Formal Languages, Automata and Codes

Oleg Gutik

Lecture 13

4.3 Identifying Nonregular Languages

Regular languages can be infinite, as most of our examples have demonstrated. The fact that regular languages are associated with automata that have finite memory, however, imposes some limits on the structure of a regular language. Some narrow restrictions must be obeyed if regularity is to hold. Intuition tells us that a language is regular only if, in processing any string, the information that has to be remembered at any stage is strictly limited. This is true, but has to be shown precisely to be used in any meaningful way. There are several ways in which this can be done.

Using the Pigeonhole Principle

The term "pigeonhole principle" is used by mathematicians to refer to the following simple observation. If we put n objects into m boxes (pigeonholes), and if $n > m$ then at least one box must have more than one item in it. This is such an obvious fact that it is surprising how many deep results can be obtained from it.

Regular languages can be infinite, as most of our examples have demonstrated. The fact that regular languages are associated with automata that have finite memory, however, imposes some limits on the structure of a regular language. Some narrow restrictions must be obeyed if regularity is to hold. Intuition tells us that a language is regular only if, in processing any string, the information that has to be remembered at any stage is strictly limited. This is true, but has to be shown precisely to be used in any meaningful way. There are several ways in which this can be done.

Using the Pigeonhole Principle

The term "pigeonhole principle" is used by mathematicians to refer to the following simple observation. If we put n objects into m boxes (pigeonholes), and if $n > m$ then at least one box must have more than one item in it. This is such an obvious fact that it is surprising how many deep results can be obtained from it.

Regular languages can be infinite, as most of our examples have demonstrated. The fact that regular languages are associated with automata that have finite memory, however, imposes some limits on the structure of a regular language. Some narrow restrictions must be obeyed if regularity is to hold. Intuition tells us that a language is regular only if, in processing any string, the information that has to be remembered at any stage is strictly limited. This is true, but has to be shown precisely to be used in any meaningful way. There are several ways in which this can be done.

Using the Pigeonhole Principle

The term "pigeonhole principle" is used by mathematicians to refer to the following simple observation. If we put n objects into m boxes (pigeonholes), and if $n > m$ then at least one box must have more than one item in it. This is such an obvious fact that it is surprising how many deep results can be obtained from it.

4.3 Identifying Nonregular Languages

Regular languages can be infinite, as most of our examples have demonstrated. The fact that regular languages are associated with automata that have finite memory, however, imposes some limits on the structure of a regular language. Some narrow restrictions must be obeyed if regularity is to hold. Intuition tells us that a language is regular only if, in processing any string, the information that has to be remembered at any stage is strictly limited. This is true, but has to be shown precisely to be used in any meaningful way. There are several ways in which this can be done.

Using the Pigeonhole Principle

The term "pigeonhole principle" is used by mathematicians to refer to the following simple observation. If we put n objects into m boxes (pigeonholes), and if $n > m$ then at least one box must have more than one item in it. This is such an obvious fact that it is surprising how many deep results can be obtained from it.

Regular languages can be infinite, as most of our examples have demonstrated. The fact that regular languages are associated with automata that have finite memory, however, imposes some limits on the structure of a regular language. Some narrow restrictions must be obeyed if regularity is to hold. Intuition tells us that a language is regular only if, in processing any string, the information that has to be remembered at any stage is strictly limited. This is true, but has to be shown precisely to be used in any meaningful way. There are several ways in which this can be done.

Using the Pigeonhole Principle

The term "pigeonhole principle" is used by mathematicians to refer to the following simple observation. If we put n objects into m boxes (pigeonholes), and if $n > m$ then at least one box must have more than one item in it. This is such an obvious fact that it is surprising how many deep results can be obtained from it.

Regular languages can be infinite, as most of our examples have demonstrated. The fact that regular languages are associated with automata that have finite memory, however, imposes some limits on the structure of a regular language. Some narrow restrictions must be obeyed if regularity is to hold. Intuition tells us that a language is regular only if, in processing any string, the information that has to be remembered at any stage is strictly limited. This is true, but has to be shown precisely to be used in any meaningful way. There are several ways in which this can be done.

Using the Pigeonhole Principle

The term "pigeonhole principle" is used by mathematicians to refer to the following simple observation. If we put n objects into m boxes (pigeonholes), and if $n > m$ then at least one box must have more than one item in it. This is such an obvious fact that it is surprising how many deep results can be obtained from it.

Regular languages can be infinite, as most of our examples have demonstrated. The fact that regular languages are associated with automata that have finite memory, however, imposes some limits on the structure of a regular language. Some narrow restrictions must be obeyed if regularity is to hold. Intuition tells us that a language is regular only if, in processing any string, the information that has to be remembered at any stage is strictly limited. This is true, but has to be shown precisely to be used in any meaningful way. There are several ways in which this can be done.

Using the Pigeonhole Principle

The term "pigeonhole principle" is used by mathematicians to refer to the following simple observation. If we put n objects into m boxes (pigeonholes), and if $n > m$ then at least one box must have more than one item in it. This is such an obvious fact that it is surprising how many deep results can be obtained from it.

Regular languages can be infinite, as most of our examples have demonstrated. The fact that regular languages are associated with automata that have finite memory, however, imposes some limits on the structure of a regular language. Some narrow restrictions must be obeyed if regularity is to hold. Intuition tells us that a language is regular only if, in processing any string, the information that has to be remembered at any stage is strictly limited. This is true, but has to be shown precisely to be used in any meaningful way. There are several ways in which this can be done.

Using the Pigeonhole Principle

The term "pigeonhole principle" is used by mathematicians to refer to the following simple observation. If we put n objects into m boxes (pigeonholes), and if $n > m$ then at least one box must have more than one item in it. This is such an obvious fact that it is surprising how many deep results can be obtained from it.

Regular languages can be infinite, as most of our examples have demonstrated. The fact that regular languages are associated with automata that have finite memory, however, imposes some limits on the structure of a regular language. Some narrow restrictions must be obeyed if regularity is to hold. Intuition tells us that a language is regular only if, in processing any string, the information that has to be remembered at any stage is strictly limited. This is true, but has to be shown precisely to be used in any meaningful way. There are several ways in which this can be done.

Using the Pigeonhole Principle

The term "pigeonhole principle" is used by mathematicians to refer to the following simple observation. If we put n objects into m boxes (pigeonholes), and if $n > m$ then at least one box must have more than one item in it. This is such an obvious fact that it is surprising how many deep results can be obtained from it.

Regular languages can be infinite, as most of our examples have demonstrated. The fact that regular languages are associated with automata that have finite memory, however, imposes some limits on the structure of a regular language. Some narrow restrictions must be obeyed if regularity is to hold. Intuition tells us that a language is regular only if, in processing any string, the information that has to be remembered at any stage is strictly limited. This is true, but has to be shown precisely to be used in any meaningful way. There are several ways in which this can be done.

Using the Pigeonhole Principle

The term "pigeonhole principle" is used by mathematicians to refer to the following simple observation. If we put n objects into m boxes (pigeonholes), and if $n > m$ then at least one box must have more than one item in it. This is such an obvious fact that it is surprising how many deep results can be obtained from it.

Regular languages can be infinite, as most of our examples have demonstrated. The fact that regular languages are associated with automata that have finite memory, however, imposes some limits on the structure of a regular language. Some narrow restrictions must be obeyed if regularity is to hold. Intuition tells us that a language is regular only if, in processing any string, the information that has to be remembered at any stage is strictly limited. This is true, but has to be shown precisely to be used in any meaningful way. There are several ways in which this can be done.

Using the Pigeonhole Principle

The term "pigeonhole principle" is used by mathematicians to refer to the following simple observation. If we put n objects into m boxes (pigeonholes), and if $n > m$ then at least one box must have more than one item in it. This is such an obvious fact that it is surprising how many deep results can be obtained from it.

Regular languages can be infinite, as most of our examples have demonstrated. The fact that regular languages are associated with automata that have finite memory, however, imposes some limits on the structure of a regular language. Some narrow restrictions must be obeyed if regularity is to hold. Intuition tells us that a language is regular only if, in processing any string, the information that has to be remembered at any stage is strictly limited. This is true, but has to be shown precisely to be used in any meaningful way. There are several ways in which this can be done.

Using the Pigeonhole Principle

The term "pigeonhole principle" is used by mathematicians to refer to the following simple observation. If we put n objects into m boxes (pigeonholes), and if $n > m$ then at least one box must have more than one item in it. This is such an obvious fact that it is surprising how many deep results can be obtained from it.

Regular languages can be infinite, as most of our examples have demonstrated. The fact that regular languages are associated with automata that have finite memory, however, imposes some limits on the structure of a regular language. Some narrow restrictions must be obeyed if regularity is to hold. Intuition tells us that a language is regular only if, in processing any string, the information that has to be remembered at any stage is strictly limited. This is true, but has to be shown precisely to be used in any meaningful way. There are several ways in which this can be done.

Using the Pigeonhole Principle

The term "pigeonhole principle" is used by mathematicians to refer to the following simple observation. If we put n objects into m boxes (pigeonholes), and if $n > m$ then at least one box must have more than one item in it. This is such an obvious fact that it is surprising how many deep results can be obtained from it.

Regular languages can be infinite, as most of our examples have demonstrated. The fact that regular languages are associated with automata that have finite memory, however, imposes some limits on the structure of a regular language. Some narrow restrictions must be obeyed if regularity is to hold. Intuition tells us that a language is regular only if, in processing any string, the information that has to be remembered at any stage is strictly limited. This is true, but has to be shown precisely to be used in any meaningful way. There are several ways in which this can be done.

Using the Pigeonhole Principle

The term "pigeonhole principle" is used by mathematicians to refer to the following simple observation. If we put n objects into m boxes (pigeonholes), and if $n > m$ then at least one box must have more than one item in it. This is such an obvious fact that it is surprising how many deep results can be obtained from it.

4.3 Identifying Nonregular Languages

Regular languages can be infinite, as most of our examples have demonstrated. The fact that regular languages are associated with automata that have finite memory, however, imposes some limits on the structure of a regular language. Some narrow restrictions must be obeyed if regularity is to hold. Intuition tells us that a language is regular only if, in processing any string, the information that has to be remembered at any stage is strictly limited. This is true, but has to be shown precisely to be used in any meaningful way. There are several ways in which this can be done.

Using the Pigeonhole Principle

The term "pigeonhole principle" is used by mathematicians to refer to the following simple observation. If we put n objects into m boxes (pigeonholes), and if $n > m$ then at least one box must have more than one item in it. This is such an obvious fact that it is surprising how many deep results can be obtained from it.

4.3 Identifying Nonregular Languages

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned} \delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f. \end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned} \delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f. \end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned} \delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f. \end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned} \delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f. \end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned} \delta^*(q_0, a^m b^n) = \delta^*(\delta^*(q_0, a^m), b^n) = \\ = \delta^*(q, b^n) = \\ = q_f. \end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned} \delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f. \end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned} \delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f. \end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned} \delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f. \end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned} \delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f. \end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned} \delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f. \end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned} \delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f. \end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned} \delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f. \end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned}\delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f.\end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned} \delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f. \end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned} \delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f. \end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned}\delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f.\end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned} \delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f. \end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned} \delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f. \end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned} \delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f. \end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

Example 4.6

Is the language $L = \{a^n b^n : n \geqslant 0\}$ regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some DFA $M = (Q, \{a, b\}, \delta, q_0, F)$ exists for it. Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$. Since there are an unlimited number of i's, but only a finite number of states in M, the Pigeonhole Principle tells us that there must be some state, say q, such that

$$\delta^*(q_0, a^n) = q$$

and

$$\delta^*(q_0, a^m) = q,$$

with $n \neq m$. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we can conclude that

$$\begin{aligned} \delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) = \\ &= \delta^*(q, b^n) = \\ &= q_f. \end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if $n = m$, and leads us to conclude that L cannot be regular.

In this argument, the Pigeonhole Principle is just a way of stating unambiguously what we mean when we say that a finite automaton has a limited memory. To accept all $a^n b^n$, an automaton would have to differentiate between all prefixes a^n and a^m. But since there are only a finite number of internal states with which to do this, there are some n and m for which the distinction cannot be made.

In order to use this type of argument in a variety of situations, it is convenient to codify it as a general theorem. There are several ways to do this; the one we give here is perhaps the most famous one.

In this argument, the Pigeonhole Principle is just a way of stating unambiguously what we mean when we say that a finite automaton has a limited memory. To accept all $a^n b^n$, an automaton would have to differentiate between all prefixes a^n and a^m. But since there are only a finite number of internal states with which to do this, there are some n and m for which the distinction cannot be made.

In order to use this type of argument in a variety of situations, it is convenient to codify it as a general theorem. There are several ways to do this; the one we give here is perhaps the most famous one.

In this argument, the Pigeonhole Principle is just a way of stating unambiguously what we mean when we say that a finite automaton has a limited memory. To accept all $a^n b^n$, an automaton would have to differentiate between all prefixes a^n and a^m. But since there are only a finite number of internal states with which to do this, there are some n and m for which the distinction cannot be made.

In order to use this type of argument in a variety of situations, it is convenient to codify it as a general theorem. There are several ways to do this; the one we give here is perhaps the most famous one.

In this argument, the Pigeonhole Principle is just a way of stating unambiguously what we mean when we say that a finite automaton has a limited memory. To accept all $a^n b^n$, an automaton would have to differentiate between all prefixes a^n and a^m. But since there are only a finite number of internal states with which to do this, there are some n and m for which the distinction cannot be made.

In order to use this type of argument in a variety of situations, it is convenient to codify it as a general theorem. There are several ways to do this; the one we give here is perhaps the most famous one.

In this argument, the Pigeonhole Principle is just a way of stating unambiguously what we mean when we say that a finite automaton has a limited memory. To accept all $a^n b^n$, an automaton would have to differentiate between all prefixes a^n and a^m. But since there are only a finite number of internal states with which to do this, there are some n and m for which the distinction cannot be made.

In order to use this type of argument in a variety of situations, it is convenient to codify it as a general theorem. There are several ways to do this; the one we give here is perhaps the most famous one.

In this argument, the Pigeonhole Principle is just a way of stating unambiguously what we mean when we say that a finite automaton has a limited memory. To accept all $a^n b^n$, an automaton would have to differentiate between all prefixes a^n and a^m. But since there are only a finite number of internal states with which to do this, there are some n and m for which the distinction cannot be made.

In order to use this type of argument in a variety of situations, it is convenient to codify it as a general theorem. There are several ways to do this; the one we give here is perhaps the most famous one.

In this argument, the Pigeonhole Principle is just a way of stating unambiguously what we mean when we say that a finite automaton has a limited memory. To accept all $a^n b^n$, an automaton would have to differentiate between all prefixes a^n and a^m. But since there are only a finite number of internal states with which to do this, there are some n and m for which the distinction cannot be made.

In order to use this type of argument in a variety of situations, it is convenient to codify it as a general theorem. There are several ways to do this; the one we give here is perhaps the most famous one.

In this argument, the Pigeonhole Principle is just a way of stating unambiguously what we mean when we say that a finite automaton has a limited memory. To accept all $a^n b^n$, an automaton would have to differentiate between all prefixes a^n and a^m. But since there are only a finite number of internal states with which to do this, there are some n and m for which the distinction cannot be made.

In order to use this type of argument in a variety of situations, it is convenient to codify it as a general theorem. There are several ways to do this; the one we give here is perhaps the most famous one.

In this argument, the Pigeonhole Principle is just a way of stating unambiguously what we mean when we say that a finite automaton has a limited memory. To accept all $a^n b^n$, an automaton would have to differentiate between all prefixes a^n and a^m. But since there are only a finite number of internal states with which to do this, there are some n and m for which the distinction cannot be made.

In order to use this type of argument in a variety of situations, it is convenient to codify it as a general theorem. There are several ways to do this; the one we give here is perhaps the most famous one.

In this argument, the Pigeonhole Principle is just a way of stating unambiguously what we mean when we say that a finite automaton has a limited memory. To accept all $a^n b^n$, an automaton would have to differentiate between all prefixes a^n and a^m. But since there are only a finite number of internal states with which to do this, there are some n and m for which the distinction cannot be made.

In order to use this type of argument in a variety of situations, it is convenient to codify it as a general theorem. There are several ways to do this; the one we give here is perhaps the most famous one.

In this argument, the Pigeonhole Principle is just a way of stating unambiguously what we mean when we say that a finite automaton has a limited memory. To accept all $a^n b^n$, an automaton would have to differentiate between all prefixes a^n and a^m. But since there are only a finite number of internal states with which to do this, there are some n and m for which the distinction cannot be made.

In order to use this type of argument in a variety of situations, it is convenient to codify it as a general theorem. There are several ways to do this; the one we give here is perhaps the most famous one.

4.3 Identifying Nonregular Languages

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

4.3 Identifying Nonregular Languages

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string $w_1 v w_2$ is in the language, so must be the strings $w_1 w_2$, $w_1 v v w_2$, $w_1 v v v w_2$, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

4.3 Identifying Nonregular Languages

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string $w_1 v w_2$ is in the language, so must be the strings $w_1 w_2$, $w_1 vv w_2$, $w_1 vvv w_2$, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

4.3 Identifying Nonregular Languages

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

4.3 Identifying Nonregular Languages

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

4.3 Identifying Nonregular Languages

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

4.3 Identifying Nonregular Languages

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

4.3 Identifying Nonregular Languages

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

4.3 Identifying Nonregular Languages

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

Here is what we know about transition graphs for regular languages:

- If the transition graph has no cycles, the language is finite and therefore regular.
- If the transition graph has a cycle with a nonempty label, the language is infinite. Conversely, every infinite regular language has a DFA with such a cycle.
- If there is a cycle, this cycle can either be skipped or repeated an arbitrary number of times. So, if the cycle has label v and if the string w_1vw_2 is in the language, so must be the strings w_1w_2, w_1vvw_2, w_1vvvw_2, and so on.
- We do not know where in the DFA this cycle is, but if the DFA has m states, the cycle must be entered by the time m symbols have been read.

If, for some language L, there is even one string w that does not have this property, L cannot be regular. This observation can be formally stated as a theorem called the Pumping Lemma.

The Pumping Lemma

The following result, known as the Pumping Lemma for regular languages, uses the Pigeonhole Principle in another form. The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, …, q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, …, q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \qquad (1)$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \qquad (1)$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \qquad (1)$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

4.3 Identifying Nonregular Languages

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \qquad (1)$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, …, q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \qquad (1)$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, …, q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

Theorem 4.8

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geqslant m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leqslant m,$$

and

$$|y| \geqslant 1,$$

such that

$$w_i = xy^i z, \tag{1}$$

is also in L for all $i = 0, 1, 2, \ldots$.

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another string in L. We say that the middle string is "pumped," hence the term Pumping Lemma for this result.

Proof. If L is regular the there exists a DFA that recognizes it. Let such a DFA have states labeled q_0, q_1, q_2, ..., q_n. Now take a string w in L such that $|w| \geqslant m = n + 1$. Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0, q_i, q_j, \ldots, q_f.$$

4.3 Identifying Nonregular Languages

Since this sequence has exactly $|w|+1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$

$$\delta^*(q_r, y) = q_r,$$

$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n+1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$

$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ■

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

4.3 Identifying Nonregular Languages

Since this sequence has exactly $|w| + 1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$

$$\delta^*(q_r, y) = q_r,$$

$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n + 1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$

$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ■

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

Since this sequence has exactly $|w| + 1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$

$$\delta^*(q_r, y) = q_r,$$

$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n + 1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$

$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ■

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

4.3 Identifying Nonregular Languages

Since this sequence has exactly $|w| + 1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$

$$\delta^*(q_r, y) = q_r,$$

$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n + 1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$

$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ■

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

4.3 Identifying Nonregular Languages

Since this sequence has exactly $|w|+1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$

$$\delta^*(q_r, y) = q_r,$$

$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n+1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$

$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ■

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

Since this sequence has exactly $|w| + 1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$

$$\delta^*(q_r, y) = q_r,$$

$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n + 1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$

$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ∎

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

4.3 Identifying Nonregular Languages

Since this sequence has exactly $|w| + 1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$

$$\delta^*(q_r, y) = q_r,$$

$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n + 1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$

$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ∎

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

4.3 Identifying Nonregular Languages

Since this sequence has exactly $|w| + 1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$

$$\delta^*(q_r, y) = q_r,$$

$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n + 1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$

$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ∎

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

Since this sequence has exactly $|w| + 1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$
$$\delta^*(q_r, y) = q_r,$$
$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n + 1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$
$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ■

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

4.3 Identifying Nonregular Languages

Since this sequence has exactly $|w|+1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$

$$\delta^*(q_r, y) = q_r,$$

$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n+1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$

$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ■

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

Since this sequence has exactly $|w| + 1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$
$$\delta^*(q_r, y) = q_r,$$
$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n + 1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$
$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ■

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

Since this sequence has exactly $|w| + 1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$
$$\delta^*(q_r, y) = q_r,$$
$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n + 1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$
$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ∎

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

Since this sequence has exactly $|w| + 1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$

$$\delta^*(q_r, y) = q_r,$$

$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n + 1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2 z) = q_f,$$

$$\delta^*(q_r, xy^3 z) = q_f,$$

and so on, completing the proof of the theorem. ■

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

Since this sequence has exactly $|w| + 1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$
$$\delta^*(q_r, y) = q_r,$$
$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n + 1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$
$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ■

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

Since this sequence has exactly $|w| + 1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$

$$\delta^*(q_r, y) = q_r,$$

$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n + 1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$

$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ■

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

Since this sequence has exactly $|w| + 1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$
$$\delta^*(q_r, y) = q_r,$$
$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n + 1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2 z) = q_f,$$
$$\delta^*(q_r, xy^3 z) = q_f,$$

and so on, completing the proof of the theorem. ■

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

Since this sequence has exactly $|w| + 1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \dots, q_r, \dots, q_r, \dots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$

$$\delta^*(q_r, y) = q_r,$$

$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n + 1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$

$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ■

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

4.3 Identifying Nonregular Languages

Since this sequence has exactly $|w|+1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$

$$\delta^*(q_r, y) = q_r,$$

$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n+1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$

$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ■

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

4.3 Identifying Nonregular Languages

Since this sequence has exactly $|w| + 1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \dots, q_r, \dots, q_r, \dots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$

$$\delta^*(q_r, y) = q_r,$$

$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n + 1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$

$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ■

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

Since this sequence has exactly $|w|+1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$

$$\delta^*(q_r, y) = q_r,$$

$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n+1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$

$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ■

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

Since this sequence has exactly $|w| + 1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$

$$\delta^*(q_r, y) = q_r,$$

$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n + 1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$

$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ∎

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

Since this sequence has exactly $|w|+1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicating there must be substrings x, y, z of w such that

$$\delta^*(q_0, x) = q_r,$$

$$\delta^*(q_r, y) = q_r,$$

$$\delta^*(q_r, z) = q_f,$$

with $|xy| \leqslant n+1 = m$ and $|y| \geqslant 1$. From this it immediately follows that

$$\delta^*(q_0, xz) = q_f,$$

as well as

$$\delta^*(q_0, xy^2z) = q_f,$$

$$\delta^*(q_r, xy^3z) = q_f,$$

and so on, completing the proof of the theorem. ∎

We have given the Pumping Lemma only for infinite languages. Finite languages, although always regular, cannot be pumped because pumping automatically creates an infinite set. The theorem does hold for finite languages, but it is vacuous. The m in the Pumping Lemma is to be taken larger than the longest string, so that no string can be pumped.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

4.3 Identifying Nonregular Languages

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

4.3 Identifying Nonregular Languages

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

4.3 Identifying Nonregular Languages

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular.
Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

The Pumping Lemma, like the pigeonhole argument in Example 4.6, is used to show that certain languages are not regular. The demonstration is always by contradiction. There is nothing in the Pumping Lemma, as we have stated it here, that can be used for proving that a language is regular. Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 that allows us to conclude from this that the language is regular.

Example 4.7

Use the Pumping Lemma to show that $L = \{a^n b^n : n \geqslant 0\}$ is not regular. Assume that L is regular, so that the Pumping Lemma must hold. We do not know the value of m, but whatever it is, we can always choose $n = m$. Therefore, the substring y must consist entirely of a's. Suppose $|y| = k$. Then the string obtained by using $i = 0$ in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is

$$w_0 = a^{m-k} b^m$$

and is clearly not in L. This contradicts the Pumping Lemma and thereby indicates that the assumption that L is regular must be false.

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m.
3. The opponent chooses the decomposition xyz,
4. We try to pick i in such a way that the pumped string w_i,

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m.
3. The opponent chooses the decomposition xyz,
4. We try to pick i in such a way that the pumped string w_i,

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m.
3. The opponent chooses the decomposition xyz,
4. We try to pick i in such a way that the pumped string w_i,

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L$ and $|w| \geq m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leq m$, $|y| \geq 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.
4. We try to pick i in such a way that the pumped string w_i, defined in Equation (1)
$$w_i = xy^iz, \tag{1}$$
is not in L. If we can do so, we win the game.

4.3 Identifying Nonregular Languages

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L$ and $|w| \geq m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leq m$, $|y| \geq 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.
4. We try to pick i in such a way that the pumped string w_i, defined in Equation (1)
$$w_i = xy^iz, \tag{1}$$
is not in L. If we can do so, we win the game.

4.3 Identifying Nonregular Languages

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m.
3. The opponent chooses the decomposition xyz,
4. We try to pick i in such a way that the pumped string w_i,

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L$ and $|w| \geq m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leq m$, $|y| \geq 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.
4. We try to pick i in such a way that the pumped string w_i, defined in Equation (1)
$$w_i = xy^iz, \qquad (1)$$
is not in L. If we can do so, we win the game.

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m.
3. The opponent chooses the decomposition xyz,
4. We try to pick i in such a way that the pumped string w_i,

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L$ and $|w| \geq m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leq m$, $|y| \geq 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.
4. We try to pick i in such a way that the pumped string w_i, defined in Equation (1)
$$w_i = xy^iz, \qquad (1)$$
is not in L. If we can do so, we win the game.

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m.
3. The opponent chooses the decomposition xyz,
4. We try to pick i in such a way that the pumped string w_i,

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L$ and $|w| \geq m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leq m$, $|y| \geq 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.
4. We try to pick i in such a way that the pumped string w_i, defined in Equation (1)
$$w_i = xy^iz, \tag{1}$$
is not in L. If we can do so, we win the game.

4.3 Identifying Nonregular Languages

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L$ and $|w| \geq m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leq m$, $|y| \geq 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.
4. We try to pick i in such a way that the pumped string w_i, defined in Equation (1)
$$w_i = xy^iz, \qquad (1)$$
is not in L. If we can do so, we win the game.

4.3 Identifying Nonregular Languages

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m.
3. The opponent chooses the decomposition xyz,
4. We try to pick i in such a way that the pumped string w_i,

4.3 Identifying Nonregular Languages

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m.
3. The opponent chooses the decomposition xyz,
4. We try to pick i in such a way that the pumped string w_i,

4.3 Identifying Nonregular Languages

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L$ and $|w| \geqslant m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leqslant m$, $|y| \geqslant 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.
4. We try to pick i in such a way that the pumped string w_i, defined in Equation (1)
$$w_i = xy^i z, \tag{1}$$
is not in L. If we can do so, we win the game.

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L$ and $|w| \geqslant m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leqslant m$, $|y| \geqslant 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.
4. We try to pick i in such a way that the pumped string w_i, defined in Equation (1)
$$w_i = xy^i z, \tag{1}$$
is not in L. If we can do so, we win the game.

4.3 Identifying Nonregular Languages

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L$ and $|w| \geqslant m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leqslant m$, $|y| \geqslant 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.
4. We try to pick i in such a way that the pumped string w_i, defined in Equation (1)
$$w_i = xy^i z, \tag{1}$$
is not in L. If we can do so, we win the game.

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L$ and $|w| \geqslant m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leqslant m$, $|y| \geqslant 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.
4. We try to pick i in such a way that the pumped string w_i, defined in Equation (1)

$$w_i = xy^iz, \tag{1}$$

is not in L. If we can do so, we win the game.

4.3 Identifying Nonregular Languages

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L$ and $|w| \geqslant m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leqslant m$, $|y| \geqslant 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.
4. We try to pick i in such a way that the pumped string w_i, defined in Equation (1)
$$w_i = xy^iz, \tag{1}$$
is not in L. If we can do so, we win the game.

4.3 Identifying Nonregular Languages

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L$ and $|w| \geqslant m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leqslant m$, $|y| \geqslant 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.
4. We try to pick i in such a way that the pumped string w_i, defined in Equation (1)
$$w_i = xy^iz, \tag{1}$$
is not in L. If we can do so, we win the game.

4.3 Identifying Nonregular Languages

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L$ and $|w| \geqslant m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leqslant m$, $|y| \geqslant 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.
4. We try to pick i in such a way that the pumped string w_i, defined in Equation (1)
$$w_i = xy^iz, \tag{1}$$
is not in L. If we can do so, we win the game.

4.3 Identifying Nonregular Languages

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L$ and $|w| \geqslant m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leqslant m$, $|y| \geqslant 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.
4. We try to pick i in such a way that the pumped string w_i, defined in Equation (1)
$$w_i = xy^i z, \tag{1}$$
is not in L. If we can do so, we win the game.

4.3 Identifying Nonregular Languages

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L$ and $|w| \geqslant m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leqslant m$, $|y| \geqslant 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.
4. We try to pick i in such a way that the pumped string w_i, defined in Equation (1)
$$w_i = xy^i z, \qquad (1)$$
is not in L. If we can do so, we win the game.

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L$ and $|w| \geqslant m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leqslant m$, $|y| \geqslant 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.
4. We try to pick i in such a way that the pumped string w_i, defined in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is not in L. If we can do so, we win the game.

4.3 Identifying Nonregular Languages

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L$ and $|w| \geqslant m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leqslant m$, $|y| \geqslant 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.
4. We try to pick i in such a way that the pumped string w_i, defined in Equation (1)

$$w_i = xy^i z, \tag{1}$$

is not in L. If we can do so, we win the game.

In applying the Pumping Lemma, we must keep in mind what the theorem says. We are guaranteed the existence of an m as well as the decomposition xyz, but we do not know what they are. We cannot claim that we have reached a contradiction just because the Pumping Lemma is violated for some specific values of m or xyz. On the other hand, the Pumping Lemma holds for every $w \in L$ and every i. Therefore, if the Pumping Lemma is violated even for one w or i, then the language cannot be regular.

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the Pumping Lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L$ and $|w| \geqslant m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leqslant m$, $|y| \geqslant 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.
4. We try to pick i in such a way that the pumped string w_i, defined in Equation (1)
$$w_i = xy^i z, \tag{1}$$
is not in L. If we can do so, we win the game.

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the Pumping Lemma on our next move.

Example 4.8

Show that $L = \{ww^R : w \in \Sigma^*\}$ is not regular.
Whatever m the opponent picks on Step 1, we can always choose a w as shown in the Figure.

Because of this choice, and the requirement that $|xy| \leqslant m$, the opponent is restricted in Step 3 to choosing a string y that consists entirely of a's. In Step 4, we use $i = 0$. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R. Therefore, L is not regular.

4.3 Identifying Nonregular Languages

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the Pumping Lemma on our next move.

Example 4.8

Show that $L = \{ww^R : w \in \Sigma^*\}$ is not regular.

Whatever m the opponent picks on Step 1, we can always choose a w as shown in the Figure.

Because of this choice, and the requirement that $|xy| \leqslant m$, the opponent is restricted in Step 3 to choosing a string y that consists entirely of a's. In Step 4, we use $i = 0$. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R. Therefore, L is not regular.

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the Pumping Lemma on our next move.

Example 4.8

Show that $L = \{ww^R : w \in \Sigma^*\}$ is not regular.
Whatever m the opponent picks on Step 1, we can always choose a w as shown in the Figure.

Because of this choice, and the requirement that $|xy| \leqslant m$, the opponent is restricted in Step 3 to choosing a string y that consists entirely of a's. In Step 4, we use $i = 0$. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R. Therefore, L is not regular.

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the Pumping Lemma on our next move.

Example 4.8

Show that $L = \{ww^R : w \in \Sigma^*\}$ is not regular.
Whatever m the opponent picks on Step 1, we can always choose a w as shown in the Figure.

$$\overbrace{aa\ldots a\ldots\ldots a\ldots a}^{a^m}\overbrace{bb\ldots\ldots\ldots\ldots b}^{b^m}\overbrace{b\ldots\ldots\ldots\ldots\ldots b}^{b^m}\overbrace{aa\ldots\ldots\ldots\ldots a}^{a^m}$$

$$x \qquad y \qquad\qquad\qquad\qquad\qquad\qquad z$$

Because of this choice, and the requirement that $|xy| \leqslant m$, the opponent is restricted in Step 3 to choosing a string y that consists entirely of a's. In Step 4, we use $i = 0$. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R. Therefore, L is not regular.

4.3 Identifying Nonregular Languages

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the Pumping Lemma on our next move.

Example 4.8

Show that $L = \{ww^R : w \in \Sigma^*\}$ is not regular.

Whatever m the opponent picks on Step 1, we can always choose a w as shown in the Figure.

Because of this choice, and the requirement that $|xy| \leqslant m$, the opponent is restricted in Step 3 to choosing a string y that consists entirely of a's. In Step 4, we use $i = 0$. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R. Therefore, L is not regular.

4.3 Identifying Nonregular Languages

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the Pumping Lemma on our next move.

Example 4.8

Show that $L = \{ww^R : w \in \Sigma^*\}$ is not regular.

Whatever m the opponent picks on Step 1, we can always choose a w as shown in the Figure.

Because of this choice, and the requirement that $|xy| \leqslant m$, the opponent is restricted in Step 3 to choosing a string y that consists entirely of a's. In Step 4, we use $i = 0$. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R. Therefore, L is not regular.

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the Pumping Lemma on our next move.

Example 4.8

Show that $L = \{ww^R : w \in \Sigma^*\}$ is not regular.
Whatever m the opponent picks on Step 1, we can always choose a w as shown in the Figure.

Because of this choice, and the requirement that $|xy| \leqslant m$, the opponent is restricted in Step 3 to choosing a string y that consists entirely of a's. In Step 4, we use $i = 0$. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R. Therefore, L is not regular.

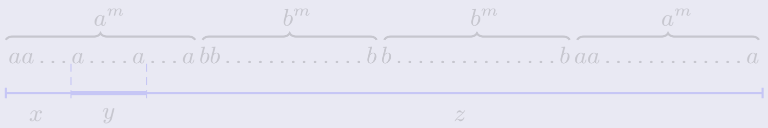
4.3 Identifying Nonregular Languages

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the Pumping Lemma on our next move.

Example 4.8

Show that $L = \{ww^R \colon w \in \Sigma^*\}$ is not regular.

Whatever m the opponent picks on Step 1, we can always choose a w as shown in the Figure.



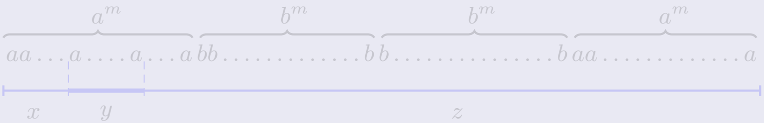
Because of this choice, and the requirement that $|xy| \leqslant m$, the opponent is restricted in Step 3 to choosing a string y that consists entirely of a's. In Step 4, we use $i = 0$. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R. Therefore, L is not regular.

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the Pumping Lemma on our next move.

Example 4.8

Show that $L = \{ww^R : w \in \Sigma^*\}$ is not regular.

Whatever m the opponent picks on Step 1, we can always choose a w as shown in the Figure.



Because of this choice, and the requirement that $|xy| \leqslant m$, the opponent is restricted in Step 3 to choosing a string y that consists entirely of a's. In Step 4, we use $i = 0$. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R. Therefore, L is not regular.

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the Pumping Lemma on our next move.

Example 4.8

Show that $L = \{ww^R : w \in \Sigma^*\}$ is not regular.

Whatever m the opponent picks on Step 1, we can always choose a w as shown in the Figure.

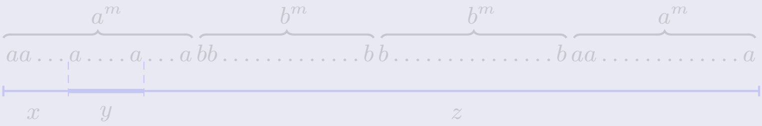
Because of this choice, and the requirement that $|xy| \leqslant m$, the opponent is restricted in Step 3 to choosing a string y that consists entirely of a's. In Step 4, we use $i = 0$. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R. Therefore, L is not regular.

4.3 Identifying Nonregular Languages

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the Pumping Lemma on our next move.

Example 4.8

Show that $L = \left\{ww^R : w \in \Sigma^*\right\}$ is not regular.
Whatever m the opponent picks on Step 1, we can always choose a w as shown in the Figure.



Because of this choice, and the requirement that $|xy| \leqslant m$, the opponent is restricted in Step 3 to choosing a string y that consists entirely of a's. In Step 4, we use $i = 0$. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R. Therefore, L is not regular.

4.3 Identifying Nonregular Languages

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the Pumping Lemma on our next move.

Example 4.8

Show that $L = \{ww^R : w \in \Sigma^*\}$ is not regular.
Whatever m the opponent picks on Step 1, we can always choose a w as shown in the Figure.

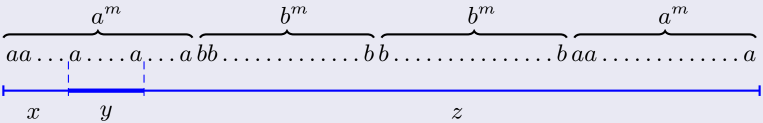
Because of this choice, and the requirement that $|xy| \leqslant m$, the opponent is restricted in Step 3 to choosing a string y that consists entirely of a's. In Step 4, we use $i = 0$. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R. Therefore, L is not regular.

4.3 Identifying Nonregular Languages

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the Pumping Lemma on our next move.

Example 4.8

Show that $L = \{ww^R \colon w \in \Sigma^*\}$ is not regular.
Whatever m the opponent picks on Step 1, we can always choose a w as shown in the Figure.

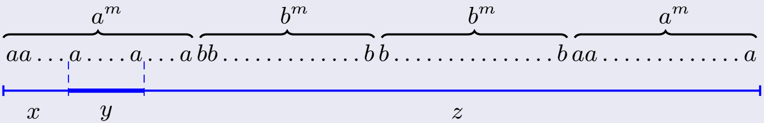


Because of this choice, and the requirement that $|xy| \leqslant m$, the opponent is restricted in Step 3 to choosing a string y that consists entirely of a's. In Step 4, we use $i = 0$. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R. Therefore, L is not regular.

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the Pumping Lemma on our next move.

Example 4.8

Show that $L = \{ww^R : w \in \Sigma^*\}$ is not regular.
Whatever m the opponent picks on Step 1, we can always choose a w as shown in the Figure.



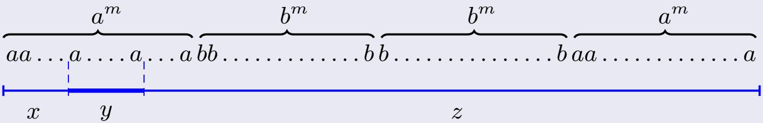
Because of this choice, and the requirement that $|xy| \leqslant m$, the opponent is restricted in Step 3 to choosing a string y that consists entirely of a's. In Step 4, we use $i = 0$. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R. Therefore, L is not regular.

4.3 Identifying Nonregular Languages

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the Pumping Lemma on our next move.

Example 4.8

Show that $L = \{ww^R \colon w \in \Sigma^*\}$ is not regular.
Whatever m the opponent picks on Step 1, we can always choose a w as shown in the Figure.

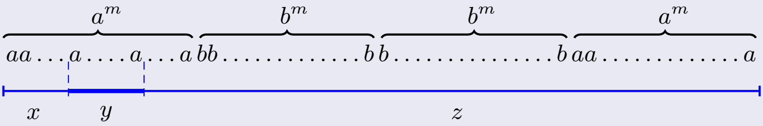


Because of this choice, and the requirement that $|xy| \leqslant m$, the opponent is restricted in Step 3 to choosing a string y that consists entirely of a's. In Step 4, we use $i = 0$. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R. Therefore, L is not regular.

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the Pumping Lemma on our next move.

Example 4.8

Show that $L = \{ww^R : w \in \Sigma^*\}$ is not regular.
Whatever m the opponent picks on Step 1, we can always choose a w as shown in the Figure.



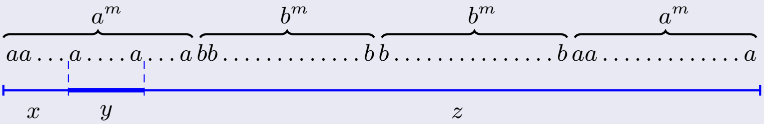
Because of this choice, and the requirement that $|xy| \leqslant m$, the opponent is restricted in Step 3 to choosing a string y that consists entirely of a's. In Step 4, we use $i = 0$. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R. Therefore, L is not regular.

4.3 Identifying Nonregular Languages

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the Pumping Lemma on our next move.

Example 4.8

Show that $L = \{ww^R \colon w \in \Sigma^*\}$ is not regular.
Whatever m the opponent picks on Step 1, we can always choose a w as shown in the Figure.



Because of this choice, and the requirement that $|xy| \leqslant m$, the opponent is restricted in Step 3 to choosing a string y that consists entirely of a's. In Step 4, we use $i = 0$. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R. Therefore, L is not regular.

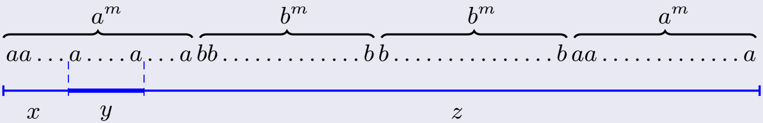
4.3 Identifying Nonregular Languages

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the Pumping Lemma on our next move.

Example 4.8

Show that $L = \{ww^R : w \in \Sigma^*\}$ is not regular.

Whatever m the opponent picks on Step 1, we can always choose a w as shown in the Figure.



Because of this choice, and the requirement that $|xy| \leqslant m$, the opponent is restricted in Step 3 to choosing a string y that consists entirely of a's. In Step 4, we use $i = 0$. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R. Therefore, L is not regular.

Example 4.8 (continuation)

Note that if we had chosen a string w too short, then the opponent could have chosen a string y with an even number of b's. In that case, we could not have reached a violation of the Pumping Lemma on the last step. We would also fail if we were to choose a string consisting of all a's, say,

$$w = a^{2m},$$

which is in L. To defeat us, the opponent need only pick

$$y = aa.$$

Now w_i is in L for all i, and we lose.

To apply the Pumping Lemma we cannot assume that the opponent will make a wrong move. If, in the case where we pick $w = a^{2m}$, the opponent were to pick

$$y = a,$$

then w_0 is a string of odd length and therefore not in L. But any argument that assumes that the opponent is so accommodating is automatically incorrect.

Example 4.8 (continuation)

Note that if we had chosen a string w too short, then the opponent could have chosen a string y with an even number of b's. In that case, we could not have reached a violation of the Pumping Lemma on the last step. We would also fail if we were to choose a string consisting of all a's, say,

$$w = a^{2m},$$

which is in L. To defeat us, the opponent need only pick

$$y = aa.$$

Now w_i is in L for all i, and we lose.

To apply the Pumping Lemma we cannot assume that the opponent will make a wrong move. If, in the case where we pick $w = a^{2m}$, the opponent were to pick

$$y = a,$$

then w_0 is a string of odd length and therefore not in L. But any argument that assumes that the opponent is so accommodating is automatically incorrect.

Example 4.8 (continuation)

Note that if we had chosen a string w too short, then the opponent could have chosen a string y with an even number of b's. In that case, we could not have reached a violation of the Pumping Lemma on the last step. We would also fail if we were to choose a string consisting of all a's, say,

$$w = a^{2m},$$

which is in L. To defeat us, the opponent need only pick

$$y = aa.$$

Now w_i is in L for all i, and we lose.

To apply the Pumping Lemma we cannot assume that the opponent will make a wrong move. If, in the case where we pick $w = a^{2m}$, the opponent were to pick

$$y = a,$$

then w_0 is a string of odd length and therefore not in L. But any argument that assumes that the opponent is so accommodating is automatically incorrect.

Example 4.8 (continuation)

Note that if we had chosen a string w too short, then the opponent could have chosen a string y with an even number of b's. In that case, we could not have reached a violation of the Pumping Lemma on the last step. We would also fail if we were to choose a string consisting of all a's, say,

$$w = a^{2m},$$

which is in L. To defeat us, the opponent need only pick

$$y = aa.$$

Now w_i is in L for all i, and we lose.

To apply the Pumping Lemma we cannot assume that the opponent will make a wrong move. If, in the case where we pick $w = a^{2m}$, the opponent were to pick

$$y = a,$$

then w_0 is a string of odd length and therefore not in L. But any argument that assumes that the opponent is so accommodating is automatically incorrect.

Example 4.8 (continuation)

Note that if we had chosen a string w too short, then the opponent could have chosen a string y with an even number of b's. In that case, we could not have reached a violation of the Pumping Lemma on the last step. We would also fail if we were to choose a string consisting of all a's, say,

$$w = a^{2m},$$

which is in L. To defeat us, the opponent need only pick

$$y = aa.$$

Now w_i is in L for all i, and we lose.

To apply the Pumping Lemma we cannot assume that the opponent will make a wrong move. If, in the case where we pick $w = a^{2m}$, the opponent were to pick

$$y = a,$$

then w_0 is a string of odd length and therefore not in L. But any argument that assumes that the opponent is so accommodating is automatically incorrect.

Example 4.8 (continuation)

Note that if we had chosen a string w too short, then the opponent could have chosen a string y with an even number of b's. In that case, we could not have reached a violation of the Pumping Lemma on the last step. We would also fail if we were to choose a string consisting of all a's, say,

$$w = a^{2m},$$

which is in L. To defeat us, the opponent need only pick

$$y = aa.$$

Now w_i is in L for all i, and we lose.

To apply the Pumping Lemma we cannot assume that the opponent will make a wrong move. If, in the case where we pick $w = a^{2m}$, the opponent were to pick

$$y = a,$$

then w_0 is a string of odd length and therefore not in L. But any argument that assumes that the opponent is so accommodating is automatically incorrect.

Example 4.8 (continuation)

Note that if we had chosen a string w too short, then the opponent could have chosen a string y with an even number of b's. In that case, we could not have reached a violation of the Pumping Lemma on the last step. We would also fail if we were to choose a string consisting of all a's, say,

$$w = a^{2m},$$

which is in L. To defeat us, the opponent need only pick

$$y = aa.$$

Now w_i is in L for all i, and we lose.

To apply the Pumping Lemma we cannot assume that the opponent will make a wrong move. If, in the case where we pick $w = a^{2m}$, the opponent were to pick

$$y = a,$$

then w_0 is a string of odd length and therefore not in L. But any argument that assumes that the opponent is so accommodating is automatically incorrect.

Example 4.8 (continuation)

Note that if we had chosen a string w too short, then the opponent could have chosen a string y with an even number of b's. In that case, we could not have reached a violation of the Pumping Lemma on the last step. We would also fail if we were to choose a string consisting of all a's, say,

$$w = a^{2m},$$

which is in L. To defeat us, the opponent need only pick

$$y = aa.$$

Now w_i is in L for all i, and we lose.

To apply the Pumping Lemma we cannot assume that the opponent will make a wrong move. If, in the case where we pick $w = a^{2m}$, the opponent were to pick

$$y = a,$$

then w_0 is a string of odd length and therefore not in L. But any argument that assumes that the opponent is so accommodating is automatically incorrect.

Example 4.8 (continuation)

Note that if we had chosen a string w too short, then the opponent could have chosen a string y with an even number of b's. In that case, we could not have reached a violation of the Pumping Lemma on the last step. We would also fail if we were to choose a string consisting of all a's, say,

$$w = a^{2m},$$

which is in L. To defeat us, the opponent need only pick

$$y = aa.$$

Now w_i is in L for all i, and we lose.

To apply the Pumping Lemma we cannot assume that the opponent will make a wrong move. If, in the case where we pick $w = a^{2m}$, the opponent were to pick

$$y = a,$$

then w_0 is a string of odd length and therefore not in L. But any argument that assumes that the opponent is so accommodating is automatically incorrect.

Example 4.8 (continuation)

Note that if we had chosen a string w too short, then the opponent could have chosen a string y with an even number of b's. In that case, we could not have reached a violation of the Pumping Lemma on the last step. We would also fail if we were to choose a string consisting of all a's, say,

$$w = a^{2m},$$

which is in L. To defeat us, the opponent need only pick

$$y = aa.$$

Now w_i is in L for all i, and we lose.

To apply the Pumping Lemma we cannot assume that the opponent will make a wrong move. If, in the case where we pick $w = a^{2m}$, the opponent were to pick

$$y = a,$$

then w_0 is a string of odd length and therefore not in L. But any argument that assumes that the opponent is so accommodating is automatically incorrect.

Example 4.8 (continuation)

Note that if we had chosen a string w too short, then the opponent could have chosen a string y with an even number of b's. In that case, we could not have reached a violation of the Pumping Lemma on the last step. We would also fail if we were to choose a string consisting of all a's, say,

$$w = a^{2m},$$

which is in L. To defeat us, the opponent need only pick

$$y = aa.$$

Now w_i is in L for all i, and we lose.

To apply the Pumping Lemma we cannot assume that the opponent will make a wrong move. If, in the case where we pick $w = a^{2m}$, the opponent were to pick

$$y = a,$$

then w_0 is a string of odd length and therefore not in L. But any argument that assumes that the opponent is so accommodating is automatically incorrect.

Example 4.8 (continuation)

Note that if we had chosen a string w too short, then the opponent could have chosen a string y with an even number of b's. In that case, we could not have reached a violation of the Pumping Lemma on the last step. We would also fail if we were to choose a string consisting of all a's, say,

$$w = a^{2m},$$

which is in L. To defeat us, the opponent need only pick

$$y = aa.$$

Now w_i is in L for all i, and we lose.

To apply the Pumping Lemma we cannot assume that the opponent will make a wrong move. If, in the case where we pick $w = a^{2m}$, the opponent were to pick

$$y = a,$$

then w_0 is a string of odd length and therefore not in L. But any argument that assumes that the opponent is so accommodating is automatically incorrect.

Example 4.8 (continuation)

Note that if we had chosen a string w too short, then the opponent could have chosen a string y with an even number of b's. In that case, we could not have reached a violation of the Pumping Lemma on the last step. We would also fail if we were to choose a string consisting of all a's, say,

$$w = a^{2m},$$

which is in L. To defeat us, the opponent need only pick

$$y = aa.$$

Now w_i is in L for all i, and we lose.

To apply the Pumping Lemma we cannot assume that the opponent will make a wrong move. If, in the case where we pick $w = a^{2m}$, the opponent were to pick

$$y = a,$$

then w_0 is a string of odd length and therefore not in L. But any argument that assumes that the opponent is so accommodating is automatically incorrect.

Example 4.8 (continuation)

Note that if we had chosen a string w too short, then the opponent could have chosen a string y with an even number of b's. In that case, we could not have reached a violation of the Pumping Lemma on the last step. We would also fail if we were to choose a string consisting of all a's, say,

$$w = a^{2m},$$

which is in L. To defeat us, the opponent need only pick

$$y = aa.$$

Now w_i is in L for all i, and we lose.

To apply the Pumping Lemma we cannot assume that the opponent will make a wrong move. If, in the case where we pick $w = a^{2m}$, the opponent were to pick

$$y = a,$$

then w_0 is a string of odd length and therefore not in L. But any argument that assumes that the opponent is so accommodating is automatically incorrect.

Example 4.8 (continuation)

Note that if we had chosen a string w too short, then the opponent could have chosen a string y with an even number of b's. In that case, we could not have reached a violation of the Pumping Lemma on the last step. We would also fail if we were to choose a string consisting of all a's, say,

$$w = a^{2m},$$

which is in L. To defeat us, the opponent need only pick

$$y = aa.$$

Now w_i is in L for all i, and we lose.

To apply the Pumping Lemma we cannot assume that the opponent will make a wrong move. If, in the case where we pick $w = a^{2m}$, the opponent were to pick

$$y = a,$$

then w_0 is a string of odd length and therefore not in L. But any argument that assumes that the opponent is so accommodating is automatically incorrect.

Example 4.8 (continuation)

Note that if we had chosen a string w too short, then the opponent could have chosen a string y with an even number of b's. In that case, we could not have reached a violation of the Pumping Lemma on the last step. We would also fail if we were to choose a string consisting of all a's, say,

$$w = a^{2m},$$

which is in L. To defeat us, the opponent need only pick

$$y = aa.$$

Now w_i is in L for all i, and we lose.

To apply the Pumping Lemma we cannot assume that the opponent will make a wrong move. If, in the case where we pick $w = a^{2m}$, the opponent were to pick

$$y = a,$$

then w_0 is a string of odd length and therefore not in L. But any argument that assumes that the opponent is so accommodating is automatically incorrect.

Example 4.9

Let $\Sigma = \{a, b\}$. The language

$$L = \{w \in \Sigma^* : n_a(w) < n_b(w)\}$$

is not regular.

Suppose we are given m. Since we have complete freedom in choosing w, we pick $w = a^m b^{m+1}$. Now, because $|xy|$ cannot be greater than m, the opponent cannot do anything but pick a string y with all a's, that is

$$y = a^k, \qquad 1 \leqslant k \leqslant m.$$

We now pump up, using $i = 2$. The resulting string

$$w_2 = a^{m+k} b^{m+1}$$

is not in L. Therefore, the Pumping Lemma is violated, and L is not regular.

Example 4.9

Let $\Sigma = \{a, b\}$. The language

$$L = \left\{ w \in \Sigma^* : n_a(w) < n_b(w) \right\}$$

is not regular.

Suppose we are given m. Since we have complete freedom in choosing w, we pick $w = a^m b^{m+1}$. Now, because $|xy|$ cannot be greater than m, the opponent cannot do anything but pick a string y with all a's, that is

$$y = a^k, \qquad 1 \leqslant k \leqslant m.$$

We now pump up, using $i = 2$. The resulting string

$$w_2 = a^{m+k} b^{m+1}$$

is not in L. Therefore, the Pumping Lemma is violated, and L is not regular.

Example 4.9

Let $\Sigma = \{a, b\}$. The language

$$L = \{w \in \Sigma^* : n_a(w) < n_b(w)\}$$

is not regular.

Suppose we are given m. Since we have complete freedom in choosing w, we pick $w = a^m b^{m+1}$. Now, because $|xy|$ cannot be greater than m, the opponent cannot do anything but pick a string y with all a's, that is

$$y = a^k, \qquad 1 \leqslant k \leqslant m.$$

We now pump up, using $i = 2$. The resulting string

$$w_2 = a^{m+k} b^{m+1}$$

is not in L. Therefore, the Pumping Lemma is violated, and L is not regular.

Example 4.9

Let $\Sigma = \{a, b\}$. The language

$$L = \{w \in \Sigma^* : n_a(w) < n_b(w)\}$$

is not regular.

Suppose we are given m. Since we have complete freedom in choosing w, we pick $w = a^m b^{m+1}$. Now, because $|xy|$ cannot be greater than m, the opponent cannot do anything but pick a string y with all a's, that is

$$y = a^k, \qquad 1 \leqslant k \leqslant m.$$

We now pump up, using $i = 2$. The resulting string

$$w_2 = a^{m+k} b^{m+1}$$

is not in L. Therefore, the Pumping Lemma is violated, and L is not regular.

Example 4.9

Let $\Sigma = \{a, b\}$. The language

$$L = \{w \in \Sigma^* : n_a(w) < n_b(w)\}$$

is not regular.

Suppose we are given m. Since we have complete freedom in choosing w, we pick $w = a^m b^{m+1}$. Now, because $|xy|$ cannot be greater than m, the opponent cannot do anything but pick a string y with all a's, that is

$$y = a^k, \qquad 1 \leqslant k \leqslant m.$$

We now pump up, using $i = 2$. The resulting string

$$w_2 = a^{m+k} b^{m+1}$$

is not in L. Therefore, the Pumping Lemma is violated, and L is not regular.

Example 4.9

Let $\Sigma = \{a, b\}$. The language

$$L = \{w \in \Sigma^* : n_a(w) < n_b(w)\}$$

is not regular.

Suppose we are given m. Since we have complete freedom in choosing w, we pick $w = a^m b^{m+1}$. Now, because $|xy|$ cannot be greater than m, the opponent cannot do anything but pick a string y with all a's, that is

$$y = a^k, \qquad 1 \leqslant k \leqslant m.$$

We now pump up, using $i = 2$. The resulting string

$$w_2 = a^{m+k} b^{m+1}$$

is not in L. Therefore, the Pumping Lemma is violated, and L is not regular.

Example 4.9

Let $\Sigma = \{a, b\}$. The language

$$L = \{w \in \Sigma^* : n_a(w) < n_b(w)\}$$

is not regular.

Suppose we are given m. Since we have complete freedom in choosing w, we pick $w = a^m b^{m+1}$. Now, because $|xy|$ cannot be greater than m, the opponent cannot do anything but pick a string y with all a's, that is

$$y = a^k, \qquad 1 \leqslant k \leqslant m.$$

We now pump up, using $i = 2$. The resulting string

$$w_2 = a^{m+k} b^{m+1}$$

is not in L. Therefore, the Pumping Lemma is violated, and L is not regular.

Example 4.9

Let $\Sigma = \{a, b\}$. The language

$$L = \{w \in \Sigma^* : n_a(w) < n_b(w)\}$$

is not regular.

Suppose we are given m. Since we have complete freedom in choosing w, we pick $w = a^m b^{m+1}$. Now, because $|xy|$ cannot be greater than m, the opponent cannot do anything but pick a string y with all a's, that is

$$y = a^k, \qquad 1 \leqslant k \leqslant m.$$

We now pump up, using $i = 2$. The resulting string

$$w_2 = a^{m+k} b^{m+1}$$

is not in L. Therefore, the Pumping Lemma is violated, and L is not regular.

Example 4.9

Let $\Sigma = \{a, b\}$. The language

$$L = \{w \in \Sigma^* : n_a(w) < n_b(w)\}$$

is not regular.

Suppose we are given m. Since we have complete freedom in choosing w, we pick $w = a^m b^{m+1}$. Now, because $|xy|$ cannot be greater than m, the opponent cannot do anything but pick a string y with all a's, that is

$$y = a^k, \qquad 1 \leqslant k \leqslant m.$$

We now pump up, using $i = 2$. The resulting string

$$w_2 = a^{m+k} b^{m+1}$$

is not in L. Therefore, the Pumping Lemma is violated, and L is not regular.

Example 4.9

Let $\Sigma = \{a, b\}$. The language
$$L = \{w \in \Sigma^* : n_a(w) < n_b(w)\}$$
is not regular.

Suppose we are given m. Since we have complete freedom in choosing w, we pick $w = a^m b^{m+1}$. Now, because $|xy|$ cannot be greater than m, the opponent cannot do anything but pick a string y with all a's, that is

$$y = a^k, \qquad 1 \leqslant k \leqslant m.$$

We now pump up, using $i = 2$. The resulting string

$$w_2 = a^{m+k} b^{m+1}$$

is not in L. Therefore, the Pumping Lemma is violated, and L is not regular.

Example 4.9

Let $\Sigma = \{a, b\}$. The language

$$L = \{w \in \Sigma^* : n_a(w) < n_b(w)\}$$

is not regular.

Suppose we are given m. Since we have complete freedom in choosing w, we pick $w = a^m b^{m+1}$. Now, because $|xy|$ cannot be greater than m, the opponent cannot do anything but pick a string y with all a's, that is

$$y = a^k, \qquad 1 \leqslant k \leqslant m.$$

We now pump up, using $i = 2$. The resulting string

$$w_2 = a^{m+k} b^{m+1}$$

is not in L. Therefore, the Pumping Lemma is violated, and L is not regular.

Example 4.9

Let $\Sigma = \{a, b\}$. The language

$$L = \left\{w \in \Sigma^* : n_a(w) < n_b(w)\right\}$$

is not regular.

Suppose we are given m. Since we have complete freedom in choosing w, we pick $w = a^m b^{m+1}$. Now, because $|xy|$ cannot be greater than m, the opponent cannot do anything but pick a string y with all a's, that is

$$y = a^k, \qquad 1 \leqslant k \leqslant m.$$

We now pump up, using $i = 2$. The resulting string

$$w_2 = a^{m+k} b^{m+1}$$

is not in L. Therefore, the Pumping Lemma is violated, and L is not regular.

Example 4.9

Let $\Sigma = \{a, b\}$. The language

$$L = \{w \in \Sigma^* : n_a(w) < n_b(w)\}$$

is not regular.

Suppose we are given m. Since we have complete freedom in choosing w, we pick $w = a^m b^{m+1}$. Now, because $|xy|$ cannot be greater than m, the opponent cannot do anything but pick a string y with all a's, that is

$$y = a^k, \qquad 1 \leqslant k \leqslant m.$$

We now pump up, using $i = 2$. The resulting string

$$w_2 = a^{m+k} b^{m+1}$$

is not in L. Therefore, the Pumping Lemma is violated, and L is not regular.

Example 4.9

Let $\Sigma = \{a, b\}$. The language

$$L = \{w \in \Sigma^* : n_a(w) < n_b(w)\}$$

is not regular.

Suppose we are given m. Since we have complete freedom in choosing w, we pick $w = a^m b^{m+1}$. Now, because $|xy|$ cannot be greater than m, the opponent cannot do anything but pick a string y with all a's, that is

$$y = a^k, \qquad 1 \leqslant k \leqslant m.$$

We now pump up, using $i = 2$. The resulting string

$$w_2 = a^{m+k} b^{m+1}$$

is not in L. Therefore, the Pumping Lemma is violated, and L is not regular.

Example 4.9

Let $\Sigma = \{a, b\}$. The language

$$L = \left\{w \in \Sigma^* : n_a(w) < n_b(w)\right\}$$

is not regular.

Suppose we are given m. Since we have complete freedom in choosing w, we pick $w = a^m b^{m+1}$. Now, because $|xy|$ cannot be greater than m, the opponent cannot do anything but pick a string y with all a's, that is

$$y = a^k, \qquad 1 \leqslant k \leqslant m.$$

We now pump up, using $i = 2$. The resulting string

$$w_2 = a^{m+k} b^{m+1}$$

is not in L. Therefore, the Pumping Lemma is violated, and L is not regular.

Example 4.9

Let $\Sigma = \{a, b\}$. The language

$$L = \{w \in \Sigma^* : n_a(w) < n_b(w)\}$$

is not regular.

Suppose we are given m. Since we have complete freedom in choosing w, we pick $w = a^m b^{m+1}$. Now, because $|xy|$ cannot be greater than m, the opponent cannot do anything but pick a string y with all a's, that is

$$y = a^k, \qquad 1 \leqslant k \leqslant m.$$

We now pump up, using $i = 2$. The resulting string

$$w_2 = a^{m+k} b^{m+1}$$

is not in L. Therefore, the Pumping Lemma is violated, and L is not regular.

Example 4.9

Let $\Sigma = \{a, b\}$. The language

$$L = \{w \in \Sigma^* : n_a(w) < n_b(w)\}$$

is not regular.

Suppose we are given m. Since we have complete freedom in choosing w, we pick $w = a^m b^{m+1}$. Now, because $|xy|$ cannot be greater than m, the opponent cannot do anything but pick a string y with all a's, that is

$$y = a^k, \qquad 1 \leqslant k \leqslant m.$$

We now pump up, using $i = 2$. The resulting string

$$w_2 = a^{m+k} b^{m+1}$$

is not in L. Therefore, the Pumping Lemma is violated, and L is not regular.

Example 4.9

Let $\Sigma = \{a, b\}$. The language

$$L = \big\{w \in \Sigma^* : n_a(w) < n_b(w)\big\}$$

is not regular.

Suppose we are given m. Since we have complete freedom in choosing w, we pick $w = a^m b^{m+1}$. Now, because $|xy|$ cannot be greater than m, the opponent cannot do anything but pick a string y with all a's, that is

$$y = a^k, \qquad 1 \leqslant k \leqslant m.$$

We now pump up, using $i = 2$. The resulting string

$$w_2 = a^{m+k} b^{m+1}$$

is not in L. Therefore, the Pumping Lemma is violated, and L is not regular.

Example 4.10

Let $\Sigma = \{a, b\}$. The language

$$L = \left\{ (ab)^n a^k : n > k, k \geqslant 0 \right\}$$

is not regular.

Given m, we pick as our string

$$w = (ab)^{m+1} a^m,$$

which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.10

Let $\Sigma = \{a, b\}$. The language

$$L = \left\{ (ab)^n a^k : n > k, k \geqslant 0 \right\}$$

is not regular.

Given m, we pick as our string

$$w = (ab)^{m+1} a^m,$$

which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.10

Let $\Sigma = \{a, b\}$. The language

$$L = \left\{(ab)^n a^k : n > k, k \geqslant 0\right\}$$

is not regular.

Given m, we pick as our string

$$w = (ab)^{m+1} a^m,$$

which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.10

Let $\Sigma = \{a, b\}$. The language

$$L = \left\{(ab)^n a^k : n > k, k \geqslant 0\right\}$$

is not regular.

Given m, we pick as our string

$$w = (ab)^{m+1} a^m,$$

which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.10

Let $\Sigma = \{a, b\}$. The language

$$L = \left\{ (ab)^n a^k : n > k, k \geqslant 0 \right\}$$

is not regular.

Given m, we pick as our string

$$w = (ab)^{m+1} a^m,$$

which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.10

Let $\Sigma = \{a, b\}$. The language
$$L = \left\{ (ab)^n a^k : n > k, k \geqslant 0 \right\}$$
is not regular.

Given m, we pick as our string
$$w = (ab)^{m+1} a^m,$$
which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.10

Let $\Sigma = \{a, b\}$. The language

$$L = \left\{(ab)^n a^k : n > k, k \geqslant 0\right\}$$

is not regular.

Given m, we pick as our string

$$w = (ab)^{m+1} a^m,$$

which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.10

Let $\Sigma = \{a, b\}$. The language

$$L = \left\{(ab)^n a^k : n > k, k \geqslant 0\right\}$$

is not regular.

Given m, we pick as our string

$$w = (ab)^{m+1} a^m,$$

which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.10

Let $\Sigma = \{a, b\}$. The language
$$L = \left\{(ab)^n a^k : n > k, k \geqslant 0\right\}$$
is not regular.

Given m, we pick as our string
$$w = (ab)^{m+1} a^m,$$
which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.10

Let $\Sigma = \{a, b\}$. The language
$$L = \left\{(ab)^n a^k : n > k, k \geqslant 0\right\}$$
is not regular.

Given m, we pick as our string

$$w = (ab)^{m+1} a^m,$$

which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.10

Let $\Sigma = \{a, b\}$. The language

$$L = \left\{(ab)^n a^k : n > k, k \geqslant 0\right\}$$

is not regular.

Given m, we pick as our string

$$w = (ab)^{m+1} a^m,$$

which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.10

Let $\Sigma = \{a, b\}$. The language

$$L = \left\{(ab)^n a^k : n > k, k \geqslant 0\right\}$$

is not regular.

Given m, we pick as our string

$$w = (ab)^{m+1} a^m,$$

which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.10

Let $\Sigma = \{a, b\}$. The language

$$L = \left\{(ab)^n a^k : n > k, k \geqslant 0\right\}$$

is not regular.

Given m, we pick as our string

$$w = (ab)^{m+1} a^m,$$

which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.10

Let $\Sigma = \{a, b\}$. The language

$$L = \left\{(ab)^n a^k : n > k, k \geqslant 0\right\}$$

is not regular.

Given m, we pick as our string

$$w = (ab)^{m+1} a^m,$$

which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.10

Let $\Sigma = \{a, b\}$. The language

$$L = \left\{ (ab)^n a^k : n > k, k \geqslant 0 \right\}$$

is not regular.

Given m, we pick as our string

$$w = (ab)^{m+1} a^m,$$

which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.10

Let $\Sigma = \{a, b\}$. The language
$$L = \left\{(ab)^n a^k : n > k, k \geqslant 0\right\}$$
is not regular.

Given m, we pick as our string

$$w = (ab)^{m+1} a^m,$$

which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.10

Let $\Sigma = \{a, b\}$. The language

$$L = \left\{(ab)^n a^k : n > k, k \geqslant 0\right\}$$

is not regular.

Given m, we pick as our string

$$w = (ab)^{m+1} a^m,$$

which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.10

Let $\Sigma = \{a, b\}$. The language

$$L = \left\{(ab)^n a^k : n > k, k \geqslant 0\right\}$$

is not regular.

Given m, we pick as our string

$$w = (ab)^{m+1} a^m,$$

which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.10

Let $\Sigma = \{a, b\}$. The language

$$L = \left\{ (ab)^n a^k : n > k, k \geqslant 0 \right\}$$

is not regular.

Given m, we pick as our string

$$w = (ab)^{m+1} a^m,$$

which is in L. Since of the constraint $|xy| \leqslant m$, both x and y must be in the part of the string made up of ab's. The choice of x does not affect the argument, so let us see what can be done with y. If our opponent picks $y = a$, we choose $i = 0$ and get a string not in $L((ab)^* a^*)$. If the opponent picks $y = ab$, then we can choose $i = 0$ again. Now we get the string $(ab)^m a^m$, which is not in L. In the same way, we can deal with any possible choice by the opponent, thereby proving our claim.

Example 4.11

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n : n \text{ is a perfect square}\}$$

is not regular.

Given the opponent's choice of m, we pick

$$w = a^{m^2}.$$

If $w = xyz$ is the decomposition, then clearly

$$y = a^k$$

with $1 \leqslant k \leqslant m$. In that case,

$$w_0 = a^{m^2 - k}$$

But $m^2 - k > (m-1)^2$, so that w_0 cannot be in L. Therefore, the language is not regular.

In some cases, closure properties can be used to relate a given problem to one we have already classified. This may be simpler than a direct application of the Pumping Lemma.

Example 4.11

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n : n \text{ is a perfect square}\}$$

is not regular.

Given the opponent's choice of m, we pick

$$w = a^{m^2}.$$

If $w = xyz$ is the decomposition, then clearly

$$y = a^k$$

with $1 \leqslant k \leqslant m$. In that case,

$$w_0 = a^{m^2 - k}$$

But $m^2 - k > (m-1)^2$, so that w_0 cannot be in L. Therefore, the language is not regular.

In some cases, closure properties can be used to relate a given problem to one we have already classified. This may be simpler than a direct application of the Pumping Lemma.

Example 4.11

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n : n \text{ is a perfect square}\}$$

is not regular.

Given the opponent's choice of m, we pick

$$w = a^{m^2}.$$

If $w = xyz$ is the decomposition, then clearly

$$y = a^k$$

with $1 \leqslant k \leqslant m$. In that case,

$$w_0 = a^{m^2 - k}$$

But $m^2 - k > (m-1)^2$, so that w_0 cannot be in L. Therefore, the language is not regular.

In some cases, closure properties can be used to relate a given problem to one we have already classified. This may be simpler than a direct application of the Pumping Lemma.

Example 4.11

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n : n \text{ is a perfect square}\}$$

is not regular.

Given the opponent's choice of m, we pick

$$w = a^{m^2}.$$

If $w = xyz$ is the decomposition, then clearly

$$y = a^k$$

with $1 \leqslant k \leqslant m$. In that case,

$$w_0 = a^{m^2 - k}$$

But $m^2 - k > (m-1)^2$, so that w_0 cannot be in L. Therefore, the language is not regular.

In some cases, closure properties can be used to relate a given problem to one we have already classified. This may be simpler than a direct application of the Pumping Lemma.

Example 4.11

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n : n \text{ is a perfect square}\}$$

is not regular.

Given the opponent's choice of m, we pick

$$w = a^{m^2}.$$

If $w = xyz$ is the decomposition, then clearly

$$y = a^k$$

with $1 \leqslant k \leqslant m$. In that case,

$$w_0 = a^{m^2-k}$$

But $m^2 - k > (m-1)^2$, so that w_0 cannot be in L. Therefore, the language is not regular.

In some cases, closure properties can be used to relate a given problem to one we have already classified. This may be simpler than a direct application of the Pumping Lemma.

Example 4.11

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n : n \text{ is a perfect square}\}$$

is not regular.

Given the opponent's choice of m, we pick

$$w = a^{m^2}.$$

If $w = xyz$ is the decomposition, then clearly

$$y = a^k$$

with $1 \leqslant k \leqslant m$. In that case,

$$w_0 = a^{m^2 - k}$$

But $m^2 - k > (m-1)^2$, so that w_0 cannot be in L. Therefore, the language is not regular.

In some cases, closure properties can be used to relate a given problem to one we have already classified. This may be simpler than a direct application of the Pumping Lemma.

Example 4.11

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n : n \text{ is a perfect square}\}$$

is not regular.

Given the opponent's choice of m, we pick

$$w = a^{m^2}.$$

If $w = xyz$ is the decomposition, then clearly

$$y = a^k$$

with $1 \leqslant k \leqslant m$. In that case,

$$w_0 = a^{m^2 - k}$$

But $m^2 - k > (m-1)^2$, so that w_0 cannot be in L. Therefore, the language is not regular.

In some cases, closure properties can be used to relate a given problem to one we have already classified. This may be simpler than a direct application of the Pumping Lemma.

Example 4.11

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n : n \text{ is a perfect square}\}$$

is not regular.

Given the opponent's choice of m, we pick

$$w = a^{m^2}.$$

If $w = xyz$ is the decomposition, then clearly

$$y = a^k$$

with $1 \leqslant k \leqslant m$. In that case,

$$w_0 = a^{m^2 - k}$$

But $m^2 - k > (m-1)^2$, so that w_0 cannot be in L. Therefore, the language is not regular.

In some cases, closure properties can be used to relate a given problem to one we have already classified. This may be simpler than a direct application of the Pumping Lemma.

Example 4.11

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n : n \text{ is a perfect square}\}$$

is not regular.

Given the opponent's choice of m, we pick

$$w = a^{m^2}.$$

If $w = xyz$ is the decomposition, then clearly

$$y = a^k$$

with $1 \leqslant k \leqslant m$. In that case,

$$w_0 = a^{m^2 - k}$$

But $m^2 - k > (m-1)^2$, so that w_0 cannot be in L. Therefore, the language is not regular.

In some cases, closure properties can be used to relate a given problem to one we have already classified. This may be simpler than a direct application of the Pumping Lemma.

Example 4.11

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n : n \text{ is a perfect square}\}$$

is not regular.

Given the opponent's choice of m, we pick

$$w = a^{m^2}.$$

If $w = xyz$ is the decomposition, then clearly

$$y = a^k$$

with $1 \leqslant k \leqslant m$. In that case,

$$w_0 = a^{m^2 - k}$$

But $m^2 - k > (m-1)^2$, so that w_0 cannot be in L. Therefore, the language is not regular.

In some cases, closure properties can be used to relate a given problem to one we have already classified. This may be simpler than a direct application of the Pumping Lemma.

Example 4.11

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n : n \text{ is a perfect square}\}$$

is not regular.

Given the opponent's choice of m, we pick

$$w = a^{m^2}.$$

If $w = xyz$ is the decomposition, then clearly

$$y = a^k$$

with $1 \leqslant k \leqslant m$. In that case,

$$w_0 = a^{m^2 - k}$$

But $m^2 - k > (m-1)^2$, so that w_0 cannot be in L. Therefore, the language is not regular.

In some cases, closure properties can be used to relate a given problem to one we have already classified. This may be simpler than a direct application of the Pumping Lemma.

Example 4.11

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n : n \text{ is a perfect square}\}$$

is not regular.

Given the opponent's choice of m, we pick

$$w = a^{m^2}.$$

If $w = xyz$ is the decomposition, then clearly

$$y = a^k$$

with $1 \leqslant k \leqslant m$. In that case,

$$w_0 = a^{m^2-k}$$

But $m^2 - k > (m-1)^2$, so that w_0 cannot be in L. Therefore, the language is not regular.

In some cases, closure properties can be used to relate a given problem to one we have already classified. This may be simpler than a direct application of the Pumping Lemma.

Example 4.11

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n : n \text{ is a perfect square}\}$$

is not regular.

Given the opponent's choice of m, we pick

$$w = a^{m^2}.$$

If $w = xyz$ is the decomposition, then clearly

$$y = a^k$$

with $1 \leqslant k \leqslant m$. In that case,

$$w_0 = a^{m^2 - k}$$

But $m^2 - k > (m-1)^2$, so that w_0 cannot be in L. Therefore, the language is not regular.

In some cases, closure properties can be used to relate a given problem to one we have already classified. This may be simpler than a direct application of the Pumping Lemma.

Example 4.11

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n : n \text{ is a perfect square}\}$$

is not regular.

Given the opponent's choice of m, we pick

$$w = a^{m^2}.$$

If $w = xyz$ is the decomposition, then clearly

$$y = a^k$$

with $1 \leqslant k \leqslant m$. In that case,

$$w_0 = a^{m^2 - k}$$

But $m^2 - k > (m-1)^2$, so that w_0 cannot be in L. Therefore, the language is not regular.

In some cases, closure properties can be used to relate a given problem to one we have already classified. This may be simpler than a direct application of the Pumping Lemma.

Example 4.11

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n : n \text{ is a perfect square}\}$$

is not regular.

Given the opponent's choice of m, we pick

$$w = a^{m^2}.$$

If $w = xyz$ is the decomposition, then clearly

$$y = a^k$$

with $1 \leqslant k \leqslant m$. In that case,

$$w_0 = a^{m^2 - k}$$

But $m^2 - k > (m-1)^2$, so that w_0 cannot be in L. Therefore, the language is not regular.

In some cases, closure properties can be used to relate a given problem to one we have already classified. This may be simpler than a direct application of the Pumping Lemma.

Example 4.11

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n : n \text{ is a perfect square}\}$$

is not regular.

Given the opponent's choice of m, we pick

$$w = a^{m^2}.$$

If $w = xyz$ is the decomposition, then clearly

$$y = a^k$$

with $1 \leqslant k \leqslant m$. In that case,

$$w_0 = a^{m^2-k}$$

But $m^2 - k > (m-1)^2$, so that w_0 cannot be in L. Therefore, the language is not regular.

In some cases, closure properties can be used to relate a given problem to one we have already classified. This may be simpler than a direct application of the Pumping Lemma.

Example 4.11

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n : n \text{ is a perfect square}\}$$

is not regular.

Given the opponent's choice of m, we pick

$$w = a^{m^2}.$$

If $w = xyz$ is the decomposition, then clearly

$$y = a^k$$

with $1 \leqslant k \leqslant m$. In that case,

$$w_0 = a^{m^2-k}$$

But $m^2 - k > (m-1)^2$, so that w_0 cannot be in L. Therefore, the language is not regular.

In some cases, closure properties can be used to relate a given problem to one we have already classified. This may be simpler than a direct application of the Pumping Lemma.

Example 4.11

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n : n \text{ is a perfect square}\}$$

is not regular.

Given the opponent's choice of m, we pick

$$w = a^{m^2}.$$

If $w = xyz$ is the decomposition, then clearly

$$y = a^k$$

with $1 \leqslant k \leqslant m$. In that case,

$$w_0 = a^{m^2 - k}$$

But $m^2 - k > (m-1)^2$, so that w_0 cannot be in L. Therefore, the language is not regular.

In some cases, closure properties can be used to relate a given problem to one we have already classified. This may be simpler than a direct application of the Pumping Lemma.

Example 4.12

Let $\Sigma = \{a, b, c\}$. The language

$$L = \left\{a^n b^k c^{n+k} : n \geqslant 0, k \geqslant 0\right\}$$

is not regular.

It is not difficult to apply the Pumping Lemma directly, but it is even easier to use closure under homomorphism. Take

$$h(a) = a, \quad h(b) = a, \quad h(c) = c,$$

then

$$h(L) = \left\{a^{n+k} c^{n+k} : n + k \geqslant 0\right\} = \\ = \left\{a^i c^i : i \geqslant 0\right\}.$$

But we know this language is not regular; therefore, L cannot be regular either.

Example 4.12

Let $\Sigma = \{a, b, c\}$. The language

$$L = \left\{a^n b^k c^{n+k} : n \geqslant 0, k \geqslant 0\right\}$$

is not regular.

It is not difficult to apply the Pumping Lemma directly, but it is even easier to use closure under homomorphism. Take

$$h(a) = a, \quad h(b) = a, \quad h(c) = c,$$

then

$$h(L) = \left\{a^{n+k} c^{n+k} : n + k \geqslant 0\right\} = \\ = \left\{a^i c^i : i \geqslant 0\right\}.$$

But we know this language is not regular; therefore, L cannot be regular either.

Example 4.12

Let $\Sigma = \{a, b, c\}$. The language

$$L = \left\{a^n b^k c^{n+k} : n \geqslant 0, k \geqslant 0\right\}$$

is not regular.

It is not difficult to apply the Pumping Lemma directly, but it is even easier to use closure under homomorphism. Take

$$h(a) = a, \quad h(b) = a, \quad h(c) = c,$$

then

$$h(L) = \left\{a^{n+k} c^{n+k} : n + k \geqslant 0\right\} = \\ = \left\{a^i c^i : i \geqslant 0\right\}.$$

But we know this language is not regular; therefore, L cannot be regular either.

Example 4.12

Let $\Sigma = \{a, b, c\}$. The language

$$L = \left\{a^n b^k c^{n+k} : n \geqslant 0, k \geqslant 0\right\}$$

is not regular.

It is not difficult to apply the Pumping Lemma directly, but it is even easier to use closure under homomorphism. Take

$$h(a) = a, \quad h(b) = a, \quad h(c) = c,$$

then

$$h(L) = \left\{a^{n+k} c^{n+k} : n + k \geqslant 0\right\} = \\ = \left\{a^i c^i : i \geqslant 0\right\}.$$

But we know this language is not regular; therefore, L cannot be regular either.

Example 4.12

Let $\Sigma = \{a, b, c\}$. The language

$$L = \left\{a^n b^k c^{n+k} : n \geqslant 0, k \geqslant 0\right\}$$

is not regular.

It is not difficult to apply the Pumping Lemma directly, but it is even easier to use closure under homomorphism. Take

$$h(a) = a, \quad h(b) = a, \quad h(c) = c,$$

then

$$h(L) = \left\{a^{n+k} c^{n+k} : n + k \geqslant 0\right\} = \\ = \left\{a^i c^i : i \geqslant 0\right\}.$$

But we know this language is not regular; therefore, L cannot be regular either.

Example 4.12

Let $\Sigma = \{a, b, c\}$. The language

$$L = \left\{a^n b^k c^{n+k} : n \geqslant 0, k \geqslant 0\right\}$$

is not regular.

It is not difficult to apply the Pumping Lemma directly, but it is even easier to use closure under homomorphism. Take

$$h(a) = a, \quad h(b) = a, \quad h(c) = c,$$

then

$$h(L) = \left\{a^{n+k} c^{n+k} : n + k \geqslant 0\right\} = \\ = \left\{a^i c^i : i \geqslant 0\right\}.$$

But we know this language is not regular; therefore, L cannot be regular either.

Example 4.12

Let $\Sigma = \{a, b, c\}$. The language

$$L = \left\{a^n b^k c^{n+k} : n \geqslant 0, k \geqslant 0\right\}$$

is not regular.

It is not difficult to apply the Pumping Lemma directly, but it is even easier to use closure under homomorphism. Take

$$h(a) = a, \quad h(b) = a, \quad h(c) = c,$$

then

$$h(L) = \left\{a^{n+k} c^{n+k} : n + k \geqslant 0\right\} =$$
$$= \left\{a^i c^i : i \geqslant 0\right\}.$$

But we know this language is not regular; therefore, L cannot be regular either.

Example 4.12

Let $\Sigma = \{a, b, c\}$. The language

$$L = \left\{a^n b^k c^{n+k} : n \geqslant 0, k \geqslant 0\right\}$$

is not regular.

It is not difficult to apply the Pumping Lemma directly, but it is even easier to use closure under homomorphism. Take

$$h(a) = a, \quad h(b) = a, \quad h(c) = c,$$

then

$$h(L) = \left\{a^{n+k} c^{n+k} : n + k \geqslant 0\right\} = \\ = \left\{a^i c^i : i \geqslant 0\right\}.$$

But we know this language is not regular; therefore, L cannot be regular either.

Example 4.12

Let $\Sigma = \{a, b, c\}$. The language

$$L = \left\{ a^n b^k c^{n+k} : n \geqslant 0, k \geqslant 0 \right\}$$

is not regular.

It is not difficult to apply the Pumping Lemma directly, but it is even easier to use closure under homomorphism. Take

$$h(a) = a, \quad h(b) = a, \quad h(c) = c,$$

then

$$h(L) = \left\{ a^{n+k} c^{n+k} : n + k \geqslant 0 \right\} = \\ = \left\{ a^i c^i : i \geqslant 0 \right\}.$$

But we know this language is not regular; therefore, L cannot be regular either.

Example 4.12

Let $\Sigma = \{a, b, c\}$. The language

$$L = \left\{ a^n b^k c^{n+k} : n \geqslant 0, k \geqslant 0 \right\}$$

is not regular.

It is not difficult to apply the Pumping Lemma directly, but it is even easier to use closure under homomorphism. Take

$$h(a) = a, \quad h(b) = a, \quad h(c) = c,$$

then

$$h(L) = \left\{ a^{n+k} c^{n+k} : n + k \geqslant 0 \right\} = \\ = \left\{ a^i c^i : i \geqslant 0 \right\}.$$

But we know this language is not regular; therefore, L cannot be regular either.

Example 4.12

Let $\Sigma = \{a, b, c\}$. The language

$$L = \left\{a^n b^k c^{n+k} : n \geqslant 0, k \geqslant 0\right\}$$

is not regular.

It is not difficult to apply the Pumping Lemma directly, but it is even easier to use closure under homomorphism. Take

$$h(a) = a, \quad h(b) = a, \quad h(c) = c,$$

then

$$h(L) = \left\{a^{n+k} c^{n+k} : n + k \geqslant 0\right\} = \\ = \left\{a^i c^i : i \geqslant 0\right\}.$$

But we know this language is not regular; therefore, L cannot be regular either.

Example 4.12

Let $\Sigma = \{a, b, c\}$. The language

$$L = \left\{a^n b^k c^{n+k} : n \geqslant 0, k \geqslant 0\right\}$$

is not regular.

It is not difficult to apply the Pumping Lemma directly, but it is even easier to use closure under homomorphism. Take

$$h(a) = a, \quad h(b) = a, \quad h(c) = c,$$

then

$$h(L) = \left\{a^{n+k} c^{n+k} : n + k \geqslant 0\right\} = \\ = \left\{a^i c^i : i \geqslant 0\right\}.$$

But we know this language is not regular; therefore, L cannot be regular either.

Example 4.12

Let $\Sigma = \{a, b, c\}$. The language

$$L = \left\{a^n b^k c^{n+k} : n \geqslant 0, k \geqslant 0\right\}$$

is not regular.

It is not difficult to apply the Pumping Lemma directly, but it is even easier to use closure under homomorphism. Take

$$h(a) = a, \quad h(b) = a, \quad h(c) = c,$$

then

$$h(L) = \left\{a^{n+k} c^{n+k} : n + k \geqslant 0\right\} =$$
$$= \left\{a^i c^i : i \geqslant 0\right\}.$$

But we know this language is not regular; therefore, L cannot be regular either.

Example 4.12

Let $\Sigma = \{a, b, c\}$. The language

$$L = \left\{a^n b^k c^{n+k} : n \geqslant 0, k \geqslant 0\right\}$$

is not regular.

It is not difficult to apply the Pumping Lemma directly, but it is even easier to use closure under homomorphism. Take

$$h(a) = a, \quad h(b) = a, \quad h(c) = c,$$

then

$$\begin{aligned} h(L) = \left\{a^{n+k} c^{n+k} : n + k \geqslant 0\right\} = \\ = \left\{a^i c^i : i \geqslant 0\right\}. \end{aligned}$$

But we know this language is not regular; therefore, L cannot be regular either.

4.3 Identifying Nonregular Languages

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m+1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i-1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i-1)k = (m+1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, L and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language
$$L = \{a^n b^l : n \neq l\}$$
is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that
$$m! + (i - 1)k = (m + 1)!$$
This is always possible because
$$i = 1 + \frac{m\,m!}{k}$$
and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, L and the language
$$L_1 = \overline{L} \cap L(a^* b^*)$$
would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, L and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, $\overline{L}$ and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m+1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i-1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i-1)k = (m+1)!$$

This is always possible because

$$i = 1 + \frac{m\,m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, L and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, L and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly.

Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m+1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i-1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i-1)k = (m+1)!$$

This is always possible because

$$i = 1 + \frac{m\,m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, L and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\,m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, $\overline{L}$ and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m+1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i-1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i-1)k = (m+1)!$$

This is always possible because

$$i = 1 + \frac{m\,m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, $\overline{L}$ and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\,m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, L and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m+1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i-1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i-1)k = (m+1)!$$

This is always possible because

$$i = 1 + \frac{m\,m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, L and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, $\overline{L}$ and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$.

If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, L and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, $\overline{L}$ and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\,m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, $\overline{L}$ and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m+1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i-1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i-1)k = (m+1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, $\overline{L}$ and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m+1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i-1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i-1)k = (m+1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, $\overline{L}$ and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, $\overline{L}$ and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, $\overline{L}$ and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, $\overline{L}$ and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, $\overline{L}$ and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, $\overline{L}$ and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, $\overline{L}$ and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, L and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, L and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, L and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m+1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i-1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i-1)k = (m+1)!$$

This is always possible because

$$i = 1 + \frac{m\, m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, L and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\,m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, L and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\,m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, L and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

Example 4.13

Let $\Sigma = \{a, b\}$. The language

$$L = \{a^n b^l : n \neq l\}$$

is not regular.

Here we need a bit of ingenuity to apply the Pumping Lemma directly. Choosing a string with $n = l + 1$ or $n = l + 2$ will not do, because our opponent can always choose a decomposition that will make it impossible to pump the string out of the language (that is, pump it so that it has an equal number of a's and b's). We must be more inventive. Let us take $n = m!$ and $l = (m + 1)!$. If the opponent now chooses a string y (by necessity consisting of all a's) of length $k < m$, we pump i times to generate a string with $m! + (i - 1)k$ a's. We can get a contradiction of the Pumping Lemma if we can pick i such that

$$m! + (i - 1)k = (m + 1)!$$

This is always possible because

$$i = 1 + \frac{m\,m!}{k}$$

and $k \leqslant m$. The right side is therefore an integer, and we have succeeded in violating the conditions of the Pumping Lemma.

However, there is a much more elegant way of solving this problem. Suppose L were regular. Then by Theorem 4.1, L and the language

$$L_1 = \overline{L} \cap L(a^* b^*)$$

would also be regular. But $L_1 = \{a^n b^n : n \geqslant 0\}$, which we have already classified as nonregular. Consequently, L cannot be regular.

The Pumping Lemma is difficult to understand and it is easy to go astray when applying it. Here are some common pitfalls. Watch out for them.

One mistake is to try using the Pumping Lemma to show that a language is regular. Even if you can show that no string in a language L can ever be pumped out, you cannot conclude that L is regular. The Pumping Lemma can only be used to prove that a language is not regular.

Another mistake is to start (usually inadvertently) with a string not in L. For example, suppose we try to show that

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular. An argument that starts with "Given m, let $w = a^m \ldots$," is incorrect because m is not necessarily prime. To avoid this pitfall, we need to start with something like "Given m, let $w = a^M$, where M is a prime number larger than m."

The Pumping Lemma is difficult to understand and it is easy to go astray when applying it. Here are some common pitfalls. Watch out for them.

One mistake is to try using the Pumping Lemma to show that a language is regular. Even if you can show that no string in a language L can ever be pumped out, you cannot conclude that L is regular. The Pumping Lemma can only be used to prove that a language is not regular.

Another mistake is to start (usually inadvertently) with a string not in L. For example, suppose we try to show that

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular. An argument that starts with "Given m, let $w = a^m \ldots$," is incorrect because m is not necessarily prime. To avoid this pitfall, we need to start with something like "Given m, let $w = a^M$, where M is a prime number larger than m."

The Pumping Lemma is difficult to understand and it is easy to go astray when applying it. Here are some common pitfalls. Watch out for them.

One mistake is to try using the Pumping Lemma to show that a language is regular. Even if you can show that no string in a language L can ever be pumped out, you cannot conclude that L is regular. The Pumping Lemma can only be used to prove that a language is not regular.

Another mistake is to start (usually inadvertently) with a string not in L. For example, suppose we try to show that

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular. An argument that starts with "Given m, let $w = a^m \ldots$," is incorrect because m is not necessarily prime. To avoid this pitfall, we need to start with something like "Given m, let $w = a^M$, where M is a prime number larger than m."

The Pumping Lemma is difficult to understand and it is easy to go astray when applying it. Here are some common pitfalls. Watch out for them.

One mistake is to try using the Pumping Lemma to show that a language is regular. Even if you can show that no string in a language L can ever be pumped out, you cannot conclude that L is regular. The Pumping Lemma can only be used to prove that a language is not regular.

Another mistake is to start (usually inadvertently) with a string not in L. For example, suppose we try to show that

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular. An argument that starts with "Given m, let $w = a^m \ldots$," is incorrect because m is not necessarily prime. To avoid this pitfall, we need to start with something like "Given m, let $w = a^M$, where M is a prime number larger than m."

The Pumping Lemma is difficult to understand and it is easy to go astray when applying it. Here are some common pitfalls. Watch out for them.

One mistake is to try using the Pumping Lemma to show that a language is regular. Even if you can show that no string in a language L can ever be pumped out, you cannot conclude that L is regular. The Pumping Lemma can only be used to prove that a language is not regular.

Another mistake is to start (usually inadvertently) with a string not in L. For example, suppose we try to show that

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular. An argument that starts with "Given m, let $w = a^m \ldots$," is incorrect because m is not necessarily prime. To avoid this pitfall, we need to start with something like "Given m, let $w = a^M$, where M is a prime number larger than m."

The Pumping Lemma is difficult to understand and it is easy to go astray when applying it. Here are some common pitfalls. Watch out for them.

One mistake is to try using the Pumping Lemma to show that a language is regular. Even if you can show that no string in a language L can ever be pumped out, you cannot conclude that L is regular. The Pumping Lemma can only be used to prove that a language is not regular.

Another mistake is to start (usually inadvertently) with a string not in L. For example, suppose we try to show that

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular. An argument that starts with "Given m, let $w = a^m \ldots$," is incorrect because m is not necessarily prime. To avoid this pitfall, we need to start with something like "Given m, let $w = a^M$, where M is a prime number larger than m."

4.3 Identifying Nonregular Languages

The Pumping Lemma is difficult to understand and it is easy to go astray when applying it. Here are some common pitfalls. Watch out for them.

One mistake is to try using the Pumping Lemma to show that a language is regular. Even if you can show that no string in a language L can ever be pumped out, you cannot conclude that L is regular. The Pumping Lemma can only be used to prove that a language is not regular.

Another mistake is to start (usually inadvertently) with a string not in L. For example, suppose we try to show that

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular. An argument that starts with "Given m, let $w = a^m \ldots$," is incorrect because m is not necessarily prime. To avoid this pitfall, we need to start with something like "Given m, let $w = a^M$, where M is a prime number larger than m."

The Pumping Lemma is difficult to understand and it is easy to go astray when applying it. Here are some common pitfalls. Watch out for them.

One mistake is to try using the Pumping Lemma to show that a language is regular. Even if you can show that no string in a language L can ever be pumped out, you cannot conclude that L is regular. The Pumping Lemma can only be used to prove that a language is not regular.

Another mistake is to start (usually inadvertently) with a string not in L. For example, suppose we try to show that

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular. An argument that starts with "Given m, let $w = a^m \ldots$," is incorrect because m is not necessarily prime. To avoid this pitfall, we need to start with something like "Given m, let $w = a^M$, where M is a prime number larger than m."

The Pumping Lemma is difficult to understand and it is easy to go astray when applying it. Here are some common pitfalls. Watch out for them.

One mistake is to try using the Pumping Lemma to show that a language is regular. Even if you can show that no string in a language L can ever be pumped out, you cannot conclude that L is regular. The Pumping Lemma can only be used to prove that a language is not regular.

Another mistake is to start (usually inadvertently) with a string not in L. For example, suppose we try to show that

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular. An argument that starts with "Given m, let $w = a^m \ldots$," is incorrect because m is not necessarily prime. To avoid this pitfall, we need to start with something like "Given m, let $w = a^M$, where M is a prime number larger than m."

The Pumping Lemma is difficult to understand and it is easy to go astray when applying it. Here are some common pitfalls. Watch out for them.

One mistake is to try using the Pumping Lemma to show that a language is regular. Even if you can show that no string in a language L can ever be pumped out, you cannot conclude that L is regular. The Pumping Lemma can only be used to prove that a language is not regular.

Another mistake is to start (usually inadvertently) with a string not in L. For example, suppose we try to show that

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular. An argument that starts with "Given m, let $w = a^m \ldots$," is incorrect because m is not necessarily prime. To avoid this pitfall, we need to start with something like "Given m, let $w = a^M$, where M is a prime number larger than m."

The Pumping Lemma is difficult to understand and it is easy to go astray when applying it. Here are some common pitfalls. Watch out for them.

One mistake is to try using the Pumping Lemma to show that a language is regular. Even if you can show that no string in a language L can ever be pumped out, you cannot conclude that L is regular. The Pumping Lemma can only be used to prove that a language is not regular.

Another mistake is to start (usually inadvertently) with a string not in L. For example, suppose we try to show that

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular. An argument that starts with "Given m, let $w = a^m \ldots$," is incorrect because m is not necessarily prime. To avoid this pitfall, we need to start with something like "Given m, let $w = a^M$, where M is a prime number larger than m."

The Pumping Lemma is difficult to understand and it is easy to go astray when applying it. Here are some common pitfalls. Watch out for them.

One mistake is to try using the Pumping Lemma to show that a language is regular. Even if you can show that no string in a language L can ever be pumped out, you cannot conclude that L is regular. The Pumping Lemma can only be used to prove that a language is not regular.

Another mistake is to start (usually inadvertently) with a string not in L. For example, suppose we try to show that

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular. An argument that starts with "Given m, let $w = a^m \ldots$," is incorrect because m is not necessarily prime. To avoid this pitfall, we need to start with something like "Given m, let $w = a^M$, where M is a prime number larger than m."

The Pumping Lemma is difficult to understand and it is easy to go astray when applying it. Here are some common pitfalls. Watch out for them.

One mistake is to try using the Pumping Lemma to show that a language is regular. Even if you can show that no string in a language L can ever be pumped out, you cannot conclude that L is regular. The Pumping Lemma can only be used to prove that a language is not regular.

Another mistake is to start (usually inadvertently) with a string not in L. For example, suppose we try to show that

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular. An argument that starts with "Given m, let $w = a^m \ldots$," is incorrect because m is not necessarily prime. To avoid this pitfall, we need to start with something like "Given m, let $w = a^M$, where M is a prime number larger than m."

The Pumping Lemma is difficult to understand and it is easy to go astray when applying it. Here are some common pitfalls. Watch out for them.

One mistake is to try using the Pumping Lemma to show that a language is regular. Even if you can show that no string in a language L can ever be pumped out, you cannot conclude that L is regular. The Pumping Lemma can only be used to prove that a language is not regular.

Another mistake is to start (usually inadvertently) with a string not in L. For example, suppose we try to show that

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular. An argument that starts with "Given m, let $w = a^m \ldots$," is incorrect because m is not necessarily prime. To avoid this pitfall, we need to start with something like "Given m, let $w = a^M$, where M is a prime number larger than m."

The Pumping Lemma is difficult to understand and it is easy to go astray when applying it. Here are some common pitfalls. Watch out for them.

One mistake is to try using the Pumping Lemma to show that a language is regular. Even if you can show that no string in a language L can ever be pumped out, you cannot conclude that L is regular. The Pumping Lemma can only be used to prove that a language is not regular.

Another mistake is to start (usually inadvertently) with a string not in L. For example, suppose we try to show that

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular. An argument that starts with "Given m, let $w = a^m \ldots$," is incorrect because m is not necessarily prime. To avoid this pitfall, we need to start with something like "Given m, let $w = a^M$, where M is a prime number larger than m."

Finally, perhaps the most common mistake is to make some assumptions about the decomposition xyz. The only thing we can say about the decomposition is what the Pumping Lemma tells us, namely, that y is not empty and that $|xy| \leqslant m$; that is, that y must be within m symbols of the left end of the string. Anything else makes the argument invalid. A typical mistake in trying to prove that the language in Equation (2)

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular is to say that $y = a^k$, with k odd. Then of course $w = xz$ is an even-length string and thus not in L. But the assumption on k is not permitted and the proof is wrong.

But even if you master the technical difficulties of the Pumping Lemma, it may still be hard to see exactly how to use it. The Pumping Lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win. If you can apply the Pumping Lemma correctly to some of the more difficult cases in this course of lectures, you are to be congratulated.

Finally, perhaps the most common mistake is to make some assumptions about the decomposition xyz. The only thing we can say about the decomposition is what the Pumping Lemma tells us, namely, that y is not empty and that $|xy| \leqslant m$; that is, that y must be within m symbols of the left end of the string. Anything else makes the argument invalid. A typical mistake in trying to prove that the language in Equation (2)

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular is to say that $y = a^k$, with k odd. Then of course $w = xz$ is an even-length string and thus not in L. But the assumption on k is not permitted and the proof is wrong.

But even if you master the technical difficulties of the Pumping Lemma, it may still be hard to see exactly how to use it. The Pumping Lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win. If you can apply the Pumping Lemma correctly to some of the more difficult cases in this course of lectures, you are to be congratulated.

Finally, perhaps the most common mistake is to make some assumptions about the decomposition xyz. The only thing we can say about the decomposition is what the Pumping Lemma tells us, namely, that y is not empty and that $|xy| \leqslant m$; that is, that y must be within m symbols of the left end of the string. Anything else makes the argument invalid. A typical mistake in trying to prove that the language in Equation (2)

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular is to say that $y = a^k$, with k odd. Then of course $w = xz$ is an even-length string and thus not in L. But the assumption on k is not permitted and the proof is wrong.

But even if you master the technical difficulties of the Pumping Lemma, it may still be hard to see exactly how to use it. The Pumping Lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win. If you can apply the Pumping Lemma correctly to some of the more difficult cases in this course of lectures, you are to be congratulated.

Finally, perhaps the most common mistake is to make some assumptions about the decomposition xyz. The only thing we can say about the decomposition is what the Pumping Lemma tells us, namely, that y is not empty and that $|xy| \leqslant m$; that is, that y must be within m symbols of the left end of the string. Anything else makes the argument invalid. A typical mistake in trying to prove that the language in Equation (2)

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular is to say that $y = a^k$, with k odd. Then of course $w = xz$ is an even-length string and thus not in L. But the assumption on k is not permitted and the proof is wrong.

But even if you master the technical difficulties of the Pumping Lemma, it may still be hard to see exactly how to use it. The Pumping Lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win. If you can apply the Pumping Lemma correctly to some of the more difficult cases in this course of lectures, you are to be congratulated.

Finally, perhaps the most common mistake is to make some assumptions about the decomposition xyz. The only thing we can say about the decomposition is what the Pumping Lemma tells us, namely, that y is not empty and that $|xy| \leqslant m$; that is, that y must be within m symbols of the left end of the string. Anything else makes the argument invalid. A typical mistake in trying to prove that the language in Equation (2)

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular is to say that $y = a^k$, with k odd. Then of course $w = xz$ is an even-length string and thus not in L. But the assumption on k is not permitted and the proof is wrong.

But even if you master the technical difficulties of the Pumping Lemma, it may still be hard to see exactly how to use it. The Pumping Lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win. If you can apply the Pumping Lemma correctly to some of the more difficult cases in this course of lectures, you are to be congratulated.

Finally, perhaps the most common mistake is to make some assumptions about the decomposition xyz. The only thing we can say about the decomposition is what the Pumping Lemma tells us, namely, that y is not empty and that $|xy| \leqslant m$; that is, that y must be within m symbols of the left end of the string. Anything else makes the argument invalid. A typical mistake in trying to prove that the language in Equation (2)

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular is to say that $y = a^k$, with k odd. Then of course $w = xz$ is an even-length string and thus not in L. But the assumption on k is not permitted and the proof is wrong.

But even if you master the technical difficulties of the Pumping Lemma, it may still be hard to see exactly how to use it. The Pumping Lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win. If you can apply the Pumping Lemma correctly to some of the more difficult cases in this course of lectures, you are to be congratulated.

Finally, perhaps the most common mistake is to make some assumptions about the decomposition xyz. The only thing we can say about the decomposition is what the Pumping Lemma tells us, namely, that y is not empty and that $|xy| \leqslant m$; that is, that y must be within m symbols of the left end of the string. Anything else makes the argument invalid. A typical mistake in trying to prove that the language in Equation (2)

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular is to say that $y = a^k$, with k odd. Then of course $w = xz$ is an even-length string and thus not in L. But the assumption on k is not permitted and the proof is wrong.

But even if you master the technical difficulties of the Pumping Lemma, it may still be hard to see exactly how to use it. The Pumping Lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win. If you can apply the Pumping Lemma correctly to some of the more difficult cases in this course of lectures, you are to be congratulated.

Finally, perhaps the most common mistake is to make some assumptions about the decomposition xyz. The only thing we can say about the decomposition is what the Pumping Lemma tells us, namely, that y is not empty and that $|xy| \leqslant m$; that is, that y must be within m symbols of the left end of the string. Anything else makes the argument invalid. A typical mistake in trying to prove that the language in Equation (2)

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular is to say that $y = a^k$, with k odd. Then of course $w = xz$ is an even-length string and thus not in L. But the assumption on k is not permitted and the proof is wrong.

But even if you master the technical difficulties of the Pumping Lemma, it may still be hard to see exactly how to use it. The Pumping Lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win. If you can apply the Pumping Lemma correctly to some of the more difficult cases in this course of lectures, you are to be congratulated.

Finally, perhaps the most common mistake is to make some assumptions about the decomposition xyz. The only thing we can say about the decomposition is what the Pumping Lemma tells us, namely, that y is not empty and that $|xy| \leqslant m$; that is, that y must be within m symbols of the left end of the string. Anything else makes the argument invalid. A typical mistake in trying to prove that the language in Equation (2)

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular is to say that $y = a^k$, with k odd. Then of course $w = xz$ is an even-length string and thus not in L. But the assumption on k is not permitted and the proof is wrong.

But even if you master the technical difficulties of the Pumping Lemma, it may still be hard to see exactly how to use it. The Pumping Lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win. If you can apply the Pumping Lemma correctly to some of the more difficult cases in this course of lectures, you are to be congratulated.

Finally, perhaps the most common mistake is to make some assumptions about the decomposition xyz. The only thing we can say about the decomposition is what the Pumping Lemma tells us, namely, that y is not empty and that $|xy| \leqslant m$; that is, that y must be within m symbols of the left end of the string. Anything else makes the argument invalid. A typical mistake in trying to prove that the language in Equation (2)

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular is to say that $y = a^k$, with k odd. Then of course $w = xz$ is an even-length string and thus not in L. But the assumption on k is not permitted and the proof is wrong.

But even if you master the technical difficulties of the Pumping Lemma, it may still be hard to see exactly how to use it. The Pumping Lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win. If you can apply the Pumping Lemma correctly to some of the more difficult cases in this course of lectures, you are to be congratulated.

Finally, perhaps the most common mistake is to make some assumptions about the decomposition xyz. The only thing we can say about the decomposition is what the Pumping Lemma tells us, namely, that y is not empty and that $|xy| \leqslant m$; that is, that y must be within m symbols of the left end of the string. Anything else makes the argument invalid. A typical mistake in trying to prove that the language in Equation (2)

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular is to say that $y = a^k$, with k odd. Then of course $w = xz$ is an even-length string and thus not in L. But the assumption on k is not permitted and the proof is wrong.

But even if you master the technical difficulties of the Pumping Lemma, it may still be hard to see exactly how to use it. The Pumping Lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win. If you can apply the Pumping Lemma correctly to some of the more difficult cases in this course of lectures, you are to be congratulated.

Finally, perhaps the most common mistake is to make some assumptions about the decomposition xyz. The only thing we can say about the decomposition is what the Pumping Lemma tells us, namely, that y is not empty and that $|xy| \leqslant m$; that is, that y must be within m symbols of the left end of the string. Anything else makes the argument invalid. A typical mistake in trying to prove that the language in Equation (2)

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular is to say that $y = a^k$, with k odd. Then of course $w = xz$ is an even-length string and thus not in L. But the assumption on k is not permitted and the proof is wrong.

But even if you master the technical difficulties of the Pumping Lemma, it may still be hard to see exactly how to use it. The Pumping Lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win. If you can apply the Pumping Lemma correctly to some of the more difficult cases in this course of lectures, you are to be congratulated.

Finally, perhaps the most common mistake is to make some assumptions about the decomposition xyz. The only thing we can say about the decomposition is what the Pumping Lemma tells us, namely, that y is not empty and that $|xy| \leqslant m$; that is, that y must be within m symbols of the left end of the string. Anything else makes the argument invalid. A typical mistake in trying to prove that the language in Equation (2)

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular is to say that $y = a^k$, with k odd. Then of course $w = xz$ is an even-length string and thus not in L. But the assumption on k is not permitted and the proof is wrong.

But even if you master the technical difficulties of the Pumping Lemma, it may still be hard to see exactly how to use it. The Pumping Lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win. If you can apply the Pumping Lemma correctly to some of the more difficult cases in this course of lectures, you are to be congratulated.

Finally, perhaps the most common mistake is to make some assumptions about the decomposition xyz. The only thing we can say about the decomposition is what the Pumping Lemma tells us, namely, that y is not empty and that $|xy| \leqslant m$; that is, that y must be within m symbols of the left end of the string. Anything else makes the argument invalid. A typical mistake in trying to prove that the language in Equation (2)

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular is to say that $y = a^k$, with k odd. Then of course $w = xz$ is an even-length string and thus not in L. But the assumption on k is not permitted and the proof is wrong.

But even if you master the technical difficulties of the Pumping Lemma, it may still be hard to see exactly how to use it. The Pumping Lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win. If you can apply the Pumping Lemma correctly to some of the more difficult cases in this course of lectures, you are to be congratulated.

Finally, perhaps the most common mistake is to make some assumptions about the decomposition xyz. The only thing we can say about the decomposition is what the Pumping Lemma tells us, namely, that y is not empty and that $|xy| \leqslant m$; that is, that y must be within m symbols of the left end of the string. Anything else makes the argument invalid. A typical mistake in trying to prove that the language in Equation (2)

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular is to say that $y = a^k$, with k odd. Then of course $w = xz$ is an even-length string and thus not in L. But the assumption on k is not permitted and the proof is wrong.

But even if you master the technical difficulties of the Pumping Lemma, it may still be hard to see exactly how to use it. The Pumping Lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win. If you can apply the Pumping Lemma correctly to some of the more difficult cases in this course of lectures, you are to be congratulated.

Finally, perhaps the most common mistake is to make some assumptions about the decomposition xyz. The only thing we can say about the decomposition is what the Pumping Lemma tells us, namely, that y is not empty and that $|xy| \leqslant m$; that is, that y must be within m symbols of the left end of the string. Anything else makes the argument invalid. A typical mistake in trying to prove that the language in Equation (2)

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular is to say that $y = a^k$, with k odd. Then of course $w = xz$ is an even-length string and thus not in L. But the assumption on k is not permitted and the proof is wrong.

But even if you master the technical difficulties of the Pumping Lemma, it may still be hard to see exactly how to use it. The Pumping Lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win. If you can apply the Pumping Lemma correctly to some of the more difficult cases in this course of lectures, you are to be congratulated.

Finally, perhaps the most common mistake is to make some assumptions about the decomposition xyz. The only thing we can say about the decomposition is what the Pumping Lemma tells us, namely, that y is not empty and that $|xy| \leqslant m$; that is, that y must be within m symbols of the left end of the string. Anything else makes the argument invalid. A typical mistake in trying to prove that the language in Equation (2)

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular is to say that $y = a^k$, with k odd. Then of course $w = xz$ is an even-length string and thus not in L. But the assumption on k is not permitted and the proof is wrong.

But even if you master the technical difficulties of the Pumping Lemma, it may still be hard to see exactly how to use it. The Pumping Lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win. If you can apply the Pumping Lemma correctly to some of the more difficult cases in this course of lectures, you are to be congratulated.

Finally, perhaps the most common mistake is to make some assumptions about the decomposition xyz. The only thing we can say about the decomposition is what the Pumping Lemma tells us, namely, that y is not empty and that $|xy| \leqslant m$; that is, that y must be within m symbols of the left end of the string. Anything else makes the argument invalid. A typical mistake in trying to prove that the language in Equation (2)

$$L = \{a^n : n \text{ is a prime number}\} \tag{2}$$

is not regular is to say that $y = a^k$, with k odd. Then of course $w = xz$ is an even-length string and thus not in L. But the assumption on k is not permitted and the proof is wrong.

But even if you master the technical difficulties of the Pumping Lemma, it may still be hard to see exactly how to use it. The Pumping Lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win. If you can apply the Pumping Lemma correctly to some of the more difficult cases in this course of lectures, you are to be congratulated.

Thank You for attention!