

Languages that are not Regular

[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [12](#) [13](#)



Languages that are not Regular

Regular and Nonregular Languages

Construct a DFA that accepts the language:

$$L = \{x \mid \text{the difference between the number of } a\text{'s and } b\text{'s in any prefix of } x \text{ does not exceed } 1\}$$

over the alphabet $\Sigma = \{a, b\}$.

[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [12](#) [13](#)

Languages that are not Regular

Regular and Nonregular Languages

How would the DFA need to be changed so that the difference between the number of a 's and b 's in any prefix does not exceed 2?

How would the DFA need to be changed so that we don't care about the relative number of a 's and b 's in the prefix, only in the whole string: i.e., construct a DFA that accepts

$$L = \{x \mid \text{the difference between the number of } a\text{'s and } b\text{'s does not exceed } 1\}$$

If it's too difficult to construct a DFA, try constructing an NFA.

[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [12](#) [13](#)

Languages that are not Regular

Regular and Nonregular Languages

As you have probably discovered, it is not easy to construct a finite automaton that accepts this last language.

What is the difficulty in trying to construct a finite automaton that accepts the language

$$L = \{a^n b^n \mid n \geq 0\}?$$

[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [12](#) [13](#)

Languages that are not Regular

[prev](#) | [slides](#) | [next](#)

Regular and Nonregular Languages

As we've already discussed, a language is regular if and only if it is accepted by some finite automaton. The fact that we've found a couple of languages that we can't find a finite automaton for suggests that these languages may not be regular.

The question is, however,

is there really some finite automaton that does accept these languages but that we can't easily construct,

or is it really impossible to ever construct a finite automaton that accepts a language like these last ones we've examined?

1 2 3 4 5 6 7 8 9 10 11 12 13

Languages that are not Regular

[prev](#) | [slides](#) | [next](#)

The Pumping Lemma

We don't know if each state is entered, but one thing we do know is that some state is entered at least twice.

Suppose that the state q is entered at least twice, and suppose that this occurs when the j^{th} and k^{th} symbols of w are processed.

Thus, the DFA is in state q immediately after processing $w(j)$ and again immediately after processing $w(k)$.

Let us write w as xyz where

$$x = w(1)..w(j), \quad y = w(j+1)..w(k), \quad z = w(k+1)..w(n)$$

where $n = |w|$.

1 2 3 4 5 6 7 8 9 10 11 12 13

Languages that are not Regular

[prev](#) | [slides](#) | [next](#)

The Pumping Lemma

As it turns out, there is a theorem that can be used in an indirect proof (or a proof by contradiction) to show that some languages are not regular.

Consider a deterministic finite automata with N states and recall the pigeonhole principle. What must be true about the number of times each state is visited if a string w with $|w| > N$ is processed by this DFA?

1 2 3 4 5 6 7 8 9 10 11 12 13

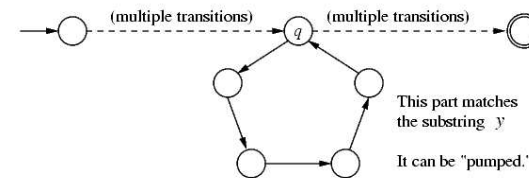
Languages that are not Regular

[prev](#) | [slides](#) | [next](#)

The Pumping Lemma

Clearly, if $w = xyz$ is accepted by the DFA then the strings $xyyz$ and $xyyyz$ will be as well, since at the end of each y substring the DFA will be in state q_k .

In fact, any string of the form $xy^i z$, $i \geq 0$, will also be accepted.



1 2 3 4 5 6 7 8 9 10 11 12 13

Languages that are not Regular

[prev](#) | [slides](#) | [next](#)

The Pumping Lemma

The formal statement of the Pumping Lemma is

Theorem: Let L be a regular language. There is an integer $n \geq 1$ such that any string w in L with $|w| \geq n$ can be rewritten as $w = xyz$ such that $|y| > 0$, $|xz| \leq n$, and $xy^i z$ is in L for each $i \geq 0$.

This theorem (a *lemma* is a little theorem, usually used to support other, more important theorems) is often used to show a language is not regular. This is strange, is it not, since the pumping lemma refers to regular languages?

1 2 3 4 5 6 7 8 9 10 11 12 13

Languages that are not Regular

[prev](#) | [slides](#) | [next](#)

The Pumping Lemma

Proof: (continued) The substring $y = w(i) \dots w(j)$ is nonempty because $i < j$.

Further, because this substring can be removed or repeated in place any number of times without changing whether or not M accepts w , we know that $xy^i z$ will be accepted for all $i \geq 0$.

Finally, if $w = xyz$ then $|xz| \leq n$ because otherwise the string xz could itself be "pumped."

1 2 3 4 5 6 7 8 9 10 11 12 13

Languages that are not Regular

[prev](#) | [slides](#) | [next](#)

The Pumping Lemma

Proof: Because L is regular there is a DFA M that accepts it. Let n be the number of states in that DFA and consider a string w with $|w| \geq n$. Let w_k denote the suffix of w beginning with the k^{th} symbol of w . The first $n+1$ configurations as M processes w look like

$$(q_0, w_1) \vdash (q_1, w_2) \vdash \dots \vdash (q_i, w_{i+1}) \vdash \dots \vdash (q_j, w_{j+1}) \vdash \dots \vdash (q_n, w_{n+1})$$

where $w_{n+1} = \Lambda$ if $|w| = n$.

As M processes w it must be the case that there are integers $0 \leq i < j \leq n$ such that $q_i = q_j$. This follows from the pigeonhole principle and the fact that we have visited $n+1$ states while there are only n states in M .

1 2 3 4 5 6 7 8 9 10 11 12 13

Languages that are not Regular

[prev](#) | [slides](#) | [next](#)

Example

As already mentioned, the prime use of the pumping lemma is to show that certain languages are not regular.

For example one language we considered earlier was $L = \{a^n b^n \mid n \geq 0\}$. We can now show that this language is not regular, and hence it is impossible to find a finite automata that accepts it.

Let's begin by assuming that L is regular and look for a contradiction. If L is regular then there is an n state DFA that accepts it. Take $w = a^i b^i$ so that $2i \geq n$.

1 2 3 4 5 6 7 8 9 10 11 12 13

Languages that are not Regular

[prev](#) | [slides](#) | [next](#)

Example

According to the pumping lemma there is a way to partition w into xyz so that y is nonempty and can be removed or repeated any number of times. There are three cases:

1. y contains only a 's. In this case repeating y leads to an imbalance in the number of a 's and b 's, so that strings like $xyyz$ are not in L .
2. y contains only b 's. This is the same as the last case; there will be an imbalance the number of a 's and b 's.
3. y contains both a 's and b 's. If y is repeated then there will be b 's in front of a 's in the string, so the string cannot be in L .

No matter what part of the string $a^i b^i$ we choose for y , it is impossible to pump y . Thus L must not be a regular language.

[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [12](#) [13](#)