FORMAL LANGUAGES AND AUTOMATA THEORY

Module 2

Rijin IK

Assistant Professor
Department of Computer Science and Engineering
Vimal Jyothi Engineering College
Chemperi

September 24, 2023

Outline

Course Outcomes

After the completion of the course the student will be able to

1. Classify a given formal language into Regular, Context-Free, Context Sensitive, Recursive or Recursively Enumerable. [Cognitive knowledge level: Understand]
2. Explain a formal representation of a given regular language as a finite state automaton, regular grammar, regular expression and Myhill-Nerode relation. [Cognitive knowledge level: Understand]
3. Design a Pushdown Automaton and a Context-Free Grammar for a given context-free language. [Cognitive knowledge level : Apply]
4. Design Turing machines as language acceptors or transducers. [Cognitive knowledge level: Apply]
5. Explain the notion of decidability. [Cognitive knowledge level: Understand]

Regular Expression (RE)

- The languages accepted by finite automata are easily described by simple expressions called regular expressions.

Formal Definition of Regular Expression (RE)

- Let Σ be an alphabet. The regular expressions over Σ and the sets that they denote are defined recursively as follows.
  1. ϕ is a regular expression and denotes the empty set.
  2. ϵ is a regular expression and denotes the set $\{\epsilon\}$.
  3. For each a in Σ, a is a regular expression and denotes the set $\{a\}$.
  4. If r and s are regular expressions denoting the languages R and S, respectively, then

$$(r+s),(rs),\ and\ (r^*)$$

are regular expressions that denote the sets

$$R \cup S, RS, and\ R^*, \text{respectively.}$$

Regular Expression (RE)

Q:Write regular expressions for each of the following languages over $\Sigma = \{0, 1\}$.

1. The set representing $\{00\}$.
    - 00
2. The set representing all strings of 0's and 1's.
    - $(0 + 1)^*$
3. The set of all strings representing with at least two consecutive 0's.
    - $(0 + 1)^*00(0 + 1)^*$
4. The set of all strings ending in 011.
    - $(0 + 1)^*011$
5. The set of all strings representing any number of 0's followed by any number of 1's followed by any number of 2's.
    - $0^*1^*2^*$
6. The set of all strings starting with 011.
    - $011(0 + 1)^*$

Identity Rules Related to Regular Expressions

Given r, s and t are regular expressions, the following identities hold:

$$
\begin{aligned}
\phi^* &= \epsilon \\
\epsilon^* &= \epsilon \\
r^+ &= rr^* = r^* r \\
r^* r^* &= r^* \\
(r^*)^* &= r^* \\
r + s &= s + r \\
(r + s) + t &= r + (s + t) \\
(rs)t &= r(st) \\
r(s + t) &= rs + rt \\
(r + s)t &= rt + st \\
(\epsilon + r)^* &= r^*
\end{aligned}
$$

Regular Expression (RE)

Identity Rules Related to Regular Expressions cont..

$$
\begin{aligned}
(r+s)^* &= (r^*s^*)^* = (r^*+s^*)^* = (r+s^*)^* \\
r+\phi &= \phi+r = r \\
r\epsilon &= \epsilon r = r \\
\phi L &= L\phi = \phi \\
r+r &= r \\
\epsilon + rr^* &= \epsilon + r^*r = r^*
\end{aligned}
$$

Regular Expression (RE)

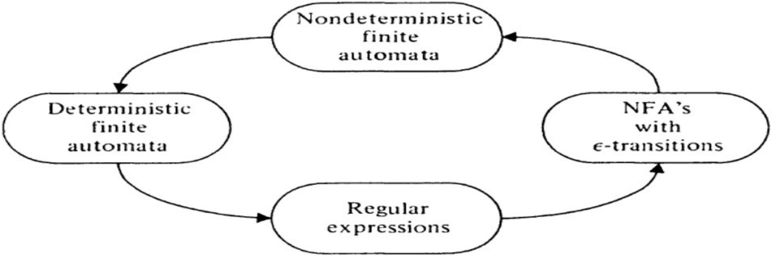


Figure: Equivalence of Finite Automata and Regular Expressions

Equivalence of Finite Automata and Regular Expressions

- The languages accepted by finite automata are precisely the languages denoted by regular expressions.
- For every regular expression there is an equivalent NFA with $\epsilon - transitions$.
- For every DFA there is a regular expression denoting its language.

Theorem

Let r be a regular expression then there exists an NFA with $\epsilon - transition$ that accept $L(r)$

Proof

Zero operators

- The expression r must be ϵ, ϕ, or a for some a in Σ. The NFA's for zero operators are

(a) $r = \epsilon$ (b) $r = \varnothing$ (c) $r = \mathbf{a}$

Proof cont.. **One or more operators**

- Let r have i operators. There are three cases depending on the form of r.

Case 1: Union ($r = r1 + r2.$)

- There are NFA's $M1 = (Q1, \Sigma1, \delta1, q1, \{f1\})$ and $M2 = (Q2, \Sigma2, \delta2, q2, \{f2\})$ with $L(M1) = L(r1)$ and $L(M2) = L(r2)$.Construct $M = (Q1 \cup Q2 \cup \{q0, f0\}, \Sigma1 \cup \Sigma2, \delta, q0, \{f0\})$ where δ is defined by

$$
\begin{aligned}
\delta(q0, \epsilon) &= \{q1, q2\} \\
\delta(q, a) &= \delta1(q, a) \; for \; q \; in \; Q1 - \{f1\} \; and \; a \; in \; \Sigma1 \cup \{\epsilon\} \\
\delta(q, a) &= \delta2(q, a) \; for \; q \; in \; Q2 - \{f2\} \; and \; a \; in \Sigma2 \cup \{\epsilon\} \\
\delta(f1, \epsilon) &= \delta1(f2, \epsilon) = \{f0\}
\end{aligned}
$$

Proof cont..

$L(M) = L(M1) \cup L(M2)$

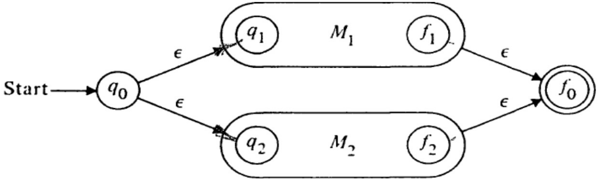


Figure: $L(M) = L(M1) \cup L(M2)$

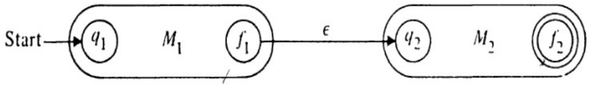
Proof cont..

Case 2: Concatenation $(r = r1r2)$

- Let M1 and M2 be as in Case 1 and construct $M = (Q1 \cup Q2, \Sigma1 \cup \sigma2, \delta, q1, \{f2\})$
- where δ is defined by

$$\begin{aligned}
\delta(q,a) &= \delta1(q,a) \textit{ for } q \textit{ in } Q1 - \{f1\} \textit{ and } a \textit{ in } \Sigma1 \cup \{\epsilon\} \\
\delta(f1,\epsilon) &= \{q2\} \\
\delta(q,a) &= \delta2(q,a) \textit{ for } q \textit{ in } Q2 \textit{ and } a \textit{ in } \Sigma2 \cup \{\epsilon\}
\end{aligned}$$

$L(M) = \{xy|\ x \textit{ is in } L(M1) \textit{ and } y \textit{ is in } L(M2)\} \textit{ and } L(M) = L(M1)L(M2)$

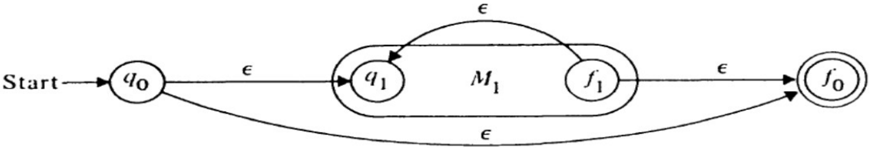


Proof cont..

Case 3: $Closure(r = r1^*)$

- Let $M1 = (Q1, \Sigma1, \delta1, q1, \{f1\})$ *and* $L(M1) = r1$.
- Construct $M = (Q1 \cup \{q0, f0\}, \Sigma1, \delta, q0, \{f0\})$, where δ is defined by

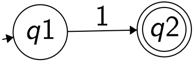
$$\begin{aligned} \delta(q0, \epsilon) &= \delta(f1, \epsilon) = \{q1, f0\} \\ \delta(q, a) &= \delta1(q, a) \text{ for } q \text{ in } Q1 - \{f1\} \text{ and } a \text{ in } \Sigma1 \cup \{\epsilon\} \end{aligned}$$



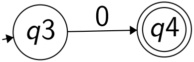
Equivalence of Finite Automata and Regular Expressions

Q: Construct an NFA for the regular expression $01^* + 1$

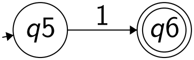
- Regular expression is of the form $r1 + r2$, where $r1 = 01^*$ and $r2 = 1$. The automaton for r2 is



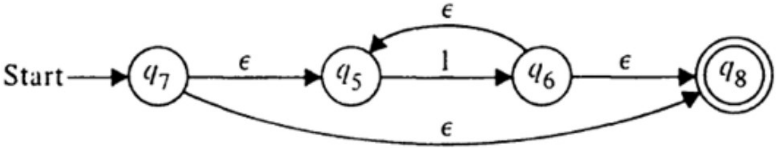
- Express r1 as r3 and r4, where r3=0 and $r4 = 1^*$ The automaton for r3 is



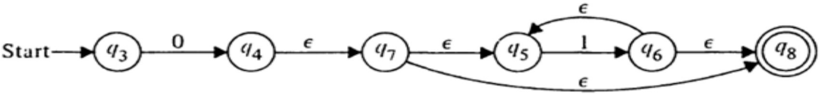
- r4 is $r5^*$ where r5=1 The NFA for r5 is



- To construct an NFA for $r4 = r5^*$ use the construction of closure. The resulting NFA for r4 is

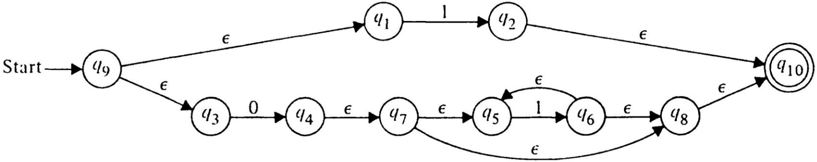


- Then, for $r1 = r3r4$ use the construction of concatenation.



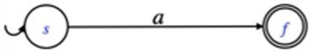
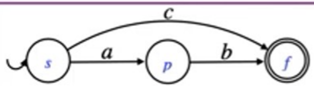
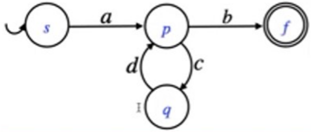
Equivalence of Finite Automata and Regular Expressions

- Finally, use the construction of union to find the NFA for $r = r1 + r2$



Conversion of Finite Automata to Regular Expression

- Any Regular Language can be represented by a Regular Expression.

NFA $\mathscr{A}$	**Language** $L(\mathscr{A})$	**RE** $R_{\mathscr{A}}$
	$\{a\}$	a
	$\{c, ab\}$	$c + ab$
	$\{ab, ac \cdot (dc)^i \cdot db \mid i \geq 0\}$	$ab + ac \cdot (dc)^* \cdot db$

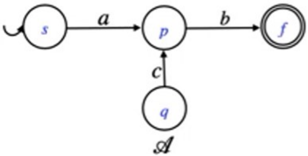
Equivalence of Finite Automata and Regular Expressions

Let A be an NFA over an alphabet set Σ. Then:

$$L_{p,q} = \{w \in \Sigma^* \,|\, \exists a \text{ path from } p \text{ to } q \text{ with label as } w\}$$

where p,q $\in$ Q

Example



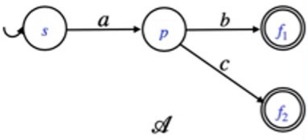
Set of Strings $L_{q,f}$	RE
$L_{q,f} = \{cb\}$	cb

Let A be an NFA over an alphabet set Σ. Then the language of A

$$L(A) = \bigcup_{f \in F} L_{s,f}$$

where Q is the set of all states and F is the final state

Example



Strings/Language	RE
$L_{s,f_1} = \{ab\}$	$R_{L_{s,f_1}} = ab$
$L_{s,f_2} = \{ac\}$	$R_{L_{s,f_2}} = ac$
$L(\mathscr{A}) = L_{s,f_1} \cup L_{s,f_2}$	$R_{L_{s,f_1}} + R_{L_{s,f_2}}$

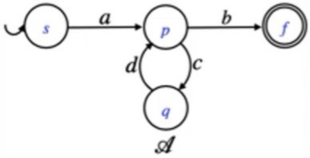
Equivalence of Finite Automata and Regular Expressions

Let A be an NFA over an alphabet set Σ. Then

$$L_{p,q}^{X} = \{w \in \Sigma^* \mid \exists \textit{a path from p q with label as w passing through states in } X\}$$

where p, q $\in$ Q and X $\subseteq$ Q

Example



Set of Strings/Language	RE
$L_{s,f}^{\{p\}} = \{ab\}$	ab
$L_{s,f}^{\{p,q\}} = \{ab, ac \cdot (dc)^i \cdot db \mid i \geq 0\}$	$ab + ac \cdot (dc)^* \cdot db$

Equivalence of Finite Automata and Regular Expressions

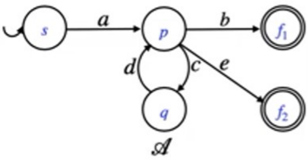
Let A be an NFA over an alphabet set Σ. Then the language of A

$$L(A) = \bigcup_{f \in F} L^{Q}_{s,f}$$

$$R_{L(A)} = \underset{f \in F}{+} R_{L^{Q}_{s,f}}$$

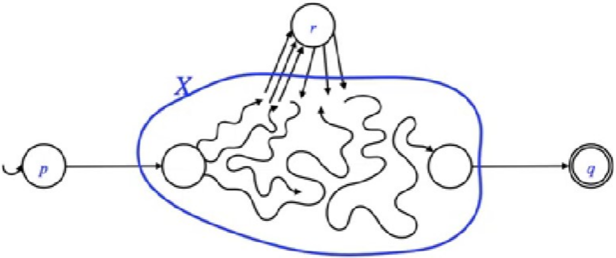
where Q is the set of all states and F is the final state

Example



Strings/Language	RE
$L^{Q}_{s,f_1} = \{ab, ac \cdot (dc)^i \cdot db \mid i \geq 0\}$	$R_{L^{Q}_{s,f_1}} = ab + ac \cdot (dc)^* \cdot db$
$L^{Q}_{s,f_2} = \{ae, ac \cdot (dc)^i \cdot de \mid i \geq 0\}$	$R_{L^{Q}_{s,f_2}} = ae + ac \cdot (dc)^* \cdot de$
$L(\mathscr{A}) = L^{Q}_{s,f_1} \cup L^{Q}_{s,f_2}$	$R_{L^{Q}_{s,f_1}} + R_{L^{Q}_{s,f_2}}$

Inductively defining RE from an NFA



$$R_{L_{p,q}^{X\cup\{r\}}} = R_{L_{p,q}^{X}} + R_{L_{p,r}^{X}} \cdot \left(R_{L_{r,r}^{X}}\right)^{*} \cdot R_{L_{r,q}^{X}}$$

Equivalence of Finite Automata and Regular Expressions

Kleen's Construction

1. Begin with $R_{L^{Q}_{s,f}}$ for all f $\in$ F
2. Simplify using the terms with strictly small X's

$$R_{L^{x\cup\{r\}}_{p,q}} = R_{L^{x}_{p,q}} + R_{L^{x}_{p,r}}.(R_{L^{x}_{r,r}})^{*}.R_{L^{x}_{r,q}}$$

3. For the base terms, observe that:

$$R_{L^{\phi}_{p,q}} = \begin{cases} a, & \text{if } p \neq q \text{ and } \exists \text{ an edge for a from } p \; to \; q \\ a + \epsilon, & \text{if } p = q \text{ and } \exists \text{ an edge for a from } p \; to \; q \end{cases}$$

Equivalence of Finite Automata and Regular Expressions

Construction of regular expressions for the given finite Automata

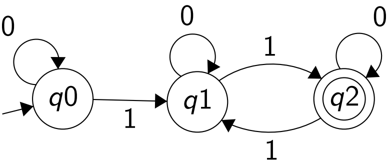
Arden's Theorem

Let P and Q be two regular expressions over Σ,and if P does not contain epsilon, then $R = Q + RP$ has a unique solution $R = QP^*$

Procedure:

- Assume the given finite automata should not contain any epsilons.
  1. Find the reachability for each and every state in given Finite automata.
     - **Reachability** of a state is the set of states whose edges enter into that state.
  2. For the initial state of finite automata ,add epsilon to the reachability equation.
  3. Solve the equations by using Arden's Theorem.
  4. Substitute the results of each state equation into the final state equation,to get the regular expression for the given DFA.

Q: Construct regular expression for the given finite automaton.



Solution:

1. Find the reachability for each and every state in given Finite automata.

$$q_0 = q_0 0 \quad (1)$$

$$q_1 = q_0 1 + q_1 0 + q_2 1 \quad (2)$$

$$q_2 = q_1 1 + q_2 0 \quad (3)$$

Solution cont..

2 For the initial state of finite automata, add epsilon to the reachability equation. $q_0 = q0_0 + \epsilon$

3 Solve the equations by using Arden's Theorem. After applying arden's theorem for equation 3

$$q_2 = q_1 10^* \tag{4}$$

Substitute equation 4 in equation 2

$$q_1 = q_0 1 + q_1 0 + q_1 10^* 1$$

$$q_1 = q_0 1 + q_1 (0 + 10^* 1) \tag{5}$$

Apply arden's theorem on equation 5

$$q_1 = q_0 1 (0 + 10^* 1)^* \tag{6}$$

Solution cont.. Apply arden's theorem on equation 1

$$q_0 = q_0 0 + \epsilon$$

$$q_0 = \epsilon 0^* \tag{7}$$

Substitute equation 7 in equation 6

$$q_1 = \epsilon 0^* 1(0 + 10^* 1)^* \tag{8}$$

4 Substitute the results of each state equation into the final state equation, to get the regular expression for the given DFA

$$q_2 = \epsilon 0^* 1(0 + 10^* 1)^* 10^* \tag{9}$$

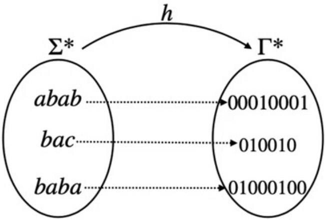
Therefore, the regular expression for the given DFA is $0^* 1(0 + 10^* 1)^* 10^*$

Homomorphisms

- A function mapping strings of one alphabet set to strings of other alphabet set.

Example

- Let $\Sigma = \{a, b, c\}$ and $\Gamma = \{0, 1\}$ be two alphabet sets. Consider a homomorphism h defined as follows:
  $h(abab) = 00010001$ $h(bac) = 010010$ $h(baba) = 01000100$



Homomorphisms

Homomorphisms-Definition

- A Homomorphisms is a function h: $\Sigma^* \rightarrow \Gamma^*$ satisfying the following condition:

$$\forall_{x,y} \in \Sigma^* \Big| h(xy) = h(x)h(y)$$

Example:

- Let $\Sigma = \{a, b, c\}$ and $\Gamma = \{0, 1\}$ be two alphabet sets. consider a homomorphism h defined as follows:
  - $h(a) = 00, h(b) = 01, h(c) = 10$ then ,
  - $h(abcb) = h(a).h(b).h(c).h(b) = 00.01.10.01 = 00011001$
- it follows from the property $h(xy) = h(x)h(y)$ of a homomorphism h that:

$$\forall_{a_i} \in \Sigma \Big| h(a_1 a_2 .....a_n) = h(a_1).h(a_2)...h(a_{n-1}).h(a_n)$$

Homomorphisms

Theorem

If L is a regular language, and h is a homomorphism on its alphabet, then $h(L)$ is also a regular language

Proof

- Let E be a regular expression for L.
- Apply h to each symbol in E.
- Language of resulting RE is h(L).

Homomorphisms

Q:Using homomorphism on Regular Languages, Prove that the language $L = \{a^n b^n c^{2n} | n \geq 0\}$ is not regular. Given that the language $\{a^n b^n : n \geq 0\}$ is not regular.

Solution: Proof by contradiction

- Assume L is regular That means that if h is a homomorphism and L is a regular language then h(L) is also regular.
- In This case you can take the homomorphism $h(a) = a, h(b) = b, h(c) = \epsilon$
- Which maps the language L to the language $h(L) = L' = \{a^n b^n : n \geq 0\}$ which you already know is not regular
- This contradiction shows that the language L is not regular.

Inverse Homomorphisms

- Let h:$\Sigma^* \rightarrow \Gamma^*$ be a homomorphism and $B \subseteq \Gamma^*$. Then its inverse homomorphism $h^{-1} : \Gamma^* \rightarrow \Sigma^*$ is defined as follows.

$$h^{-1}(B) = \{x \in \Sigma^* |\ h(x) \in B\}$$

Example:

- $h(a) = 0, h(b) = 1, h(c) = 01$
- if we take $L = \{0011001\}$ then
- $h^{-1}(L) = \{aabbaab, aabbac, acbaab, acbac\}$

Pumping Lemma for Regular Sets

- Pumping lemma, which is a powerful tool for proving certain languages non-regular.
- It is also useful in the development of algorithms to answer certain questions concerning finite automata, such as whether the language accepted by a given FA is finite or infinite.

Pumping Lemma for regular languages

Lemma

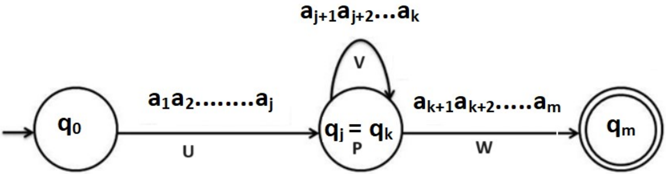
- Let L be a regular set. Then there is a constant n such that if x is any word in L, and $|x| \geq n$, we may write $x = uvw$ in such a way that $|uv| \leq n$, $v \geq 1$, and for all $i \geq 0$, $uv^i w$ is in L.
- Furthermore, n is no greater than the number of states of the smallest FA accepting L.

Proof

- Since L regular there is a DFA M which accept L.
- Let n be the number of states in M
- Consider an input of symbols $x = a_1 a_2 a_3 \ldots ..a_m \;\in L$ where $m \geq n$
- Suppose we have a compassion sequence $q_0, q_1, q_2, ...q_m$, representing the execution of automation M on input string x, and q_m is an accepting state
- Here we have only n distinct state , thus it is not possible for each state of M ($q_0 \; to \; q_m$) to be distinct. So at least two states should be equal. ($q_j = q_k \; say \; P \; where \; 0 \leq j \leq k \leq n$)

ProofCont..

- The path labeled $a_1a_2a_3.\ldots.a_m$ in the transition diagram of M is



- If q_m is in f (i.e $a_1a_2a_3.\ldots..a_m$ is in $L(m)$) we can break the input x into three ie i.e $x = uvw$
  Where

$$\begin{aligned} u &= a_1a_2...a_j \\ v &= a_{j+1}a_{j+2}...a_k \\ w &= a_{k+1}a_{k+2}...a_m \end{aligned}$$

Pumping Lemma for regular languages

ProofCont..

- So the transition function $\hat{\delta}(q_0, x) \in f$ becomes
  - $\hat{\delta}(q_0, u) = p$
  - $\hat{\delta}(p, v) = p$
  - $\hat{\delta}(p, w) = q_m$
  - and also $\hat{\delta}(p, uw) = q_m$

  Now it is clear for any $k \geq 0,\ uv^k w$ takes $q_0\ to\ q_m$ which is accepting. there for $xy^i z\ \in L$

Pumping Lemma for regular languages

Applications of Pumping Lemma

- Pumping Lemma is to be applied to show that certain languages are not regular.

Method to prove that a language L is not regular

1. Assume that L is regular. Let n be the number of states in the corresponding finite automation
2. Choose a string x such that $|x| \geq n$. Use pumping lemma to write $x = uvw$, with $|uv| \leq n$ and $|v| \geq 1$
3. Find a suitable integer i such that $uv^iw \notin L$. This contradicts our assumption. Hence L is not regular.

The important part of proof is Step 3. There, we need to find i such that $uv^iw \notin L$.

- In some cases, we prove $uv^iw \notin L$. by considering $|uv^iw|$.
- In some cases, we may have to use the structure of strings in L.

Pumping Lemma for regular languages

Q:Let $L = \{0^k 1^k : \ k \in N\}$. Prove that L is not regular
Solution: .

- By way of contradiction, suppose L is regular.
- Let n be an integer in the Pumping Lemma.

$$\begin{aligned} Let\ x &= 0^n 1^n \\ Then &\quad x \in L \ \ [definition\ of\ \ L] \\ and\ |x| &= 2n \geq n. \end{aligned}$$

By Pumping Lemma, there are strings u, v, w such that

$$\begin{aligned} (i) &\quad x = uvw, \\ (ii) &\quad |v| \geq 1, \\ (iii) &\quad |uv| \leq n, \\ (iv) &\quad uv^i w \in L \ for\ all\ i \in N. \end{aligned}$$

Solution Cont..

Three cases are there

Case 1: v contain only 0's

- $v = 0^k$ whewr k$\geq$ 1
- there for $w = 0^{n-k}0^k1^n$
    - take i=0

$$\begin{aligned} uv^iw &= uv^0w = uw \\ &= 0^{n-k}0^n \notin L \end{aligned}$$

Case 2: v contain only 1's

- $v = 1^k$ whewr k$\geq$ 1
- there for $w = 0^n1^k1^{n-k}$
    - take i=0

$$\begin{aligned} uv^iw &= uw \\ &= 0^n1^{n-k} \notin L \end{aligned}$$

Pumping Lemma for regular languages

Solution Cont..

Case 3: v contain both 0's and 1's

- $v = 0^l 1^m$ whewr l&m ≥ 1
- there for $w = 0^{n-l}0^l 1^m 1^{n-m}$
  - take i=0

$$\begin{aligned} uv^i w &= uw \\ &= 0^{n-l}1^{n-m} \quad [\notin L \ or \ \in L \ depends \ on \ l \ \& \ m] \end{aligned}$$

  - take i=2

$$\begin{aligned} uv^i w &= uv^2 w \\ &= 0^{n-l}(0^l 1^m)^2 1^{n-m} \\ &= 0^{n-l}0^l 1^m 0^l 1^m 1^{n-m} \\ &= 0^n 1^m 0^l 1^n \quad \notin L \end{aligned}$$

Ultimate periodicity

Ultimate periodic Set

- A set which is periodic after a finite prefix is called an ultimately periodic set.

Example

- Consider the following subset of N_0

$$\{0, 3, 7, 11, 19, 20, 23, 26, 29, 32, 35, 38, 41, 44, 47, 50, ...\}$$

After the element 19, it is a periodic set with period 3. That is it contain every third element in N_0 from 20.

Formal Definition

A set U is called ultimately periodic if ther exists two natural numbers n$\geq$0 and $p > 0$ such that $\forall m \geq n, m \in U \iff m + p \in U$

- For an UP set U neither n nor p may be unique.
- Regular languages over singleton alphabet sets and UP sets are strongly related.

Ultimate periodicity

Theorem

Let A $\subseteq$ {a}*. Then A is regular if and only if the set $\{m|a^m \in A\}$, the set of lengths of strings in A, is ultimately periodic.

Proof:

Part 1: If A is regular, then the set $\{m|a^m \in A\}$ is ultimately periodic.

- Assume that A is regular, which means there exists a finite automaton that recognizes A. We want to show that the set $\{m|a^m \in A\}$ is ultimately periodic.
- Consider the lengths of strings in A. For each m, we want to determine whether a^m is in A.
- We can simulate this using the finite automaton for A. Starting from the initial state, we can repeatedly apply transitions labeled 'a' for m times. If we end up in an accepting state after these m transitions, then a^m is in A. Otherwise, it's not.

Ultimate periodicity

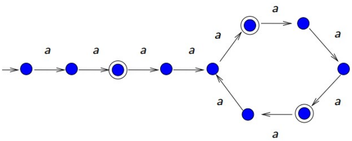
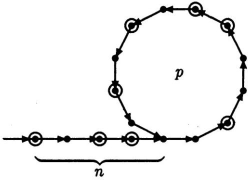


Figure: Example

$lengths(L(A)) = \{2, 5, 8, 11, 14, 17, 20, ...\}$

- Now, observe that as we increase m, we are essentially repeating the same set of states in the finite automaton for A.
- Since there are a finite number of states in the automaton, there must come a point where the states repeat.
- This implies that the set of lengths $\{m|a^m \in A\}$ is ultimately periodic.

Proofcont..



- Let p be the length of the loop,
- and let n be the length of the initial tail preceding the first time we enter the loop.
- For all strings a^m with $m \geq n$, the automaton is in the loop part after scanning a^m . Then a^m is accepted iff $a^m + p$ is,
- since the automaton moves around the loop once under the last p a's of $a^m + p$
- Thus it is in the same state after scanning both strings.
- Therefore, the set of lengths of accepted strings is ultimately periodic.

Proofcont..

Part 2: If the set $\{m | a^m \in A\}$ is ultimately periodic, then A is regular

- Given any ultimately periodic set U,
- let p be the period and let n be the starting point of the periodic behavior.
- Then one can build an automaton with a tail of length n and loop of length p accepting exactly the set of strings in $\{a\}^*$ whose lengths are in U.

Ultimate periodicity

Corollary

Let L be any regular language over an arbitrary set Σ. Then the set $\{|x| \big| x \in L\}$ is ultimately periodic.

Proof

- Take any regular language L.
- Define the homomorphism $h : \Sigma \rightarrow \{a\}$ *by* $h(b) = a$ for all $b \in \Sigma$.
- Then $h(x) = a^{|x|}$. Since h preserves length,
- we have that lengths $A = \textit{lengths } h(A)$.
- But h(L) is a regular subset of $\{a\}^*$, since the regular sets are closed under homomorphic image;
- therefore, lengths h(L) is ultimately periodic.

Application for ultimate periodicity

- From the corollary we know that length set is ultimately periodic is a necessary condition for any regular language.
- Therefore, if we canprove that the length set of a given language L is not ultimately periodic, then we can conclude that L is not regular
- **Example:**
  - Let $L = \{a^{n!} \big| n \geq 0\}$ the length set :$\{1, 2, 6, 24, 120, ...\}$ is not ultimately periodic since the gap is monotonically increasing. Hence L is non regular
- Length set is ultimatelyperiodic is not a sufficient condition for a language to be regular
- **Example:**
  - $L = \{a^n b^n \big| n \geq 0\}$ the length set :$\{0, 2, 4, 6, 8, 10, 12...\}$ is ultimately periodic .But the language is not regular.

Closure Properties of Regular Languages

Let L and M be regular languages. Then the following languages are all regular:

1. Union: $L \cup M$
2. Intersection: $L \cap M$
3. Complement: $\bar{N}$
4. Difference: $L - M$
5. Reversal: $L^R = w^R : w \in L$
6. Kleene Closure: L^*
7. Concatenation: $L.M$
8. Homomorphism:
   - $h(L) = \{h(w) : w \in L, hisahomom.\}$
9. Inverse homomorphism:
   - $h^{-1}(L) = \{w | h(w) \ is \ in \ L\}$

Closure Properties of Regular Languages

Union: $L \cup M$

- For any regular L and M, $L \cup M$ is regular.
- The regular Languages are closed under Union
- Let $L = L(E)$ and $M = L(F)$. Then $L \cup M = L(E + F)$ by the definition of the $+$ operator.

Intersection: $L \cap M$

- If L and M are regular, then so is $L \cap M$.
- The regular Languages are closed under Intersection

Complement: $\bar{N}$

- The complement of a language L (with respect to an alphabet Σ such that $\Sigma *$ contains L) is $\Sigma^* - L$.
- Since $\Sigma *$ is surely regular, the complement of a regular language is always regular.
- The regular Languages are closed under Complement

Closure Properties of Regular Languages

Difference: $L - M$

- If L and M are regular languages, then so is $L–M$ = strings in L but not M.
- The regular Languages are closed under Difference: $L - M$

Reversal: $L^R = w^R : w \in L$

- If L is a regular language, then so is L^R.
- The regular Languages are closed under Reversal

Kleene Closure: L^*

- If L is a regular language,then so is L^*.
- The regular Languages are closed under Kleene Closure

Concatenation: $L.M$

- If L_1 &L_2 are two regular language,then so is $L_1 L_2$ is also regular.
- The regular Languages are closed under Concatenation

Closure Properties of Regular Languages

Homomorphism

- If L is a regular language, and h is a homomorphism on its alphabet, then $h(L) = \{h(w) \big| \ w \ is \ in \ L\}$ is also a regular language.
- The regular Languages are closed under Homomorphism

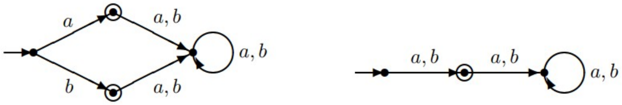
Inverse Homomorphisms

- If h is a homomorphism from alphabet Σ to alphabet T, and L is a regular language over alphabet T, then $h^{-1}(L)$ is also a regular language
- The regular Languages are closed under Inverse Homomorphisms

Minimization of DFA

- Minimization of DFA means reducing the number of states from given FA
- The minimization process consists of two stages
  1. Get rid of inaccessible states
  2. Collapse "equivalent" states

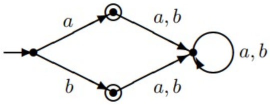
Example

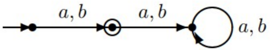


Minimization of DFA

- Minimization of DFA means reducing the number of states from given FA
- The minimization process consists of two stages
  1. Get rid of inaccessible states
  2. Collapse "equivalent" states

Example





Minimization of DFA-The Quotient Construction

One of the popular techniques for DFA state minimization is called the Quotient Construction.

- The Quotient Construction is based on the concept of equivalence classes.
- In this technique, the states of the original DFA are partitioned into different equivalence classes based on their behavior or language recognition capabilities.
- States within the same equivalence class are considered equivalent because they cannot be distinguished by any input sequence; they lead to the same final or non-final state.

Minimization of DFA-The Quotient Construction

- The main idea is a process that takes a DFA and combines states of it in a step-by-step fashion, where each steps yields an equivalent automaton.
  - We never combine a final state and a non-final state. Otherwise the language recognized by the automaton would change.
  - If we merge states p and q, then we have to combine $\delta(p, a)$ and $\delta(q, a)$, for each $a \in \Sigma$
  - Contrarily, if $\delta(p, a)$ *and* $\delta(q, a)$ are not equivalent states, then p and q can not be equivalent.

Minimization of DFA-The Quotient Construction

DFA state equivalence

$$p \approx q \stackrel{def}{\Longleftrightarrow} iff\ \forall x \in \Sigma^*(\hat{\delta}(p,x) \in F \Leftrightarrow \hat{\delta}(q,x) \in F)$$

where F is the set of final states of the automaton

- $\approx$ is an equivalence relation, i.e., it is reflexive, symmetric, and transitive:

$$\begin{aligned} p &\approx p \\ p &\approx q \implies q \approx p \\ p &\approx q \wedge q \approx r \implies p \approx r \end{aligned}$$

Quotient automaton

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. The quotient automaton is $M/\approx = (Q', \Sigma, \delta', q_0', F')$ where

$$\begin{aligned}
Q' &\stackrel{\text{def}}{=} \{[p] \big| p \in Q\} \\
\delta'([p], a) &\stackrel{\text{def}}{=} [\delta(p, a)] \\
q_0' &\stackrel{\text{def}}{=} [q_0] \\
F' &\stackrel{\text{def}}{=} \{[p \big| p \in F]\}
\end{aligned}$$

Quotient automaton

- If M is a DFA that recognizes L, then $M/\approx$ is a DFA that recognizes L. There is no DFA that both recognizes L and has fewer states than $M/\approx$

Minimization of DFA-The Quotient Construction

State Minimization Algorithm

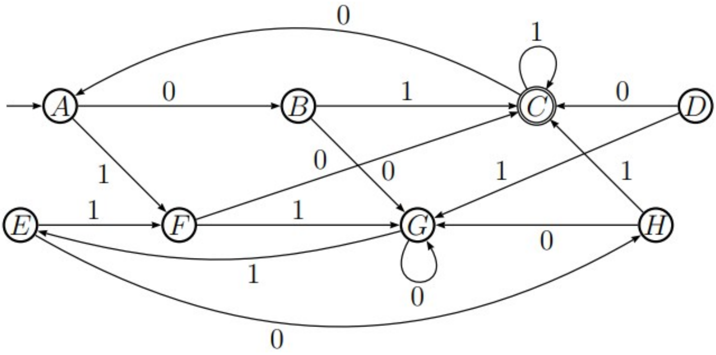
1. Write down a table of all pairs $\{p, q\}$, initially unmarked.
2. Mark $\{p, q\}$ if $p \in F$ and $q \notin F$ or vice versa.
3. Repeat the following until no more changes occur:
   - if there exists an unmarked pair $\{p, q\}$ such that $\{\delta(p, a), \delta(q, a)\}$ is marked for some $a \in \Sigma$, then mark $\{p, q\}$.
4. When done, $p \approx q$ iff $\{p, q\}$ is not marked.

Minimization of DFA-The Quotient Construction

Here are some things to note about this algorithm:

- If $\{p, q\}$ is marked in step 2, then p and q are surely not equivalent: take $x = \epsilon$ in the definition of $\approx$.
- We may have to look at the same pair $\{p, q\}$ many times in step 3, since any change in the table may suddenly allow $\{p, q\}$ to be marked. We stop only after we make an entire pass through the table with no new marks.
- The algorithm runs for only a finite number of steps, since there are only $\begin{pmatrix} n \\ 2 \end{pmatrix}$ possible marks that can be made, and we have to make at least one new mark in each pass to keep going.
- Step 4 is really a statement of the theorem that the algorithm correctly computes $\approx$.

Example: Minimize the following DFA



Minimization of DFA-The Quotient Construction

Solution:

- We start by setting up our table. We will be able to restrict our attention to the lower left triangle, since equivalence is symmetric.
- Also, each box on the diagonal will be marked with $\approx$, since every state is equivalent to itself.
- We also notice that state D is not reachable, so we will ignore it.

	A	B	C	D	E	F	G	H
A	$\approx$	—	—	—	—	—	—	—
B		$\approx$	—	—	—	—	—	—
C			$\approx$	—	—	—	—	—
D	—	—	—	—	—	—	—	—
E				—	$\approx$	—	—	—
F				—		$\approx$	—	—
G				—			$\approx$	—
H				—				$\approx$

Solution cont..

- Now we split the states into final and non-final.
- Thus, a box indexed by p, q will be labelled with an X if p is a final state and q is not, or vice versa.

	A	B	C	D	E	F	G	H
A	$\approx$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
B		$\approx$	$-$	$-$	$-$	$-$	$-$	$-$
C	X_0	X_0	$\approx$	$-$	$-$	$-$	$-$	$-$
D	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
E			X_0	$-$	$\approx$	$-$	$-$	$-$
F			X_0	$-$		$\approx$	$-$	$-$
G			X_0	$-$			$\approx$	$-$
H			X_0	$-$				$\approx$

Solution cont..

- State C is inequivalent to all other states.
- Thus the row and column labelled by C get filled in with X0. (We will subscript each X with the step at which it is inserted into the table.)
- However, note that C, C is not filled in, since C$\approx$ C.
- Now we have the following pairs of states to consider:

$$\{AB, AE, AF, AG, AH, BE, BF, BG, BH, EF, EG, EH, FG, FH, GH\}$$

- Now we introduce some notation which compactly captures how the machine transitions from a pair of states to another pair of states.

$$p_1 p_2 \xleftarrow{0} q_1 q_2 \xrightarrow{1} r_1 r_2$$

- means $q_1 \xrightarrow{0} p_1$ and $q_2 \xrightarrow{0} p_2$ and $q_1 \xrightarrow{1} r_1$ and $q_2 \xrightarrow{1} r_2$

Solution cont..

- If one of $p_1, p_2,\ r_1,\ or\ r_2$ are already marked in the table, then there is a way to distinguish $q_1\ and\ q_2$: they transition to inequivalent states.
- Therefore $q_1 \not\approx q2$ and the box labelled by q_1q_2 will become marked. For example, if we take the state pair AB, we have

$$BG \xleftarrow{0} AB \xrightarrow{1} FC$$

- and since FC is marked, AB becomes marked as well.

	A	B	C	D	E	F	G	H
A	$\approx$	–	–	–	–	–	–	–
B	X_1	$\approx$	–	–	–	–	–	–
C	X_0	X_0	$\approx$	–	–	–	–	–
D	–	–	–	–	–	–	–	–
E			X_0	–	$\approx$	–	–	–
F			X_0	–		$\approx$	–	–
G			X_0	–			$\approx$	–
H			X_0	–				$\approx$

Minimization of DFA-The Quotient Construction

- In a similar fashion, we examine the remaining unassigned pairs:

$$BH \xleftarrow{0} AE \quad \xrightarrow{1} \quad FF.Unabletomark.$$
$$BC \xleftarrow{0} AF \quad \xrightarrow{1} \quad FG.Mark, since\ BC\ ismarked.$$
$$BG \xleftarrow{0} AG \quad \xrightarrow{1} \quad FE.Unabletomark.$$
$$BG \xleftarrow{0} AH \quad \xrightarrow{1} \quad FC.Mark, since\ FC\ ismarked.$$
$$GH \xleftarrow{0} BE \quad \xrightarrow{1} \quad CF.Mark, since\ CF\ ismarked.$$
$$GC \xleftarrow{0} BF \quad \xrightarrow{1} \quad CG.Mark, since\ CG\ ismarked.$$
$$GG \xleftarrow{0} BG \quad \xrightarrow{1} \quad CE.Mark, since\ CE\ ismarked.$$
$$GG \xleftarrow{0} BH \quad \xrightarrow{1} \quad CC.Unabletomark.$$
$$HC \xleftarrow{0} EF \quad \xrightarrow{1} \quad FG.Mark, since\ CH\ ismarked.$$
$$HG \xleftarrow{0} EG \quad \xrightarrow{1} \quad FE.Unabletomark.$$
$$HG \xleftarrow{0} EH \quad \xrightarrow{1} \quad FC.Mark, since\ CF\ ismarked.$$
$$CG \xleftarrow{0} FG \quad \xrightarrow{1} \quad GE.Mark, since\ CG\ ismarked.$$
$$CG \xleftarrow{0} FH \quad \xrightarrow{1} \quad GC.Mark, since\ CG\ ismarked.$$
$$GG \xleftarrow{0} GH \quad \xrightarrow{1} \quad EC.Mark, since\ EC\ ismarked.$$

Solution cont..

- The resulting table is

	A	B	C	D	E	F	G	H
A	$\approx$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
B	X_1	$\approx$	$-$	$-$	$-$	$-$	$-$	$-$
C	X_0	X_0	$\approx$	$-$	$-$	$-$	$-$	$-$
D	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
E		X_1	X_0	$-$	$\approx$	$-$	$-$	$-$
F	X_1	X_1	X_0	$-$	X_1	$\approx$	$-$	$-$
G		X_1	X_0	$-$		X_1	$\approx$	$-$
H	X_1		X_0	$-$	X_1	X_1	X_1	$\approx$

Next round. The following pairs need to be considered:

$$\{AE, AG, BH, EG\}$$

Solution cont..

- The previously calculated transitions can be re-used;
- all that will have changed is whether the 'transitioned-to' states have been subsequently marked with an X1:
  - AE: unable to mark
  - AG: mark because BG is now marked.
  - BH: unable to mark
  - EG: mark because HG is now marked

Minimization of DFA-The Quotient Construction

Solution cont..

- The resulting table is

	A	B	C	D	E	F	G	H
A	$\approx$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
B	X_1	$\approx$	$-$	$-$	$-$	$-$	$-$	$-$
C	X_0	X_0	$\approx$	$-$	$-$	$-$	$-$	$-$
D	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
E		X_1	X_0	$-$	$\approx$	$-$	$-$	$-$
F	X_1	X_1	X_0	$-$	X_1	$\approx$	$-$	$-$
G	X_2	X_1	X_0	$-$	X_2	X_1	$\approx$	$-$
H	X_1		X_0	$-$	X_1	X_1	X_1	$\approx$

- Next round. The following pairs remain: $\{AE, BH\}$.
- However, neither makes a transition to a marked pair, so the round adds no new markings to the table.
- We are therefore done. The quotiented state set is

$$\{\{A, E\}, \{B, H\}, \{F\}, \{C\}, \{G\}\}$$

Solution cont..

- In other words, we have been able to merge states A and E, and B and H. The final automaton is given by the following diagram.