UNIVERSITY OF WATERLOO

Faculty of Mathematics
Waterloo, Ontario N2L 3G1

Centre for Education in
Mathematics and Computing

Grade 11/12 Math Circles

October 27/28, 2015

Formalism and Languages: Regular Languages

Presenter: Dr. Troy Vasiga
David R. Cheriton School of Computer Science

Overview:

- Review of Selected Exercises
- Regular Languages Review
- Complicated Closure Examples
- Finite State Machines
- Exercises (see Problem Set document)

Review of Selected Exercises

These questions are from last week's problem set.

Question 2: Are the following sets closed under the following operations? If so, give some reasoning. If not, give a counter-example.

Part (a): The odd integers under multiplication.

Solution: Yes, the odd integers are closed under multiplication. Consider two odd integers a and b such that $a = 2k + 1$ and $b = 2j + 1$ for some integers k and j. Then, when we multiply a and b we get:

$$ab = (2k + 1)(2j + 1) = 4kj + 2j + 2k + 1 = 2(2kj + j + k) + 1 = 2m + 1$$

where $m = 2kj + j + k$ which is an integer. Thus, the product of a and b is an odd integer. $\square$

Part (b): The negative integers under subtraction.

Solution: No, negative integers under subtraction are not closed. Consider the integers $a = -4$ and $b = -10$. Then

$$a - b = -4 - -10 = -4 + 10 = 6$$

which is not a negative integer. □

Part (c): The real numbers under division.

Solution: No, the real numbers under division are not closed. Consider $a = 1$ and $b = 0$ and then a/b is not a real number. Notice that $\mathbb{R} - \{0\}$ would be closed under division. □

Question 5: Write the regular expressions for all words over $\Sigma = \{a, b\}$

(a) with no more than 3 a's;

Solution: $b^*(a|\epsilon)b^*(a|\epsilon)b^*(a|\epsilon)b^*$ □

(b) with the number of a's being a multiple of 3;

Solution: $(b^*ab^*ab^*a)^*b^*$ □

(c) with only one occurrence of the subword aaa.

Solution: Let us assume that we mean "exactly one occurrence of aaa." Then, we have

$$(b^*(\epsilon|a|aa)b)^*aaa(b^*(\epsilon|a|aa)b)^*$$

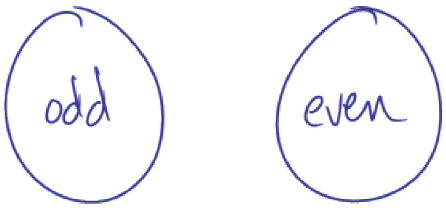
□

Outline of a Finite State Machine (FSM)

- A *finite state machine* (aka *finite automata*) is a machine that recognizes languages
- It is a way to model the world:
  - what I know about the world
  - what changes in my world with input

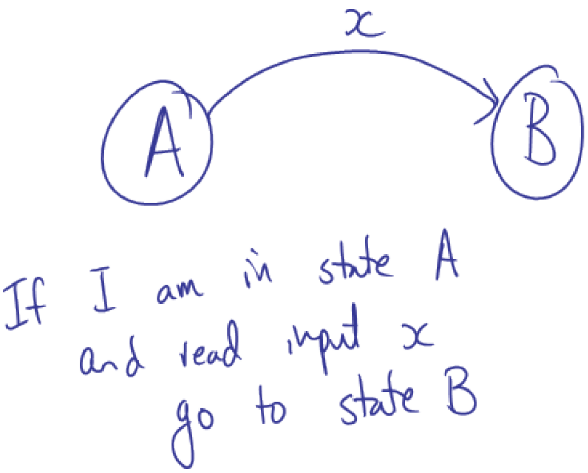
What is a State?

A description of what we know or what is true.



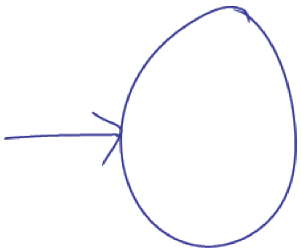
What is Input?

A transition from one state to another.



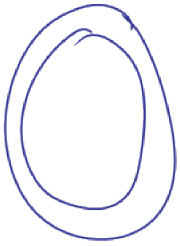
Start State

How do we know where to begin?



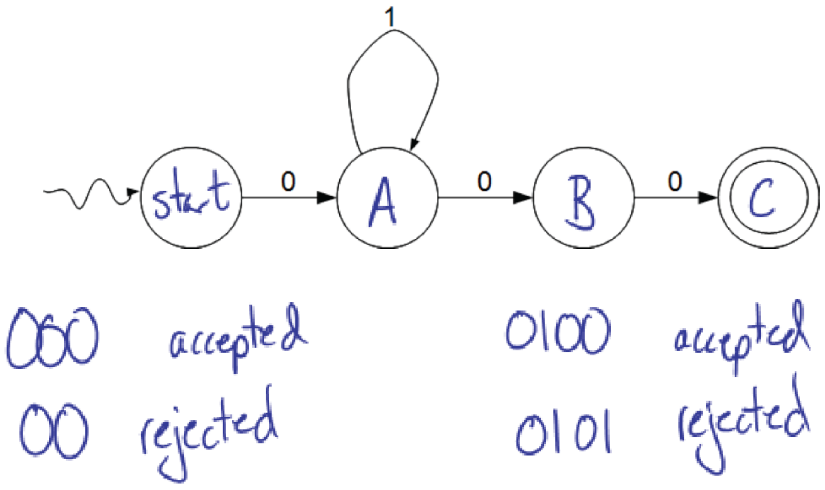
Final/accepting states

How do we know that the input was good/valid/accepted?



Binary number example

Consider $\Sigma = \{0, 1\}$. What words does the following FSM accept?



Is there a nice way to write this?

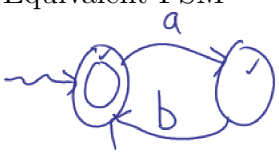
Back to Regular Expressions (RE)

Last time we defined regular expressions and described some operations on them. For instance, here is a regular expression. Can you determine what words it specifies?

$$01^*00$$

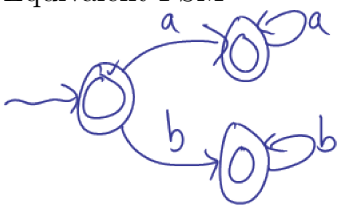
$(ab)^*$ Words specified by the RE: ϵ, ab, $abab$, ...

Equivalent FSM



$(a^*|b^*)$ Words specified by the RE: ϵ, a, b, aa, bb, ...

Equivalent FSM



Example

Suppose we want a FSM that accepts all words over $\Sigma = \{a, b, c\}$ that have the word abc contained in them.

Some words that are accepted:

abc, $aabc$, $abcabcabc$, $aabbbbababccccc$

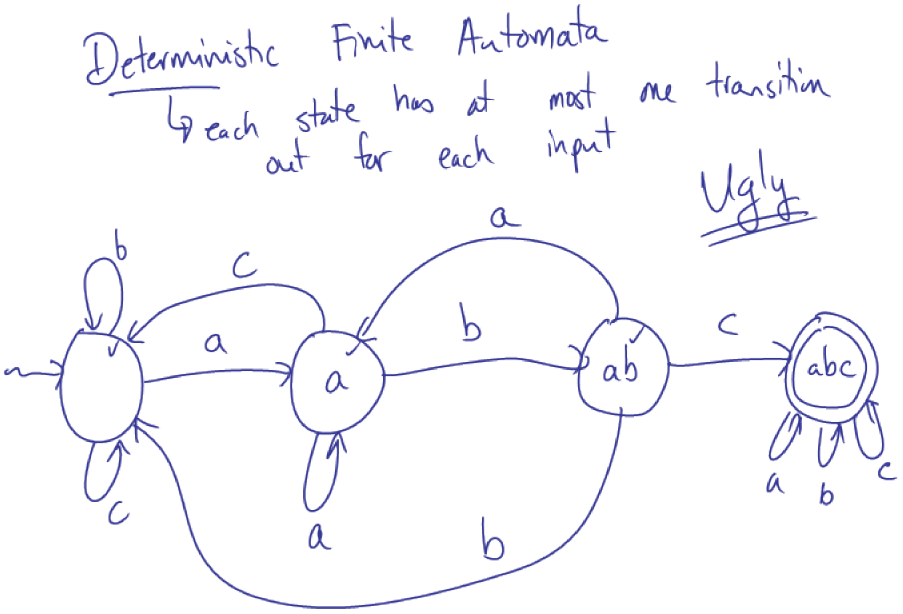
Some words that are rejected:

ϵ, $aabbcc$

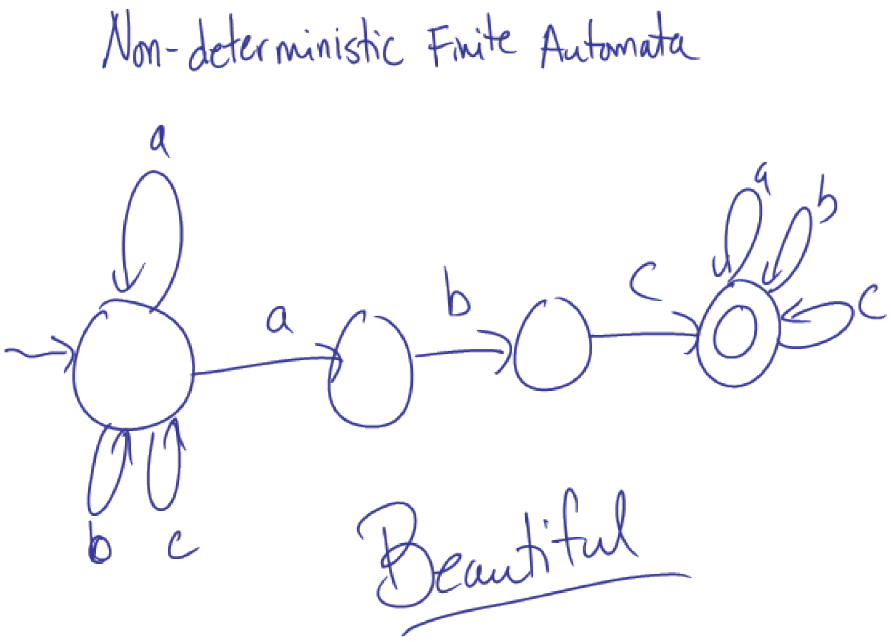
Here is another regular expression. Determine the words that it specifies.

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

DFA


Deterministic Finite Automata
each state has at most one transition out for each input
Ugly
a
b
c
a
b
ab
c
abc
a
b
c
a
b
c
a
b

NFA


Non-deterministic Finite Automata
a
a
b
c
a
b
c
b
c
Beautiful

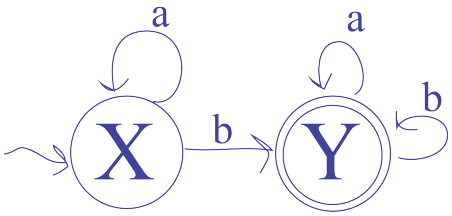
More Formalism

Formally, a deterministic finite automata is 5-tuple $(Q, \Sigma, \delta, s, A)$ where:

- Q is the set of states
- Σ is the input alphabet
- $\delta : Q \times \Sigma \to Q$ is the *transition function*
- $s \in Q$ is the start state
- $A \subseteq Q$ is the set of final/accepting states

Here is an example:

- $Q = \{X, Y\}$
- $\Sigma = \{a, b\}$
- All possible values for inputs in δ according to the following diagram:



  - $\delta(X, a) = X$
  - $\delta(X, b) = Y$
  - $\delta(Y, a) = Y$
  - $\delta(Y, b) = Y$
- $s = \{X\}$
- $A = \{Y\}$

What is the difference between a DFA and an NFA?

Instead of δ, we have $\Delta : Q \times \Sigma \to 2^Q$.

What is 2^Q?

The *powerset* of Q, which is the set of all subsets of Q.

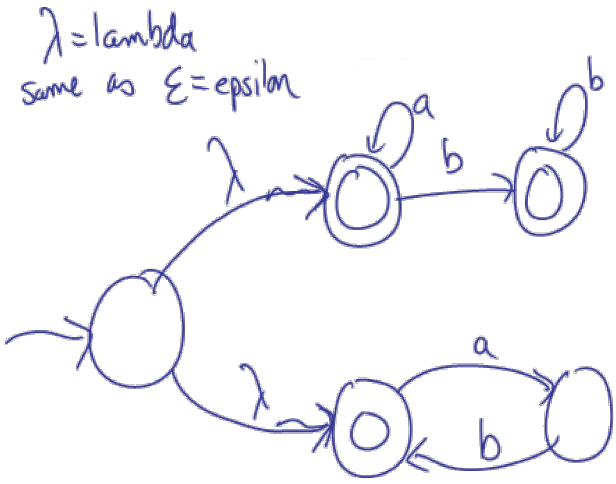
For example, if $Q = \{X, Y\}$ then

$$2^Q = \{\{\}, \{X\}, \{Y\}, \{X, Y\}\}$$

λ-NFA

Suppose you wanted to create a FSM that accepts all words of the form

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



Closure properties of Regular Languages

Recall that a set is *closed under an operation* if applying the operation on any element in the set causes the result to still be in the set.

Example: The set of integers is closed under addition, subtraction and multiplication, but *not* closed under division.

Regular languages are closed under the following operations:

- union
- concatenation
- Kleene star (repetition)
- complementation (If L is regular then $\overline{L} = \Sigma^* - L$ is regular)
- intersection ($L_1 \cap L_2$)

The first three are easily shown. How about the last two?

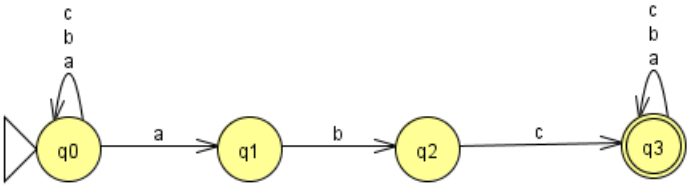
Equivalency of DFA, NFA, λ-NFA, RE

NFA $\rightarrow$ DFA: Subset construction

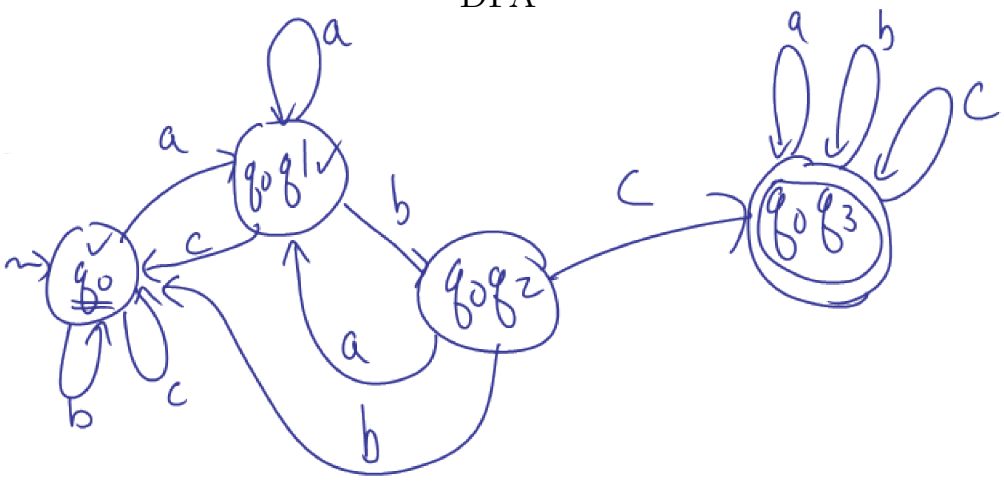
- start in the start state
- for each possible input, create *a state* from the set of "current" states
- finished when all transitions out of all states in NFA are covered

Subset Construction Example

NFA



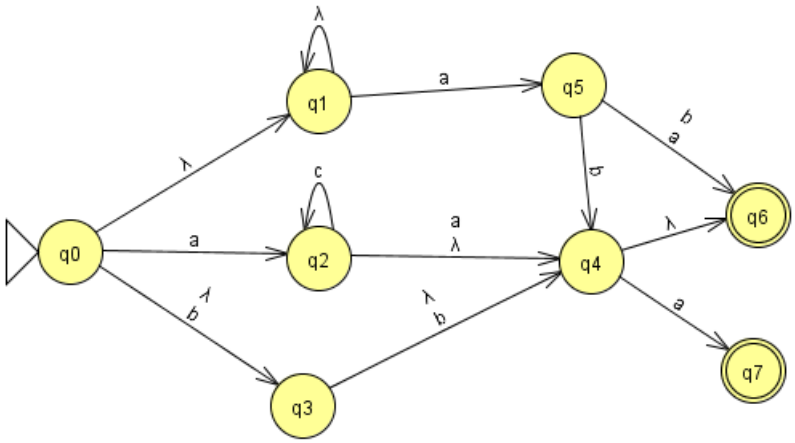
DFA

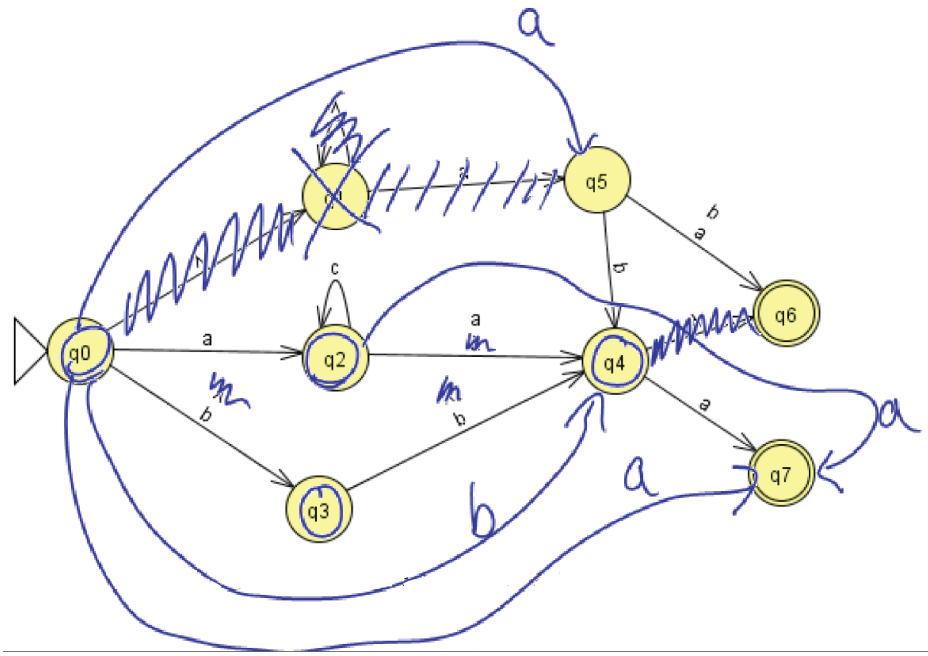


Equivalency of DFA, NFA, λ-NFA, RE

λ-NFA $\rightarrow$ NFA: Shortcut removal

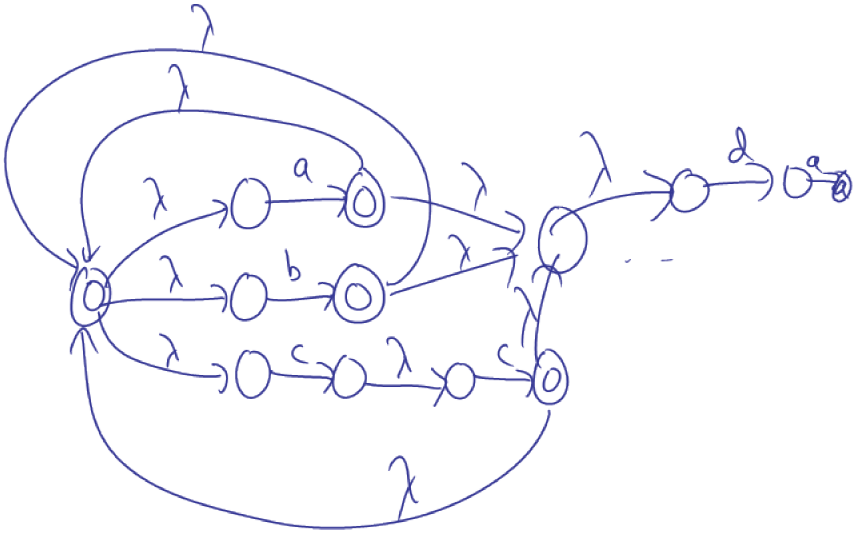
- take shortcuts of the form λX (to just X)
- pull back final states
- delete all λ transition
- remove dead states





RE $\rightarrow$ λ-NFA: Use RE definition

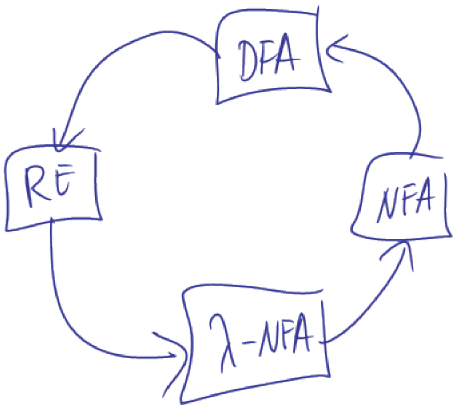
$$(a|b|cc)^*(da|bd)^*$$



DFA $\rightarrow$ RE

Left as an exercise.

The Circle of Life



Back to Closure

Let's show that regular languages are closed under complement.

That is, suppose that L is regular. We would like to show that $\overline{L} = \Sigma^* - L$ is also regular.

We know that if L is regular, we can write a DFA for L.

Let $M = (Q, \Sigma, \delta, s, A)$ be such a DFA and ensure that we have the *error state* explicitly included in Q.

Then $\overline{M} = (Q, \Sigma, \delta, s, Q - A)$ accepts $\overline{L}$.

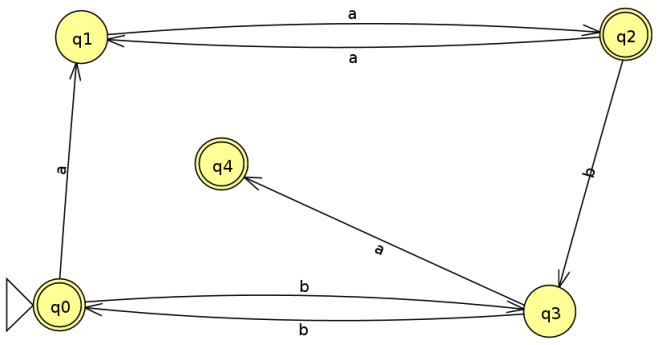
Minimizing a DFA

A neat idea from J. Brzozowski (Waterloo Professor Emeritus).

1. Reverse the edges of the DFA.
2. Convert this NFA to an DFA (using the construction outlined above)
3. Reverse the edges of this DFA.
4. Convert this NFA to a DFA.
5. This DFA is equivalent to the original DFA but is minimal (in the number of states).

Minimizing Example

Consider the following DFA:



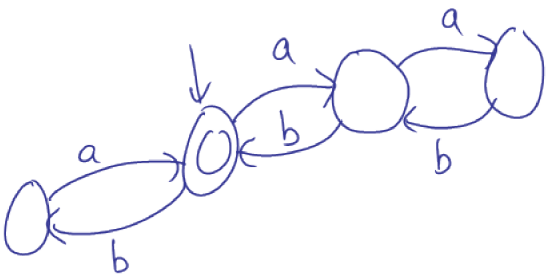
A language which is not regular

Consider the set of words over $\Sigma = \{a, b\}$ which have an equal number of as and bs.

Example words:

- ϵ
- ab
- $abab$
- $aabb$
- ba
- $bbabaa$

A DFA for this language:



A very useful lemma: the pumping lemma

Theorem

Let L be an infinite regular language. Then there are strings x, y and z such that $y \neq \lambda$ and $xy^nz \in L$ for each $n \geq 0$.

Try this on a simple language $L = \{a^k b^k\}$ (which is a subset of the language we saw earlier), and prove that this language is not regular.

Let

- $x = \lambda$
- $y = aa$
- $z = bb$
- $n = 2$