Closure Properties of Regular Languages
Closure Properties of Regular Languages

Closure Properties of a set are those operations you can perform on element(s) of the set, where the result of the operation is also an element of the set.

Example: “The set of integers is closed under addition.”
The Complement of a Language

• Given a language $L \subseteq \Sigma^*$, the *complement* of that language (denoted $\overline{L}$) is the language of all strings in $\Sigma^*$ that aren't in $L$.

• Formally:

$$\overline{L} = \Sigma^* - L$$
The Complement of a Language

- Given a language \( L \subseteq \Sigma^* \), the complement of that language (denoted \( \overline{L} \)) is the language of all strings in \( \Sigma^* \) that aren't in \( L \).
- Formally:

\[
\overline{L} = \Sigma^* - L
\]
The Complement of a Language

• Given a language $L \subseteq \Sigma^*$, the complement of that language (denoted $\overline{L}$) is the language of all strings in $\Sigma^*$ that aren't in $L$.

• Formally:

$$\overline{L} = \Sigma^* - L$$
The Complement of a Language

• Given a language \( L \subseteq \Sigma^* \), the *complement* of that language (denoted \( \overline{L} \)) is the language of all strings in \( \Sigma^* \) that aren't in \( L \).

• Formally:

\[
\overline{L} = \Sigma^* - L
\]
Complements of Regular Languages

- As we saw a few minutes ago, a regular language is a language recognized by some DFA (or NFA).

- **Question**: If \( L \) is a regular language, is \( \overline{L} \) necessarily a regular language?

- If the answer is “yes”: if there is a way to construct a DFA for \( L \), then there must be some way to construct a DFA for \( \overline{L} \).

- If the answer is “no”: some language \( L \) can be recognized by some DFA, but \( \overline{L} \) cannot be recognized by any DFA.
input

Computational Device for $L$

Yep!

Nope!
Computational Device for $L$
Computational Device for $L$

Computational Device for $\overline{L}$
Computational Device for $L$

input

Computational Device for $\overline{L}$

input
Complementing Regular Languages

$L = \{ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring } \}$

$\bar{L} = \{ w \in \{a, b\}^* \mid w \text{ does not contain } aa \text{ as a substring } \}$
Closure Properties

• **Theorem:** If $L$ is a regular language, then $\overline{L}$ is also a regular language.

• As a result, we say that the regular languages are *closed under complement*. 

![Diagram showing closure under complement]

Regular languages

All languages
The Union of Two Languages

- If $L_1$ and $L_2$ are languages over the alphabet $\Sigma$, the language $L_1 \cup L_2$ is the language of all strings in at least one of the two languages.
- If $L_1$ and $L_2$ are regular languages, is $L_1 \cup L_2$?
The Union of Two Languages

- If $L_1$ and $L_2$ are languages over the alphabet $\Sigma$, the language $L_1 \cup L_2$ is the language of all strings in at least one of the two languages.
- If $L_1$ and $L_2$ are regular languages, is $L_1 \cup L_2$?
The Union of Two Languages

- If $L_1$ and $L_2$ are languages over the alphabet $\Sigma$, the language $L_1 \cup L_2$ is the language of all strings in at least one of the two languages.
- If $L_1$ and $L_2$ are regular languages, is $L_1 \cup L_2$?
The Union of Two Languages

- If $L_1$ and $L_2$ are languages over the alphabet $\Sigma$, the language $L_1 \cup L_2$ is the language of all strings in at least one of the two languages.
- If $L_1$ and $L_2$ are regular languages, is $L_1 \cup L_2$?
The Union of Two Languages

- If $L_1$ and $L_2$ are languages over the alphabet $\Sigma$, the language $L_1 \cup L_2$ is the language of all strings in at least one of the two languages.

- If $L_1$ and $L_2$ are regular languages, is $L_1 \cup L_2$?
The Intersection of Two Languages

- If $L_1$ and $L_2$ are languages over $\Sigma$, then $L_1 \cap L_2$ is the language of strings in both $L_1$ and $L_2$.

- Question: If $L_1$ and $L_2$ are regular, is $L_1 \cap L_2$ regular as well?
The Intersection of Two Languages

- If $L_1$ and $L_2$ are languages over $\Sigma$, then $L_1 \cap L_2$ is the language of strings in both $L_1$ and $L_2$.
- Question: If $L_1$ and $L_2$ are regular, is $L_1 \cap L_2$ regular as well?
The Intersection of Two Languages

- If $L_1$ and $L_2$ are languages over $\Sigma$, then $L_1 \cap L_2$ is the language of strings in both $L_1$ and $L_2$.

- Question: If $L_1$ and $L_2$ are regular, is $L_1 \cap L_2$ regular as well?
The Intersection of Two Languages

- If $L_1$ and $L_2$ are languages over $\Sigma$, then $L_1 \cap L_2$ is the language of strings in both $L_1$ and $L_2$.

- Question: If $L_1$ and $L_2$ are regular, is $L_1 \cap L_2$ regular as well?
The Intersection of Two Languages

- If $L_1$ and $L_2$ are languages over $\Sigma$, then $L_1 \cap L_2$ is the language of strings in both $L_1$ and $L_2$.
- Question: If $L_1$ and $L_2$ are regular, is $L_1 \cap L_2$ regular as well?

Hey, it's De Morgan's laws!
Concatenation
String Concatenation

- If $w \in \Sigma^*$ and $x \in \Sigma^*$, the **concatenation** of $w$ and $x$, denoted $wx$, is the string formed by tacking all the characters of $x$ onto the end of $w$.

- Example: if $w = \text{quo}$ and $x = \text{kka}$, the concatenation $wx = \text{quokka}$.

- Analogous to the + operator for strings in many programming languages.

- Some facts about concatenation:
  - The empty string $\varepsilon$ is the **identity element** for concatenation:
    \[ w\varepsilon = \varepsilon w = w \]
  - Concatenation is **associative**:
    \[ wxy = w(xy) = (wx)y \]
Concatenation

• The *concatenation* of two languages $L_1$ and $L_2$ over the alphabet $\Sigma$ is the language

$$L_1L_2 = \{ wx \in \Sigma^* \mid w \in L_1 \land x \in L_2 \}$$
Concatenation Example

- Let $\Sigma = \{ a, b, ..., z, A, B, ..., Z \}$ and consider these languages over $\Sigma$:
  - $\textit{Noun} = \{ \text{Puppy, Rainbow, Whale, ...} \}$
  - $\textit{Verb} = \{ \text{Hugs, Juggles, Loves, ...} \}$
  - $\textit{The} = \{ \text{The} \}$
  - The language $\text{TheNounVerbTheNoun}$ is
Concatenation

- The *concatenation* of two languages $L_1$ and $L_2$ over the alphabet $\Sigma$ is the language

$$L_1L_2 = \{ wx \in \Sigma^* \mid w \in L_1 \land x \in L_2 \}$$

- Two views of $L_1L_2$:
  - The set of all strings that can be made by concatenating a string in $L_1$ with a string in $L_2$.
  - The set of strings that can be split into two pieces: a piece from $L_1$ and a piece from $L_2$.

- Conceptually similar to the Cartesian product of two sets, only with strings.
Concatenating Regular Languages

- If $L_1$ and $L_2$ are regular languages, is $L_1L_2$?
- Intuition – can we split a string $w$ into two strings $xy$ such that $x \in L_1$ and $y \in L_2$?
Concatenating Regular Languages

- If $L_1$ and $L_2$ are regular languages, is $L_1L_2$?
- Intuition – can we split a string $w$ into two strings $xy$ such that $x \in L_1$ and $y \in L_2$?

Machine for $L_1$  

Machine for $L_2$
Concatenating Regular Languages

- If $L_1$ and $L_2$ are regular languages, is $L_1L_2$?
- Intuition – can we split a string $w$ into two strings $xy$ such that $x \in L_1$ and $y \in L_2$?

![Machine for $L_1$](image)

- Machine for $L_1$

![Machine for $L_2$](image)

- Machine for $L_2$
Concatenating Regular Languages

- If $L_1$ and $L_2$ are regular languages, is $L_1L_2$?
- Intuition – can we split a string $w$ into two strings $xy$ such that $x \in L_1$ and $y \in L_2$?

Machine for $L_1$

Machine for $L_2$

bookkeeper
Concatenating Regular Languages

- If $L_1$ and $L_2$ are regular languages, is $L_1L_2$?
- Intuition – can we split a string $w$ into two strings $xy$ such that $x \in L_1$ and $y \in L_2$?

Start

Machine for $L_1$

Machine for $L_2$

book

keeper
Concatenating Regular Languages

• If \( L_1 \) and \( L_2 \) are regular languages, is \( L_1L_2 \)?

• Intuition – can we split a string \( w \) into two strings \( xy \) such that \( x \in L_1 \) and \( y \in L_2 \)?

• **Idea**: Run the automaton for \( L_1 \) on \( w \), and whenever \( L_1 \) reaches an accepting state, optionally hand the rest off \( w \) to \( L_2 \).

  • If \( L_2 \) accepts the remainder, then \( L_1 \) accepted the first part and the string is in \( L_1L_2 \).
  
  • If \( L_2 \) rejects the remainder, then the split was incorrect.
Concatenating Regular Languages
Concatenating Regular Languages

Machine for $L_1$
Concatenating Regular Languages

Machine for $L_1$

Machine for $L_2$
Concatenating Regular Languages

Machine for $L_1$

Machine for $L_2$
Concatenating Regular Languages

Machine for $L_1$

Machine for $L_2$
Concatenating Regular Languages

Machine for $L_1$

Machine for $L_2$

Machine for $L_1L_2$
Lots and Lots of Concatenation

• Consider the language \( L = \{ \text{aa, b} \} \)

• \( LL \) is the set of strings formed by concatenating pairs of strings in \( L \).

\[
\{ \text{aaaa, aab, baa, bb} \}
\]

• \( LLL \) is the set of strings formed by concatenating triples of strings in \( L \).

\[
\{ \text{aaaaaa, aaaaab, aabaa, aabb, baaaa, baab, bbaa, bbb} \}
\]

• \( LLLL \) is the set of strings formed by concatenating quadruples of strings in \( L \).

\[
\{ \text{aaaaaaaa, aaaaaab, aaaaaba, aaaaabb, aabaaaa, aabaab, aabbaa, aabbb, baaaaaa, baaaaab, baabaa, baabb, bbaaaa, bbaab, bbbaa, bbbb} \}
\]
Language Exponentiation

- We can define what it means to “exponentiate” a language as follows:

  - $L^0 = \{ \varepsilon \}$
    - The set containing just the empty string.
    - Idea: Any string formed by concatenating zero strings together is the empty string.
  
  - $L^{n+1} = LL^n$
    - Idea: Concatenating $(n+1)$ strings together works by concatenating $n$ strings, then concatenating one more.

- **Question:** Why define $L^0 = \{ \varepsilon \}$?
The Kleene Closure

- An important operation on languages is the **Kleene Closure**, which is defined as
  \[ L^* = \{ w \in \Sigma^* \mid \exists n \in \mathbb{N}. w \in L^n \} \]
- Mathematically:
  \[ w \in L^* \iff \exists n \in \mathbb{N}. w \in L^n \]
- Intuitively, all possible ways of concatenating zero or more strings in \( L \) together, possibly with repetition.
The Kleene Closure

If \( L = \{ a, bb \} \), then \( L^* = \{ \)

\[ \varepsilon, \]

\[ a, bb, \]

\[ aa, aabb, bba, bbbb, \]

\[ aaaa, aabbb, abba, aabbb, bbbaa, bbabb, bbbba, bbbbbbb, \]

\[ ... \]

\} 

Think of \( L^* \) as the set of strings you can make if you have a collection of stamps – one for each string in \( L \) – and you form every possible string that can be made from those stamps.
Reasoning about Infinity

• If $L$ is regular, is $L^*$ necessarily regular?

• ⚠ A Bad Line of Reasoning: ⚠
  
  • $L^0 = \{ \varepsilon \}$ is regular.
  • $L^1 = L$ is regular.
  • $L^2 = LL$ is regular
  • $L^3 = L(LL)$ is regular
  • ...
  
  • Regular languages are closed under union.
  • So the union of all these languages is regular.
Reasoning About the Infinite

- If a series of finite objects all have some property, the “limit” of that process does not necessarily have that property.
- In general, it is not safe to conclude that some property that always holds in the finite case must hold in the infinite case.
  - (This is why calculus is interesting).
Idea: Can we directly convert an NFA for language $L$ to an NFA for language $L^*$?
The Kleene Star

Machine for $L$
The Kleene Star

Machine for $L$
The Kleene Star

Machine for $L$
The Kleene Star

Machine for $L$
The Kleene Star

Machine for $L$
The Kleene Star

Machine for $L$

Machine for $L^*$
The Kleene Star

Question: Why add the new state out front? Why not just make the old start state accepting?
Closure Properties

- **Theorem:** If $L_1$ and $L_2$ are regular languages over an alphabet $\Sigma$, then so are the following languages:
  - $\overline{L_1}$
  - $L_1 \cup L_2$
  - $L_1 \cap L_2$
  - $L_1L_2$
  - $L_1^*$

- These properties are called **closure properties of the regular languages**.