

The Pumping Lemma

The Pumping Lemma is a powerful technique for proving that certain languages are *not* regular.

Claim: $B = \{0^n 1^n : n \geq 0\}$ is not regular

Informal argument.

If there were a DFA that recognizes B , it would need to remember the number of 0's read from the input string. This would require a way to store an arbitrarily large number, but any DFA has only a finite amount of memory (given by the fixed number of states). \square

But need to be careful: both

$$L_1 = \{w : w \text{ has an equal number of 0s and 1s}\}$$

$$L_2 = \{w : w \text{ has an equal no. of occurrences of 01 and 10 as substrings}\}$$

seem to require infinite memory to recognize. Now L_1 is not regular but

Exercise (*Moderately hard*). Prove that L_2 is regular.

The Pumping Lemma, in poetic form

“Any regular language L has a magic number p
And any long-enough word in L has the following property:
Amongst its first p symbols is a segment you can find
Whose repetition or omission leaves x amongst its kind.”

“So if you find a language L which fails this acid test,
And some long word you pump becomes distinct from all the rest,
By contradiction you have shown that language L is not
A regular guy, resilient to the damage you have wrought.”

“But if, upon the other hand, x stays within its L ,
Then either L is regular, or else you chose not well.
For w is xyz , and y cannot be null,
And y must come before p symbols have been read in full.”

“As mathematical postscript, an addendum to the wise:
The basic proof we outlined here does certainly generalize.
So there is a pumping lemma for all languages context-free,
Although we do not have the same for those that are r.e.”

By Martin Cohn and Harry Mairson

The Pumping Lemma

Pumping Lemma. If A is a regular language, then there is a number p – the *pumping length* – such that if $s \in A$ of length at least p , then s may be divided into three pieces, $s = x y z$, satisfying:

- (i) for each $i \geq 0$, $x y^i z \in A$
‘Words “pumped up” from s belong to A .’
- (ii) $|y| > 0$
- (iii) $|x y| \leq p$.

Note: without (ii), the Lemma is vacuous (because $\epsilon^i = \epsilon$ for all $i \geq 0$).

The Pumping Lemma is a complex statement: it is equivalent to

$$\forall L \in \mathbf{Reg}. \exists p \geq 1. \forall s \in L. \exists x, y, z \in \Sigma^*. \forall i \geq 0. \exists$$

where \exists is $|s| \geq p \rightarrow s = x y z \wedge |x y| \leq p \wedge |y| > 0 \wedge x y^i z \in L$.

Proof of the Pumping Lemma

Let $M = (Q, \Sigma, \delta, q_{\text{init}}, F)$ be a DFA that accepts A , and let $p = |Q|$.

Suppose $s = a_1 \cdots a_n \in L(M)$ where $n \geq p$. We have

$$\underbrace{q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_p} q_p \cdots \cdots \xrightarrow{a_n} q_n}_{p+1 \text{ states}} \in F$$

where $q_0 = q_{\text{init}}$. By the Pigeonhole Principle, q_0, \cdots, q_p cannot all be distinct.

So $q_j = q_{j'}$ for some $0 \leq j < j' \leq p$. Thus the above transition sequence is

$$q_0 \xrightarrow{x}^* q_j \xrightarrow{y}^* q_{j'} (= q_j) \xrightarrow{z}^* q_n \in F$$

where $x = a_1 \cdots a_j$, $y = a_{j+1} \cdots a_{j'}$ and $z = a_{j'} \cdots a_n$. We have

$|x y| \leq p$ and $|y| > 0$, and for every $i \geq 0$, $x y^i z \in L(M)$ as

$$q_0 \xrightarrow{x}^* \underbrace{q_j \xrightarrow{y}^* q_j \cdots \xrightarrow{y}^* q_j}_i \xrightarrow{z}^* q_n \in F$$

□

Example: $B = \{ 0^i 1^i : i \geq 0 \}$ is not regular.

Proof. Suppose, for a contradiction, B is regular. Let the pumping length be p .

Take $s = 0^p 1^p \in B$. Since $|s| > p$, by the Lemma, there are x, y, z such that $s = x y z$ where $|x y| \leq p$ and $|y| > 0$. Hence $x = 0^a, y = 0^b$ where $b > 0$, and $a + b \leq p$.

The Lemma further asserts: for each $i \geq 0, 0^a 0^{bi} 0^{p-a-b} 1^p \in B$. In particular (taking $i = 0$) $0^a 0^{p-a-b} 1^p = 0^{p-b} 1^p \in B$, a contradiction. \square

Exercise. Convince yourself that the same argument above can be used to show that $\{ w : w \text{ has equal no. of 0s and 1s} \}$ is not regular.

The Pumping Lemma is not always easy to apply: the trick is to identify an appropriate word to “pump”. It is often useful to “pump down” i.e. take $i = 0$.

A powerful characterization of regular languages

Let $x, y \in \Sigma^*$ be strings and let $L \subseteq \Sigma$.

We say that x and y are L -*indistinguishable*, written $x \equiv_L y$, if for every $z \in \Sigma^*, xz \in L$ iff $yz \in L$.

Fact. \equiv_L is an equivalence relation.

We define the *index* of L to be the number of equivalence classes of L .

The index of L may be finite or infinite.

Examples

Take $\Sigma = \{0, 1\}$.

(i) $L_3 = \{w : w \text{ has even length}\}$.

$u \equiv_{L_3} v$ iff $|u| \equiv |v| \pmod{2}$.

Now \equiv_{L_3} has two equivalence classes:

$[\epsilon] = [00] = [10] = \dots = \{w : |w| \text{ even}\}$ and

$[1] = [010] = [110] = \dots = \{w : |w| \text{ odd}\}$.

(ii) $L_4 = \{w : w \text{ has equal numbers of 0s and 1s}\}$.

For any $i, j \geq 0$, if $i \neq j$ then $0^i \not\equiv_{L_4} 0^j$ (because $0^i 1^i \in L_4$ but $0^j 1^i \notin L_4$).

Therefore the index of L_4 is infinite.

A powerful characterization of regular languages (con't)

Myhill-Nerode Theorem: A language L is regular iff \equiv_L has finite index. Moreover the index is the size (= number of states) of the *smallest* DFA accepting L .

Note: The Pumping Lemma is *not* a characterization of regular languages: it is *not* an if-and-only-if statement.

Proof of the Myhill-Nerode Theorem

It suffices to prove:

- (i) If L is accepted by a DFA with k states, then L has index at most k .
 - (ii) If L has a finite index k (say), then it is accepted by a DFA with k states.
- (i): Suppose L is accepted by a DFA $M = (Q, \Sigma, \delta, q_0, F)$. We check that

$$\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y) \implies x \equiv_L y.$$

(Recall: for $x \in \Sigma^*$, we have $\hat{\delta}(q, x) = q'$ iff $q \xrightarrow{x}^* q'$.)

Take $x_1, \dots, x_{k+1} \in \Sigma^*$, all distinct. Since M has only k states, by the Pigeonhole Principle, for some $1 \leq i < j \leq k + 1$, we have $\hat{\delta}(q_0, x_i) = \hat{\delta}(q_0, x_j)$, and so, $x_i \equiv_L x_j$. It follows that \equiv_L has at most k equivalence classes.

(ii) Assume that L has a finite index. Define a structure $M = (\Sigma, Q, q_0, \delta, F)$ as follows:

$$\begin{aligned} Q &= \{ [x] : x \in \Sigma^* \} \\ q_0 &= [\epsilon] \\ \delta([x], a) &= [xa] \\ F &= \{ [w] : w \in L \} \end{aligned}$$

We need to verify: (a) M is a DFA, (b) M accepts L .

For (a), $|Q|$ is the index of L which is finite, and δ is a well-defined function because $x \equiv_L y$ implies $xa \equiv_L ya$ for any $a \in \Sigma$.

For (b), for any $w \in \Sigma^*$, $w \in L(M)$ iff $[\epsilon] \xrightarrow{w}^* [w] \in F$ iff $w \in L$.

□

Example: $L = \{ ww : w \in \{0, 1\}^* \}$ is not regular

Take any distinct $i, j \geq 0$. We have $0^i 1 \not\equiv_L 0^j 1$ because $0^i 10^i 1 \in L$ but $0^j 10^i 1 \notin L$. Hence \equiv_L has an infinite index. Thus L is not regular by Myhill-Nerode.

Closure Properties of Regular Languages

Regular languages are closed under the following operations:

1. The regular operations: union, concatenation, star
2. Intersection
3. Complementation
4. Word reversal
5. Homomorphism: Given a function $\phi : \Sigma_1 \rightarrow \Sigma_2^*$. Define $\phi^*(a_1 \cdots a_n) = \phi(a_1) \cdots \phi(a_n)$, and for any language L_1 over Σ_1 , define

$$\phi(L_1) \stackrel{\text{def}}{=} \{ \phi^*(w) : w \in L_1 \}.$$

If L_1 is regular, so is $\phi(L_1)$.

6. Inverse homomorphism: For any L_2 over Σ_2 . Define

$$\phi^{-1}(L_2) \stackrel{\text{def}}{=} \{w \in \Sigma_1^* : \phi(w) \in L_2\}.$$

If L_2 is regular, so is $\phi^{-1}(L_2)$.