Regular Languages: Properties, Extensions, Limits and Next Steps

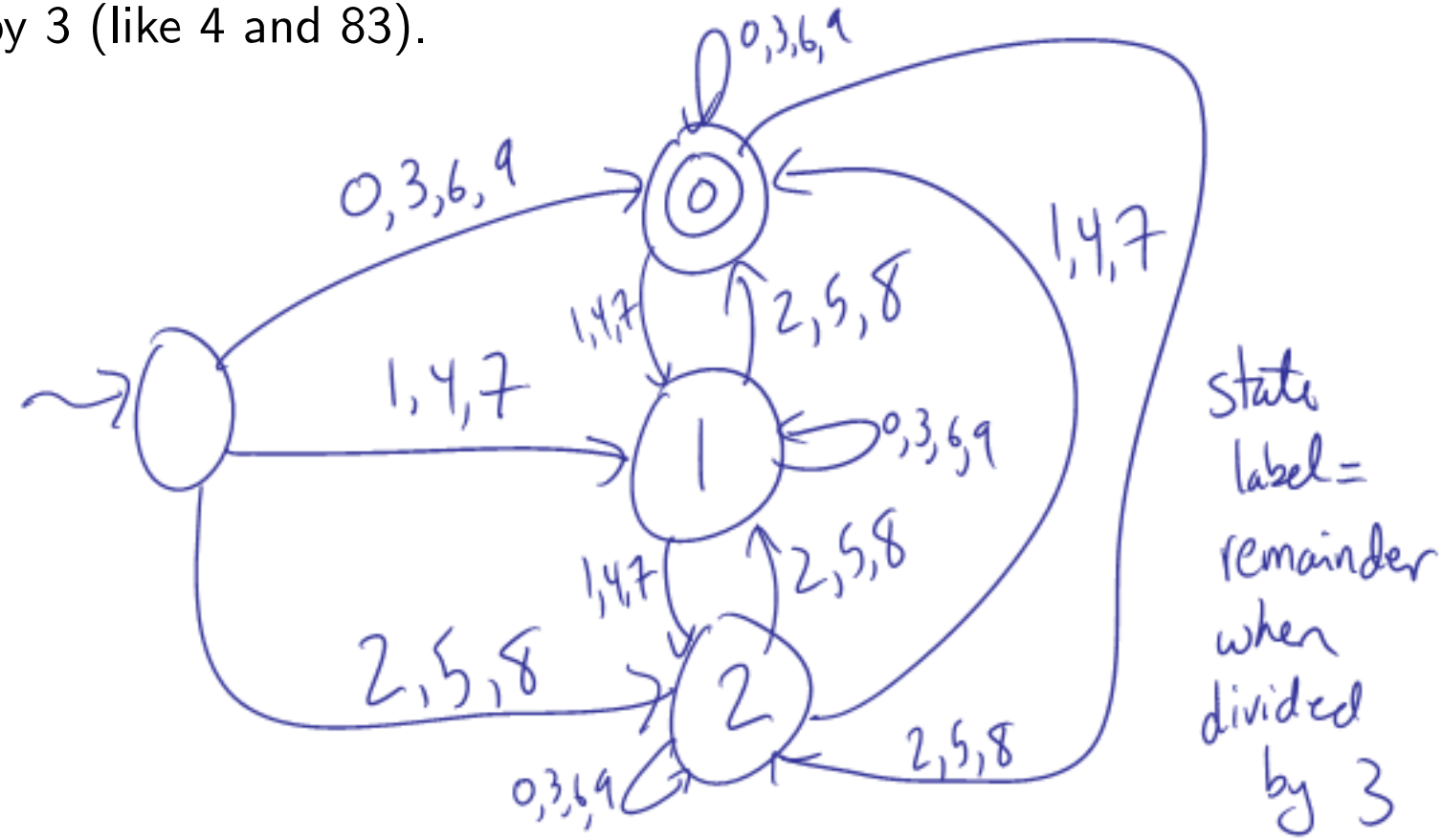
Dr. Troy Vasiga
David R. Cheriton School of Computer Science
University of Waterloo

Outline

- Recap of some previous exercises
- Equivalency of DFA, NFA, λ-NFA, RE
- Minimizing a DFA
- Closure properties of Regular Languages
- A language which is not regular
- A very useful lemma: the pumping lemma
- Classifying the previous non-regular language
- Context-free grammars: examples
- Exercises

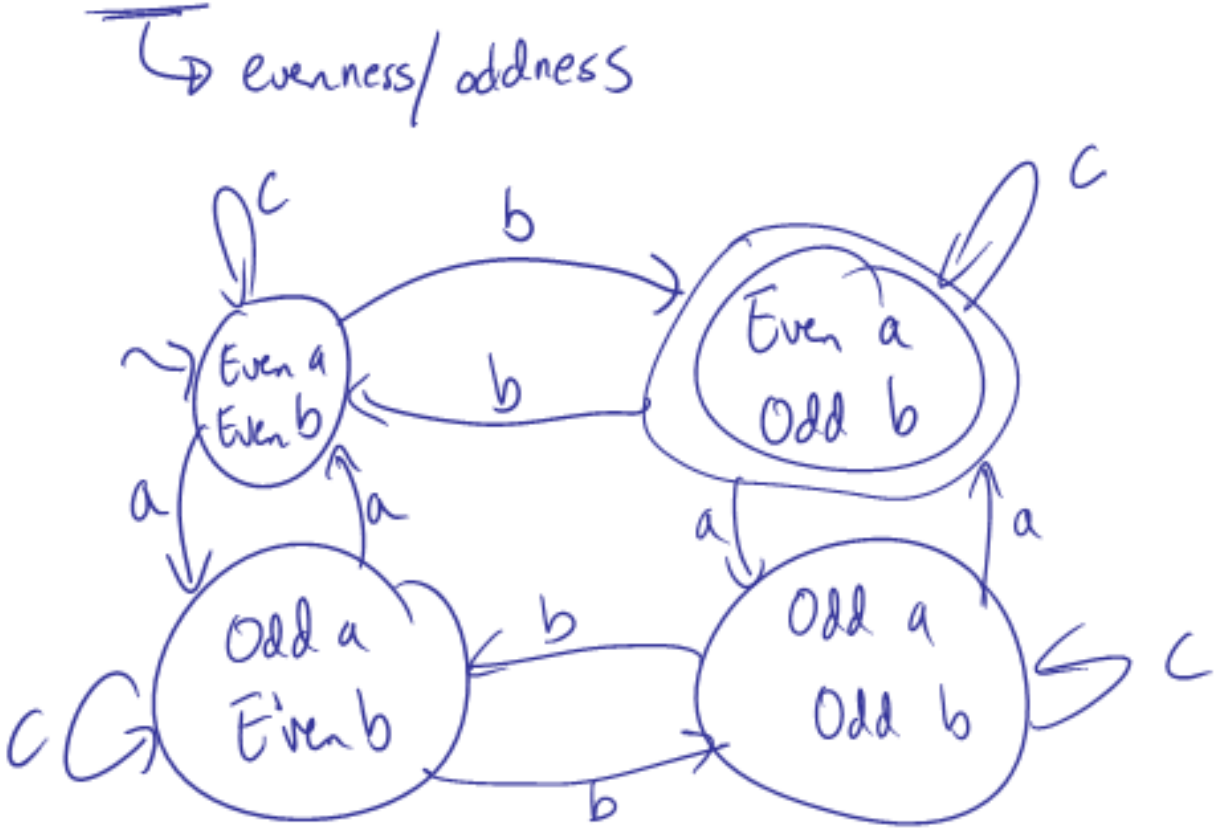
Recap of some previous exercises

Question 7: Speaking of threes, construct a FSA which accepts decimal numbers which are divisible by 3. Note that 0, 3, 21, 33960210 are divisible by 3. (**HINT:** You should need exactly 4 states to do this, and think about what it means to be divisible by 3). Try it on some numbers which are divisible by 3 (like 3 and 27) and other numbers which are not divisible by 3 (like 4 and 83).



Recap of some previous exercises

Question 12: Construct a FSA over the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ which accepts words which contain an even number of a's and an odd number of b's. There are no restrictions on the number of c's. You should verify that aabaacc, b and abacababc are accepted and that aaaacacac and bb are rejected. Hint: You need 4 states, and you should keep track of the parity of both a's and b's.

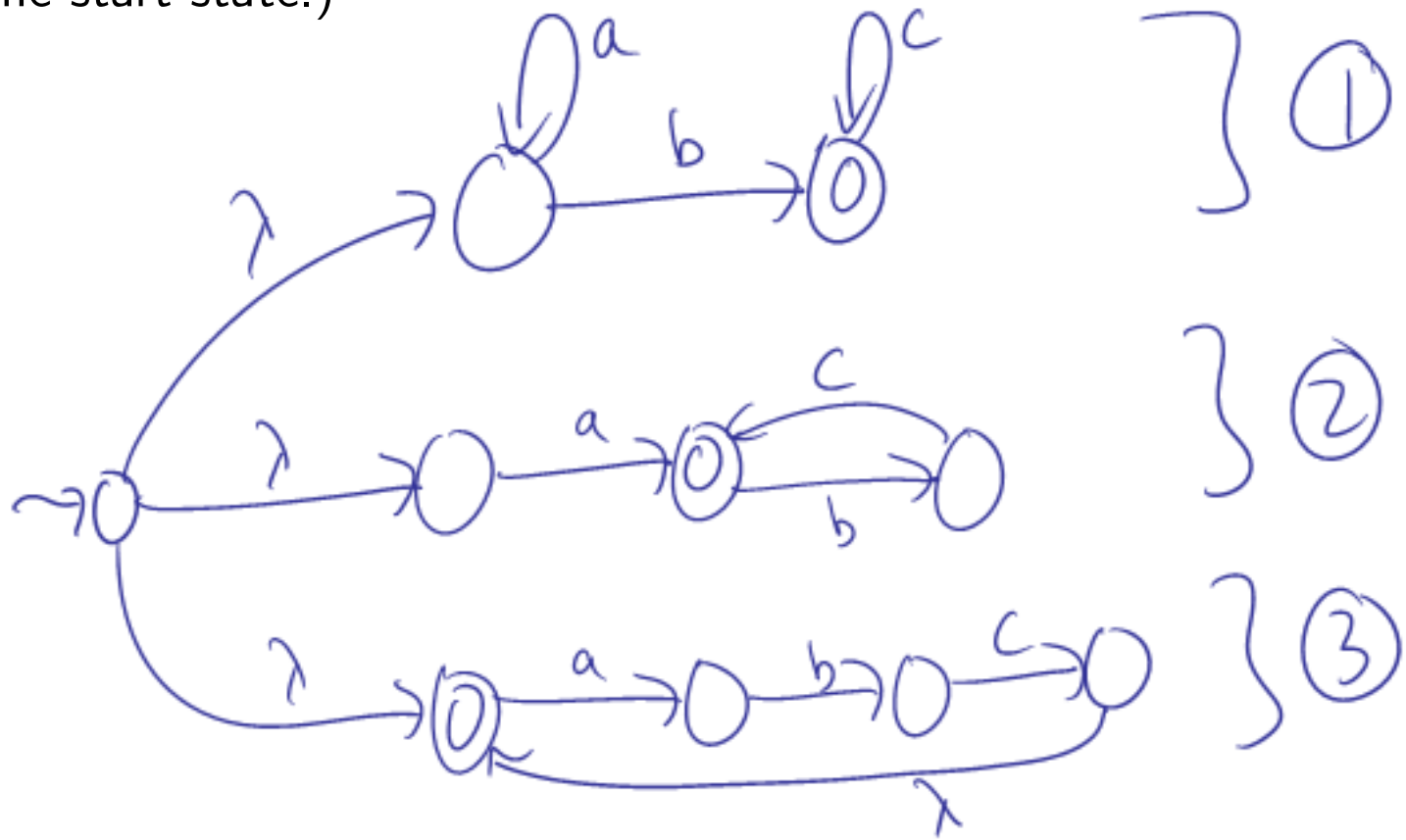


Recap of some previous exercises

Question 13: Construct the FSA for the following regular expression:

$$\underbrace{a^*bc^*}_{①}|\underbrace{a(bc)^*}_{②}|\underbrace{(abc)^*}_{③}$$

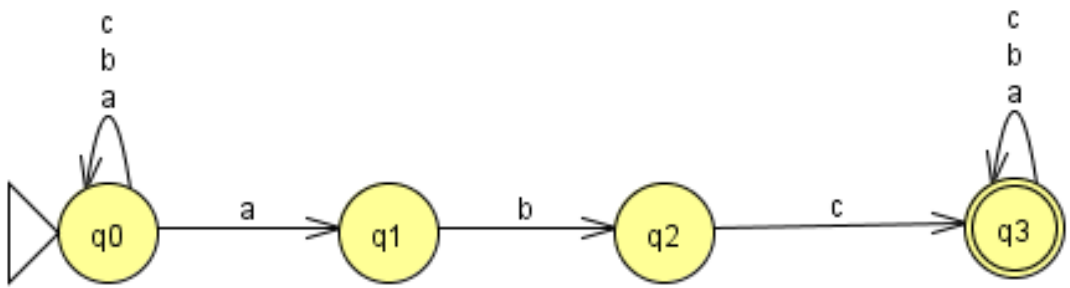
(Hint: you should think about creating three distinct FSMs, for each part of the regular expression. Then, connect them using a λ-transition from the start state.)



Equivalency of DFA, NFA, λ-NFA, RE

NFA $\rightarrow$ DFA: Subset construction

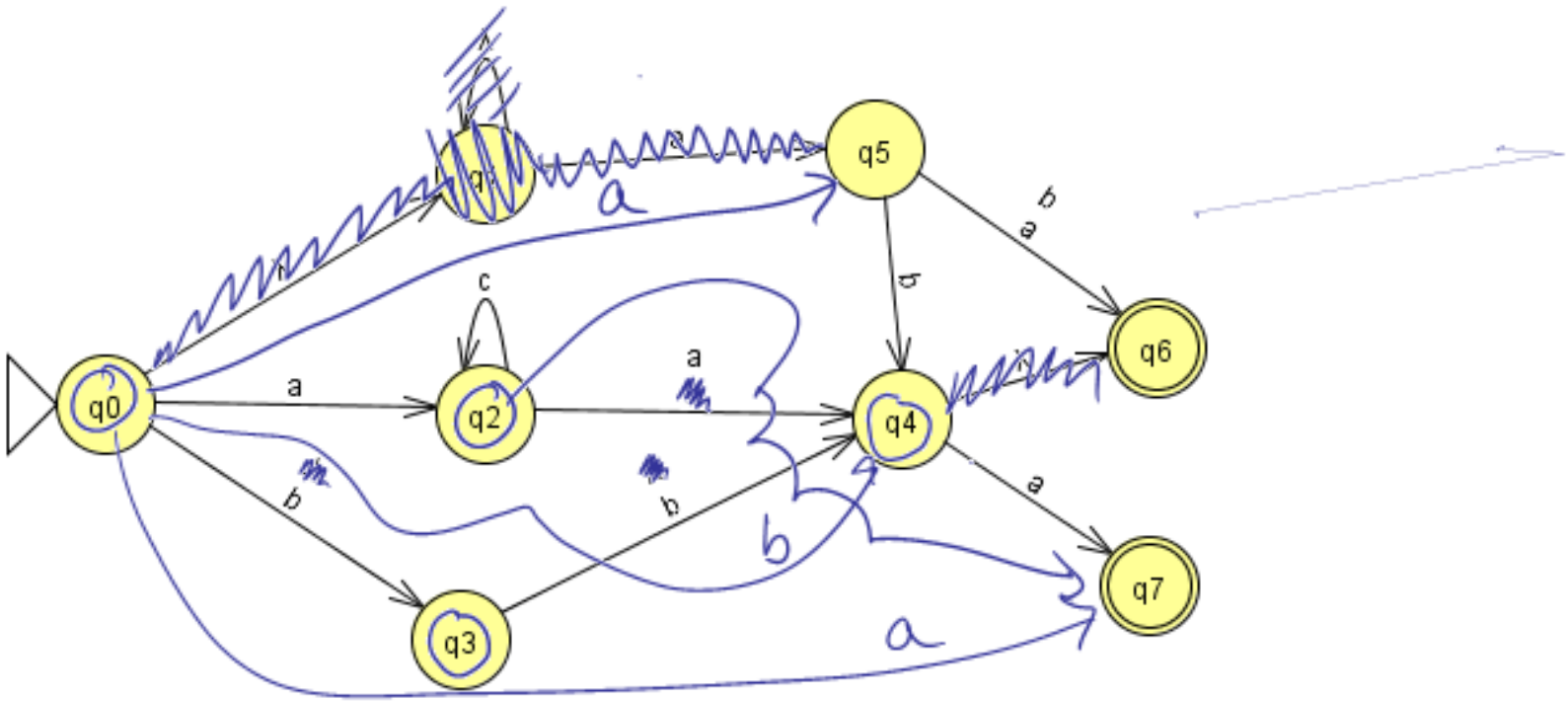
- start in the start state
- for each possible input, create a state from the set of "current" states
- finished when all transitions out of all states in NFA are covered



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



Equivalency of DFA, NFA, λ-NFA, RE

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

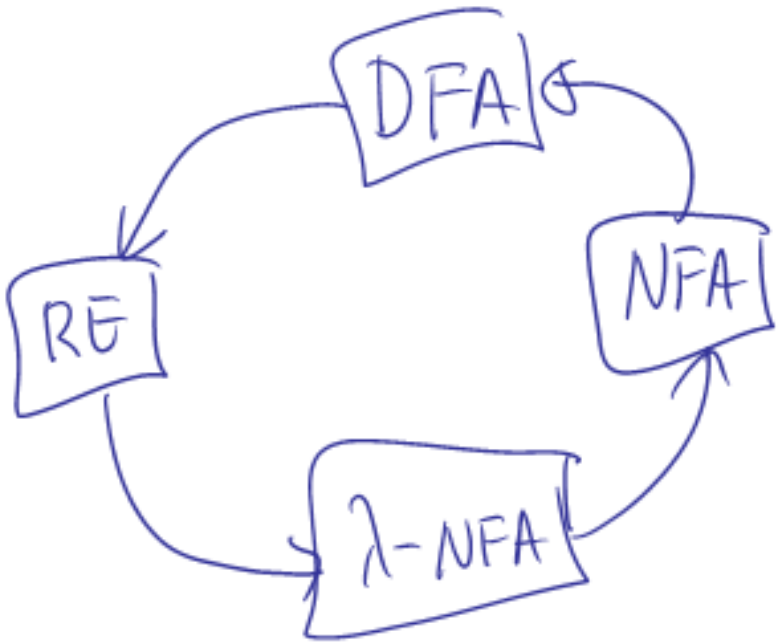
$$(a|b|cc)^*(da|bd)^*$$

See Question 13 (i.e. break into parts)

Equivalency of DFA, NFA, λ-NFA, RE

DFA $\rightarrow$ RE

Left as an exercise.

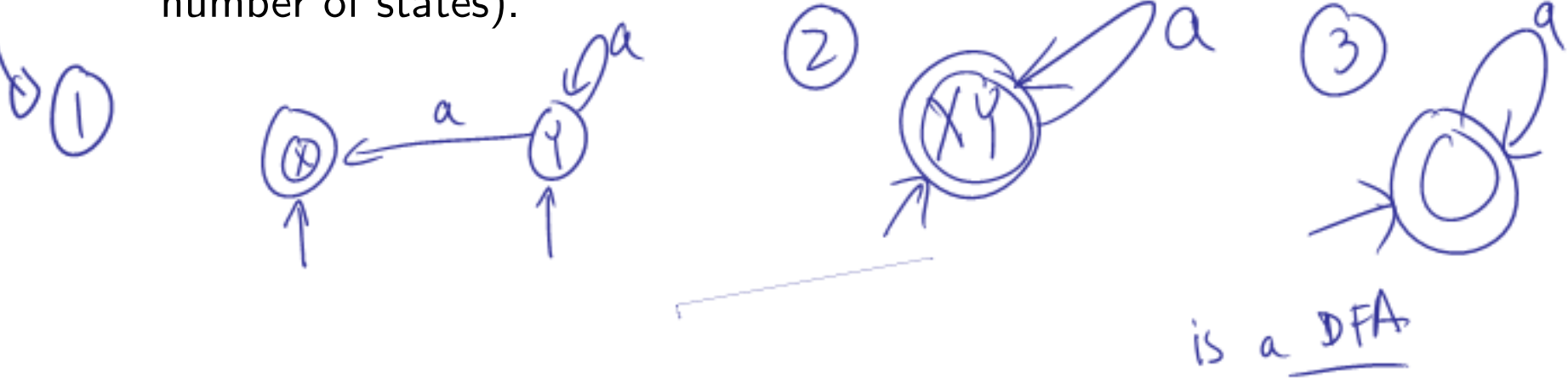


"Circle of life"

Minimizing a DFA

A neat idea from J. Brzozowski (retired Waterloo professor).

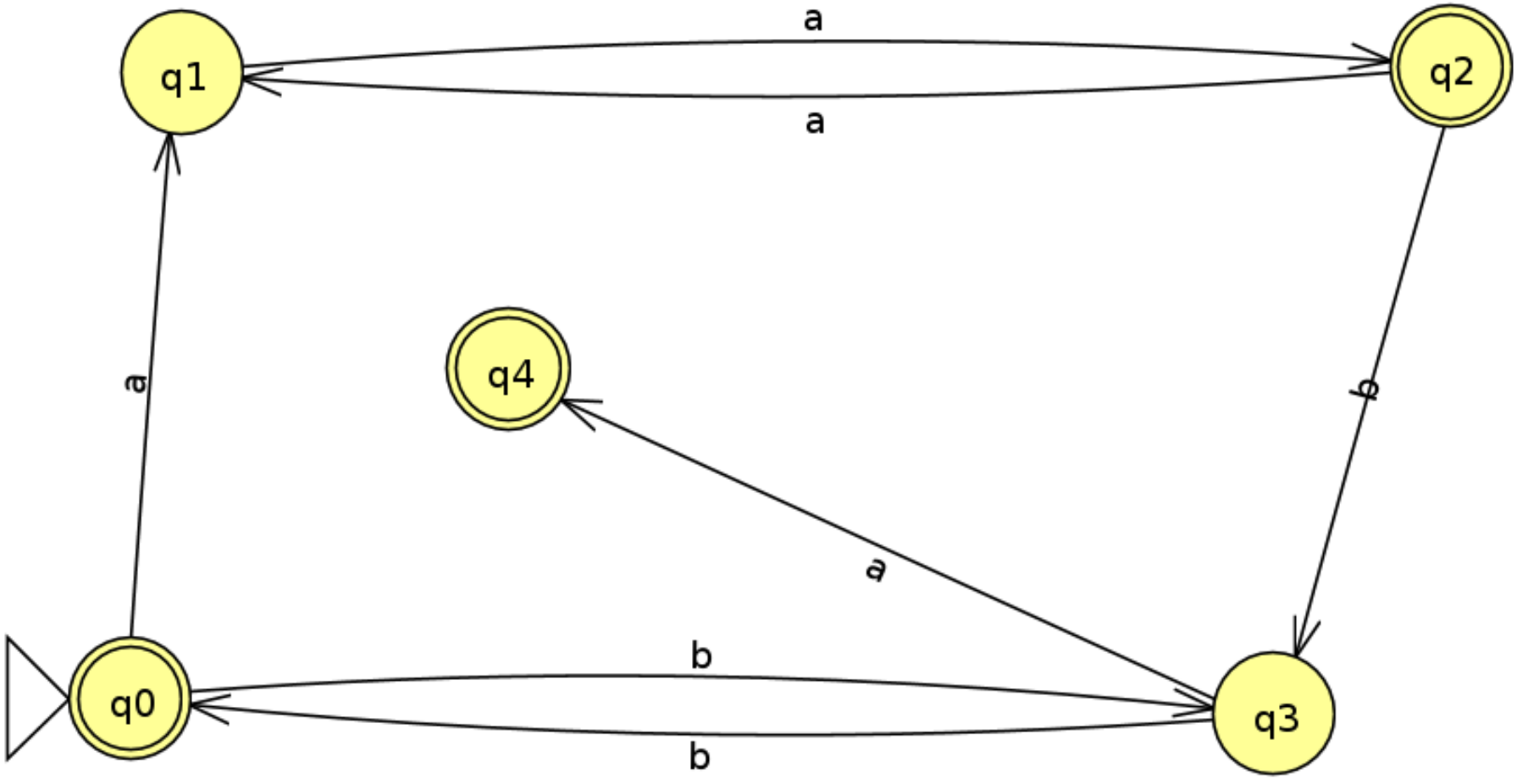
1. Reverse the edges of the DFA. (start states $\Leftrightarrow$ final states)
2. Convert this NFA to an DFA (using the construction outlined above)
3. Reverse the edges of this DFA.
4. Convert this NFA to a DFA.
5. This DFA is equivalent to the original DFA but is minimal (in the number of states).



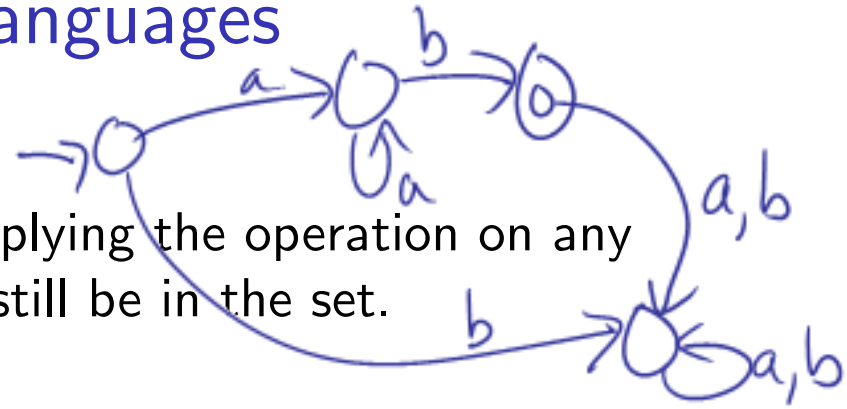
Minimizing example

Question 15: Minimize the following DFA:

Exercise



Closure properties of Regular Languages



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.

Example: The set of integers is *not* closed under division.

Regular languages are closed under the following operations:

- union
- concatenation
- Kleene star
- complementation
- intersection

$\Sigma = \{a, b\}$

$L_1 \cup L_2$ $\quad R_1 \mid R_2$

$L_1 L_2$ $\quad R_1 R_2$

L_1^* $\quad (R_1)^*$

R_1 is reg exp for L_1

R_2 is a reg exp for L_2

$L_1 \cap L_2$ exercise

DFA "complete" $\Rightarrow$ swap accepting non-accepting

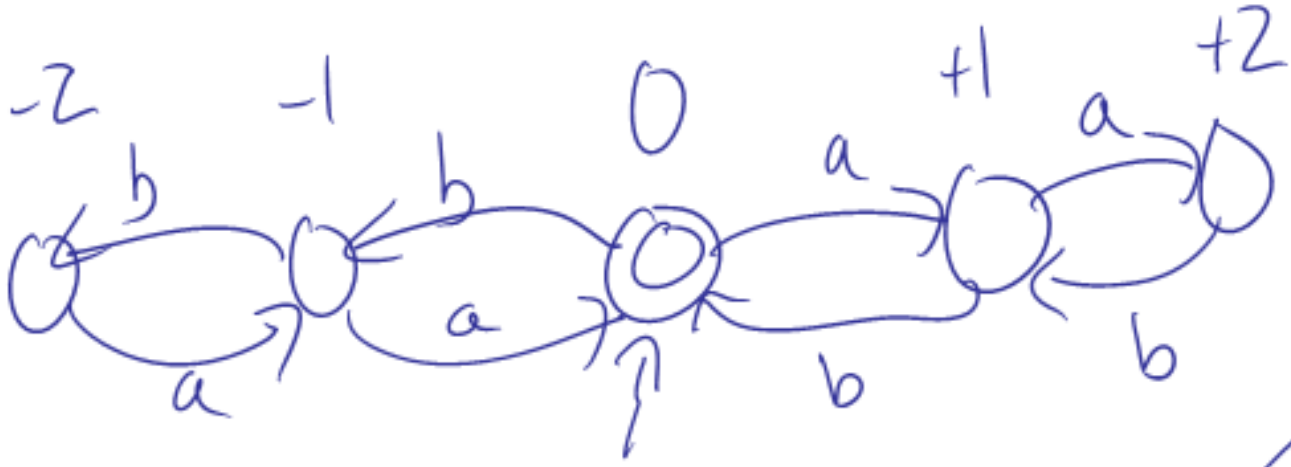
A language which is not regular

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

λ, ab, $aabb$, $abab$, $abbbbaaa$

Example words:

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 $xy^n z \in L$ for each $n \geq 0$.

Let's try this on a simple language $L = \{a^k b^k\}$ (which is a subset of the language we saw earlier).

next day.

Context-free grammars: Examples

What about the set of words $L = \{a^k b^k\}$?

Classifying the previous non-regular language

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

Context-free grammars: Examples

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

Context-free grammars: Usage

We can use the context-free grammar to create the words.

Context-free grammars: Final notes

Why context-free?

Does it ever reject a word?

Exercises