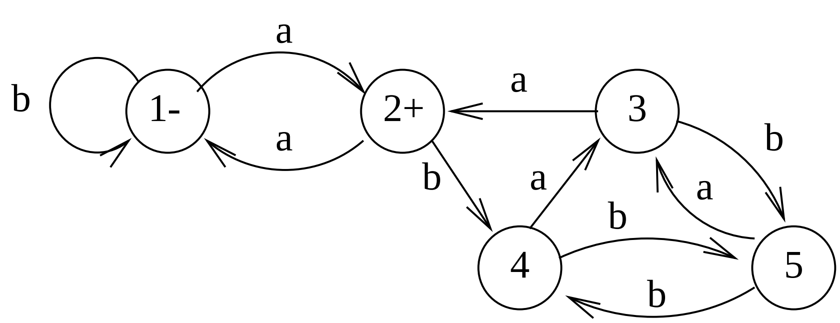
Chapter 10

Nonregular Languages

10.1 Introduction

Example: Consider the following FA having 5 states:



- Let's process the string *ababbaa* on the FA:

$$1 \xrightarrow{a} 2 \xrightarrow{b} 4 \xrightarrow{a} 3 \xrightarrow{b} 5 \xrightarrow{b} 4 \xrightarrow{a} 3 \xrightarrow{a} 2$$

- Since 2 is a final state, we accept the string *ababbaa*.
- In general,
  - We always start in initial state.
  - After reading first letter of input string,

    * we end may go to another state or return to initial state.
    * the maximum number of different states that we could have visited after reading the first letter is 2.
  - After reading the first 2 letters of input string, the maximum number of different states that we could have visited is 3.
  - In general, after reading the first m letters of input string, the maximum number of different states that we could have visited is $m + 1$.

- In our example above, after reading 5 letters, the maximum number of different states that we could have visited is $5 + 1 = 6$. But since the FA has 5 states, we know that after reading in 5 letters, we must have visited some state twice.

- Consider the string $aaabaa$.

  - The string has length 6, which is more than the number of states in the above FA.
  - We process the string as follows:

$$1 \xrightarrow{a} 2 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{b} 4 \xrightarrow{a} 3 \xrightarrow{a} 2$$

    and so it is accepted.
  - Notice that state 1 is the first state that we visit twice.

- In general, if we have an FA with N states and we process a string w with length$(w) \geq N$, then there exists at least one state that we visit at least twice.

  - Let u be the first state that we visit twice.
  - Break up string w as $w = xyz$, where x, y, and z are 3 strings such that
    * string x is the letters at the beginning of w that are read by the FA until the state u is hit for the first time.
    * string y is the letters used by the FA starting from the first time we are in state u until we hit state u the second time.
    * string z is the rest of the letters in w.

- For example, for the string $w = ababbaa$ processed on the above FA, we have $u = 2$, and $x = ab$, $y = abb$, $z = aa$.

- For example, for the string $w = aaabaa$ processed on the above FA, we have $u = 1$, and $x = \Lambda$, $y = aa$, $z = abaa$.

10.2 Definition of Nonregular Languages

Definition: A language that cannot be defined by a regular expression is called a *nonregular language*.

By Kleene's Theorem, a nonregular language cannot be accepted by any FA or TG.

- Consider

$$\begin{aligned} L &= \{\Lambda, ab, aabb, aaabbb, aaaabbbb, \ldots\} \\ &= \{a^n b^n : n = 0, 1, 2, \ldots, \} \ \equiv \ \{a^n b^n\} \end{aligned}$$

- We will show that L is a nonregular language by contradiction.
- Suppose that there is some FA that accepts L.
- By definition, this FA must have a finite number of states, say 5.
- Consider the path the FA takes on the word $a^6 b^6$.
- The first 6 letters of the word are a's.
- When processing the first 6 letters, the FA must visit some state u at least twice since there are only 5 states in the FA.
- We say that the path has a *circuit*, which consists of those edges that are taken from the first time u is visited to the second time u is visited.
- Suppose the circuit consists of 3 edges.
- After the first b is read, the path goes elsewhere and eventually we end up in a final state where the word $a^6 b^6$ is accepted.
- Now consider the string $a^{6+3} b^6$.
- When processing the a part of the string, the FA eventually hits state u.
- From state u, we can take the circuit and return to u by using up 3 a's.

- From then on, we read in the rest of the a's exactly as before and go on to read in the 6 b's in the same way as before.
- Thus, when processing $a^{6+3}b^6$, we end up again in a final state.
- Hence, we are supposed to accept a^9b^6.
- However, a^9b^6 is not in L since it does not have an equal number of a's and b's.
- Thus, we have a contradiction, and so L must not be regular.
- We can use the same argument with any string $a^6(a^3)^kb^6$, for $k = 0, 1, 2, \ldots$.

10.3 First Version of Pumping Lemma

Theorem 13 (Pumping Lemma) *Let L be any regular language that has infinitely many words. Then there exists some three strings x, y, and z such that y is not the null string and that all strings of the form*

$$xy^kz \textit{ for } k = 0, 1, 2, \ldots$$

are words in L.

Proof.

- Since L is a regular language, there exists some FA that accepts L by Kleene's theorem.
- FA must have a finite number of states N.
- Since L is an infinite language and since alphabets are always finite, L must consist of arbitrarily long words.
- Consider any word w accepted by FA with length(w) $= m$, and assume that $m \geq N$.
- Since length(w) $= m$, when processing w on the FA, we visit $m + 1$ states, not necessarily all unique.

- Since $m + 1 \geq N + 1$, when processing w on the FA, we visit at least $N + 1$ states.
- But since the FA has only N states in total, some state must be visited twice when processing w on the FA.
- Let u be the first state visited twice when processing the string w on the FA.
- Thus, there is a circuit in FA corresponding to state u for this string w.
- We break up w into three substrings x, y, z:
  1. x consists of the all letters starting at the beginning of w up to those consumed by the FA when state u is reached for the first time. Note that x may be null.
  2. y consists of the letters after x that are consumed by the FA as it travels around the circuit.
  3. z consists of the letters after y to the end of w.
- Note that the following statements hold:
  1. $w = xyz$.
  2. Note that y is not null since at least one letter is consumed by traveling around the circuit. The circuit starts in a particular state, and ends in the same state. Thus, traveling the circuit requires at least one transition, which means that at least one letter is consumed.
  3. The strings x and y satisfy length(x) + length$(y) \leq N$, which we can show as follows. Let v be the string xy except for the last letter of xy. By the way that we constructed x and y, when we process v on the FA starting in the initial state, we never visit any state twice since it is only on reading the last letter of y do we first visit some state twice. Thus, processing v on the FA results in visiting at most N states, which corresponds to reading at most $N - 1$ letters. Since xy is the same as v with one more letter attached, we must have that xy has length at most N.
- When processing $w = xyz$,
  - the FA first processes substring x and ends in state u.

  - then it starts processing substring y starting in state u and ends in state u.
  - then it starts processing substring z starting in state u and ends in some final state v.

- Now process the word $xyyz$ on FA.
  - For the substring x, the FA follows exactly the same path as when it processed the x-part of w.
  - For the first substring u, the FA starts in state u and returns to state u.
  - For the second substring u, the FA starts in state u and returns to state u.
  - For the substring z, the FA starts in u and processes exactly as before for the word w, and so it ends in the final state v.
  - Thus, $xyyz$ is accepted by FA.

- Similarly, we can show that any string xy^kz, $k = 0, 1, 2, \ldots$, is accepted by FA.

■

10.4 Another Version of Pumping Lemma

Theorem 14 *Let L be a language accepted by an FA with N states. Then for all words $w \in L$ such that length$(w) \geq N$, there are strings x, y, and z such that*

P1. $w = xyz$;

P2. *y is not null;*

P3. *length(x) + length$(y) \leq N$;*

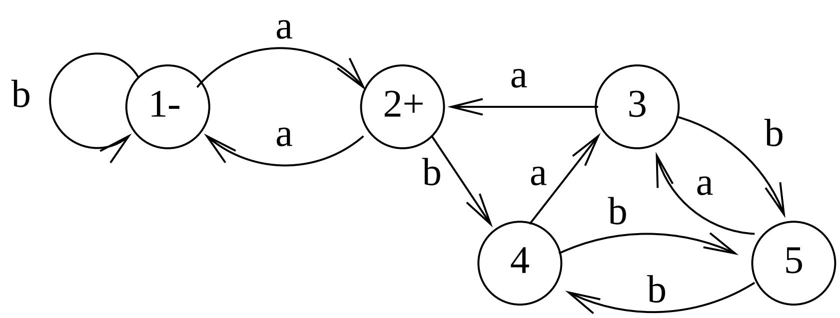
P4. $xy^kz \in L$ *for all* $k = 0, 1, 2, \ldots$.

Proof. The proof of Theorem 13 actually establishes Theorem 14. ∎

Remarks:

- In the textbook Theorem 14 also assumes that L is infinite. However, this additional assumption is not needed.

Example:



$w = ababbaa$
$x = ab$
$y = abb$
$z = aa$

Example: Prove $L =$ PALINDROME is nonregular.

We cannot use the first version of the Pumping Lemma (Theorem 13) since

$$x = a, \quad y = b, \quad z = a,$$

satisfy the lemma and do not contradict the language since all words of the form

$$xy^k z = ab^k a$$

are in PALINDROME.

We will instead apply Theorem 14 to show that PALINDROME is nonregular.

Proof.

- Suppose that PALINDROME is a regular language.
- Then by definition, PALINDROME must have a regular expression.
- Kleene's Theorem then implies that there is a finite automaton for PALINDROME.
- Assume that the FA for PALINDROME has N states, for some $N \geq 1$.
- Consider the string

  $$w = a^N b a^N$$

  which is in PALINDROME.
- Note that length$(w) = 2N + 1 \geq N$.
- Thus, all of the assumptions of Theorem 14 hold, so the conclusions of Theorem 14 must hold; i.e., there exist strings x, y, and z such that

  P1. $w = xyz$;

  P2. y is not null;

  P3. length(x) + length$(y) \leq N$;

  P4. $xy^k z \in L$ for all $k = 0, 1, 2, \ldots$.
- P1 of Theorem 14 says that $w = xyz$, so
  - x must be at the beginning of w,
  - y must be somewhere in the middle of w,

  - z must be at the end of w.

- P2 of Theorem 14 says that x and y together have at most N letters.
- Since w has N a's in the beginning and x and y are at the beginning of w, x and y must consist solely of a's.
- P1 and P3 of Theorem 14 imply that x and y must consist solely of a's.
- Since z is the rest of the string after x and y, we must have that z consists of zero or more a's, followed by 1 b and then N a's.
- In other words,

$$\begin{aligned} x &= a^i \text{ for some } i \geq 0, \\ y &= a^j \text{ for some } j \geq 0, \\ z &= a^l b a^N \text{ for some } l \geq 0. \end{aligned}$$

- Since $y \neq \Lambda$ by P2 of Theorem 14, we must have $j \geq 1$.
- Also, since $w = xyz$ by P1 of Theorem 14, note that

$$w = a^N b a^N = xyz = a^i a^j a^l b a^N = a^{i+j+l} b a^N,$$

  so $i + j + l = N$.
- Now consider the string $xyyz$, which is supposed to be in L by P4 of Theorem 14.
- Note that

$$xyyz = a^i a^j a^j a^l b a^N = a^{i+2j+l} b a^N = a^{N+j} b a^N$$

  since $i + j + l = N$.
- But $a^{N+j} b a^N \notin$ PALINDROME since reverse$(a^{N+j} b a^N) \neq a^{N+j} b a^N$.
- This is a contradiction, and so PALINDROME must be nonregular.

∎

Can use first version of Pumping Lemma (Theorem 13) to show that $L = \{a^n b^n : n \geq 0\}$ is not a regular language:

- Suppose L is a regular language.
- Pumping Lemma says that there exist strings x, y, and z such that all words of the form xy^kz are in L, where y is not null.
- All words in L are of the form a^nb^n.
- How do we break up a^nb^n into substrings x, y, z with y nonempty?
  - If y consists solely of a's, then $xyyz$ has more a's than b's, and so it is not in L.
  - If y consists solely of b's, then $xyyz$ has more b's than a's, and so it is not in L.
  - If y consists of a's and b's, then all of the a's in y must come before all of the b's. However, $xyyz$ then has some b's appearing before some a's, and so $xyyz$ is not in L.
- Thus, L is not a regular language.

Example:

- Let $\Sigma = \{a, b\}$.
- For any string $w \in \Sigma^*$, define $n_a(w)$ to be the number of a's in w, and $n_b(w)$ to be the number of b's in w.
- Define the language $L = \{w \in \Sigma^* : n_a(w) \geq n_b(w)\}$; i.e., L consists of strings w for which the number of a's in w is at least as large as the number of b's in w.
- For example, $abbaa \in L$ since the string has 3 a's and 2 b's, and $3 \geq 2$.
- We can prove that L is a nonregular language using the pumping lemma.
- What string $w \in L$ should we use to get a contradiction?

Example: Consider the language EQUAL $= \{\Lambda, ab, ba, aabb, abab, abba, baba, bbaa, \ldots\}$, which consists of all words having an equal number of a's and b's. We now prove that EQUAL is a non-regular language.

- We will prove this by contradiction, so suppose that EQUAL is a regular language.
- Note that $\{a^n b^n : n \geq 0\} = \mathbf{a}^*\mathbf{b}^* \ \cap$ EQUAL
- Recall that the intersection of two regular languages is a regular language.
- Note that $\mathbf{a}^*\mathbf{b}^*$ is a regular expression, and so its language is regular.
- If EQUAL were a regular language, then $\{a^n b^n : n \geq 0\}$ would be the intersection of two regular languages.
- This would imply that $\{a^n b^n : n \geq 0\}$ is a regular language, which is not true.
- Thus, EQUAL must not be a regular language.

10.5 Prefix Languages

Definition: If R and Q are languages, then Pref(Q in R) is the language of "the prefixes of Q in R," which is the set of all strings of letters that can be concatenated to the front of some word in Q to produce some word in R; i.e.,

$$\text{Pref}(Q \text{ in } R) = \{ \text{ strings } p : \exists\ q \in Q \text{ such that } pq \in R \ \}$$

Example: $Q = \{aba,\ aaabb,\ baaaba,\ bbaaaabb,\ aaaa\}$
$R = \{baabaaba,\ aaabb,\ abbabbaaaabb\}$
Pref(Q in R) $= \{baaba,\ \Lambda,\ abbabba,\ abba\}$

Example: $Q = \{aba,\ aaabb,\ baaaba,\ bbaaaabb,\ aaaa\}$
$R = \{baab,\ ababb\}$
Pref(Q in R) $= \emptyset$

Example: $Q = \mathbf{ab}^*\mathbf{a}$
$R = (\mathbf{ba})^*$
Pref(Q in R) $= (\mathbf{ba})^*\mathbf{b}$

Theorem 16 *If R is a regular language and Q is any language whatsoever, then the language*

$$P = \mathit{Pref}(Q \text{ in } R)$$

is regular.

Proof.

- Since R is a regular language, it has some finite automaton FA_1 that accepts it.
- FA_1 has one start state and several (possibly none or one) final states.
- For each state s in FA_1, do the following:
  - Using s as the start state, process all words in the language Q on FA_1.
  - When starting s, if some word in Q ends in the final state of FA_1, then paint state s blue.
- So for each state s in FA_1 that is painted blue, there exists some word in Q that can be processed on FA_1 starting from s and end up in a final state.
- Now construct another machine FA_2:
  - FA_2 has the same states and arcs as FA_1.
  - The start state of FA_2 is the same as that of FA_1.
  - The final states of FA_2 are the ones that were previously painted blue (regardless if they were final states in FA_1).
- We will now show that FA_2 accepts exactly the prefix language

  $$P = \text{Pref}(Q \text{ in } R).$$

- To prove this, we have to show two things:
  - Every word in P is accepted by FA_2.
  - Every word accepted by FA_2 is in P.
- First, we show that every word accepted by FA_2 is in P.

  - Consider any word w accepted by FA_2.
  - Starting in the start state of FA_2, process the word w on FA_2, and we end up in a final state of FA_2.
  - Final states of FA_2 were painted blue.
  - Now we can start from here and process some word from Q and end up in a final state of FA_1.
  - Thus, the word $w \in P$.

- Now we prove that every word in P is accepted by FA_2.
  - Consider any word $p \in P$.
  - By definition, there exists some word $q \in Q$ and a word $w \in R$ such that $pq = w$.
  - This implies that if pq is processed on FA_1, then we end up in a final state of FA_1.
  - When processing the string pq on FA_1, consider the state s we are in just after finishing processing p and at the beginning of processing q.
  - State s must be a blue state since we can start here and process q and end in a final state.
  - Hence, by processing p, we must start in the start state and end in state s.
  - Thus, p is accepted by FA_2.