Finite Automata

Part Three

Hello Wonderful Condensed Slide Readers!

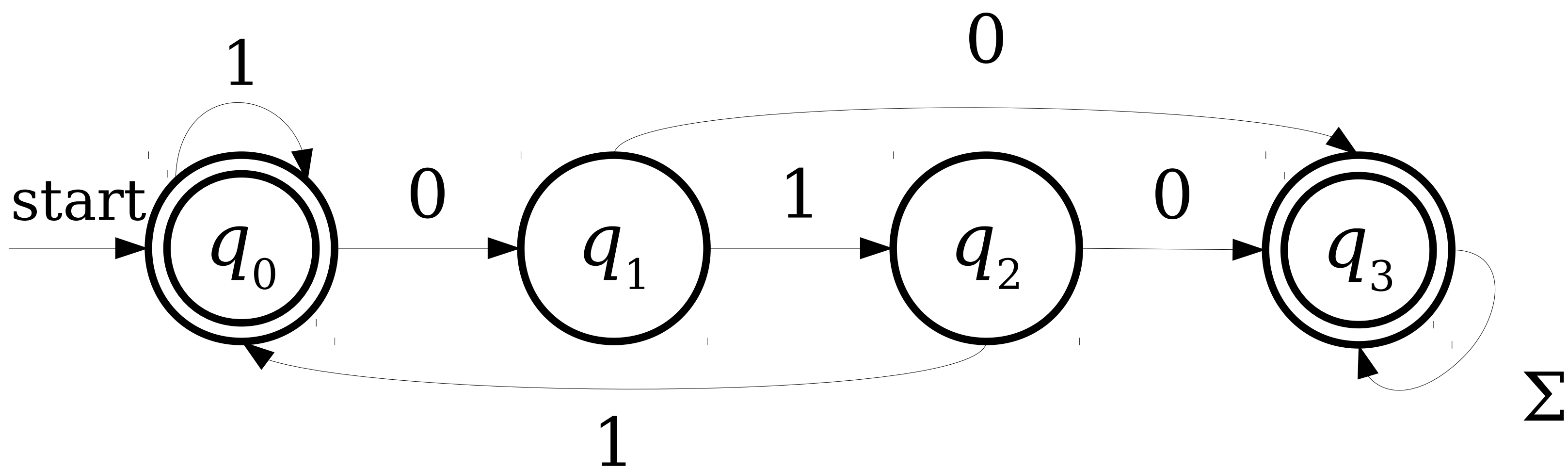
The first half of this lecture consists almost exclusively of animations of the subset construction, which aren't present in these condensed slides. You might want to read over the full version of these slides to get a better sense for how that construction works.

Hope this helps!

-Keith

Recap from Last Time

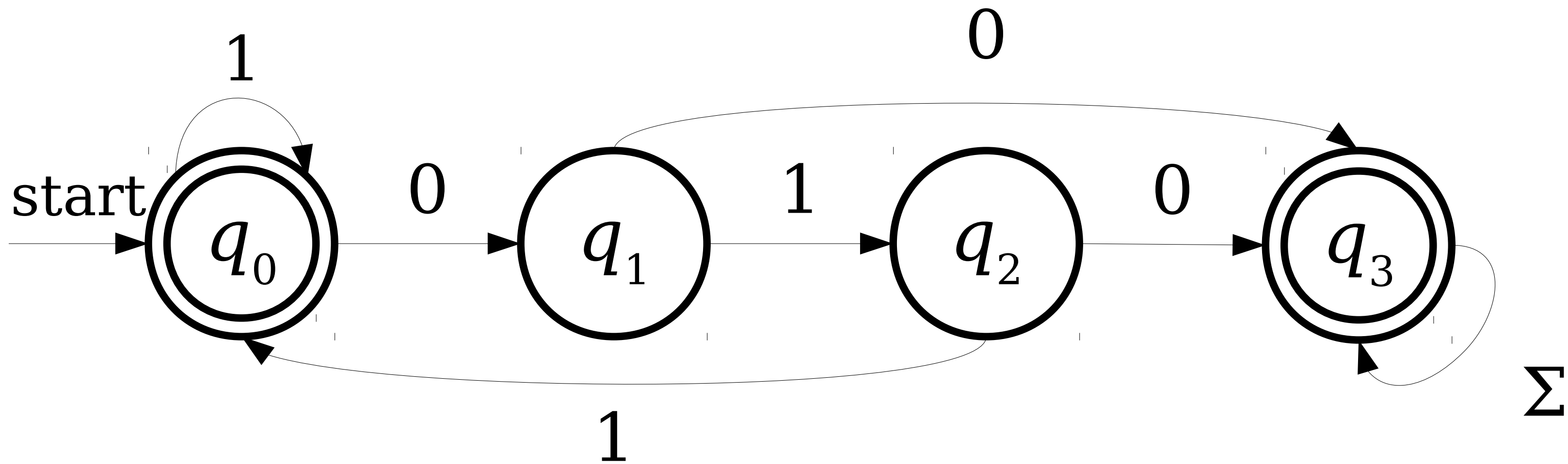
Tabular DFAs



	0	1
$*q_0$	q_1	q_0
q_1	q_3	q_2
q_2	q_3	q_0
$*q_3$	q_3	q_3

These stars indicate accepting states.

Tabular DFAs



	0	1
$*q_0$	q_1	q_0
q_1	q_3	q_2
q_2	q_3	q_0
$*q_3$	q_3	q_3

Since this is the first row, it's the start state.

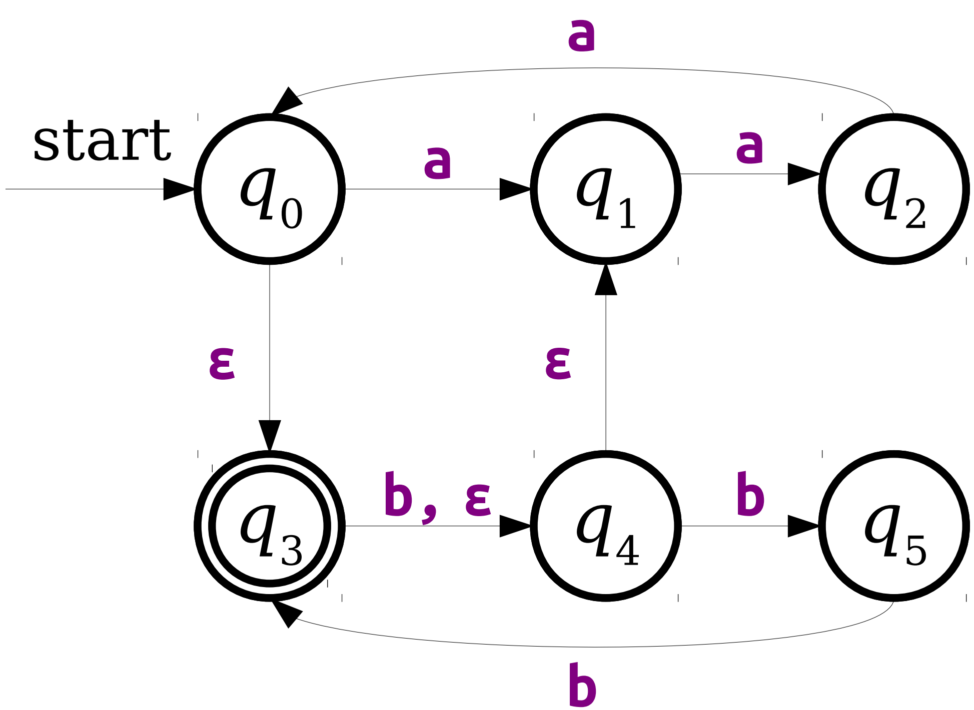
A language L is called a ***regular language***
if there exists a DFA D such that $\mathscr{L}(D) = L$.

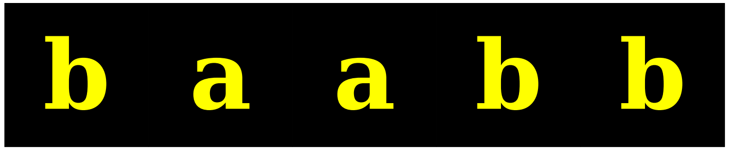
NFAs

- An ***NFA*** is a
  - ***N***ondeterministic
  - ***F***inite
  - ***A***utomaton
- Can have missing transitions or multiple transitions defined on the same input symbol.
- Accepts if *any possible series of choices* leads to an accepting state.

ε-Transitions

- 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.





Perfect Guessing

- We can view nondeterministic machines as having ***Magic Superpowers*** that enable them to guess the correct choice of moves to make.
  - If there is at least one choice that leads to an accepting state, the machine will guess it.
  - If there are no choices, the machine guesses any one of the wrong guesses.
- No known physical analog for this style of computation – this is totally new!

Massive Parallelism

- An NFA can be thought of as a DFA that can be in many states at once.
- At each point in time, when the NFA needs to follow a transition, it tries all the options at the same time.
- The NFA accepts if *any* of the states that are active at the end are accepting states. It rejects otherwise.

Just how powerful *are* NFAs?

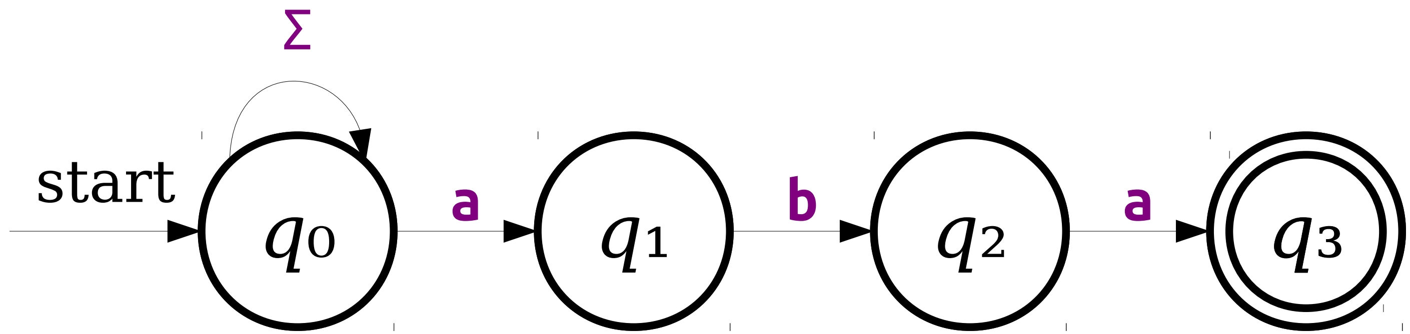
New Stuff!

NFAs and DFAs

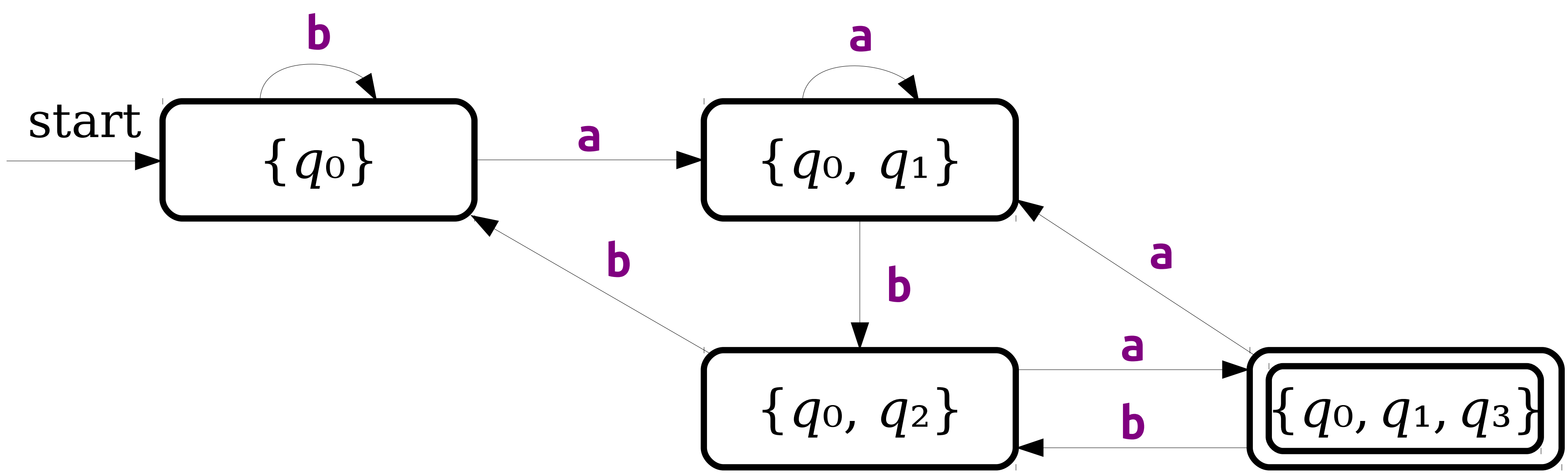
- Any language that can be accepted by a DFA can be accepted by an NFA.
- Why?
    - Every DFA essentially already *is* an NFA!
- ***Question:*** Can any language accepted by an NFA also be accepted by a DFA?
- Surprisingly, the answer is ***yes***!

Thought Experiment:

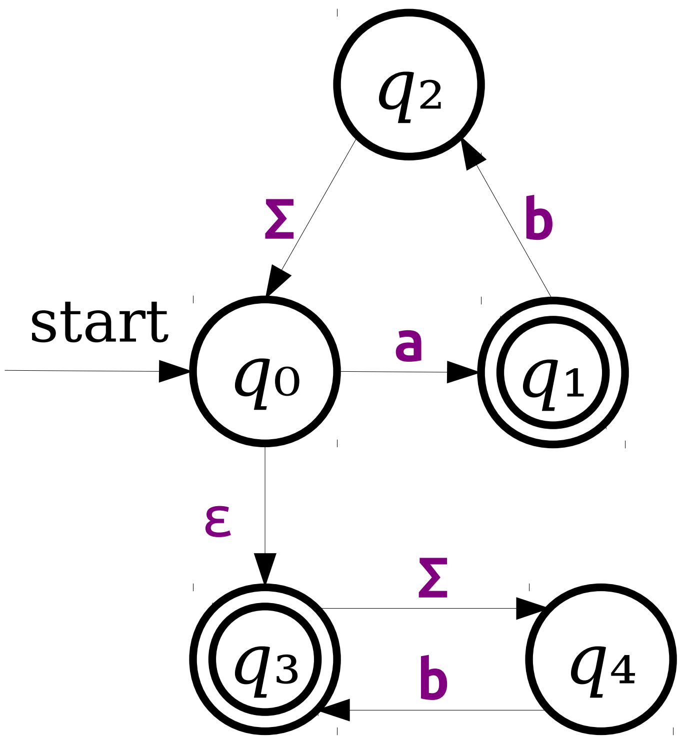
How would you simulate an NFA in software?



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



Once More, With Epsilons!



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

The Subset Construction

- This construction for transforming an NFA into a DFA is called the ***subset construction*** (or sometimes the ***powerset construction***).
  - Each state in the DFA is associated with a set of states in the NFA.
  - The start state in the DFA corresponds to the start state of the NFA, plus all states reachable via ε-transitions.
  - If a state q in the DFA corresponds to a set of states S in the NFA, then the transition from state q on a character `a` is found as follows:
    - Let S' be the set of states in the NFA that can be reached by following a transition labeled `a` from any of the states in S. *(This set may be empty.)*
    - Let S'' be the set of states in the NFA reachable from some state in S' by following zero or more epsilon transitions.
    - The state q in the DFA transitions on `a` to a DFA state corresponding to the set of states S''.
- ***Read Sipser for a formal account.***

The Subset Construction

- In converting an NFA to a DFA, the DFA's states correspond to sets of NFA states.
- ***Useful fact:*** $|\wp(S)| = 2^{|S|}$ for any finite set S.
- In the worst-case, the construction can result in a DFA that is *exponentially larger* than the original NFA.
- Interesting challenge: Find a language for which this worst-case behavior occurs (there are infinitely many of them!)

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

An Important Result

Theorem: A language L is regular iff there is some NFA N such that $\mathscr{L}(N) = L$.

Proof Sketch: If L is regular, there exists some DFA for it, which we can easily convert into an NFA. If L is accepted by some NFA, we can use the subset construction to convert it into a DFA that accepts the same language, so L is regular. ■

Why This Matters

- We now have two perspectives on regular languages:
  - Regular languages are languages accepted by DFAs.
  - Regular languages are languages accepted by NFAs.
- We can now reason about the regular languages in two different ways.

Time-Out for Announcements!

Problem Set Five

- PS5 is due this Friday at the start of class.
- Need help?
  - Stop by office hours!
  - Ask question on Piazza!
- We’ve released a new Inductive Proofwriting Checklist containing some common mistakes on PS5. Take a minute or two to read through it before submitting.

Extra Practice Problems

- We’ve released four more sets of extra practice problems (EPP4 – EPP7) in case you’d like to get some extra practice with the topics from PS3 – PS5.
- We’ll release solutions next week.

SLAE Monthly Event

- ***SLAE*** (***S***tanford ***L***atin***A*** ***E***ngineers) is holding their Very First Monthly Event this Friday from 8PM – 10PM in the Centro Lounge.
- They're providing snacks and Netflix. ☺

BLACK IN COMPUTER SCIENCE PRESENTS

BLACK IN CS
KICKBACK

5 - 7PM | FRIDAY, MAY 12TH
NITERY | COUCHES

FOOD + FUN + FRIENDS

DINNER PROVIDED

Your Questions

"What do you think is the ideal role of technology in education?"

I have to preface this by saying "I'm not an expert," because I'm definitely not.

From experience, technology works well in education when it (1) acts as a force multiplier or (2) makes the abstract tangible. We'll be releasing tools for designing and testing DFAs, which is (1) on the backend and (2) on the frontend. Chris Piech is doing some truly groundbreaking work on option (1). Khan Academy is both (1) and (2), and MOOCs are a (very weak) (1).

You should be skeptical when someone says that they have a new technology that will revolutionize education. All technology does is change the cost of things. It's not going to fix a highly troubled and complicated system overnight.

“Keith. According to the midterm, I am terrible at CS 103 but I love thinking about computers, machines and this class. How do I reconcile the love and the grade”

A single low exam score does <u>not</u> mean that you are terrible at this. It's a single data point saying how you did at a specific point in time. Treat it as a signal talking about where to focus your efforts and improve.

If you like thinking about this stuff, that's great! Keep doing so. Use your curiosity to drive your exploration of these topics.

Hang in there. You can do this!

Back to CS103!

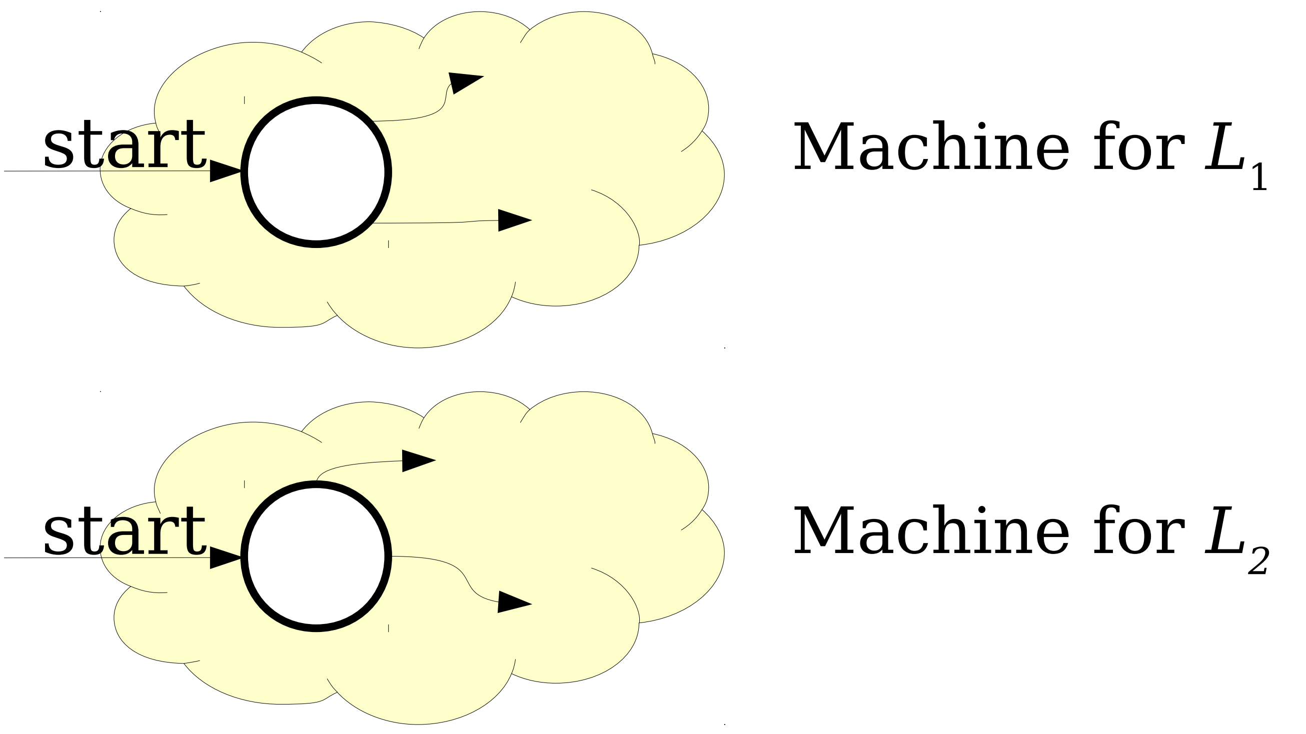
Properties of Regular Languages

The Union of Two Languages

- If L_1 and L_2 are languages over the alphabet Σ, the language $L_1 \cup L_2$ is the language of all strings in at least one of the two languages.
- If L_1 and L_2 are regular languages, is $L_1 \cup L_2$?

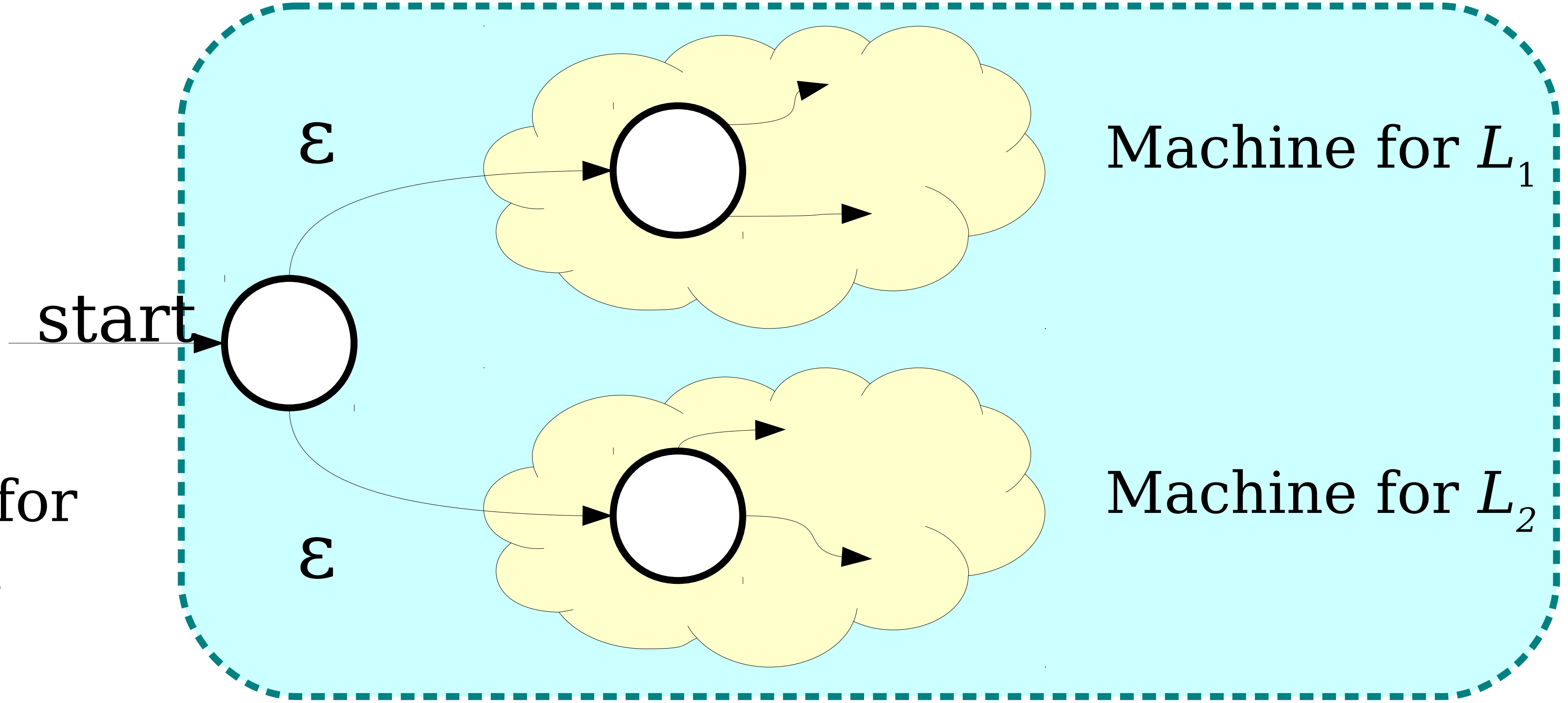
The Union of Two Languages

- If L_1 and L_2 are languages over the alphabet Σ, the language $L_1 \cup L_2$ is the language of all strings in at least one of the two languages.
- If L_1 and L_2 are regular languages, is $L_1 \cup L_2$?



The Union of Two Languages

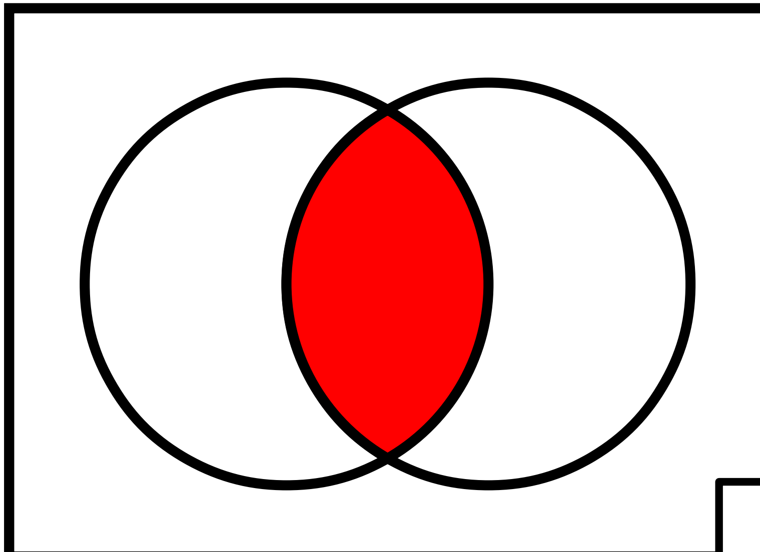
- If L_1 and L_2 are languages over the alphabet Σ, the language $L_1 \cup L_2$ is the language of all strings in at least one of the two languages.
- If L_1 and L_2 are regular languages, is $L_1 \cup L_2$?



Machine for $L_1 \cup L_2$

The Intersection of Two Languages

- If L_1 and L_2 are languages over Σ, then $L_1 \cap L_2$ is the language of strings in both L_1 and L_2.
- Question: If L_1 and L_2 are regular, is $L_1 \cap L_2$ regular as well?



$$\overline{\overline{L_1} \cup \overline{L_2}}$$

Hey, it's De Morgan's laws!

Concatenation

String Concatenation

- If $w \in \Sigma^*$ and $x \in \Sigma^*$, the ***concatenation*** of w and x, denoted wx, is the string formed by tacking all the characters of x onto the end of w.
- Example: if w = **quo** and x = **kka**, the concatenation wx = **quokka**.
- Analogous to the + operator for strings in many programming languages.
- Some facts about concatenation:
  - The empty string ε is the ***identity element*** for concatenation:

    $$w\varepsilon = \varepsilon w = w$$
  - Concatenation is ***associative***:

    $$wxy = w(xy) = (wx)y$$

Concatenation

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

$$\boldsymbol{L_1 L_2 = \{ wx \in \Sigma^* \mid w \in L_1 \land x \in L_2 \}}$$

Concatenation Example

- Let Σ = { **a**, **b**, …, **z**, **A**, **B**, …, **Z** } and consider these languages over Σ:
  - ***Noun*** = { **Puppy**, **Rainbow**, **Whale**, … }
  - ***Verb*** = { **Hugs**, **Juggles**, **Loves**, … }
  - ***The*** = { **The** }
- The language ***TheNounVerbTheNoun*** is
  - { **ThePuppyHugsTheWhale**, **TheWhaleLovesTheRainbow**, **TheRainbowJugglesTheRainbow**, … }

Concatenation

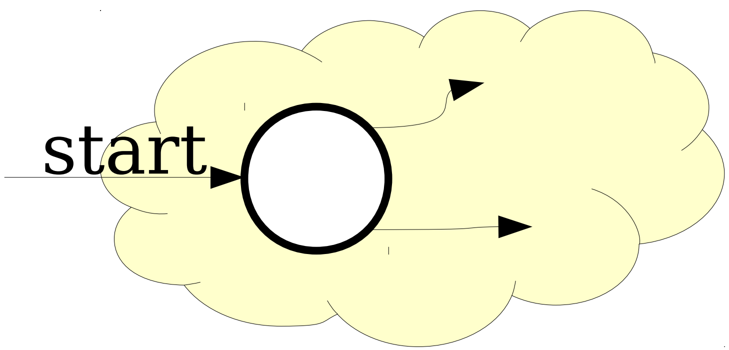
- The ***concatenation*** of two languages L_1 and L_2 over the alphabet Σ is the language

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

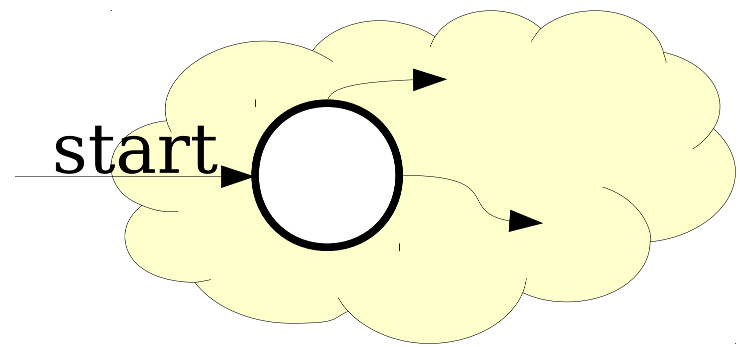
- Two views of L_1L_2:
  - The set of all strings that can be made by concatenating a string in L_1 with a string in L_2.
  - The set of strings that can be split into two pieces: a piece from L_1 and a piece from L_2.
- Conceptually similar to the Cartesian product of two sets, only with strings.

Concatenating Regular Languages

- If L_1 and L_2 are regular languages, is L_1L_2?
- Intuition – can we split a string w into two strings xy such that $x \in L_1$ and $y \in L_2$?



Machine for L_1



Machine for L_2

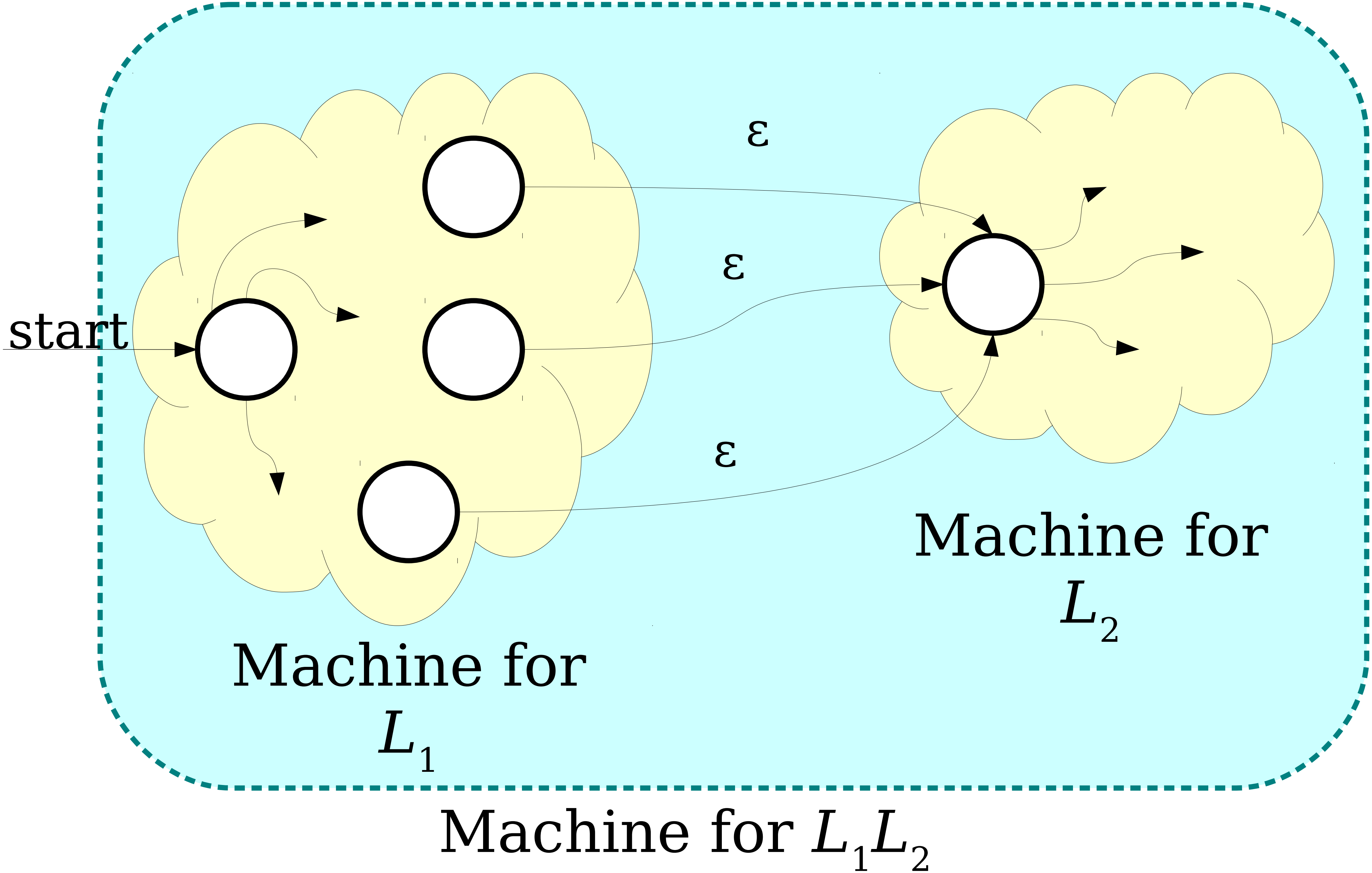
b	**o**	**o**	**k**

k	**e**	**e**	**p**	**e**	**r**

Concatenating Regular Languages

- If L_1 and L_2 are regular languages, is L_1L_2?
- Intuition – can we split a string w into two strings xy such that $x \in L_1$ and $y \in L_2$?
- ***Idea***: Run the automaton for L_1 on w, and whenever L_1 reaches an accepting state, optionally hand the rest off w to L_2.
  - If L_2 accepts the remainder, then L_1 accepted the first part and the string is in L_1L_2.
  - If L_2 rejects the remainder, then the split was incorrect.

Concatenating Regular Languages



Summary

- NFAs are a powerful type of automaton that allows for ***nondeterministic*** choices.
- NFAs can also have ***ε-transitions*** that move from state to state without consuming any input.
- The ***subset construction*** shows that NFAs are not more powerful than DFAs, because any NFA can be converted into a DFA that accepts the same language.
- The union, intersection, complement, and concatenation of regular languages are all regular languages.