The Regular Operations
Introduction

- Once automata and computation have been defined, their properties need be studied.
Introduction

• Once automata and computation have been defined, their properties need be studied

• This requires appropriate tools and technique which we initiate here
Introduction

• Once automata and computation have been defined, their properties need be studied.

• This requires appropriate tools and technique which we initiate here.

• We also need tools and techniques for studying non-regular languages, i.e., languages which are beyond the capability of finite automata.
Contrast

- **Arithmetic:** basic objects are **numbers** and the **tools** are **operations** for number manipulation, such as $+$, $\times$. 
Contrast

- **Arithmetic**: basic objects are **numbers** and the tools are **operations** for number manipulation, such as $+$, $\times$.
- **Theory of Computation**: objects are **languages** and the tools for language manipulation, specifically designed...
Contrast

- **Arithmetic**: basic objects are **numbers** and the tools are **operations** for number manipulation, such as +, ×.

- **Theory of Computation**: objects are **languages** and the tools for language manipulation, **specifically designed**.

- The three common operations on languages, **regular operations**: union \( \cup \), concatenation \( \circ \), and **star** \( \star \).
Definition 1.10

Let $A$ and $B$ be languages. We define regular operations union, concatenation and star as follows:
Definition 1.10

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

- **Union:** $A \cup B = \{ x \mid x \in A \lor x \in B \}$
Definition 1.10

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

- **Union:** $A \cup B = \{x \mid x \in A \lor x \in B\}$
- **Concatenation:** $A \circ B = \{xy \mid x \in A \land y \in B\}$
Definition 1.10

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

- **Union:** $A \cup B = \{ x \mid x \in A \lor x \in B \}$
- **Concatenation:** $A \circ B = \{ xy \mid x \in A \land y \in B \}$
- **Star:** $A^* = \{ x_1 x_2 \ldots x_k \mid k \geq 0 \land x_i \in A, 1 \leq i \leq k \}$
Question

- $A \cup B$, $A \circ B$ and $A^*$ are languages
Question

- \( A \cup B, A \circ B \) and \( A^* \) are languages

- Consequently \( A \cup B, A \circ B \) and \( A^* \) are sets of strings over some alphabets
Question

• \( A \cup B, A \circ B \) and \( A^* \) are languages

• Consequently \( A \cup B, A \circ B \) and \( A^* \) are sets of strings over some alphabets

• What are these alphabets?
Question

- $A \cup B$, $A \circ B$ and $A^*$ are languages
- Consequently $A \cup B$, $A \circ B$ and $A^*$ are sets of strings over some alphabets
- What are these alphabets?

  - Answer: if $\Sigma_A$ and $\Sigma_B$ are the alphabets of $A$ and $B$ then these alphabets are $\Sigma_A \cup \Sigma_B$
Question

- $A \cup B$, $A \circ B$ and $A^*$ are languages
- Consequently $A \cup B$, $A \circ B$ and $A^*$ are sets of strings over some alphabets
- What are these alphabets?
  - Answer: if $\Sigma_A$ and $\Sigma_B$ are the alphabets of $A$ and $B$ then these alphabets are $\Sigma_A \cup \Sigma_B$
  - Reason: Any language over $\Sigma_A$ or $\Sigma_B$ is certainly a language over $\Sigma_A \cup \Sigma_B$. Hence, we may assume $\Sigma_A = \Sigma_B$
Observations

- **Union** is the usual operation on sets
Observations

- **Union** is the usual operation on sets.
- **Concatenation** is a little trickier; it attaches a string from $A$ in front of a string from $B$ in all possible ways to get strings from $A \circ B$. 

The Regular Operations – p.6/19
Observations

- **Union** is the usual operation on sets.
- **Concatenation** is a little trickier; it attaches a string from $A$ in front of a string from $B$ in all possible ways to get strings from $A \circ B$.
- **Star** operation is different; it applies to one language, i.e., it is *unary* rather than binary. Star works by attaching any number of strings in $A$ together to get strings in $A^*$. 
Because “any number” includes 0, \( \epsilon \in A^* \), no matter what \( A \) is.
Example 1.11

Let $\Sigma = \{a, b, \ldots, z\}$. If $A = \{good, bad\}$ and $B = \{boy, girl\}$, then:
Example 1.11

Let $\Sigma = \{a, b, \ldots, z\}$. If $A = \{good, bad\}$ and $B = \{boy, girl\}$, then:

- $A \cup B = \{good, bad, boy, girl\}$
Example 1.11

Let $\Sigma = \{a, b, \ldots, z\}$. If $A = \{\text{good, bad}\}$ and $B = \{\text{boy, girl}\}$, then:

- $A \cup B = \{\text{good, bad, boy, girl}\}$
- $A \circ B = \{\text{goodboy, goodgirl, badboy, badgirl}\}$
Example 1.11

Let $\Sigma = \{a, b, \ldots, z\}$. If $A = \{\text{good, bad}\}$ and $B = \{\text{boy, girl}\}$, then:

- $A \cup B = \{\text{good, bad, boy, girl}\}$
- $A \circ B = \{\text{goodboy, goodgirl, badboy, badgirl}\}$
- $A^* = \{\varepsilon, \text{goob, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, goodgoodbad, goodbadgood, goodbadbad, \ldots}\}$
Closed set

If $A$ is a set and $f : A \times A \rightarrow A$ is total, i.e.,
$\forall a, b \in A : f(a, b) \in A$ then we say that $A$ is closed under $f$. 

Example: consider $A$ and $B$ where $A$ is number multiplication and $B$ is number division.
Since multiplication of natural numbers is total (i.e., over all defined) it is closed under division.
Since the division of two natural numbers is not always a natural number, for example, is not closed under division.
Closed set

If $A$ is a set and $f : A \times A \to A$ is total, i.e.,

$$\forall a, b \in A : f(a, b) \in A$$

then we say that $A$ is closed under $f$

Example: consider $\mathcal{N} = \{1, 2, 3, \ldots\}$ and $\ast, \div : \mathcal{N} \times \mathcal{N} \to \mathcal{N}$ where $\ast$ is number multiplication and $\div$ is number division.
If $A$ is a set and $f : A \times A \rightarrow A$ is total, i.e.,
$$\forall a, b \in A : f(a, b) \in A$$
then we say that $A$ is closed under $f$.

Example: consider $\mathcal{N} = \{1, 2, 3, \ldots\}$ and $\times, / : \mathcal{N} \times \mathcal{N} \rightarrow \mathcal{N}$ where $\times$ is number multiplication and $/$ is number division.

- Since multiplication of natural numbers is total (i.e., over all defined) $\mathcal{N}$ is closed under $\times$. 
If $A$ is a set and $f : A \times A \rightarrow A$ is total, i.e.,
\[ \forall a, b \in A : f(a, b) \in A \]
then we say that $A$ is closed under $f$

Example: consider $\mathcal{N} = \{1, 2, 3, \ldots\}$ and $\ast, \div : \mathcal{N} \times \mathcal{N} \rightarrow \mathcal{N}$ where $\ast$ is number multiplication and $\div$ is number division.

- Since multiplication of natural numbers is total (i.e., over all defined) $\mathcal{N}$ is closed under $\ast$

- Since the division of two natural numbers is not always a natural number, for example $1/2 \notin \mathcal{N}$, $\mathcal{N}$ is not closed under division
Closure of regular languages

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

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

- **This property** provides **useful tools** for manipulating regular languages and for understanding the power of finite automata.
The class of regular languages is closed under union operation, i.e. if $A_1$ and $A_2$ are regular then $A_1 \cup A_2$ is regular.
Proof idea

• Because $A_1$ and $A_2$ are regular there are automata $M_1$ and $M_2$ recognizing $A_1$ and $A_2$, respectively.
Proof idea

- Because $A_1$ and $A_2$ are regular there are automata $M_1$ and $M_2$ recognizing $A_1$ and $A_2$, respectively.
- To prove that $A_1 \cup A_2$ is regular we construct a finite automaton $M$ from $M_1$ and $M_2$ that recognizes $A_1 \cup A_2$. 

The Regular Operations – p.12/19
Proof idea

- **Because** $A_1$ and $A_2$ are regular there are automata $M_1$ and $M_2$ recognizing $A_1$ and $A_2$, respectively.
- **To prove that** $A_1 \cup A_2$ **is regular** we construct a finite automaton $M$ from $M_1$ and $M_2$ that recognizes $A_1 \cup A_2$.
- **The machine** $M$ **works by** simulating $M_1$ and $M_2$. 

The Regular Operations – p.12/19
Proof idea

- Because $A_1$ and $A_2$ are regular there are automata $M_1$ and $M_2$ recognizing $A_1$ and $A_2$, respectively.
- To prove that $A_1 \cup A_2$ is regular we construct a finite automaton $M$ from $M_1$ and $M_2$ that recognizes $A_1 \cup A_2$.
- The machine $M$ works by simulating $M_1$ and $M_2$.
- Simulation: pretend that you are $M$. As you read the input symbols you simulate both $M_1$ and $M_2$, simultaneously.
Proof idea

- Because $A_1$ and $A_2$ are regular there are automata $M_1$ and $M_2$ recognizing $A_1$ and $A_2$, respectively.
- To prove that $A_1 \cup A_2$ is regular we construct a finite automaton $M$ from $M_1$ and $M_2$ that recognizes $A_1 \cup A_2$.
- The machine $M$ works by simulating $M_1$ and $M_2$.
- **Simulation**: pretend that you are $M$. As you read the input symbols you simulate both $M_1$ and $M_2$, simultaneously.
- To keep track of both simulations, need to remember the state each machine would be in if it had read up to this point in the input.
More on proof idea

- To remember the states of $M_1$ and $M_2$ you need to remember a pair of states.
More on proof idea

- To remember the states of $M_1$ and $M_2$ you need to remember a pair of states.
- If $M_1$ has $k_1$ states and $M_2$ has $k_2$ states then you need $k_1 \times k_2$ states to simulate simultaneously $M_1$ and $M_2$. 
More on proof idea

- To remember the states of $M_1$ and $M_2$ you need to remember a pair of states.
- If $M_1$ has $k_1$ states and $M_2$ has $k_2$ states then you need $k_1 \times k_2$ states to simulate simultaneously $M_1$ and $M_2$.
- Transitions of $M$ goes from pair to pair, updating the state for both $M_1$ and $M_2$. 

The Regular Operations – p.13/19
More on proof idea

- To remember the states of $M_1$ and $M_2$ you need to remember a pair of states.
- If $M_1$ has $k_1$ states and $M_2$ has $k_2$ states then you need $k_1 \times k_2$ states to simulate simultaneously $M_1$ and $M_2$.
- Transitions of $M$ goes from pair to pair, updating the state for both $M_1$ and $M_2$.
- The start state of $M$ is the pair of start states of $M_1$ and $M_2$; the accept states of $M$ is the set of pairs containing an accept state of $M_1$ or $M_2$. 

The Regular Operations – p.13/19
Proof

By construction. Let $M_1$ recognize $A_1$ where

$M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$, and $M_2$ recognize $A_2$ where

$M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$
Proof

By construction. Let $M_1$ recognize $A_1$ where

$M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$, and $M_2$ recognize $A_2$ where

$M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$

Construct $M$ to recognize $A_1 \cup A_2$,

$M = (Q, \Sigma, \delta, q_0, F')$, where:
Construction of $M$

- $Q = \{ (r_1, r_2) \mid r_1 \in Q_1, r_2 \in Q_2 \}$, i.e., $Q = Q_1 \times Q_2$
Construction of $M$

- $Q = \{(r_1, r_2) \mid r_1 \in Q_1, r_2 \in Q_2\}$, i.e., $Q = Q_1 \times Q_2$
- $\Sigma$ is assumed the same for $M_1$ and $M_2$. If $M_1$ and $M_2$ have different alphabets, $\Sigma_1$ and $\Sigma_2$ then $\Sigma = \Sigma_1 \cup \Sigma_2$
Construction of $M$

- $Q = \{(r_1, r_2) \mid r_1 \in Q_1, r_2 \in Q_2\}$, i.e., $Q = Q_1 \times Q_2$
- $\Sigma$ is assumed the same for $M_1$ and $M_2$. If $M_1$ and $M_2$ have different alphabets, $\Sigma_1$ and $\Sigma_2$ then $\Sigma = \Sigma_1 \cup \Sigma_2$
- For each $(r_1, r_2) \in Q$ and $a \in \Sigma$, $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$
Construction of $M$

- $Q = \{(r_1, r_2) \mid r_1 \in Q_1, r_2 \in Q_2\}$, i.e., $Q = Q_1 \times Q_2$
- $\Sigma$ is assumed the same for $M_1$ and $M_2$. If $M_1$ and $M_2$ have different alphabets, $\Sigma_1$ and $\Sigma_2$ then $\Sigma = \Sigma_1 \cup \Sigma_2$
- For each $(r_1, r_2) \in Q$ and $a \in \Sigma$, 
  $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$
- $q_0 = (q_0^1, q_0^2)$
Construction of $M$

- $Q = \{(r_1, r_2) \mid r_1 \in Q_1, r_2 \in Q_2\}$, i.e., $Q = Q_1 \times Q_2$
- $\Sigma$ is assumed the same for $M_1$ and $M_2$. If $M_1$ and $M_2$ have different alphabets, $\Sigma_1$ and $\Sigma_2$ then $\Sigma = \Sigma_1 \cup \Sigma_2$
- For each $(r_1, r_2) \in Q$ and $a \in \Sigma$, 
  $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$
- $q_0 = (q_0^1, q_0^2)$
- $F = \{(r_1, r_2) \mid r_1 \in F_1 \lor r_2 \in F_2\}$, i.e., $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$. Note, this is not the same as $F_1 \times F_2$
Corollary

Class of regular languages is closed under intersection.
Corollary

Class of regular languages is closed under intersection.

Proof: For two regular languages $A$ and $B$, recognized by the automata $M_A$ and $M_B$ the automaton that recognizes the language $A \cap B$ is constructed in the same way as the automaton that recognizes the language $A \cup B$ with the final states defined by $F = \{(r_1, r_2) | (r_1, r_2) \in F_1 \times F_2\}$.
• This construction is fairly simple and thus its correctness is evident from the strategy described by proof idea
Note

- This construction is fairly simple and thus its correctness is evident from the strategy described by proof idea.
- More complicated constructions require additional discussion to prove correctness.
Note

- This construction is fairly simple and thus its correctness is evident from the strategy described by proof idea.
- More complicated constructions require additional discussion to prove correctness.
- A formal correctness proof for a construction of this type usually proceeds by induction. We will illustrate it further.
Theorem 1.26

The class of regular languages is closed under concatenation operation, i.e. if $A_1$ and $A_2$ are regular languages then so is $A_1 \circ A_2$.

Proof idea: As before, we can start with finite automata $M_1$ and $M_2$ recognizing $A_1$ and $A_2$ and construct the automaton $M$ to recognize $A_1 \circ A_2$. 
The class of regular languages is closed under concatenation operation, i.e. if $A_1$ and $A_2$ are regular languages then so is $A_1 \circ A_2$.

Proof idea: As before, we can start with finite automata $M_1$ and $M_2$ recognizing $A_1$ and $A_2$ and construct the automaton $M$ to recognize $A_1 \circ A_2$. 
Ideas for construction of $M$

- Instead of constructing $M$ to accept its input if either $M_1$ or $M_2$ accept, $M$ must accept if its input can be broken into two pieces where $M_1$ accepts first piece and $M_2$ accepts second piece.
Ideas for construction of $M$

- Instead of constructing $M$ to accept its input if either $M_1$ or $M_2$ accept, $M$ must accept if its input can be broken into two pieces where $M_1$ accepts first piece and $M_2$ accepts second piece.

- The problem is that $M$ does not know where to break its input.
Ideas for construction of $M$

- Instead of constructing $M$ to accept its input if either $M_1$ or $M_2$ accept, $M$ must accept if its input can be broken into two pieces where $M_1$ accepts first piece and $M_2$ accepts second piece.
- The problem is that $M$ does not know where to break its input.
- To solve this problem we need to introduce a new technique: the *nondeterminism*. 