CS112: Theory of Computation (LFA)

Lecture 7: Nonregular Languages

Dumitru Bogdan

Faculty of Computer Science
University of Bucharest

April 8, 2025

Table of contents

Section 1

Previously on CS112

Equivalence with finite automata

Theorem

A language is regular if and only if some regular expression describes it.

Proof.

Two directions proof. We state and prove each direction as a separate lemma □

Equivalence with finite automata

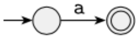
Lemma

If a language is described by a regular expression, then it is regular ($\Leftarrow$).

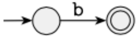
Lemma

If a language is regular, then it is described by a regular expression ($\Rightarrow$).

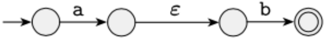
a



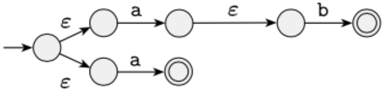
b



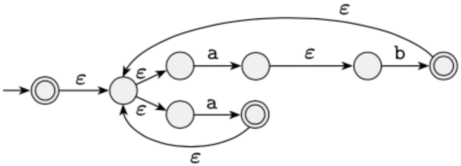
ab



$ab \cup a$



$(ab \cup a)^*$



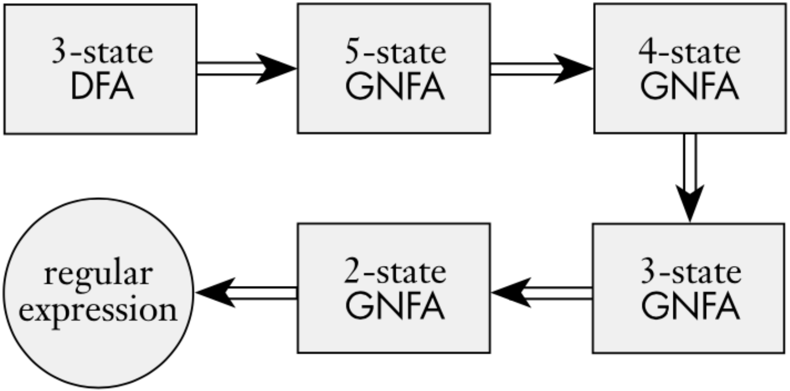


Figure: Typical stages in converting a DFA to a regular expression

$\Longrightarrow$

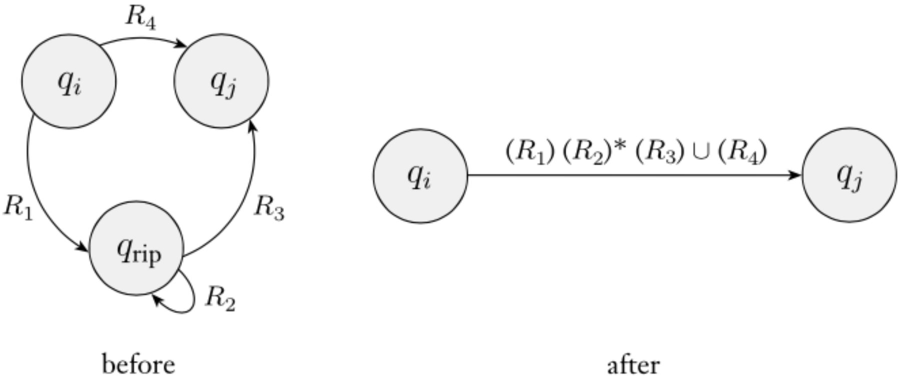


Figure: Constructing an equivalent GNFA with one fewer state

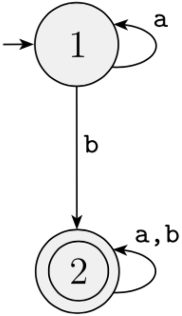


Figure: The NFA N_1

Let us take a two-state DFA.

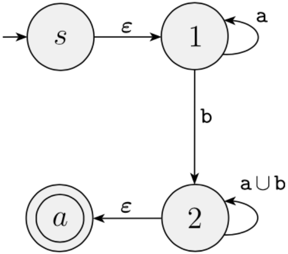


Figure: The NFA N_1

- We make a four-state GNFA by adding a new start state and a new accept state, called s and a for convenience
- We replace a, b on the self loop at state 2 with $a \cup b$

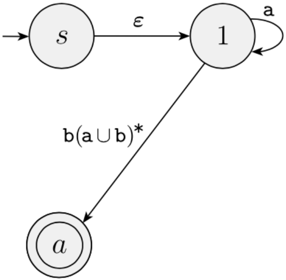


Figure: The NFA N_1

- We remove state 2 and update the remaining arrow labels
- The only label that changes is the one from 1 to a at it becomes $b(a \cup b)^*$
- q_i is state 1, q_j is state a and q_{rip} is 2. So we have $R_1 = b$, $R_2 = a \cup b$, $R_3 = \epsilon$ and $R_4 = \emptyset$.

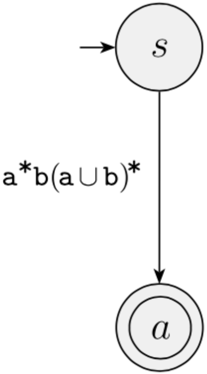


Figure: The NFA N_1

- We remove state 1 and follow the same procedure
- Because only the start and accept states remain, the label on the arrow joining them is the regular expression that is equivalent to the original DFA

Section 2

Context setup

Context setup

Corresponding to Sipser 1.4

Section 3

Nonregular Languages

Nonregular Languages

- Fine automata proved to be quite powerful for such simple model
- However, they are limited in the sense that **there are languages not recognized by any finite automaton**
- For example $B = \{0^n 1^n | n \geq 0\}$. Any attempt to find a DFA that recognize B will fail
- The DFA must remember all number of 0 seen so far and the number is not finite and we cannot do that having finite number of states
- We will study a method for proving that languages such as B are **not regular**

Nonregular Languages

- Let look at this two languages over $\Sigma = \{0, 1\}$

$$C = \{w | w \text{ has an equal number of 0s and 1s}\}$$
$$D = \{w | w \text{ has an equal number of occurrences of 01 and 10 as substrings}\}$$

- C is not regular but D is ($\Leftarrow$ first to find it will get a CS112 T-shirt) which is contrary to out intuition
- In this lecture we show how to prove that certain languages are not regular

Section 4

Pumping Lemma for Regular Languages

Pumping Lemma for Regular Languages

- A technique for proving nonregularity stems from a theorem about regular languages, traditionally called the **pumping lemma**
- This theorem states that **all regular languages have a special property**
- If we can show that a language **does not have this property**, we are guaranteed that it **is not regular**
- The property states that all strings in the language can be "pumped" if they are at least as long as a certain special value, called **the pumping length**
- That means each such string **contains a section that can be repeated any number of times** with the resulting string remaining in the language

Pumping Lemma for Regular Languages

Theorem

If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. *for each $i \geq 0$, $xy^iz \in A$*
2. $|y| > 0$
3. $|xy| \leq p$

- When s is divided into xyz, either x or z may be ϵ, but condition 2 says that $y \neq \epsilon$. Without condition 2 the theorem would be trivially true
- Condition 3 states that the pieces x and y together have length at most p. It is an extra technical condition that we occasionally find useful when proving certain languages to be nonregular

Pumping Lemma for Regular Languages

Proof idea

Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA that recognize A.

- We assign the pumping length p to be the number of states of M. We show that any string s in A of length at least p may be broken into the three pieces xyz, satisfying our three conditions.
- What if no strings in A are of length at least p? Then our task is even easier because the theorem becomes vacuously true: Obviously the three conditions hold for all strings of length at least p if there aren't any such strings :)
- If s in A has length at least p, consider the sequence of states that M goes through when computing with input s. It starts with q_1 the start state, then goes to, say, q_3, then, say, q_{20}, then q_9, and so on, until it reaches the end of s in state q_{13}. With s in A, we know that M accepts s, so q_{13} is an accept state

Pumping Lemma for Regular Languages

Proof idea

- If we let n be the length of s, the sequence of states $q_1, q_3, q_{20}, q_9, \ldots, q_{13}$ has length $n+1$
- Because n is at least p, we know that $n+1$ is greater than p, the number of states of M
- Therefore, the sequence must contain a repeated state (by pigeonhole principle)

Pumping Lemma for Regular Languages

The following figure shows the string s and the sequence of states that M goes through when processing s. State q_9 is the one that repeats:

$$
\begin{array}{cccccccccccccccccc}
s = & & s_1 & & s_2 & & s_3 & & s_4 & & s_5 & & s_6 & & \cdots & & s_n & \\
& \uparrow & & \uparrow & & \uparrow & & \uparrow & & \uparrow & & \uparrow & & \uparrow & & \uparrow & & \uparrow \\
& q_1 & & q_3 & & q_{20} & & q_9 & & q_{17} & & q_9 & & q_6 & & q_{35} & & q_{13}
\end{array}
$$

Figure: Example showing state q_9 repeating when M reads s

Pumping Lemma for Regular Languages

Proof idea

- We now divide s into the three pieces x, y, and z. Piece x is the part of s appearing before q_9, piece y is the part between the two appearances of q_9, and piece z is the remaining part of s, coming after the second occurrence of q_9
- So x takes M from the state q_1 to q_9, y takes M from q_9 back to q_9 and z takes M from q_9 to the accept state q_{13}

Pumping Lemma for Regular Languages

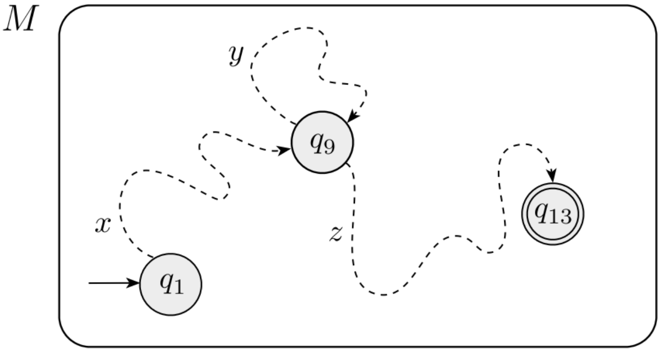


Figure: Example showing how the strings x, y, and z affect M

Pumping Lemma for Regular Languages

Proof idea

Now we check why this division of s satisfies the three conditions:

- Suppose that we run M on input $xyyz$
- We know that x takes M from q_1 and then first y takes it from q_9 back to q_9 and the second y does the same. At last z takes it to q_{13}
- Since q_{13} is an accept state, M accepts input $xyyz$
- Similary, will accept xy^iz for any $i > 0$. If $i = 0$ then $xy^iz = xz$ also accepted. So, condition 1 is satisfied
- We see that $|y| > 0$ as it was the part of s that occurred between two different occurrences of state q_9. So, condition 2 is satisfied
- To get condition 3, we make sure that q_9 is the first repetition in the sequence. By pigeonhole priciple, the first $p + 1$ states in the sequence, must contain a repetition. So, $|xy| \leq p$

Pumping Lemma for Regular Languages

Theorem

If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. *for each* $i \geq 0$, $xy^iz \in A$
2. $|y| > 0$
3. $|xy| \leq p$

Pumping Lemma for Regular Languages

Proof.

Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA that recognize A. Let p be the number of states of M. Let $s = s_1 s_2 \dots s_n$ be a string in A of length $n \geq p$.
Let $r_1, \dots, r_{n+1}$ be the sequence of states that M enters while processing s, so $r_{i+1} = \delta(r_i, s_i)$ for $1 \leq i \leq n$. This sequence has length $n+1$, which is at least $p+1$. Among the first $p+1$ elements in the sequence, two must be the same state, by the pigeonhole principle. We call the first of these r_j and the second r_l. Because r_l occurs among the first $p+1$ places in a sequence starting at r_1, we have $l \leq p+1$. Now let $x = s_1 \dots s_{j-1}$, $y = s_j \dots s_{l-1}$ and $z = s_l \dots s_n$
As x takes M from r_1 to r_j, y takes M from r_j to r_j and z takes M from r_j to r_{n+1}, which is an accept state, M must accept xy^iz for $i \geq 0$. We know that $j \neq l$ so $|y| > 0$ and $l \leq p+1$ so $|xy| \leq p$. So we have satisfied all conditions of the pumping lemma. $\square$

Section 5

Examples

Example 1 I

- Let B be the language $\{0^n 1^n | n \geq 0\}$
- We use the pumping lemma to prove that B is not regular
- The proof is by contradiction
- Assume to the contrary that B is regular. Let p be the pumping length given by the pumping lemma
- Choose s to be the string $0^p 1^p$
- Because s is a member of B and s has length more than p, the pumping lemma guarantees that s can be split into three pieces, $s = xyz$, where for any $i \geq 0$ the string $xy^i z$ is in B. We consider three cases to show that this result is impossible
- The string y consists only of 0s. In this case, the string $xyyz$ has more 0s than 1s and so is not a member of B, violating condition 1 of the pumping lemma. This case is a contradiction.
- The string y consists only of 1s. This case also gives a contradiction

Example 1 II

- The string y consists of both 0s and 1s. In this case, the string $xyyz$ may have the same number of 0s and 1s, but they will be out of order with some 1s before 0s. Hence it is not a member of B, which is a contradiction
- Thus a contradiction is unavoidable if we make the assumption that B is regular, so B is not regular. Note that we can simplify this argument by applying condition 3 of the pumping lemma to eliminate cases 2 and 3
- In this example, finding the string s was easy because any string in B of length p or more would work. Next examples requires additional care

Example 2 I

- Let C be the language $\{w|w$ has an equal number of 0s and 1s$\}$. We use the pumping lemma to prove that C is not regular. The proof is by contradiction.
- Assume to the contrary that C is regular. Let p be the pumping length given by the pumping lemma. Let s be the string 0^p1^p. With s being a member of C and having length more than p, the pumping lemma guarantees that s can be split into three pieces, $s = xyz$, where for any $i \geq 0$ the string xy^iz is in C. We would like to show that this outcome is impossible. But it is possible! If we let x and z be the empty string and y be the string 0^p1^p, then xy^iz always has an equal number of 0s and 1s and hence is in C. So it seems that s can be pumped.
- Here condition 3 in the pumping lemma is useful. It stipulates that when pumping s, it must be divided so that $|xy| \leq p$. That restriction on the way that s may be divided makes it easier to show that the string $s = 0^p1^p$ we selected cannot be pumped. If $|xy| \leq p$, then y must consist only of 0s, so $xyyz \notin C$.

Example 2 II

- Therefore, s cannot be pumped. That gives us the desired contradiction.
- Selecting the string s in this example required more care. If we had chosen $s = (01)^p$ instead, we would have run into trouble because we need a string that cannot be pumped and that string can be pumped, even taking condition 3 into account.
- Can you see how to pump it? One way to do so sets $x = \epsilon$, $y = 01$ and $y = (01)^{p-1}$. Then $xy^iz \in C$ for every value of i. If you fail on your first attempt to find a string that cannot be pumped, don't despair.
- An alternative method of proving that C is nonregular follows from our knowledge that B is nonregular. If C were regular then $C \cap 0^*1^*$ will be regular also. Why? ($\Leftarrow$ get a CS112 T-Shirt)

Example 3 I

- Let D be the language $\{1^{n^2} | n \geq 0\}$
- We use the pumping lemma to prove that D is not regular
- The proof is by contradiction
- Assume to the contrary that D is regular. Let p be the pumping length given by the pumping lemma
- Let s be the string 1^{p^2}
- Because s is a member of D and s has length more than p, the pumping lemma guarantees that s can be split into three pieces, $s = xyz$, , where for any $i \geq 0$ the string xy^iz is in D
- Notice the gap between successive members of this sequence: $0, 1, 4, 9, 16, 25, 36, 49, \ldots$
- Large members of this sequence cannot be near each other
- Now consider the two strings xyx and xy^2z. These strings differ from each other by a single repetition of y so their lengths differ by the length of y

Example 3 II

- By condition 3 of the pumping lemma $|xy| \leq p$ meaning $|y| \leq p$. We have $|xyz| = p^2$ and $|xy^2z| \leq p^2 + p$. But $p^2 + p < (p+1)^2$
- Therefore, the length of xy^2z lies strictly between the consecutive perfect squares. Hence this length cannot be a perfect square itself. Contradiction