Regular Operations II

Regular Operations

- Some important operations over regular languages:
  - Union:
    - $A \cup B = \{ w \mid w \in A \text{ or } w \in B \}$
  - Concatenation:
    - $AB = \{ wx \mid w \in A \text{ and } x \in B \}$
  - Kleene Star

$$A^* = \bigcup_{i=0}^{\infty} A^i = A^0 \cup A^1 \cup A^2 \cup A^3 \cup A^4 \ldots$$

Closure

- Regular Languages are closed under each of these operations:
  - Union
    - If A is regular and B is regular then $A \cup B$ is regular.
  - Concatenation
    - If A is regular and B is regular then AB is regular.
  - Kleene Star
    - If A is regular, then A^* is regular.

  - We already showed for union.
    - Constructive Proof
      - We'll do the other 2 as well as union again using NFAs

From the Previous Lecture

- For every DFA, there is an NFA that accepts the same language
- For every NFA, there is a DFA that accepts the same language

- DFAs, NFAs, are equivalent!

Concatenation

- The class of regular languages is closed under concatenation:
  - Need to show: If A_1 and A_2 are regular languages, then A_1A_2 is regular
  - Since A_1 and A_2 are regular, we know there are NFAs, N_1 and N_2 such that $A_1 = L(N_1)$ and $A_2 = L(N_2)$
  - We will build an NFA, N that will accept A_1A_2
  - Since NFAs and DFAs are equivalent:
    - If we build an NFA that accepts, there is a DFA that accepts
    - Thus the concatenation language is regular.

Finally – we put NFAs to use!

Concatenation

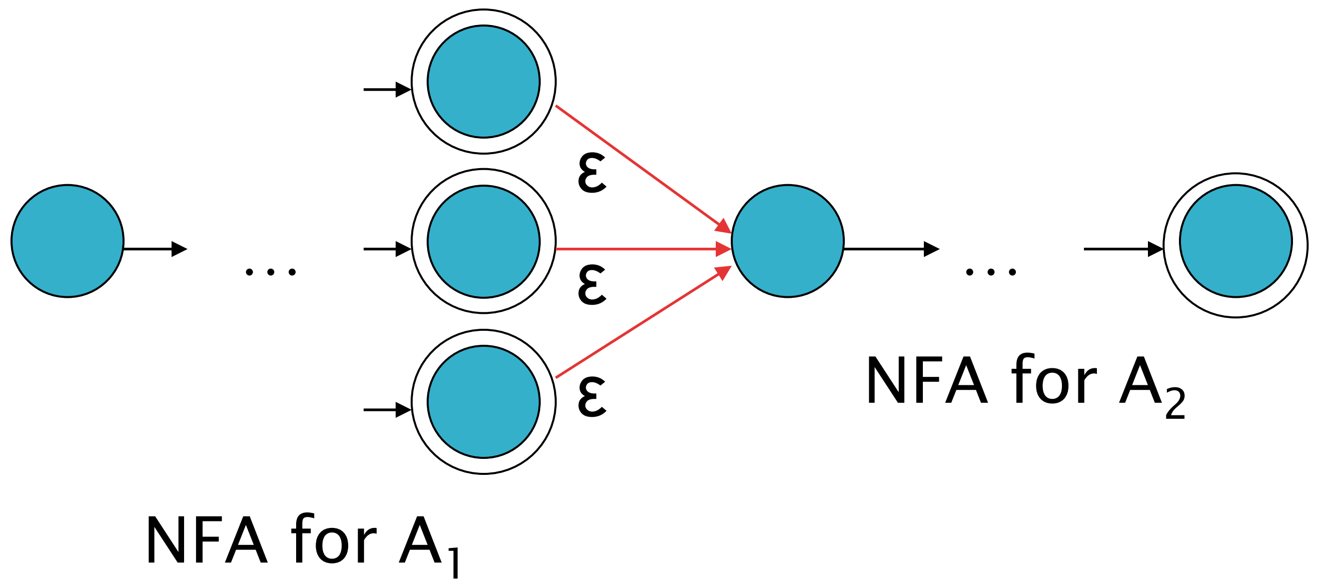
- Let
  - $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
  - $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

- We will build
  - $N = (Q, \Sigma, \delta, q_0, F)$

- Such that
  - $L(N) = L(N_1)\, L(N_2) = A_1 A_2$

Concatenation

- Basic idea
  - Build N to start at the start state of N_1 and from any accepting state of N_1 move directly to the start state of N_2 via an ε–transition.



What about the accepting states?

Concatenation

- Basic idea
  - Build N to start at the start state of N_1 and from any accepting state of N_1 move directly to the start state of N_2 via an ε-transition. (And stay where you are in the first machine!)

  - After a first portion of the string is accepted by the 1st machine, an extra branch is created to test the remainder of the string on the 2nd machine

    - The set of final states for the new machine consists of final states from the second machine

Concatenation

- Let's formalize this:
  - $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
  - $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

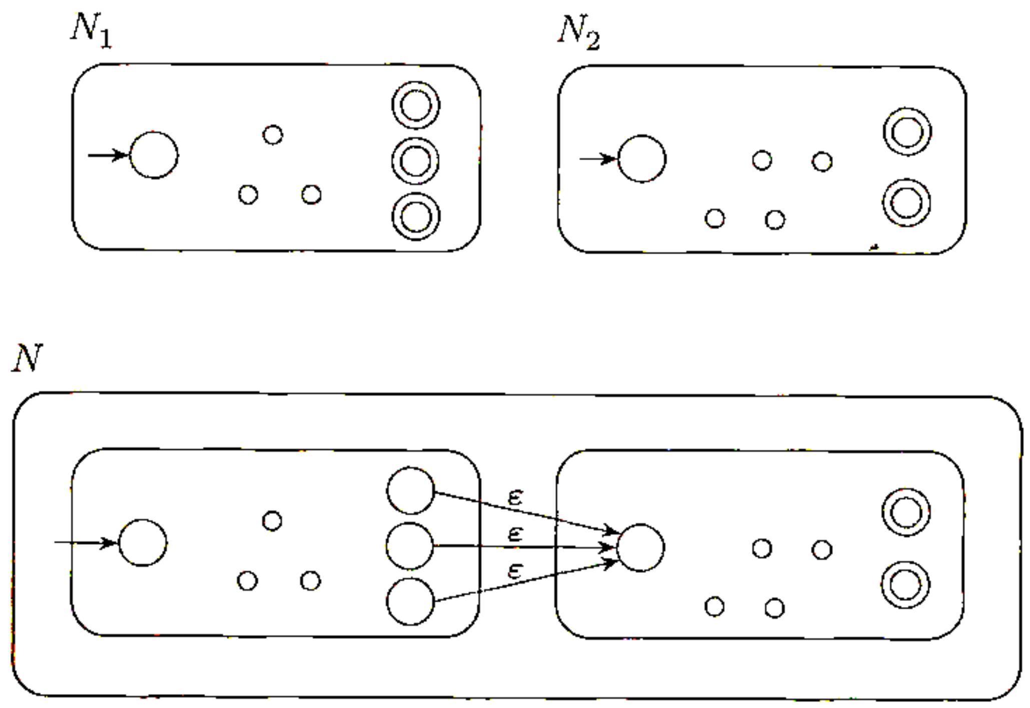
  - $N = (Q, \Sigma, \delta, q_0, F)$
    - $Q = Q_1 \cup Q_2$
    - $q_0 = q_1$
    - $F = F_2$

Concatenation

- Let’s formalize this:
  - Transition function δ :
    - $\delta(q, a) = \delta_1(q, a)$ for $q \in Q_1 - F_1$, $a \in \Sigma\varepsilon$
    - $\delta(q, a) = \delta_1(q, a)$ for $q \in F_1$ and $a \in \Sigma$
    - $\delta(q, \varepsilon) = \delta_1(q, \varepsilon) \cup \{q_2\}$ for $q \in F_1$ ★
    - $\delta(q, a) = \delta_2(q, a)$ for $q \in Q_2$, $a \in \Sigma\varepsilon$

If we are in an accept state for N_1 we define there to be an ε-transition to the start state of N_2 in addition to wherever N_1 ε-transitions anyway.

Concatenation



- Group the machines together
- Add ε–transitions from final states of N_1 to the start state of N_2
- Make the final states of N_2 be the only final states of N

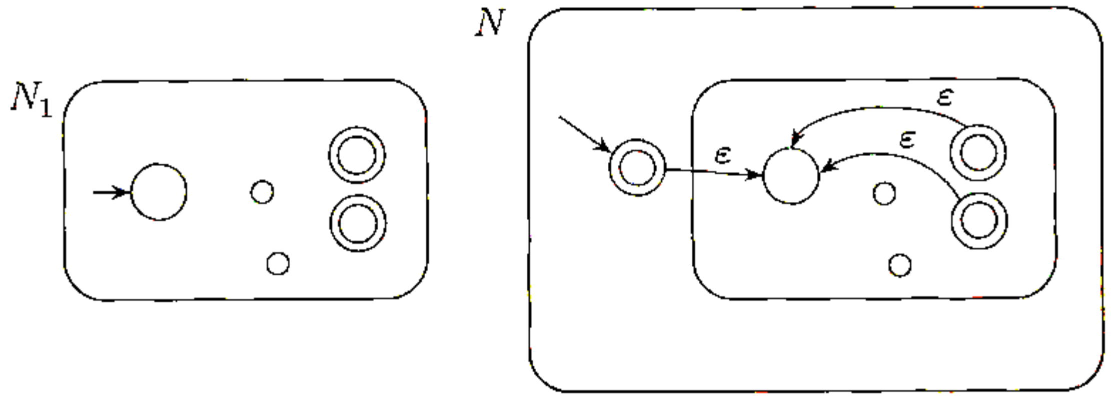
Kleene Star

- Let
  - $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
- We will build
  - $N = (Q, \Sigma, \delta, q_0, F)$
- Such that
  - $L(N) = (L(N_1))^*$

Kleene Star

- Basic idea
  - Create a new accepting start state
  - Go from new start state to original start state via an ϵ–transition
  - Create ϵ–transitions from all final states back to original start state (to allow for repetition)

Kleene Star



Why do we we need a new start state?
Because we need to accomplish two things
1.Make sure the empty string is accepted
2.ϵ−transition to the start after each final state is reached
But if we choose to make the original start state accepting, we might inadvertently allow other strings to be accepted that we don't want (if/when the machine naturally transitions back to its starting state)

Kleene Star

- Let's formalize this:
  - $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$

  - $N = (Q, \Sigma, \delta, q_0, F)$
    - $Q = Q_1 \cup \{q_{newstart}\}$
    - $q_0 = q_{newstart}$
    - $F = F_1 \cup \{q_{newstart}\}$

Kleene Star

- Let's formalize this:
  - Transition function δ :
    - $\delta(q, a) = \delta_1(q, a)$ for $q \in Q_1 - F_1,\ a \in \Sigma\varepsilon$
    - $\delta(q, a) = \delta_1(q, a)$ for $q \in F_1$ and $a \in \Sigma$
    - $\delta(q, \varepsilon) = \delta_1(q, \varepsilon) \cup \{q_1\}$ for $q \in F_1$
    - $\delta(q_{newstart}, \varepsilon) = \{q_1\}$
    - $\delta(q_{newstart}, a) = \varnothing$ for $a \in \Sigma$

These two just define the behavior for the new start state.

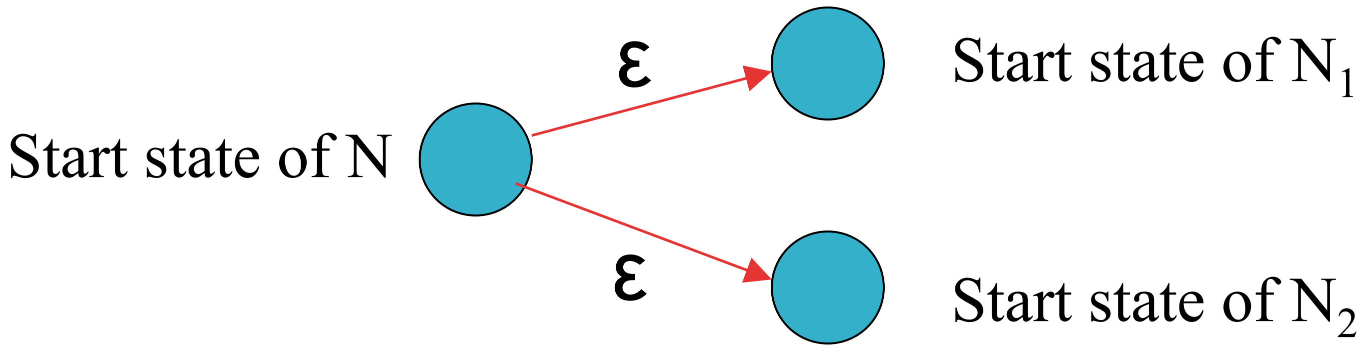
Just like for concatenation, we define there to be an ε-transition to the start state of N_1 in addition to wherever N_1 ε-transitions anyway.

Union – an Alternative Approach

- Let
  - $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
  - $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$
- We will build
  - $N = (Q, \Sigma, \delta, q_0, F)$
- Such that
  - $L(N) = L(N_1) \cup L(N_2)$

Union

- Basic idea
  - Using ε–transitions, create a "branch" where the machine can either follow one branch (representing N_1) or the other branch (representing N_2)



Union

- Let's formalize this:
  - $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
  - $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

  - $N = (Q, \Sigma, \delta, q_0, F)$
    - $Q = Q_1 \cup Q_2 \cup \{q_{newstart}\}$
    - $q_0 = q_{newstart}$
    - $F = F_1 \cup F_2$

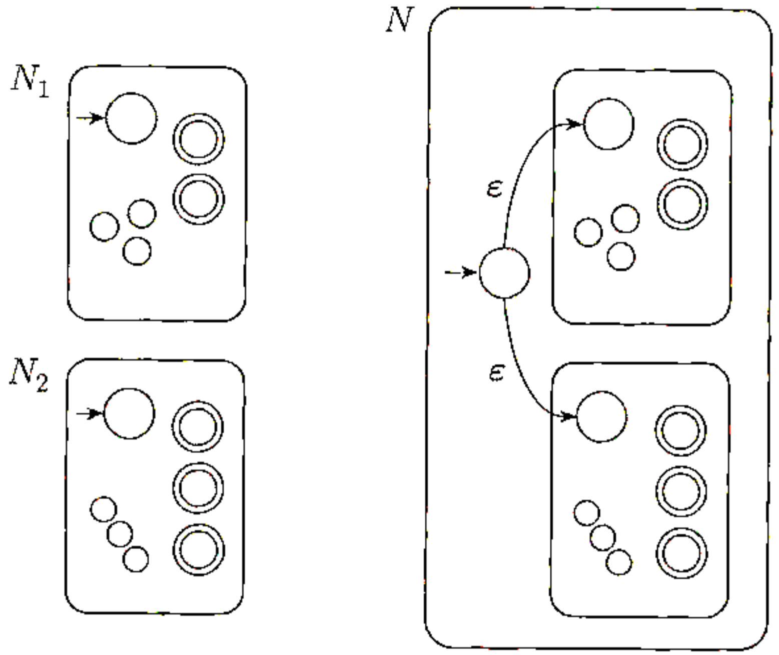
Union

- Let's formalize this:
  - Transition function δ :
    - $\delta(q, a) = \delta_1(q, a)$ for $q \in Q_1$, $a \in \Sigma\varepsilon$
    - $\delta(q, a) = \delta_2(q, a)$ for $q \in Q_2$, $a \in \Sigma\varepsilon$
    - $\delta(q_{newstart}, \varepsilon) = \{q_1, q_2\}$
    - $\delta(q_{newstart}, a) = \emptyset$ for $a \in \Sigma$

This is the only change. These two just define the behavior for the new start state.

The behavior of the states in the original machines stays the same.

Union

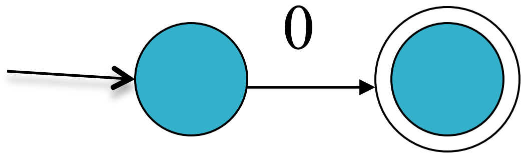


Example

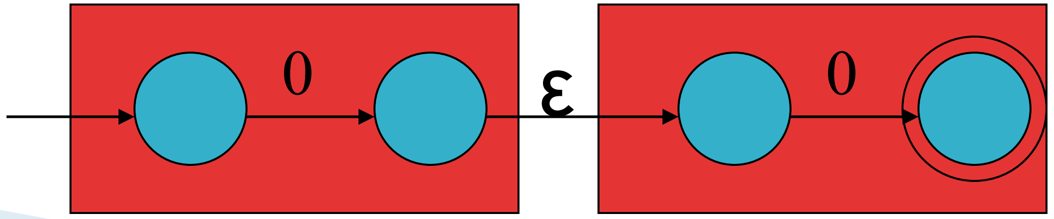
- Let’s try an example
  - Start with the languages {0} and {1}
  - Create an NFA for the language:
    - $(\{0\}\{0\} \cup \{1\})^*$

Example

- First, build the language {0}{0}
  - This is the language formed by concatenating the language {0} with itself
  - NFA for the language {0}

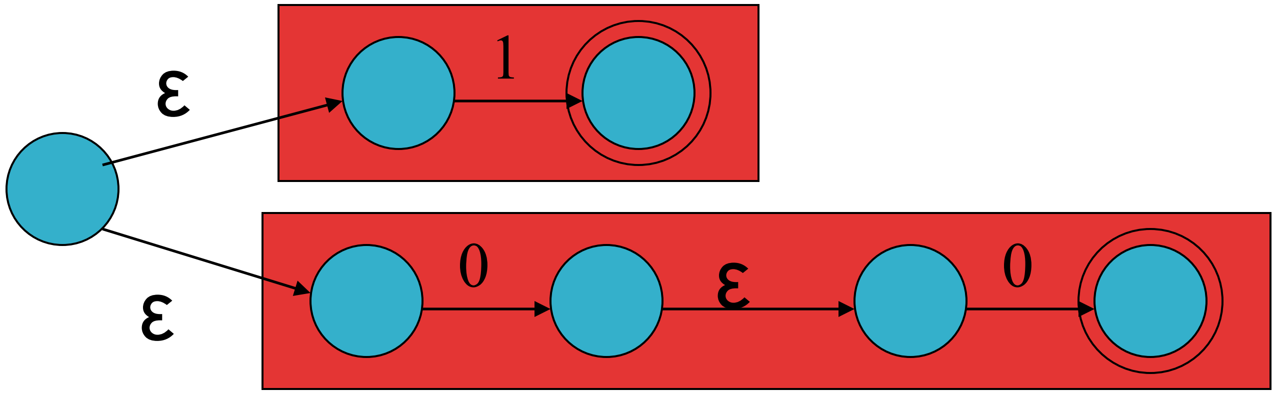


  - NFA for concatenation



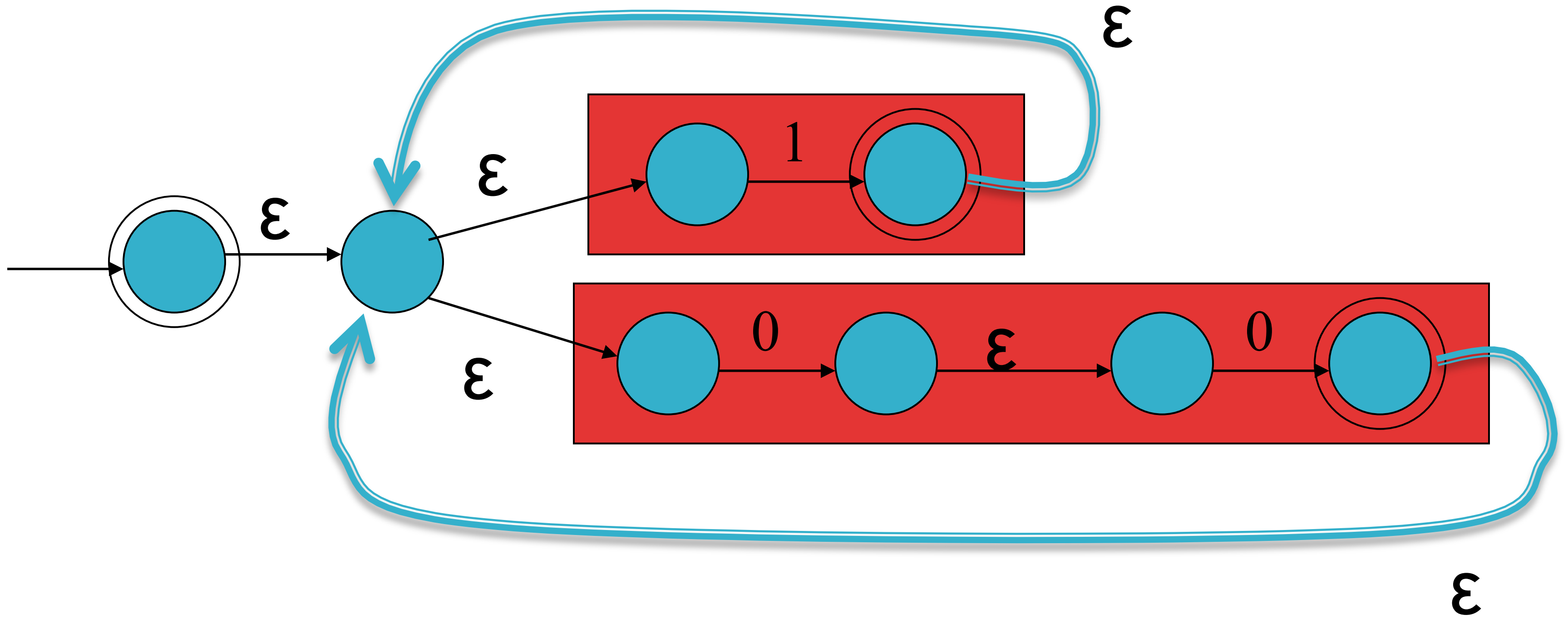
Example

- Next, take the union of {0}{0} and {1}
  - ({0}{0} ∪ {1})



Example

- Finally, take the Kleene star operation
  - ({0}{0} ∪ {1})*



Summary

- Constructive proof of the closure of regular languages under:
  - Union
  - Concatenation
  - Kleene Star