CS5371
Theory of Computation

Lecture 6: Automata Theory IV
(Regular Expression = NFA = DFA)

Objectives

- Give formal definition of Regular Expression
- Show that the power of Regular Expression = the power of NFA = the power of DFA
  - in terms of describing a language

Regular Expression
(Formal Definition)

- We say R is a regular expression if R is
  - a for some a in the alphabet Σ, or
  - ε, or
  - $\emptyset$, or
  - $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions, or
  - $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, or
  - (R_1^*), where R_1 is a regular expression

Don't confuse ε with $\emptyset$

True or False?

$R \cup \emptyset = R$	True
$R \circ \varepsilon = R$	True
$R \cup \varepsilon = R$	False
$R \circ \emptyset = R$	False

Equivalence with NFA
(Part I)

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

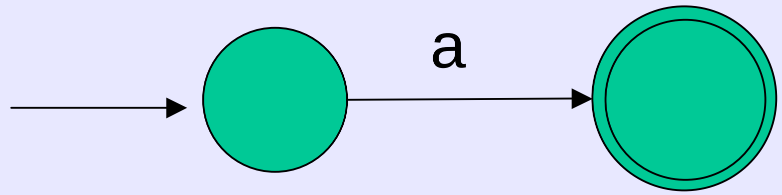
Proof: Let R be the regular expression and L be the language described by R.

Note: L is sometimes written as L(R)

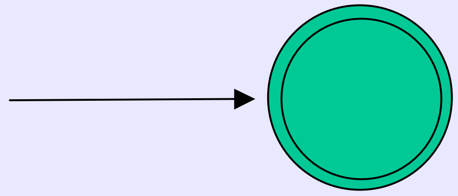
We show how to convert R into an NFA recognizing L(R).

Six Cases to Consider

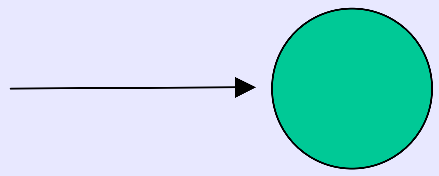
(1) $R = a$ for some a in the alphabet Σ.
Then $L(R) = \{a\}$, and the following NFA recognizes $L(R)$



(2) $R = \varepsilon$. Then $L(R) = \{\varepsilon\}$, and the following NFA recognizes $L(R)$



(3) $R = \emptyset$. Then $L(R) = \{\ \}$, and the following NFA recognizes $L(R)$



For the last three cases:

(4) $R = R_1 \cup R_2$

(5) $R = R_1 \circ R_2$

(6) $R = R_1^*$

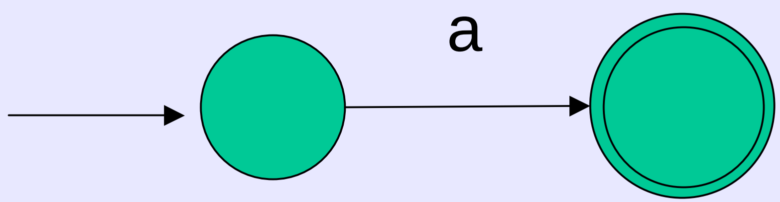
we use the constructions given in the proofs that the class of regular language is closed under the regular operations.

- In other words, we construct NFA for R from NFA for R_1 and NFA for R_2

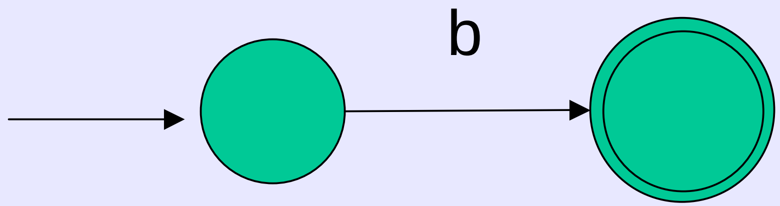
Converting R to NFA (Example)

$R = (ab \cup a)^*$

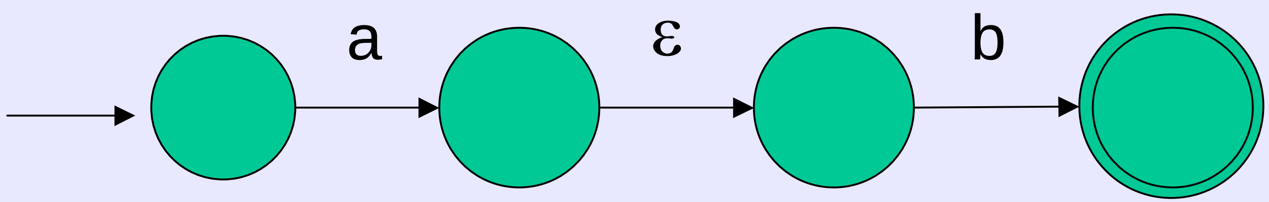
a



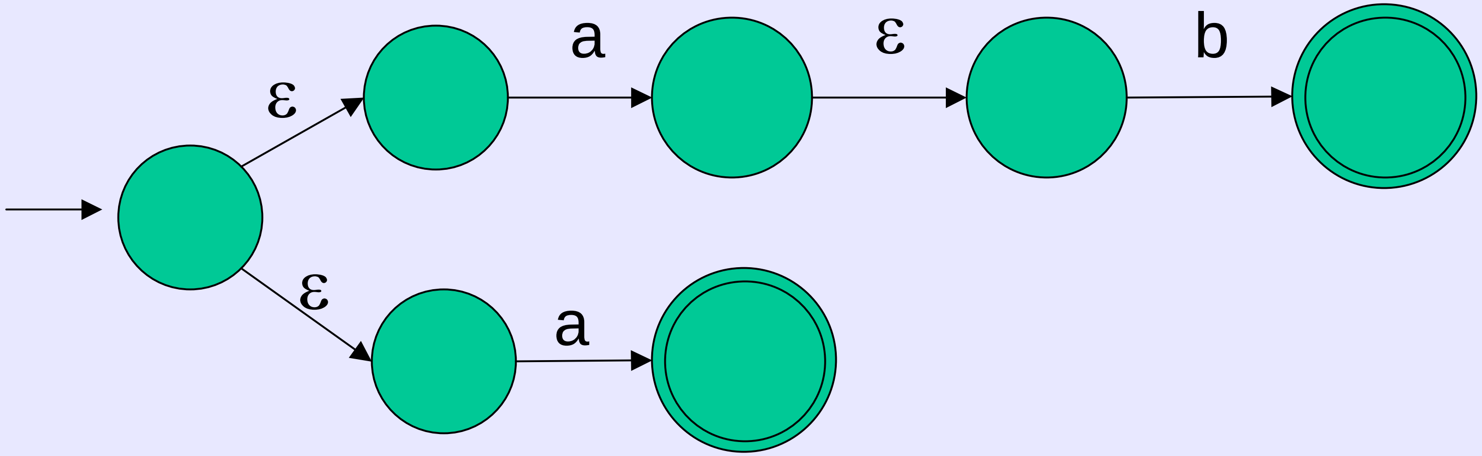
b



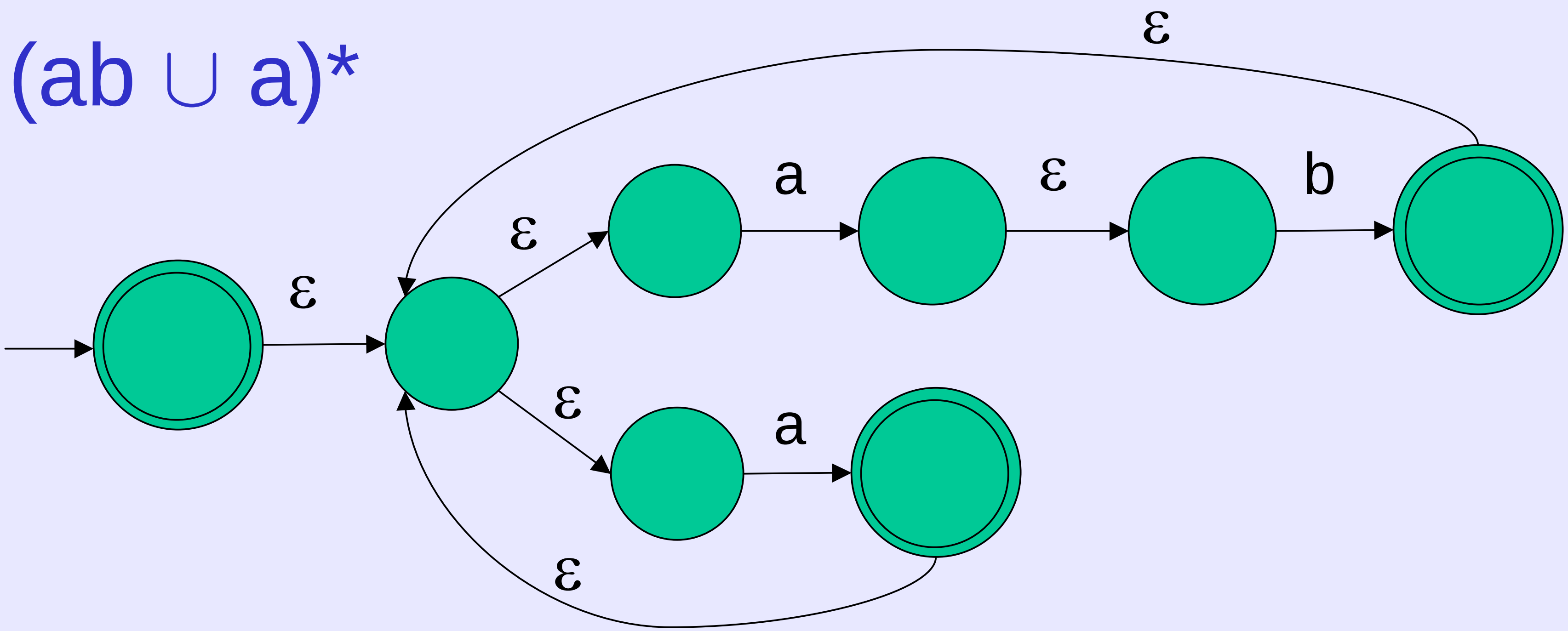
ab



ab ∪ a



(ab ∪ a)*



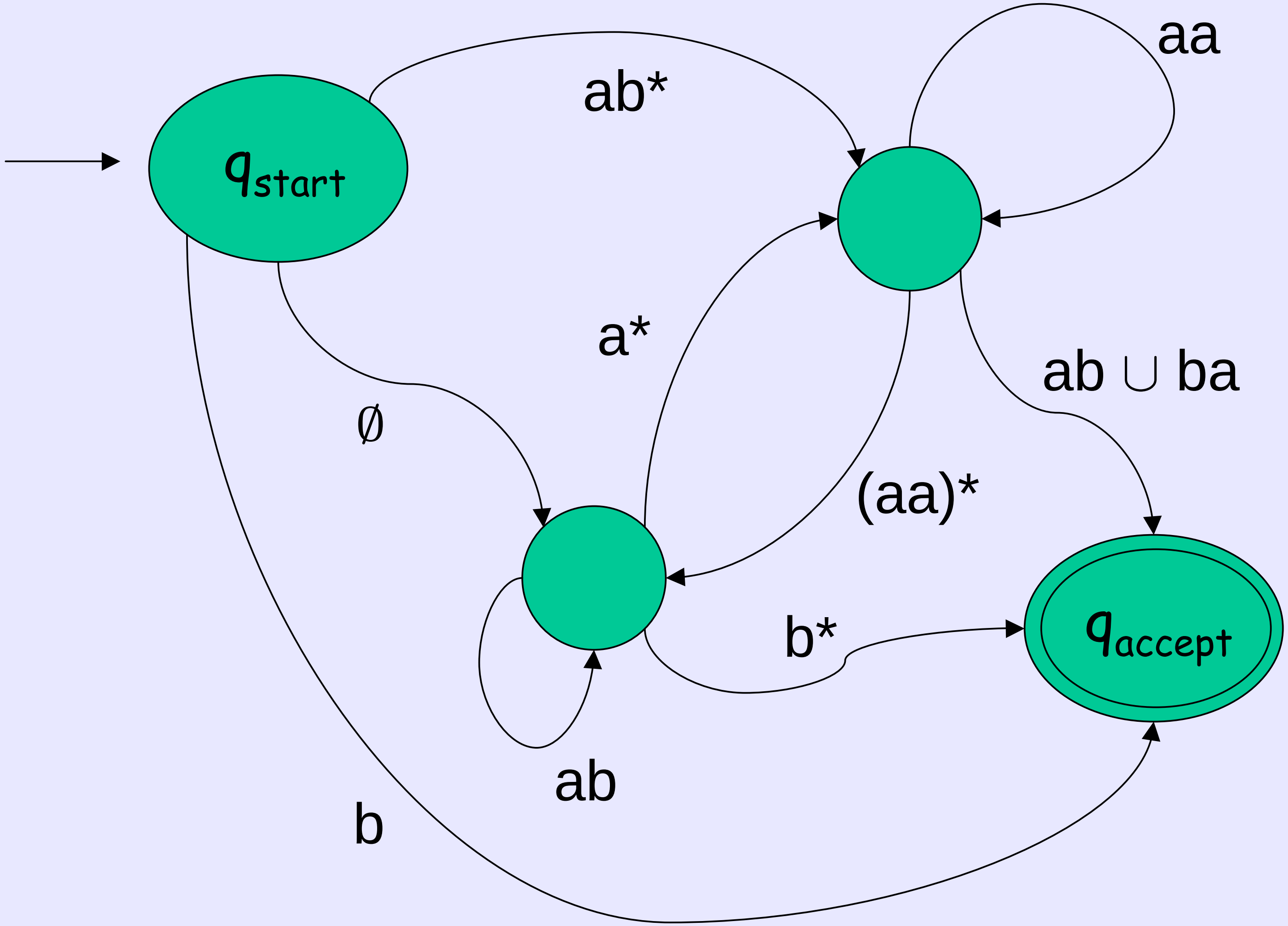
Equivalence with NFA
(Part II)

Lemma: If a language is regular, it can be described by a regular expression.

Proof: Let L be the regular language. We will convert the DFA for L into a regular expression. First, we introduce a new type of automaton: the generalized non-deterministic finite automaton (GNFA)

Later, we show DFA → GNFA → Reg Ex

GNFA (Example)



GNFA

- Similar to NFA, except that the labels on the transition arrows are regular expressions (instead of a character or ε)
- To move along a transition arrow, we read blocks of characters such that it matches the description of the regular expression on that arrow
- An input string is accepted if there is a way to read the input string such that the GNFA is in an accepting state after processing the whole input string

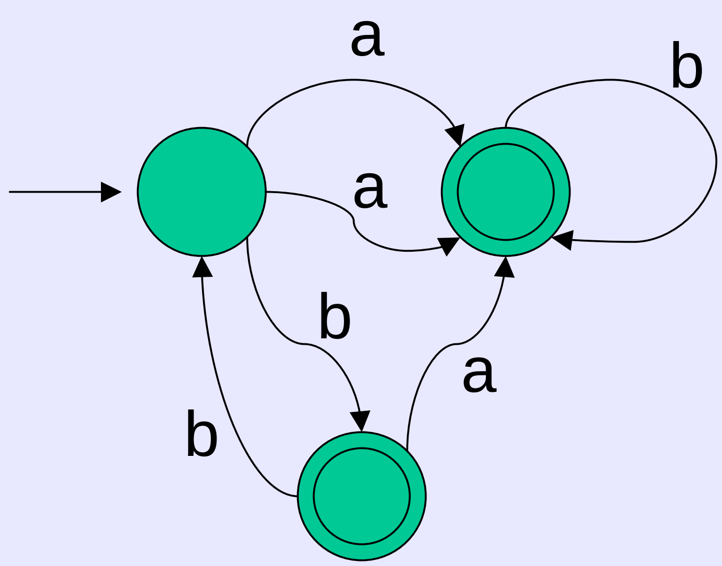
GNFA (further assumptions)

- Only one start state q_{start}, with no incoming arrows
- Only one accepting state q_{accept}, with no outgoing arrows
- Each state (except q_{start} and q_{accept}) has exactly one arrow going to every other state and also itself

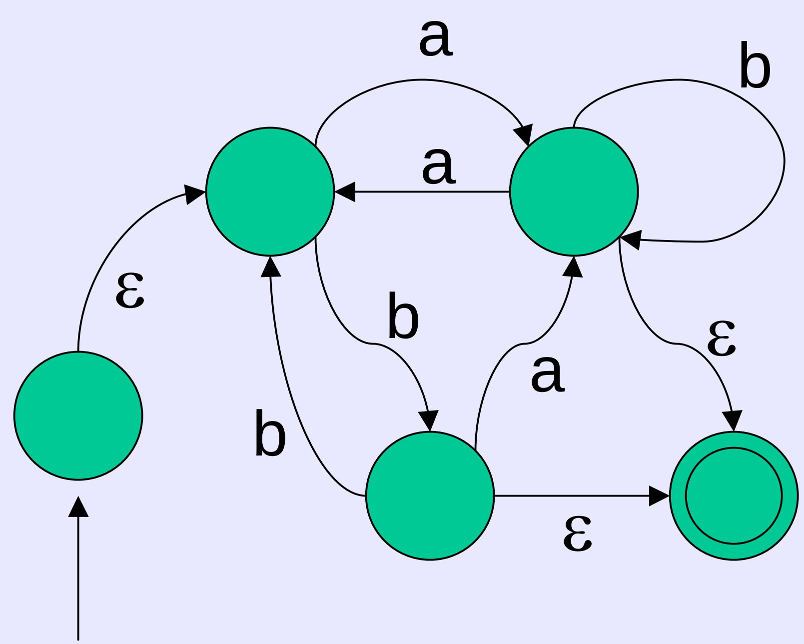
Back to the Proof: Converting DFA to GNFA

- Add a new start state, with ε arrow to the original start state
- Add a new accept state, with ε arrow from each of the original accept state
- If original arrow has multiple labels, we replace this with a new arrow whose label is a regular expression formed by the union of the labels
- If originally no arrow between two states, we add a new arrow whose label is $\emptyset$

Converting DFA to GNFA
(Example)



DFA



GNFA

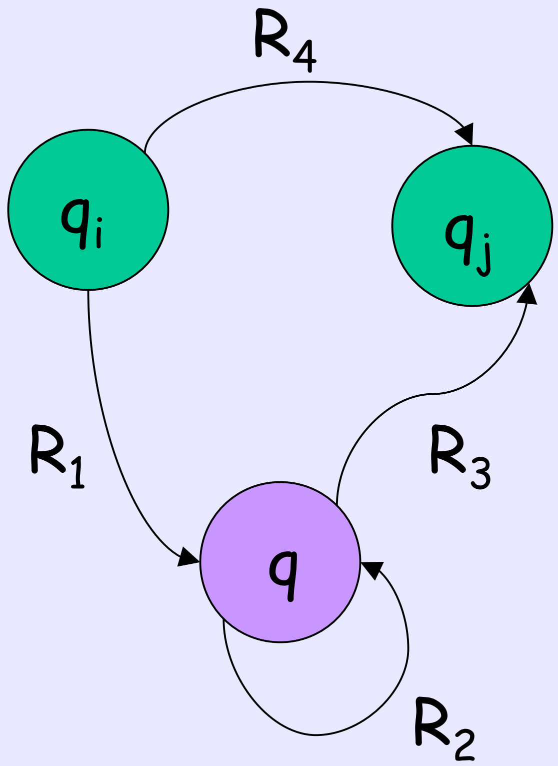
Converting GNFA to Regular Expression

- We iteratively remove one state in GNFA, such that after each state removal, the new GNFA obtained will recognize the same language as the previous one
- When the number of states of GNFA is 2, we have the regular expression (why??)

How to remove a state?

- Select any state q except q_{start} and q_{accept}
- Remove q
  - To compensate the absence of q, the new label on the arrow from q_i to q_j becomes a regular expression that describes all strings that would take the GNFA to go from q_i to q_j, either directly or via q

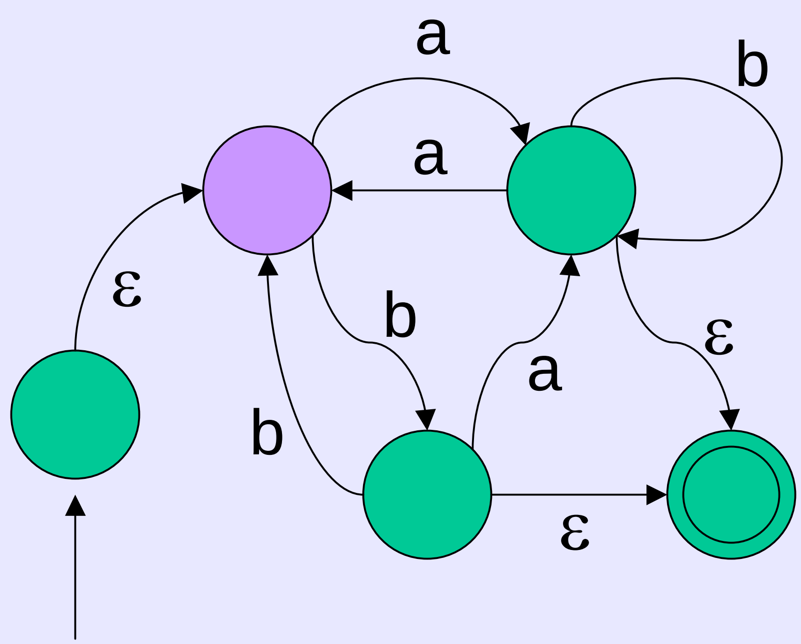
How to remove a state?



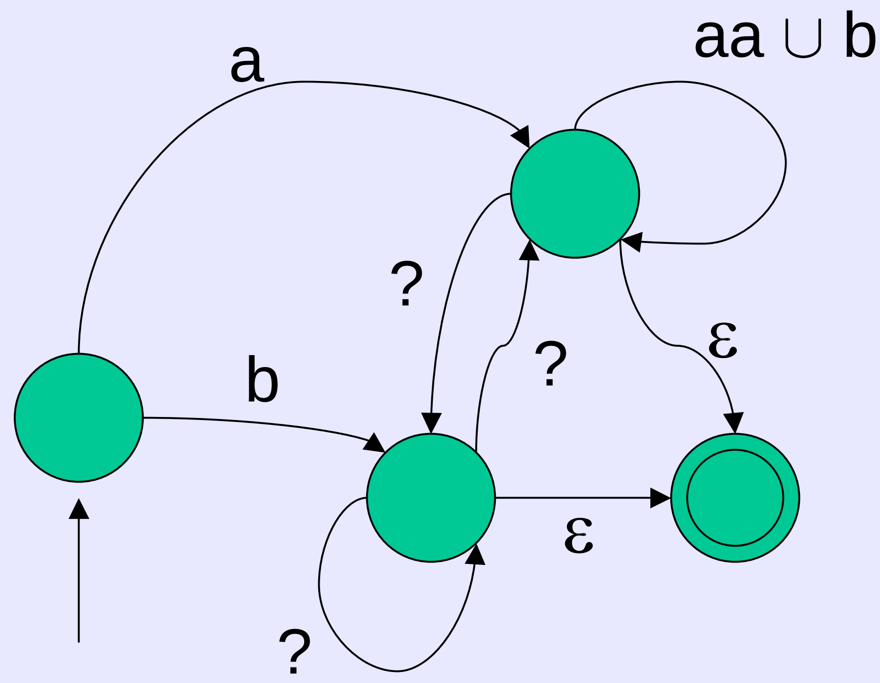
Before Removal

After Removal

Previous Example

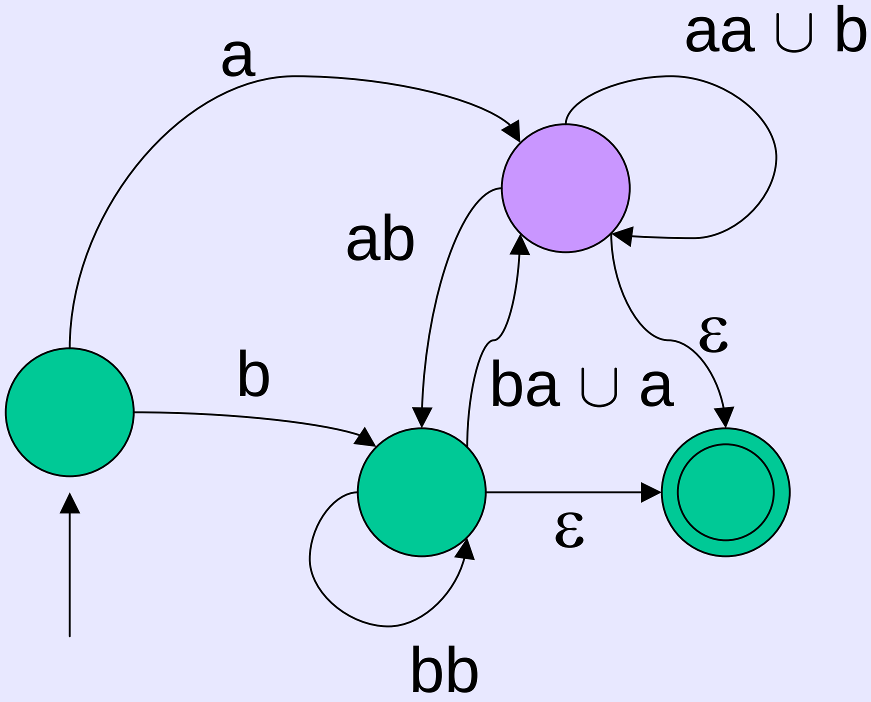


Before Removal

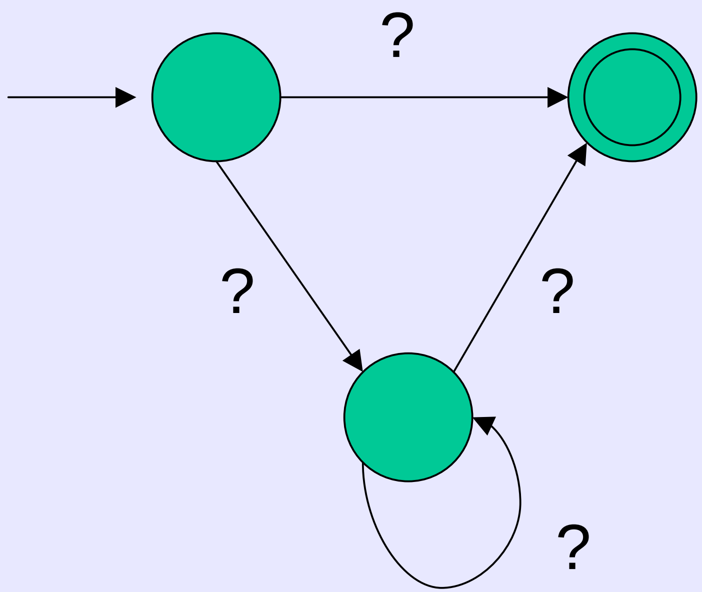


After Removal

Previous Example



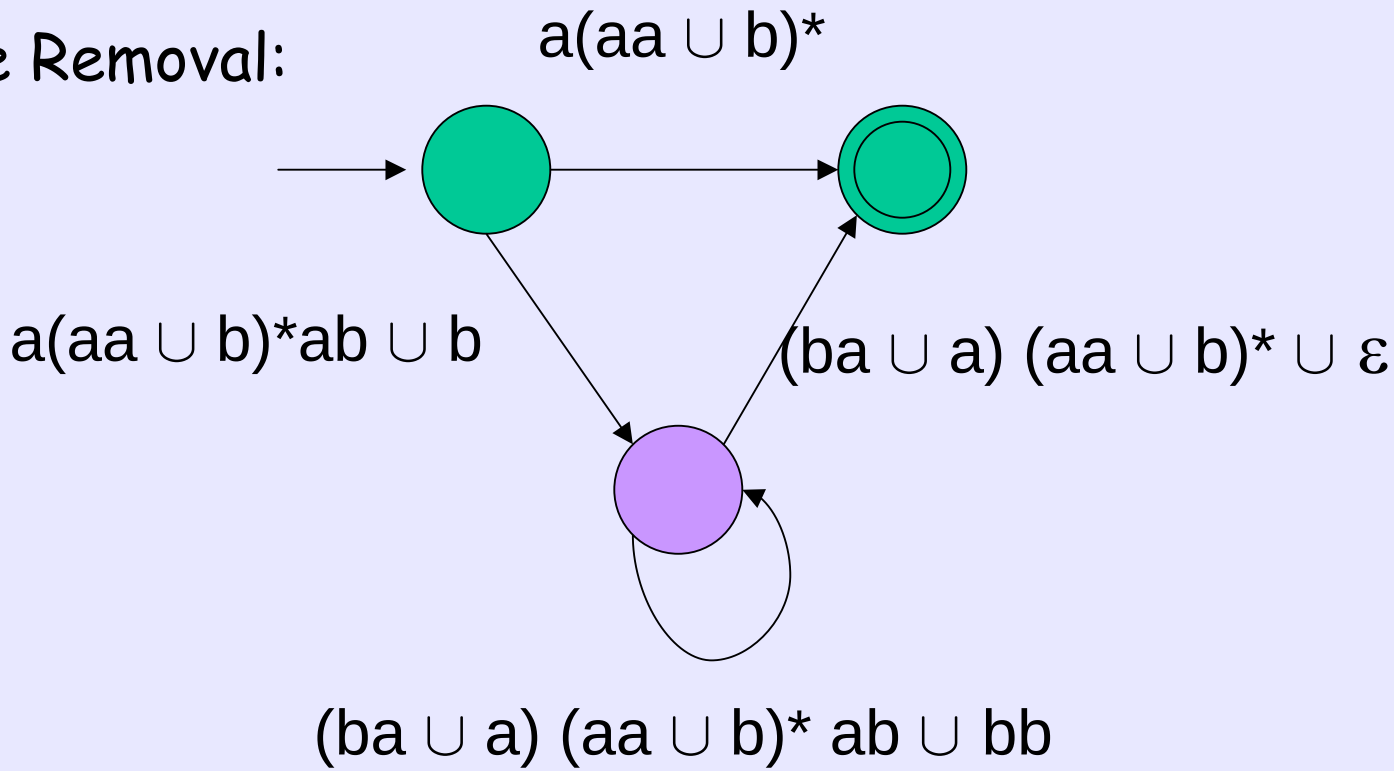
Before Removal



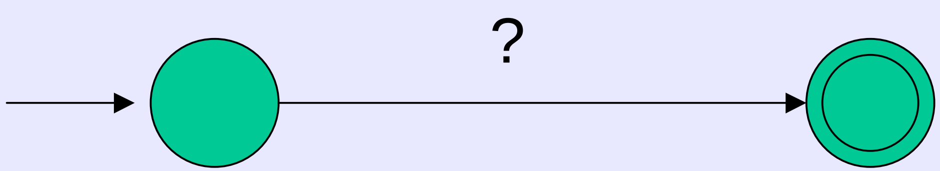
After Removal

Previous Example

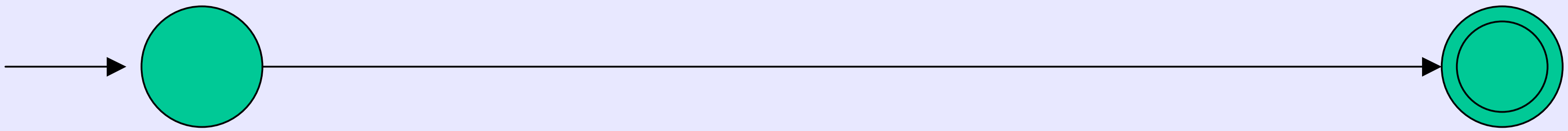
Before Removal:



After Removal:



Final Step



$$(a(aa \cup b)^*ab \cup b)((ba \cup a)(aa \cup b)^* ab \cup bb)^*$$
$$((ba \cup a)(aa \cup b)^* \cup \varepsilon) \cup a(aa \cup b)^*$$

What we have learnt so far

- DFA = NFA
  - proof by construction
- Regular Expression = DFA
  - proof by construction
- Pumping Lemma
  - proof by contradiction
- Existence of Non-regular Languages
  - pumping lemma

Next Time

- Context Free Grammar
  - A more powerful way to describe a language