Attendance

https://forms.gle/iMXPHXs6xW4qn5Ht8

- Midterm 2 on Tuesday (NO ACE SECTION)
- Covers material from lectures 6-16 and problem sets 3-5 and autograded parts of 6
- ACE Midterm resources:
  - Review Session: Saturday 1pm-2:45pm, location TBA (likely LATHROP 298)
    - 60-90 min concept review (recorded)
    - 15-45 min practice questions
  - Printable practice exam (combination of practice resources)
- Resources: Academic Coaching, CTL Tutoring

Strings and Languages

- An **alphabet** is a non-empty set of letters/characters/symbols
  - Typical symbol: Σ
- A **string** is a finite sequence of characters
  - Typical symbol: *w*
  - ε is the **empty string**: $|\varepsilon| = 0$
- A **language** is a set of strings
  - Languages are defined over alphabets, meaning the strings in the language are made of symbols from the alphabet
  - Σ^* is **the set of all strings** made from symbols in Σ
  - Typical symbol: *L*

Languages represent problems

Finite Automata

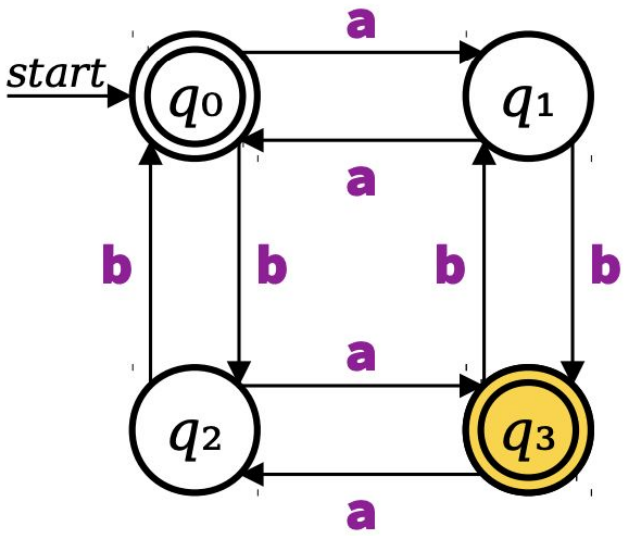
An **automata** is a way we can represent computers in proofs + logic statements

Made of **states, transitions**

Has a **start state**

Some states are **accepting states -** program will accept or reject the input

Finite Automata - has limited memory and produces a yes/no answer



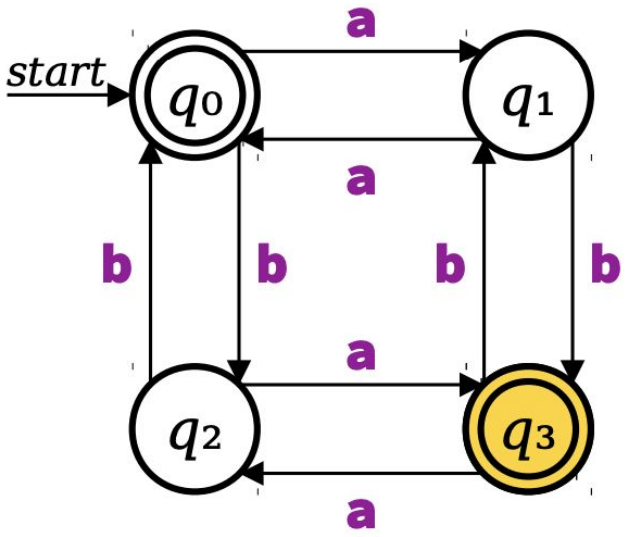
Deterministic Finite Automata (DFA)

For each state, there is **exactly 1** transition for each character option.

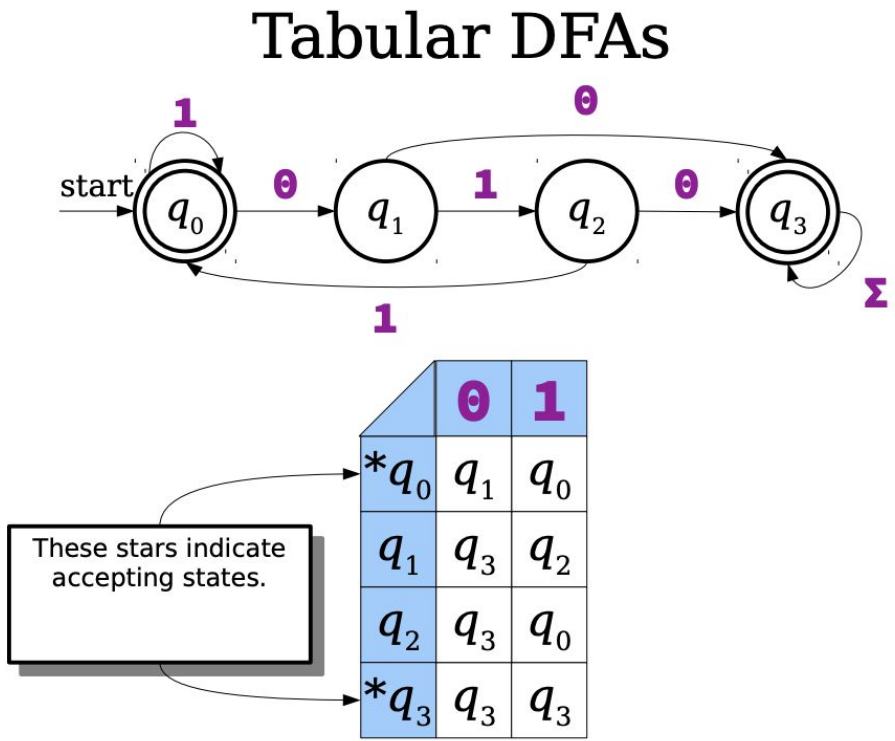
There is **one** start state, and >=0 accepting states

We will never be **unsure** of where to go - it is

DETERMINISTIC



DFAs can also be represented by tables



Dealing with strings

Alphabet Let $\Sigma = \{a, b\}$.

The empty string ε.

Languages We say that L is a ***language over Σ*** if it is a set of strings over Σ.

("a language **made of** the alphabet", ex. English made of the Latin alphabet)

The set of all strings composed from letters in Σ is denoted Σ^*.

(all words you could make out of the Latin alphabet, no matter the language)

Formally, we say that L is a language over Σ if $L \subseteq \Sigma^*$.

Language of Automata

In this way, we can define languages *by* the automata that accepts it

Regular Languages are

recognized by DFAs

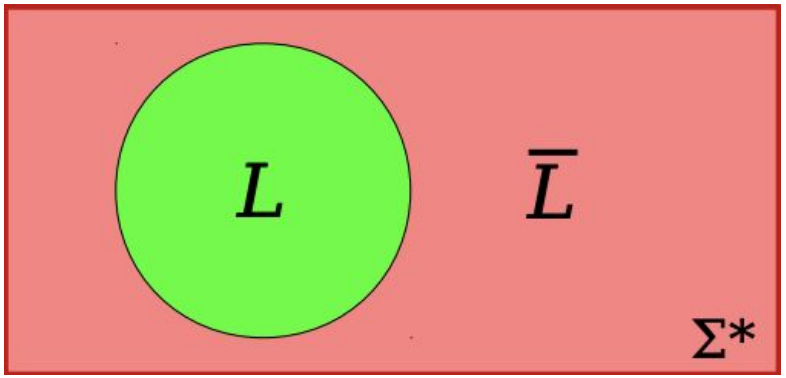
The ***language of A***, denoted $\mathscr{L}(A)$, is the set of strings over Σ that A accepts:

$$\mathscr{L}(A) = \{ w \in \Sigma^* \mid A \text{ accepts } w \}$$

A language L is called a ***regular language*** if there exists a DFA D such that $\mathscr{L}(D) = L$.

If L is a language and $\mathscr{L}(D) = L$, we say that D ***recognizes*** the language L.

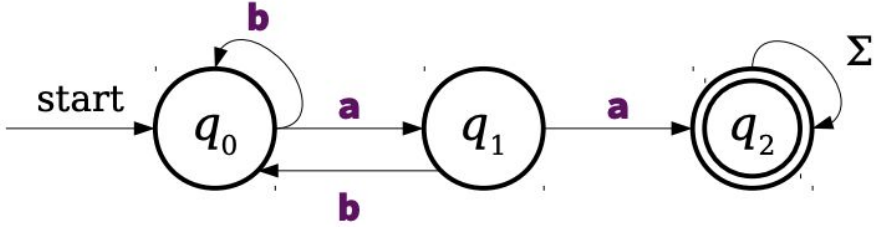
Language Complements



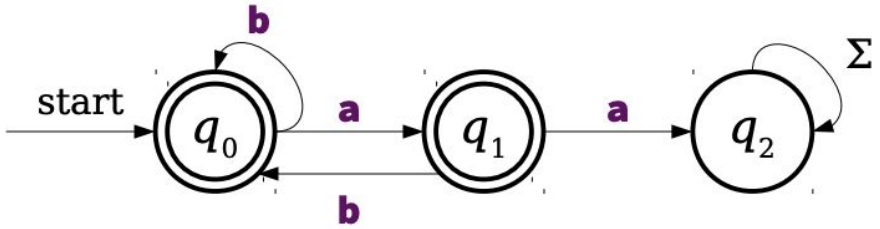
$$\overline{L} = \Sigma^* - L$$

Complements - Swap the accepting states!

$L = \{ w \in \{\mathbf{a}, \mathbf{b}\}^* \mid w \text{ contains } \mathbf{aa} \text{ as a substring} \}$



$\overline{L} = \{ w \in \{\mathbf{a}, \mathbf{b}\}^* \mid w \textbf{\textit{ does not }} \text{contain } \mathbf{aa} \text{ as a substring} \}$



- ***Theorem:*** If L is a regular language, then $\overline{L}$ is also a regular language.

  Because we were able to make a DFA for $\overline{L}$

  If there exists a DFA that **recognizes** the language complement, it is **regular**

- As a result, we say that the regular languages are ***closed under complementation***.

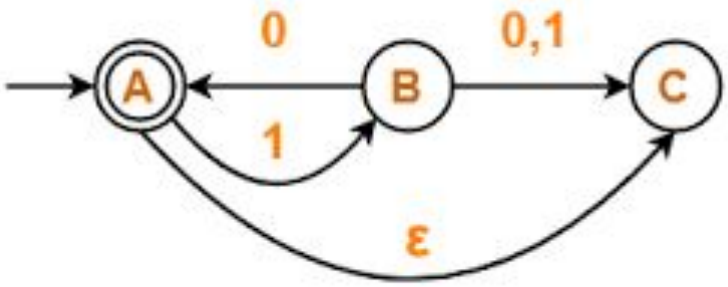
  We will get to many more **closure properties** later on

Nondeterministic Finite Automata (NFA)

- All about CHOICES - each transition can bring you to one, or **many** states
- Acceptance: NFA will accept **if there exists** a path that puts you in an accepting state
- The **epsilon transition:**

> NFAs have a special type of transition called the **ε-transition**.
>
> An NFA may follow any number of ε-transitions at any time without consuming any input.

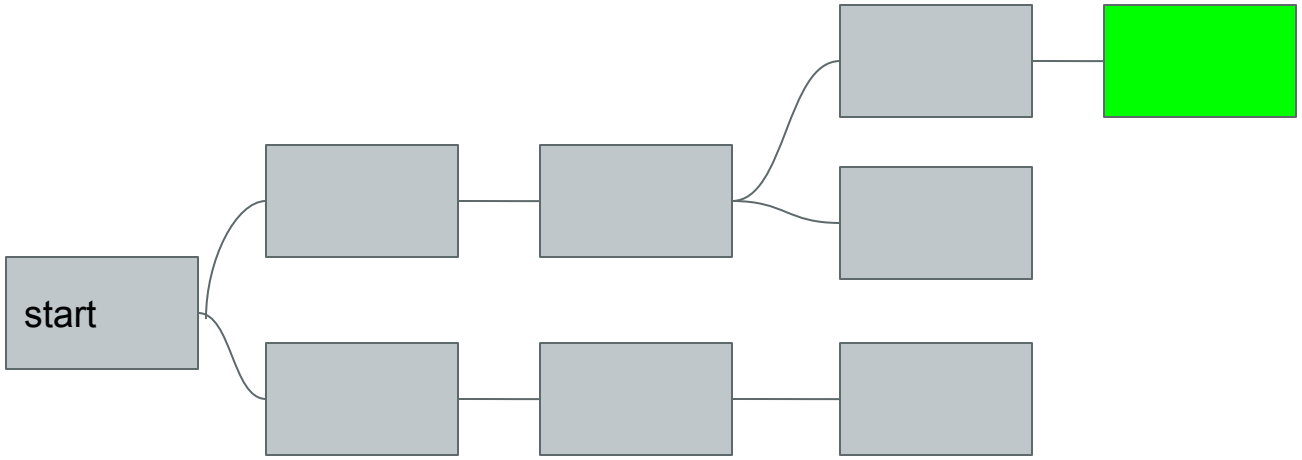
- The **sigma transition**: any char in alphabet



NFA's Cool Capabilities

Perfect Positive Guessing - the machine will pick the right path if there is one

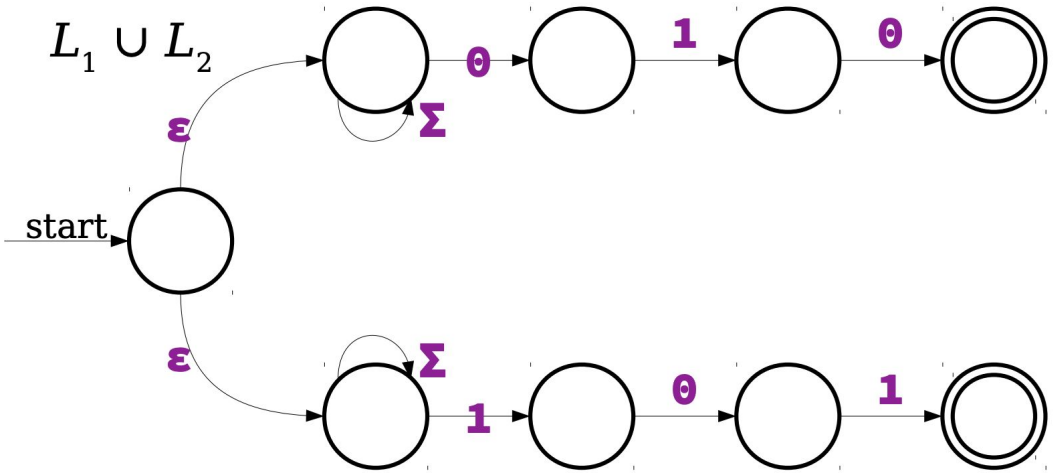
Parallelism - an NFA is sorta a DFA trying all options at once



Building an NFA – Template by doing unions

Epsilon transitions as “bridges” to the different sub-parts of the language

$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$



Languages and Automata

All languages that can be recognized by a DFA can be recognized by an NFA

Why? Because a DFA is an NFA

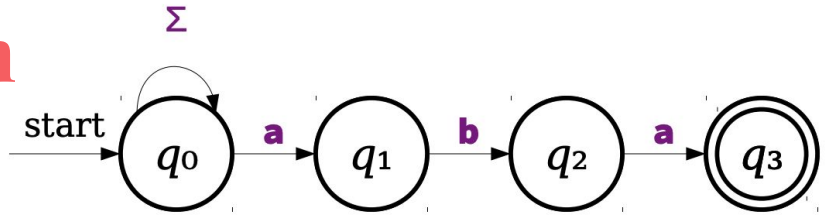
All languages that can be recognized by an NFA can also be recognized by a DFA

Why? **The subset construction** allows us to make DFAs from NFAs

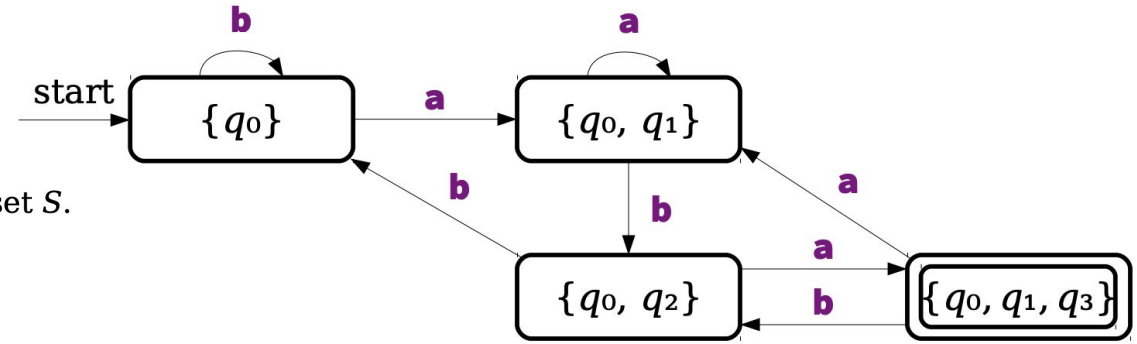
Subset Construction

To convert NFA to DFA, make a table of all the simultaneous states you can occupy, and the options from each set of states

At most, the DFA will have 2^|S| states for an NFA with S states



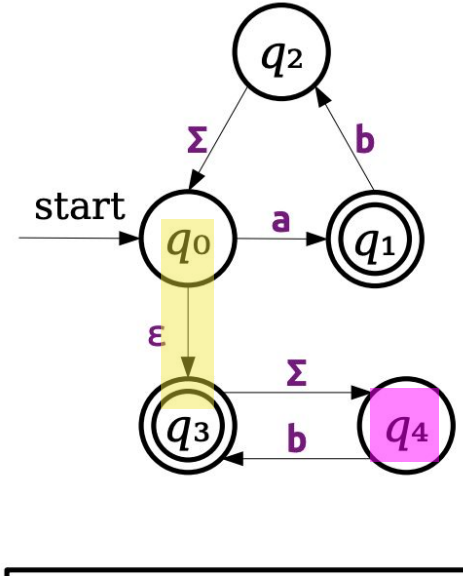
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$*\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



- ***Useful fact:*** $|\wp(S)| = 2^{|S|}$ for any finite set S.

The Trickier example from the “Guide to Subset Construction”

To note: Epsilon transitions and “dying off”



	a	b
*$\{q_0, q_3\}$	$\{q_1, q_4\}$	$\{q_4\}$
*$\{q_1, q_4\}$	Ø	$\{q_2, q_3\}$
$\{q_4\}$	Ø	$\{q_3\}$
*$\{q_2, q_3\}$	$\{q_0, q_3, q_4\}$	$\{q_0, q_3, q_4\}$
*$\{q_3\}$	$\{q_4\}$	$\{q_4\}$
*$\{q_0, q_3, q_4\}$	$\{q_1, q_4\}$	$\{q_3, q_4\}$
*$\{q_3, q_4\}$	$\{q_4\}$	$\{q_3, q_4\}$
Ø	Ø	Ø

Regular languages & closure properties

Regular languages: languages that we can build a DFA or NFA for

- Why do DFAs and NFAs have equivalent “computing power”?
  - Every DFA is an NFA
  - NFAs can be converted into DFAs via the subset construction
- Regular languages are **closed** under set operations
  - In other words, doing the operation produces another regular language

(What do nonregular languages look like? Stay tuned!)

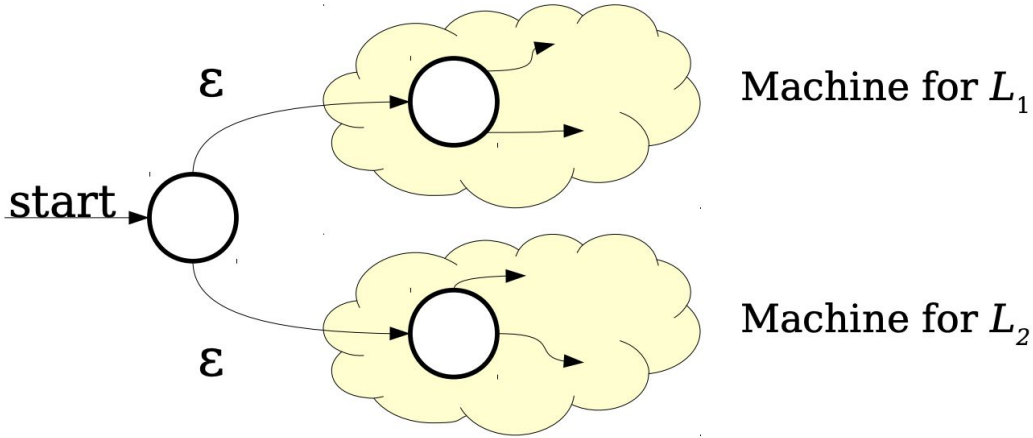
Closure Properties

When combinations of regular languages **stay** regular languages

Union

Proof by constructing the NFA for the new language

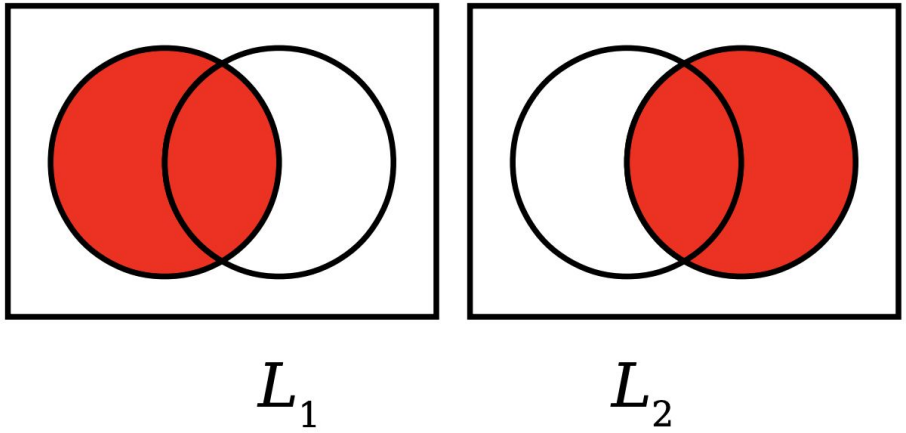
If L_1 and L_2 are regular languages, is $L_1 \cup L_2$?

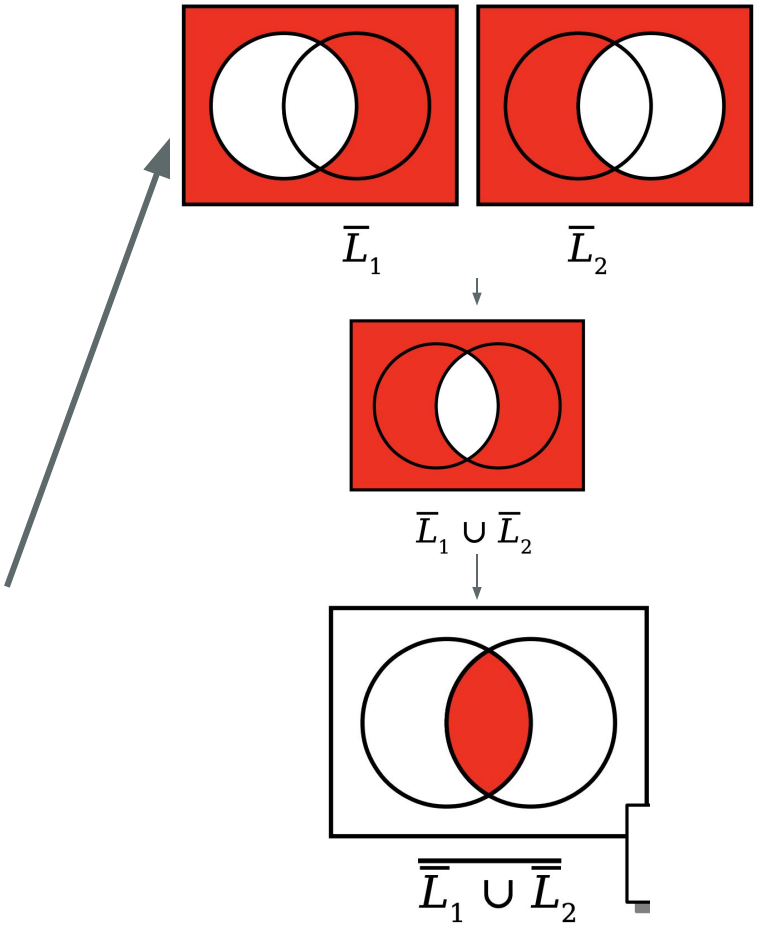


Intersection

Proof by using the complement + union rules

Question: If L_1 and L_2 are regular, is $L_1 \cap L_2$ regular as well?





Language concatenation

- Intuition: $\mathbf{L^n}$ contains all strings made of **n** strings from L **stuck together**
- Formal definition (use this when writing proofs!)

$$L_1 L_2 = \{w \mid \exists x \in L_1.\ \exists y \in L_2.\ w = xy\}$$

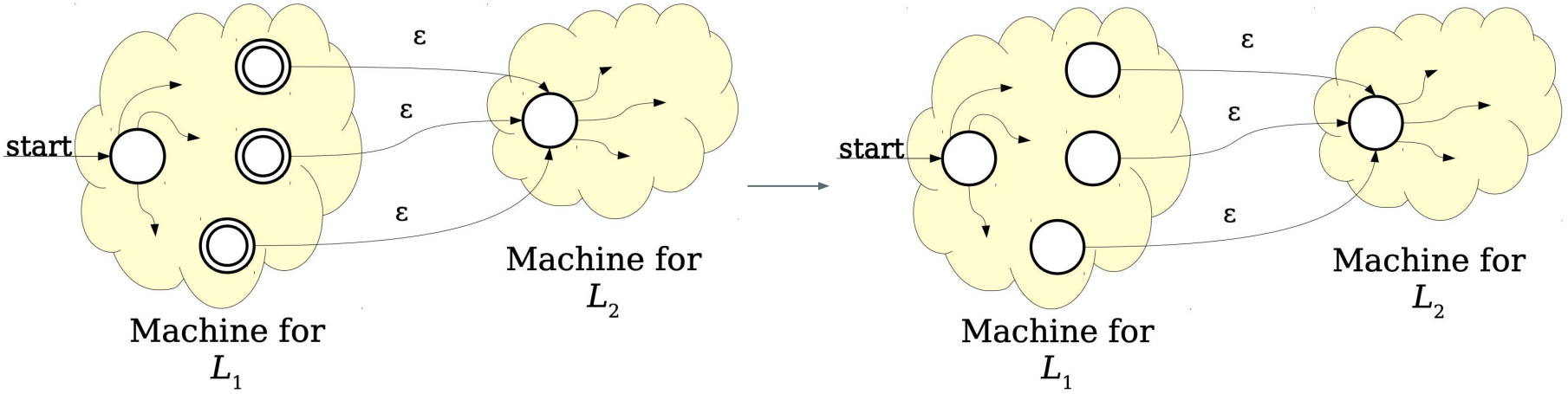
$$L^0 = \{\varepsilon\} \qquad L^{n+1} = LL^n$$

Concatenation

Proof by constructing the NFA for the new language

The ***concatenation*** of two languages L_1 and L_2 over the alphabet Σ is the language

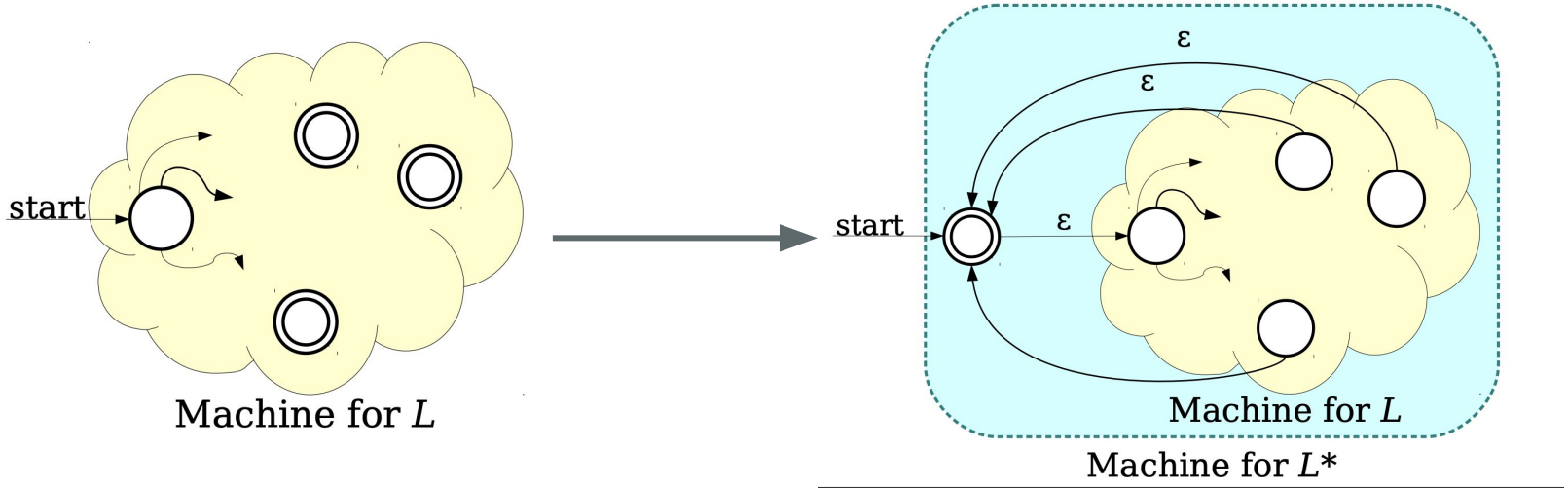
$$L_1L_2 = \{ wx \in \Sigma^* \mid w \in L_1 \wedge x \in L_2 \}$$



The Kleene Star

Proof by constructing the NFA

$$L^* = \{ w \in \Sigma^* \mid \exists n \in \mathbb{N}.\ w \in L^n \}$$



Regular Expressions – a new way to represent a language

R_1R_2 - concatenation

$R_1 \cup R_2$ - union (“or”)

R^* - Kleene closure (zero or more)

Operator Precedence:

$$(R)$$

$$R^*$$

$$R_1R_2$$

$$R_1 \cup R_2$$

Language of a Regex

$$\mathcal{L}(\boldsymbol{\varepsilon}) = \{\varepsilon\}$$

$$\mathcal{L}(\boldsymbol{\emptyset}) = \emptyset$$

$$\mathcal{L}(\mathbf{a}) = \{\mathbf{a}\}$$

$$\mathcal{L}(R_1R_2) = \mathcal{L}(R_1)\,\mathcal{L}(R_2)$$

$$\mathcal{L}(R_1 \cup R_2) = \mathcal{L}(R_1) \cup \mathcal{L}(R_2)$$

$$\mathcal{L}(R^*) = \mathcal{L}(R)^*$$

$$\mathcal{L}((R)) = \mathcal{L}(R)$$

More Shorthand

$\mathbf{R^n}$ is shorthand for **RR** … **R** (n times).

- Edge case: define $\mathbf{R^0} = \varepsilon$.

Σ is shorthand for "any character in Σ."

R? is shorthand for **(R ∪ ε)**, meaning "zero or one copies of R."

$\mathbf{R^+}$ is shorthand for **RR***, meaning "one or more copies of R."