Lecture 4 - Regular Languages

Eric A. Autry

Hurricane

Class canceled on Thursday

Office Hours: the 4-8 pm hours on Wednesday are canceled

Homework due next Thursday

High winds and flooding possible. Make sure you have prepared for potentially dangerous high winds, loss of power, and loss of water for multiple days.

Last time: Regular Expressions

This time: Converting DFAs to Regular Expressions

Next time: Nonregular Languages

Homework 2 on Sakai and due **next THURSDAY the 20th!**

Quiz 0 on Sakai

Final Warning! I will be checking submissions after class today.

Regular Languages VS Regular Expressions

Theorem

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

DFAs, NFAs, and Regular Expressions are all equivalent!

Equivalence of NFAs and Regular Expressions

Let's prove that theorem...

Proof ($\Leftarrow$): Assume that the regular expression R describes some language $L(R)$. We want to prove that this language is regular.

- How?
- Build an NFA.

Equivalence of NFAs and Regular Expressions

We say that R is a regular expression if R is one of the following:

1. a for some a in the alphabet Σ (i.e., a single symbol)
2. ε (this represents the language with just the empty string)
3. $\varnothing$ (this is the empty language)
4. $(R_1 \mid R_2)$, where R_1 and R_2 are regular expressions
5. $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions
6. (R_1^*), where R_1 is a regular expression

We built NFAs for each of the above.

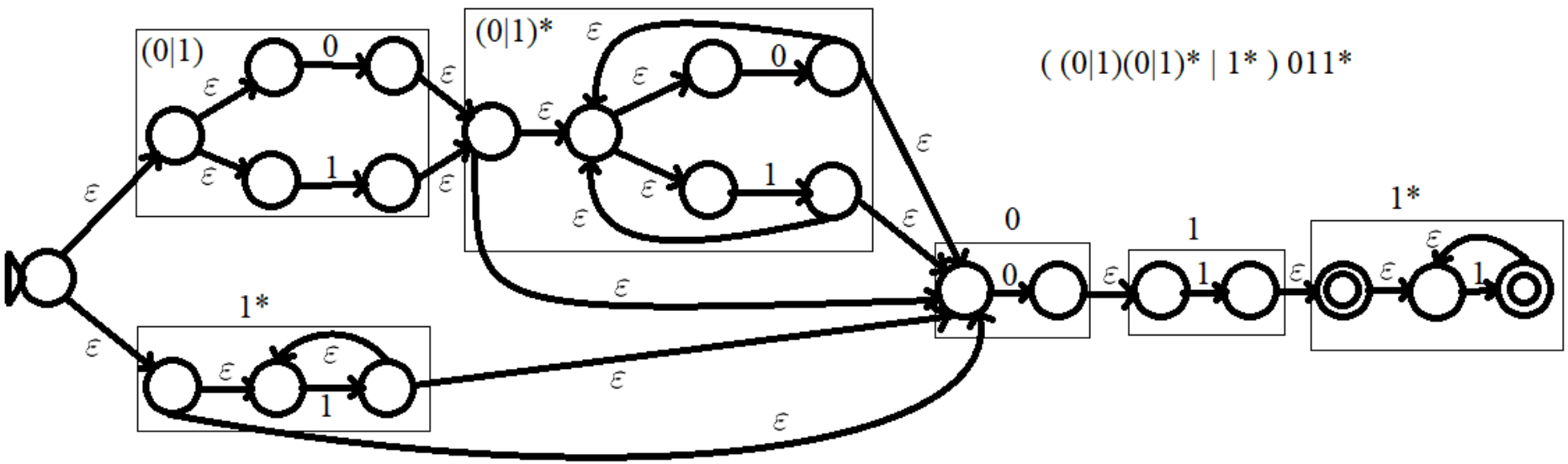
Now we combine them. Example:

$$\left(\, (0|1)^+ \mid 1^* \, \right) 01^+ = \left(\, (0|1)(0|1)^* \mid 1^* \, \right) 011^*$$

Equivalence of NFAs and Regular Expressions

Now we can combine these machines for more complicated expressions. Example:

$$\left(\, (0|1)^{+} \, | \, 1^{*} \, \right) 01^{+} = \left(\, (0|1)(0|1)^{*} \, | \, 1^{*} \, \right) 011^{*}$$



Equivalence of NFAs and Regular Expressions

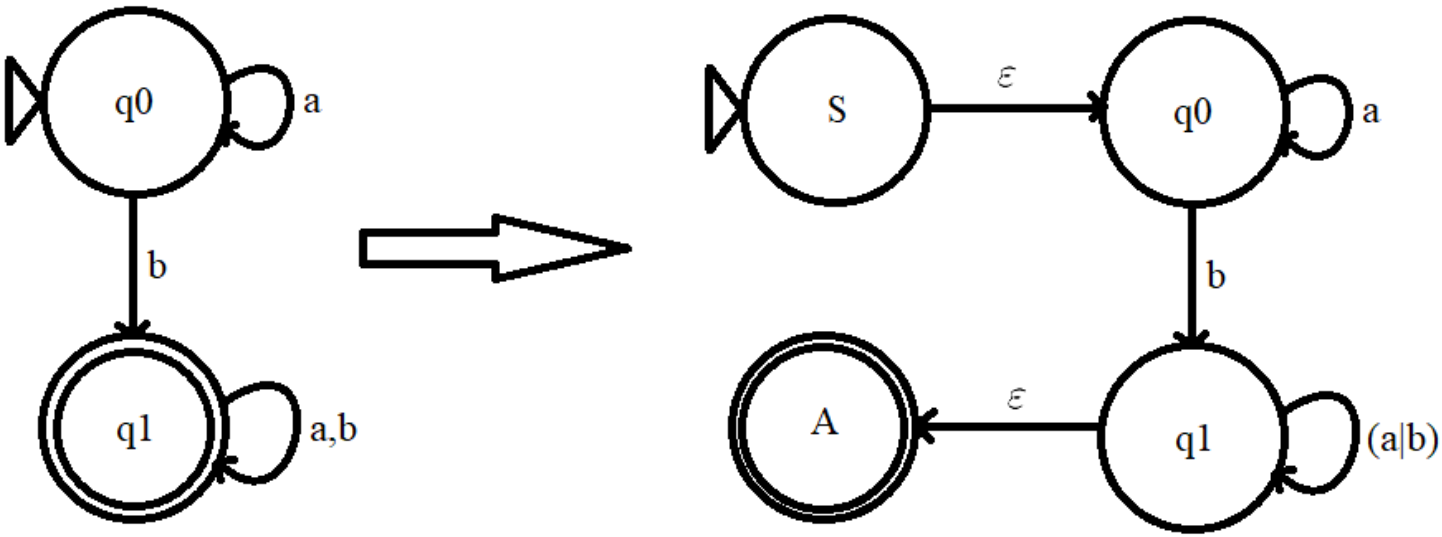
Let's prove that theorem...

Proof $(\Rightarrow)$: Assume that the language A is regular. We want to show that this means there exists a regular expression R that describes it, i.e., $L(R) = A$.

- Since language A is regular, it means that there exists a DFA that recognizes it.
- Let's create a procedure for converting a DFA into a regular expression.

Converting DFAs to Regular Expressions

Step 1: We will make a generalized nondeterministic finite automaton (GNFA) that has 1 start state, 1 accepting state, and allows transitions to be regular expressions instead of just single symbols.



We added the two new states with ε transitions.

Note the new $(a|b)$ transition.

Converting DFAs to Regular Expressions

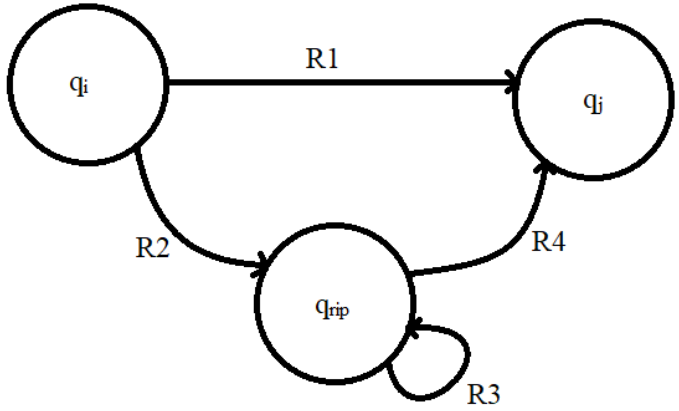
Step 2: We will eliminate one state at a time until we have a single transition from the start state S to the accept state A.

The regular expression for that single transition will be the regular expression corresponding to the language recognized by the machine!

How do we remove states?

Converting DFAs to Regular Expressions

How do we remove states?



Say we want to remove state q_{rip}. We find all paths from state q_i to q_j that go through q_{rip} and convert them to a regular expression:

- To get from q_i to q_{rip}, we must see a string matching R_2.
- We can then see any number of strings matching R_3 to return to q_{rip}, i.e., this represents $(R_3)^*$.
- We can then get to q_j by seeing a string matching R_4.

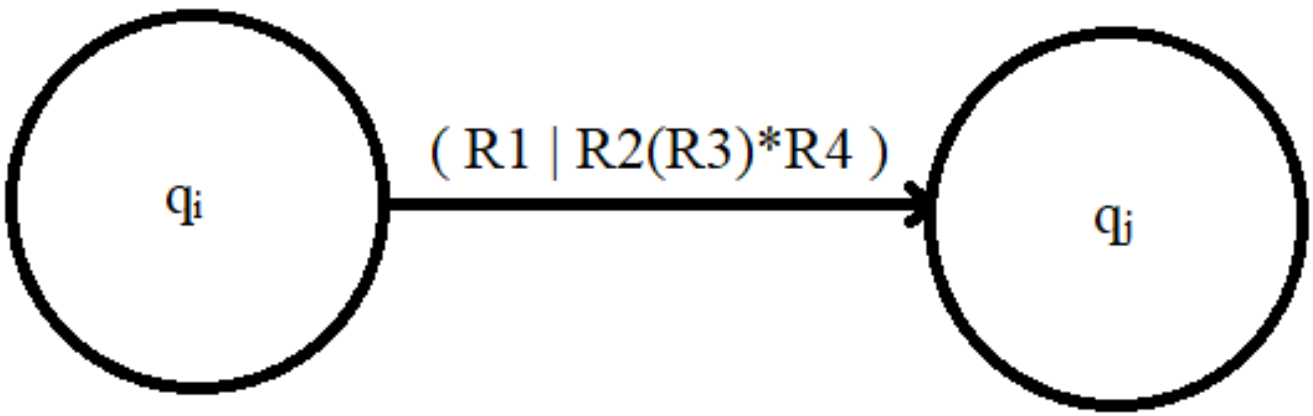
Converting DFAs to Regular Expressions

How do we remove states?

We put it all together to see that the path from q_i to q_j going through state q_{rip} corresponds to strings that match:

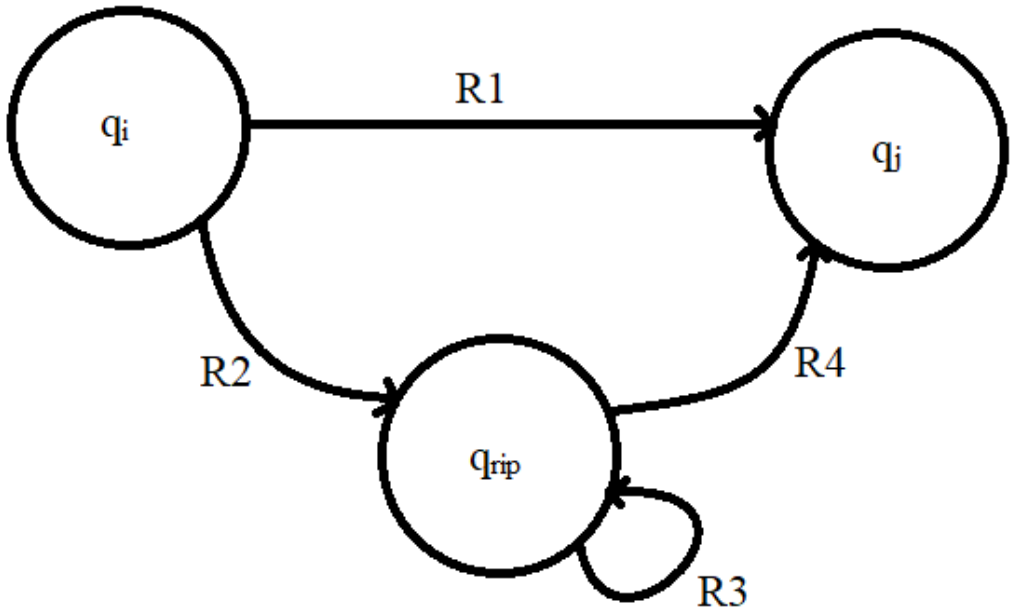
$$R_2(R_3)^*R_4.$$

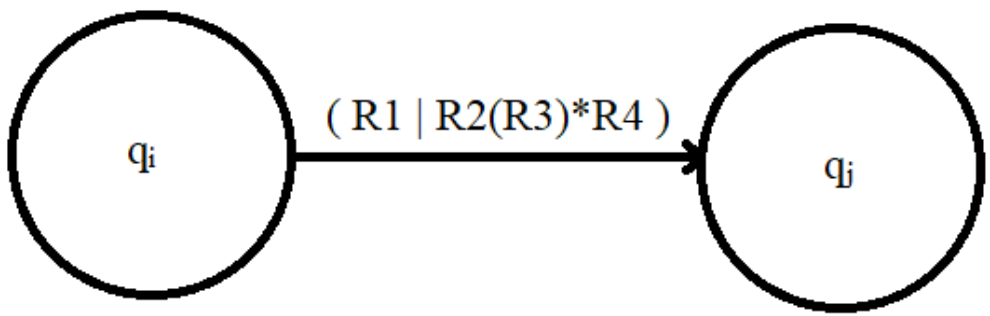
Finally, we take the union of this expression and the expression that transitions directly from q_i to q_j, which was R_1.



Converting DFAs to Regular Expressions

How do we remove states?

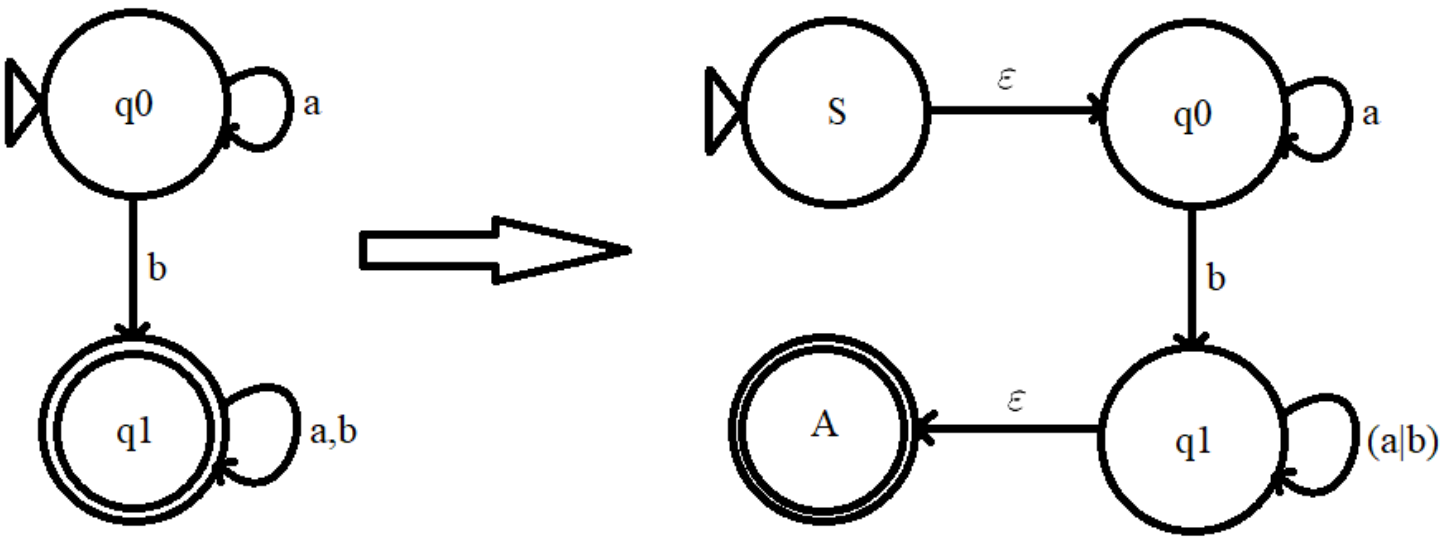




Converting DFAs to Regular Expressions

Back to our example...

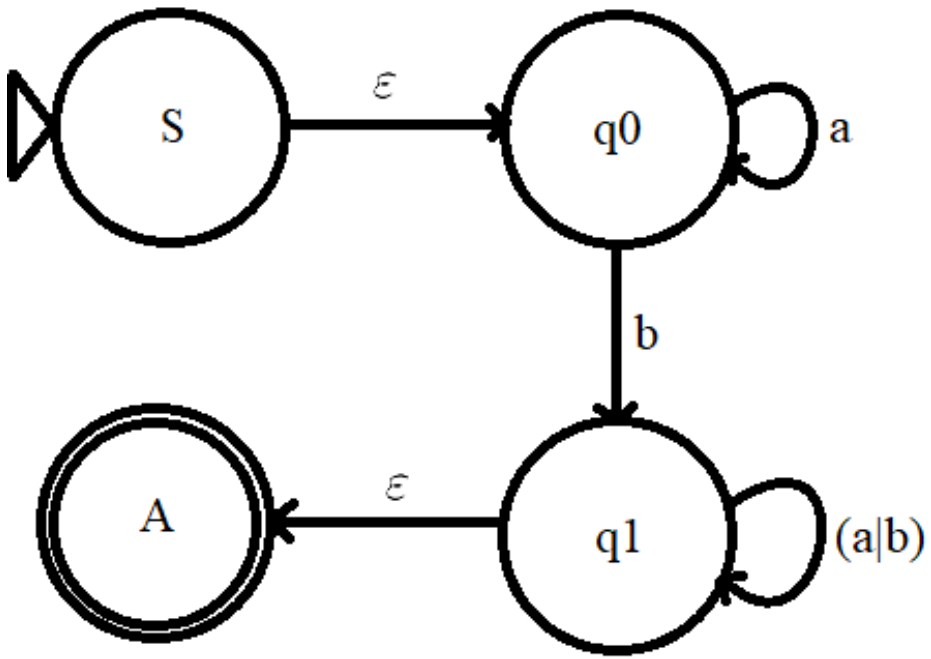
Step 1: Create a GNFA with 1 start state and 1 accept state.



Converting DFAs to Regular Expressions

Back to our example...

Step 2: Remove states one at a time until there is a single transition between S and A.



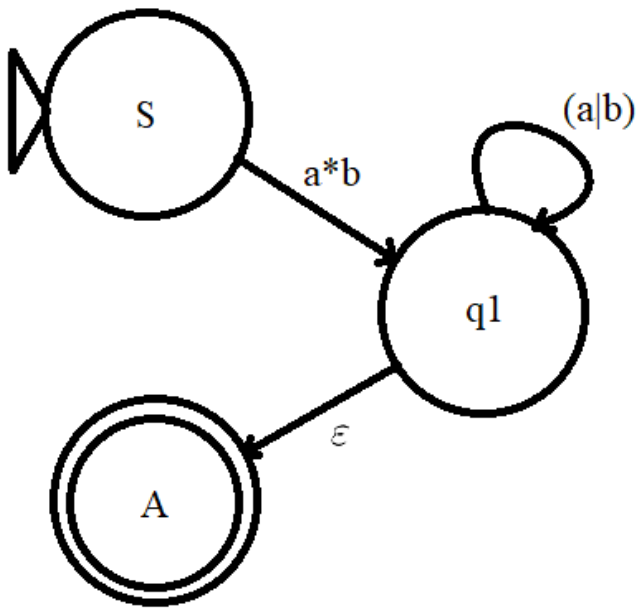
Converting DFAs to Regular Expressions

Step 2: Remove states one at a time until there is a single transition between S and A.

Let's remove q_0.

- There is now a connection from S to q_2, that transitions on the regular expression:

$$a^*b$$



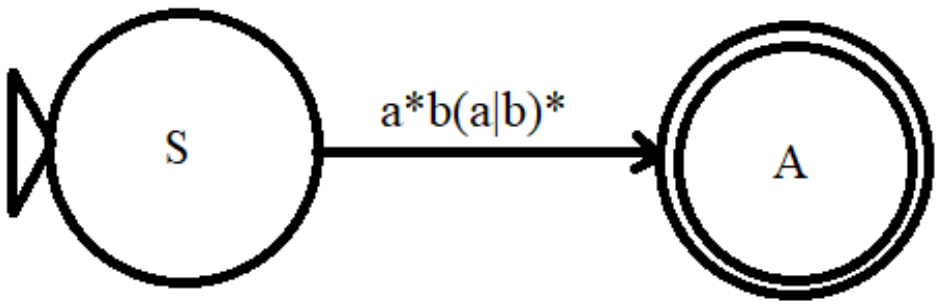
Converting DFAs to Regular Expressions

Step 2: Remove states one at a time until there is a single transition between S and A.

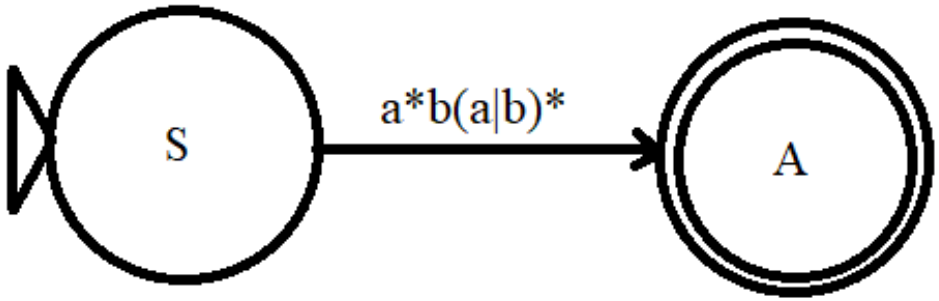
Let's remove q_1.

- There is now a connection from S to A, that transitions on the regular expression:

$$a^*b(a|b)^*$$



Converting DFAs to Regular Expressions

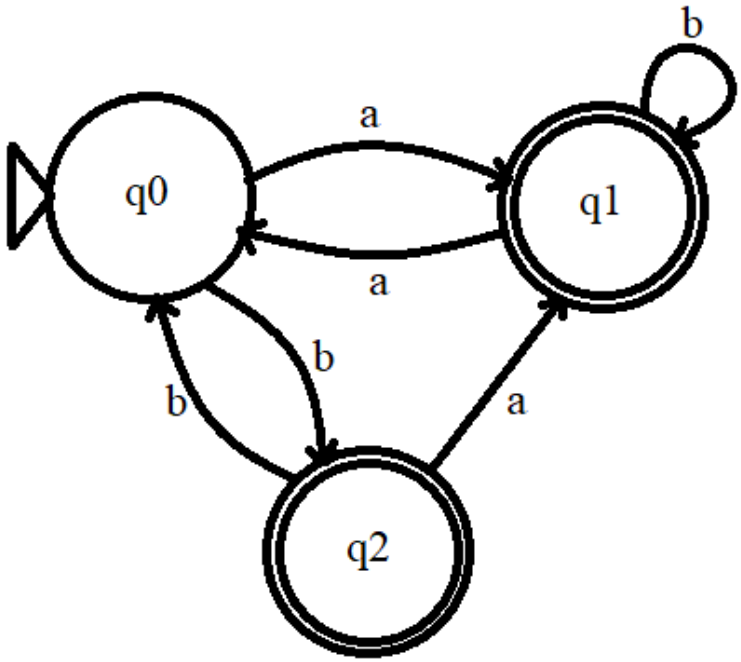


We have only a single transition between start and accept, so the regular expression for our DFA is:

$$a^*b(a|b)^*$$

Converting DFAs to Regular Expressions

Let's try another example:

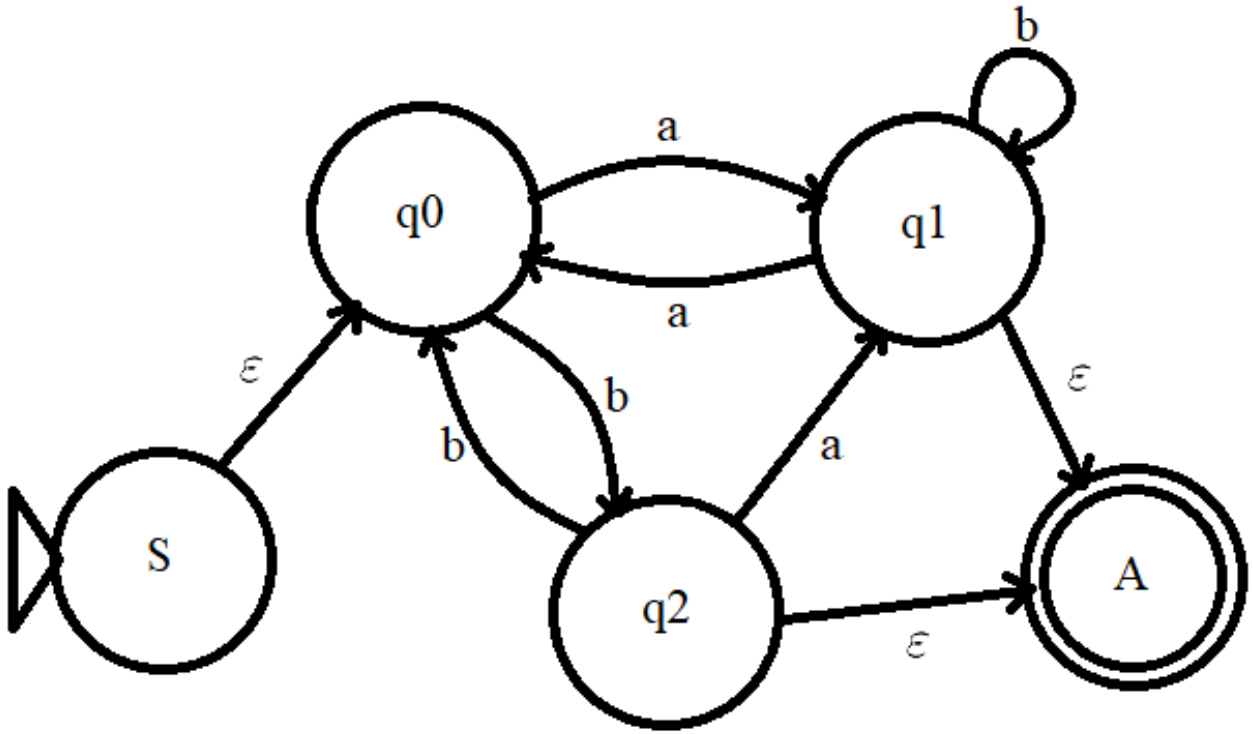


Step 1: Create a GNFA with 1 start state and 1 accept state.

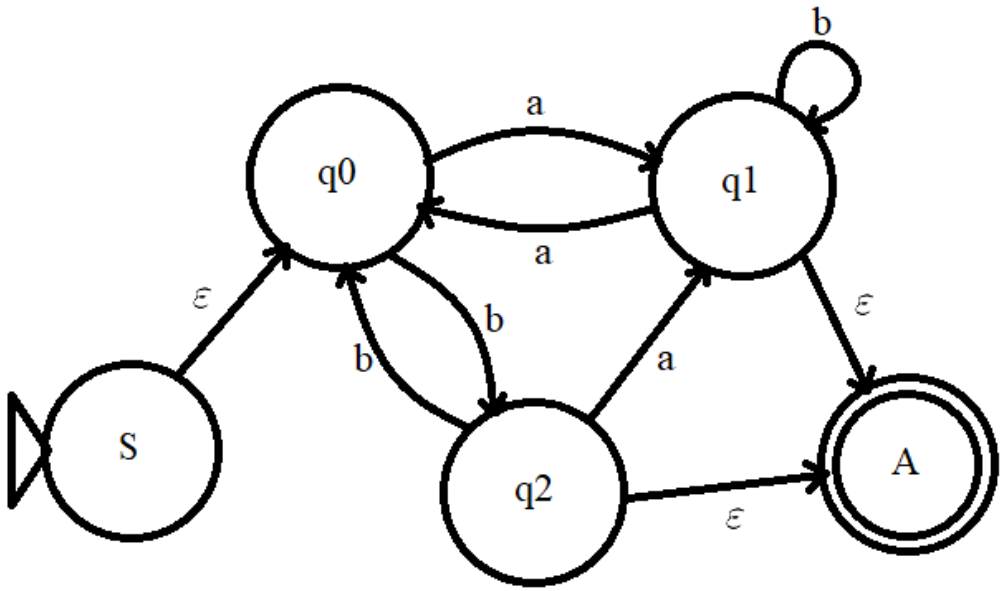
Step 2: Remove states one at a time until there is a single transition between S and A.

Converting DFAs to Regular Expressions

Step 1: Create a GNFA with 1 start state and 1 accept state.



Converting DFAs to Regular Expressions



Step 2: Remove states one at a time until there is a single transition between S and A.

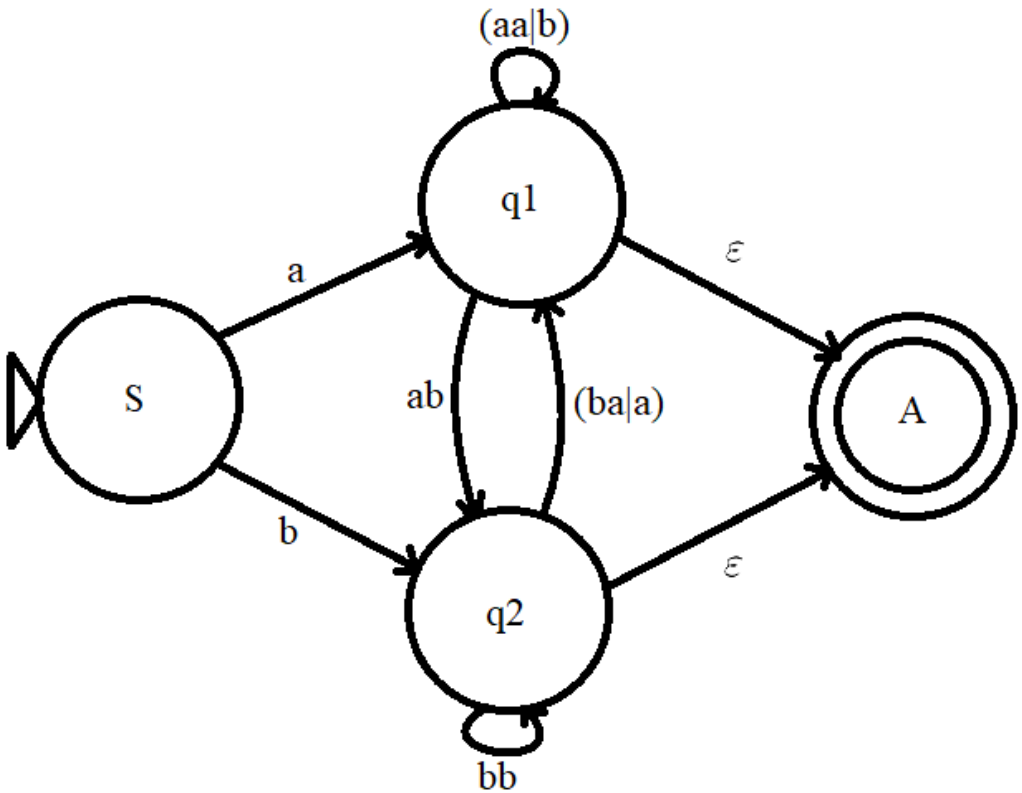
Now let's remove state q_0, updating the following transitions:

- $S \to q_1$: a
- $S \to q_2$: b
- $q_1 \to q_2$: ab
- $q_2 \to q_1$: $(ba|a)$
- $q_1 \to q_1$: $(aa|b)$
- $q_2 \to q_2$: bb

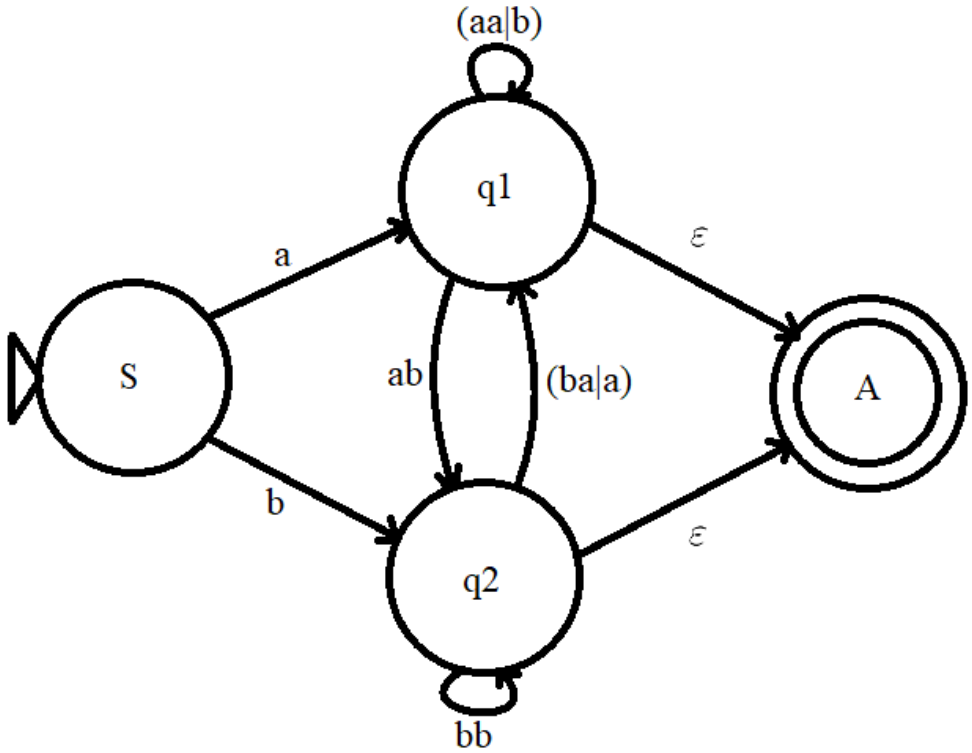
Converting DFAs to Regular Expressions

Now let's remove state q_0, updating the following transitions:

- $S \to q_1$: a
- $S \to q_2$: b
- $q_1 \to q_2$: ab
- $q_2 \to q_1$: $(ba|a)$
- $q_1 \to q_1$: $(aa|b)$
- $q_2 \to q_2$: bb



Converting DFAs to Regular Expressions



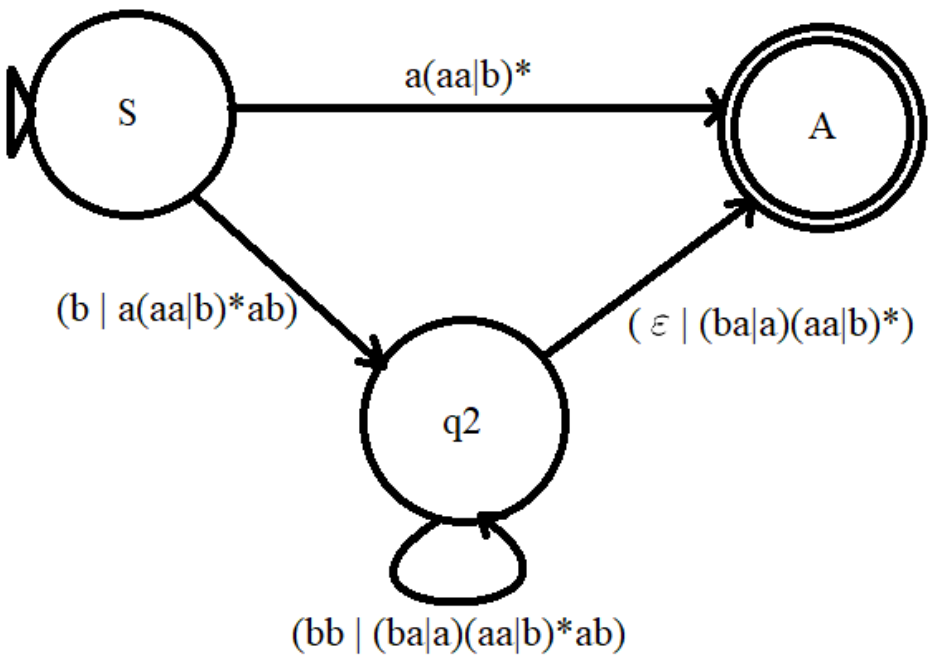
Now let's remove state q_1, updating the following transitions:

- $S \to A$: $a(aa|b)^*$
- $S \to q_2$: $(b \mid a(aa|b)^*ab)$
- $q_2 \to A$: $(\varepsilon \mid (ba|a)(aa|b)^*)$
- $q_2 \to q_2$: $(bb \mid (ba|a)(aa|b)^*ab)$

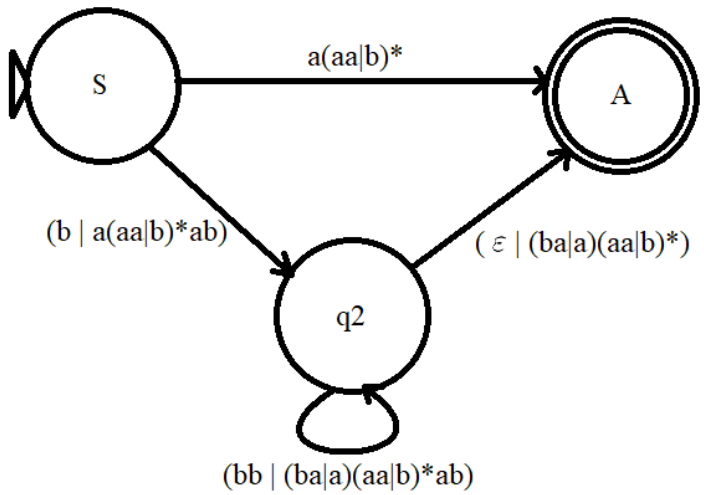
Converting DFAs to Regular Expressions

Now let's remove state q_1, updating the following transitions:

- $S \to A$: $a(aa|b)^*$
- $S \to q_2$: $(b \mid a(aa|b)^*ab)$
- $q_2 \to A$: $(\varepsilon \mid (ba|a)(aa|b)^*)$
- $q_2 \to q_2$: $(bb \mid (ba|a)(aa|b)^*ab)$

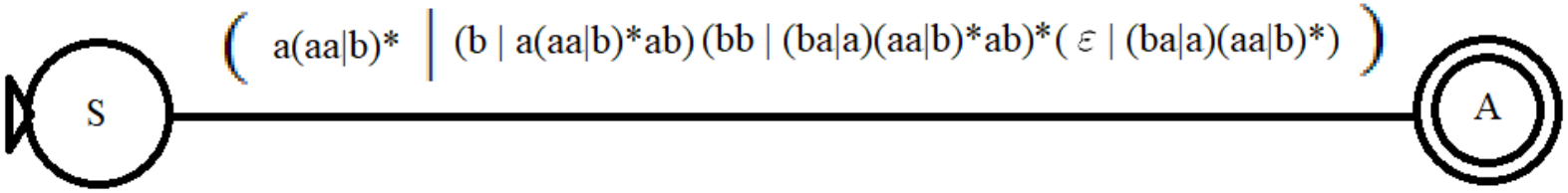


Converting DFAs to Regular Expressions



Now let's remove state q_2 and update our machine with the single transition (and therefore our regular expression):

$$(a(aa|b)^* \mid (b \mid a(aa|b)^*ab)\,(bb \mid (ba|a)(aa|b)^*ab)^*\,(\varepsilon \mid (ba|a)(aa|b)^*))$$



Converting DFAs to Regular Expressions

Does it matter what order we eliminate states in?

No. We will be able to obtain a regular expression for the machine regardless of the order in which we eliminate states.

Eliminating states in different order **can** result in different regular expressions. The resulting expressions will be **equivalent**, i.e., they will all describe the same language.

Regular Languages

Theorem

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

DFAs, NFAs, and Regular Expressions are all equivalent!

Regular Languages

Question: are all languages regular?

Ex:

$$A = \{w \mid w \text{ has the same number of } 1\text{s and } 0\text{s}\}.$$

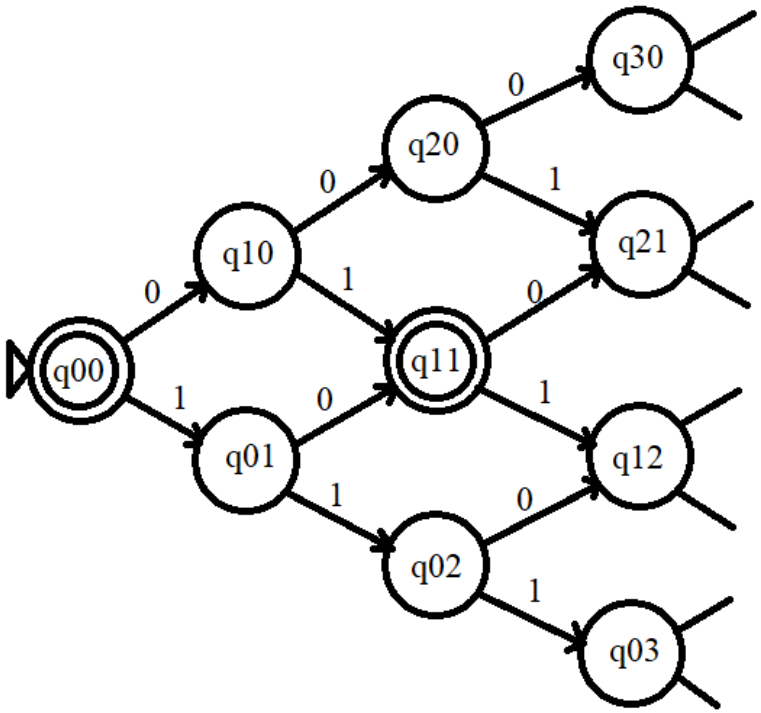
How can we build a machine to recognize this language?

Regular Languages

$$A = \{w \mid w \text{ has the same number of } 1\text{s and } 0\text{s}\}.$$

Attempt 1: Let's try to keep track of both the number of 0s and the number of 1s.

Define the state $q_{ij} = i$ number of 0s and j number of 1s.



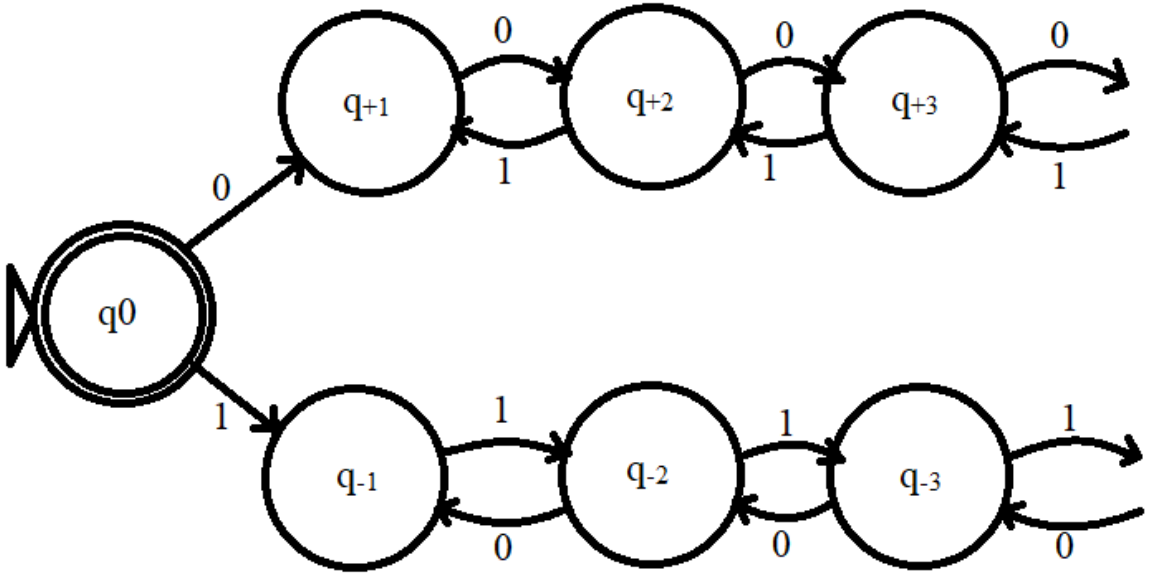
Infinite number of states! No longer a **finite** state machine...

Regular Languages

$$A = \{w \mid w \text{ has the same number of } 1\text{s and } 0\text{s}\}.$$

Attempt 2: Let's try to keep track of how many more 0s than 1s.

Define the state $q_{+i} = i$ more 0s than 1s, and the state $q_{-i} = i$ fewer 0s than 1s.



Still an **infinite** number of states!

(Note: same number of states as the first attempt)

Nonregular Languages

Some languages are not regular languages and cannot be recognized by any DFAs, NFAs, or Regular Expressions.

Or rather:

MOST languages are not regular languages and cannot be recognized by any DFAs, NFAs, or Regular Expressions.

Nonregular Languages

Can we prove if a language is regular or not?

Yes.

- To prove that it is, build a DFA, NFA, or Regular Expression that recognizes it.
- To prove that it is not, use the Pumping Lemma...
  - We will not cover this lemma in this overview of automata theory.

Next time: Nonregular languages and the machines that recognize them.