Friday, September 2, 2022 2:05 PM

CSCE 355 9/13/2022

regex → (today) ε-NFA

ε-NFA → (next) regex

A regex over Σ is an expression built from

∅, a (a ∈ Σ) using

binary [+ (union), • (concatenation)

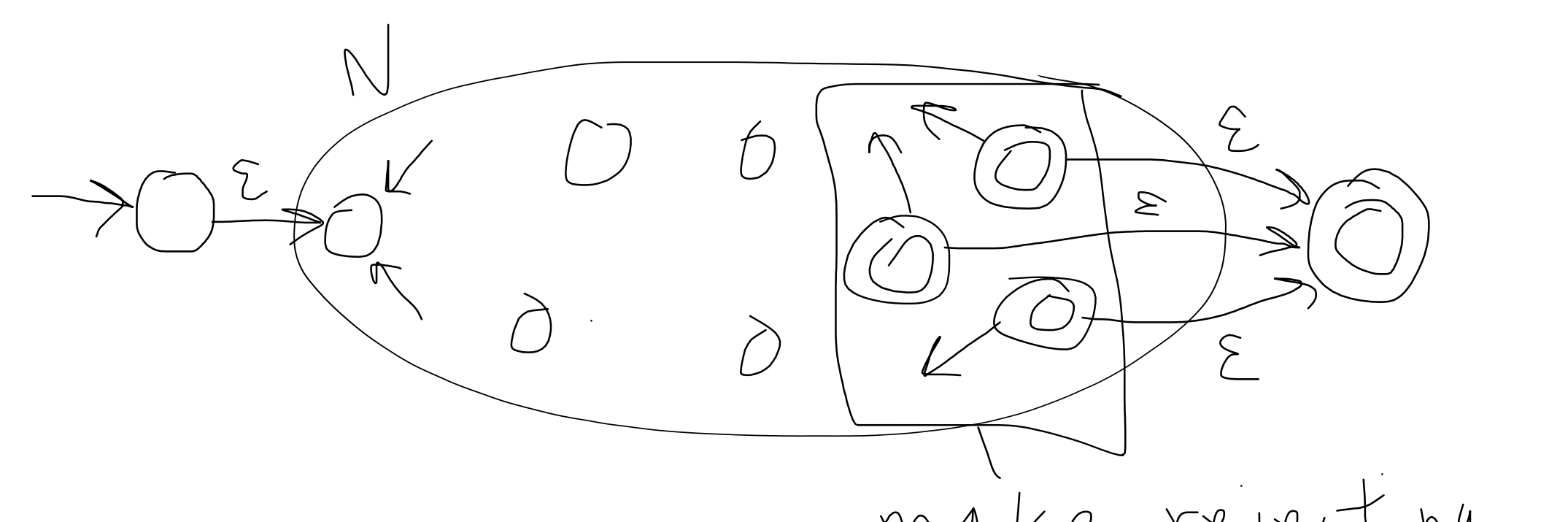
unary postfix [* (Kleene closure, Kleene * operator)

Goal today: Given a regex as input, construct an equivalent ε-NFA recursively on the syntax of the input regex.

<u>Def</u>: An ε-NFA is <u>clean</u> if
1) It has exactly one accept state
2) The accept state is not the start state
3) no transitions into the start state
4) no transitions out of the accept state

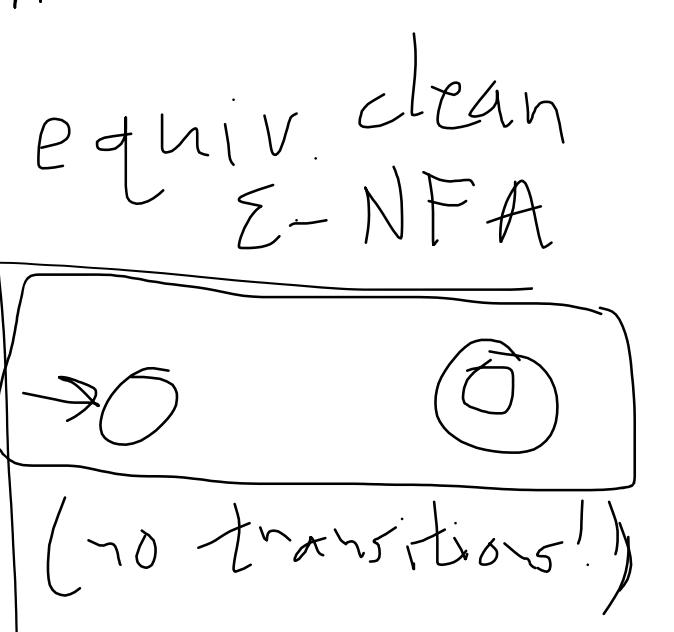
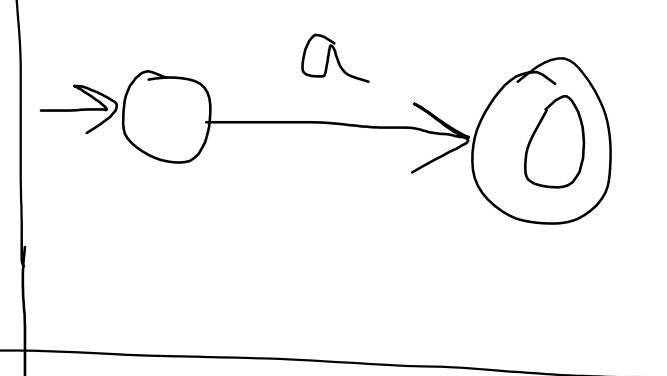
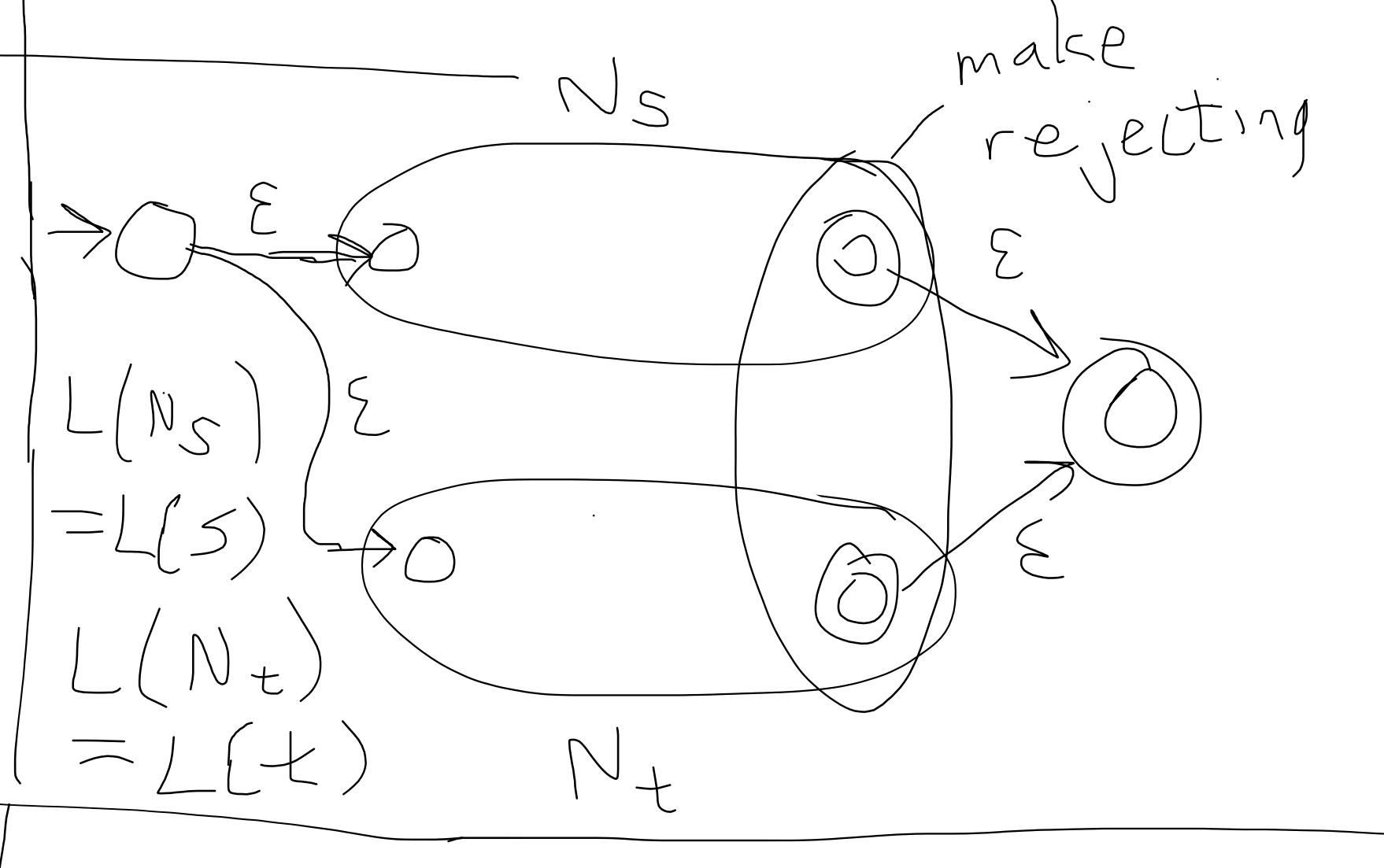
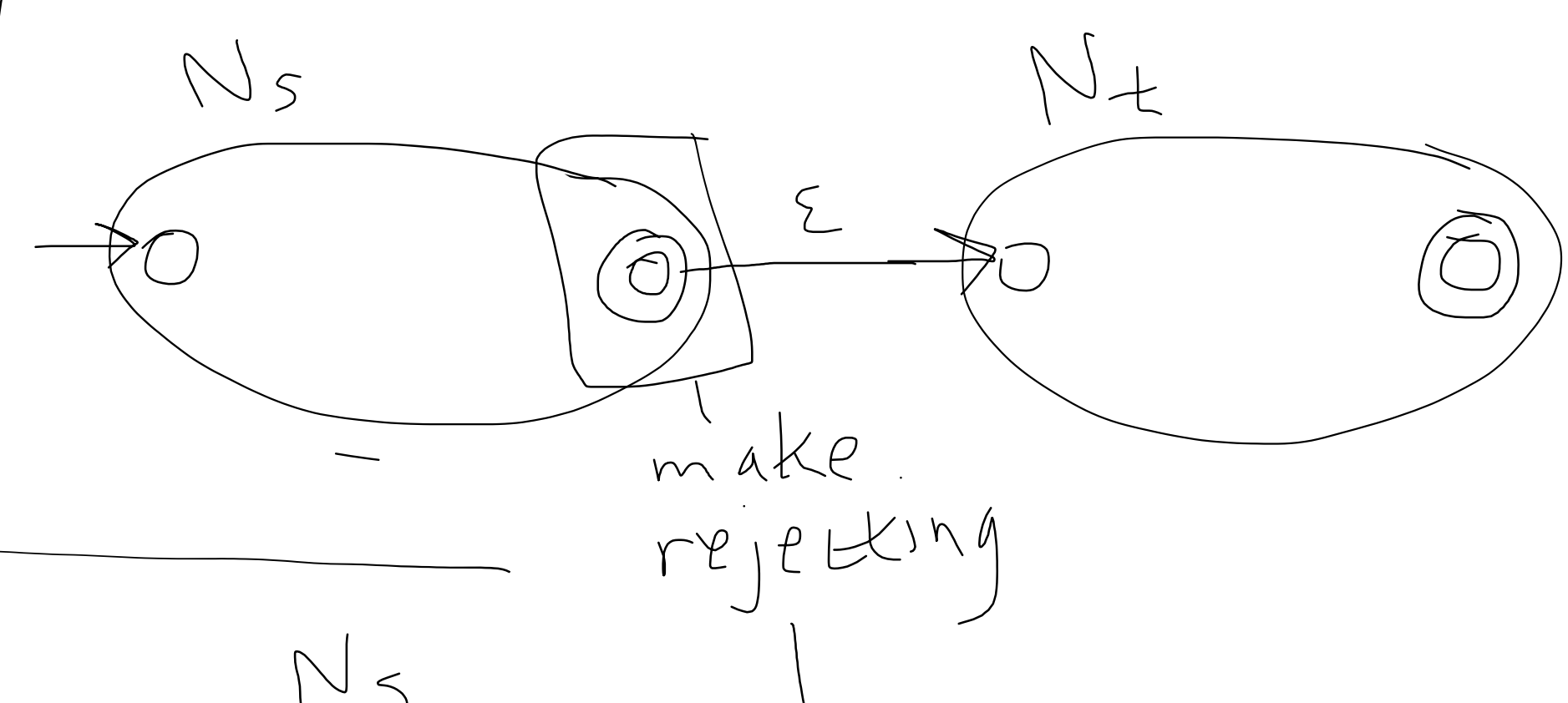
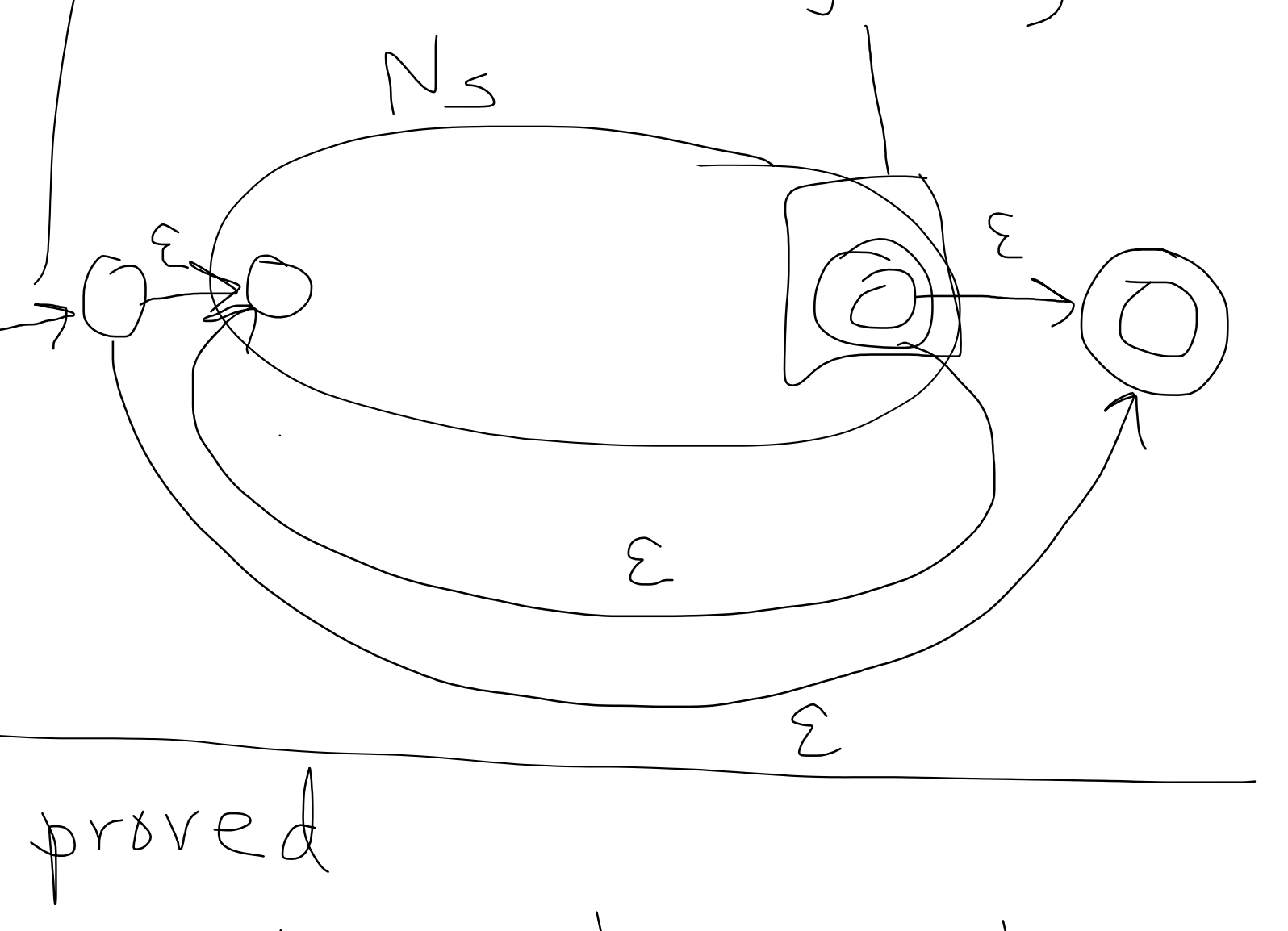
<u>Lemma</u>: For any ε-NFA, there exists an equivalent clean ε-NFA.

<u>Proof</u>: Let N be any ε-NFA.



This ε-NFA is clean and equiv to N ▨

Rules for converting any regex into an equivalent clean ε-NFA:

regex r	L(r)	equiv clean ε-NFA
∅	∅	 (no transitions!)
a (any a ∈ Σ)	{a}	
s + t (s, t regexes)	L(s) ∪ L(t)	
st	L(s)L(t)	
s*	L(s)*	

We've proved

<u>Thm</u>: For every regex, there exists an equivalent ε-NFA.

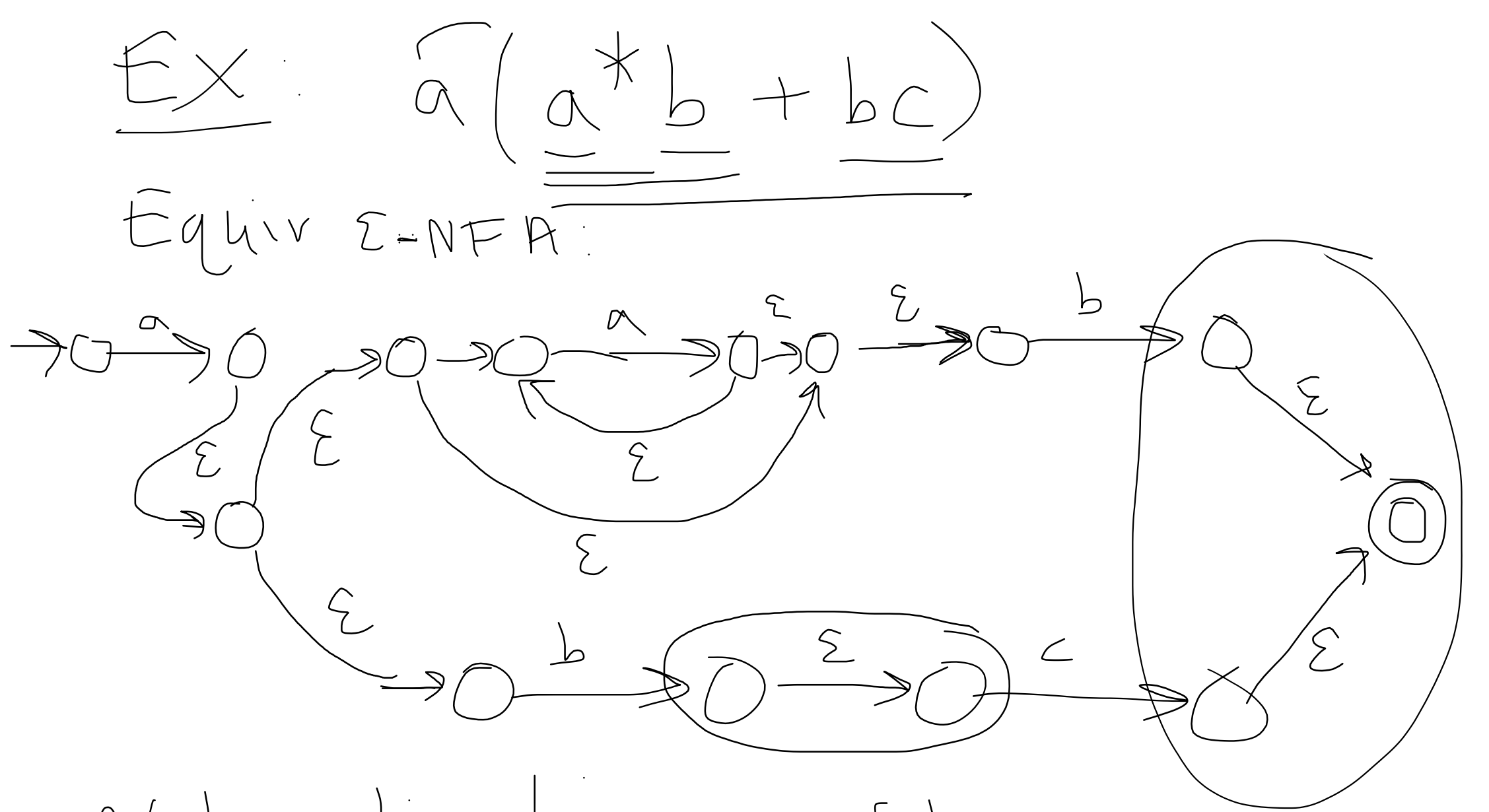
<u>Cor</u>: Every language denoted by a regex is regular.

<u>Pf</u>: regex r

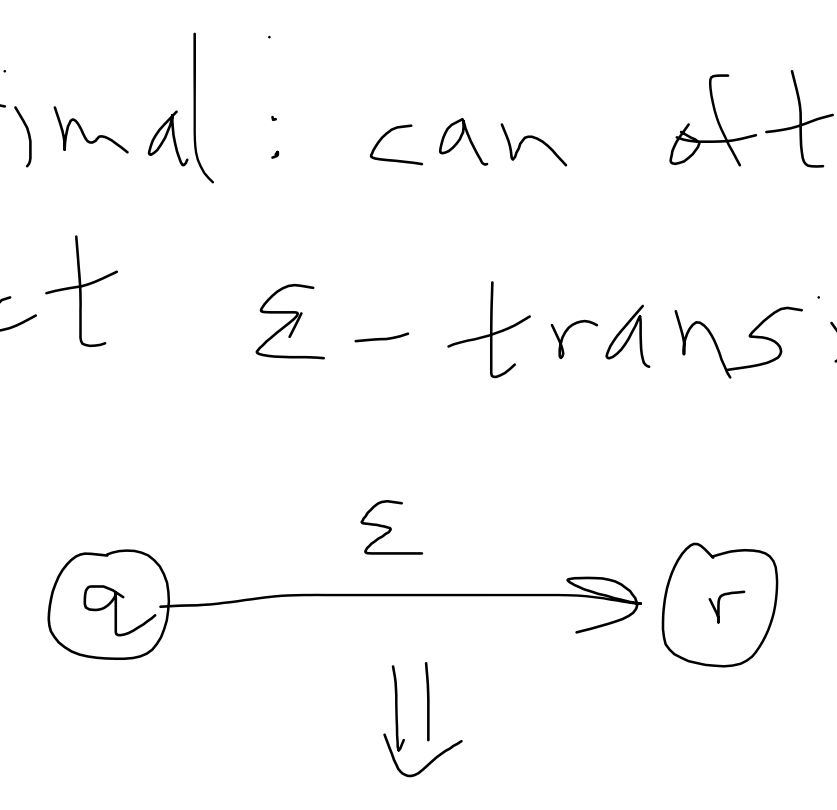
r → equiv ε-NFA (today) → equiv NFA (last week) → equiv DFA (set-of-states construction) ▨

<u>Ex</u>: a(a*b + bc)

Equiv ε-NFA:



Not optimal: can often contract ε-transitions



After doing all possible contractions: