Pumping Lemma

Limitations of the Regular Languages The Pumping Lemma

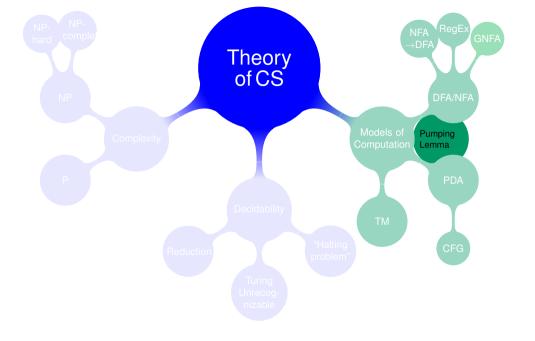
Dr Kamal Bentahar

School of Computing, Electronics and Mathematics Coventry University

Lecture 4

Pigeon-hole principle

 $a^n b^n$



Pumping Lemma

Mindmap

Proofs

roof by existence

Observation

Unary alphabet
Pigeon-hole principle
Binary alphabet

Pumping

PL Game! Examples a"b"

ww Pumping down

nlications

nplications

Regular Languages

The class of regular languages can be:

- **1** Recognized by NFAs. (equiv. GNFA or ε -NFA or NFA or DFA).
- Described using Regular Expressions.

Today:

- See the limit of regular languages.
- 2 How to show a language is not regular.

Mindmap

Proof b

Proof by existence Proof by contradiction

Observation
Unary alphabet

lnary alphabet igeon-hole principle inary alphabet

> Game! mples

a"b"

w mping down

plications onstant Space

 $a^n b^n$

We show a language is regular using "proof by existence":

- Construct an NFA recognizing it.
- Write a Regular Expression for it. Using closure under the union, concatenation and star operations.

We show a language is regular using "proof by existence":

- Construct an NFA recognizing it.
- Write a Regular Expression for it. Using closure under the union, concatenation and star operations.

However, if a languages is *not regular* then how can we show that?!

Is it raining now? – example of proof by contradiction

Is it raining now?

Pumping Lemma

Proof by

contradiction

Pigeon-hole principle Binary alphabet

 $a^n b^n$ ww

Proof by contradiction

Pigeon-hole principle Binary alphabet

 $a^n b^n$

Suppose it is.

Proof by contradiction

 $a^n b^n$

- Is it raining now?
- Suppose it is.
- Let us go outside where it is supposed to be raining.

- Is it raining now?
- Suppose it is.
- Let us go outside where it is supposed to be raining.
 - If it is raining then we should get wet. (No umberlla, etc.)

Mindmap

Proofs

Proof by existe Proof by contradiction

Observation

Unary alphabet
Pigeon-hole princip

Pumping

Examples a"b"

vw

ımping down

nlications

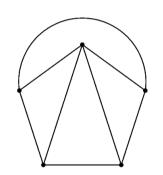
- Is it raining now?
- Suppose it is.
- Let us go outside where it is supposed to be raining.
 - If it is raining then we should get wet. (No umberlla, etc.)
- However, we did not get wet!

Proof by contradiction

 $a^n b^n$

- Is it raining now?
- Suppose it is.
- Let us go outside where it is supposed to be raining.
 - If it is raining then we should get wet. (No umberlla, etc.)
- However, we did not get wet!
- Thus, it is **not** raining!

Is it possible to traverse this graph by travelling along each edge exactly once?



Pumping Lemma

/lindmap

Proofs

Proof by contradiction

Observation

Unary alphabet
Pigeon-hole principle
Binary alphabet

umping

Examples a"b"

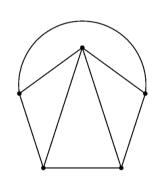
vw

umping down

plications

Is it possible to traverse this graph by travelling along each edge exactly once?

Suppose it is possible.



Pumping Lemma

Mindmap

Proofs

Proof by existen

Proof by

contradiction

Observation

Unary alphabet
Pigeon-hole principle
Binary alphabet

umping

PL Game! Examples

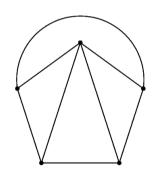
a"b" ww

v ımping dowr

plications

Constant Space

Is it possible to traverse this graph by travelling along each edge exactly once?



- Suppose it is possible.
- How many times would each vertex be visited?

Pumping Lemma

/lindmap

Proofs

Proof by existence Proof by contradiction

Observation

Unary alphabet
Pigeon-hole principle
Binary alphabet

Pumping

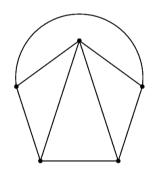
PL Game! Examples

aⁿbⁿ ww

imping down

mplications

Is it possible to traverse this graph by travelling along each edge exactly once?

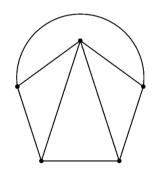


- Suppose it is possible.
- How many times would each vertex be visited?
 - Every time a vertex is entered, it is also exited.

Proof by contradiction

 $a^n b^n$

Is it possible to traverse this graph by travelling along each edge exactly once?



- Suppose it is possible.
- How many times would each vertex be visited?
 - Every time a vertex is entered, it is also exited.
 - So, each vertex must have an even number of neighbours.

Mindmap

Proofs

Proof by contradiction

Observation

Unary alphabet
Pigeon-hole principle
Binary alphabet

Pumping

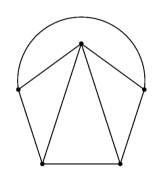
emma L Game!

aⁿbⁿ

mpina down

nplications

Constant Space



- Suppose it is possible.
- How many times would each vertex be visited?
 - Every time a vertex is entered, it is also exited.
 - So, each vertex must have an even number of neighbours.
 - The starting and ending vertices are exceptions: odd number of neighbours.

viinamap

Proofs

Proof by contradiction

Observation

Unary alphabet
Pigeon-hole principle

Pumping

emma PL Game! Examples

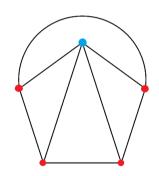
aⁿbⁿ ww

w umping down

plications

plications
onstant Space

Is it possible to traverse this graph by travelling along each edge exactly once?



- Suppose it is possible.
- How many times would each vertex be visited?
 - Every time a vertex is entered, it is also exited.
 - So, each vertex must have an even number of neighbours.
 - The starting and ending vertices are exceptions: odd number of neighbours.
 - There can only be 0 or 2 such exceptions.

Pumping Lemma

Иindmaр

Proofs

Proof by contradiction

Observation

Unary alphabet
Pigeon-hole principle
Binary alphabet

Pumping

.emma PL Game! Examples

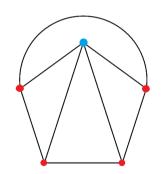
aⁿbⁿ ww

'**w** umping down

nplications

plications

Is it possible to traverse this graph by travelling along each edge exactly once?

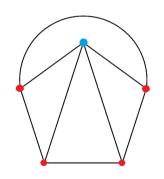


- Suppose it is possible.
- How many times would each vertex be visited?
 - Every time a vertex is entered, it is also exited.
 - So, each vertex must have an even number of neighbours.
 - The **starting** and **ending** vertices are exceptions: odd number of neighbours.
 - There can only be 0 or 2 such exceptions.
- However, this graph has 4 exceptions!

Proof by contradiction

 $a^n b^n$

Is it possible to traverse this graph by travelling along each edge exactly once?



- Suppose it is possible.
- How many times would each vertex be visited?
 - Every time a vertex is entered, it is also exited.
 - So, each vertex must have an even number of neighbours.
 - The starting and ending vertices are exceptions: odd number of neighbours.
 - There can only be 0 or 2 such exceptions.
- However, this graph has 4 exceptions!
- Thus, it is impossible to traverse this graph by travelling along each path exactly once.

/lindmap

Proofs

Proof by contradiction

Observation

Unary alphabet
Pigeon-hole principle
Binary alphabet

Pumping

emma PL Game! Examples

aⁿbⁿ ww

umping down

nplications constant Space To prove a language is not regular, we can use **proof by contradiction**.

- We need a **property** that all regular languages must satisfy.
- Then, if a given language does not satisfy it then it cannot be regular.

- We need a **property** that all regular languages must satisfy.
- Then, if a given language does not satisfy it then it cannot be regular.

Let us try to understand the regular languages (RLs) a bit more...

- Let us examine some examples in the next few slides...
- For each automaton, let us think about the path taken by an accepted string – is it "straight" or does it loop?





Proofs

Proof by exist Proof by contradiction

Observation

Unary alphabet
Pigeon-hole principle

Binary alphabet

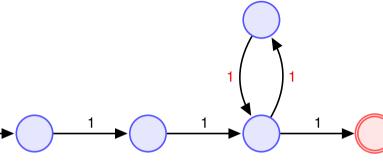
Lemma

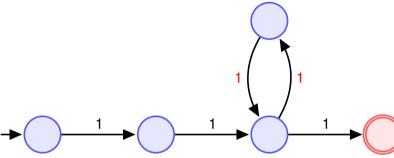
Examples a"b"

ww Pumping dow

umping down

nplications





■ 111 takes a "straight path" to the accept state

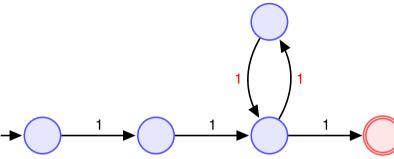
Pumping Lemma

Unary alphabet

Binary alphabet

 $a^n b^n$

ww



- 111 takes a "straight path" to the accept state
- 11111 goes through a loop.

Pumping Lemma

Mindmap

Proofs

Proof by existe
Proof by
contradiction

Observation

Unary alphabet
Pigeon-hole principle

Binary alphabet

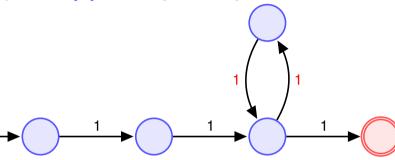
emma PL Game!

aⁿbⁿ

ww

nlications

mplications



- 111 takes a "straight path" to the accept state
- 11111 goes through a loop.
- Repeating the looped part produces longer strings:

```
11 11 11 1, 11 11 11 11, 11 11 11 11, 11
```

Pumping Lemma

Mindmap

Proofs

Proof by exister

Observation

Unary alphabet

Pigeon-hole principle Binary alphabet

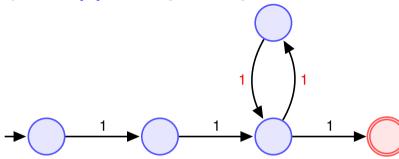
_emma

a"b"

w

nnlications

Constant Space



- 111 takes a "straight path" to the accept state
- 11111 goes through a loop.
- Repeating the looped part produces longer strings:

```
11 11 11 1, 11 11 11 11, 11 11 11 11 11,
```

■ In fact, we can also omit the 11 loop to get: 111.

Pumping Lemma

Mindmap

Proofs

Proof by existe
Proof by
contradiction

Observation

Unary alphabet
Pigeon-hole principl

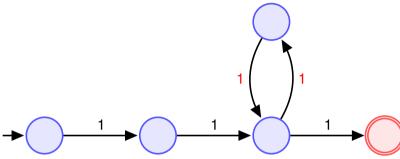
Binary alphabet

PL Game! Examples aⁿbⁿ

w w

umping down

Implications
Constant Space



- 111 takes a "straight path" to the accept state
- 11111 goes through a loop.
- Repeating the looped part produces longer strings:
- In fact, we can also omit the 11 loop to get: 111.

We say: we **pump** the substring 11.

Pumping Lemma

iviiriumaș

Proofs

Proof by contradiction

Observation

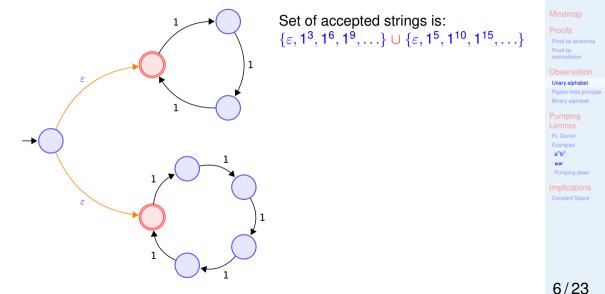
Unary alphabet
Pigeon-hole principle
Binary alphabet

Game! amples

aⁿbⁿ

v Imping down

mplications



 $(111)^* + (11111)^*$

Pumping

Lemma

Pumping

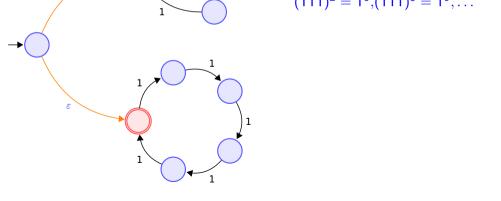
Set of accepted strings is: $\{\varepsilon, 1^3, 1^6, 1^9, \ldots\} \cup \{\varepsilon, 1^5, 1^{10}, 1^{15}, \ldots\}$

■ 111 can be pumped to give: $(111)^0 = \varepsilon, (111)^1 = 1^3,$

$$(111)^3 = \varepsilon, (111)^3 = 1^3, (111)^2 = 1^6, (111)^3 = 1^9, \dots$$

Unary alphabet

 $a^n b^n$



6/23

Unary alphabet

Pumping

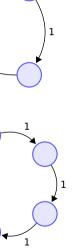
Set of accepted strings is: $\{\varepsilon, 1^3, 1^6, 1^9, \ldots\} \cup \{\varepsilon, 1^5, 1^{10}, 1^{15}, \ldots\}$

- 111 can be pumped to give: $(111)^0 = \varepsilon, (111)^1 = 1^3,$ $(111)^2 = 1^6, (111)^3 = 1^9, \dots$
- 11111 can be pumped to give: $(11111)^0 = \varepsilon, (11111)^1 = 1^5,$

 $(11111)^2 = 1^{10}, (11111)^3 = 1^{15}, \dots$

 $a^n b^n$ ww

6/23



oofs

roof by existe

bservation

Unary alphabet Pigeon-hole principle Binary alphabet

mping nma

Game! amples

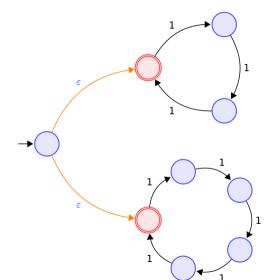
aⁿbⁿ
ww

mping down

lications

Set of accepted strings is: $\{\varepsilon, 1^3, 1^6, 1^9, \ldots\} \cup \{\varepsilon, 1^5, 1^{10}, 1^{15}, \ldots\}$

- 111 can be pumped to give: $(111)^0 = \varepsilon, (111)^1 = 1^3,$ $(111)^2 = 1^6, (111)^3 = 1^9, \dots$
- 11111 can be pumped to give: $(11111)^0 = \varepsilon, (11111)^1 = 1^5,$ $(11111)^2 = 1^{10}, (11111)^3 = 1^{15}, \dots$
- The shortest string that can be pumped is: 111.



Set of accepted strings is:

$$\{\varepsilon, 1^3, 1^6, 1^9, \ldots\} \cup \{\varepsilon, 1^5, 1^{10}, 1^{15}, \ldots\}$$

- 111 can be pumped to give: $(111)^0 = \varepsilon, (111)^1 = 1^3,$ $(111)^2 = 1^6, (111)^3 = 1^9, \dots$
- 11111 can be pumped to give: $(11111)^0 = \varepsilon, (11111)^1 = 1^5,$ $(11111)^2 = 1^{10}, (11111)^3 = 1^{15}, \dots$
- The shortest string that can be pumped is: 111.
- 3, the length of 111, is called: pumping length.

Unary alphabet

 $a^n b^n$ ww

6/23

The language *L* is:

- either **finite**. in which case it is regular trivially.
- or **infinite**. in which case its DFA will have to **loop**:
 - The DFA that recognizes L has a finite number of states.
 - Any string in L determines a path through the DFA.
 - So any sufficiently long string must visit a state twice.
 - This forms a loop.

This looped part can be repeated any arbitrary number of times to produce other strings in *L*.

Pigeon-hole principle

 $a^n b^n$

Unary alphabet

Let L be a regular language over a unary alphabet $\Sigma = \{1\}$.

- The language *L* is:
 - either **finite**, in which case it is regular trivially,
 - or **infinite**, in which case its DFA will have to **loop**:
 - The DFA that recgnizes *L* has a finite number of states.
 - Any string in *L* determines a path through the DFA.
 - So any sufficiently long string must visit a state twice.
 - This forms a loop.

This looped part can be repeated any arbitrary number of times to produce other strings in L.

Pigeon-hole principle

If we put **more than** n pigeons into n holes then there must be a hole with more than one pigeon in.

Pumping Lemma

Mindmap

roofs

Proof by existence
Proof by
contradiction

Unary alphabet

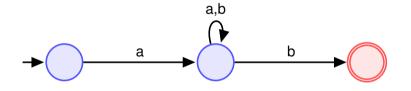
Binary alphabet

PL Game! Examples aⁿbⁿ

.

ımping down

olications estant Space $a\Sigma^*b$



Pumping Lemma

WIIIIGITI

Proofs

Proof by existen
Proof by
contradiction

Observation

Pigeon-hole princip

Binary alphabet

umping

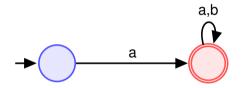
Examples a^nb^n

ww Pumping down

Pumping down

nplications

 $a\Sigma^*$



Pumping Lemma

iviiiiaiii

Proofs

Proof by
Proof by

Observation

Pigeon-hole prin

Binary alphabet

Pumping

PL Game! Examples a"b"

ww

nlications

 Σ^*b



Pumping Lemma

WIIIIGITI

Proofs

Proof by Proof by

Observation

Unary alphabet
Pigeon-hole prince

Binary alphabet

Pumping

emma L Game!

Examples a"b"

ww
Pumping down

plications

 $\Sigma^*1\Sigma$



WIIIIGITI

Proofs

Proof by existe

Observation

Unary alphabet
Pigeon-hole princ

Binary alphabet

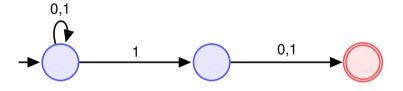
umping

Examples anbn

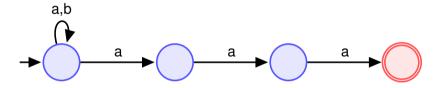
a"b" ww

imping down

anlications



Σ*aaa



Pumping Lemma

WIIIIGITI

Proofs

Proof by existen
Proof by
contradiction

Observation

Unary alphabet
Pigeon-hole princip

Binary alphabet

Pumping

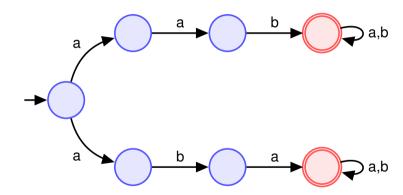
PL Game! Examples

aⁿbⁿ

umping down

rumping down

$aab\Sigma^* + aba\Sigma^*$



Pumping Lemma

Mindmap

Proofs

Proof by existent Proof by contradiction

Observation

Pigeon-hole princip Binary alphabet

Binary alphabet

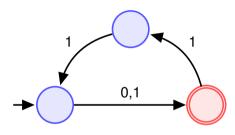
emma L Game!

a"b"

ww

olicatione

 $(\Sigma 11)^*\Sigma$



Pumping Lemma

Binary alphabet

 a^nb^n

ww

Mindmap

roots

Proof by existe Proof by contradiction

Observation

Unary alphabet Pigeon-hole pri

Binary alphabet

umping

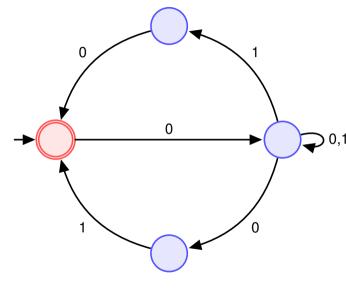
PL Game! Examples aⁿbⁿ

aⁿbⁿ ww

w umpina dov

olications

Constant Sp



Finite number of states → DFA repeats one or more states if the string is long.

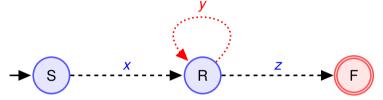
Pumping Lemma

Binary alphabet

 $a^n b^n$

16/23

Finite number of states → DFA repeats one or more states if the string is long.



Pumping Lemma

Mindmap

Proofs

Proof by existence
Proof by
contradiction

Observation

Unary alphabet
Pigeon-hole principle

Binary alphabet

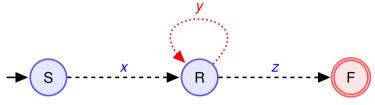
PL Game! Examples

aⁿbⁿ

vw tumping dow

mplications

Finite number of states \rightarrow DFA repeats one or more states if the string is long.



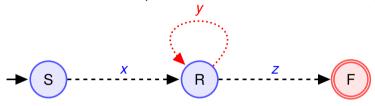
- When a DFA repeats a state R, divide the string into 3 parts:
 - The substring x before the first occurrence of R
 - The substring v between the first and last occurrence of R
 - The substring z after the last occurrence of R

Pumping I emma

Binary alphabet

 $a^n b^n$

Finite number of states → DFA repeats one or more states if the string is long.



- When a DFA repeats a state *R*, divide the string into 3 parts:
 - The substring x before the first occurrence of R
 - The substring y between the first and last occurrence of R
 - The substring z after the last occurrence of R
- **x**, z can be ε but y cannot be ε . (y forms a genuine loop.)

Pumping Lemma

Mindmap

Proofs

Proof by existence
Proof by
contradiction

Observation

Pigeon-hole principle
Binary alphabet

Pumping

PL Game!

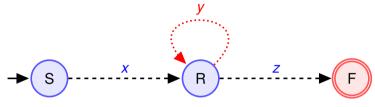
Examples a"b"

ww

nplications

nplications
Constant Space

Finite number of states → DFA repeats one or more states if the string is long.



- When a DFA repeats a state R, divide the string into 3 parts:
 - The substring x before the first occurrence of R
 - The substring v between the first and last occurrence of R
 - The substring z after the last occurrence of R
- **x**, z can be ε but y cannot be ε . (y forms a genuine loop.)
- Then, if the DFA accepts xyz then it accepts all of:

 $XZ, XVZ, XVVZ, XVVVZ, \dots$

Mindmap

Proofs

Proof by existence
Proof by
contradiction

Observation

Pigeon-hole principl
Binary alphabet

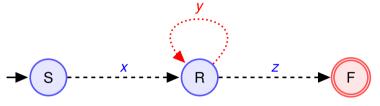
Pumping

Examples

ww ww

nlications

Finite number of states → DFA repeats one or more states if the string is long.



- When a DFA repeats a state R, divide the string into 3 parts:
 - The substring x before the first occurrence of R The substring v between the first and last occurrence of R

 - The substring z after the last occurrence of R
- **x**, z can be ε but y cannot be ε . (y forms a genuine loop.)
- Then, if the DFA accepts xyz then it accepts all of:

 $XZ, XYZ, XYYZ, XYYYZ, \dots$

For any RL L, it is possible to divide an accepted string, that is "long enough", into 3 substrings xyz, in such a way that xy^*z is a subset of L.

Binary alphabet

 $a^n b^n$

The Pumping Lemma (PL)

- We will denote a **pumping length** by ρ .
- The precise meaning of "long enough" will be: $|w| \ge p$.
- \blacksquare y has to be in the first p symbols of w.

Pumping Lemma

Mindmap

Proofs

Proof by existent
Proof by
contradiction

Observation

Unary alphabet
Pigeon-hole principl

Pumping Lemma

Example and

ww

Pumping down

Implications

The Pumping Lemma (PL)

- We will denote a **pumping length** by **p**.
- The precise meaning of "long enough" will be: $|w| \ge p$.
- \bigvee has to be in the first \wp symbols of w.

Pumping Lemma (PL)

Let L be a regular language. Then, there exists a constant p such that every string w from L, with |w| > p, can be broken into three substrings xyz such that

1 $y \neq \varepsilon$

(or equivalently: $|y| \neq 0$ or |y| > 0)

 $|xy| \leq p$

- (v) is in the first p symbols of w)
- For any $k \ge 0$, the string xy^kz is also in L
- $(xv^*z\subset L)$

Pumping I emma

Pumping Lemma

 $a^n b^n$

The Pumping Lemma (PL)

- We will denote a **pumping length** by ρ .
- The precise meaning of "long enough" will be: $|w| \ge p$.
- \blacksquare y has to be in the first p symbols of w.

Pumping Lemma (PL)

Let L be a regular language. Then, there exists a constant p such that every string w from L, with $|w| \ge p$, can be broken into three substrings xyz such that

1 $y \neq \varepsilon$

(or equivalently: $|y| \neq 0$ or |y| > 0)

 $|xy| \leq p$

- (y is in the first p symbols of w)
- For any $k \ge 0$, the string xy^kz is also in L

 $(xy^*z\subset L)$

Its main purpose in practice is to prove that a language is **not** regular.

That is, if we can show that a language L does not have this property, then we conclude that L cannot be recognized by a DFA/NFA or expressed as a regular expression.

Pumping Lemma

/lindmap

Proofs

Proof by contradiction

Observation

Unary alphabet
Pigeon-hole principle
Binary alphabet

Pumping Lemma

Examples

a"b"

mping down

The PL Game!

When the PL is used to prove that a language *L* is **not regular**, the proof can be viewed as a "game" between a **Prover** and a **Falsifier** as follows:

Pumping Lemma

Mindmap

Proofs

Proof by existence
Proof by
contradiction

Observation

Unary alphabet
Pigeon-hole principle
Binary alphabet

Lemma
PL Game!

Examples a"b"

a"b" ww

w umping do

plications

The PL Game!

When the PL is used to prove that a language *L* is **not regular**, the proof can be viewed as a "game" between a **Prover** and a **Falsifier** as follows:

1 Prover claims L is regular and fixes a pumping length p.

Proof by existence
Proof by
contradiction

Observation

Unary alphabet Pigeon-hole principle Binary alphabet

Lemma

Examples a"b"

'b" w

umping down

mplications

2 Falsifier challenges Prover and picks a string $w \in L$ of length at least p symbols.

Observation

Unary alphabet
Pigeon-hole principle
Binary alphabet

Lemma Lemma

Examples a"b"

"b" w

w umping down

mplications
Constant Space

3 Prover writes w = xyz where |xy| < p and $y \neq \varepsilon$.

Palsifier challenges Prover and picks a string $w \in L$ of length at least p symbols.

Pl Gamel

 $a^n b^n$

Pl Gamel

 $a^n b^n$

• Prover claims L is regular and fixes a pumping length p.

3 Prover writes w = xyz where $|xy| \le p$ and $y \ne \varepsilon$.

Palsifier challenges Prover and picks a string $w \in L$ of length at least p symbols.

4 Falsifier wins by finding a value for k such that xy^kz is **not** in L.

fixes a pumping length p.

|xy| < p and $y \neq \varepsilon$.

1 Prover claims L is regular and

3 Prover writes w = xyz where

Pl Gamel

 $a^n b^n$

picks a string $w \in L$ of length at least p symbols.

Palsifier challenges Prover and

4 Falsifier wins by finding a value for k such that xv^kz is **not** in L.

If **Falsifier** always wins then *L* is **not regular**.

If **Prover** always wins then **L** may be regular.

Pigeon-hole principle Binary alphabet

 $a^n b^n$

ww

2 Falsifier challenges Prover and picks $w = a^p b^p \in L$. $(|w| = 2p \ge p)$

Pumping Lemma

Pigeon-hole principle

 $a^n b^n$

ww

Prover tries to split $a \dots ab \dots b$ into xyz such that $|xy| \le p$

Since y must be within the first p symbols then y is made of a's only.

2 Falsifier challenges Prover and picks $w = a^p b^p \in L$. (|w| = 2p > p)

$$w = \underbrace{a \dots a b \dots k}_{p \text{ symbols}}$$

 $a^n b^n$

3 Prover tries to split
$$w = a \dots ab \dots b$$
 into xyz such that $|xy| \le p$

Since v must be within the first p symbols then v is made of a's only.

2 Falsifier challenges Prover and picks $w = a^p b^p \in L$. (|w| = 2p > p)

$$w = \underbrace{a \dots a_b \dots k}_{p \text{ symbols}}$$

4 Falsifier now can for example build

Hence $xy^2z \notin L$, and L is not regular.

 $xy^2z = xyyz = \underbrace{a \dots ab \dots b}_{\text{more than } p \text{ symbols}} \underbrace{b \dots b}_{\text{still } p \text{ symbols}}$

Pumping

I emma

 $a^n b^n$

a pumping length p.

lindmap

Proofs

Proof by existence
Proof by
contradiction

Observation

Unary alphabet
Pigeon-hole principle
Binary alphabet

umping

Examples a"b"

ww Dumming of

olications

nplications
onstant Space

> 2 Falsifier challenges Prover and Choose $w = 0^p 1 0^p 1 \in L$. This has length $|w| = (p+1) + (p+1) \ge p$.

 $a^n b^n$

ww

Prover tries to split w 0...010...01 into xyz such that $\underbrace{0\ldots\ldots010\ldots01}_{z}$

Since y must be within the first p symbols then y is made of 0's only.

2 Falsifier challenges Prover and Choose $w = 0^p 1.0^p 1 \in L$. This has length |w| = (p+1) + (p+1) > p.

$$w = \underbrace{0 \dots 0}_{p \text{ symbols}} 1 \underbrace{0 \dots 0}_{p \text{ symbols}} 1$$

 $a^n b^n$

ww

9 Prover tries to split
$$w = 0...01, 0...01$$
 into xyz such that $|xy| \le p$

$$0....01, 0...01$$

Since y must be within the first p symbols then y is made of 0's only.

2 Falsifier challenges Prover and Choose $w = 0^p 1 0^p 1 \in L$. This has length $|w| = (p+1) + (p+1) \ge p$.

$$w = \underbrace{0 \dots 0}_{p \text{ symbols}} 1 \underbrace{0 \dots 0}_{p \text{ symbols}} 1$$

4 Falsifier pumps y to produce $xy^2z = \underbrace{0 \dots 0}_{\text{more than } p \text{ symbols}} \underbrace{1 \underbrace{0 \dots 0}_{\text{still } p \text{ symbols}} 1}$

Hence $xy^2z \notin L$, and L is not regular.

roofo

Proof by exist

contradiction

Unary alphabet
Pigeon-hole princip

Game!

aⁿbⁿ ww

w umping down

0/23

Example $(L = \{a^i b^j \mid i > j\})$

1 Prover claims L is regular and fixes a pumping length p.

Pumping Lemma

Mindmap

Proofs

Proof by existent Proof by contradiction

bservation

Pigeon-hole principle Binary alphabet

> umping emma

Examples $a^n b^n$

a″b″ ww

ww Pumping down

plications

2 Falsifier challenges Prover and chooses $w = a^{p+1}b^p$. Here |w| = (p+1) + p > p. Pumping Lemma

Mindmap

Proof by exist

Proof by contradiction

Observation
Unary alphabet

Pigeon-hole principle Binary alphabet

> mping mma

Examples a"b"

ww Pumping down

nplications

3 Prover splits $w = a \dots aab \dots b$ into XYZ:

So v is made of a's only.

2 Falsifier challenges Prover and chooses $w = a^{p+1}b^p$. Here |w| = (p+1) + p > p.

$$w = \underbrace{a \dots a b \dots k}_{p+1 \text{ symbols}}$$

 $a^n b^n$

Pumping down

Example $(L = \{a^i b^j \mid i > i\})$

1 Prover claims *L* is regular and fixes a pumping length p.

3 Prover splits $w = a \dots aab \dots b$ into XYZ: $\underbrace{a \dots \underbrace{a \dots aab \dots b}_{x}}_{x}$

So v is made of a's only.

2 Falsifier challenges Prover and chooses $w = a^{p+1}b^p$. Here |w| = (p+1) + p > p.

4 Falsifier pumps y down and forms

 $xv^0z = xz$ $xy^0z = \underbrace{a \dots ab \dots b}_{\text{at most } p \text{ symbols}} \underbrace{b \dots b}_{\text{still } p \text{ symbols}}$

Hence $xy^0z \notin L$, and L is not regular.

Pumping

I emma

 $a^n b^n$ Pumping down

If "modern computer" = Finite Automaton then:

We can only store a fixed finite amount of data, say

 $2^{2^{43}} \approx 10^{2,647,887,844,335}$ states – a finite number still!

 $1TB = 1024^4 \times 8 = 2^{43}$ bits of information, i.e. a maximum of

 $a^n b^n$

Implications

- We can only store a fixed finite amount of data, say $1TB = 1024^4 \times 8 = 2^{43}$ bits of information, i.e. a maximum of $2^{2^{43}} \approx 10^{2,647,887,844,335}$ states – a finite number still!
- So, our "modern computer" is not even able to recognize the (entire) language aⁿbⁿ!

 $a^n b^n$

Implications

- We can only store a fixed finite amount of data, say $1TB = 1024^4 \times 8 = 2^{43}$ bits of information, i.e. a maximum of $2^{2^{43}} \approx 10^{2,647,887,844,335}$ states a finite number still!
- So, our "modern computer" is not even able to recognize the (entire) language aⁿbⁿ!
 - At some point, our "modern computer" can no longer keep track of how many a's there are in the input.

This occurs when the number of a's becomes greater than $2^{2^{43}}$.

Mindmap

Proofs

Proof by existence Proof by contradiction

Observation

Unary alphabet
Pigeon-hole principle
Binary alphabet

umping

PL Game! Examples aⁿbⁿ

,

nlications

Implications

- We can only store a fixed finite amount of data, say $1TB = 1024^4 \times 8 = 2^{43}$ bits of information, i.e. a maximum of $2^{2^{43}} \approx 10^{2,647,887,844,335}$ states a finite number still!
- So, our "modern computer" is not even able to recognize the (entire) language aⁿbⁿ!
 - At some point, our "modern computer" can no longer keep track of how many a's there are in the input.

This occurs when the number of a's becomes greater than 2^{2⁴³}.

We have assumed that the input string is not stored in the computer... (otherwise, it would just run out of memory anyway).

Mindmap

Proofs

Proof by existenc
Proof by
contradiction

Observation

Unary alphabet
Pigeon-hole principle
Binary alphabet

umping

Examples a"b"

w

Implications

If "modern computer" = Finite Automaton then:

- We can only store a fixed finite amount of data, say $1TB = 1024^4 \times 8 = 2^{43}$ bits of information, i.e. a maximum of $2^{2^{43}} \approx 10^{2,647,887,844,335}$ states a finite number still!
- So, our "modern computer" is not even able to recognize the (entire) language $a^nb^n!$
 - At some point, our "modern computer" can no longer keep track of how many a's there are in the input.

This occurs when the number of a's becomes greater than $2^{2^{43}}$.

- We have assumed that the input string is not stored in the computer... (otherwise, it would just run out of memory anyway).
- However, at 3GHz for example, this would take... a length of time so inconceivably huge that the age of the universe would be negligible by comparison. (So, do we care?)

/lindmap

Proofs

Proof by existence Proof by contradiction

Observation

Unary alphabet
Pigeon-hole principle
Binary alphabet

umping

Examples aⁿbⁿ

w Impina down

Implications

 $a^n b^n$

Constant Space

Finite Automaton: good model for algorithms which require constant **space** (i.e. space used does not grow with respect to the input size).

Some languages cannot be recognized by NFAs. Space used in computation must **grow** with respect to the input size.

■ We will see a more powerful model of computation next week!