# cs2001

# Introductior Theory of Computation

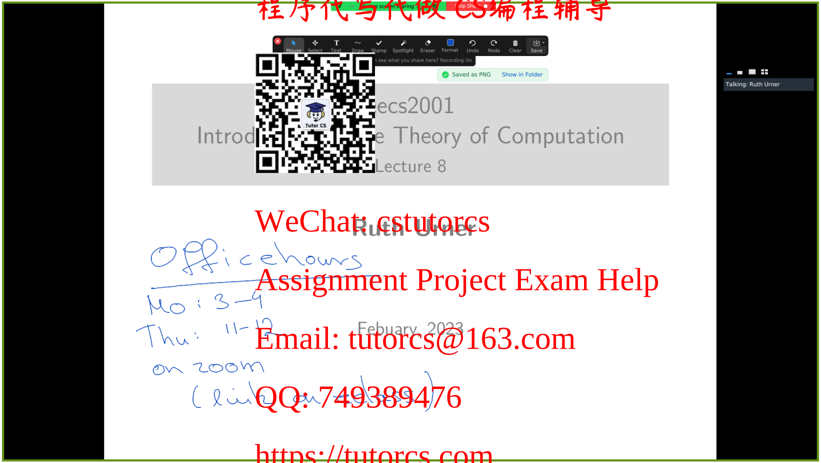
ecture 8

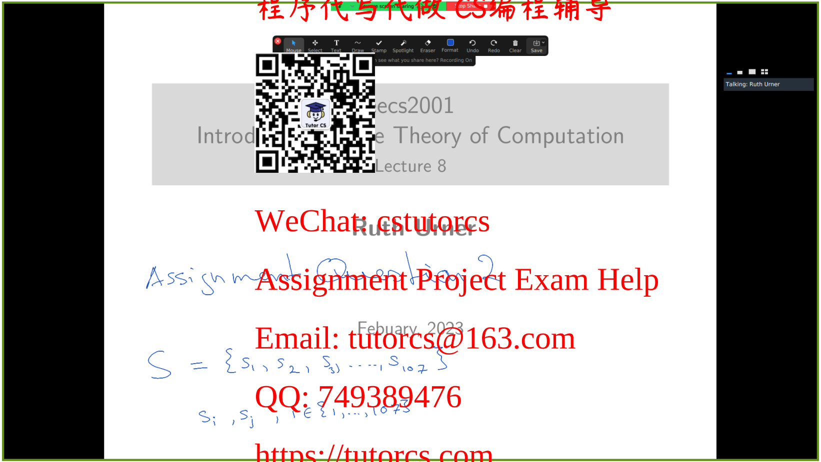


**Ruth Urner**



Febuary, 2023


ecs2001
Introd e Theory of Computation
Lecture 8
Ruth Urner
Febuary, 2023
Office hours
Mo : 3–4
Thu : 11-12
on zoom
Talking: Ruth Urner



Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save
Talking: Ruth Urner
ecs2001
Introd
e Theory of Computation
Lecture 8
Ruth Urner
Assignment Question 2
Febuary, 2023
S = {s1, s2, s3, ...., s107}
si , sj , i ∈ {1,...,107}

# Finite Automata



Reading: ITC Section 1.1

# Finite Automata



A finite automaton or finite state machine is a simple computational model.



We will work with this model of computation for the next part of this course.

# A simple automaton–sliding door example

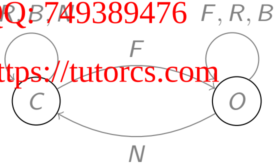
Consider an automatic sliding door sliding door with two pads that receive signals if someone is standing on them:






We can model the controller of the sliding door as a simple automaton:

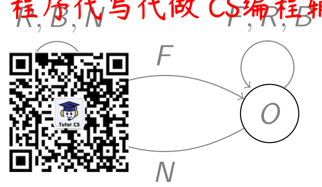





Here we use: $C = CLOSED$, $O = OPEN$, $F = FRONT$, $R = REAR$, $B = BOTH$, $N = NEITHER$

# A simple automaton–sliding door example






Here we use: $C = CLOSED$, $O = OPEN$, $F = FRONT$, $R = REAR$, $B = BOTH$, $N = NEITHER$



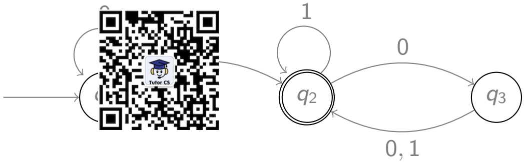
The behavior of the door can be described in terms of the following transition function:

| | NEITHER | FRONT | REAR | BOTH |
|---|---|---|---|---|
| CLOSED | CLOSED | OPEN | CLOSED | CLOSED |
| OPEN | CLOSED | OPEN | OPEN | OPEN |

# State diagram of $M_1$

We can use a state diagram to describe a finite automaton $M_1$:








**Interpretation of the state diagram:** The arrow "coming out of nowhere" going into the leftmost state, signals, that this state is the start state. This automaton can read letters from the alphabet $\Sigma = \{0, 1\}$. Being in some state $q$, receiving letter $\sigma$, the computation finds the outgoing edge from $q$ that has a label $\sigma$, and moves along that arrow to a new state.

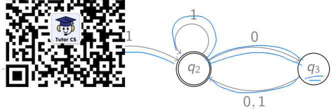


**Examples:**

- If we feed the string 10010 to $M_1$, we move through the states $q_1, q_2, q_3, q_2, q_2, q_3$, and end up in state $q_3$, which is not an accept state.
- If we feed the string 1101 to $M_1$, we end up in state $q_2$, which is an accept state (accept states are the nodes with a double circle).
- If we feed the empty string $\epsilon$ to $M_1$, we end up in state $q_1$, which is not an accept state.

## State diagram of $M_1$

We can use a st … scribe a finite automaton $M_1$:






**Interpretation of the state diagram:** The arrow "coming out of nowhere" going into the leftmost state, signals, that this state is the start state. This automaton can read letters from the alphabet $\Sigma = \{0, 1\}$. Being in some state $q$, receiving letter $\sigma$, the computation finds the outgoing edge from $q$ that has a label $\sigma$, and moves along that arrow to a new state.



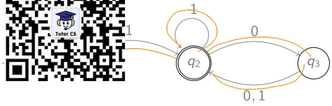
**Examples:**

- If we feed the string 10010 to $M_1$ …

# State diagram of $M_1$



We can use a st… …scribe a finite automaton $M_1$:






**Interpretation of the state diagram:** The arrow "coming out of nowhere" going into the leftmost state, signals, that this state is the start state. This automaton can read letters from the alphabet $\Sigma = \{0, 1\}$. Being in some state $q$, receiving letter $\sigma$, the computation finds the outgoing edge from $q$ that has a label $\sigma$, and moves along that arrow to a new state.



**Examples:**

- If we feed the string 10010 to $M_1$, we move through the states $q_1, q_2, q_3, q_2, q_2, q_3$, and end up in state $q_3$, which is not an accept state.
- If we feed the string 1101 to $M_1$, ends up in $q_2$, string is accepted.

**Definition**

A 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. $\Sigma$ is a finite set called the alphabet,
3. $\delta : Q \times \Sigma \to Q$ 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 definition of a finite



**Definition**

A 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. $\Sigma$ is a finite set called the alphabet,
3. $\delta : Q \times \Sigma \to Q$ 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 $M_1$




The above state diagram corresponds to the following formal description: $M_1 = (Q, \Sigma, \delta, q_1, F)$, where



1. $Q = \{q_1, q_2, q_3\}$,
2. $\Sigma = \{0, 1\}$,
3. $\delta$ is defined by the following table:

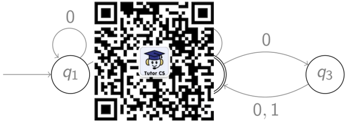
| $\delta$ | 0 | 1 |
|---|---|---|
| $q_1$ | $q_1$ | $q_2$ |
| $q_2$ | $q_3$ | $q_2$ |
| $q_3$ | $q_2$ | $q_2$ |

4. $q_1$ is the start state, and
5. $F \subseteq \{q_2\}$.

Given the description of an automaton, we can ask: which strings will lead to an accept state when fed into the automaton? As we have seen in the example computations with $M_1$ before, some strings do and others don't. The set of strings that do lead to an accept state form a language over $\Sigma$, the language of $M_1$.

# Formal description of $M_1$






The above state diagram corresponds to the following formal description:
$M_1 = (Q, \Sigma, \delta, q_1, F)$, where

$Q = \{q_1, q_2, q_3\}$

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

$q_1$ is starting state

$F = \{q_2\}$

| $\delta$ | 0 | 1 |
|---|---|---|
| $q_1$ | $q_1$ | $q_2$ |
| $q_2$ | $q_3$ | $q_2$ |
| $q_3$ | $q_2$ | $q_2$ |

# Language accepted by an automaton



Let $\Sigma$ be the alphabet of the automaton $M$. Then we let

$$L(M) = \{w \in \Sigma^k \mid k \in \mathbb{N} \text{ and } w \text{ is accepted by } M\}$$



denote the language of machine $M$. The is $L(M)$ is the set of all words over $\Sigma$ that are accepted by machine $M$.



For the language $A = L(M)$ we also say machine $M$ **recognizes** (or **accepts**) $A$.

# Formal definition of acceptance



**Definition**

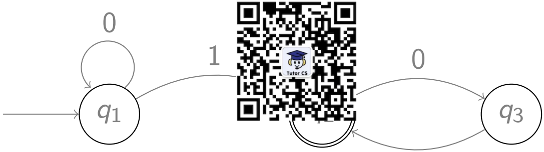
Let $M = (Q, \Sigma, \delta, q_0, F)$ a finite automaton and $\mathbf{w} = w_1 w_2 \ldots w_n$ a string over $\Sigma$. We say that $M$ accepts $\mathbf{w}$ if there exists a sequence $s_0 s_1 s_2 \ldots s_n$ of states such that



1. $s_0 = q_0$,
2. $\delta(s_i, w_{i+1}) = s_{i+1}$ for $i = 0, 1, \ldots, n-1$, and
3. $s_n \in F$.

# Language accepted by $M_1$





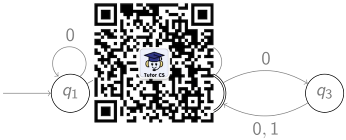



For the machine $M_1$ we get

$$L(M_1) = \{\mathbf{w} \mid \mathbf{w} \text{ contains at least one 1 and the number of 0s after the last 1 is even}\}$$

**Task for you:** convince yourself that this is exactly the set of words accepted by this automaton.

# Language accepted by $M_1$




What is the language of $M_1$?

- all words that end with 1 are accepted

$\{ w = w_1 w_2 \dots w_k \in \Sigma^* \mid w_k = 1 \} \subseteq L(M_1)$

- but there are other words (e.g. 100) that don't end with 1 and are also accepted.

# Language accepted by $M_1$




0100 gets accepted

For the machine $M_1$ we get

$L(M_1) =$



**Task for you:** figure out what exactly is the set of words accepted by this automaton.

# Language accepted by $M_1$




For the machine $M_1$ we get

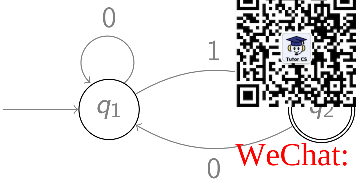
$L(M_1) =$ { w | w contains at least one 1 and the number of 0's after the last 1 is even }



**Task for you:** figure out what exactly is the set of words accepted by this automaton.

# Examples automaton $M_2$



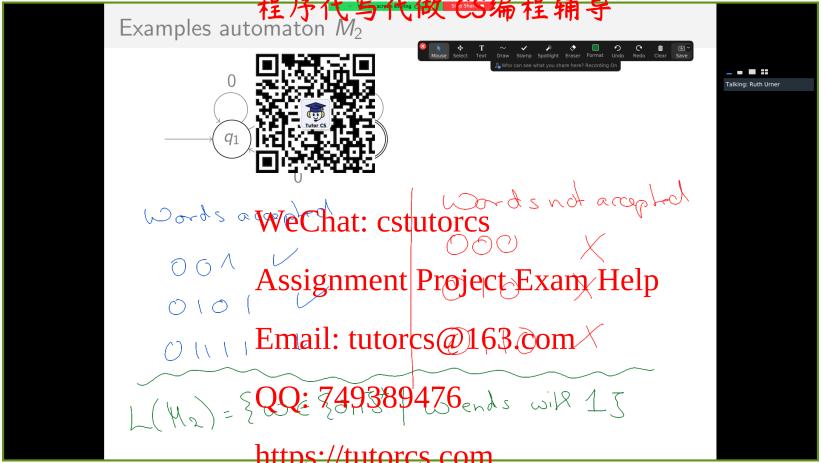


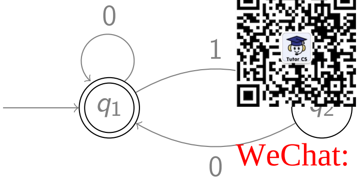



For the machine $M_2$ we get

$$L(M_2) = \{w \in \{0,1\}^* \mid w \text{ ends with letter } 1\}$$

**Task for you:** convince yourself that this is exactly the set of words accepted by this automaton.


Examples automaton $M_2$
0
$q_1$
0
Words accepted
001
0101
01111
Words not accepted
000
Talking: Ruth Urner

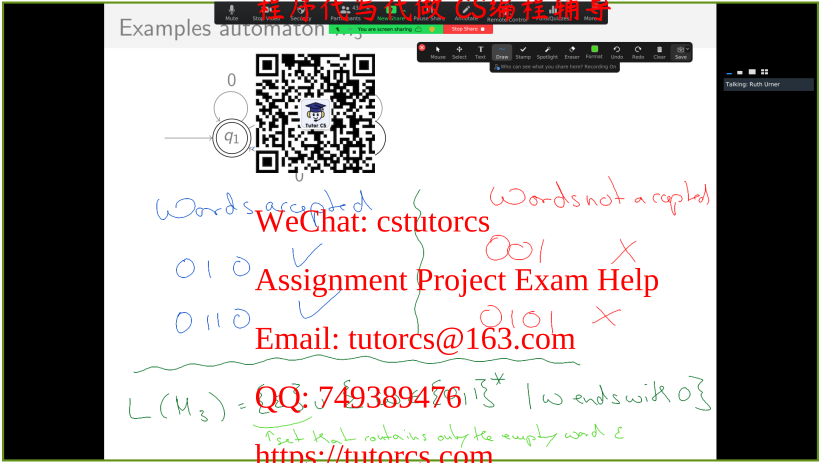





For the machine $M_3$ we get

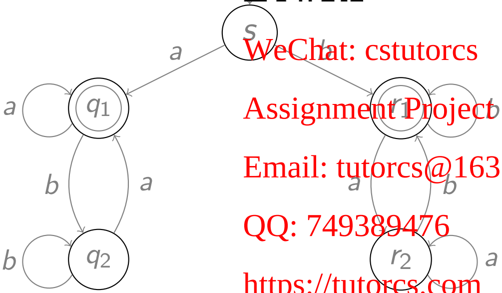
$$L(M_3) = \{w \mid w \in \{0,1\}^* \text{ is the empty string or ends with letter } 0\}$$

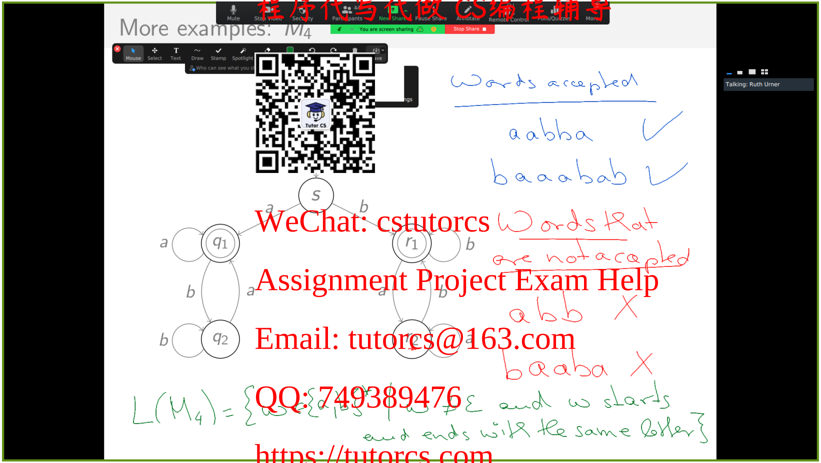
**Task for you:** convince yourself that this is exactly the set of words accepted by this automaton.


Examples automaton M3
0
q1
0
Words accepted
010
0110
Words not accepted
001
0101
L(M3) = {ε} ∪ {w ∈ {0,1}* | w ends with 0}
set that contains only the empty word ε
You are screen sharing
Stop Share
Talking: Ruth Urner

# More examples: $M_4$






More examples: $M_4$
s
a
b
$q_1$
$r_1$
a
b
b
a
a
b
$q_2$
$r_2$
b
a
Words accepted
aabba ✓
baaabab ✓
Words that are not accepted
abb X
baaba X
$L(M_4) = \{w \in \{a,b\}^* \mid w \neq \varepsilon$ and $w$ starts and ends with the same letter$\}$
Talking: Ruth Urner

# Design an automaton that verifies if a string contains an odd number of 1s



# Design an automaton that verifies if a string contains 001 as a substring

# Regular Languages



Reading: ITC Section 1.1

# Regular language



**Definition**

A language $L$ over some alphabet $\Sigma$ is called a regular language if there exists a finite automaton $M$ such that $L = L(M)$, that is, if there exists a finite automaton that recognizes it.

## Regular language

$\Sigma$

**Definition**

A language $L$ is called a regular language if there exists a finite automaton $M$ such that $L = L(M)$, that is, if there exists a finite automaton that recognizes it.



Examples of regular languages

- $\{ w \in \{0,1\}^* \mid w \text{ contains } 001 \text{ as a substring} \}$
- $\{ w \in \{0,1\}^* \mid \text{the number of } 1 \text{ is odd} \}$

**Definition:**
Let $A$ and $B$ be languages. Then we define the following operations that each form a new language:



- **Union:** $A \cup B = \{\mathbf{w} \mid \mathbf{w} \in A \text{ or } \mathbf{w} \in B\}$.
- **Concatenation:** $A \circ B = \{\mathbf{wv} \mid \mathbf{w} \in A \text{ and } \mathbf{v} \in B\}$.
- **Star:**
  $A^* = \{\mathbf{w}_1\mathbf{w}_2 \ldots \mathbf{w}_k \mid k \geq 0 \text{ and } \mathbf{w}_i \in A \;\forall\, i \in \{0, 1, \ldots, k\}\}$

## Regular o…

In particular for an alphabet $\Sigma$, the set $\Sigma^*$ is the ~~the~~ set of all finite words over $\Sigma$.

Ex: $A = \{01 \ldots 01\}$
$B = \{1, 1 \ldots$

$A \times B \ni (01, 110)$ not $\in \{0,1\}^*$

**Definition:**
Let $A$ and $B$ … Then we define the following operations that each form a new language:

- **Union:** $A \cup B = \{w \mid w \in A \text{ or } w \in B\}$.

Ex $A \cup B = \{01, 001, 0001, 1, 11, 110, 10\}$

- **Concatenation:** $A \circ B = \{wv \mid w \in A \text{ and } v \in B\}$

Ex $A \circ B = \{011, 0011, 00011, 0111, 00111, 000111, 01110, 001110, 0001110, 0110, 00110, 000110\}$

- **Star:**
$A^* = \{w_1 w_2 \ldots w_k \mid k \geq 0 \text{ and } w_i \in A \; \forall i \in \{0, 1, \ldots, k\}\}$

Ex $A^* = \{\varepsilon, 01, 001, 0001, 0101, 00101, 01001, 000101, 0010001, 0001001, 010101, 0100101, \ldots\}$

infinite set of words…

Talking: Ruth Urner

# Regular operations

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

Ex: $A = \{01, \ldots\}$, $B = \{1, \ldots\}$

**Definition:**
Let $A$ and $B$ be languages. Then we define the following operations that each form a new language:

- **Union:** $A \cup B = \{w \mid w \in A \text{ or } w \in B\}$.

Ex $A \cup B = \{01, 001, 0001, 1, 11, 110, 10\}$

- **Concatenation:** $A \circ B = \{wv \mid w \in A \text{ and } v \in B\}$

Ex $A \circ B = \{011, 0011, 00011, 0111, 00111, 000111, 01110, 001110, 0001110, 0110, 00110, 000110\}$

- **Star:**
$A^* = \{w_1 w_2 \ldots w_k \mid k \geq 0 \text{ and } w_i \in A\ \forall i \in \{0, 1, \ldots, k\}\}$

Ex $A^* = \{\varepsilon, 01, 001, 0001, 0101, 00101, 01001, 010001, 000101, 0010001, 0001001, 010101, 0100101, \ldots\}$

infinite set of words...

In particular for an alphabet $\Sigma$, the set $\Sigma^*$ is the set of all finite words over $\Sigma$.

$B \notin A^*$, $A \notin B^*$

$A \times B \ni (01, 110)$

not $\in \{0,1\}^*$

$C = \{0,1\}$

$B \subsetneq C^*$

$A \subsetneq C^*$

$D = \{0, 1, 00, 11\}$

$B \subsetneq D^*$

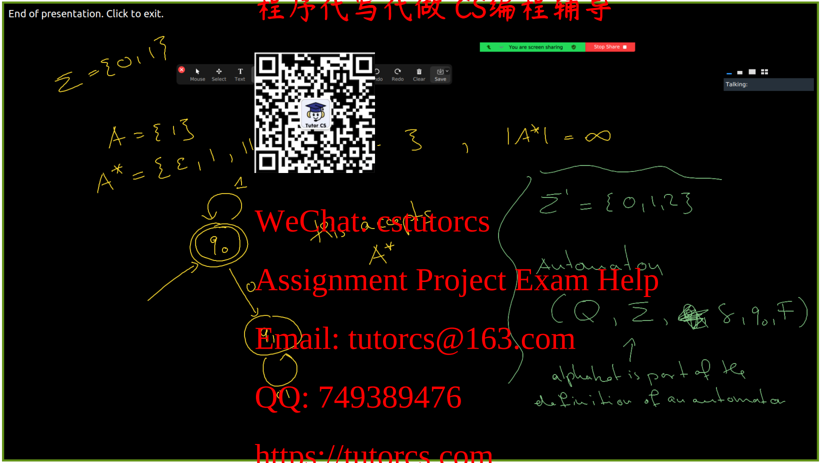
$A \subsetneq D^*$

Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save

Who can see what you share here?

You are screen sharing Stop Share

Talking: Ruth Urner



End of presentation. Click to exit.
You are screen sharing
Stop Share
Mouse
Select
Text
Redo
Clear
Save
Talking:
Σ = {0,1}
A = {1}
A* = {ε, 1, 11, ... }
|A*| = ∞
Σ' = {0,1,2}
Automaton
(Q, Σ, δ, q0, F)
alphabet is part of the definition of an automaton

# Regular operations reproduce regular languages



**Theorem**
The set of all regular languages is closed under the three regular operations.



That is: if $A$ and $B$ are regular languages, then $A^*$, $A \cup B$ and $A \circ B$ are also regular languages.

# Problem 1.4 from ITC

Here we design an automaton that recognizes the intersection of two languages for which we have simple automata recognizing them.



We will now use a similar “product” for the general construction of an automaton that recognizes the union of two regular languages.

# Problem 1.4 from ITC

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

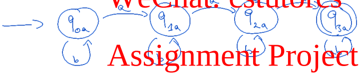
Here we design a [...] cognizes the intersection of two languages for which we have si[...]nizing them.

Given two [...] A and B.

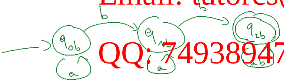
$A = \{ w \in$ [...] ontains at least 3 a's $\}$

$B = \{ w \in \Sigma^*$ | w contains at least 2 b's $\}$

We define automaton $M_1$ with $L(M_1) = A$



and automaton $M_2$ with $L(M_2) = B$



Talking: Ruth Urner

# Problem 1.4 from ITC

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

Here we design a [QR code covers text] ...cognizes the intersection of two languages for which we have si[QR code covers text]...nizing them.

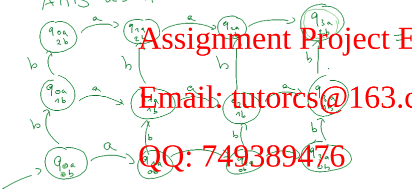
Given two ... A and B.

$A = \{ w \in$ ... contains at least 3 a's $\}$

$B = \{ w \in \Sigma^* \mid w$ contains at least 2 b's $\}$

We can now construct automaton M that A∩B as follows:

$A \cap B = \{ w \in \Sigma^* \mid w$ contains at least 3 a's and at least 2 b's $\}$



$M = (Q_1 \times Q_2,$

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

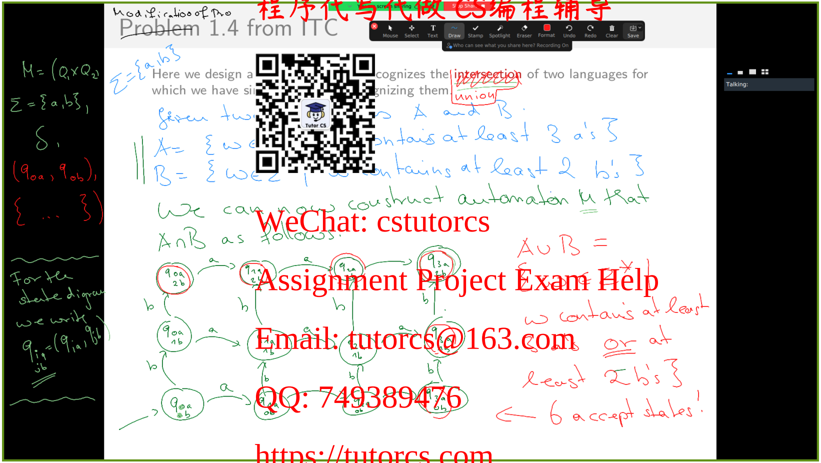
$\delta,$

$(q_{0a}, q_{0b}),$

$\{(q_{3a}, q_{2b})\})$

For the state diagram we write $q_{ij} = (q_{ia}, q_{jb})$

Talking: Asish Dan Philips .


Modification of Pro
Problem 1.4 from ITC
Here we design a ... cognizes the intersection of two languages for which we have si... gnizing them.
union
$\Sigma = \{a, b\}$
Given two ... A and B.
A = { w ... contains at least 3 a's }
B = { w ... contains at least 2 b's }
We can now construct automaton M that
A∩B as follows:
$M = (Q_1 \times Q_2,$
$\Sigma = \{a, b\},$
$\delta,$
$(q_{0a}, q_{0b}),$
{ ... })
For the state diagram we write
$q_{i_a j_b} = (q_{i_a}, q_{j_b})$
A ∪ B =
{ w contains at least 3 a's or at least 2 b's }
← 6 accept states!

# Regular languages closed under unions



We start by proving that if $A$ and $B$ are regular languages over some alphabet $\Sigma$, then so is $A \cup B$.

**Proof:**
To prove that $A \cup B$ is regular, we need to show that there exists an automaton



$$M = (Q, \Sigma, \delta, q_0, F)$$



with



$$L(M) = A \cup B$$

Since $A$ and $B$ are regular, we know there exist automata $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ with





$$L(M_1) = A \text{ and } L(M_2) = B$$

# Regular languages closed under unions



general $\Sigma$

We start by p[...] $A$ and $B$ are regular languages over some alphabe[t] [...] $A \cup B$.

Goal for [...]

To show that $A \cup B$ is regular, we need show that there exists an automaton that recognizes $A \cup B$.

We know that $A$ and $B$ are regular languages [...] there exist automata $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 (Q_2, \Sigma, \delta_2, q_2, F_2)$ such that $A = L(M_1)$ and $B = L(M_2)$.

Given $M_1$ and $M_2$ the goal is to construct an automaton that recognizes $A \cup B$ using the components of $M_1$ and $M_2$

**Proof continued:**

We now define the re[illegible]tomaton $M$ as follows:

1. $Q = Q_1 \times Q_2$
2. We use the same alphabet $\Sigma$
3. $\delta((s_1, s_2), a) = (\delta_1(s_1, a), \delta_2(s_2, a))$
4. $q_0 = (q_1, q_2)$
5. $F = \{(s_1, s_2) \in Q \mid s_1 \in F_1 \text{ or } s_2 \in F_2\}$

## Regular languages closed

$M_1 = (Q_1, \Sigma,$
$\delta_1, q_1, F_1)$

$M_2 = (Q_2, \Sigma,$
$\delta_2, q_2, F_2)$

$L(M_1) = A$
$L(M_2) = B$

**Proof contin**

We now defin automaton $M$ as follows:

1. $Q = Q_1 \times Q_2$
2. We use the same alphabet $\Sigma$
3. $\delta((s_1, s_2), a) = (\delta_1(s_1, a), \delta_2(s_2, a))$
4. $q_0 = (q_1, q_2)$
5. $F = \{(s_1, s_2) \in Q \mid s_1 \in F_1 \text{ or } s_2 \in F_2\}$

Talking: Ruth Urner

# Non-deterministic finite automata



Reading: ITC Section 1.2

# DFA versus NFAs



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

$$L(N_1) =$$


Example: the NFA $N_1$
Differences
ε can be "read"
0, 1
0, 1
$q_1$
$q_2$
ε
$q_3$
1
$q_4$
some states have several instructions for the same letter
some states
for all letters
Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save
Who can see what you share here? Recording On
Talking: Ruth Urner


Example: the NFA $N_1$
0, 1
0, 1
$q_1$
$q_2$
$q_3$
$q_4$
ε
1
Differences
ε can be "read"
Examples of words
10110
accepted
1000
not accepted
some states have several instructions for the same letter
some states
for all letters
A word is accepted by an NFA if there is at
the word that leads to an accept state.
Talking: Asish Dan Philips .

Example: the NFA $N_1$
Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save
Who can see what you share here?
Talking: Asish Dan Philips .
Differences
ε can be "read"
0, 1
0, 1
$q_1$
$q_2$
$q_3$
$q_4$
ε
1
Examples of words
10110 ✓
accepted
1000 X
not accepted
some states have several instructions for the same letter
some states
for all letters
1111 = 1ε111
ε
A word is accepted by an NFA if there is at least one reading the word that leads to an accept state.