SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

FOUNDATIONS OF COMPUTER SCIENCE
LECTURE 2: Non-deterministic Automata and Closure Properties

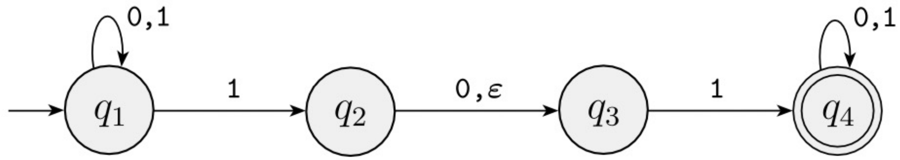
Prof. Daniele Gorla

Deterministic vs Non-deterministic Automata

- Automata defined and seen in the previous class are ***deterministic*** (in short DFA), in the sense that, when the machine is in a given state and reads the next input symbol, we exactly know what the next state will be.

- By contrast, in a non-deterministic automaton (in short NFA), several choices may exist for the next state at any point. (multiple possible evolutions, we don't know what to take a priori)

- Example:



  Here

  - State q_1 has one exiting arrow for 0, but it has two for 1;
  - q_2 has one arrow for 0, but it has none for 1;
  - q_2 is able to move without reading any input character (i.e., by reading ε)
    → zero, one, or many arrows labeled with ε may exit from each state.

- This is in sharp contrast with DFA, where every state always has exactly one exiting transition arrow for each symbol in the alphabet (and doesn't move with ε).

Computing with NFA

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- We run an NFA on an input string starting from the initial state (like for DFA)
- Let *q* be the current state and *a* the next input symbol

behavior of a NFA

  - For every *a*-transition, the machine splits into multiple copies of itself and follows *all* the possibilities in parallel. Each copy of the machine takes one of the possible ways to proceed and continues from the new state with the next input.
  - Furthermore, for every ε-transition, the machine splits into multiple copies without reading any input, one following each of the exiting ε-labeled arrows. Then the machine proceeds from the new state, still with input *a* (that has not been consumed in the transition).
  - If no *a*-transition nor any ε-transition exits from *q* in a copy of the machine, that copy dies, along with the branch of the computation associated with it.
- if *any one* of these copies of the machine is in an accept state at the end of the input, the NFA accepts the input string. trivially, if they ALL reject, the string is rejected

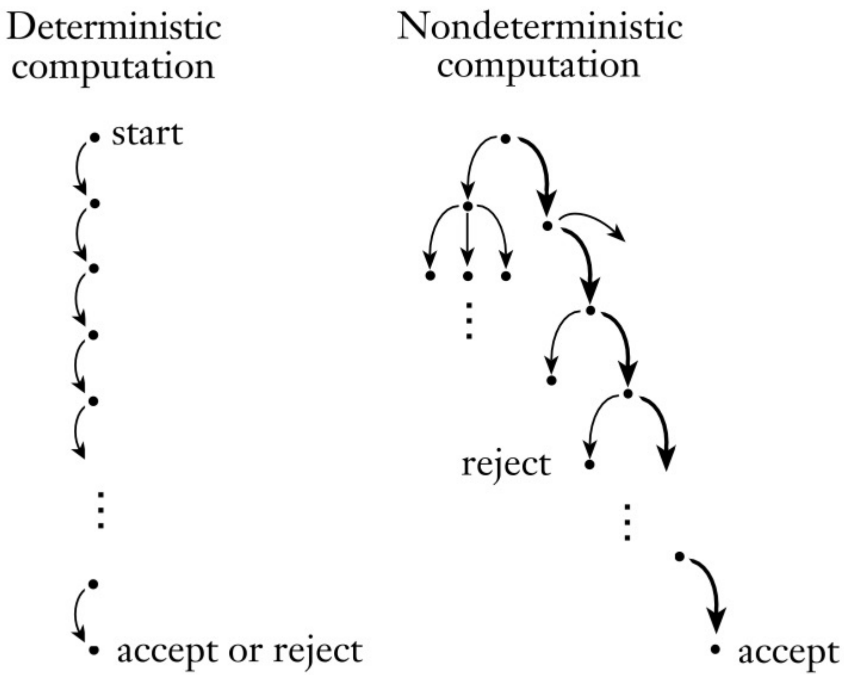
Hence, Non-determinism is a kind of parallel computation wherein multiple independent "threads" run concurrently. When the NFA splits to follow several choices, that corresponds to a process "forking" into several children, each proceeding separately.

If at least one of these processes accepts, then the entire computation accepts.

Computing with NFA (cont'd)

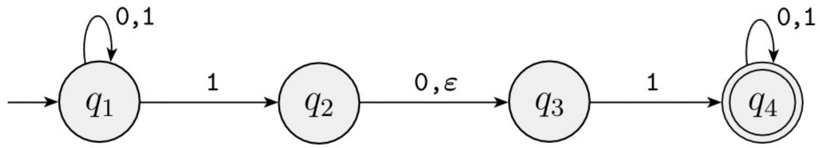
Another way to think of a non-deterministic computation is as a tree of possibilities:

- The root of the tree is the start of the computation
- Every branching point in the tree corresponds to a point in the computation at which the machine has multiple choices.
- The machine accepts if there exists at least one of the computation branches that ends in an accept state:

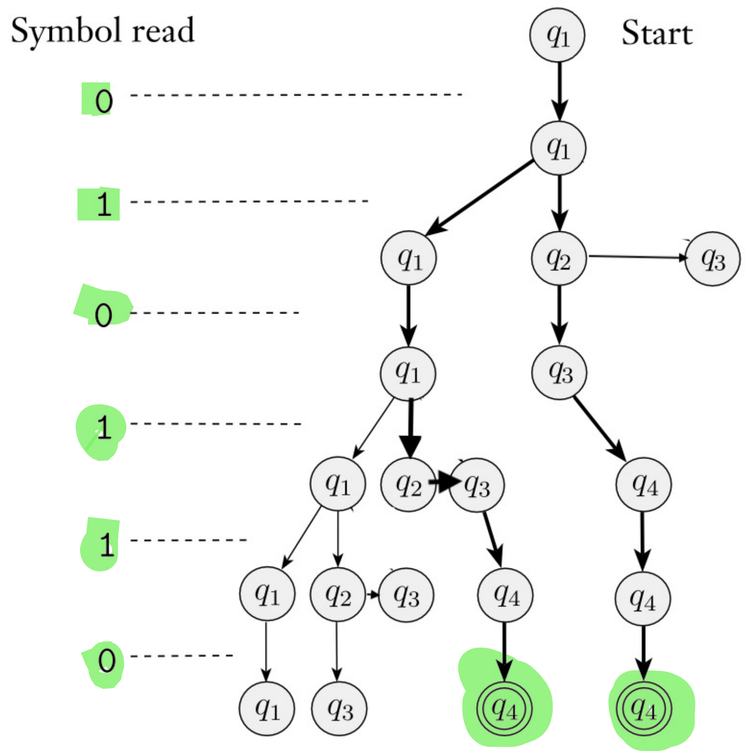


Example

Consider again the automaton:



The computation tree on input 010110 is:



Since *<u>at least one</u>* path in the tree terminates in an accepting state, the NFA accepts 010110

NFA: Formal definition

The formal definition of a NFA is similar to that of a DFA.

The crucial difference is in the transition function:

- In a DFA, the transition function takes a state and an input symbol and produces ***the*** next state;
- In an NFA, the transition function takes a state and an input symbol ***or the empty string*** and produces ***the set of possible next states***.

Notation:

- For any set Q, we write $\mathcal{P}(Q)$ to be the set of all subsets of Q ($\mathcal{P}(Q)$ is called the ***power set*** of Q).
- For any alphabet Σ, we write Σ_ε to denote $\Sigma \cup \{\varepsilon\}$.

DEFINITION

A ***nondeterministic finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\boxed{\delta \colon Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)}$ is the transition function, this is possibly empty
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

NFA: Acceptance

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and w a string over the alphabet Σ. Then we say that N ***accepts*** w if we can write w as $w = y_1 y_2 \cdots y_m$, where each y_i is a member of Σ_ε and a sequence of states $r_0, r_1, \ldots, r_m$ exists in Q with three conditions:

1. $r_0 = q_0$, state 0 is the initial state
2. $\boxed{r_{i+1} \in \delta(r_i, y_{i+1}),}$ for $i = 0, \ldots, m-1$, and
3. $r_m \in F$. final state BELONGS to accepting states

in next state i+1, it BELONGS to the transition fcn (set) associated to state i reading symbol i+1

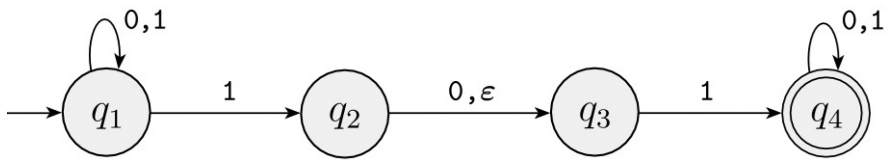
this was an EQUALITY in DFA

Notice that, in general, $m \geq |w|$ because some of the y_i's can be ε (actually, $m = |w|$ if and only if none of the y_i's is ε)

NFAs are usually less demanding than DFAs

Example

Let's again consider:



Its formal description is:

$$(Q, \Sigma, \delta, q_1, F)$$

1. $Q = \{q_1, q_2, q_3, q_4\}$,

2. $\Sigma = \{0,1\}$,

3. δ is given as

	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$,

4. q_1 is the start state, and

5. $F = \{q_4\}$.

Acceptance of 010110 can be obtained either

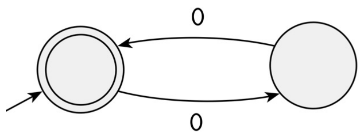
- by decomposning 010110 into 0-1-0-1-1-0 and having the path $q_1\, q_2\, q_3\, q_4\, q_4\, q_4$, or
- by decomposning 010110 into 0-1-0-1-ε-1-0 and having the path $q_1\, q_1\, q_1\, q_2\, q_3\, q_4\, q_4$

Another example

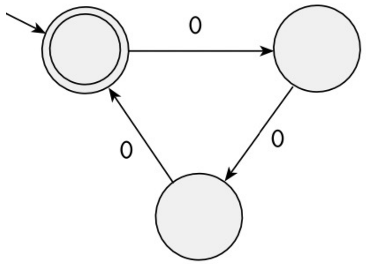
We want to build an automaton that recognizes $\{0^k : \ k \text{ is multiple of 2 or of 3}\}$

Idea:

It is easy to recognize 0^k, for k multiple of 2:



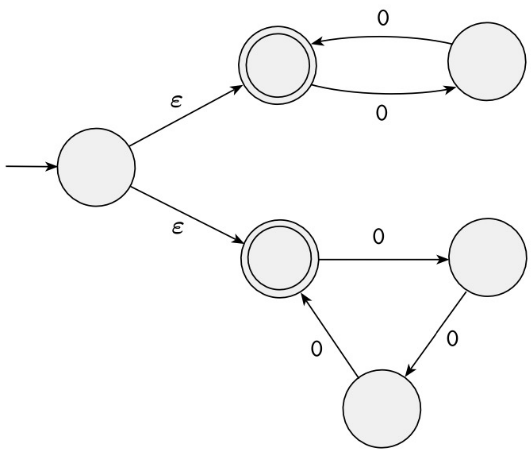
Similarly, it is easy to recognize 0^k, for k multiple of 3:



Thus, by adding a new state and two ε-transitions, we can recognize the required language:

In particular,

- To accept 00 or 0000, the upper path must be taken
- To accept 000, the lower path must be taken
- To accept 000000, either path can be taken
- No path will lead to accept 0 or 00000



REM: epsilon has 0 zeros (even number of zeros)

We need AT LEAST ONE acceptance (see computation tree)

REM: DFA is a special case of NFA

Equivalence of NFA and DFA (1)

Thm.: For every NFA N there exists a DFA M such that $L(N) = L(M)$ (the language of N is accepted by M)

PROOF Let $N = (Q, \Sigma, \delta, q_0, F)$ be the NFA recognizing some language A. We construct a DFA $M = (Q', \Sigma, \delta', q_0', F')$ recognizing A.

let's first consider the easier case wherein N has no ε arrows.

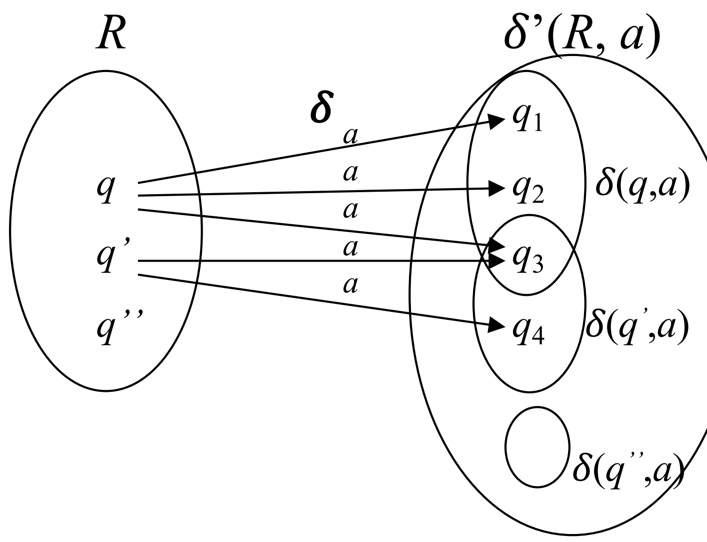
most trivial case as DFAs don't accept epsilon transitions

Q' = all the possible subsets of N states

$$Q' = \mathcal{P}(Q) \qquad q_0' = \{q_0\} \qquad F' = \{R \in Q' \mid R \text{ contains an accept state of } N\}$$

For $R \in Q'$ and $a \in \Sigma$, let $\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$

Visually:



one state of the DFA contains one or more states of the NFA

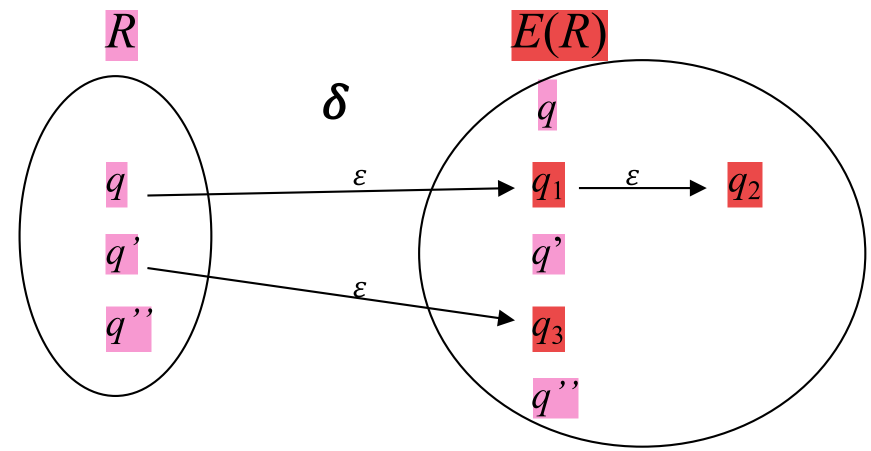
Here, R (DFA) contains three NFA states

Equivalence of NFA and DFA (2)

If N has ε-transitions, let $E(R)$ denote the ε-closure of $R \subseteq Q$:

$$E(R) = \{q \mid q \text{ can be reached from } R \text{ by traveling along 0 or more } \varepsilon \text{ arrows}\}$$

Visually:



R in pink

in red states reachable with one or more epsilon transitions

Then, the construction of the previous slide is modified as follows:

this is because you don't arrive only to one state, but to all the possible ones reached by epsilon

$$q_0' = E(\{q_0\}) \qquad \delta'(R,a) = \bigcup_{r \in R} E(\delta(r,a))$$

Q.E.D.

Cor.: L is regular if and only if there exists a NFA N such that $L = L(N)$

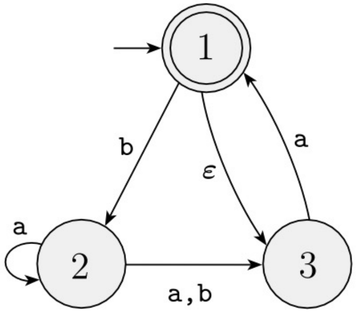
we can provide both a DFA and a NFA for proving if L is regular

Remark: M can have a numb. of states that is 2^n, where n is the numb. of states of N!!

(M is DFA)

Equivalence of NFA and DFA: An example

Consider the NFA:

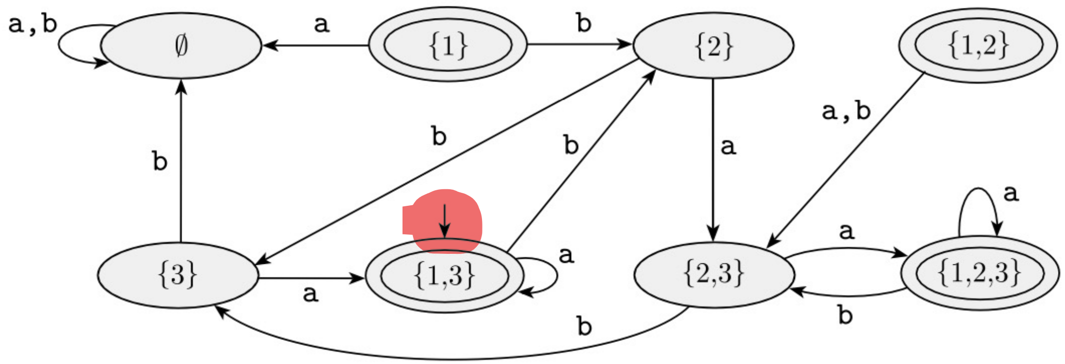


Its deterministic counterpart has as states the set $\{\emptyset, \{1\}, \{2\}, \{3\}, \{1{,}2\}, \{1{,}3\}, \{2{,}3\}, \{1{,}2{,}3\}\}$

Its starting state is $E(\{1\}) = \{1, 3\}$

whereas its final states are $\{\{1\}, \{1{,}2\}, \{1{,}3\}, \{1{,}2{,}3\}\}$ 1 is the only final state in NFA, in DFA EVERY set that contains 1 is final

Finally, the transition relation is the following:



Notice that $\{1\}$ and $\{1{,}2\}$ are unreachable, and so the DFA can be simplified (we're not interested in this now)

Regular operations on languages

DEFINITION

Let A and B be languages. We define the regular operations ***union***, ***concatenation***, and ***star*** as follows:

- **Union**: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.
- **Concatenation**: $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$.
- **Star**: $A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$.

EXAMPLE

Let the alphabet Σ be the standard 26 letters $\{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$. If $A = \{\mathtt{good}, \mathtt{bad}\}$ and $B = \{\mathtt{boy}, \mathtt{girl}\}$, then

$A \cup B = \{\mathtt{good}, \mathtt{bad}, \mathtt{boy}, \mathtt{girl}\}$,

$A \circ B = \{\mathtt{goodboy}, \mathtt{goodgirl}, \mathtt{badboy}, \mathtt{badgirl}\}$, and

$$A^* = \{\varepsilon, \mathtt{good}, \mathtt{bad}, \mathtt{goodgood}, \mathtt{goodbad}, \mathtt{badgood}, \mathtt{badbad}, \mathtt{goodgoodgood}, \mathtt{goodgoodbad}, \mathtt{goodbadgood}, \mathtt{goodbadbad}, \ldots\}.$$

This contains infinitely many elements, as you concatenate A (or B) with itself as many times as you want

Closure properties for regular languages

SAPIENZA
Università di Roma
Dipartimento di Informatica

A collection of objects is ***closed*** under some operation if applying that operation to members of the collection returns an object still in the collection.

<u>EXAMPLE:</u>

- Let **N** = {0,1,2,3,...} be the set of natural numbers; then, **N** is *closed under sum and multiplication*, i.e., for any x and y in **N**, $x+y$ and xy are also in **N**.
- By contrast, **N** is not closed under division, as 1 and 2 are in **N** but 1/2 is not.

We show that the collection of regular languages is closed under the three regular operations.

Because of what we've just proved, given two automata (deterministic or not) for A and B, it suffices to show that there exists NFA able to recognize $A \cup B$, $A \circ B$, and A^*

Closure under union

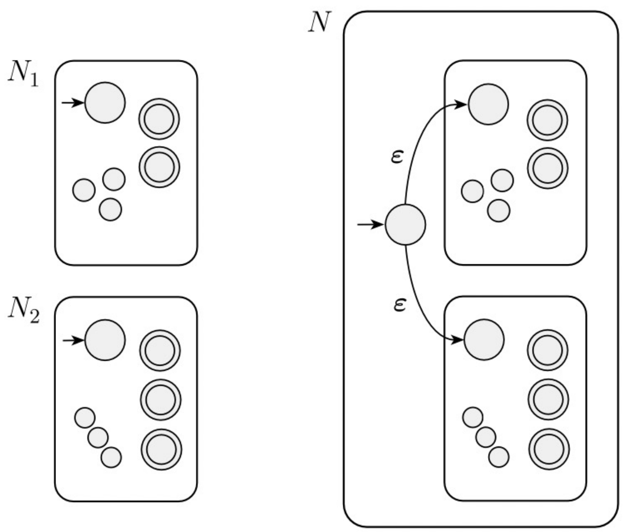
Thm.: if A_1 and A_2 are regular, then $A_1 \cup A_2$ is regular too.

PROOF

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$.

Intuitively:



Formally:

$Q = \{q_0\} \cup Q_1 \cup Q_2$.

The state q_0 is the start state of N.

The set of accept states $F = F_1 \cup F_2$.

Define δ so that for any $q \in Q$ and any $a \in \Sigma_\varepsilon$,

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \\ \delta_2(q,a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon \end{cases}$$

Q.E.D.

Closure under concatenation

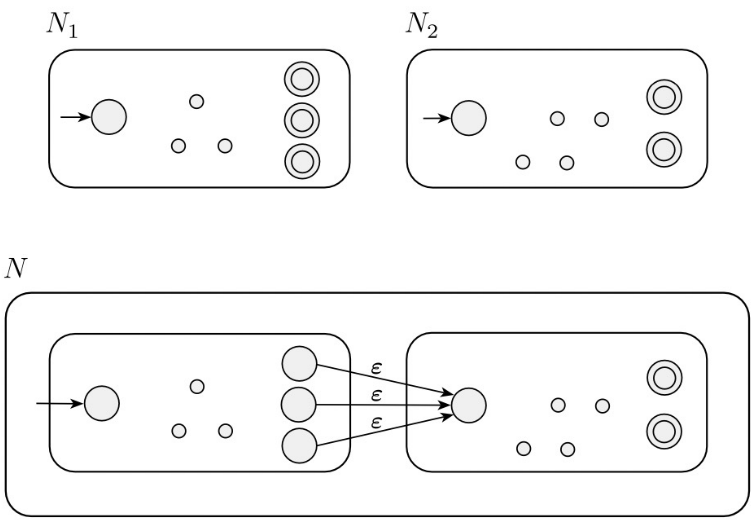
Thm.: if A_1 and A_2 are regular, then $A_1 \circ A_2$ is regular too.

PROOF

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.

Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$.

Intuitively:



Formally:

1. $Q = Q_1 \cup Q_2$.

2. The state q_1 is the same as the start state of N_1.

3. The accept states F_2 are the same as the accept states of N_2.

4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\varepsilon$,

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q,a) & q \in Q_2. \end{cases}$$

i can do wtv i could do before with other epsilon
transition PLUS get to the other automaton

Q.E.D.

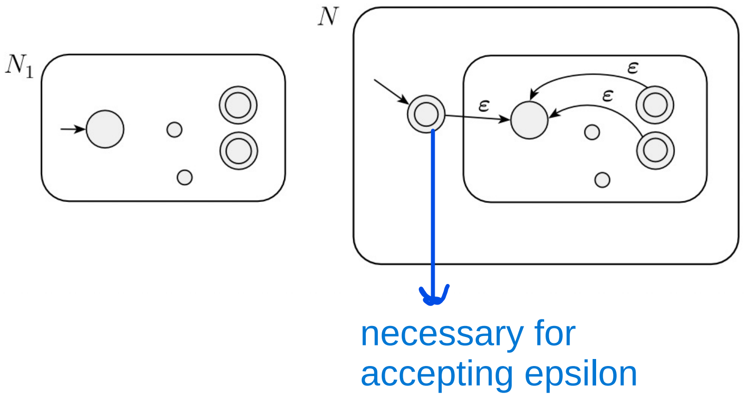
Closure under star

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Thm.: if A_1 is regular, then A_1* is regular too.

PROOF Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1.
Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize A_1^*.

Intuitively:



Formally:

1. $Q = \{q_0\} \cup Q_1$.

2. The state q_0 is the new start state.

3. $F = \{q_0\} \cup F_1$.

4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\varepsilon$,

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \text{ and } a = \varepsilon \\ \{q_1\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon. \end{cases}$$

Q.E.D.

Other Closure Properties
(complement, intersection, difference)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Thm.: if A_1 and A_2 are regular, then $\overline{A_1}$, $A_1 \cap A_2$ and $A_1 \setminus A_2$ are regular.

Proof:

Let $M = (Q,\Sigma,\delta,q_0,F)$ be a DFA such that $L(M) = A_1$.

Then, $M' = (Q,\Sigma,\delta,q_0,Q \setminus F)$ is a DFA such that $L(M') = \overline{A_1}$.   !A is whatever is not in A, so it doesn't accept A

By De Morgan's Law, $A_1 \cap A_2 = \overline{\overline{A_1} \cup \overline{A_2}}$.

Since regular languages are closed by union and complement, they are also closed by intersection.

By basic set theory, we have that $A_1 \setminus A_2 = A_1 \cap \overline{A_2}$.

Since regular languages are closed by intersection and complement, they are also closed by difference.

If you compose closed operators you still get a closed operation

<u>Q.E.D.</u>

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Closure under reversal

Recall that the reversal of a string w is the string w^R, where

$$w^R = a_n \ldots a_1 \qquad \text{whenever} \quad w = a_1 \ldots a_n$$

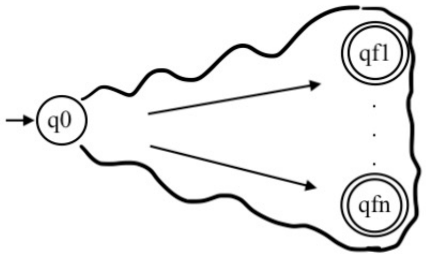
Clearly, $\varepsilon^R = \varepsilon$ (obatained from above whenever $n = 0$).

So, given a language L, we define $L^R = \{w^R \mid w \in L\}$.

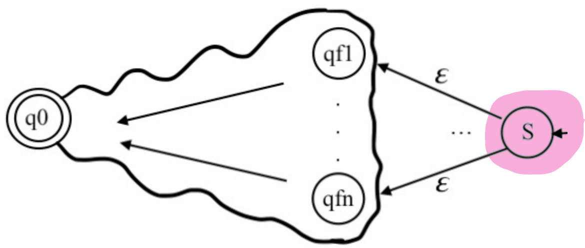
Thm.: If L is regular, also L^R is regular.

Proof

Consider a NFA for L:



From it, we can derive a NFA (with ε moves) for L^R:



where S is a new state.

basically reverse the automaton

Q.E.D.

EXERCISE: Formalize this intuitive proof.