Finite Automata

Part Three

Recap from Last Time

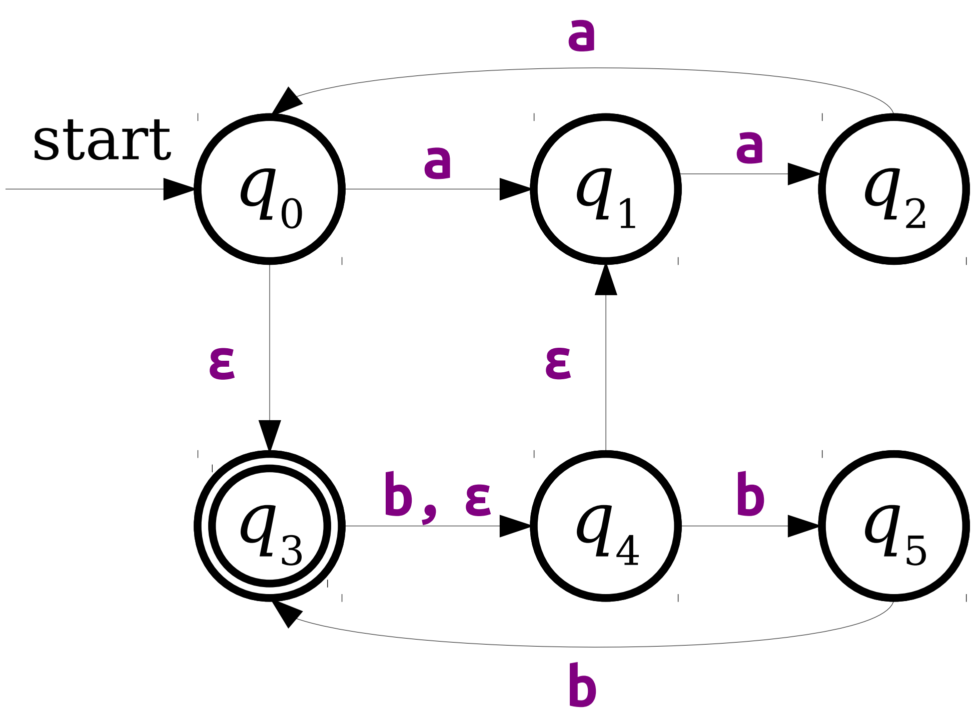
A language L is called a ***regular language*** if there exists a DFA D such that $\mathscr{L}(D) = L$.

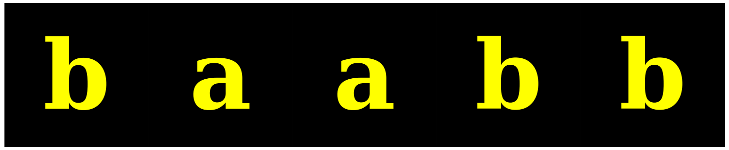
NFAs

- 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.





New Stuff!

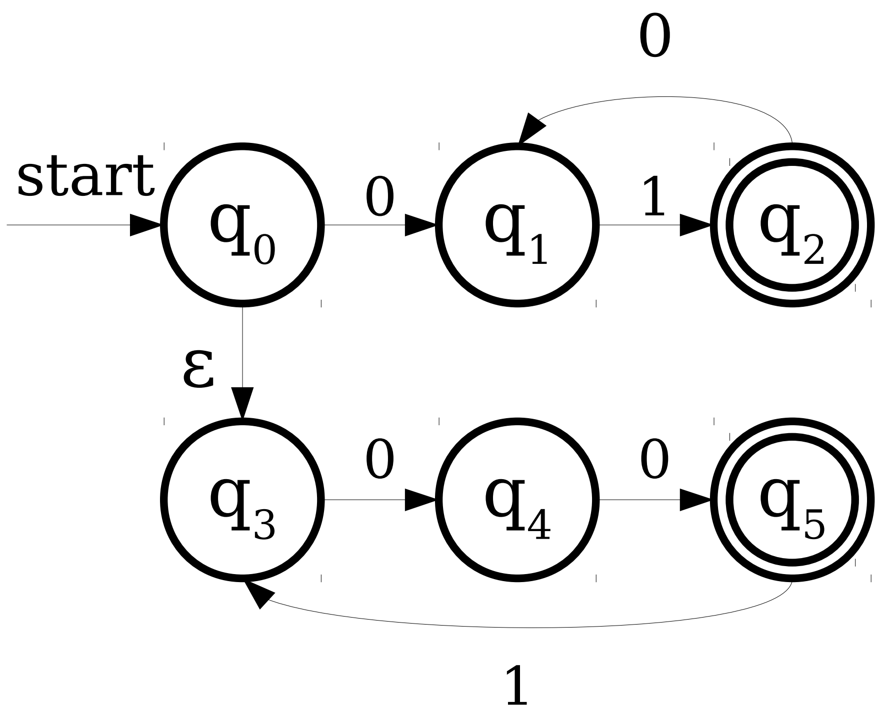
NFAs and DFAs

- Any language that can be accepted by a DFA can be accepted by an NFA.
- Why?
  - Just use the same set of transitions as before.
- ***Question:*** Can any language accepted by an NFA also be accepted by a DFA?
- Surprisingly, the answer is ***yes!***

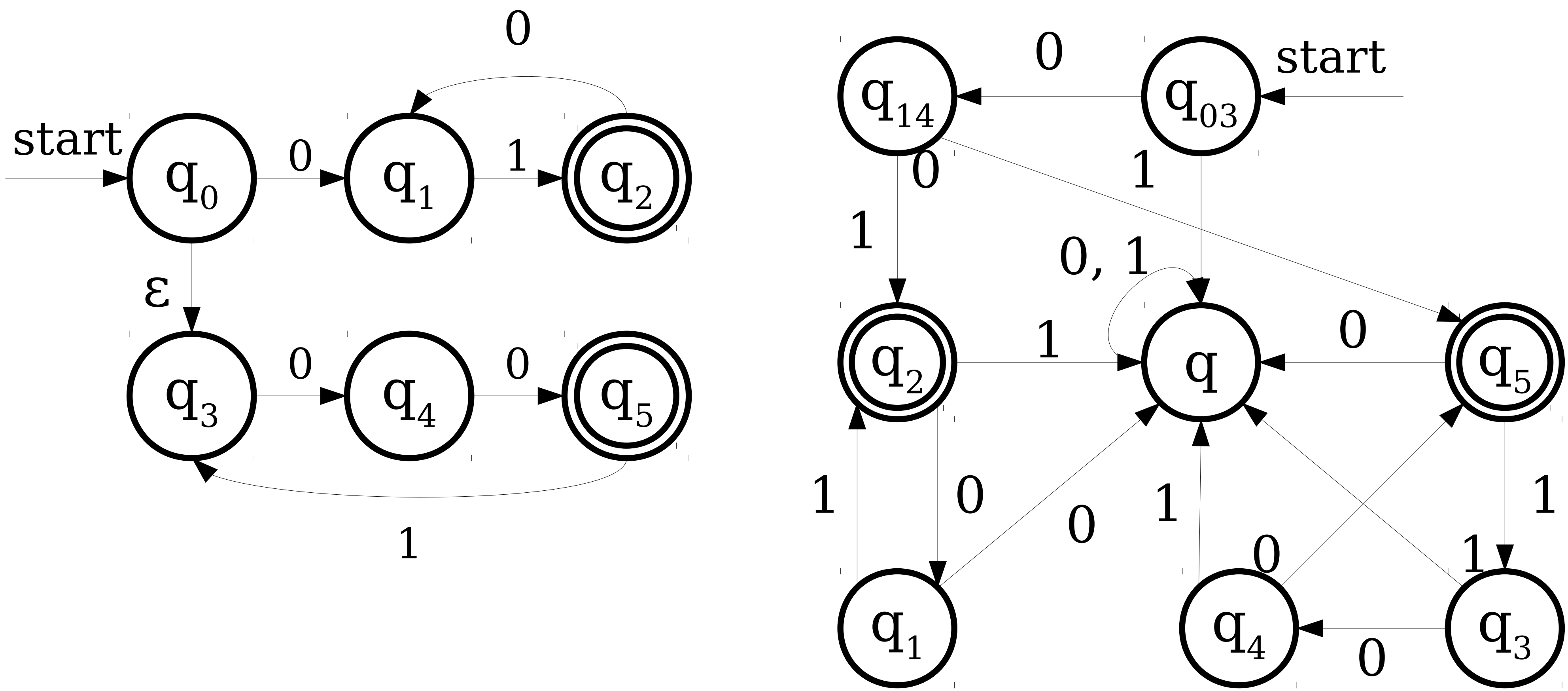
Finite Automata

- NFAs and DFAs are *finite* automata; there can only be finitely many states in an NFA or DFA.
- An NFA can be in any combination of its states, but there are only finitely many possible combations.
- ***Idea:*** Build a DFA where each state of the DFA corresponds to a *set* of states in the NFA.

Simulating an NFA with a DFA



Simulating an NFA with a DFA

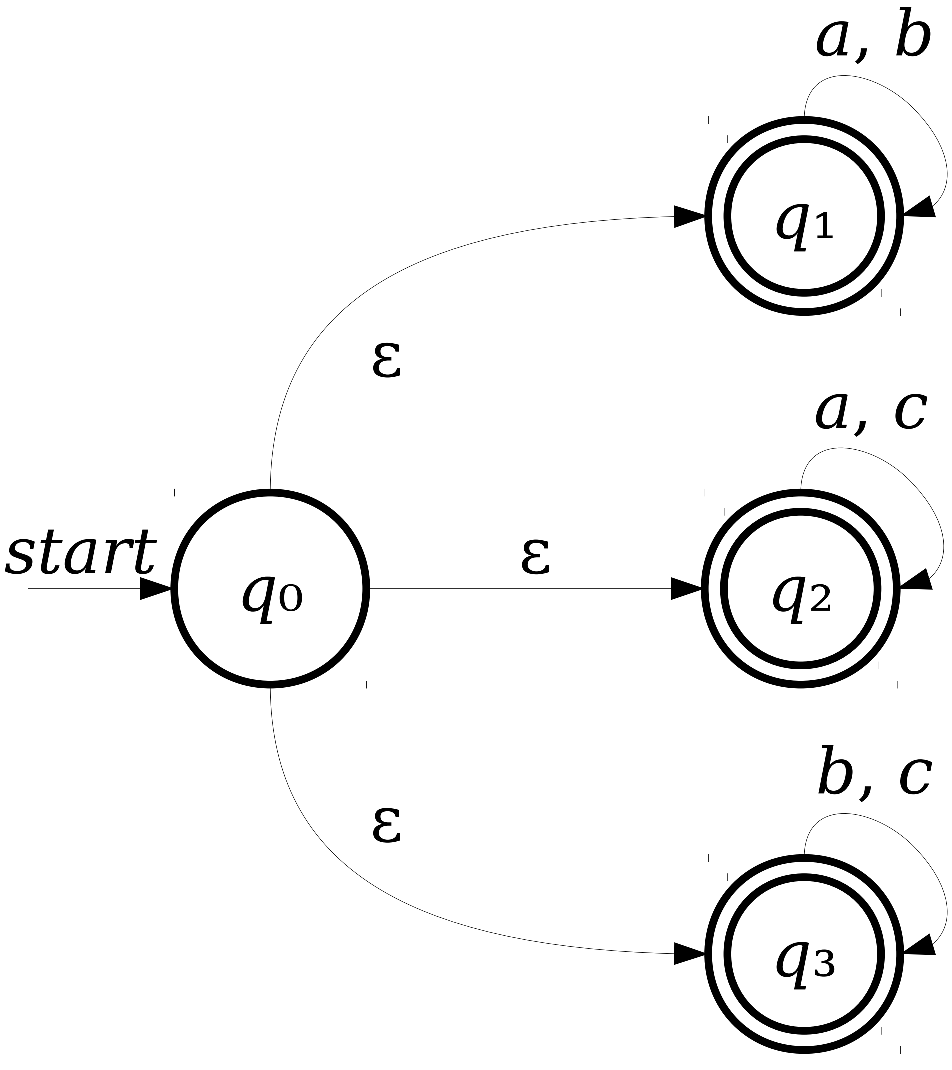


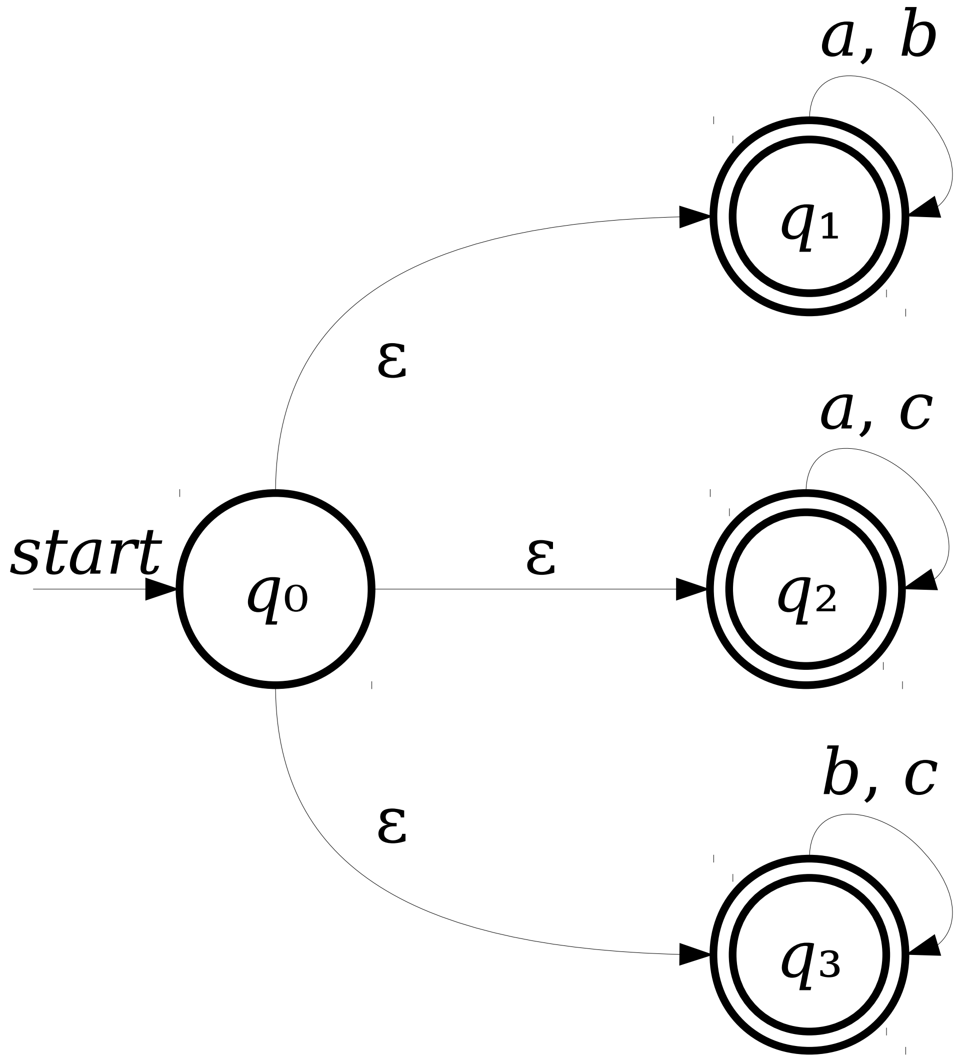
The Subset Construction

- 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, b
a, c
b, c
ε
ε
ε
start
q0
q1
q2
q3



	a	b	c
$*q_{0123}$	q_{12}	q_{13}	q_{23}
$*q_{12}$	q_{12}	q_1	q_2
$*q_{13}$	q_1	q_{13}	q_3
$*q_{23}$	q_2	q_3	q_{23}
$*q_1$	q_1	q_1	q
$*q_2$	q_2	q	q_2
$*q_3$	q	q_3	q_3
q	q	q	q

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 Sets

- Problem Set Five is due right now.
  - Feel free to use late days if you'd like.
- Problem Set Six goes out now. It's due on Friday, November 6.
  - Play around with finite automata, regular expressions, and their properties!
  - Some of the material requires Monday's lecture; it should be well-marked as such.
- A note: ***the printed problem sets are missing the last page.*** Sorry about that – I'm not sure what went wrong there. Check the website for the most up-to-date version.
- Problem Set Four should be graded by the end of class today.

Stanford Women
In Computer Science

CASUAL CS DINNER

{w}

Tuesday Nov. 3 from 5-7pm at the WCC
RSVP at http://bit.ly/1O41hkS

Come have dinner with CS students and faculty.
Everyone is welcome, especially students
just starting out in CS!

My Questions

This is a really rough time of the quarter for a lot of people.

How are *you* doing? How are your dormmates / roommates / friends doing?

What can we do to help out?

Your Questions

"Tell us a cool story from your undergrad ☺"

I'm just going to tell this story in class. Sorry for those of you reading online!

“Why isn't a domino called a biomino? Is a square a monomino?”

Three things is a triple. Four things is a quadruple. Why isn't a pair a duple? Or a buple? I think the answer is “that sounds silly.” And I think that's the answer to your question. ☺

“What are the consequences associated with getting a No Credit on your transcript? (I'm in CS107 right now but seriously feel that I don't have time for this. I want to take it next quarter but am afraid it will cause me trouble later on)”

To be perfectly honest – it's not as bad as you might think. It hurts your GPA for a quarter and you might get a call from the registrar, but if you retake the course your old grade is replaced (you get an “RP” on your transcript) and there shouldn't be any lingering effects. If you get a *lot* of NC or NP grades, then it's a different story. Also, if you're on a scholarship, it might be a problem, but generally the university is supportive when this happens. The main questions they'll ask are “is everything okay?” and “what can we do to help?” rather than “why didn't you pass?”

“What are your favorite novels?”

I confess that I haven't read as many as I would have liked (I tend to read a lot more nonfiction than fiction). I really liked “The Trial” by Kafka and “Radetzky March” by Joseph Roth.

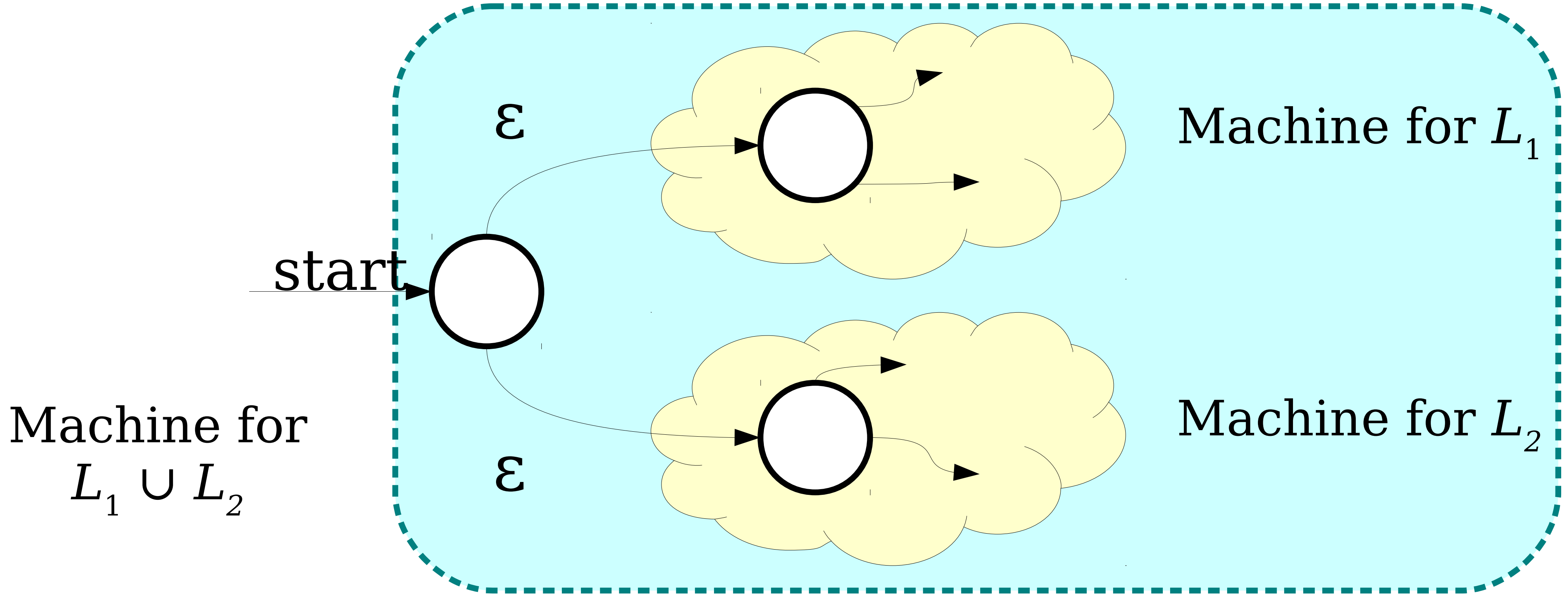
Do you have any suggestions for me? If so, please send me an email!

Back to CS103!

Properties of Regular Languages

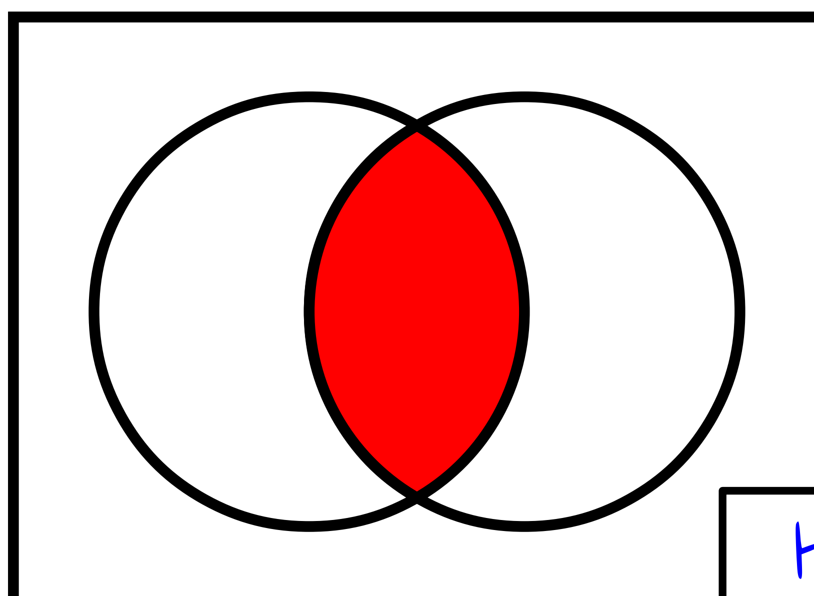
The Union of Two Languages

- 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 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.

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

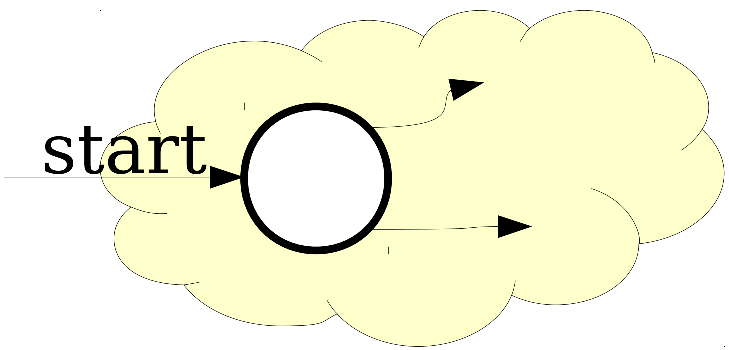
- The ***concatenation*** of two languages L_1 and L_2 over the alphabet Σ is the language

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

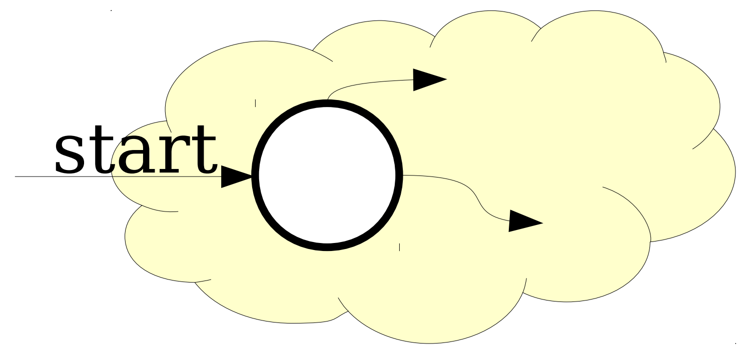
- Two views of L_1L_2:
  - The set of all strings that can be made by concatenating a string in L_1 with a string in L_2.
  - The set of strings that can be split into two pieces: a piece from L_1 and a piece from L_2.
- Conceptually similar to the Cartesian product of two sets, only with strings.

Concatenating Regular Languages

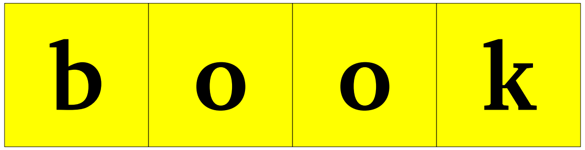
- If L_1 and L_2 are regular languages, is L_1L_2?
- Intuition – can we split a string w into two strings xy such that $x \in L_1$ and $y \in L_2$?

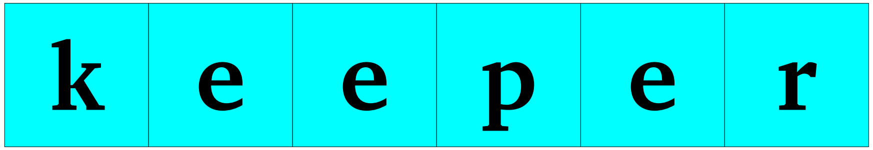


Machine for L_1



Machine for L_2

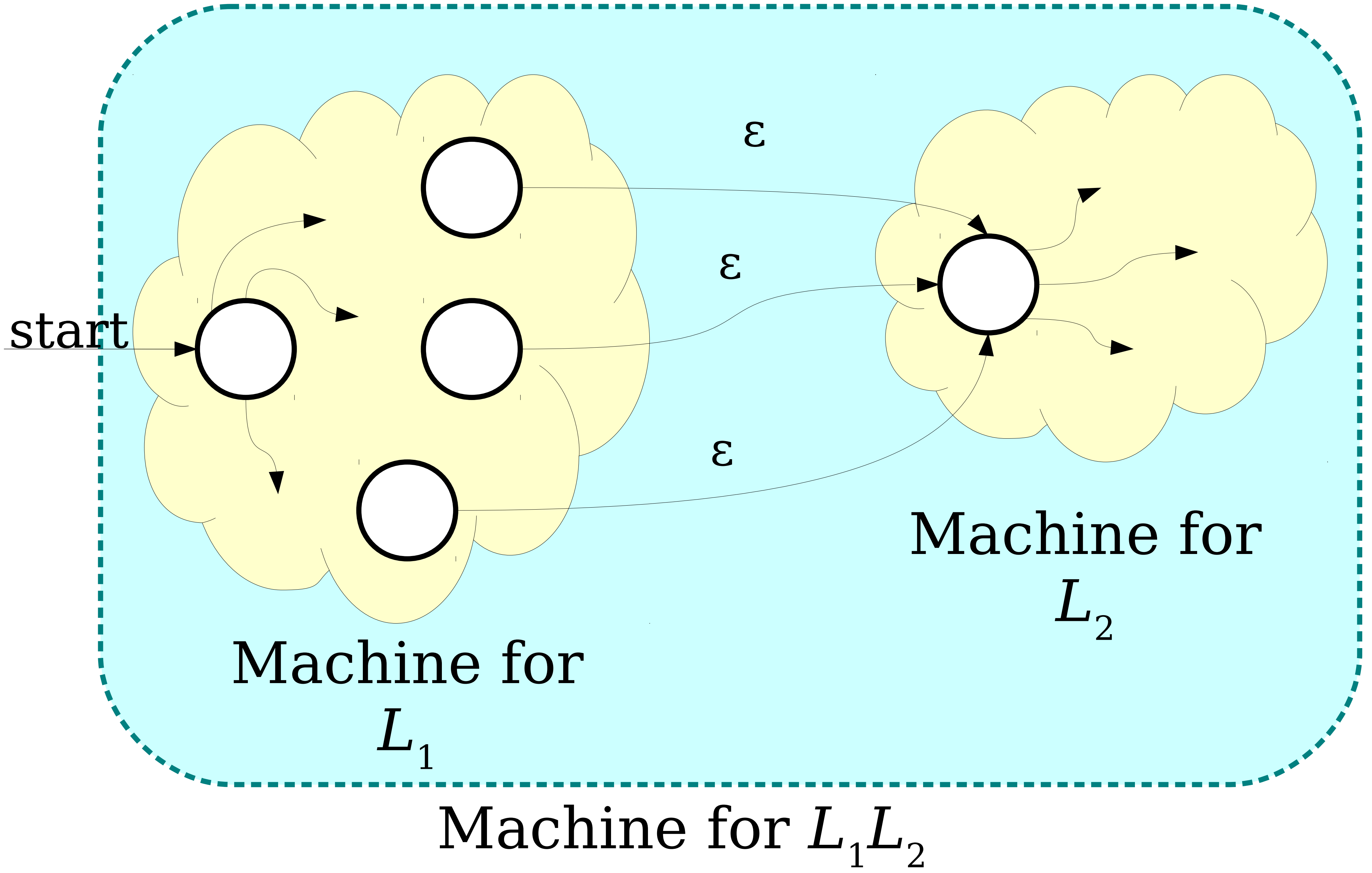




Concatenating Regular Languages

- 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



Lots and Lots of Concatenation

- Consider the language L = { **aa**, **b** }
- LL is the set of strings formed by concatenating pairs of strings in L.

{ **aaaa**, **aab**, **baa**, **bb** }

- LLL is the set of strings formed by concatenating triples of strings in L.

{ **aaaaaa**, **aaaab**, **aabaa**, **aabb**, **baaaa**, **baab**, **bbaa**, **bbb**}

- $LLLL$ is the set of strings formed by concatenating quadruples of strings in L.

{ **aaaaaaaa**, **aaaaaab**, **aaaabaa**, **aaaabb**, **aabaaaa**, **aabaab**, **aabbaa**, **aabbb**, **baaaaaa**, **baaaab**, **baabaa**, **baabb**, **bbaaaa**, **bbaab**, **bbbaa**, **bbbb**}

Language Exponentiation

- We can define what it means to “exponentiate” a language as follows:
- $L^0 = \{\varepsilon\}$
  - The set containing just the empty string.
  - Idea: Any string formed by concatenating zero strings together is the empty string.
- $L^{n+1} = LL^n$
  - Idea: Concatenating ($n+1$) strings together works by concatenating n strings, then concatenating one more.
- ***Question:*** Why define $L^0 = \{\varepsilon\}$?

The Kleene Closure

- An important operation on languages is the ***Kleene Closure***, which is defined as

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

- Mathematically:

$$\boldsymbol{w \in L^* \quad \textbf{iff} \quad \exists n \in \mathbb{N}.\ w \in L^n}$$

- Intuitively, all possible ways of concatenating any number of copies of strings in L together.

The Kleene Closure

If L = { **a**, **bb** }, then L^* = {

ε,

a, **bb**,

aa, **abb**, **bba**, **bbbb**,

aaa, **aabb**, **abba**, **abbbb**, **bbaa**, **bbabb**, **bbbba**, **bbbbbb**,

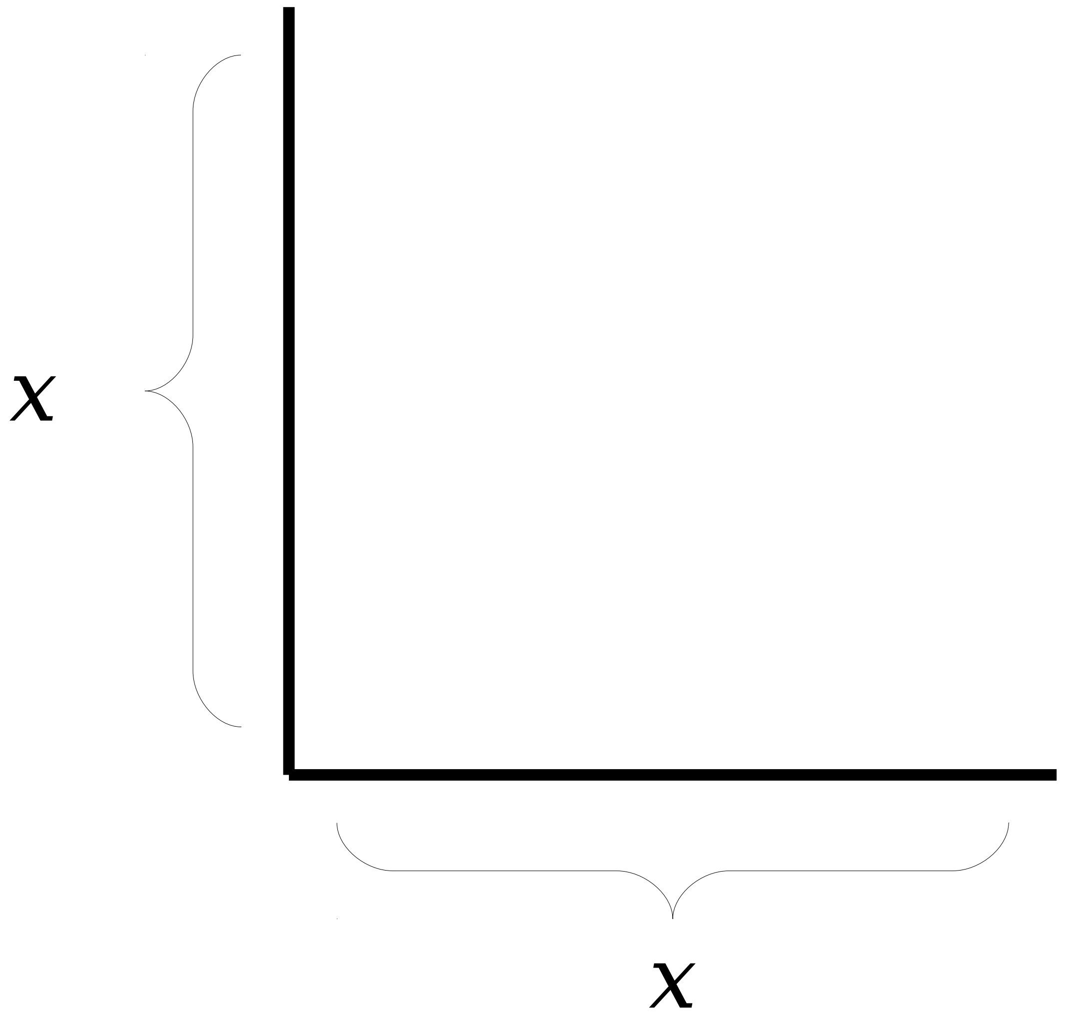
…

}

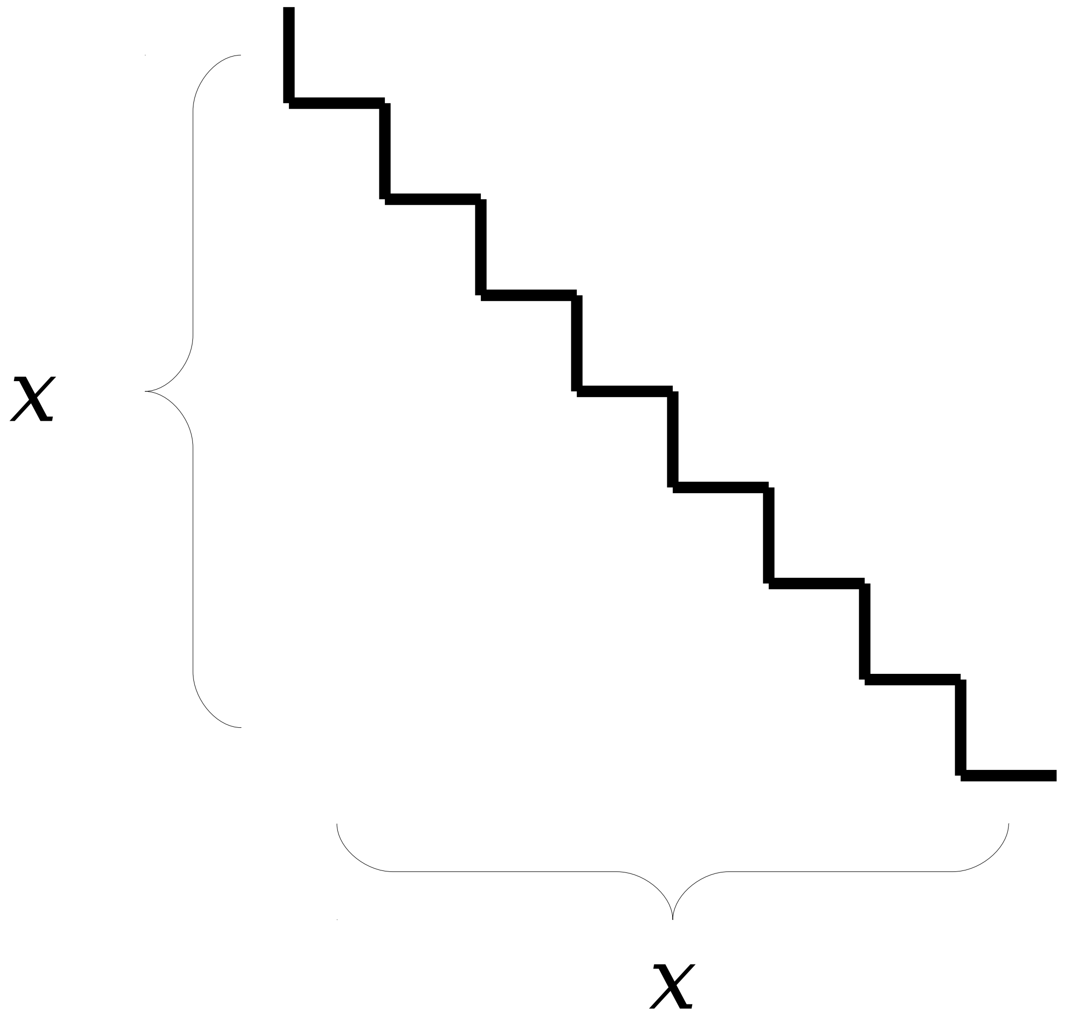
Reasoning about Infinity

- If L is regular, is L^* necessarily regular?
- **A Bad Line of Reasoning**:
  - $L^0 = \{ \varepsilon \}$ is regular.
  - $L^1 = L$ is regular.
  - $L^2 = LL$ is regular
  - $L^3 = L(LL)$ is regular
  - …
  - Regular languages are closed under union.
  - So the union of all these languages is regular.

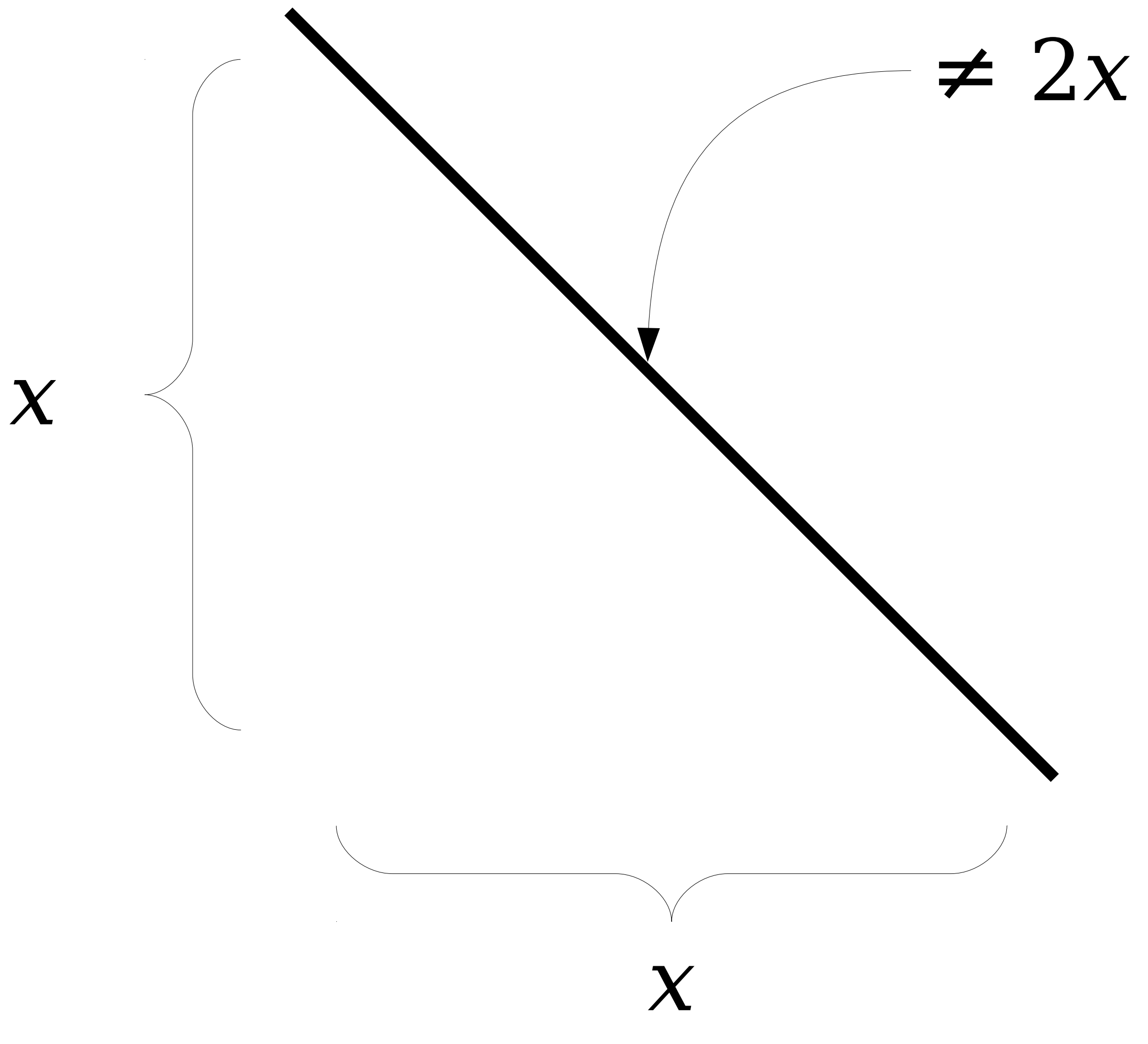
Reasoning about Infinity



Reasoning about Infinity



Reasoning about Infinity



Reasoning about Infinity

$$0.999 < 1$$

Reasoning about Infinity

$$0.9999\overline{9} \not< 1$$

Reasoning about Infinity

2 is finite

Reasoning about Infinity

∞ is finite

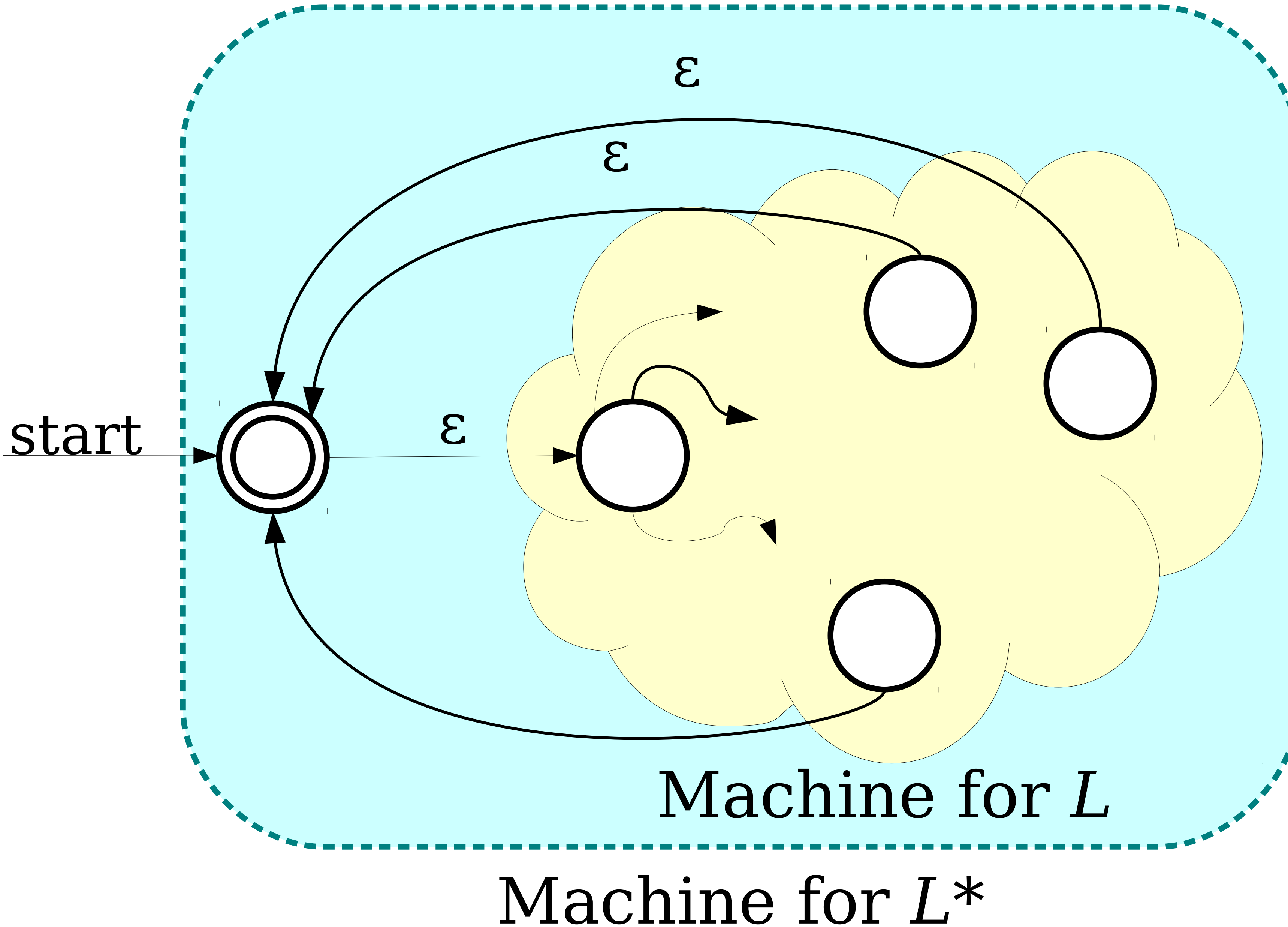
^ not

Reasoning About the Infinite

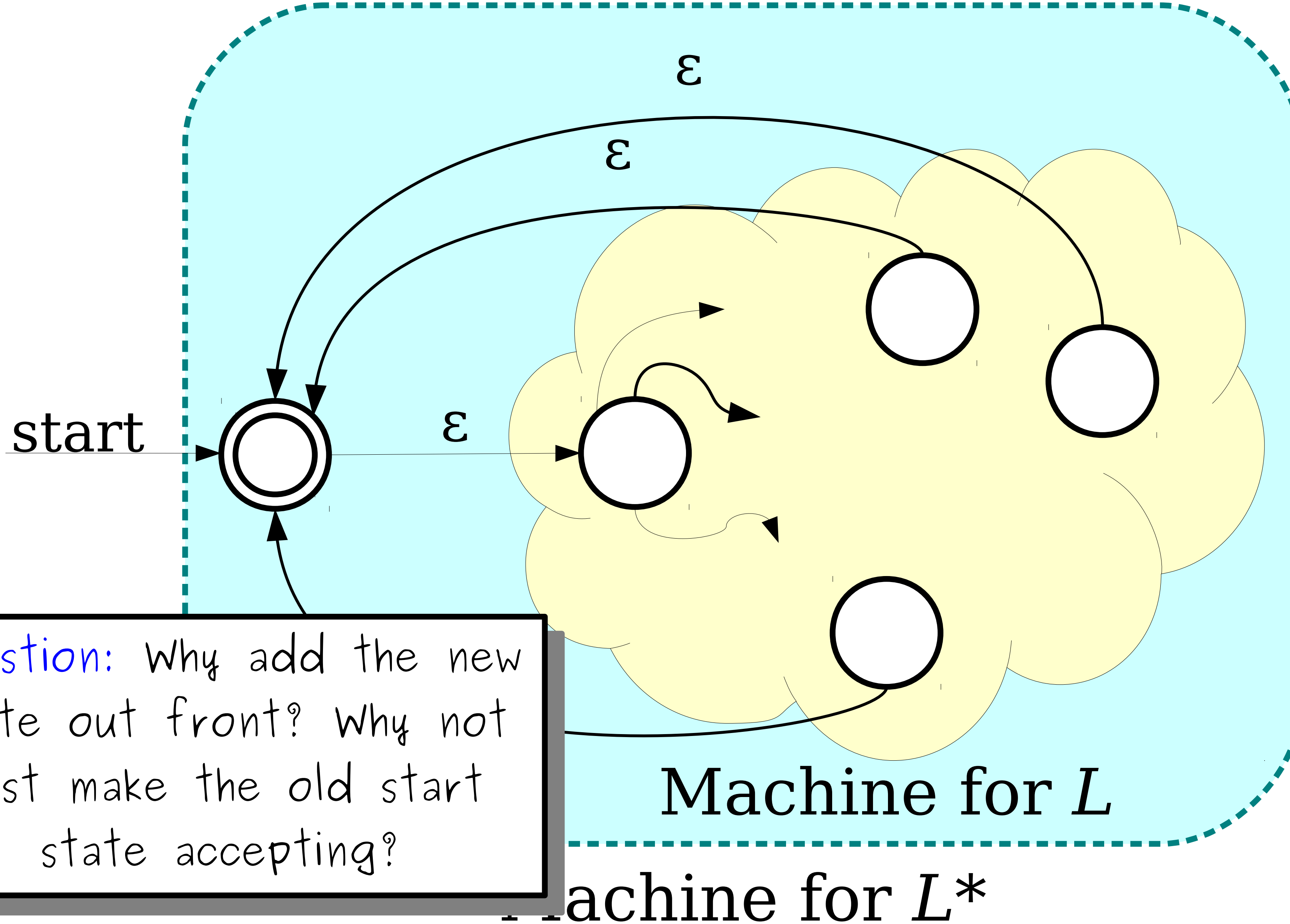
- If a series of finite objects all have some property, the “limit” of that process *does not* necessarily have that property.
- In general, it is not safe to conclude that some property that always holds in the finite case must hold in the infinite case.
  - (This is why calculus is interesting).

Idea: Can we directly convert an NFA for language L to an NFA for language L^*?

The Kleene Star



The Kleene Star



Question: Why add the new state out front? Why not just make the old start state accepting?

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, concatenation, and Kleene closure of regular languages are all regular languages.