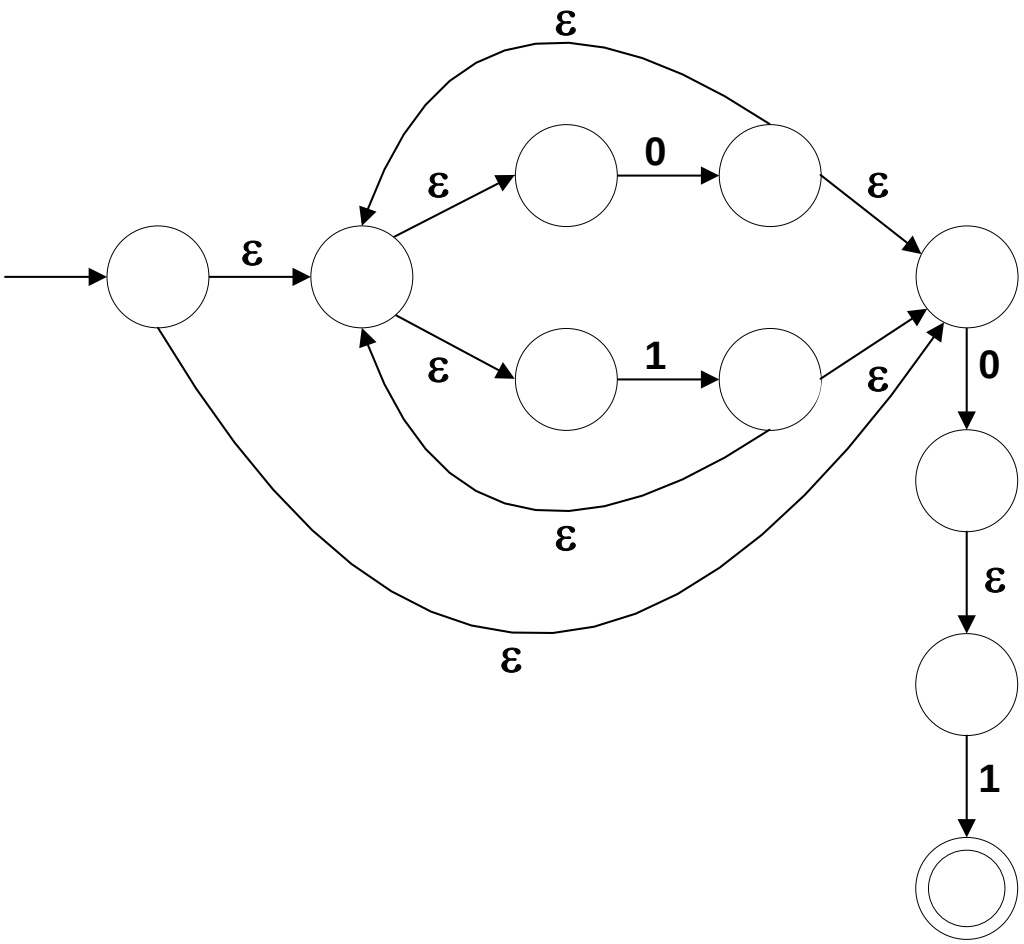


NFA for 01:



REGULAR EXPRESSIONS TO NFA

NFA for (0 + 1)*01



FINITE AUTOMATA TO REGULAR EXPRESSION

- The previous discussions have shown that if a language is described by a regular expression, then it is a regular language.

- This is because the regular expression describing a language can be converted to an NFA. And since the language now has an NFA, then it is a regular language.

- It will now be shown that if a language is regular, then there is a regular expression that describes it.

FINITE AUTOMATA TO REGULAR EXPRESSION

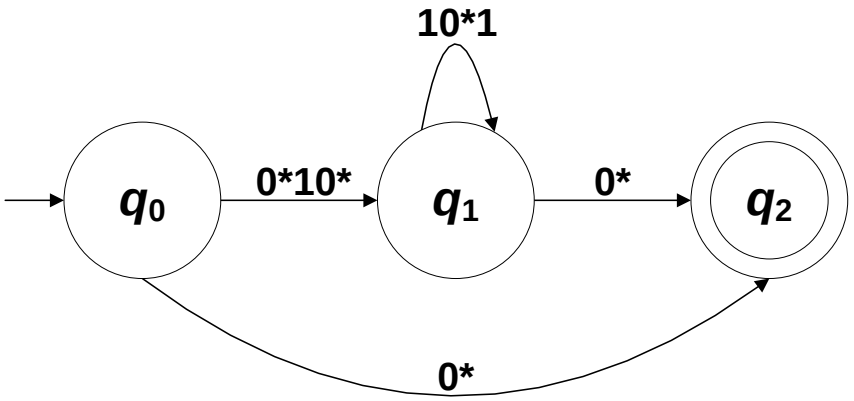
- The procedure that will be presented here is for the conversion of a DFA into a regular expression.
- If the given finite automaton is an NFA, simply convert it first to its equivalent DFA using the procedure outlined in previous lessons.
- Before discussing the conversion process, it is appropriate to first discuss a new type of finite automaton called a ***generalized nondeterministic finite automaton*** (GNFA).

FINITE AUTOMATA TO REGULAR EXPRESSION

- Generalized Nondeterministic Finite Automaton

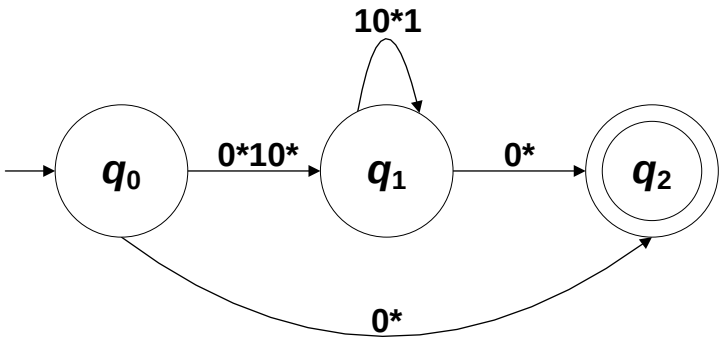
  A GNFA is an NFA in which each transition is labeled using a regular expression. An NFA can only have a single input symbol as label for each transition.

  Example:



FINITE AUTOMATA TO REGULAR EXPRESSION

- A GNFA may accept blocks of input symbols that match the regular expression on the outgoing transition before going to another state.



Example:

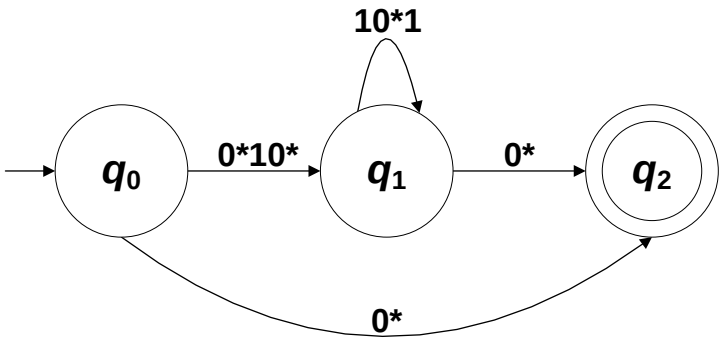
In the given example, the GNFA will go from state q_0 to q_1 if it receives a string that has exactly one 1 such as 1, 01, 010, and 000100.

If the GNFA is at state q_1, it will stay there if the input string has exactly two 1s such 11, 101, and 10001.

- Sample strings that the given GNFA will accept are the empty string ε, 00000, 11111, and 1111010000.

FINITE AUTOMATA TO REGULAR EXPRESSION

- Major Characteristics of a GNFA



1. It has a start state that has no incoming transitions from any other state (indegree = 0) but has an outgoing transition to every other state. The start state cannot be a final state.

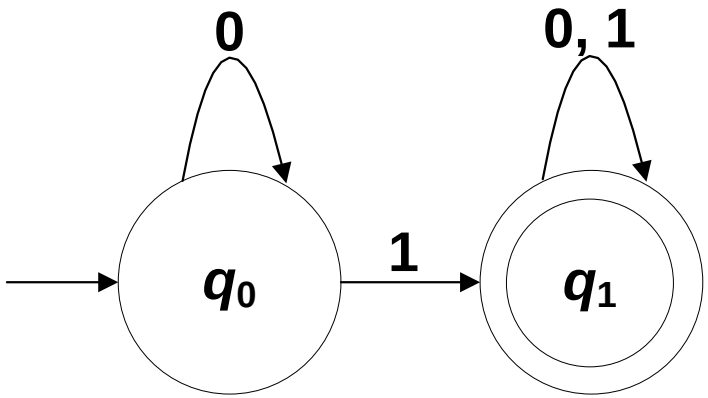
2. A GNFA has one final state with no outgoing transitions to any other state (outdegree = 0) and has an incoming transition from every other state.

3. Every state (excluding the start and final states) has one and only one outgoing transition to every other state (including itself).

FINITE AUTOMATA TO REGULAR EXPRESSION

- The first step in converting a DFA to a regular expression is to convert the DFA to a GNFA.

- Example: Convert the following DFA to a GNFA

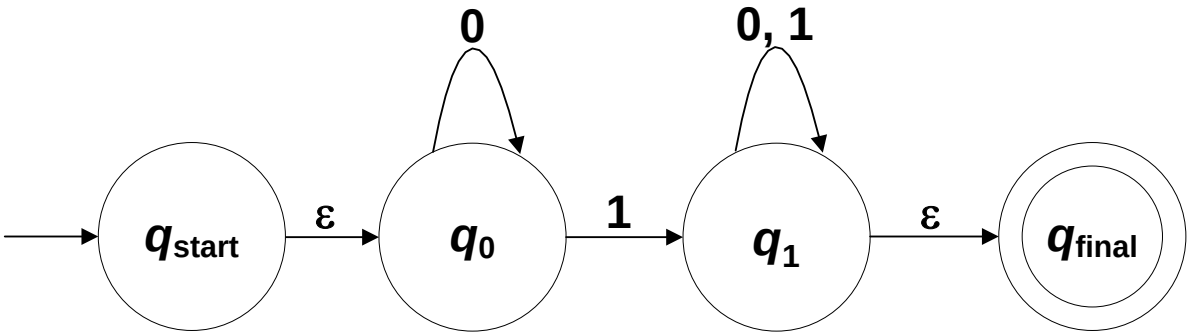


FINITE AUTOMATA TO REGULAR EXPRESSION

- Procedure to convert an DFA to GNFA:

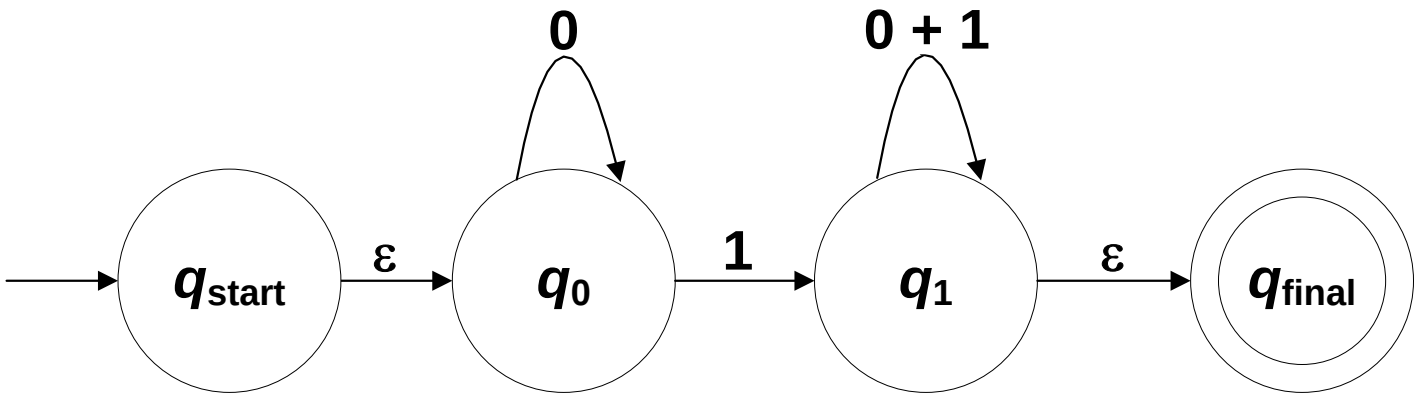
1. Create a new start state and add an ε-transition connecting this state to the original start state of the DFA.

2. Create a new final state and add ε-transitions connecting all final states of the DFA to this state. Convert all final states of the DFA to ordinary states.



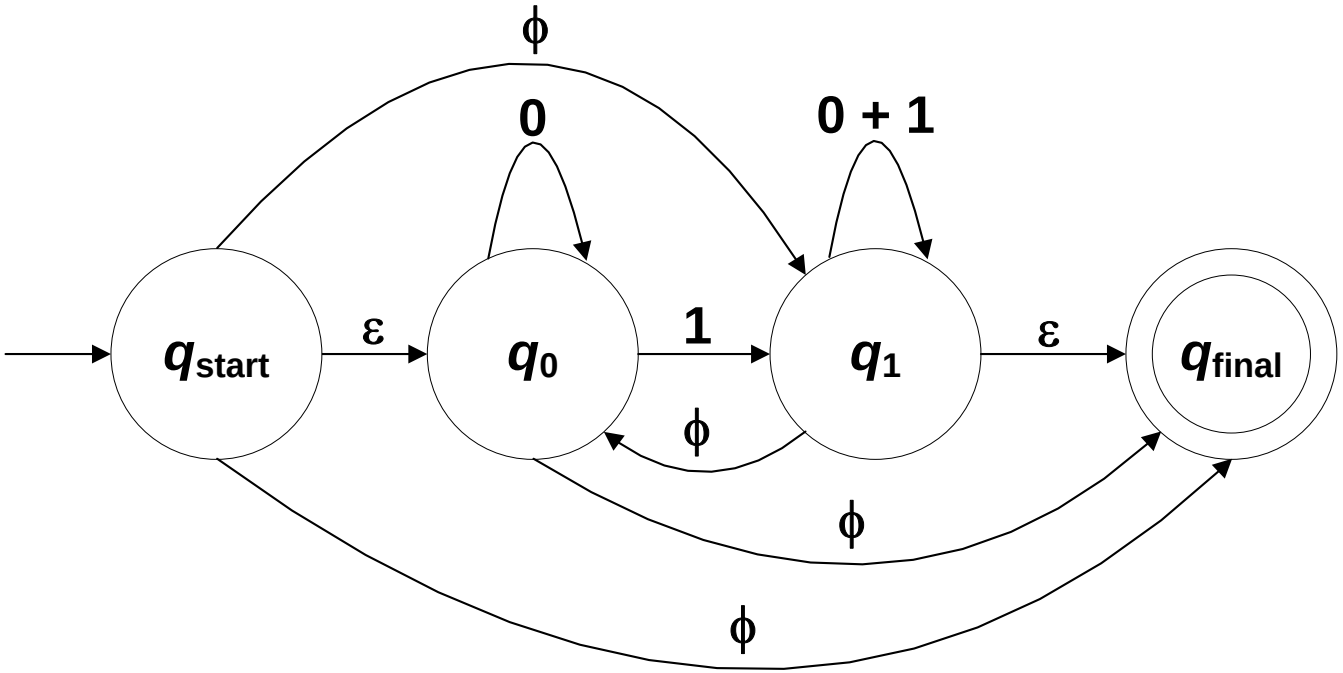
FINITE AUTOMATA TO REGULAR EXPRESSION

3. If a transition is labeled with multiple symbols, change the label to the union of the symbols.



FINITE AUTOMATA TO REGULAR EXPRESSION

4. States that are not connected must be connected with a transition whose label is ϕ.



FINITE AUTOMATA TO REGULAR EXPRESSION

- GNFA to Regular Expression

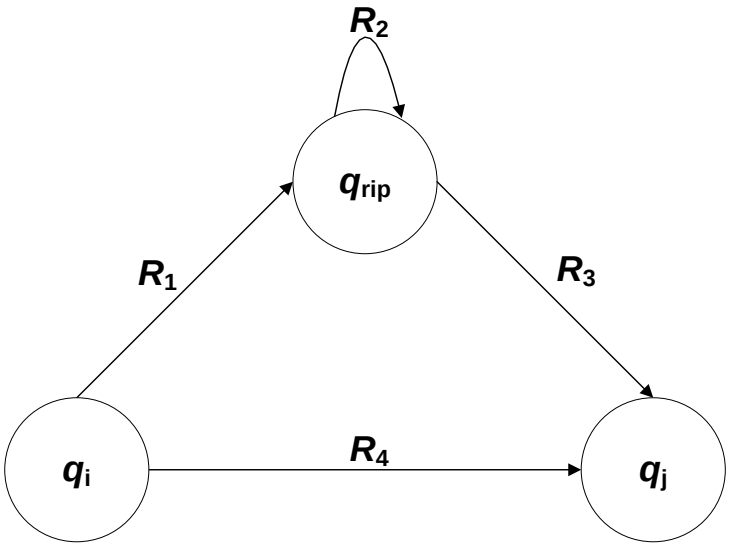
  After constructing the GNFA, the next step is to eliminate states until there are only two states left—the start and final states.

  The procedure will be as follows:

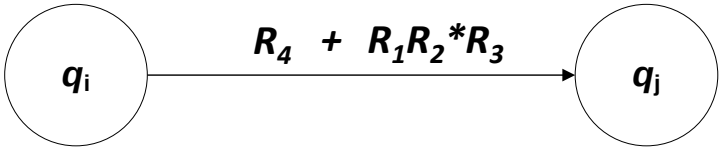
  1. If the number of states is greater than 2, reduce the total number of states by 1:

     - Choose a state to be removed. This state will be referred to as q_{rip}. The start or final state cannot be selected as q_{rip}.
     - Remove q_{rip} from the GNFA.

FINITE AUTOMATA TO REGULAR EXPRESSION



After removing q_{rip}:



- To make up for the removal of q_{rip}, reconnect the loose transitions. The GNFA must be repaired by changing the label of the remaining transition edges.

The revised labels must restore the lost computations caused by the removal of q_{rip}.

The new label going from a state q_i, to a state q_j is a regular expression that describes all strings that would take the machine from q_i to q_j either directly or via q_{rip}.

FINITE AUTOMATA TO REGULAR EXPRESSION

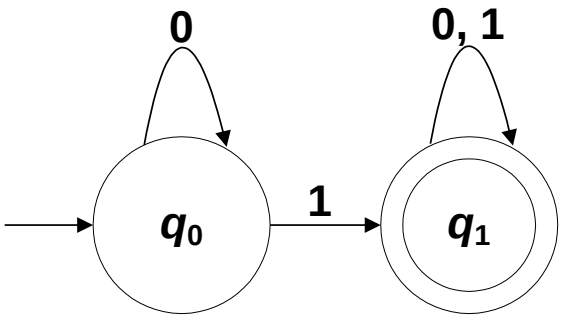
- Keep on eliminating states until there are only two states left.

2. If the number of states is equal to 2, which are the start and final states, then the conversion is complete.

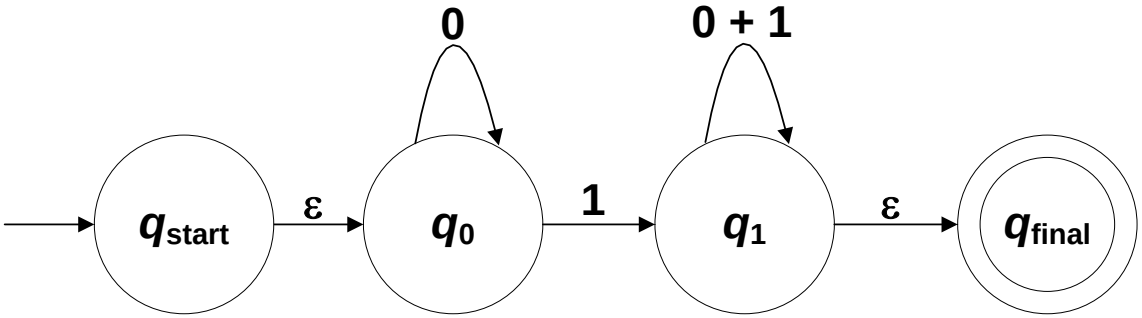
   The label of the transition edge connecting q_{start} to q_{final} is the regular expression that is equivalent to the given GNFA

FINITE AUTOMATA TO REGULAR EXPRESSION

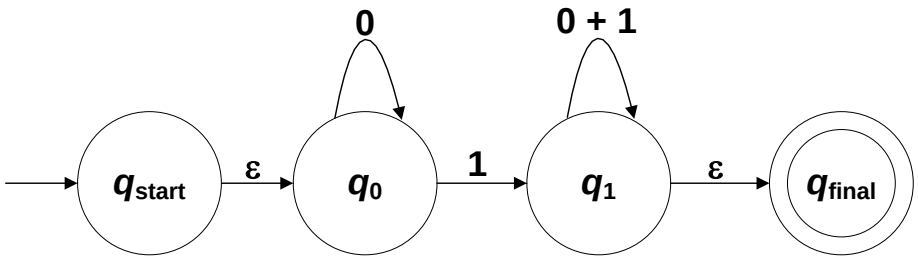
Example: Convert the following DFA into a regular expression



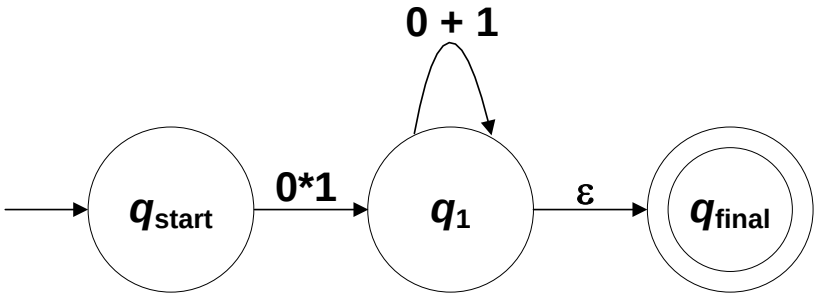
Converting the DFA into its equivalent GNFA



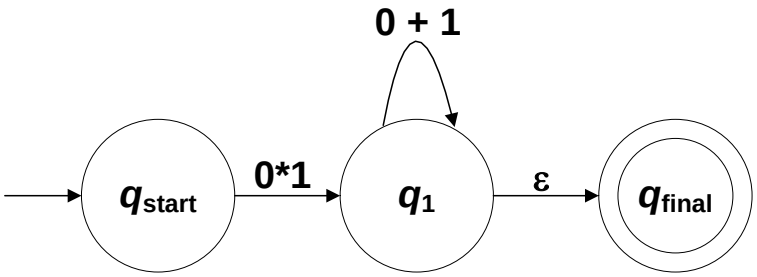
FINITE AUTOMATA TO REGULAR EXPRESSION



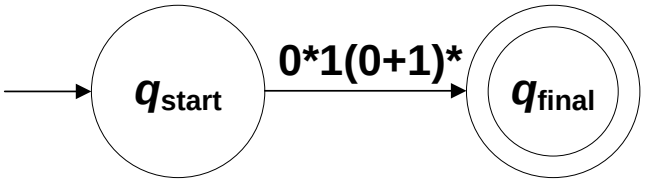
Removing q_0:



FINITE AUTOMATA TO REGULAR EXPRESSION

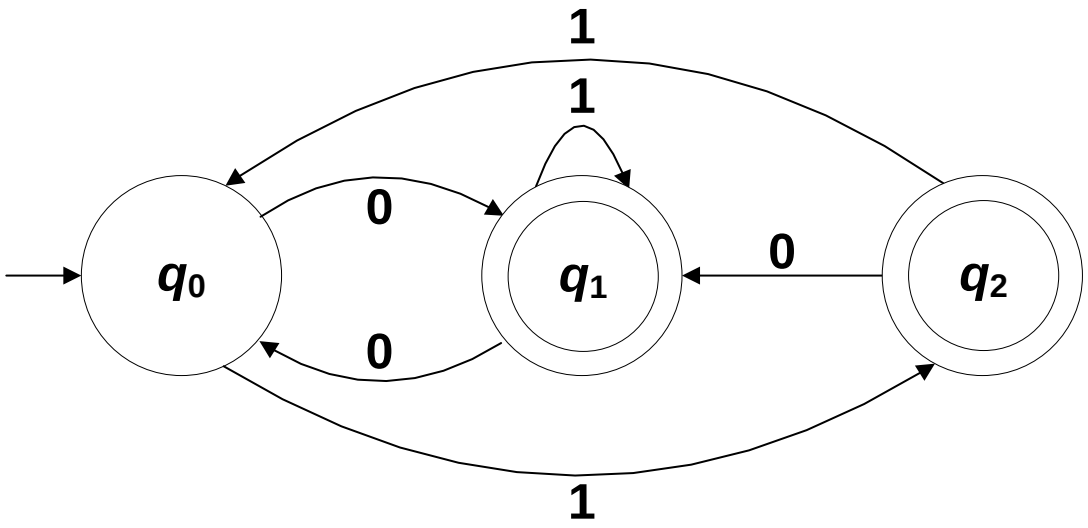


Removing q_1:



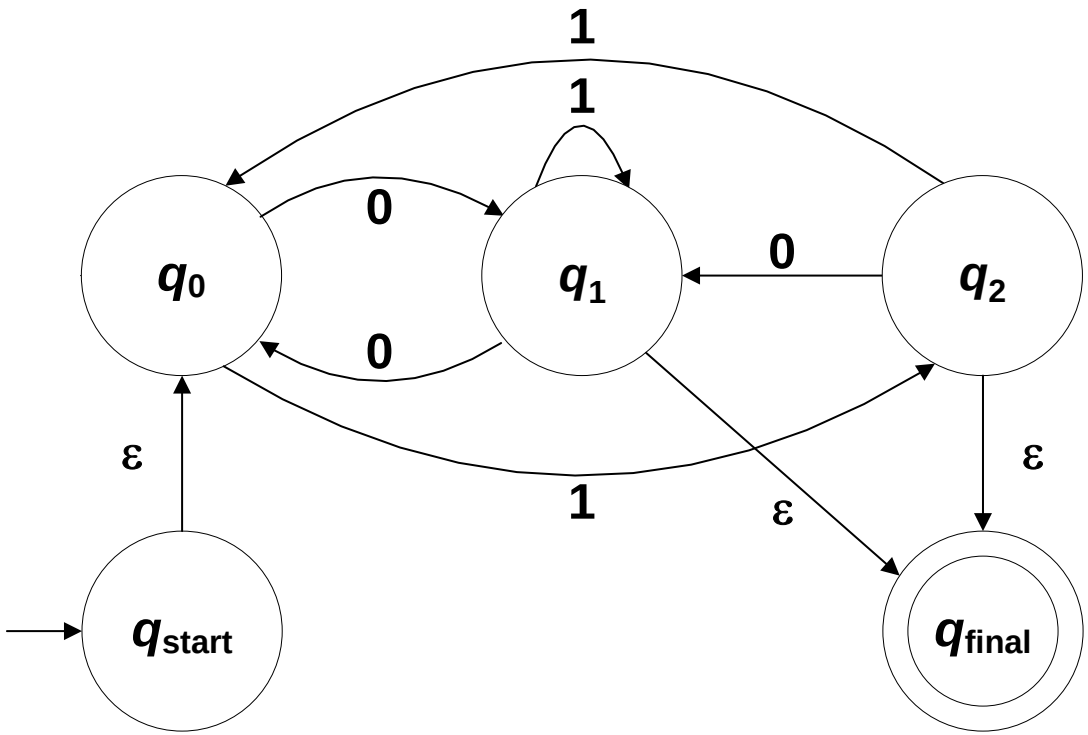
FINITE AUTOMATA TO REGULAR EXPRESSION

Example: Convert the following DFA into a regular expression

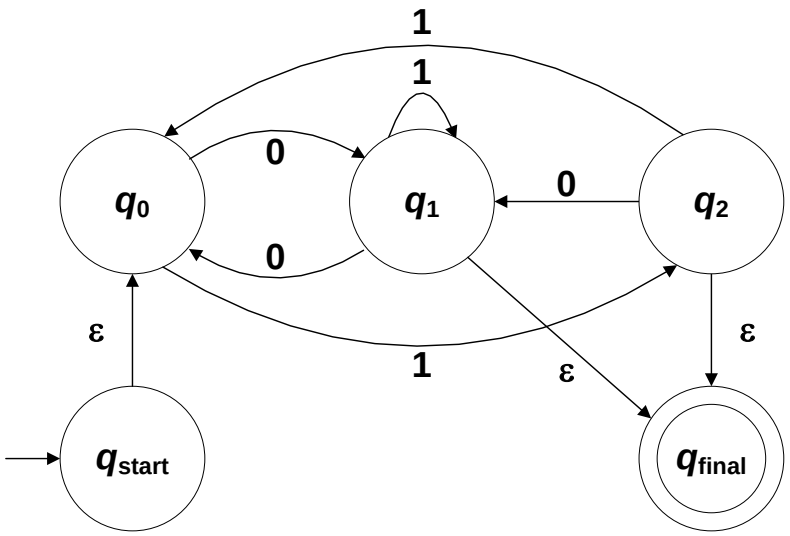


FINITE AUTOMATA TO REGULAR EXPRESSION

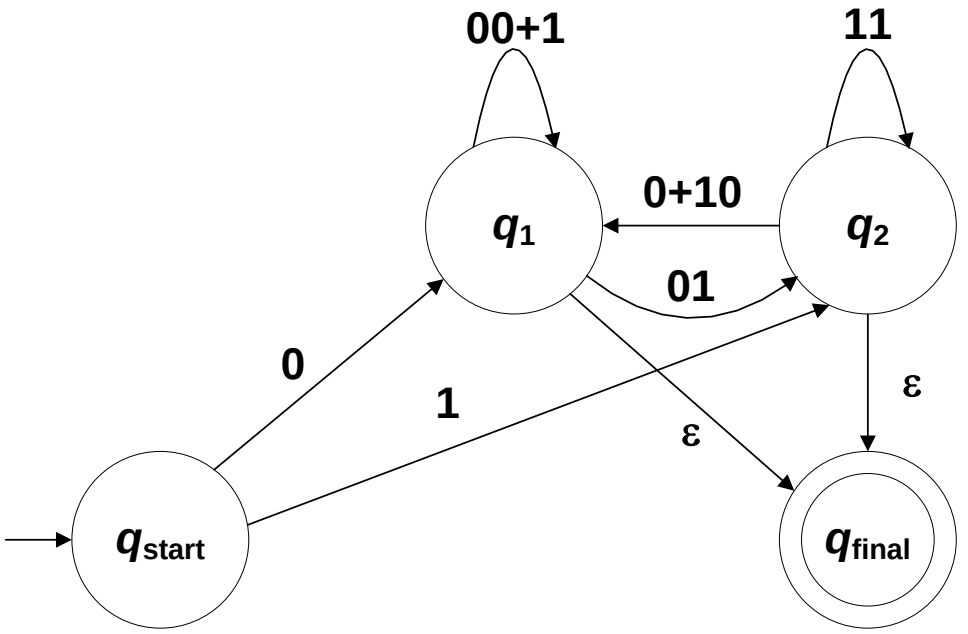
Converting the DFA into its equivalent GNFA:



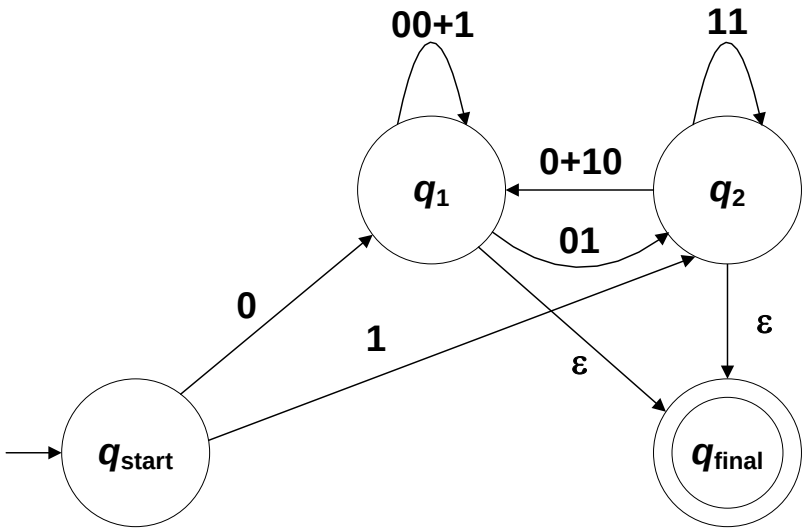
FINITE AUTOMATA TO REGULAR EXPRESSION



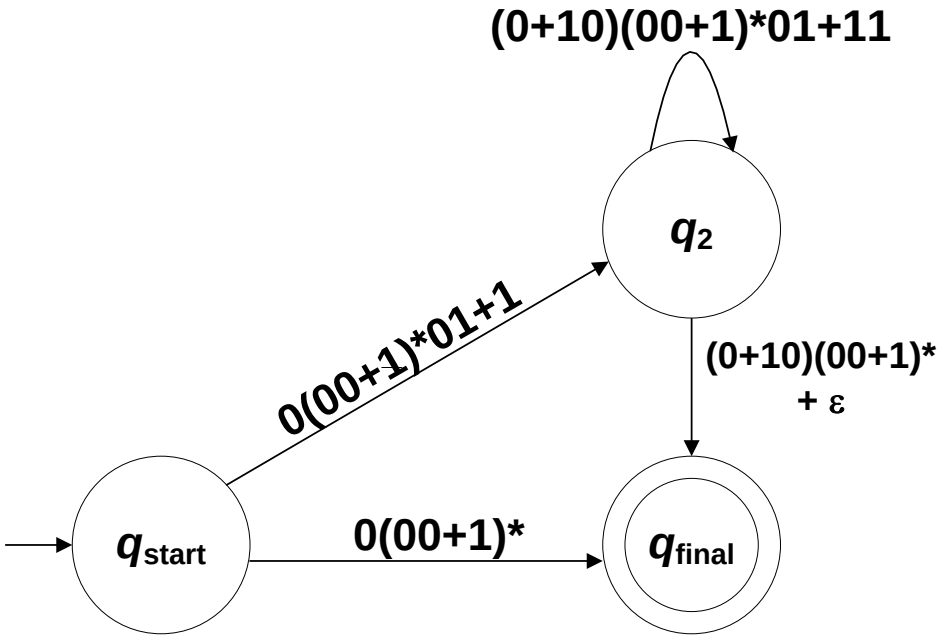
Removing q_0:



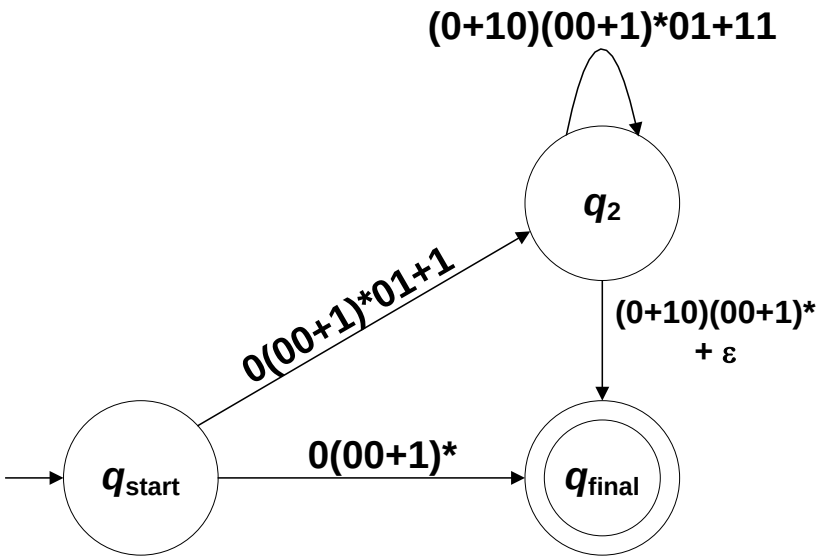
FINITE AUTOMATA TO REGULAR EXPRESSION



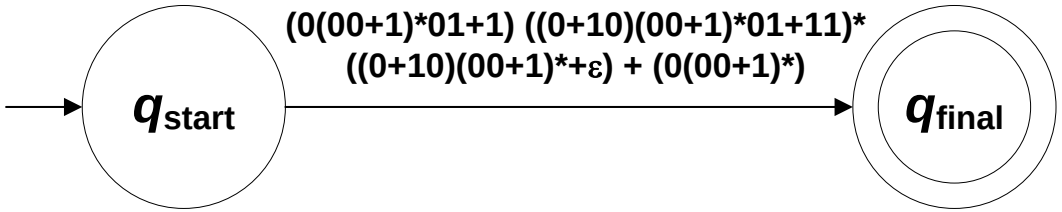
Removing q_1:



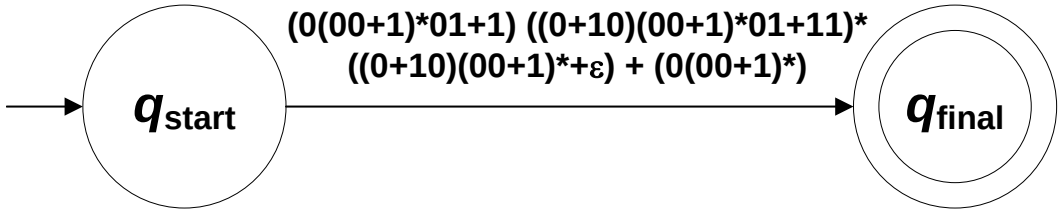
FINITE AUTOMATA TO REGULAR EXPRESSION



Removing q_2:



FINITE AUTOMATA TO REGULAR EXPRESSION



The expression for the DFA is therefore

$$R = (0(00+1)^*01+1)\ ((0 + 10)(00 + 1)^*01 + 11)^*\ ((0+10)(00+1)^*+\varepsilon) +$$

$$(0(00+1)^*)$$

FINITE AUTOMATA TO REGULAR EXPRESSION

- Theorem 5:

  A language is regular if and only if some regular expression describes it.

- Lemma 5.1

  If a language is described by a regular expression, then it is regular.

- Lemma 5.2

  If a language is regular, then it is described by a regular expression.

EXERCISES

- Derive the regular expression for each of the following languages:

1. The set of all strings ending with 11.
2. The set of all strings not ending with 11.
3. The set of all strings with alternating 0s and 1s.
4. The set of all strings with at most one pair of consecutive 1s.
5. The set of all strings not containing the substring 101.
6. The set of all strings with an even number of 0s.

EXERCISES

- Convert the regular expression (1*01)*1*(0 + ε) into an NFA.

- Convert the following DFAs into a single regular expression: