cs2001
Introductio Theory of Computation

cture 10

Ruth Urner

Febuary 8, 2023

Non-deterministic finite automata

Reading: ITC Section 1.2

To show that regular languages are also closed under the concatenation and star operations, it is convenient to employ the concept of a Non-deterministic Finite Automaton (NFA for short).

In the automata we have seen so far, every state has exactly one outgoing edge for every symbol. That is, for every state, the "reaction" to any input symbol is uniquely determined. Such an automaton is also called a Deterministic Finite Automaton (DFA for short).

We will generalize this to also allow non-unique (think randomized or parallelized) computations. NFAs are a model for such computations.

Example: the NFA N_1

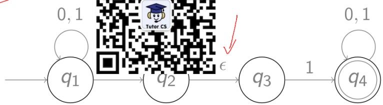
For the machine N_1 we get

$$L(N_1) = \{\mathbf{w} \mid \mathbf{w} \text{ contains 11 or 101 as a substring}\}$$

Example: the NFA N_1

Differences

ϵ can be "read"

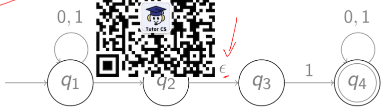


some states have several instructions for the same letter

some states don't have instructions for all letters

Example: the NFA N_1

Differences



Examples of words

10110 accepted

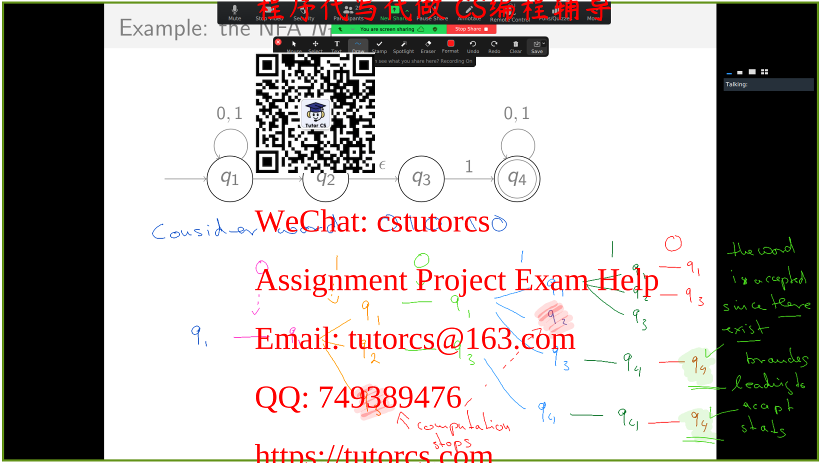
1000 X not accepted

some states have several instructions for the same letter

some states have no instructions for all letters

A word is accepted by an NFA if there is at least one reading of the word that leads to an accept state.

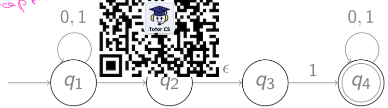
Example: the NFA N_1
Differences
ε can be "read"
0, 1
0, 1
q_1
q_2
q_3
q_4
1
ε
Examples of words
10110 accepted
1000 X not accepted
same states have several instructions for the same letter
some states have no instructions for all letters
A word is accepted by an NFA if there is at least one reading of the word that leads to an accept state.
Talking: Asish Dan Philips .



Example: the NFA N
0, 1
0, 1
q_1
q_2
q_3
q_4
ϵ
1
Consider word
computation stops
the word is accepted since there exist branches leading to accept states

Example: the NFA N_1

10001
not accepted



Reminder: For alphabet Σ, the set of all words over Σ is denoted by Σ^*.

For the machine N_1 we get

$L(N_1) = \{ w \in \{0,1\}^* \mid w$ contains 101 or 11 as a substring$\}$

For an alphabet Σ, we define $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$.

Definition

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

1. Q is a finite set called the set of states,
2. Σ is a finite set called the alphabet,
3. $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

Formal description of the NFA N_1

N_1

Formal description of $N_1 = (Q, \Sigma, \delta, q_0, F)$

$Q = \{q_1, q_2, q_3, q_4\}$, $q_0 = q_1$, $F = \{q_4\}$.

$\Sigma_\varepsilon = \{0, 1, \varepsilon\}$

δ	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	$\{q_4\}$	$\{q_4\}$	$\emptyset$

$1\varepsilon1$

$\varepsilon 1 \varepsilon 1 \varepsilon\varepsilon$

$1\varepsilon\varepsilon\varepsilon 1 \varepsilon$

these are the same word !!

Formal description of the NFA N_1

For the machine N_1 we get

$$L(N_1) = \{\mathbf{w} \mid \mathbf{w} \text{ contains 11 or 101 as a substring}\}$$

For a string $\mathbf{w} = w_1 w_2 \ldots w_n$ where $w_i \in \Sigma$ for some alphabet Σ, we let strings $\mathbf{z}_0 w_1 \mathbf{z}_1 w_2 \mathbf{z}_3 \ldots \mathbf{z}_{n-1} w_n \mathbf{z}_n$ [illegible]ere each $\mathbf{z}_i$ is a sequence of 0 or more symbols ϵ represent the [illegible] $\mathbf{w}$ as a string over Σ.

Definition

Let $N = (Q, \Sigma, \delta, q_0, F)$ a non-deterministic finite automaton (NFA) and let $\mathbf{w}$ be a string over Σ. We say that N accepts $\mathbf{w}$ if we can write $\mathbf{w} = y_1 y_2 \ldots y_m$ with $y_i \in \Sigma_\epsilon$ and there exists a sequence $s_0 s_1 s_2 \ldots s_m$ of states such that

1. $s_0 = q_0$,
2. $s_{i+1} \in \delta(s_i, y_{i+1})$ for $i = 0, 1, \ldots, m$, and
3. $s_m \in F$.

We then define the language $L(N)$ recognized by NFA N in the same way as for DFAs, namely as the set of all words that are accepted by N.

Formal definition of acceptance for NFAs

$|| = w$

$|\varepsilon|\varepsilon$

$|\varepsilon\varepsilon\,\varepsilon\varepsilon|$

$\varepsilon|\varepsilon|$

all the same word w!!

For a string $\mathbf{w} =$ … $w_i \in \Sigma$ for some alphabet Σ, we let strings $\mathbf{z}_0 w_1 \mathbf{z}_1 w_2 \mathbf{z}_3 \ldots \mathbf{z}$… where each $\mathbf{z}_i$ is a sequence of 0 or more symbols ϵ repres… $\mathbf{w}$ as a string over Σ.

Definition

Let $N = (Q, \Sigma, \delta, q_0, F)$ a non-deterministic finite automaton (NFA) and let $\mathbf{w}$ be a string over Σ. We say that N accepts $\mathbf{w}$ if we can write $\mathbf{w} = y_1 y_2 \ldots y_m$ with $y_i \in \Sigma_\epsilon$ and there exists a sequence $s_0 s_1 s_2 \ldots s_m$ of states such that

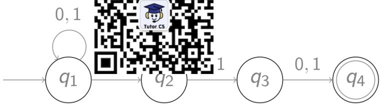
1. $s_0 = q_0$,
2. $s_{i+1} \in \delta(s_i, y_{i+1})$ for $i = 0, 1, \ldots, m$, and
3. $s_m \in F$.

Example: the NFA N_2

For the machine N_2 we get

$$L(N_2) = \{\mathbf{w} \mid \mathbf{w} \text{ contains 1 at the third position from the end}\}$$

Example: the NFA N_2



0101 ✓ 0001 ✗

1111 ✓ 00001 ✗

For the machine N_2 we get

$L(N_2) =$ { w | w has a 1 at the third position from the end }

$= \{ w = w_1 w_2 \dots w_n \in \Sigma^* \mid w_{n-2} = 1 \}$

Talking: Ruth Urner

Example: the NFA N_3

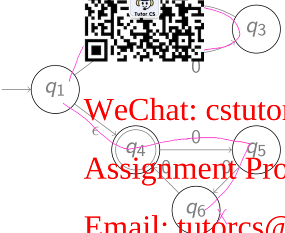
Let's consider a unary alphabet $\Sigma = \{0\}$.

For the machine N_3 we get

$$L(N_3) = \{\mathbf{w} \mid \text{The number of 0 symbols in } \mathbf{w} \text{ is divisible by 2 or by 3}\}$$

Example: the NFA N_4

Let's consid… abet $\Sigma = \{0\}$.



ε ✓

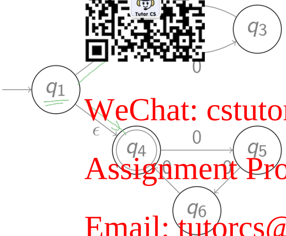
00000 ✗

00 ✓

$L(N_4) = \{ w \in \{0\}^* \mid$ the length of w is divisible by 2 or by 3 $\}$

Example:

Let's consid… abet $\Sigma = \{0\}$.



ε ✓

00000 ✗

00 ✓

is the same word as ε00

$L(N_4) = \{ w \in \{0\}^* \mid$ the length of w is divisible by 2 or by 3 $\}$

$= \{ w \in \{0\}^* \mid$ length of w even $\} \cup \{ w \in \{0\}^* \mid$ length of w divisible by 3 $\}$

Equivalence of NFAs and DFAs

Definition

We call two finite state machines equivalent if they recognize the same language.

Theorem

For every NFA, there exists an equivalent DFA.

Corrolary

A language L is regular if and only if there exists an NFA that recognizes it.

Equival

Definition

We call two machines **equivalent** if they recognize the same langua ata

Theorem

For **every NFA**, there **exists** an **equivalent DFA**.

For every NFA N, there exist a DFA M with $L(M) = L(N)$.

Corollary

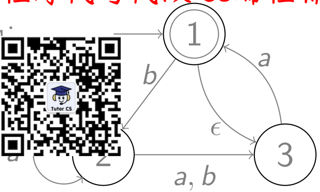
A language L is regular, if and only if there exists NFA that recognizes L.

To prove the theorem on the previous slide, we need to show that for every NFA N, there exists a DFA M with $L(M) = L(N)$.

Before we see the general proof for this, we go through one concrete example. We transform NFA N_4 into an equivalent DFA.

Transforming an NFA into a DFA–example

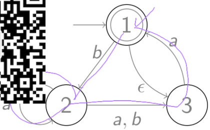
Consider the NFA N_4:



Transforming an NFA into a DFA—example

Consider th

$\Sigma = \{a, b\}$



Examples of accepted words

ε ✓

a ✓

"

εa

baba ✓

Examples of words that are not accepted:

bbb ✗

bb ✗


Transforming an NFA into a DFA–example
Consider the
Σ = {a, b}
Q = {1, 2, 3}
Examples of accepted words
ε
a
baba
Equivalent DFA
Examples of words that are not accepted:
bbb
bb
Talking: Ruth Urner


Transforming an NFA into a DFA–example
Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save
Who can see what you share here?
Talking: Ruth Urner
Consider the
Σ = {a,b}
Q = {1,2,3}
1
2
3
b
a
ε
a
a, b
Examples of accepted words
ε ✓
a ✓
= εa
baba ✓
Equivalent DFA
a,b
∅
{3}
{1,3}
{2,3}
{2}
{1,2,3}
Examples of words that are not accepted:
bbb X
bb X

Proof

Let $N = (Q, \Sigma, \delta, q_0, F)$ A.

We need to construct a $Q', \Sigma, \delta', q_0', F')$ with $L(M) = L(N)$.

Assuming that N does n ny ϵ-transitions, we construct M as follows:

1. $Q' = \mathcal{P}(Q)$
2. For $R \in Q'$ and $a \in \Sigma$, we let

$$\delta'(R, a) = \{q \in Q \mid q \in \delta(r, a) \text{ for some } r \in R\}$$

3. $q_0' = \{q_0\}$
4. $F' = \{R \in Q' \mid \text{there exists an } r \in F \text{ with } r \in R\}$.

This completes the construction for NFAs that don't contain ϵ-transitions.

General proof of Theorem

Proof

Let $N = (Q, \Sigma$ … e NFA.

We need to c… $= (Q', \Sigma, \delta', q_0', F')$ with $L(M) = L(N)$.

Assuming that N does not contain any ϵ-transitions, we construct M as follows:

$\in Q' = \mathcal{P}(Q)$

$\subseteq Q$

1. $Q' = \mathcal{P}(Q)$
2. For $R \in Q'$ and $a \in \Sigma$, we let

$$\delta'(R, a) = \{q \in Q \mid q \in \delta(r, a) \text{ for some } r \in R\}$$

3. $q_0' = \{q_0\}$
4. $F' = \{R \in Q' \mid \text{there exists an } r \in F \text{ with } r \in R\}$.

This completes the construction for NFAs that don't contain ϵ-transitions.

Proof of Theorem

Proof continued

If N does contain ϵ-trans xtend the above construction as follows:

For a state $R \in Q'$, we let $E(R)$ denote the set of states that can be reached via 0 or more ϵ-transitions from some state $r \in R$ (in N).

Then we modify the transition function of M to be

$$\delta'(R, a) = \{q \in Q \mid q \in E(\delta(r, a)) \text{ for some } r \in R\}$$

and the start state of M to be

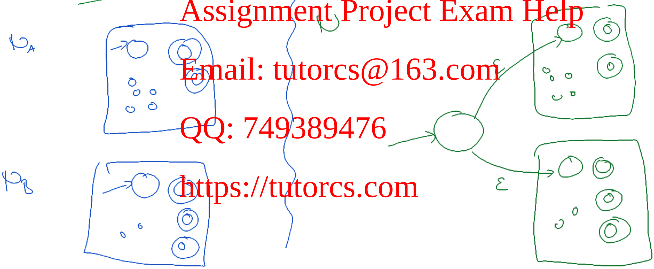
$$q_0' = E(q_0).$$

Theorem

Regular languages are closed under unions.

"Proof" by illustration

Given A and B ... NFAs N_A and N_B with $L(N_A) = A$ and $L(N_B) = B$. We need to construct an NFA N that recognizing $A \cup B$.



Theorem

Regular languages are closed under unions.

Proof (formal version)

We need to show that if two languages A and B are regular, then so is $A \cup B$. If A and B are regular, then there exist NFAs $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ such that $L(N_1) = A$ and $L(N_2) = B$. We need to show that there exists an NFA M with $L(M) = A \cup B$. The picture on the previous slide illustrates a construction for $M = (Q, \Sigma, \delta, q_0, F)$. Formally, we set:

1. $Q = Q_1 \cup Q_2 \cup \{q_0\}$, where q_0 is a new state, that is $q_0 \notin Q_1 \cup Q_2$ and this state q_0 will then also be the start state of M.

2. $\delta(q, a) = \begin{cases} \{q_1, q_2\} & \text{if} \quad q = q_0 \quad \text{and} \quad a = \epsilon \\ \delta_i(q, a) & \text{if} \quad q \in Q_i \\ \emptyset & \text{else} \end{cases}$

3. $F = F_1 \cup F_2$.

Regular languages are closed under regular operations

Theorem

Regular languages are closed under concatenations.

“Proof” by illustrat

Theorem

Regular languages are closed under concatenations.

Proof (formal version)

We need to show that if two languages A and B are regular, then so is $A \circ B$. If A and B are regular, then there exist NFAs $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ such that $L(N_1) = A$ and $L(N_2) = B$. We need to show that there exists an NFA M with $L(M) = A \circ B$. The picture on the previous slide illustrates a construction for $M = (Q, \Sigma, \delta, q_0, F)$. Formally, we set:

1. $Q = Q_1 \cup Q_2$
2. $q_0 = q_1$
3. $\delta(q, a) = \begin{cases} \delta_1(q, a) \cup \{q_2\} & \text{if} \quad q \in F_1 \text{ and } a = \epsilon \\ \delta_1(q, a) & \text{if} \quad q \in Q_1 \text{ and } (q \notin F_1 \text{ or } a \neq \epsilon) \\ \delta_2(q, a) & \text{if} \quad q \in Q_2 \end{cases}$
4. $F = F_2$.

Theorem

Regular languages are closed under the star-operation.

"Proof" by illustration

Regular languages are closed under regular operations

Theorem

Regular languages are closed under the star-operation.

Proof (formal version)

We need to show that if language A is regular, then so is A^*. If A is regular, then there exist an NFA $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ such that $L(N_1) = A$. We need to show that there exists an NFA M with $L(M) = A^*$. The picture on the previous slide illustrates a construction for $M = (Q, \Sigma, \delta, q_0, F)$. We set:

1. $Q = Q_1 \cup \{q_0\}$, where q_0 is a new state, that is $q_0 \notin Q_1 \cup Q_2$ and this state q_0 will then also be the start state of M.
2. $\delta(q, a) = \begin{cases} \{q_1\} & \text{if} \quad q = q_0 \quad \text{and} \quad a = \epsilon \\ \delta_1(q, a) & \text{if} \quad q \in Q_1 \text{ and } (q \notin F_1 \text{ or } a \neq \epsilon) \\ \delta_1(q, a) \cup \{q_1\} & \text{if} \quad q \in F_1 \text{ and } a = \epsilon \end{cases}$
3. $F = F_1 \cup \{q_0\}$.