Lecture 3 - Regular Expressions

Eric A. Autry

Last time: NFAs

This time: Regular Expressions

Next time: Equivalence of DFAs, NFAs, Regular Expressions

Homework 1 on Sakai and due today!

Homework 2 will be posted tonight and due Thursday the 13th.

Quiz 0 on Sakai

Due today!

Regular Languages

Theorem

A language is regular if and only if there exists a nondeterministic finite automaton that recognizes it.

Proof $(\Rightarrow)$: Assume that a language A is regular. Then it is recognized by some DFA that we can name M. Every DFA is also an NFA, and therefore there exists an NFA that recognizes the language A.

Proof $(\Leftarrow)$: Assume that we have an NFA M_1 that recognizes the language A, i.e., $L(M_1) = A$. We must now prove that the language A is regular.

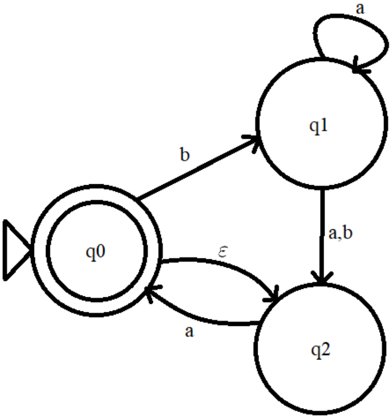
- How to prove that a language is regular?
- Let's use the NFA to build a corresponding DFA.

Equivalence of DFAs and NFAs

- To complete our proof, we would want to prove that for every NFA M_1, there exists a corresponding DFA M_2 that recognizes the same language, i.e. $L(M_1) = L(M_2)$.
- To do this, we would describe a procedure for how to build the DFA M_2 given the NFA M_1.
- For the sake of time, we will be lazy and demonstrate this procedure through an example and convince ourselves that we could formalize this proof given the time.

Equivalence of DFAs and NFAs

Ex:

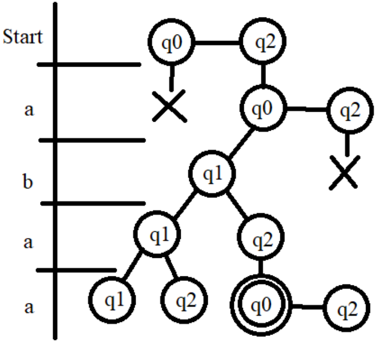


- Step 1: What information will our DFA need to store?
- Step 2: What should the start state be?
- Step 3: What should the accept states be?
- Step 4: Fill in the transitions.

Equivalence of DFAs and NFAs

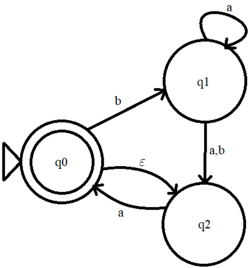
Step 1: What information will our DFA need to store?

- We are trying to simulate an NFA.
- When we were trying to simulate DFAs, we tracked what state the machine was in.



- Now we need to track a set of states...

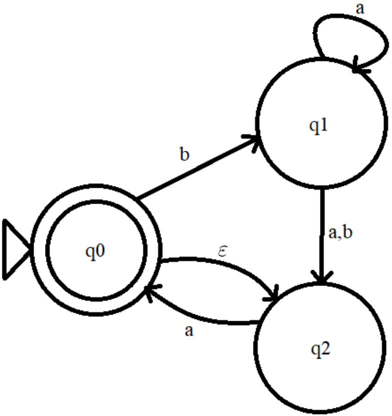
Equivalence of DFAs and NFAs



- Now we need to track a set of states.
- Power set: $\mathcal{P}(Q)$ is the set of all subsets of Q.
- New states:
  $\{\emptyset, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$
- If $|Q| = k$, then $|\mathcal{P}(Q)| = 2^k$.
- That can become a lot of states quickly.

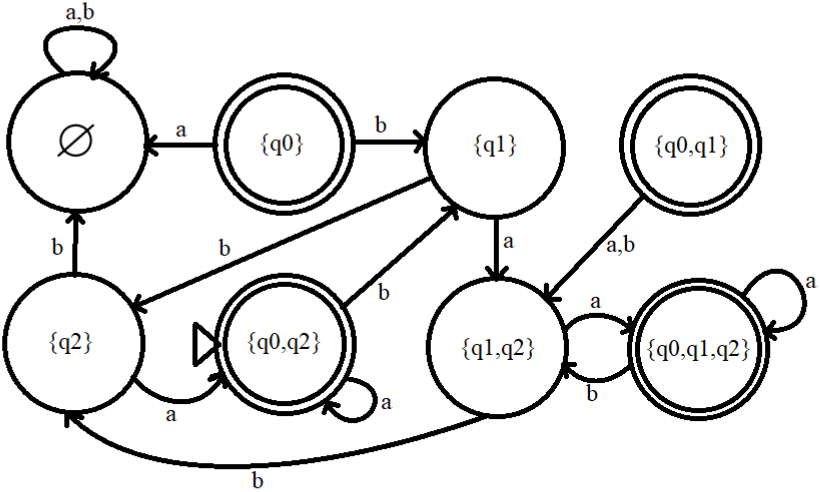
Equivalence of DFAs and NFAs

Ex:



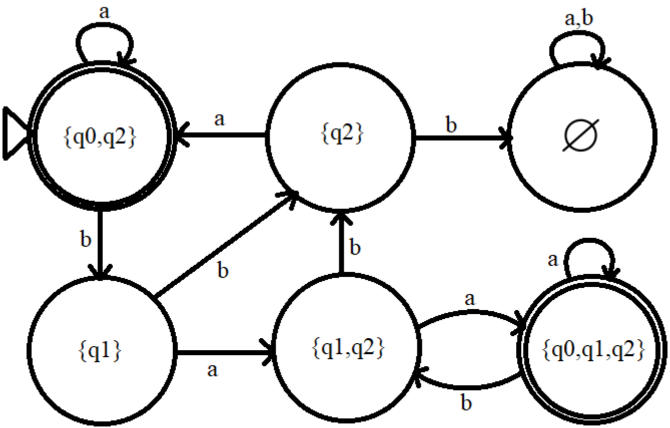
- Step 1: What information will our DFA need to store?
- Step 2: What should the start state be?
- Step 3: What should the accept states be?
- Step 4: Fill in the transitions.

Equivalence of DFAs and NFAs



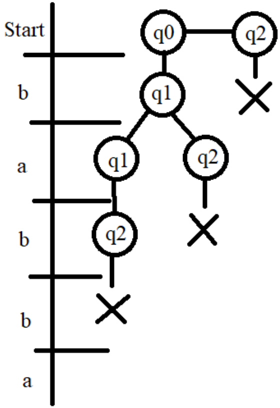
Note that there are two states here that are unreachable from the start state. These are extra states and can be removed from the machine.

Equivalence of DFAs and NFAs



Equivalence of DFAs and NFAs

Note, the new state corresponding to the empty set $\varnothing$ corresponds to cases like 'babba', where ALL versions of the NFA are unable to complete computation, i.e. where ALL of the paths end in an X.



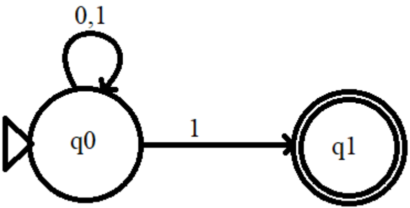
Equivalence of DFAs and NFAs

Theorem

Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

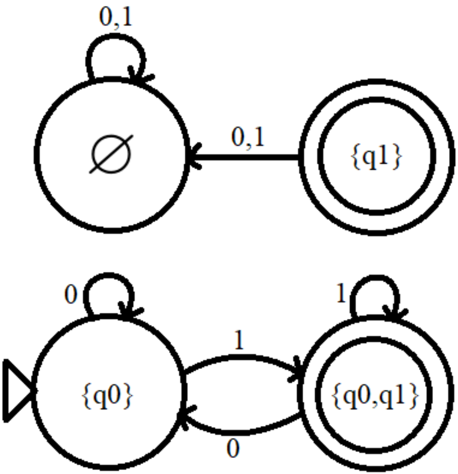
Equivalence of DFAs and NFAs

Ex:



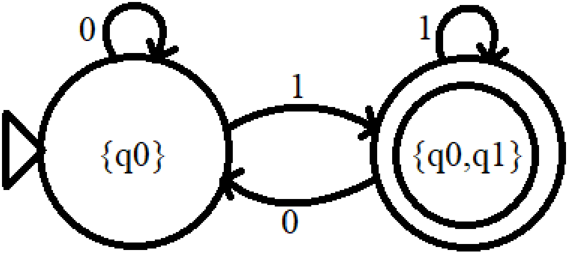
- Step 1: What information will our DFA need to store?
- Step 2: What should the start state be?
- Step 3: What should the accept states be?
- Step 4: Fill in the transitions.

Equivalence of DFAs and NFAs



Again we see unreachable states, and so we remove them from the machine.

Equivalence of DFAs and NFAs



Regular Expressions

We use regular operations to build up regular expressions that describe languages.

Ex:

$$(0|1)0^*$$

- The ‘pipe’ $|$ means ‘or’ and represents the union operation.
- The ‘asterisk’ * represents the star operation.
- We would use a circle $\circ$ to denote concatenation.
  - Much like multiplication, we write concatenation implicitly.
  - i.e., the regular expression above is really saying:

$$(\{0\} \cup \{1\}) \circ \{0\}^*$$

Aside: Applications

Regular Expressions have a number of important applications for CS:

- Pattern recognition in text.
  - *grep* in UNIX
  - *awk* in UNIX
  - Perl
- Database searches.

Regular Expressions

Another Example:

$$(0|1)^*$$

Note that the star operation can be applied to the union operation.

This language is any string that can be created from the alphabet $\{0, 1\}$.

Order of Operations: star, concatenation, union
(unless parentheses change the order)

Examples

- $R_1 = 0^*10^*$
  - $L(R_1) = \{w \mid w$ contains a single $1\}$
- $R_2 = (0|1)^*1(0|1)^*$
  - $L(R_2) = \{w \mid w$ has at least one $1\}$
- $R_3 = 1^+ = 1 \circ 1^*$
  - $L(R_3) = \{w \mid w$ has at least one 1 and is only 1's$\}$
- $R_4 = 1^7 = 1111111$
  - $L(R_4) = \{w \mid w$ is the string of seven 1's$\}$
- $R_5 = (\, 0(0|1)^*0 \mid 1(0|1)^*1 \mid 0 \mid 1 \,)$
  - $L(R_5) = \{w \mid w$ starts and ends with the same symbol$\}$

Formal Definition

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

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 already know how to build NFAs for 4, 5, and 6!

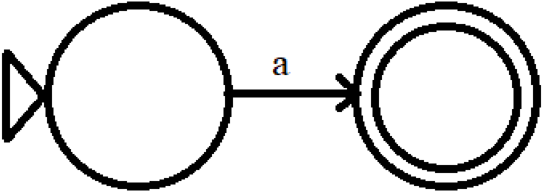
Let's build NFAs for 1, 2, and 3.

Now we combine them. Example:

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

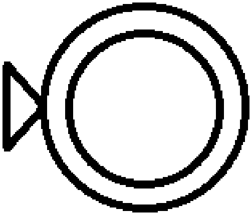
Equivalence of NFAs and Regular Expressions

1. a for some a in the alphabet Σ (i.e., a single symbol)



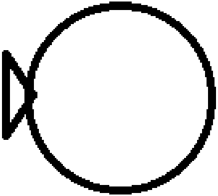
Equivalence of NFAs and Regular Expressions

2. ε (this represents the language with just the empty string)



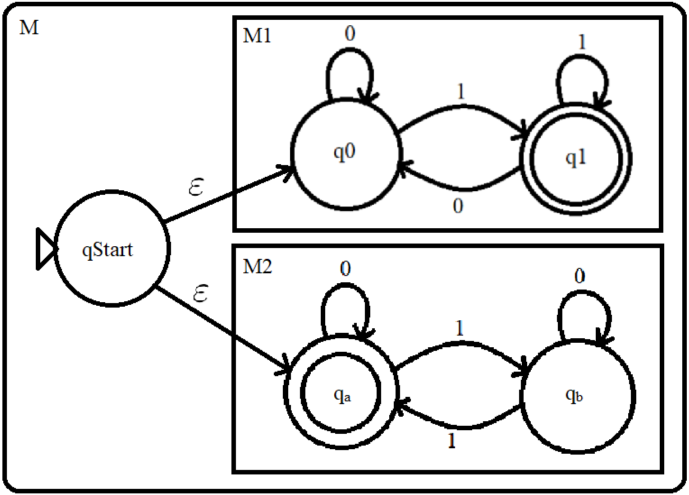
Equivalence of NFAs and Regular Expressions

3. $\varnothing$ (this is the empty language)



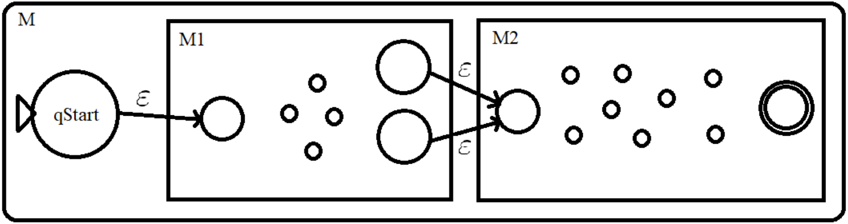
Equivalence of NFAs and Regular Expressions

4. $(R_1 \mid R_2)$, where R_1 and R_2 are regular expressions



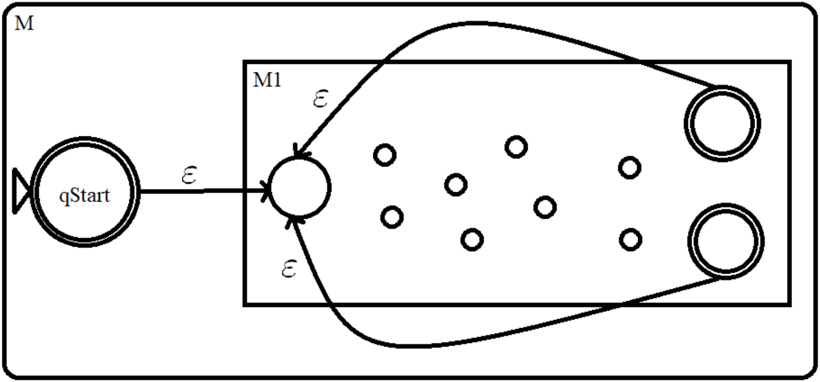
Equivalence of NFAs and Regular Expressions

5. $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions



Equivalence of NFAs and Regular Expressions

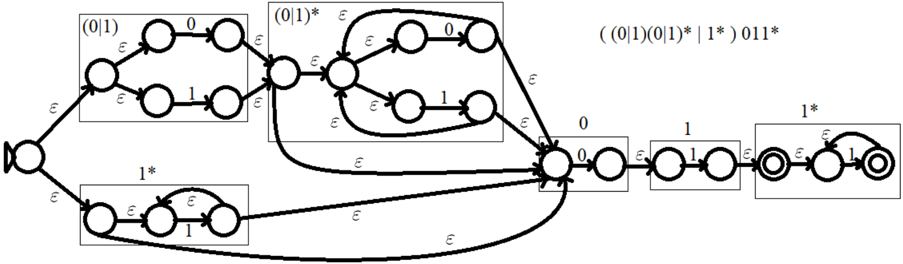
6. (R_1^*), where R_1 is a regular expression



Equivalence of NFAs and Regular Expressions

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

$$\big(\ (0|1)^+ \mid 1^*\ \big)\ 01^+ = (\ (0|1)(0|1)^* \mid 1^*\ )\ 011^*$$



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.