university of groningen

Languages and Machines

L5: Finite State Machines (Part 3)

Jorge A. Pérez

Bernoulli Institute for Math, Computer Science, and AI
University of Groningen, Groningen, the Netherlands

May 2, 2023

Languages and Their Machines

Regular	Finite State Machines (FSMs)
Context-free	Pushdown Machines
Context-sensitive	Linearly-bounded Machines
Decidable	Always-terminating Turing Machines
Semi-decidable	Turing Machines

Outline

The Many Faces of Regular Languages

Consider a **regular language** L over Σ.
By definition, L has an underlying *regular set*, a subset of Σ^*.

Each of the following is an equivalent characterization of L:

- a **regular expression**, built from $\emptyset$, ϵ, $a \in \Sigma$ using union, concatenation, power, and Kleene star;
- a **regular grammar** $G = (V, \Sigma, P, S)$, where every production rule is of the form $A \to \epsilon$ or $A \to a\,B$ (with $A, B \in V$, $a \in \Sigma$).
- a **DFSM** $M = (Q, \Sigma, \delta, q_0, F)$ where $\delta : Q \times \Sigma \to Q$, $q_0 \in Q$, $F \subseteq Q$. Variants of M can be non-deterministic, with ϵ-steps.

The Many Faces of Regular Languages

Consider a **regular language** L over Σ.
By definition, L has an underlying *regular set*, a subset of Σ^*.

Each of the following is an equivalent characterization of L:

- a **regular expression**, built from $\emptyset$, ϵ, $a \in \Sigma$ using union, concatenation, power, and Kleene star;
- a **regular grammar** $G = (V, \Sigma, P, S)$, where every production rule is of the form $A \to \epsilon$ or $A \to a\,B$ (with $A, B \in V$, $a \in \Sigma$).
- a **DFSM** $M = (Q, \Sigma, \delta, q_0, F)$ where $\delta : Q \times \Sigma \to Q$, $q_0 \in Q$, $F \subseteq Q$. Variants of M can be non-deterministic, with ϵ-steps.

Natural questions:

- Why do we identify multiple equivalent characterizations?

The Many Faces of Regular Languages

Consider a **regular language** L over Σ.
By definition, L has an underlying *regular set*, a subset of Σ^*.

Each of the following is an equivalent characterization of L:

- a **regular expression**, built from $\emptyset$, ϵ, $a \in \Sigma$ using union, concatenation, power, and Kleene star;
- a **regular grammar** $G = (V, \Sigma, P, S)$, where every production rule is of the form $A \to \epsilon$ or $A \to a\, B$ (with $A, B \in V$, $a \in \Sigma$).
- a **DFSM** $M = (Q, \Sigma, \delta, q_0, F)$ where $\delta : Q \times \Sigma \to Q$, $q_0 \in Q$, $F \subseteq Q$. Variants of M can be non-deterministic, with ϵ-steps.

Natural questions:

- Why do we identify multiple equivalent characterizations?
- Is every language a regular language? How to (dis)prove it?

There are Non Regular Sets

- Example 1:

$$\begin{aligned} L_1 &= \{a^n b^n \mid n \geq 1\} \\ &= \{ab, aabb, aaabbb \ldots\} \end{aligned}$$

There are Non Regular Sets

- Example 1:

$$\begin{aligned} L_1 &= \{a^n b^n \mid n \geq 1\} \\ &= \{ab, aabb, aaabbb \ldots\} \end{aligned}$$

- Not accepted by a *finite* automaton:
  We cannot distinguish between **infinitely many** cases with **finitely many** states (unbounded vs bounded information)

There are Non Regular Sets

- Example 1:

$$\begin{aligned} L_1 &= \{a^n b^n \mid n \geq 1\} \\ &= \{ab, aabb, aaabbb \ldots\} \end{aligned}$$

- Example 2:

$$\begin{aligned} L_2 &= \{a^{2^n} \mid n \geq 0\} \\ &= \{a, a^2, a^4, a^8, \ldots\} \\ &= \{x \in \{a\}^* \mid |x| \text{ is a power of 2}\} \end{aligned}$$

- Not accepted by a *finite* automaton:
  We cannot distinguish between **infinitely many** cases with **finitely many** states (unbounded vs bounded information)

There are Non Regular Sets

- Example 1:

$$L_1 = \{a^n b^n \mid n \geq 1\} = \{ab, aabb, aaabbb \ldots\}$$

- Example 2:

$$\begin{aligned} L_2 &= \{a^{2^n} \mid n \geq 0\} \\ &= \{a, a^2, a^4, a^8, \ldots\} \\ &= \{x \in \{a\}^* \mid |x| \text{ is a power of } 2\} \end{aligned}$$

- Not accepted by a *finite* automaton:
  We cannot distinguish between **infinitely many** cases with **finitely many** states (unbounded vs bounded information)
- **Today**: A tool for proving that sets are not regular.

Example 1: $L_1 = \{a^n b^n \mid n \geq 1\}$ (1/2)

A proof by contradiction that L_1 is not regular:

- If L_1 were regular, it would be accepted by a DFSM M
- Machine M has a finite number of states, say k

Example 1: $L_1 = \{a^n b^n \mid n \geq 1\}$ (1/2)

A proof by contradiction that L_1 is not regular:

- If L_1 were regular, it would be accepted by a DFSM M
- Machine M has a finite number of states, say k
- Consider the action of M on input $a^n b^n$, with $n \gg k$:

$$[q_0]\ \underbrace{aaaaaaaaaaaaaaa}_{n}\ \underbrace{bbbbbbbbbbbbbbb}_{n}\ [q_f]$$

Example 1: $L_1 = \{a^n b^n \mid n \geq 1\}$ (1/2)

A proof by contradiction that L_1 is not regular:

- If L_1 were regular, it would be accepted by a DFSM M
- Machine M has a finite number of states, say k
- Consider the action of M on input $a^n b^n$, with $n \gg k$:

$$[q_0]\ \underbrace{aaaaaaaaaaaaaaa}_{n}\ \underbrace{bbbbbbbbbbbbbbb}_{n}\ [q_f]$$

- By the pigeonhole principle, there is a state q_i that is visited more than once in processing the sequence of as:

$$[q_0]\ \overbrace{aaaaaaaa[q_i]aaaaa[q_i]aa}^{n}\ \overbrace{bbbbbbbbbbbbbbb}^{n}\ [q_f]$$

Example 1: $L_1 = \{a^n b^n \mid n \geq 1\}$ (2/2)

- Let's split $a^n b^n$ into three pieces (u, v, w) according to q_i:

$$[q_0]\ \underbrace{aaaaaaaa}_{u}\ [q_i]\ \underbrace{aaaaa}_{v}\ [q_i]\ \underbrace{aa\ bbbbbbbbbbbbbbb}_{w}\ [q_f]$$

We have: $\underbrace{\hat{\delta}(q_0, u) = q_i}$, $\underbrace{\hat{\delta}(q_i, v) = q_i}$, and $\underbrace{\hat{\delta}(q_i, w) = q_f \in F}$.

Example 1: $L_1 = \{a^n b^n \mid n \geq 1\}$ (2/2)

- Let's split $a^n b^n$ into three pieces (u, v, w) according to q_i:

$$[q_0]\ \underbrace{aaaaaaaa}_{u}\ [q_i]\ \underbrace{aaaaa}_{v}\ [q_i]\ \underbrace{aa\ bbbbbbbbbbbbbbb}_{w}\ [q_f]$$

We have: $\underbrace{\hat{\delta}(q_0, u) = q_i,}_{\longrightarrow}$ $\underbrace{\hat{\delta}(q_i, v) = q_i}_{\circlearrowleft}$, and $\underbrace{\hat{\delta}(q_i, w) = q_f \in F}_{\longrightarrow}$.

Example 1: $L_1 = \{a^n b^n \mid n \geq 1\}$ (2/2)

- Let's split $a^n b^n$ into three pieces (u, v, w) according to q_i:

$$[q_0]\ \underbrace{aaaaaaaa}_{u}\ [q_i]\ \underbrace{aaaaa}_{v}\ [q_i]\ \underbrace{aa\ bbbbbbbbbbbbbbb}_{w}\ [q_f]$$

We have: $\underbrace{\hat{\delta}(q_0, u) = q_i}_{\longrightarrow}$, $\underbrace{\hat{\delta}(q_i, v) = q_i}_{\circlearrowleft}$, and $\underbrace{\hat{\delta}(q_i, w) = q_f \in F}_{\longrightarrow}$.

What does $\circlearrowleft$ mean?

Example 1: $L_1 = \{a^n b^n \mid n \geq 1\}$ (2/2)

- Let's split $a^n b^n$ into three pieces (u, v, w) according to q_i:

$$[q_0]\, \underbrace{aaaaaaaa}_{u}\, [q_i]\, \underbrace{aaaaa}_{v}\, [q_i]\, \underbrace{aa\ bbbbbbbbbbbbbbb}_{w}\, [q_f]$$

We have: $\underbrace{\hat{\delta}(q_0, u) = q_i}_{\longrightarrow}$, $\underbrace{\hat{\delta}(q_i, v) = q_i}_{\circlearrowleft}$, and $\underbrace{\hat{\delta}(q_i, w) = q_f \in F}_{\longrightarrow}$.

What does $\circlearrowleft$ mean?

- We could erase v and the obtained string would be accepted!

$$[q_0]\, \underbrace{aaaaaaaa}_{u}\, [q_i]\, \underbrace{aa\ bbbbbbbbbbbbbbb}_{w}\, [q_f]$$

This is wrong: $uw = a^{n-j} b^n \in L(M)$ but $uw \notin L_1$

Example 1: $L_1 = \{a^n b^n \mid n \geq 1\}$ (2/2)

- Let's split $a^n b^n$ into three pieces (u, v, w) according to q_i:

$$[q_0]\,\underbrace{aaaaaaaa}_{u}\,[q_i]\,\underbrace{aaaaa}_{v}\,[q_i]\,\underbrace{aa\;bbbbbbbbbbbbbbb}_{w}\,[q_f]$$

We have: $\underbrace{\hat{\delta}(q_0, u) = q_i}_{\longrightarrow}$, $\underbrace{\hat{\delta}(q_i, v) = q_i}_{\circlearrowleft}$, and $\underbrace{\hat{\delta}(q_i, w) = q_f \in F}_{\longrightarrow}$.

What does $\circlearrowleft$ mean?

- We could erase v and the obtained string would be accepted!

$$[q_0]\,\underbrace{aaaaaaaa}_{u}\,[q_i]\,\underbrace{aa\;bbbbbbbbbbbbbbb}_{w}\,[q_f]$$

This is wrong: $uw = a^{n-j} b^n \in L(M)$ but $uw \notin L_1$

- We could even insert extra copies of v (say, uv^3w), and the resulting string would be wrongly accepted too!

Example 2: $L_2 = \{a^{2^n} \mid n \geq 0\}$ (1/2)

- A DFSM M with k states, with $L(M) = L_2$ and start state q_0
- Let $n \gg k$ and consider the action of M on scanning string a^{2^n}
- By the pigeonhole principle, M must repeat a state q while scanning the first n symbols of a^{2^n}
- Now let i, j, m be such that $2^n = i + j + m$, with $0 < j \leq n$ and

$$\hat{\delta}(q_0, a^i) = q \qquad \hat{\delta}(q, a^j) = q \qquad \hat{\delta}(q, a^m) = q_f \in F$$

Example 2: $L_2 = \{a^{2^n} \mid n \geq 0\}$ (1/2)

- A DFSM M with k states, with $L(M) = L_2$ and start state q_0
- Let $n \gg k$ and consider the action of M on scanning string a^{2^n}
- By the pigeonhole principle, M must repeat a state q while scanning the first n symbols of a^{2^n}
- Now let i, j, m be such that $2^n = i + j + m$, with $0 < j \leq n$ and

$$\hat{\delta}(q_0, a^i) = q \qquad \hat{\delta}(q, a^j) = q \qquad \hat{\delta}(q, a^m) = q_f \in F$$

Alternatively:

$$[q_0]\,\overbrace{\underbrace{aaaaaaaaaa}_{i}\,\underbrace{aaaaa}_{j}\,\underbrace{aaaaaaaaaaaaaaaa}_{m}}^{2^n}\,[q_f]$$

Example 2: $L_2 = \{a^{2^n} \mid n \geq 0\}$ (1/2)

- A DFSM M with k states, with $L(M) = L_2$ and start state q_0
- Let $n \gg k$ and consider the action of M on scanning string a^{2^n}
- By the pigeonhole principle, M must repeat a state q while scanning the first n symbols of a^{2^n}
- Now let i, j, m be such that $2^n = i + j + m$, with $0 < j \leq n$ and

$$\hat{\delta}(q_0, a^i) = q \qquad \hat{\delta}(q, a^j) = q \qquad \hat{\delta}(q, a^m) = q_f \in F$$

Alternatively:

$$[q_0]\overbrace{\underbrace{aaaaaaaaaa}_{i}\,[q]\,\underbrace{aaaaa}_{j}\,[q]\,\underbrace{aaaaaaaaaaaaaaaa}_{m}}^{2^n}[q_f]$$

Example 2: $L_2 = \{a^{2^n} \mid n \geq 0\}$ (2/2)

- Now, given $\hat{\delta}(p, a^j) = p$, we could insert an extra a^j, to get a^{2^n+j}, and the resulting string would be erroneously accepted:

$$[q_0]\,\underbrace{aaaaaaaaaa}_{i}\,[q]\,\underbrace{aaaaa}_{j}\,[q]\,\underbrace{aaaaa}_{j}\,[q]\,\underbrace{aaaaaaaaaaaaaaaa}_{m}\,[q_f]$$

Indeed, we can derive $\hat{\delta}(q_0, a^{2^n+j}) = q_f \in F$.

Example 2: $L_2 = \{a^{2^n} \mid n \geq 0\}$ (2/2)

- Now, given $\hat{\delta}(p, a^j) = p$, we could insert an extra a^j, to get a^{2^n+j}, and the resulting string would be erroneously accepted:

$$[q_0]\,\underbrace{aaaaaaaaaa}_{i}\,[q]\,\underbrace{aaaaa}_{j}\,[q]\,\underbrace{aaaaa}_{j}\,[q]\,\underbrace{aaaaaaaaaaaaaaaa}_{m}\,[q_f]$$

  Indeed, we can derive $\hat{\delta}(q_0, a^{2^n+j}) = q_f \in F$.

- But this is wrong, because $2^n + j$ is not a power of 2:

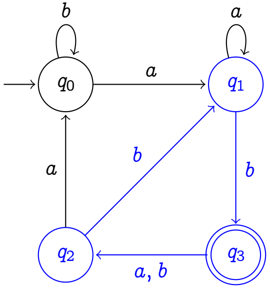
$$\begin{aligned} 2^n + j &\leq 2^n + n \\ 2^n + j &< 2^n + 2^n \\ &= 2^{n+1} \end{aligned}$$

  2^{n+1} is the next power of 2 greater than 2^n

Pumping Strings

Pump a string v: Construct new strings by repeating substrings in v.

Let $z = ababbaaab$ be a string accepted by the following machine:



- z can be split as $z = \underbrace{a}_{u}\underbrace{bab}_{v}\underbrace{baaab}_{w}$
- Strings $a(bab)^i baaab$ are obtained by pumping $v = bab$ in z
  The pumped strings uv^iw are accepted (for all $i \geq 0$)

Outline

The Pumping Lemma: Essence

$$
\begin{aligned}
&\forall L : (L \text{ is regular} \Rightarrow \\
&\quad (\exists k \geq 1 : \forall z : z \in L \wedge |z| \geq k \Rightarrow \\
&\quad\quad (\exists u, v, w : z = uvw \wedge |uv| \leq k \wedge v \neq \epsilon \\
&\quad\quad\quad \wedge (\forall i \geq 0 : uv^i w \in L)))).
\end{aligned}
$$

Intuitively:

- k is the # of states of a DFSM accepting L (depends on L)
- $|z| \geq k$ ensures that at least one state is visited twice
- Reading u gets you to the repeated state, v gets you there again
- Since v traverses a loop, using $uv^i w$ we can omit it (if $i = 0$) or repeat it (if $i > 1$)

The Pumping Lemma: Essence

$$\forall L : (L \text{ is regular} \Rightarrow$$
$$(\exists k \geq 1 : \forall z : z \in L \wedge |z| \geq k \Rightarrow$$
$$(\exists u, v, w : z = uvw \wedge |uv| \leq k \wedge v \neq \epsilon$$
$$\wedge (\forall i \geq 0 : uv^i w \in L)))).$$

Note the three conditions on u, v, w:

C1 $uv^i w \in L$

C2 $v \neq \epsilon$ (although u and w can be ϵ)

C3 $|uv| \leq k$

$L_1 = \{a^n b^n \mid n \geq 1\}$ is Not Regular (V1)

Claim. $L_1 = \{a^n b^n \mid n \geq 1\}$ is not regular.

Key steps in a proof **by contradiction**:

- Suppose, for the sake of contradiction, that L_1 is regular
- Then, the pumping lemma must apply to all suitably long $z \in L_1$
- Consider, in particular, $z = a^k b^k \in L_1$ (cf. $\exists k \geq 1 \ldots$)
- Let uvw be a decomposition of z (cf. $\exists u, v, w : z = uvw \ldots$). We consider different possibilities for $v \neq \epsilon$

$L_1 = \{a^n b^n \mid n \geq 1\}$ is Not Regular (V1)

Claim. $L_1 = \{a^n b^n \mid n \geq 1\}$ is not regular.

Key steps in a proof **by contradiction**:

- Suppose, for the sake of contradiction, that L_1 is regular
- Then, the pumping lemma must apply to all suitably long $z \in L_1$
- Consider, in particular, $z = a^k b^k \in L_1$ (cf. $\exists k \geq 1 \ldots$)
- Let uvw be a decomposition of z (cf. $\exists u, v, w : z = uvw \ldots$). We consider different possibilities for $v \neq \epsilon$ (C2):
  a) Suppose v contains only a's.
     $uvvw$ will have more a's than b's, violating (C1): contradiction
  b) Suppose v contains only b's.
     $uvvw$ will have more b's than a's, violating (C1): contradiction
  c) Suppose v contains both a's and b's.
     $uvvw$ may have the same number of a's and b's but in an incorrect sequence, as in $aaa\ ababab\ bbbbb$. This violates (C1).
- A contradiction is unavoidable in all cases: L_1 is not regular.

$L_1 = \{a^n b^n \mid n \geq 1\}$ is Not Regular (V2)

Claim. $L_1 = \{a^n b^n \mid n \geq 1\}$ is not regular.

Another proof by contradiction, now selecting z more carefully:

- Suppose, for the sake of contradiction, that L_1 is regular
- Then, the pumping lemma must apply to all suitably long $z \in L_1$
- Consider, in particular, $z = a^k b^k \in L_1$ (cf. $\exists k \geq 1 \ldots$).
  Let uvw be a decomposition of z
- Since $|uv| \leq k$ (C3) any split uv will contain only a's.
  Because $v \neq \epsilon$ (C2), v contains at least one a.
  Moreover, we should have $uw \in L_1$.
- However, the string uw contains less a's than z (i.e., those in v), while keeping the same number of b's
- Since $n_a(uw) < n_b(uw)$, we have that uw it cannot be in L_1.
  This contradicts the pumping lemma, and we conclude that L_1 is not regular.

Another Example

Claim. $L_3 = \{x \in \{a, b\}^* \mid n_a(x) = n_b(x)\}$ is not regular.

Key steps in a proof by contradiction:

- Suppose, for the sake of contradiction, that L_3 is regular
- The pumping lemma applies to all $z \in L_3$, in particular to $z = a^k b^k \in L_3$ (cf. $\exists k \geq 1 \ldots$)
- Note: by defining $u = w = \epsilon$ and $v = a^k b^k$, then the string z seems like a good candidate: $u\, v^i\, w \in L_3$.
- Here $|uv| \leq k$ (C3) is crucial. Any split uv will contain only a's. Thus, $u\, vv\, w \notin L_3$ (C1). This contradicts the pumping lemma, and we conclude that L_3 is not regular.

Another Example

Claim. $L_3 = \{x \in \{a, b\}^* \mid n_a(x) = n_b(x)\}$ is not regular.

Key steps in a proof by contradiction:

- Suppose, for the sake of contradiction, that L_3 is regular
- The pumping lemma applies to all $z \in L_3$, in particular to $z = a^k b^k \in L_3$ (cf. $\exists k \geq 1 \ldots$)
- Note: by defining $u = w = \epsilon$ and $v = a^k b^k$, then the string z seems like a good candidate: $u\, v^i\, w \in L_3$.
- Here $|uv| \leq k$ (C3) is crucial. Any split uv will contain only a's. Thus, $u\, vv\, w \notin L_3$ (C1). This contradicts the pumping lemma, and we conclude that L_3 is not regular.

The selection of z matters. Some strings are not good candidates:

- Take $z' = (ab)^k$. This string can be pumped by defining

$$u = \epsilon \qquad v = ab \qquad w = (ab)^{k-1}$$

You can check that $u\, v^i\, w \in L_3$, for any $i \geq 0$.

The Pumping Lemma (General Form)

Lemma (Pumping)

Let A be a regular set. Then the following property holds of A:

(P) *There exists $k \geq 0$ such that for any strings x, y, z with $xyz \in A$ and $|y| \geq k$, there exist strings u, v, w such that $y = uvw$, $v \neq \epsilon$, and for all $i \geq 0$, the string $xuv^iwz \in A$.*

Notice:

- Slightly more general than in the reader
  The string to be split (i.e., y) is "surrounded" by x, z
- *Intuition:* For any string in a regular set A we can find a non-null substring $v \neq \epsilon$ that can be freely "pumped" and the resulting string is still in A

Outline

The Pumping Lemma

The pumping lemma is used to prove that sets are non-regular.

Two strategies for devising a proof:

- Proofs **by contradiction** (as we have seen)
- Use the contrapositive form, and **play a game** (presented next) (Contrapositive means that to prove $Q \to P$ we prove $\neg P \to \neg Q$.)

Also: we could exploit the **closure properties** of regular languages to avoid using the pumping lemma.

The Pumping Lemma

Lemma (Pumping, **Standard Form** ($Q \to P$))

*Let A be a regular set (Q). **Then** the following property holds of A:*

(P) ***There exists*** *$k \geq 0$ such that **for any** strings x, y, z with $xyz \in A$ and $|y| \geq k$, **there exist** strings u, v, w such that $y = uvw$, $v \neq \epsilon$, and **for all** $i \geq 0$, the string $xuv^iwz \in A$.*

Notice the alternating sequence: $\exists$ - $\forall$ - $\exists$ - $\forall$.

The Pumping Lemma

Lemma (Pumping, **Standard Form** ($Q \to P$))

*Let A be a regular set (Q). **Then** the following property holds of A:*

(P) ***There exists** $k \geq 0$ such that **for any** strings x, y, z with $xyz \in A$ and $|y| \geq k$, **there exist** strings u, v, w such that $y = uvw$, $v \neq \epsilon$, and **for all** $i \geq 0$, the string $xuv^iwz \in A$.*

Notice the alternating sequence: $\exists$ - $\forall$ - $\exists$ - $\forall$.

Lemma (Pumping, **Contrapositive Form** ($\neg P \to \neg Q$))

Let A be a set of strings, which enjoys the following property:

($\neg P$) ***For all** $k \geq 0$ **there exist** strings x, y, z such that $xyz \in A$ and $|y| \geq k$, and **for all** u, v, w with $y = uvw$ and $v \neq \epsilon$, **there exists** an $i \geq 0$ such that the string $xuv^iwz \notin A$.*

Then, A is not regular ($\neg Q$).

Notice the (inverted) sequence: $\forall$ - $\exists$ - $\forall$ - $\exists$.

Games with The Pumping Lemma

Lemma (Pumping, **Contrapositive Form**)

Let A be a set of strings, which enjoys the following property:

($\neg P$) *For all $k \geq 0$ there exist strings x, y, z such that $xyz \in A$ and $|y| \geq k$, and for all u, v, w with $y = uvw$ and $v \neq \epsilon$, there exists an $i \geq 0$ such that the string $xuv^iwz \notin A$.*

Then, A is not regular ($\neg Q$).

A game against the demon

The demon wants to show A is regular; you want to show it is not:

1. The demon picks k (what is his best strategy for choosing?)
2. You pick x, y, z such that $xyz \in A$ and $|y| \geq k$
3. The demon picks u, v, w such that $y = uvw$ and $v \neq \epsilon$
4. You pick $i \geq 0$

If $xuv^iwz \notin A$, you win. Otherwise, the demon wins if $xuv^iwz \in A$.

Playing with the demon: Example

We show that the language $L_4 = \{a^n b^m \mid n \geq m\}$ is not regular:

1. *The demon picks* k
2. *We pick* x, y, z *such that* $xyz \in L_4$ *and* $|y| \geq k$
   Let's pick $x = a^k$, $y = b^k$, and $z = \epsilon$. (We'll split only b's.)
3. *The demon picks* u, v, w *such that* $y = uvw$ *and* $v \neq \epsilon$
   Let $k = j + m + n$, with $m > 0$.
   The demon picks $u = b^j$, $v = b^m$, and $w = b^n$.
4. *We pick* $i \geq 0$
   No matter the demon's choice, we can take $i = 2$ to win:

$$\begin{aligned} xuv^2wz &= a^k \ b^j b^m b^m b^n \\ &= a^k \ b^{k+m} \end{aligned}$$

Clearly, $a^k \ b^{k+m} \notin L_4$.

PL is Necessary but not Sufficient

- The lemma gives a **necessary condition** for regularity, i.e. a condition that necessarily holds for all regular languages.
- But it does not give a **sufficient condition**, i.e., a condition that suffices to guarantee that the language is regular.

PL is Necessary but not Sufficient

- The lemma gives a **necessary condition** for regularity, i.e. a condition that necessarily holds for all regular languages.
- But it does not give a **sufficient condition**, i.e., a condition that suffices to guarantee that the language is regular.
- In other words:

  language is regular $\Rightarrow$ C (i.e., C is a necessary condition)

  $C \Rightarrow$ language is regular (i.e., C is a sufficient condition)

PL is Necessary but not Sufficient

- The lemma gives a **necessary condition** for regularity, i.e. a condition that necessarily holds for all regular languages.
- But it does not give a **sufficient condition**, i.e., a condition that suffices to guarantee that the language is regular.
- In other words:

  language is regular $\Rightarrow$ C (i.e., C is a necessary condition)

  C $\Rightarrow$ language is regular (i.e., C is a sufficient condition)

- In the game formulation, each non regular set determines a different game
- There exist non regular sets that satisfy the hypotheses of the pumping lemma
- To show that a set is regular, we must construct a corresponding FSM or a regular expression

PL is Necessary but not Sufficient

- The lemma gives a **necessary condition** for regularity, i.e. a condition that necessarily holds for all regular languages.
- But it does not give a **sufficient condition**, i.e., a condition that suffices to guarantee that the language is regular.
- In other words:

  language is regular $\Rightarrow C$ (i.e., C is a necessary condition)

  $C \Rightarrow$ language is regular (i.e., C is a sufficient condition)

- In the game formulation, each non regular set determines a different game
- There exist non regular sets that satisfy the hypotheses of the pumping lemma
- To show that a set is regular, we must construct a corresponding FSM or a regular expression
- Another characterization of regular languages, not covered here: the Myhill-Nerode theorem (related to Section 3.8)

Outline

Closure Properties

Closure properties state that when one (or several) languages are in a given class C, then certain related languages are also in C.

Useful to define new languages on C (say, the regular languages) building upon existing/known ones.

Closure Properties

Closure properties state that when one (or several) languages are in a given class C, then certain related languages are also in C.

Useful to define new languages on C (say, the regular languages) building upon existing/known ones.

Regular languages are closed under the following operations:

- Union, concatenation, Kleene star
- Complement (recall: the complement $\overline{L}$ of L is $\Sigma^* \setminus L$).
- Intersection
- Reversal (i.e., if $a_1 a_2 \ldots a_{n-1} a_n \in L$ then $a_n a_{n-1} \ldots a_2 a_1 \in L$).

Closure Properties of Reg. Languages

- *Union, concatenation, Kleene star*:
  Immediate by considering regular expressions

Closure Properties of Reg. Languages

- *Union, concatenation, Kleene star*:
  Immediate by considering regular expressions
- *Complement*:
  Obtain a DFSA M of the given language, and define another DFSA M' in which the accepting states of M become non-accepting states of M', and vice versa

Closure Properties of Reg. Languages

- *Intersection*: Two possibilities
  1) Use the law $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_1}}$ (and reduce to cases above)
  2) Define a construction that runs two DFSAs in "parallel"
  Given $M_i = (Q_i, \Sigma, \delta_i, q_i, F_i)$ (with $i \in \{1, 2\}$), define
  $M = (Q_1 \times Q_2, \Sigma, \delta, (q_1, q_2), F_1 \times F_2)$, where
  $\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$.

Closure Properties of Reg. Languages

- *Intersection*: Two possibilities
  1) Use the law $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_1}}$ (and reduce to cases above)
  2) Define a construction that runs two DFSAs in “parallel”
  Given $M_i = (Q_i, \Sigma, \delta_i, q_i, F_i)$ (with $i \in \{1, 2\}$), define
  $M = (Q_1 \times Q_2, \Sigma, \delta, (q_1, q_2), F_1 \times F_2)$, where
  $\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$.
- Reversal: Given a machine M and create a machine M^R by:
  - reversing all arcs in the state diagram;
  - making the start state the only accepting state;
  - creating a new starting state, with ϵ-transitions to the accepting states

Application of Closure Properties

Claim. If L is a non-regular language, then $\overline{L}$ is non-regular too.

Idea for a proof by contradiction:

- Suppose, for the sake of contradiction, that $\overline{L}$ is regular.
- By the closure properties, $\overline{\overline{L}}$ should be regular as well.
- But $\overline{\overline{L}} = L$, which contradicts our assumption.
- We conclude that $\overline{L}$ must be non-regular.

Using Closure Properties Instead of PL

Claim. Language $L_3 = \{x \in \{a, b\}^* \mid n_a(x) = n_b(x)\}$ is non-regular.

- Suppose, for the sake of contradiction, that L_3 is regular.
- We know a^*b^* is a regular set
- Then the set $L_3 \cap a^*b^*$ would be regular, because of the closure property under intersection
- But $L_3 \cap a^*b^* = \{a^n b^n \mid n \geq 0\}$ is not regular: contradiction.

Taking Stock

Properties of regular languages:

- Pumping lemma:
  A technique to prove that languages are not regular
- Closure properties:
  Conditions to obtain new regular languages from the operation of existing ones

Next Lecture

- FSM minimization
  Not yet in the reader, but you can check the slides, and/or Chapter 14 in Kozen's book ('Automata and Computability').