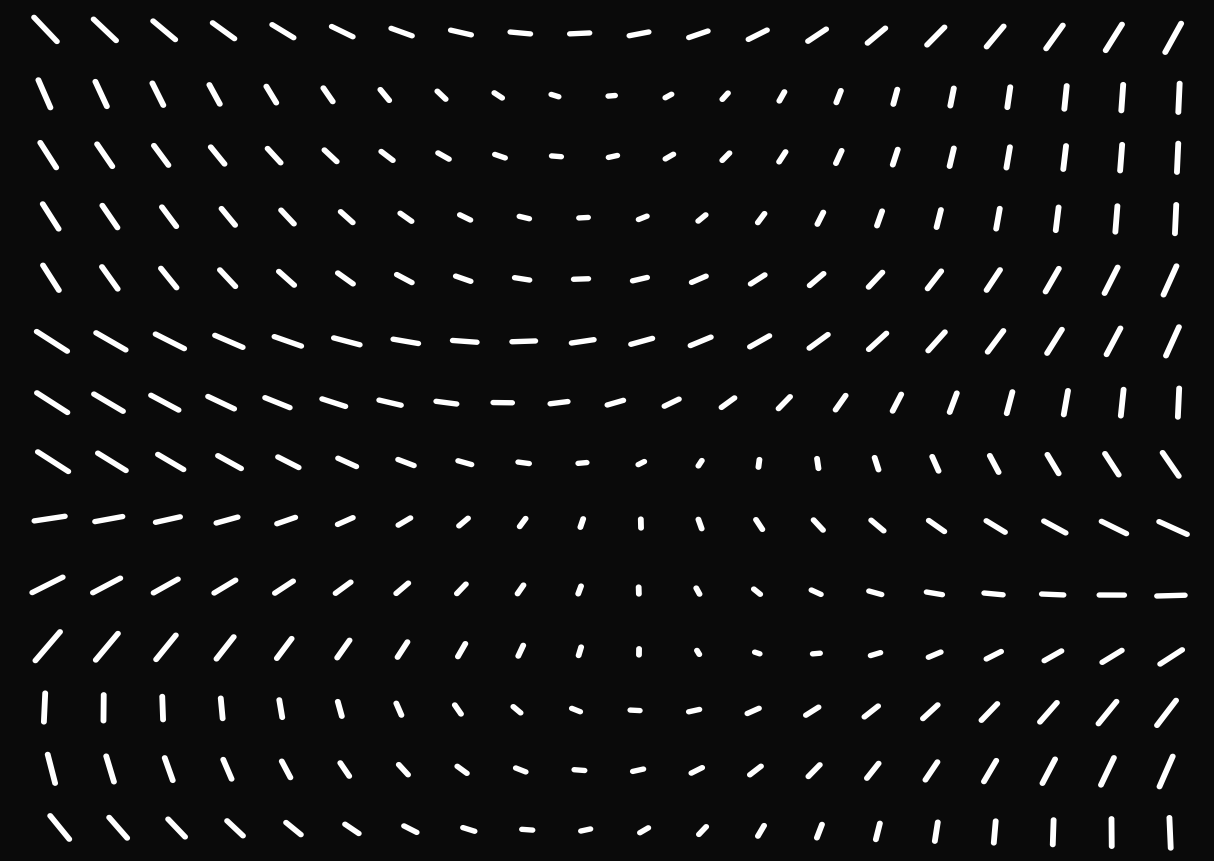


Tue Jan 4

Two Major Fund. Theoretical Results in 20th Cent.

1. Claude Shannon - communication theory
2. Alan Turing - computational theory

Main Theme
1. Languages
2. Grammar
3. Automata

- parsing techniques
- equivalences
- computational complexity

Ch. 1 Finite Automata + Regular Exp.
- neural net works, switching theory
- lexical analysis, txt editors, etc

Ch 2 Context Free Grammars
-
- automatic parse generators

Ch 3 Turing Machines + Computability
- most powerful computer
- undecidability
- complexity theory

Language

(Sigma) Σ = set of symbols alphabet

(omega) ω = string of alphabet

EX:
$\Sigma = \{0, 1\}$
$\omega = 101100$

Concatanation of Strings

$\omega_1 = a_1 a_2 \dots a_n$
$\omega_2 = b_1 b_2 \dots b_n$

$\omega_1 \omega_2 = a_1 a_2 \dots a_n b_1 b_2 \dots b_n$

EX:
$\omega_1 = 1011$
$\omega_2 = 0011$

$\omega_1 \omega_2 = 1011 0011$

Reverse of a String

ω $\quad$ ω^R

$\omega = a_1 a_2 \dots a_n \quad \omega^R = a_n a_{n-1} \dots a_2 a_1$

EX:
$\omega = 01101$
$\omega^R = 10110$

Length of a string

$|w|$ = # of alphabets in that string

EX: $w = 10110$
$|w| = 5$

Star

Σ^* - set of all strings that can be obtained from Σ

EX: $\Sigma = \{0, 1\}$

$\Sigma^* = \{\lambda, 0, 1, 00, 11, 01, 10, 000, 001, 010, 011, 100, 110, 111 \ldots\}$

(Σ^*: infinite; empty ✓)

Language

$L \subseteq \Sigma^*$

EX: $\Sigma = \{0, 1\}$

1. $L_1 = \{010, 101\}$] regular expression
2. $L_2 = \{0^m 1^n \mid m, n \geq 1\}$]
3. $L_3 = \{0^n 1^n \mid n \geq 1\}$] not regular – CFL (context-free language)
4. $\Sigma = \{0, 1, 2\}$] not regular – not CFL
   $L = \{0^n 1^n 2^n \mid n \geq 1\}$]

↳ Turing machine accepts this language
'recursively enmuvable language'

$RL \subset CFL \subset REL$

Concatanation of Languages

L_1 & L_2 are two languages

$L_1 L_2$

EX:
$\Sigma = \{a, b, c\}$
$L_1 = \{ab, abc\}$
$L_2 = \{bc, aabc\}$
$L_1 L_2 = \{abbc, abaabc, abcbc, abcaabc\}$
$|L_1| = m \quad |L_2| = n \quad |L_1 L_2| = mn$

Star closure

L = language

$L^* = \{L^0 \cup L^1 \cup L^2 \cup L^3 \ldots\}$

$L^0 = \{\lambda\} \qquad L^n = L^{n-1} L$
$L^1 = L$
$L^2 = LL$
$L^3 = L^2 L$

EX: $\Sigma = \{0, 1\}$ $L = \{0, 1\}$
$L^0 = \{\lambda\}$
$L^1 = \{0, 1\}$
$L^2 = LL = \{00, 01, 10, 11\}\ \{0, 1\}$
$L^3 = L^2 L = \{000, 001 \ldots 111\}\ \{0, 1\}$

ex: $\Sigma = \{0, 1\}$

$L = \{0\}$

$L^0 = \{\lambda\}$

$L^1 = \{0\}$

$L^2 = LL = \{0, 0\}$

$L^3 = L^2 L = \{000\}$

$L^n = \{\lambda, 0^n \mid n \geq 1\}$

Thu Jan 6, 2022

Proof Techniques

1. Direct Proof
2. Proof by Induction
3. Proof by Contradiction

Ask all q. in advance

$$\left[\begin{array}{cccc|ccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{array}\right]$$

1 0 1 0 (check bit)
1 1 1 0

find an error w/ check bit

Current 0, 1

Quantum qubits

$|q\rangle = \alpha|0\rangle + \beta|1\rangle$

read prob(0) = α^2
prob(1) = β^2
$\alpha^2 + \beta^2 = 1$

$N = pq$ $7 \times 5 = 35$ vs find factors of 35

Direct Proof

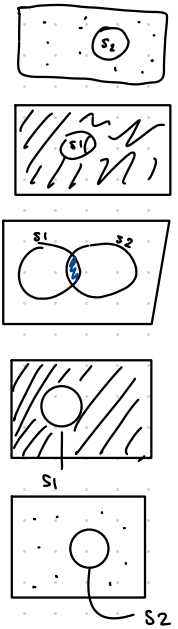
DeMorgan's Theorem
S_1 and S_2 are two sets
$\overline{S_1 \cap S_2} = \overline{S_1} \cup \overline{S_2}$

Normal
If we want to prove $A = B \Rightarrow A \subseteq B \Rightarrow A = B$
$B \subseteq A$

$\overline{S_1 \cap S_2} \subseteq \overline{S_1} \cup \overline{S_2}$ — ①
$\overline{S_1} \cup \overline{S_2} \subseteq \overline{S_1 \cap S_2}$ — ②

$x \in \overline{S_1 \cap S_2}$
$\Rightarrow x \notin S_1 \cap S_2$
$\Rightarrow x \in \overline{S_1} \cup \overline{S_2}$ — I
$\Rightarrow \overline{S_1 \cap S_2} \subseteq \overline{S_1} \cup \overline{S_2}$

② $y \in \overline{S_1} \cup \overline{S_2}$
$\Rightarrow y \in \overline{S_1}$ or $y \in \overline{S_2}$
$\Rightarrow y \notin S_1$ or $y \notin S_2$
$\Rightarrow y \notin S_1 \cap S_2$
$\Rightarrow y \in \overline{S_1 \cap S_2}$ — II



Proof by Induction

1. Base case $n = 1$ as prove
2. Assume it is true up to $n-1$
3. Prove for n

Example: $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

Base Case: $n = 1$

L.H.S $= \sum_{i=1}^{1} i^2 = 1^2 = 1$

R.H.S $= \frac{1 \cdot 2 \cdot 3}{6} = 1$

LHS = RHS

Induction Step: Assume true for up to $n-1$, prove n

$$LHS = \sum_{i=1}^{n} i^2 = 1^2 + 2^2 + 3^2 + \dots (n-1)^2 + n^2$$

$$= \frac{(n-1)n(2(n-1)+1)}{6} + n^2 \quad \hookrightarrow \text{by induction}$$

$$= \frac{(n-1)n(2n-1)}{6} + n^2 = \frac{n}{6}\left[2n^2 - 3n + 1 + 6n\right]$$

$$= \frac{n}{6}\left[2n^2 + 3n + 1\right]$$

$$= \frac{n}{6}\left(2n^2 + 2n + n + 1\right)$$

$$= \frac{n}{6}\left(2n + n + 2n + 1\right)$$

$$= \frac{n}{6}\left(n(2n+1) + 2n + 1\right)$$

$$= \frac{n}{6}(2n+1)(n+1)$$

$$= \frac{n(n+1)(2n+1)}{6}$$

Proof By Contradiction

rational numbers

$\frac{m}{n}$ n, m are rational

$\frac{2}{8}, \frac{5}{4}, \frac{9}{8} \ldots$

not rational

$\frac{1}{\pi}, e, \frac{1}{e}$

Example: prove $\sqrt{8}$ is not a rational number

Suppose $\sqrt{8}$ is rational number

$\sqrt{8} = \frac{m}{n}$ $\quad \gcd(m, n) = 1$

$n\sqrt{8} = m$

$(n\sqrt{8})^2 = m^2$

$8n^2 = m^2$ $\quad$ LHS = even

$8n^2 = (2k)^2$ $\quad \Rightarrow$ RHS = even

$8n^2 = 4k^2$ $\quad$ m is even

$2n^2 = k^2$ $\quad m = 2k$

$2n^2 = (2k_1)^2 = 4k_1^2$

$\Rightarrow n^2 = 2k_1^2$ $\quad$ LHS = even

RHS = even

k is even

$k = 2k_1$

n = even $-$ I

both m and n is even $\quad$ RHS = even

Assumed $\gcd(m, n) = 1$ $\quad \Rightarrow$ LHS = even

$\therefore$ contradiction!

Example 2: # of prime numbers is infinite

2, 3, 5, 7, 11, 13, 17, 19 . . .

Assume the # of primes is finite

$P_1, P_2 \ldots P_N$ $\rightarrow$ finite primes

$M = P_1 P_2 \ldots P_N + 1$

$M \pmod{P_i} = 1 \mod P_i \quad (= 1, 2, \ldots N)$

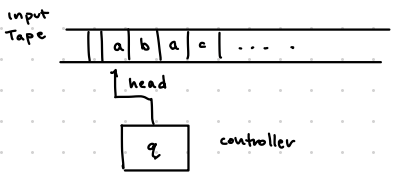
M is not divisible by P_i's

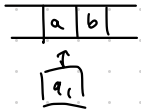
M is a prime $\rightarrow$ contradiction

$\therefore$ # of primes is <u>not</u> finite but infinit

Deterministic Finite Automata

Tue Jan 11 2022





Read the input from the tape, changes the state

$M = (Q, \Sigma, \delta, F, q_0)$

Formal Definition

Q - set of states
Σ - input alphabet
$\delta : Q \times \Sigma \rightarrow Q$
$F \subseteq Q$ → set of final states
$q_0 \in Q$ → initial state

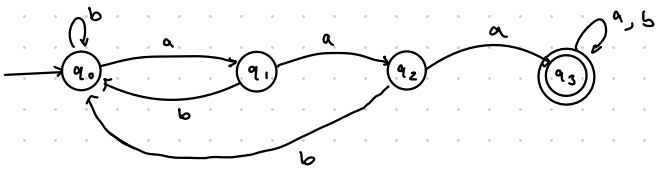
How do you design a DFA which accepts all strings if there exists 3 consecutive a's

$\Sigma = \{a, b\}$

w_1 = ab aaa baabb ✓
w_2 = ab aa bbb aaba ✗

Transition graph

q_0 → initial state
q_1 — first a is read in consecutive 3 a's
q_2 — second a is read "
q_3 — third a



Transition table

	a	b
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_3	q_0
q_3	q_3	q_3

Extended Transition Function

$$\delta^*(q, \lambda) = q \qquad \lambda \text{ - null symbol}$$

$$\delta^*(q, wa) = \delta(\delta^*(q, w), a)$$

Example:

$\delta^*(q_0, aab)$

$= \delta(\delta^*(q_0, aa)\, b)$

$= \delta(q_2, b) \quad = q_0$

Language Accepted by DFA

$L(M) = \{ w \mid \delta^*(q_0, w) \in F \}$

Example:

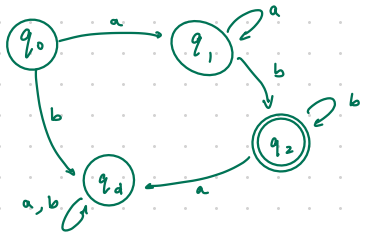
$a\,b\,a\,a\,a\,b\,b = w$

$\delta^*(q_0, w) = q_3$

Example:

$L(m) = \{ a^n b^m \mid n, m \geq 1 \}$

$\Sigma = \{ a, b \}$



Mod function

N mod M

divide N by M, take the remainder

$0 \leq N \leq m$

$(a \cdot b) \bmod m = ((a \bmod m)(b \bmod m)) \bmod m$

Ex:

$(9 \cdot 8) \bmod 5 = 2$

$= ((9 \bmod 5)(8 \bmod 5)) \bmod 5$

$= (4 \cdot 3) \bmod 5 = 2$

$72 \bmod 5 = 2 \bmod 5$

$3^6 \bmod 7 =$

$3^1 \bmod 7 = 3 \qquad 3^2 \bmod 7 = 2$

$3^3 \bmod 7 = (3 \cdot 3^2) \bmod 7 = (3 \cdot 2) \bmod 7$

$= 6$

$3^4 \bmod 7 = (3 \cdot 6) \bmod 7 = 4$

$35 \bmod 7 = (3 \cdot 4) \bmod 7 = 5$

$36 \bmod 7 = (3 \cdot 5) \bmod 7 = 1$

Ex:

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

$w = a_n a_{n-1} \dots a_0$

Accept w if the binary values of this string is 0 mod 5

1 0 1 1 0

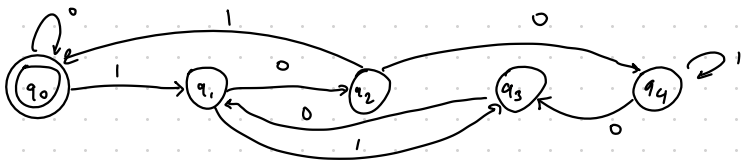
$2^4 \; 2^3 \; 2^2 \; 2^1 \; 2^0$

$2^4 + 2^2 + 2^1 = 16 + 4 + 2 = 22 \bmod 5 = 2$ ✗

There are 5 states

$q_i \quad (i = 0, 1, 2, 3, 4)$

$2 \cdot 2 + 1$



$a_{n-1} \quad a_{n-2} \quad \dots \quad a_0$

$a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} \dots a_0$

$(((a_{n-1} \cdot 2 + a_{n-2}) 2 + a_{n-3}) 2 + a_{n-4}) 2 + \dots + a_0$

10
$(2 \cdot 1 + 0)$
$= 2$
q_2

11
$(1 \cdot 2 + 1)$
$= 3$
q_3

100
$= 4$
q_4

101
$= 5$
q_5

④

$4\ 3\ 4\ 5$

$= 4 \cdot 10^3 + 3 \cdot 10^2 + 4 \cdot 10 + 5$

$\bmod 3$

$10 \bmod 3 = 1$

$10^2 \bmod 3 = 1$

$10^3 \bmod 3 = 1$

$(4 + 3 + 4 + 5) \bmod 3 = 1$

⑤ $\Sigma = \{0, 1\}$

Example 4:

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

Accept the strings which have 3rd bit from the last one is '0'

$1\ 0\ 1\ 1\ \underset{\uparrow}{0}\ 1\ 1$ → accept

$1\ 1\ 1\ 0\ 1\ 0\ 0$ → don't accept

---

possible last 3-bits

$0\ 0\ 0$ → $[3\ 2\ 1]$

$0\ 0\ 1$ → $[3\ 2]$

$0\ 1\ 0$ → $[3\ 1]$

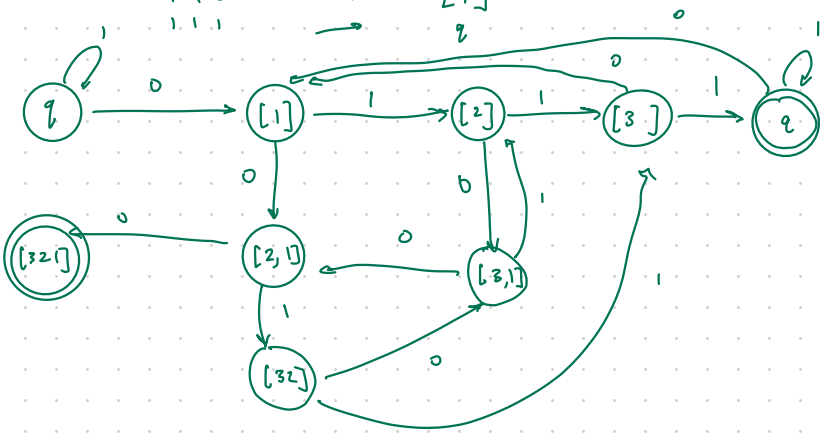
$0\ 1\ 1$ → $[3]$

$1\ 0\ 0$ → $[2\ 1]$

$1\ 0\ 1$ → $[2]$

$1\ 1\ 0$ → $[1]$

$1\ 1\ 1$ → q



Thu Jan 13 2022

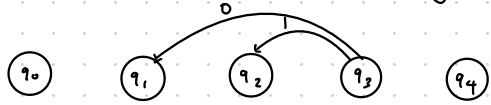
mod 5

$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

1 1 1 0 1 0

$1\cdot 2^5 + 1\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^1 = 58 \mod 5$

$3 \mod 5$



since 2^0

1 1 1 0 1 0 1 2^0 $(2\cdot 58) \mod 5$

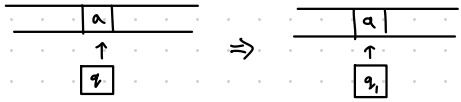
$(2\cdot 3) \mod 5 = 1 + 1$

1 1 1 0 1 0 0

Non-Deterministic Finite Automata (NFA)

3 main differences from DFA

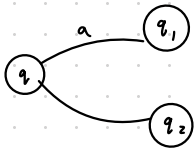
1) Allows state change without reading an input



without reading lambda, can change states

$\delta(q, \lambda) = q_1$

2) After reading an input the automata can take one among a set of states.



3) Some state transitions are not defined (go to ϕ state)

trap

Formal Definition

$M = (Q, \Sigma, \delta, q_0, F)$

Q - set of states

Σ = set of input alphabets

$\delta : Q \times \{\Sigma \cup \lambda\} \rightarrow 2^Q$

(subset of states)

q_0 - initial state

F - set of final state

Example:

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

Why DFA? two options for zero

$\delta(q_0, 0) = \{q_0, q_1\}$

$\delta(q_3, 0) = \phi$

$\delta^*(q, w)$ = set of all states reachable from q after reading w

[$w \rightarrow$ string of inputs]

Example :

$$\begin{aligned}
\delta^*(q_0, 001) &= \delta(\delta^*(q_0, 00), 1) \\
&= \delta(\delta^*(\delta^*(q_0, 0), 0), 1) \\
&= \delta(\delta^*\{q_0, q_1\}, 0), 1) \\
&= \delta((q_0, q_1, q_2), 1) \\
&= \delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1) \\
&= \underline{\{q_0, q_2, q_3\}}
\end{aligned}$$

Language Accepted by NFA

$L(M) = \{w \mid \delta^*(q_0, w) \cap F \neq \phi\}$

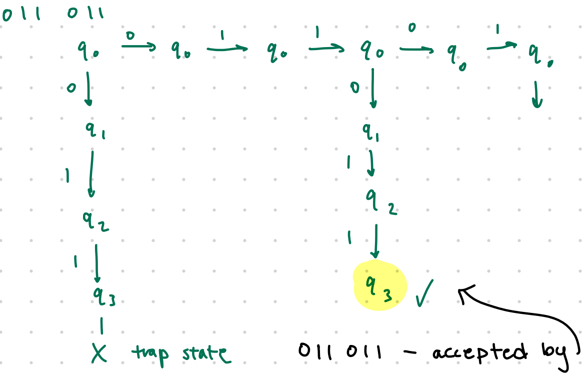
Example 1 :

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

Accept the strings if the 3rd last bit is a '0'

011 011 ✓ Accept

011 111 → ×

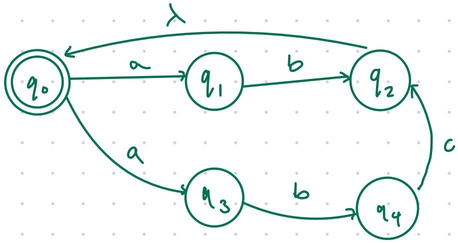


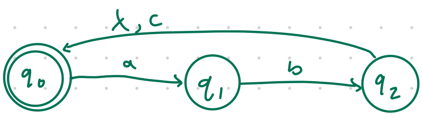
Example 2:

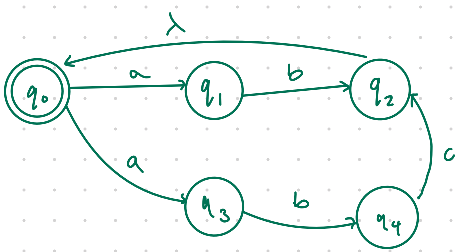
$\Sigma = \{a, b, c\}$

$L = \{ab, abc\}^*$

$L = \{\lambda, ab, abc, abab, ababc, abcab, abcabc \ldots\}$







ab
abc

$\delta^*(q_0, ab)$

$\delta^*(q_0, a) = \{q_1, q_3\}$

$\delta^*(q_0, ab) = \delta(\delta^*(q_0, a), b)$

$= \delta^*(\{q_1, q_3\}, b)$

$= \delta(q_1, b) \cup \delta(q_2, b)$

$= \delta(q_1, b) \cup \delta(q_3, b)$

$= \{q_0, q_2, q_4\}$

$$\delta^*(q_0, abc)$$
$$= \delta^*(\delta^*(q_0, ab), c)$$
$$= \delta^*(\{q_0, q_2, q_4\}, c)$$
$$= \delta(q_0, c) \cup \delta(q_2, c) \cup \delta(q_4, c)$$
$$= \phi \cup \phi \cup \{q_2, q_0\}$$

Example:

$\Sigma = \{0, 1, 2\}$

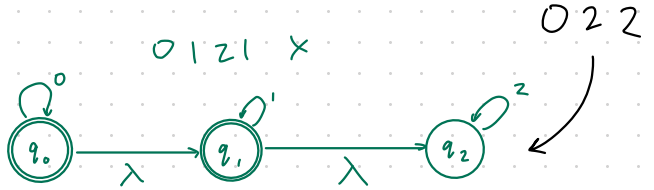
$L = \{0^m 1^n 2^p \mid m, n, p \geq 0\}$

0 ✓ λ ✓

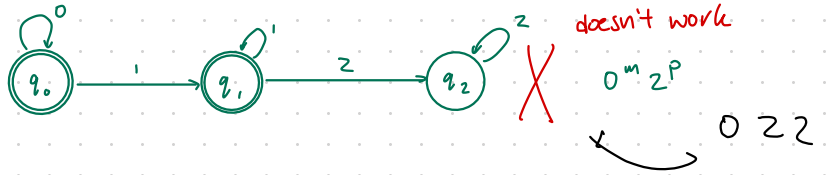
11 ✗

022 ✓

0121 ✗



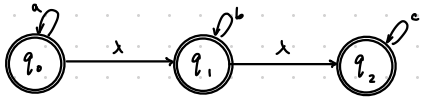
λ = w/o reading input, you can move states



$$L(NFA) = L(DFA)$$

Thu Jan 18 2022

$L = \{ a^n b^m c^p \mid n, m, p \geq 0 \}$



$\Sigma = \{a, b, c\}$

$\delta(q_0, a) = q_0$ $\quad \delta(q_1, b) = q_1$

$\delta(q_0, \lambda) = q_1$ $\quad \delta(q_1, \lambda) = q_2$

$\delta(q_2, c) = q_2$

$\delta^*(q_0, a)$ – set of states reachable from q_0 after reaching a

$\delta^*(q_0, a) = \{q_0, q_1, q_2\}$

$\delta^*(q_1, b) = \{q_1, q_2\}$

$\delta^*(q_2, c) = \{q_2\}$

$\delta^*(q_1, a) = \phi$ null → trap state

NFA ≡ DFA

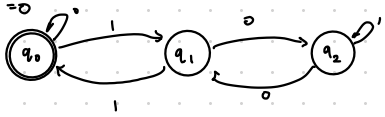
Languages accepted by both NFA and DFA are the same

$L \in DFA \Rightarrow L \in NFA$

↓ transition graph

Use the same transition graph for NFA

Binary mod 3 = 0



DFA

NFA

If a language is accepted by DFA, then it is accepted by NFA

$L \in NFA \Rightarrow L \in DFA$

NFA

$M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$

will construct a DFA

$M_D = (Q_D, \Sigma, \delta_D, [q_0], F_D)$

such that if $L \in NFA$ then $L \in DFA$

$L \in LCM_D$

Assume

$$Q_N = \{ q_0, q_1, \cdots q_{n-1} \}$$

In DFA, under construction the states are of type $[q_{i_1}, q_{i_2}, \cdots q_{i_t}]$ where $\{ q_{i_1}, q_{i_2} \cdots q_{i_t} \}$ is a subset of $\{ q_0, q_1, \cdots q_{n-1} \}$

Example: $Q_N = \{ q_0, q_1, q_2 \}$ then

$$Q_D = \{ [q_0], [q_1], [q_2], [q_0, q_1], [q_0, q_2], [q_1, q_2], [q_0, q_1, q_2] \ [\text{Trap}] \}$$

How to find the transition graph for the DFA?

① Initial state $[q_0]$

② For any $[q_{i_1}, q_{i_2} \cdots q_{i_t}]$ is reachable from $[q_0]$ so far, if some outgoing edge is missing, say for input a, find

$$\delta_N^* [\{ q_{i_1}, q_{i_2} \cdots q_{i_t} \}, a]$$

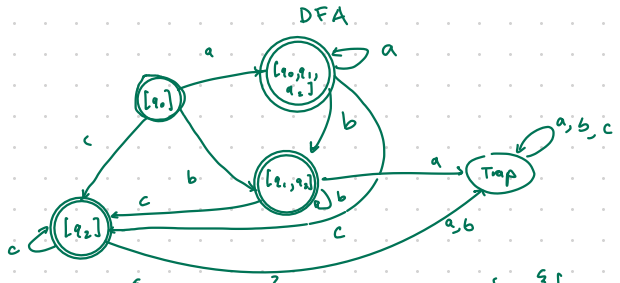
$$= \{ a_j, a_{j_2} \cdots a_{j_s} \}$$

$$\delta_D [[q_{i_1}, q_{i_2}, \cdots q_{i_t}], a] = [q_{j_1}, q_{j_2} \cdots q_{j_r}]$$

③ Make $[q_{i_1}, q_{i_2} \cdots q_{i_t}]$ as one of the final states if any of q_{i_n} is a final state

④ If NFA accepts λ then make $[q_0]$ as one of the formal states

Example:



$\delta_N^* (q_0, a) = \{ q_0, q_1, q_2 \}$
$\delta_N^* (q_0, b) = \{ q_1, q_2 \}$
$\delta_N^* (q_0, c) = \{ q_2 \}$

Find $\delta_D \{ [q_0, q_1, q_2] \ a \}$

$\delta_N^* \{ \{ q_0, q_1, q_2 \}, a \}$

$= \delta_N^* \{ q_0, a \} \cup \delta_N^* \{ q_1, a \} \cup \delta_N^* \{ q_2, a \}$

$= \{ q_0, q_1, q_2 \} \cup \phi \cup \phi$

$= \{ q_0, q_1, q_2 \}$

$\delta_D([q_0, q_1, q_2], b) =$

First find out

$\delta_N^*(\{q_0, q_1, q_2\}, b)$

$= \delta^*(q_0, b) \cup \delta^*(q_1, b) \cup \delta^*(q_2, b)$

$= \{q_1, q_2\} \cup \{q_1, q_2\} \cup \phi$

$= \{q_1, q_2\}$

$\delta_D([q_0, q_1, q_2], c)$

Find

$\delta_N^*(\{q_0, q_1, q_2\}, c)$

$= \delta^*(q_0, c) \cup \delta^*(q_1, c) \cup \delta^*(q_2, c)$

$= \{q_2\} \cup \{q_2\} \cup \{q_2\}$

$= \{q_2\}$

$[q_0, q_1]$ X $\quad$ $[q_0, q_2]$ X

---

we can prove

if $\delta^*(q_0, w) = \{q_{i_1}, q_{i_2} \cdots q_{i_t}\}$

then $\delta_D([q_0], w) = [q_{i_1}, q_{i_2}, \cdots q_{i_t}]$

prove by induction on the length of w

DFA = NFA

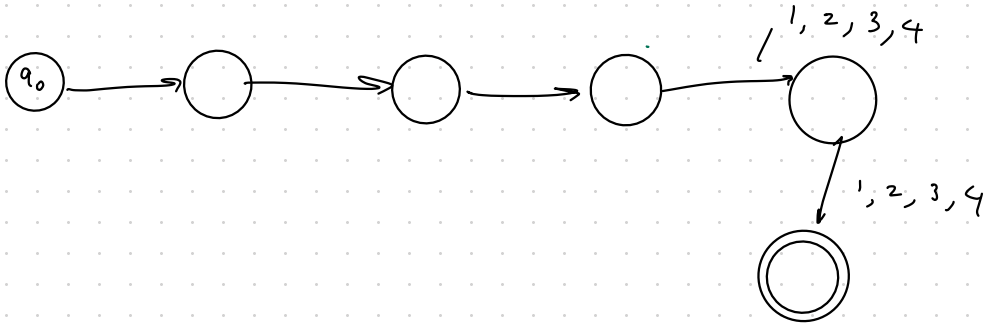
HW

$\Sigma = \{a, b, c, d, 1, 2, 3, 4\}$

length 5, 6, 7

Final $\{a, b, c, d\}$

Last 2 $\{1, 2, 3, 4\}$



2. $L = \{abab^n \mid n \geq 0\} \cup \{aba^n : n \geq n\}$

5. L - Regular

L^R - Regular

$w = abbaa$ $\quad w^R = aabba$

NFA → DFA and Backwards reverse is accepted

induction?

Regular Expression

① Φ, λ, a → Regular expressions

Φ → Language $\{\ \}$

λ → $\{\lambda\}$

a → $\{a\}$

} primitive

② If r_1 and r_2 are regular expressions then

$r_1 + r_2$ — $\{R_1\} \cup \{R_2\}$

$r_1 \cdot r_2$ — $\{R_1 R_2\}$

r_1^* — R_1^*

(r_1) — R_1

Language

$r_1 \longleftrightarrow R_1$

$r_2 \longrightarrow R_2$

precedence

1. *
2. .
3. +

$0 + 01^*$

$= 0 + (0 \cdot (1^*))$

Examples

$L = \{a^n b^m c^p \mid m, n, p \geq 0\}$

$RE = a^* b^* c^*$

② $\Sigma = \{a\}$ $\quad L = \{a^n \mid n$ is even $\}$

$RE \Rightarrow (aa)^*$

③ $L = \{a^m b^n \mid m+n$ is even $\}$

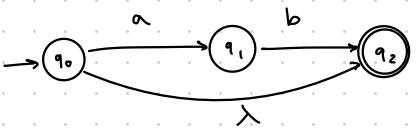
Both m and n are even

$RE \Rightarrow (aa)^* (bb)^*$

$RE = (aa)^* (bb)^* + a(aa)^* b(bb)^*$

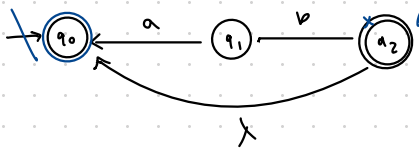
Both m and n are odd

$RE \Rightarrow a(aa)^* b(bb)^*$



1. flip transitions

2. new initial state



3. the original initial state is the final state

Regular Expression

RE	Language
ϕ	$\{\}$
λ	$\{\lambda\}$
a	$\{a\}$

If r_1 r_2 are RE then

$r_1 + r_2$	$R_1 \cup R_2$
$r_1 \cdot r_2$	$R_1 R_2$
r_1^*	R_1^*
(r_1)	R_1

r_1 accepts the language R_1

r_2 accepts the language R_2

Examples

1) $L = \{a^n b^m \mid n \geq 3, m < 4\}$

$aaa a^* (b + bb + bbb + \lambda)$

2) $L = \{a^n b^m \mid n < 4, m \leq 4\}$

$(\lambda + a + aa + aaa)(\lambda + b + bb + bbbb)$

3) $L = \{a^n b^m \mid n \geq 2, m \geq 1, nm \geq 3\}$

$aa a^* bb b^* + aaaa^* bb^*$

$aa(b^* bbb^* + aa^* bb^*)$

DFA ↪ NFA ⟺ R.E.

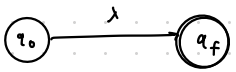
RE → There is an equivalent NFA that accepts the same language

RE

NFA

ϕ

λ



a

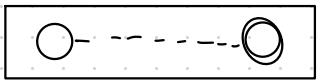
r_1

$r_1 + r_2$

$r_1 r_2$

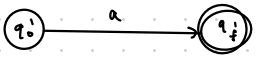
r_1^*

r_1



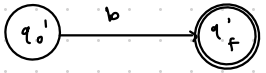
r_2

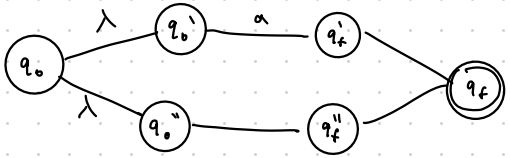
$r_1 + r_2$

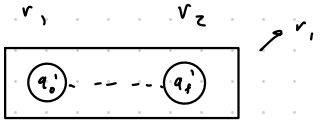


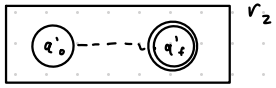
$r_1 = a$

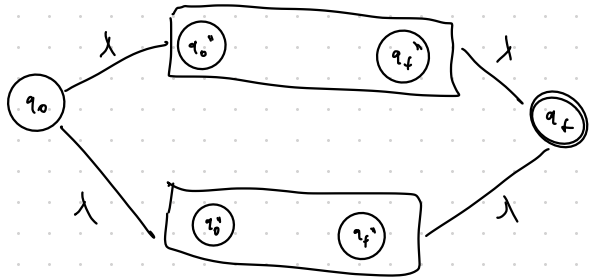
$r_1 = b$





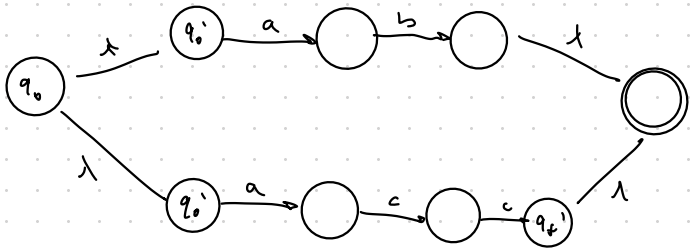

r1
r2
r1
q0'
qf'

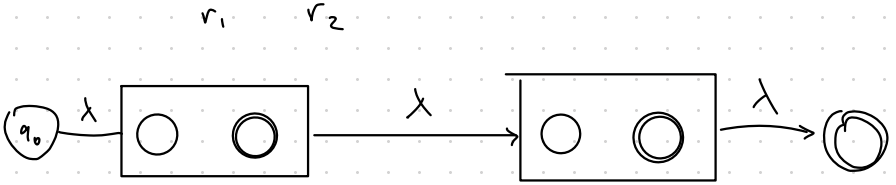

r2
q0'
qf'


λ
λ
λ
λ
q0
q0'
qf'
q0'
qf'
qf

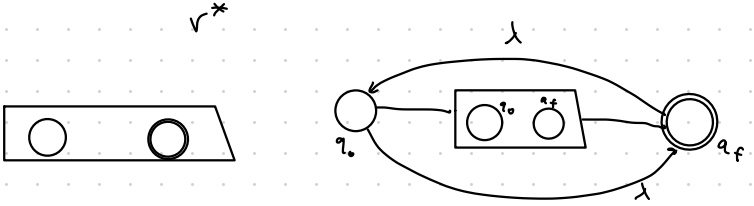
r1 r2

ab + acc


λ
a
b
λ
λ
a
c
c
λ
q0
q0'
q0'
qf'

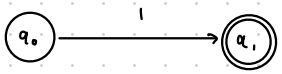

r1
r2
λ
λ
λ
q0

r*


λ
q0
qf
q0
qf
λ

Example:

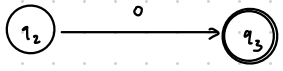
$\underbrace{1}_{r_1} + \underbrace{01^*}_{r_2}$ — RE



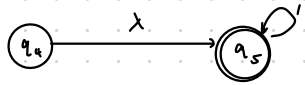
$RE = 01^* = r_{21}\, r_{22}$

$r_{21} = 0$

$r_{22} = 1^*$

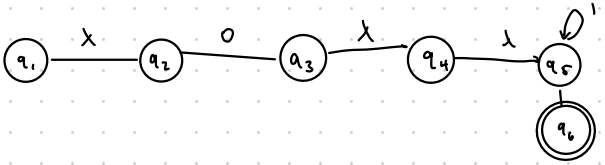


r_{21}

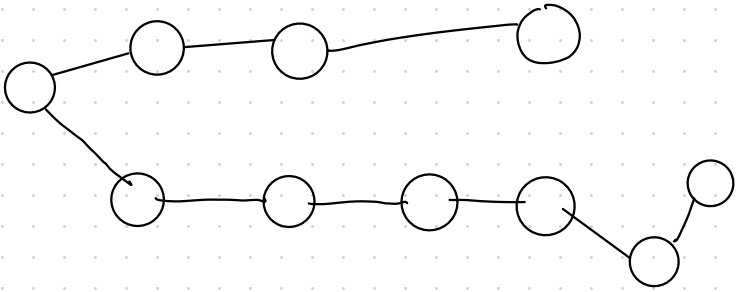


r_{22}

$r_2 = r_{21}\, r_{22}$

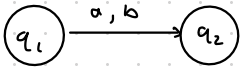


$r = r_1 + r_2$

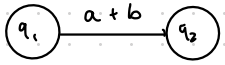


NFA ⇒ RE

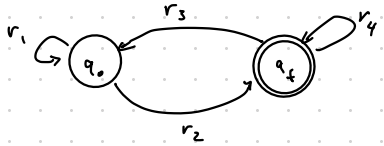
Generate Transition Graph



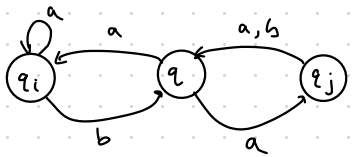
⇒



Given NFA, find one initial state and one final state using the generalized transition graph



$$RE = r_1^* r_2 (r_4^* + r_3 r_1^* r_2)^*$$



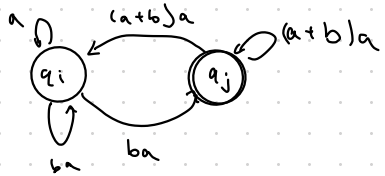
Eliminate q

1) q_i to q_i using q

2) q_i to q_j using q

3) q_j to q_i using q

4) q_j to q_j using q



Midterm 1 this Thursday

- proof techniques < induction, direct, contradiction
- DFA
- NFA
- convert from NFA → DFA

Formally: a grammar is defined by a tuple (V, T, S, P)

V is a set of non terminal variables
T is a set of terminal symbols making up strings
S is the starting variable where $S \in V$
P is a set of production rules mapping strings of terminals and variables to other strings of terminals and variables

Simple EX:

$V = \{S, A, B\}$
$T = \{a, b\}$

Example Grammar

P
$S \to aA$
$A \to aA$
$A \to bB$
$B \to bB$
$B \to \lambda$

Example Gen 1

S
1) $\Rightarrow aA$
2) $\Rightarrow aaA$
3) $\Rightarrow aabB$
4) $\Rightarrow aabbB$
4) $\Rightarrow aabbbB$
5) $\Rightarrow aabbb$

S
aA
aaA
aaaA
aaabB
aaab

1. shortest string

$S \to aA \to abB \to ab$

2. infinite or finite?

3. what language

$a^n b^m \mid n, m > 1$

$A^* + B^*$

P:
$S \to aA$
$A \to aA \mid bB$
$B \to bB \mid acA \mid d$
$abB \to bBAa$

grammars that generate regular languages

right-linear

$A \to xB$ or $A \to x$

left-linear

$A \to Bx$ or $A \to x$

a regular grammar is either right-linear or left-linear

$A \to aAb$
neither (non terminal must be before or after)

$B \to Aabc$
left-linear

$A \to Ba \mid bB$
neither

$A \to cB \mid b$
right linear

$A \to aBC$
neither (only one nonterminal)

$aA \to aB$
neither

Practice: right linear
$(ab)^* ad$

$V = \{S\}$ $T = \{a, b\}$ P:

$S \to ad$
$S \to abS$

$S \to abS \mid ad$

right

P:
$S \to abS \mid ad$
$\to ababS$
$\to ababad$

left

P:
$S \to Aad$
$A \to Aab \mid \lambda$

$\to Aad$
$\to Aabad$
$\to ababad$

$V = \{ S \}$

$T = \{ 0, 1 \}$

P:

$S \to 0 \mid 0S$
$S \to 1 \mid 1S$

$S \to 0S \mid 1S \mid \lambda$

There is a unique grammar for every language.

$L = \{a^n b^m \mid n \geq 2,\ m \geq 3\}$

$V = \{ S \}$

$T = \{ a, b \}$

P:

$S \to aaA$
$A \to aA \mid B$
$B \to \lambda \mid bbb \mid B$

Reversing Languages by Converting Between Left and Right Grammars

invert symbols

P:

$S \to abA$
$A \to abA \mid B$
$B \to cdB \mid cd$

P:

$S \to Aba$
$A \to Aba \mid B$
$B \to Bdc \mid dc$

Make a new gramar with two

$V = V_1 \cup V_2$

Definition:

A language is regular if and only if there exists a deterministic finite acceptor (DFA) that accepts the language

Construct a NFA from a Right Linear Grammar

RLG

$G = \{V, T, S, P\}$

$V = \{S, V_0, V_1, \ldots, V_k\}$

$T = \{a_1, a_2 \ldots a_n\}$

NFA

$M_N = \{Q, \Sigma, \delta, q_0, F\}$

$Q = V \cup \{f\}$ $\quad \Sigma = T$ $\quad q_0 = S$ $\quad F = \{f\}$

if $V_i \rightarrow$

if

if $V_i \rightarrow \lambda$ is in P, then $\delta^*(V_i, V)$ is in

RLG

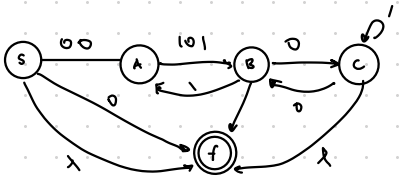
$V = \{S, A, B, C\}$

$T = \{0, 1\}$

P:

$S \rightarrow 00A \mid 0 \mid \lambda$
$A \rightarrow 101B$
$B \rightarrow 0C \mid 1A \mid 1$
$C \rightarrow 0B \mid 1C \mid \lambda$

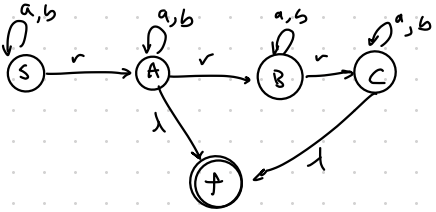
NFA



$V = \{S, A, B, C\}$

$T = \{a, b, r\}$

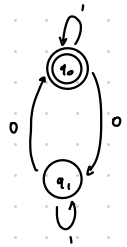
$S \rightarrow aS \mid bS \mid rA$
$A \rightarrow aA \mid bA \mid rB \mid \lambda$
$B \rightarrow aB \mid bB \mid rC$
$C \rightarrow aC \mid bC \mid \lambda$



every right-linear grammar represents a regular language!

now, vice versa proof.

Construct a Equivalent RLG from DFA



$V = \{ q_0, q_1 \}$

$T = \{ 0, 1 \}$

P :

$S = q_0$

$q_0 \rightarrow 0q_1 | 1q_0 | \lambda$

$q_1 \rightarrow 1q_1 | 0q_0$

We proved languages of Right Linear Grammars are regular

$$DFA \Leftrightarrow NFA \Leftrightarrow RE$$
$$\Downarrow RLG \Rightarrow$$

$RG \Leftrightarrow DFA \Leftrightarrow NFA \Leftrightarrow RE$

Are these regular as well?

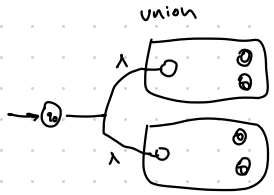
$L_1 \cup L_2$ $\quad L_1^*$

$L_1 \cap L_2$ $\quad L_1 L_2$

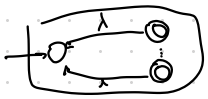
$\overline{L_1}$

Proof by construction

union



star closure



Thu Jan 27

Pumping Lemma for Regular Languages

Multiple Equivalent Representations of Regular Languages

$L = \{a^i b^i \mid i \geq 0\}$

Prove it is Regular
DFA, NFA, RE, or RG

Prove it is not regular . . .

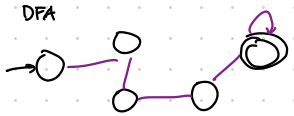
All finite languages are regular ✓
True!
Any DFA can be built

what is the pumping lemma for regular languages?

1) Find rules

2.) Test for violations

abaab



At least one state was visited more than once

n string length $p \leq n$ states

then there must be a cycle

Point 1: If a string is long enough relative to a DFA, there must be a cycle.

Point 2:

Point 3: If a language L is regular, there exists some smallest DFA that accept L

If L is a regular language, then there is p "pumping length"

- $|y| \geq 1$ The loop has at least one transition
- $|xy| \leq p$
- $xy^iz \in L$ for $i \geq 0$

Proof by Contradiction with Pumping Lemma

Step 1: Assume the language is regular and has some pumping length p.

Step 2: Select some string w in the language with length greater than p

Step 3: Show that no decomposition of w into xyz satisfies the conditions:
$|y| \geq 1$ $|xy| \leq p$ $xy^iz \in L$ for $i \geq 0$

Finding a contradiction to pumping lemma requires finding a decomposition

Ex 1. Prove $L = \{a^ib^i \mid i \geq 1\}$ is not regular

1. Assume L is regular with some pumping length p.
2. Let $w = a^pb^p$ which looks like $\underbrace{aaa\ldots a}_{p}\overbrace{bbb\ldots b}^{p}$ and has length $\geq p$
3. As $|xy| \leq p$, all valid decompositions of w can be described by:
$x = a^\alpha \quad \alpha \geq 0$
$y = a^\beta \quad \beta \geq 1$
$z = a^{p-\alpha-\beta}b^p \quad (\alpha+\beta \leq p)$

According to pumping lemma... xy^iz should be in L for any $i \geq 0$, lets try $i = 0$

$$xy^0z = a^\alpha a^{p-\alpha-\beta}b^p = a^{p-\beta}b^p \notin L$$

Contradiction! Our assumption in (1) is false and L is not regular.

EX 2.1 Prove $L = \{ a^i b^j \mid i \neq j \}$ is not regular

1) Assume L is regular with pumping length p.

2) Let $w = a^p b^{p+p!}$ $\underbrace{aaa\ldots a}_{p}\overbrace{bbb\ldots b}^{p+p!}$ and has length $2p+p!$

3) As $|xy| \leq p$, all valid decompositions of w can be described by:
$x = a^\alpha \quad \alpha \geq 0 \quad y = a^\beta \quad \beta \geq 1$, and $z = a^{p-\alpha-\beta} b^{p+p!} \quad (\alpha+\beta \leq p)$

According to Pumping lemma, $xy^i z$ should be in L for any $i \geq 0$, lets try $i = 1 + \frac{p!}{\beta}$

$$xy^{1+\frac{p!}{\beta}} z = a^\alpha a^{\beta(1+\frac{p!}{\beta})} a^{p-\alpha-\beta} b^{p+p!} = a^{p+p!} b^{p+p!} \notin L$$

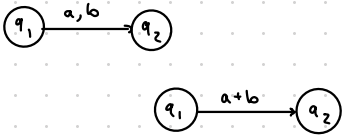
Contradiction! Therefore our assumption in (1) is false and L is not regular

Tue Feb 1

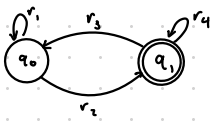
RE ⇒ NFA

NFA ⇒ RE

Generalised Transition Graph

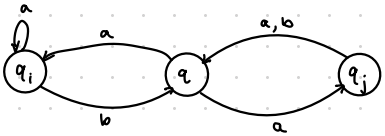


Main idea



$RE = r_1^* r_2 (r_4^* + r_3 r_1^* r_2)$

How to eliminate one internal node?



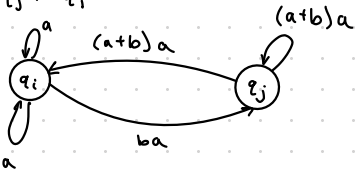
Find the walks from

1) q_i to q_i through q ✓

2) q_i to q_j "

3) q_j to q_j "

4) q_j to q_i "



1) q_0 to q_2 through q_1

2) q_0 to q_0 through q_1 ✗

3) q_2 to q_0 through q_1 ✗

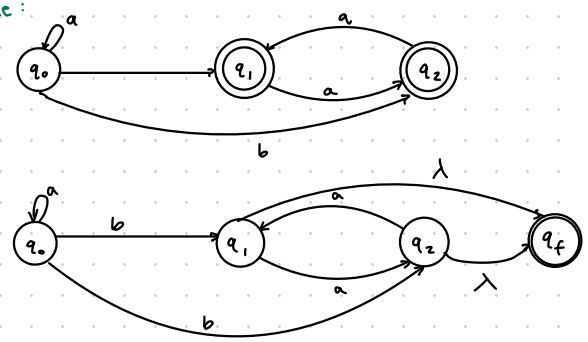
4) q_2 to q_2 through q_1 ✗

$a^*(ab^*b + \lambda)a^*$

$a^* a^* + a^*$

(ab^*b)

Example :



Eliminate q_1

1) Walk from q_0 to q_2 through q_1 ✓

2) Walk from q_0 to q_f " q_1 ✓

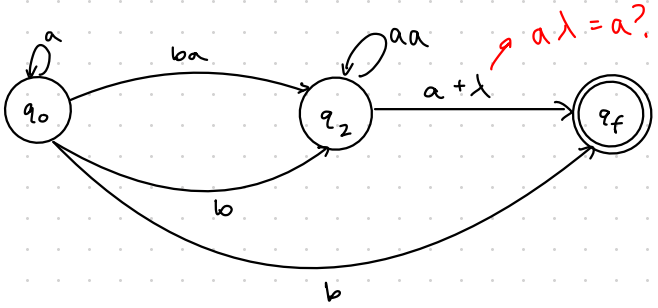
3) " q_0 to q_0 " q_1 ✗

4) " q_2 to q_0 " q_1 ✗

5) " q_2 to q_2 " q_1 ✓

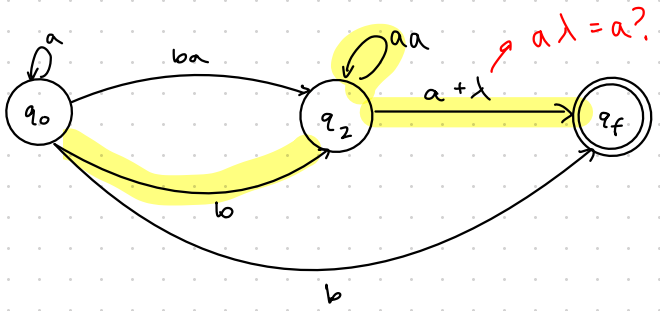
6) " q_2 to q_f " q_1 ✓

Similarly for q_f to all other nodes



- q_f to q_0 through q
- q_f to q_2 "
- q_f to q_f "

Now eliminate q_2



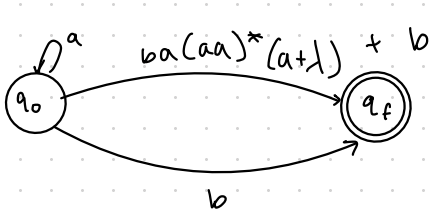
Remove q_2

1) walks from q_0 to q_0 through q_2

2) " q_0 to q_f "

3) " q_f to q_0 "

4) " q_f to q_f "



$$a^*(b + ba(aa)^*(a + \lambda))$$

$$(aa)^*(a+\lambda) = (aa)^* a + (aa)^*$$
$$a^*$$

$$a^*(b + baa^*)$$

$$a^*b + a^*baa^* = a^*ba^*$$

★ Context - Free Grammar

$G = (V, T, S, F)$

V → variables

T → terminal symbols

S → start symbols

P : production rule

$\underline{V_i} = x_1 x_2 \cdots x_n V_j y_1 y_2 \cdots y_m V_u \cdots$

↗ only one variable

Example :

$V = \{S\}$

$T = \{a, b\}$

P :

$S \rightarrow aSb \mid \lambda$

Derivation

$S \rightarrow aSb \rightarrow aaSbb \rightarrow aabb$

$S \rightarrow aSb \rightarrow aaSbb \rightarrow aaaSbbb \xrightarrow{\text{Derivation}} aaabbb$

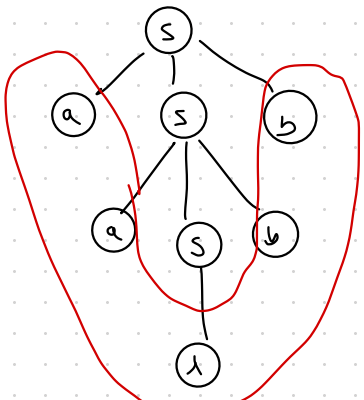
$L(G) = \{a^n b^n \mid n \geq 0\}$

$a \leftrightarrow ($

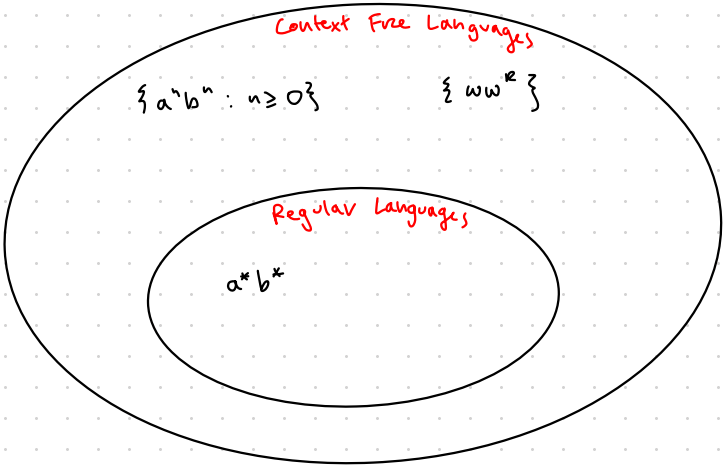
$b \leftrightarrow)$

$(((\,)))$

Derivation Tree



read left to right



A language is context free if and only if there is a context-free grammar that accepts it

G: $S \to aSa$
$S \to bSb$
$S \to \lambda$

palindrome

$S \to aSa \to abSba \to abba$

G: $S \to aSb$
$S \to SS$
$S \to \lambda$

$S \to SS \to aSbaSb \to abab$

$L(G) = \{w : n_a(w) = n_b(w), \ldots\}$

Derivation Order

1. $S \to AB$ 2. $A \to aaA$ 4. $B \to Bb$
3. $A \to \lambda$ 5. $B \to \lambda$

Leftmost derivation

go from left, derive, shift

Rightmost derivation

go from right, derive, shift

Tips for Designing Grammars

Concatenation: use separate nonterminals to generate disjoint parts of a language, then combine in production.

$S \to AB$
$A \to$
B

Closure: use recursive production

$A \to xA \mid \lambda$ 0+ xs

$A \to yA \mid y$ 1+ ys

Matching constructs:

- $S \to aSb \mid \lambda$ (start from middle)

$\{ a^n b^m \mid m \geq 2n, n \geq 0 \}$

- $S \to aSbb \mid B$
  $B \to bB \mid \lambda$

$\{ a^n b^m a^{n+m} \mid n \geq 0, m \geq 0 \}$

$a^n b^m a^m a^n$

- $S \to aSa \mid B$
- $B \to bBa \mid ba$

✗

- $S \to aSa \mid B$
  $B \to bBa \mid \lambda$

Union : use separate nonterminals

$L = \{ w \mid \#\text{ of a's in } w = \#\text{ of b's in } w \}$

$T = \{a, b\}$

Example

abaabb

(V, T, S, P)

P: S → produce equal # of a's and b's
A → " one a more than # of b's
B → " one b more than # of a's

productions:

$S \to aB$
$A \to a$
$A \to aS$
$A \to bAA$

$S \to bA$
$B \to b$
$B \to bS$
$B \to aBB$

$S \to aaaAB$
$A \to aA$

Thu Feb 3

CFG

$L = \{ w \mid w$ contains equals # of a's and b's $\}$

$G = (V, T, S, P)$

$V = \{S, A, B\}$

S → equal # of a's and b's

A → # of a's is one more than # of b's

B → # of b's is one more than # of a's

Production Rules

$S \to aB$

$B \to b$

$B \to bS$

$B \to aBB$ (equal, one more)

$S \to bA$

$A \to a$

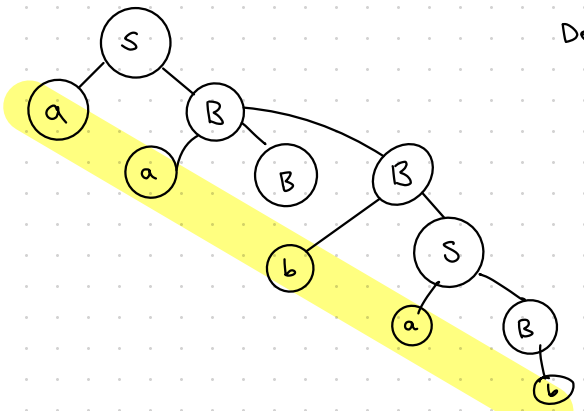
$A \to aS$

$A \to bAA$

aabbab

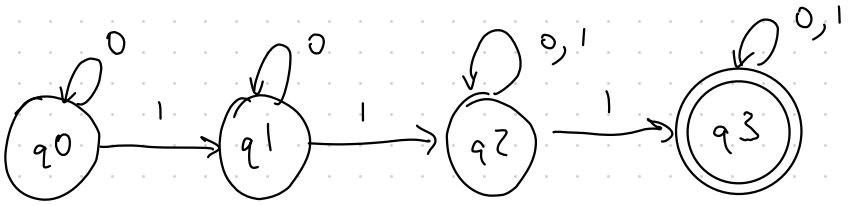
$S \to aB \to aaBB \to aabB \to aabbS \to aabbaB \to ababab$

Derivation Trees

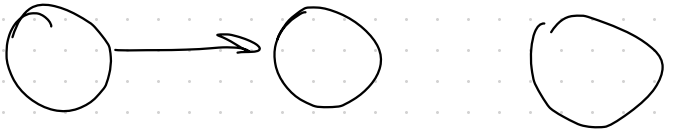


1a) $L_1 = \{w \in \{0,1\}^* \mid w \text{ contains at least three 1's}\}$

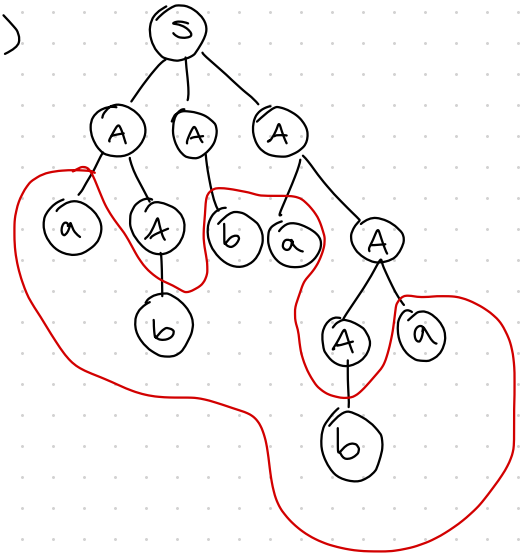
a) $L_1 = \{w \in \{0,1\}^* \mid w \text{ contains at least three 1's}\}$

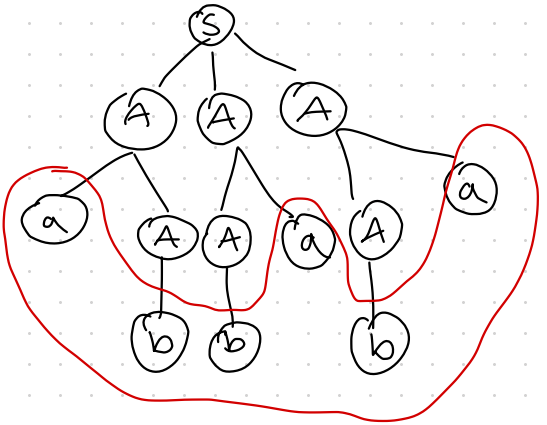


b) $L_2 = \{a^i b^j c^k \mid i, j, k \geq 0, \text{ and } i = j \text{ or } i = k\}$



b)





2a) abbaba

$S \to SS \mid AAA \mid \lambda$

$A \to aA \mid Aa \mid b$

Consider the following grammar $G = \{S, A\}, \{a, b\}, S, P\}$

$S \to SS \mid AAA \mid \lambda$

$A \to aA \mid Aa \mid b$

Now the production rules are:

$S \to SS$

$S \to AAA$

$S \to \lambda$

Next, the leftmost derivation rules will be

$A \to aA$

$A \to Aa$

$S \to SS$

$S \to AAAS$

$S \to AAA \quad (\because S \to \lambda)$

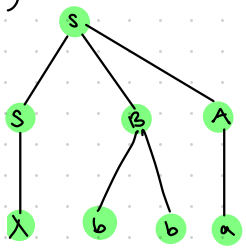
$S \to aAAA \quad (\because A \to aA)$

$S \to abAA \quad (\because A \to Aa)$

$S \to abbA \quad (\because A \to b)$

$S \to abbaba$

2. b)



bba

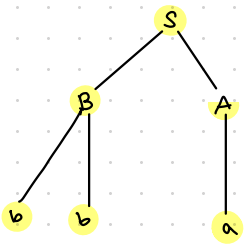
$S \to SBA$

$A \to a$

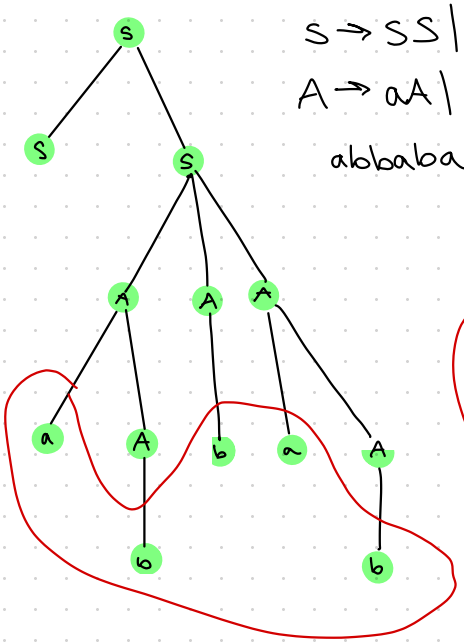
$B \to bb$

$B \to a$

$B \to \lambda$



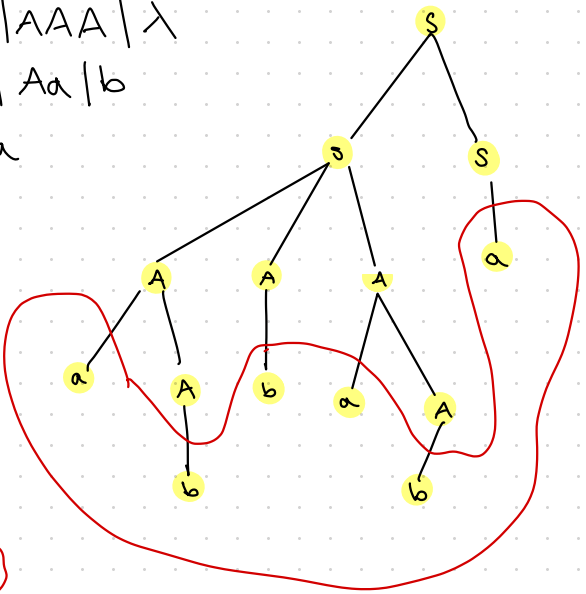
bba



$S \to SS \mid AAA \mid \lambda$

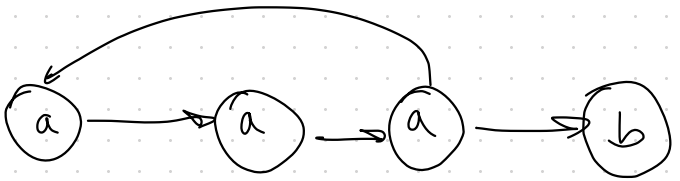
$A \to aA \mid Aa \mid b$

abbaba



3 a) $L = \{a^n b^{2n} : n \geq 2\}$

b) $L = \{aaa^* b\}$



4. Chomsky normal form

$S \to AB \mid aB$

$A \to abb \mid \lambda$

$B \to bbA$

---

5 a) CYK algorithm is in the context free language

$V = \{S, A, B, C, D\}$ and $T = \{a, b\}$

$S \to AB \mid BA$

$A \to AS \mid a$

$B \to BS \mid b$

$A \to BC$

$B \to AD$

$C \to AA$

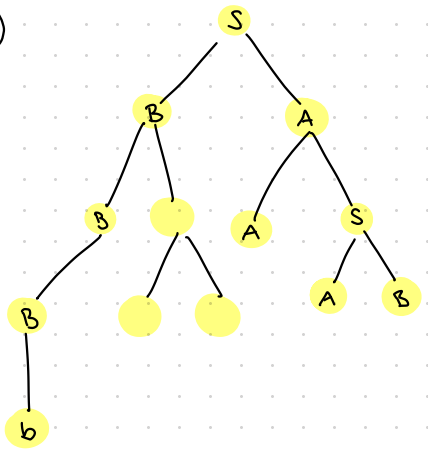
$D \to BB$

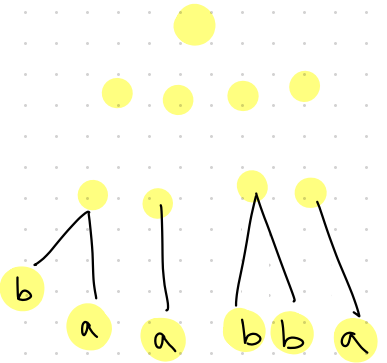
show

Ok. So.

$T = \{a, b\}$

b)





5.

$S \rightarrow AB \mid BA$

$A \rightarrow AS \mid a$

$B \rightarrow BS \mid b$

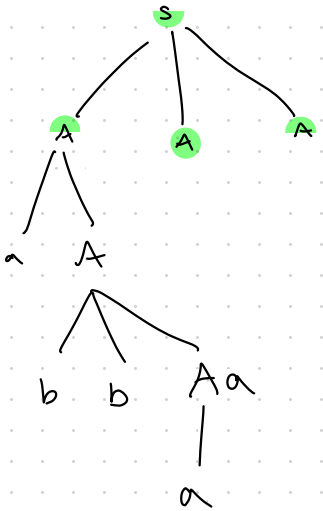
$A \rightarrow BC$

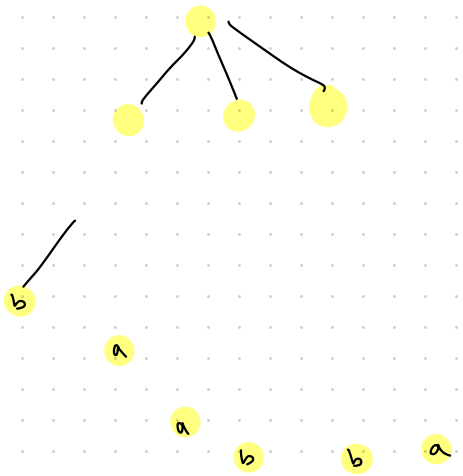
$B \rightarrow AD$

$C \rightarrow AA$

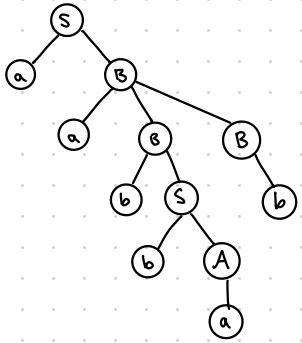
$D \rightarrow BB$

a) Show all derivation trees. baabba





a)



Another Derivation

'Ambiguous' : Language is called ambiguous if you can make more than one derivation tree for a string

Example 2

$L = \{V, T, S, P\}$

$V = \{E, I\}$

$T = \{a, b, c, +, *, (,)\}$

$S = E$

production rules

$E \rightarrow E * E$

$E \rightarrow E + E$

$E \rightarrow (E)$

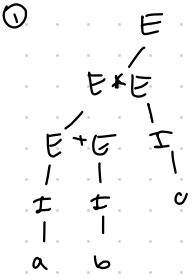
$E \rightarrow I$

$I \rightarrow a \mid b \mid c \mid * \mid +$

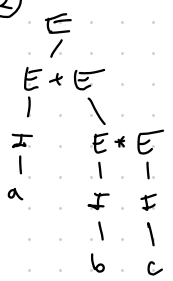
$a + b * c$

If $a = b = c = 2$, then two different answers!

①



②



Non - ambiguous CFG

$V = \{ E, T, F, I \}$

production rules

$E \to T$ $\quad$ $E \to I$

$T \to F$

$F \to I$

$E \to E + T$

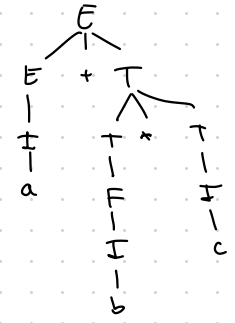
$T \to T * F$

$F \to (E)$

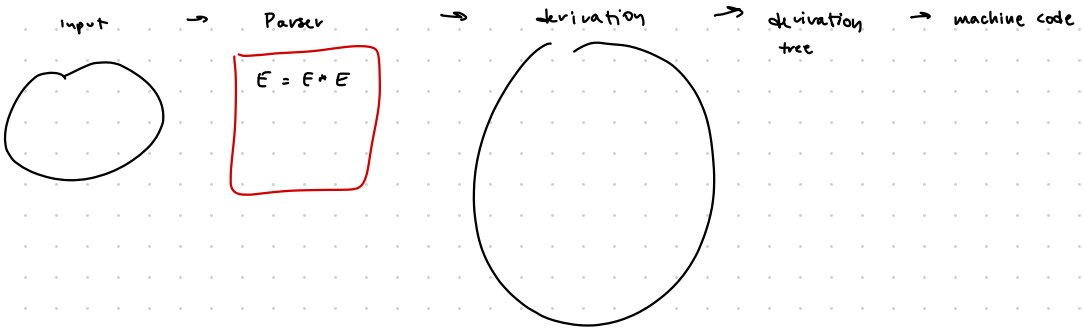
$I \to a \mid b \mid c$

$(a * b) + (c * a) + (b * c)$

$a + b * c$



Parsing



Time to drive:
length of the string

$S \to aS$
$S \to bSS$
$S \to c$

$S \to aS \to abSS \to abcS \to abcc$

Tue Feb 8

Chapter 6

Simplification of Grammars

Two standard forms

① Chomsky Normal Form

production rules

$A \to BC$ or $A \to a$

② Greibach Normal Form

production rules

$A \to aV \qquad V \in \{V\}^*$

Example:

$A \to aBCD$

$A \to bABC$

$A \to a$

$A \to abBB$ ✗

Substitution Rules

$A \to xBy$

$B \to \gamma_1 | \gamma_2 \cdots | \gamma_t$

$x, y \in T^* \qquad \gamma_i \in \{T, V\}^*$

Eliminate $B \to \gamma_1 | \gamma_2 \cdots | \gamma_t$ by substituting

$A \to x\gamma_1 y | x\gamma_2 y | x\gamma_3 y | \cdots x\gamma_t y$

Example

$A \to aBC | a$

$B \to bBC | bAB | b$

$C \to c | AB | AC$

eliminate these productions ($C \to c | AB | AC$) by substituting

$A \to aBc | aBAB | aBAC | a$

$B \to bBc | bBAB | bBAC | bAB | b$

Let us assume the language does not produce λ. $S \Rightarrow \lambda$

Even if $S \Rightarrow \lambda$ then we can modify the grammar as

$S \to S_1 | \lambda$

$S_1 \to$ { S_1 does not produce λ

1. Eliminate λ productions

2. Eliminate unit productions
   $(A \to B)$

3. Eliminate all useless productions

3. Eliminating useless productions

Example:

$S \to aA$
$S \to bB$
~~$S \to cD$~~ useless
$A \to A_1 A_2$
$B \to B_1$
skip! $B \to B_2$ X
$A_1 \to a$
$A_2 \to ab$
$B_1 \to bb$
~~$C \to c$~~ useless

Step 1
$U \neq \phi$
$U = \{ C, B_2, A_2, A_1, B, A, S \}$

Step 1. If some variables do not guarantee terminal symbols then those variables are useless

potential useful symbols:

Let $U = \phi$

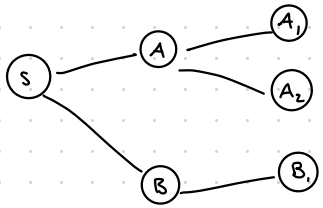
1. If $A \to a$ then put A in U
2. If $X \to z_1 X_1 z_2 X_2 z_3 X_3 \dots z_t X_t$
   where $z_i \in T^*$
   and $X_1, X_2, \dots X_t \in U$
   then put X in U.

New production rules

$S \to aA$
$S \to bB$
$A \to A_1 A_2$
$B \to B_1$
$A_1 \to a$
$A_2 \to ab$
$B_1 \to bb$
$C \to c$ X

Step 2
$\{ S, A, A_1, A_2, B, B_1 \}$

Step 2. Find all variables reachable from S



Updated production rules

$S \to aA \mid bB$
$A \to A_1 A_2$
$B \to B_1$
$B_1 \to bb$
$A_1 \to a$
$A_2 \to ab$

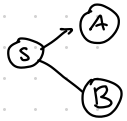
Example 2

$S \to aAB \mid a$
$A \to a$

Step 1
$A \to a$
$S \to a$

Step 2
$S \to a$

Suppose you apply step 2 first, step 1 second



Eliminate $S \to aAB$
Left with $S \to a$
$A \to a$

WRONG. still left w/ useless $A \to a$

Must do step 1 then 2.

(2) Elimination of unit production

Can I apply substitution rule?

$A \to B$
$B \to \ldots$

Example

$S \to A \mid bb$
$A \to B \mid b$
$B \to S \mid a$

Try to eliminate $S \to a$

- use substitution

$S \to B \mid b \mid bb$
$A \to B \mid b$
$B \to S \mid a$

Now we generated a new unit production $S \to B$

Eliminate $A \to B$
$S \to A \mid aa$
$A \to S \mid a \mid b$
$B \to S \mid a$

Introduce $A \to S$ (new production)

Find out all unit productions from a variable and then use substitution rule.

Given example:

$S \to A \mid bb$
$A \to B \mid b$
$B \to S \mid a$

$S \to A \to B$

$S \to A \to B$

Modify the grammar

$S \to bb \mid b \mid a$ ✓

$A \to B \to S$

$A \to b \mid a \mid bb$ ✓

$B \to S \to A$

$B \to a \mid bb \mid b$ ✓

New production rules

$S \to bb \mid b \mid a$
$A \to b \mid a \mid bb$
$B \to a \mid bb \mid b$

] useless production

① Elimination of λ productions

$A \to x_1 B x_2 B x_3 B x_4$
× $B \to \lambda$

Include new productions

$A \to x_1 B x_2 B x_3 B \mid x_1 x_2 B x_3 B \mid x_2 B x_1 x_3 B \mid x_1 B x_2 B x_3 \mid$

$x_1 x_2 x_3 B \mid x_1 B x_2 x_3 \mid x_1 x_2 B x_3 \mid x_1 x_2 x_3$

K - nullable variable
then include $2^K - 1$ new rules

Nullable variable

If $x \overset{*}{\Rightarrow} \lambda$ then x is called a nullable variable.

How to find all nullable variables?

$N = \phi$

1) If $B \to \lambda$ then put B in N
2) If $x \to x_1 x_2 \dots x_n$ where $x_1, x_2, \dots x_n \in N$ then put x in N.

Example:

$S \to aAB \mid BbbB \mid aA_1A_2$
$B \to B_1B_2 \mid b$
$B_1 \to \lambda$
$B_2 \to \lambda$
$A_1 \to a$
$A_2 \to b \mid \lambda$

① $N = \{A_2, B_1, B_2, B\}$

① $N = \{A_2, B_1, B_2, B\}$

$B_1 \to \lambda$, $B_2 \to \lambda$, $A_2 \to \lambda$

$A_2 \to b$

$A_1 \to a$

$B \to B_1 \mid B_2 \mid b$

$S \to aAB \mid BbbB \mid aA_1A_2 \mid aA \mid bbB \mid Bbb \mid bb \mid aA_1$

Order !

1. Eliminate λ production
2. Eliminate unit production
3. Eliminate useless productions

Example :

$S \to aA \mid aBB$

$A \to aaA \mid \lambda$

$B \to bB \mid bbC$

$C \to B$

1) Eliminate lambda production

$A \to \lambda$

$S \to aA \mid aBB \mid a$

$A \to aaA \mid aa$

$B \to bB \mid bbC$

$C \to B$

2) Eliminate unit production

$C \to B$

Introduce new production

$C \to bB \mid bbC$

3) Eliminate useless production

$S \to aA \mid aBB \mid a$

$A \to aaA \mid aa$

$B \to bB \mid bbC$

$C \to bB \mid bbC$

Step 1

Find $X \xrightarrow{*} T^*$

$\{A, S\}$

$\{ S \to aA \mid a \}$
$\{ A \to aaA \mid aa \}$ $S \to A$

Final production rules !

$S \to aA \mid a$
$A \to aAA \mid aa$

$S \to T_aA \mid a$ $\quad T_a \to a$

$A \to T_aT_aA \mid T_aT_a$

$A \to T_aT_aA$

$\Rightarrow A \to T'A$
$T' \to T_aT_a$

$S \to T_aA \mid a$
$A \to T'A \mid T_aT_a$
$T' \to T_aT_a$
$T_a \to a$

chomsky Normal Form

Thu Feb 10

pumping lemma

$xyz \in L$
$xy^i z \in L$

Example:

$S \to baAB$
$A \to bAB \mid \lambda$
$B \to BAa \mid A \mid \lambda$

1) Eliminate λ productions

$A \to \lambda$, $B \to \lambda$

$S \to baAB \mid baB \mid baA \mid ba$
$A \to bAB \mid bB \mid bA \mid b$
$B \to BAa \mid Aa \mid Ba \mid a \mid A$

2) Eliminate unit productions

only one is $B \to A$ ⇒ substitute

Include

$B \to BAa \mid Aa \mid Ba \mid a \mid bAB \mid bB \mid bA \mid b$

3) Eliminate useless productions

(No useless)

Final

$S \to baAB \mid baB \mid baA \mid ba$
$A \to bAB \mid bB \mid bA \mid b$
$B \to BAa \mid Aa \mid Ba \mid a \mid bAB \mid bB \mid bA \mid b$

Not in CNF

- $S \to baAB$
  $S \to T_b T_a AB$
  $\Rightarrow S \to CAB$
  $\Rightarrow S \to DB$ ✓

  $T_b \to b$ ✓
  $T_a \to a$ ✓
  $C \to T_b T_a$ ✓
  $D \to CA$ ✓

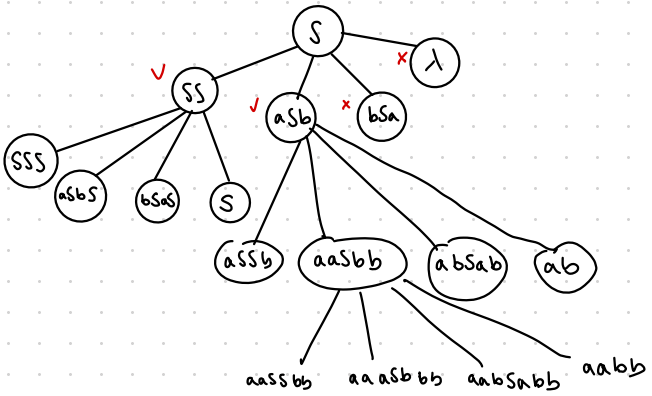
- $S \to baB$
  $S \to T_b T_a B$
  $\Rightarrow S \to CB$

- $S \to baA$
  $S \to T_b T_a A$
  $\Rightarrow S \to CA$

Grammar

$S \to SS$
$S \to aSb$
$S \to bSa$
$S \to \lambda$

aabb?
$S \overset{*}{\to} aabb$



each node has 4 possibilities

Length of the string is $|w|$
K production rules
$K + K^2 + K^3 + \dots K^w$

Dynamic Algorithm

$L = \{w \mid n_a(w) = n_b(w)\}$

bababa?

production rules

$S \to aB$ $\quad S \to bA$
$B \to b$ $\quad A \to a$
$B \to bS$ $\quad A \to aS$
$B \to aBB$ $\quad A \to bAA$

Chomsky Normal Form

$S \to A_1 B$
$S \to B_1 A$
$B \to B_1 S$
$A \to A_1 S$
$B \to A_1 D$
$D \to BB$
$A \to B_1 C$
$C \to AA$

$A_1 \to a$
$B_1 \to b$
$A \to a$
$B \to b$

bababa ?

b | ababa
B C

ba | baba

bab | aba

baba | ba

babab | a

Tue Feb 15, 2022

Example:

ababab

a | baba

ab | abab

aba | bab

abab | ab

ababa | b

x | y

$S \to T_b A \quad S \to T_a B$

$A \to T_a S \quad B \to T_b S$

$A \to T_b C \quad B \to T_a D$

$C \to AA \quad D \to BB$

$T_b \to b$

$T_a \to a$

$B \to b$

$A \to a$

	1	2	3	4	5	6
1	T_a, A					
2	S	T_B, B				
3	A	S	T_A, A			
4	S	B	S	T_B, B		
5	A	S	A	S	T_a, A	
6		B	S	B	S	T_b, B

$S \xrightarrow{*} xy$

ababab

$v_1 \; v_y \; v_{bb}$

$T_a T_b$ AB

T_a B

A T_b

abba

S

aba | a | ba

$T_a S$ $\quad ST_a$

AS $\quad SA$

A

ababa

$T_a S$

BS

ST_b

SB

$T_a S$

AS

$T_a B$ ✓

$\leq B$ ×

$A \leq x$ ×

$S \leq x$ ×

$A T_b x$ ×

$T_a B$

AB

AT_b *

AB *

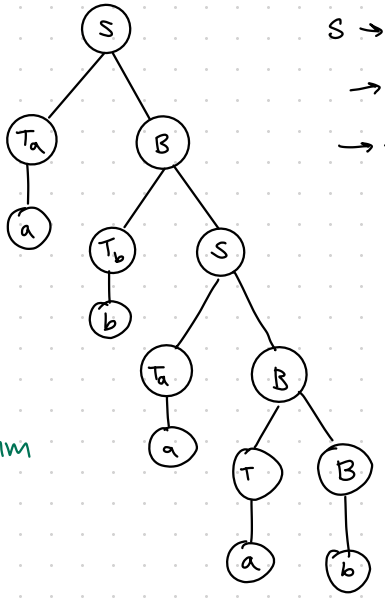
$T_b S$ ✓

SB ×

BS×

$\leq T_b$ ×

$< B$ ×



$S \to T_a B$

$\to T_a T_b S$

$\to T_a T_b T_a B$

CYK Algorithm

$|w|$

complexity $= O(|w|^3)$

$\frac{|w|^2}{2}$ entries

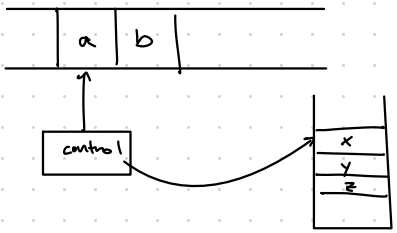
Regular Languages $\leftrightarrow$ DFA $\leftrightarrow$ NFA

CFL $\leftrightarrow$ NPDA

Nondeterministic push-down automata

NPDA $\geq$ DPBA

Nondeterministic push-down automata



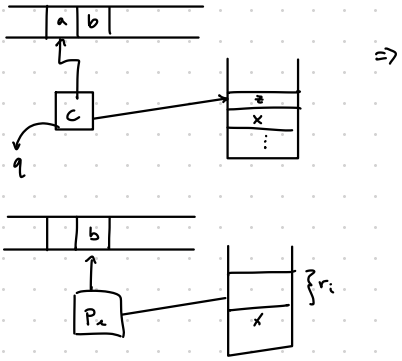
$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

'STACK'

Q – set of states
Σ – tape symbols (small letters)
T – stack symbols (capital letters)
q_0 – initial state
$z_0 \in T$ – start stack symbol
F – set of final state
δ – transition function

$\delta : Q \times \{\Sigma \cup \{\lambda\}\} \times T \rightarrow$ a subset of $Q \times T$

$\delta(q_1, a, z) = \{\{p_1, r_1\}\{p_2, r_2\} \ldots \{p_i, r_i\} \ldots\}$



Note that $r_1 = \phi$

Note that a can be λ

Example:

$L = \{w \mid \pi_a(w) = \pi_b(w),\ w \in \{a,b\}^*\}$

$\Sigma = \{a, b\}$
$Q = \{q_0, q_1, q_2\}$
$T = \{A, B, C\}$

q_0 – initial state
q_2 – final state

δ:

$\delta(q_0, \lambda, \lambda) = \{q_1, C\}$

$\delta(q_1, a, C) = \{q_1, AC\}$

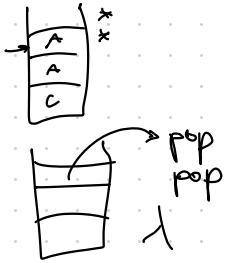
$\delta(q_1, a, A) = \{q_1, AA\}$

$\delta(q_1, b, A) = \{q_1, \lambda\}$

$\delta(q_1, \lambda, C) = \{q_2, \lambda\}$

$\delta(q_1, b, C) = \{q_1, BC\}$

$\delta(q_1, b, B) = \{q_1, BB\}$



$\delta(q_1, \lambda, C) = \{q_2, \lambda\}$

Main idea

The stack contains

[# of a's → # of b's]

abba

$(q_0, \lambda) \xrightarrow{\lambda} (q_1, C) \xrightarrow{a} (q_1, AC) \xrightarrow{b} (q_1, C) \xrightarrow{b} (q_1, BC) \xrightarrow{a} (q_1, C)$

$(q_1, C) \xrightarrow{\lambda} (q_2, \lambda)$

$(q_1, C) \xrightarrow{\lambda} (q_2, \lambda)$

$L = \{a^n, b^{2n}\}$

Q5.

dynamic scoping

8, 4, 9 dynamic

```
{ int x;
  x := 2;
  { int f(int y) {
      x := x * y;
      return (x+1);
    };
    { int x;
      x := 4;
      x := f(x-1);
    };
  };
}
```

4

8

9

y gets the 5

y is still on

after 4: [y:3, x:12, f:{}, x:2]

after 9: [x:13, f:{}, x:2]

static scoping

2: [x:2]

6: [f:{}, x:2]

after 8: [x:4, f:{}, x:2]

x := x * y;

The static

```
{ int x;
  x := 2;
  { int f(int y) {
      x := x * y;
      return (x+1);
    };
    { int x;
      x := 4;
      x := f(x-1);
    };
  };
}
```

[y:3, x:4, f:{}, x:6]

[x:7, f:{}, x:6]

static and dynamic

```
int g(int y){return f(2)};
z := g(3)
```

```
[]
[y:1]
[z:0, y:1]
[f:{}, z:0, y:1]
[g:{}, f:{}, z:0, y:1]
[y:3, g:{}, f:{}, z:0, y:1]
```

call of g
call of f

```
{int y := 1;
  {int z := 0;
    {int f(int x){return y+x};
      {int g(int y){return f(z)};
        z := g(3);
      }
    }
  }
  . . .
```

[x:2, y:3, g:{}, f:{}, z:0, y:1]

```
{int z := 0;
  {int f(int x){return x+1};
    {int g(int y){return f(y)};
      {int f(int x){return x-1};
        z := g(3);
      }
    }
  }
  . . .
```

2

```
[]
[z:0]
[f:{}, z:0]
[g:{}, f:{}, z:0]
[f:{}, g:{}, f:{}, z:0]
```

call of g
call of f

```
[y:3, f:{}, g:{}, f:{}, z:]
[x:3, y:3, f:{}, g:{}, f:{}, z:0]
```

static: [f:{}, g:{}, f:{}, z:4]

dynamic: [f:{}, g:{}, f:{}, z:2]

Access function definition

CS 321 Introduction to Theory of Computation

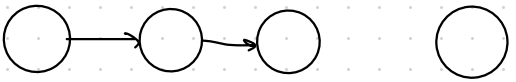
Assignment No. 5, Due: Friday February 18 2022

1. a) $L_1 = \{w \in \{0,1\}^* \mid w \text{ contains at least three 1's}\}$.

$w \rightarrow 111$

b)

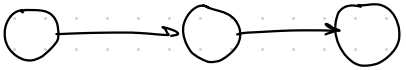
b) $L_2 = \{a^i b^j c^k \mid i, j, k \geq 0, \text{ and } i = j \text{ or } i = k\}$

2. 

3. Find an s-grammar

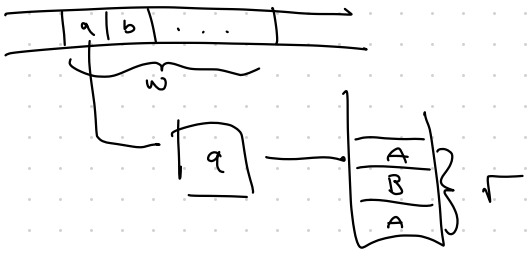
aa bbbb

n ↗ outside of the following



CS 321 Thursday Feb 17, 2022

Instantaneous Description $L = \{a^n b^{2n}\}$



$(q, w, r) \Leftrightarrow (q, abacd, ABA)$

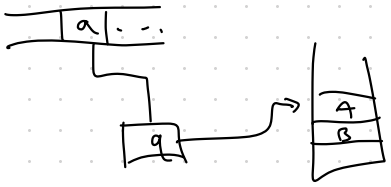
Move $(q, aw, A\alpha_1) \vdash (p, w, \beta\alpha_1)$

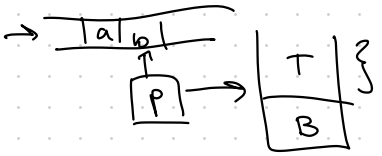
This means

$\delta(q, a, A)$ contains

$\{ (p, r) \}$

$\delta(q, a, A) = \{ (p, r) \ldots \}$





$q_1, a_1a_2 \ldots a_n, A_1A_2 \ldots A_n$

$\vdash (P_1, a_2a_3 \ldots a_n, B_1B_2 \ldots B_n, A_2 \ldots A_n)$

$\vdash (P_2, a_3 \ldots a_n, C_1C_2 \ldots C_4, B_2B_3 \ldots A_2 \ldots A_n)$

$\vdots$

$\vdash (P_n, a_n, \alpha)$

$\vdash (P_{n+1}, \lambda, \eta)$ $\quad \alpha, \eta \in \Gamma^*$

Language NDPA

(1) Using Final State

$$L(M) = \{w \mid (q_0, w, T_0) \overset{*}{\vdash} (P, \lambda, \gamma)\}$$

where $P \in F$

(2) Empty Stack (Null Stack)

$$N(m) = \{w \mid (q_0, w, T_0) \overset{*}{\vdash} (P, \lambda, \lambda)\}$$

we can prove

$$L(M) = N(M)$$

NPFA $\Leftrightarrow$ CFL

CFL $\Rightarrow$ NPDA ✓

NPDA $\Rightarrow$ CFL

Let L be a CFL described by the language $G_1 = (V, T, S, P)$

Construct

$$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$$

$$= (\{q\}, \Sigma = T, \Gamma = V, q_0, S, \{\phi\})$$

S: initial; ϕ: final

$$\delta: A \to a\alpha$$

$$\delta(q, a, A) \to (q, \alpha)$$

[Assumption:

The production in G_1 are in Greibach Normal form

$A \to av \qquad v \in V^*$]

Example:

$G_1 = (V, T, S, P)$

$V = \{S, S_1, S_2\}$

$T = \{a, b, c\}$

$P =$

$S \to aSS_1$
$S \to bSS_2$
$S \to a \mid c$
$S_1 \to a$
$S_2 \to b$

$S \to aSS_1 \to abSS_2S_1 \to abcbS_1 \to abcbaa$

palindrome

$S \to aSS_1$ $\quad \delta(q, a, S) = (q, SS_1)$

$S \to bSS_2$ $\quad \delta(q, b, S) = (q, SS_2)$

$S \to a$ $\quad \delta(q, a, S) = (q, \lambda$

$S \to c$ $\quad \delta(q, a, S) = \{(q, SS), (q, \lambda)\}$

$S_1 \to a$ $\quad \delta(q, a, S_1) = \{(q, \lambda)\}$

$S_2 \to b$ $\quad \delta(q, b, S_2) = \{(q, \lambda)\}$

abcba

$S \to aSS_1 \to abSS_2 \to abcS_2S_1 \to abcbS_1 \to abcba$

1. Pumping lemma for regular language

How does it work