Continuation of Non-regularity: Examples.

Example 2. Consider the language of integer squares in unary notation:

\[ L = \{ i^{12} \mid i \geq 0 \} = \{ \varepsilon, 1^4, 1^4, 1^9, 1^{16}, \ldots \} \]

Why do we guess that FAs can not recognize integer squares?

We prove the non-regularity of \( L \) by Pumping Lemma. Follow the same framework as in previous example.

Suppose that \( L \) were regular.

Let \( n \geq 1 \) be the Pumping Lemma constant.

Consider \( z = 1^{n^2} \in L \) with \( |z| = n^2 \geq n \).

We show that, for all \( u, v, w \) such that \( z = uvw \) and \( |u| \leq n \), \( |v| \geq 1 \)

there exists \( i \geq 0 \) such that \( u^i v^i w \in L \).

For this language, the "sliding" of \( u, v, w \) from left to right generates one (general) case.
\[ z = \frac{n^2}{u + v} \]

So, what is a possible \( z \geq 0 \) such that \( uvvw \notin L \)?

So, \( u = \frac{1}{|v|} \) by \( |v| + |v| \leq n \), \( |v| \geq 1 \)

\[ v = \frac{n^2 - 1}{|v| - 1} \]

As usual, an obvious trial is \( z = 0 \).

Try that yourself, and see if you encounter any problem(s).

(Note: Even if you did not succeed, with \( z = 0 \),

to generate a contradiction \( uvvw \notin L \),
it would just indicate that you might have made a wrong guess in \( z \)!
That failure does not necessarily mean the failure of Pumplin's Lemma, and you can not conclude that \( L \) should be regular!

In fact, there are languages that are known to be non-regular, and Pumplin's Lemma does fail to prove it — the conclusion of Pumplin's Lemma is always satisfied and no contradiction can be generated.)
Back to the example.

We try $i = 2$:

\[ uv^2w = \frac{1}{1+i} \frac{1}{1+i} \frac{1}{1} = 1 \]

We want to argue that $uv^2w \notin L$, but why?

What is the membership of $L$? "Integers" squares.

So, we want to argue that:

\[ n^2 < n^2 + 1 < (n+1)^2 \]

We want to demonstrate this, then $uv^2w \notin L$.

Proving (1):

Certainly $n^2 < n^2 + 1$ since $1 > 1$.

Why $n^2 + 1 < (n+1)^2$?

Consider the following equivalent inequalities:

\[ n^2 + 1 < (n+1)^2 \]

\[ \iff n^2 + 1 < n^2 + 2n + 1 \]

\[ \iff 1 < 2n + 1 \]

This is true since $1 \leq 1uv1 \leq n < 2n + 1$

Therefore, (1) is proven.
Two "benchmark" languages known non-regular
\[ \{ a^i b^i \mid i \geq 0 \} \text{ over alphabet } \{a,b\} \]
\[ \{ a^i \mid i \geq 0 \} \text{ over unary alphabet } \{a\} \]

Example 3. Consider the "queue language"
\[ L = \{ wvz \mid w \in \{0,1\}^*, z \in \mathbb{N} \} \]
\[ \varepsilon, 00, 11, 011011 \in L \quad 0, 1, 001, 01 \notin L \]

Educated guess: \( L \) is not regular — why?
( read similar motivation for example 1 )

Suppose that \( L \) were regular.
Let \( n \geq 1 \) be the Pumpliy Lemma constant.

Try considering \( z = 0^n 0^n \in L \) with \( 121 = 2n \geq n \).

Then, you'll see that you can not generate any contradiction — regardless how you consider all possible \( u, v, w \)...

But, the failure of \( z = 0^n 0^n \) does not necessarily mean the failure of Pumpliy Lemma.

Here, we consider \( z = 0^n 1 0^n \in L \)
with \( 121 = 2(n+1) \geq n \).
Consider all possible \( u, v, w \in \Sigma^* \) such that \( z = uvw \) and \( \forall (uv1 \leq n) \) and \( (v1 \geq 1) \).

We generate all possible cases by “sliding \( u, v, w \) from left to right.”

Case 1: \[ \begin{array}{c|c|c}
\text{on} & \text{on} & \text{1} \\
\hline
\text{on} & \text{1} & \text{on} \\
\end{array} \]

\[ u = 0^n 1 w1 \]
\[ v = 0^1 v1 \]
\[ w = 0^{n-1w-1v1} \text{1 on1} \]

What should be \( i \geq 0 \) such that \( uv^i w \in \mathcal{L} \)?
"\( i = 0 \)" works, "\( i = 1 \)" never works, "\( i = 2 \)" works.

Let us try \( i = 3 \):
\[ uv^3 w = 0^n 1 0^n 0 0^n 0^n 1 0^n 1 \]
\[ = 0^{n+21v1} 1 0^n 1 \]

Why \( uv^3 w = 0^{n+21v1} 1 0^n 1 \notin \mathcal{L} \)?
\[ \text{If } 0^{n+21v1} 1 0^n 1 \in \mathcal{L}, \]
so \( 0^{n+21v1} 1 0^n 1 = w w \) for some \( w \).

Hence \( w \) (the 2nd \( w \)) ends with 1.
That means \( w \) (the 2nd \( w \)) must be \( 0^n 1 \).
And 1st \( w \) must be \( 0^{n+21v1} \).

Noting that \( 1v1 \geq 1 \), we have \( uv^3 w = 0^{n+21v1} 1 0^n 1 \notin \mathcal{L} \).

Case 2, ... : Only case 1 — due to "\( |uv1| \leq n \)".
Example 4 (Incomplete)

\[ L = \{ w \# w | w \in \Sigma^0.13^* \} \]
(similar to the queue language in Example 3)

The language \( L \) is not regular? Exercise.

Example 5

\[ L = \{ x \in \Sigma^0.13^* | \#_a(x) = \#_b(x) \} \]

This language \( L \) is not the same as
\[ \{ a^i b^i | i \geq 0 \}, \]
\[ \text{in fact, } L \neq \{ a^i b^i | i \geq 0 \} \]

KWNON non-regular

The (known) non-regularity of \( \{ a^i b^i | i \geq 0 \} \)
does not necessarily imply its super-set,
like \( L \), to be non-regular.

(Think about \( \Sigma^* \supset \) any non-regular language)

Clearly regular

But this language \( L \) is non-regular.

Use Pumping Lemma
Suppose that \( L \) were regular.

Consider \( z = a^n b^n \in L \) (why?) with \( |z| = 2n \geq n \).

Consider all possible \( u, v, w, \ldots \)

\[
\begin{align*}
\text{identical to the arguments } & \text{ in Example 1 for the } \\
\text{non-regularity } & \exists \{a^n b^n \mid i \geq 0\}.
\end{align*}
\]

Note: when we argue that
for some chosen \( i \geq 0 \),
show that \( uv^i w \in L \)

make sure that we are aware that the membership \( z \in L \) is "\( \#(\cdot) = \#_b(\cdot) \)",

Not simply \( a \ldots b \ldots 

Hence, we develop a closure-property framework to show non-regularity, and apply it to \( L \) here.
In earlier classes, we used closure properties of regularity to show that a language $L$ is regular:

- Use some known regular languages $L_1, L_2, \ldots$
- Some known regularity-preserving operators (closure properties): $\cup, \cap, \setminus, \circ, \ast, r$

Try to demonstrate that, using some known regular languages with regularity-preserving operators, we can construct the target language $L$. For example:

$$( (L_3 \setminus L_4) \cap L_3 ) \circ L_1 = L$$

Challenge:
- What are the regular languages to be used eventually morphed to be $L$?
- What are the regularity-preserving operators to be used eventually morphed to be $L$?

See examples done in earlier classes.
How to use closure properties to show that a language $L$ is not regular?
- contradiction argument

Suppose that the target language $L$ were regular.

**Challenge:**
- Choose some known regular languages (accepted by FAs, denoted by $r_1, r_2, ...$)
- Choose some known regularity-preserving operators ($\cup, \cap, -, \o, \cdot, ^*$, etc.)
- Use closure properties

Try to demonstrate that, using
- $L$ (supposed to be regular)
- with some known regular languages
- via some regularity-preserving gadgets

we can construct a non-regular language

**For example**

\[
\left( \left( L_1 \cap L_4 \right) \cdot L_3 \right) - L_2 = \{ w w w \mid w \in \{a, b\}^* \}
\]

supposed regular \quad known regular \quad known regular \quad known regular

but this is non-regular

(see example 3)
We use the above framework to show that
\[ L = \{ x \in \Sigma^* \mid \#_a(x) = \#_b(x) \} \]
Suppose that \( L \) were regular.
Then, consider \( L \cap a^* b^* \)
\( \uparrow \)
\[ \text{this is a regular language (regular expression)} \]
\[ \text{regularity-preserving} \]
\[ \text{so this should be a regular language.} \]
\[ \text{BUT, } L \cap a^* b^* = \{ a^i b^i \mid i \geq 0 \} \]
\( \uparrow \)
\[ \text{should be regular} \]
\[ \text{why? check it} \]
\[ \text{this is a KNOWN non-regular language.} \]
\[ \text{contradiction!} \]
\[ \text{so } L \text{ is non-regular.} \]
Example 6. Consider the language
\[ L = \{ a^i b^j \mid i, j \geq 0 \land i \neq j \} \]

"Rookie mistake": This language \( L \) is not the complement of the language \( \{ a^i b^j \mid i \geq 0 \} \).

The language \( \{ a^i b^j \mid i \geq 0 \} \)
\[ = \{ x \mid x \in a^* b^* \land \#_a(x) = \#_b(x) \} \]

So, \[ \{ a^i b^j \mid i \geq 0 \} \]
\[ = \{ x \mid x \in a^* b^* \land \#_a(x) \neq \#_b(x) \} \]

That is,
\[ \{ a^i b^j \mid i \geq 0 \} \]
\[ = \{ x \mid x \notin a^* b^* \} \]
\[ \cup \{ x \mid x \in a^* b^* \land \#_a(x) \neq \#_b(x) \} \]
\[ = \{ x \mid x \notin a^* b^* \} \]
\[ \cup \{ a^i b^j \mid i, j \geq 0 \land i \neq j \} \]

\[ \left( \text{So, } \{ a^i b^j \mid i \geq 0 \} \neq \{ a^i b^j \mid i, j \geq 0 \land i \neq j \} \right) \]
\[ \text{or } \{ a^i b^j \mid i, j \geq 0 \land i \neq j \} \neq \{ a^i b^j \mid i, j \geq 0 \land i = j \}. \]
Back to example 6:
\[ L = \{ a^i b^j \mid i, j \geq 0 \land i \neq j \} \text{ is not regular.} \]

A direct application of Pumping Lemma on \( L \) is not “simple”.

- What should be a candidate \( z \in L \) with \( |z| \geq n \) such that the conclusion of Pumping Lemma fails?

- Read [Sip12] Problem 1.46 part b.

Here, we apply the “Closure properties”.

Suppose that \( L \) were regular.

Then \( L = \Sigma^* \cdot L \) would be regular

\[ L = \{ x \mid x \in a^* b^* \lor (x \in a^* b^* \land \#(x) = \#(y)) \} \]

How can we isolate the 2nd component out?

Then, \( L \cap a^* b^* = \{ x \mid x \in a^* b^* \land \#(x) = \#(y) \} \)

Check it

\[ = \{ a^i b^i \mid i \geq 3 \text{ known} \} \text{ non-regular, a contradiction!} \]

So \( L \) is not regular.
(Informal) justification of Pumping Lemma
(details in [Sip12] Theorem 10.70 and proof).

For every language \( L \), if \( L \) is regular, then \( \exists n \geq 1 \) \( \forall z \in L \) \( |z| \geq n \Rightarrow \exists u,v,w \) \( z = uvw \)
\( |uv| \leq n \) \( \wedge |v| \geq 1 \) \( \forall i \geq 1 \) \( u^i v^i w \in L \).

As \( L \) is regular, \( L = L(M) \) for some DFA \( M = (Q, \Sigma, \delta, q_0, F) \)
with \( |Q| = n \) (\( n \) states).

Consider an arbitrary \( z \in L = L(M) \) with \( |z| \geq n \).

(accepted by \( M \))

How to find \( u,v,w \) such that \( z = uvw \)
\( |uv| \leq n \)
\( |v| \geq 1 \)
and \( \forall i \geq 0 \) \( u^i v^i w \in L = L(M) \)
(accepted by \( M \))

— Digambara Principle!

Write \( z = a_1 a_2 \ldots a_m \) with \( m \geq n \) and \( a_i \in \Sigma \) for all \( i \).

Now, consider the effects on \( M \) (deterministic computations)
on the successive strings as inputs

\[
\begin{align*}
\text{On } & \varepsilon \text{ — ending state } F_0 \\
\text{On } & a_1 \text{ — ending state } ? \\
\text{On } & a_1 a_2 \text{ — ending state } ? \\
\text{On } & a_1 a_2 a_3 \text{ — ending state } ? \\
\vdots
\end{align*}
\]

\( m+1 \) (\( \geq n+1 \))

Trials, \( n \) states
So there must exist two distinct input/strings
\[ a_1 a_2 \ldots a_i \quad \text{and} \quad a_1 a_2 \ldots a_i a_{i+1} \ldots a_j \quad 0 \leq i < j \]
\[ a_1 a_2 \ldots a_i a_{i+1} \ldots a_j \quad a_{i+2} a_{i+3} \ldots a_m \]
that cause M to the same ending state

In fact, the Pigeonhole Principle gives that
\[ 0 \leq i < j \leq n \]
The "same ending state" must occur within no first \( n+1 \) trials

So,
\[ u = a_1 a_2 \ldots a_i \quad |uv| = j \leq n \]
\[ v = a_{i+1} a_{i+2} \ldots a_j \quad |v| = j-i \geq 1 \]
\[ w = a_{j+1} a_{j+2} \ldots a_m \]

and why \( uv^i w \in L \) for \( i = 0, 1, 2, \ldots \) ?
Chapter 2  Context-Freedom

\[ \text{a language } L \text{ is regular } \rightarrow \text{ } \L \text{ is context-free.} \]
\[ \text{in general} \]

Applications:
- Include the applications of regularity
- Specification of programming languages
  (Extended Backus-Naur Form)
- Compiler theory
- Bioinformatics (specification of "genetic operations")

We study the notion of context-freedom with context-free grammars (CFGs)
then with pushdown automata (PDAs).
What is a CFG?

A 4-tuple \((V, \Sigma, P, S)\) where

- \(V\) - finite non-empty set of variables (non-terminals)
- \(S \in V \setminus \{\phi\}\) - start variable
- \(\Sigma\) - alphabet (set of terminals)
- \(P\) - rewrite transition function
  - a finite set of productions
  - rewriting rules
    - each of which is of the form
      - \(A \rightarrow \alpha\)
      - where \(A \in V\) and \(\alpha \in (V \cup \Sigma)^*\)

Abbreviation, \(A \rightarrow \alpha_1, A \rightarrow \alpha_2\) in \(P\)
written as \(A \rightarrow \alpha_1 | \alpha_2\) in \(P\)

Grammar system: generator

Machine: recognizer acceptor

Language: recognize
Example: \( G = ( V, \Sigma, P, S ) \)

where:

\( V = \{ S \} \)

\( \Sigma = \{ a, b \} \)

\( P = \{ S \rightarrow \varepsilon, S \rightarrow a S b \} \)

What language does \( G \) generate?

Need to define "derivation" (generating/deriving) in machines.

Derivation of CFG \( G = ( V, \Sigma, P, S ) \):

1-step derivation, denoted by \( G \Rightarrow \)

(abbreviation \( G \Rightarrow \rightarrow \) when no context is clear)

\( \Rightarrow \) is a binary relation on \( (VU\Sigma)^* \)

For \( \alpha, \beta \in (VU\Sigma)^* \),

\( \alpha \Rightarrow \beta \) iff \( \alpha = \varepsilon, A \varepsilon_2 \) for some \( A \in V \)

\( \land \beta = \varepsilon, \varepsilon_1 \varepsilon_2 \) for some \( \varepsilon, \varepsilon_1, \varepsilon_2 \in (V\Sigma)^* \)

\( \land A \rightarrow \varepsilon \in P \)
multi-step derivation, denoted by \( \Rightarrow^* \) (abbreviation: \( \Rightarrow^* \))

\( \Rightarrow^* \) is the "star-closure" of \( \Rightarrow \)

For \( \alpha, \beta \in (\Gamma \cup \Sigma)^* \)

\[ \alpha \Rightarrow^* \beta \text{ iff } \exists \delta_1, \delta_2, \ldots, \delta_n \in (\Gamma \cup \Sigma)^* \text{ for some } n \geq 0 \text{ such that } \]

\[ \alpha = \delta_1 \Rightarrow \delta_2 \Rightarrow \ldots \Rightarrow \delta_n = \beta \text{ (when } n = 0, \alpha = \beta \) \]

String \( x \in \Sigma^* \) derived by \( S \):

\[ S \Rightarrow^* x \]

Language generated by \( G \), \( L(G) = \{ x \in \Sigma^* \mid S \Rightarrow^* x \} \)

Above example: CFG \( G = (\{ S \}, \{ a, b \}, \{ S \Rightarrow aSb \}, S) \)

What is \( L(G) \)?

What are the forms of strings derived/generated by \( S \)?

\[ L(G) = \{ a^i b^i \mid i \geq 0 \} \]

\( \uparrow \)

non-regular

Later

A language \( L \) is called context-free (CF) or a context-free language (CFL) if \( L = L(G) \) for some CFG \( G \).