Regular Language

- Deterministic finite automata
  =
- Nondeterministic finite automata
  =
- Nondeterministic finite automata with ε-move
  =
- Regular Expression
- Non regular language
  - pumping lemma
  - Equivalence theorem
  - My-Hill-Nerode Theorem ⇒ Finite Automata minimization

Context-Free Language

- Context-Free Grammars
  =
  - Ambiguity
  - Chomsky normal form
- Push-down Automata
- Non-Context-Free Language
  - Pumping Lemma

Computability Theory

- Church-Turing Thesis
  - Turing Machines
  - Countable & Uncountable sets
  - Definition of Algorithm
- Decidability
  - Decidable Languages
- Undecidability /
  - Diagonalization / Halting Problem
  - Reducibility
  - Rice's theorem

Complexity Theory

- P & NP
- NP-Completeness - Cook-Levin theorem
  - SAT, 3SAT
  - Vertex Cover / Independant Set / Clique
  - HAMC / HAMP - TSP
  - Matching / 3DM / Set Cover
  - Subset Sum

Finite Automata

A deterministic finite automata is $M = (Q, \Sigma, \delta, q_0, F)$

- Q : finite set of state
- Σ : finite set of symbols
- δ : a set of transit function $Q \times \Sigma \to Q$
- q_0 : start state
- F : set of final state

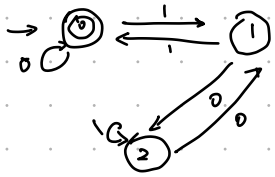
$\emptyset$: empty set

ε : empty string

Σ^* : set of all finite strings of symbol from Σ $= \{\varepsilon, 0, 1, 01, 10, \dots\}$ if $\Sigma = \{0, 1\}$

$\hat{\delta}: Q \times \Sigma^* \to Q$ $\quad \hat{\delta}(q, \varepsilon) = q$ $\quad \hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$

for input divisible by 3 $- v(xa) \bmod 3 = (2 \cdot v(x) + a) \bmod 3 = (2 \cdot v(x) \bmod 3 + a)$



NFA : $N = (Q, \Sigma, \delta, q_0, F)$ $\quad \delta: Q \times \Sigma \to \mathcal{P}(Q)$ (powerset)

Acceptance : is there a legal sequence of move from q_0 to $q \in F$

A NFA N accepts $x \in \Sigma^*$, starting with q_0, N has a legal sequence of move following q_i to some state $q \in F$

Anything can be expressed by DFA can be also expressed by NFA

Theorem : every NFA accept a language also accepted by DFA

$\forall$ NFA N $\quad \exists$ DFA M $\quad$ st. $L(M) = L(N)$

proof : Define a DFA M $\quad M = (\mathcal{P}(Q_N), \Sigma_N, \delta_M, \{q_0\}, F_M)$ $\quad F_M = \{S \subseteq Q \mid S \cap F_N \neq \emptyset\}$

$\delta_M : Q_M \times \Sigma = Q_M \quad (S \subseteq Q, a \in \Sigma) \to \bigcup_{q \in S} \delta(q, a)$

By an easy induction,

$\delta(M)(\{q_0\}, x)$ is the subset of all $q \in Q$ reachable by legal sequence of move N

Def : A subset $A \subseteq \Sigma^*$ is called a regular language if it is the set of steps accepted by some DFA/NFA

Theorem : if L_1 & L_2 are regular, $L_1 \cap L_2$ is also regular

proof : if $L_1 = L(M_1)$ $\quad L_2 = L(M_2)$

def $M = (Q_{M_1} \times Q_{M_2}, \Sigma, \delta = ((q^1, q^2), a) = (\delta^1(q^1, a), \delta^2(q^2, a)), F = F_1 \times F_2)$

if L is regular L^c is also regular

def $F = F^c$ (reverse the final / normal states) $\Rightarrow$ does not apply for NFA since its acceptance condition, works for DFA

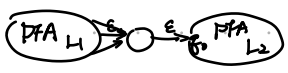
if L_1 and L_2 are regular, $L_1 \cup L_2$ also regular

create a NFA with start state connect to both DFAs.

if L_1 regular, L_1^* is also regular

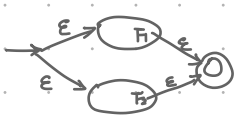
create NFA with ε- move, new start state connect to old q_0, new $q_0 \in F$ with ε-move from all $q \in F$

If L_1, L_2 regular $\quad L_1 L_2$ is also regular.

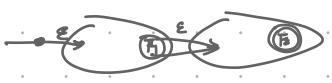
create NFA with ε- move, 

from all $q \in F_1$ to q_0^2

Union



Concat



*

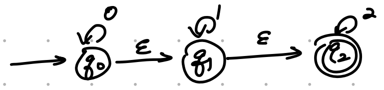
NFA with ε-move

$N_\varepsilon = (Q, \Sigma, \delta, q_0, F)$

$\delta: Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$

ε-closure: the ε-closure of a state $q \in Q$ in N_ε to all states reachable by 0 or more ε-move alone.

N-ε:



δ	0	1	2	ε
q_0	q_0	\	\	q_1
q_1	\	q_1	\	q_2
q_2	\	\	q_2	\

ε-closure $(q_0) = \{q_0, q_1, q_2\}$

$(q_1) = \{q_1, q_2\}$

$(q_2) = \{q_2\}$

NFA (without ε-move)

δ' $Q \times \Sigma \to \mathcal{P}(Q)$ $(q, a) \mapsto$ ε-closure$(\delta(q, a)) = \bigcup_{q' \in \delta(q,a)}$ ε-closure(q') = ε-closure $\left[\bigcup_{q' \in \varepsilon\text{-closure}(q)} \delta(q', a)\right]$

By induction on $|w| \geq 1$ $\forall w \in \Sigma^*$, N_3 starting with q_0, enter w, can streach exactly same set of states in N

DFA $\xrightarrow{\text{can be simulated}}$ NFA $\Longleftrightarrow$ NFA$_\varepsilon$

DFA $\xleftarrow{\text{special case}}$ NFA

Def: a set $L \subseteq \Sigma^*$ is called regular if it is accepted by a DFA (NFA, NFA-ε)

Regular Expression

ϕ empty set

ε empty string

a $\forall a \in \Sigma$

if r, s are regular expression, so is

$(r+s)$ $L(r) \cup L(s)$

$(r \cdot s)$ $L(r) \cdot L(s)$

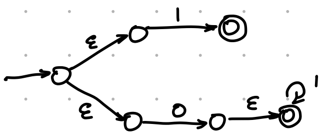
r^* $L(r)^*$

$L^* = \{x_1 \dots x_n \mid n \geq 0, x_i \in L\}$

$= \bigcup_{i \geq 0} L^i$, $L^0 = \{\varepsilon\}$, $L^{i+1} = L \cdot L^i$

Convert RE to NFA with ε

example: $01^* + 1$



Given by DFA $M = (Q, \Sigma, \delta, q_0, F)$, want to capture all strings from q_0 to F

$\forall q \in F$. if have $R_{q_0, q}$ as a regular exp that capture all strings that go from q_0 to q in M

$|F| = R_{q_0 q}, R_{q_0 q_1} + \dots$

Base case: $q = q'$, what are the string from q to q' for 0 steps ⇒ ε-string

· the regular exp for all strings of length ≤ 1 can be written:

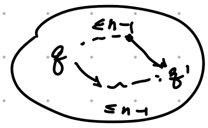
$R_{qq'} = \begin{cases} \varepsilon & \text{if no self loop} \\ \varepsilon + a + b & \text{otherwise, say it has loops labeled by a \& b} \end{cases}$

now we have $q \neq q'$ if there is only 1 step, no ε for DFA

$R_{qq'} = \begin{cases} \phi & \text{if } \phi \text{ direct transition} \\ a & \\ a+b & \text{for every transition from } q \text{ to } q' \text{ labeled by } \{a, ab, \dots\} \\ a+b+ \dots & \end{cases}$

for length n, as we have how to express 1, it recursively sub 1 and back to 1

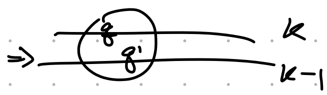
but it doesn't work as we don't have a single string



For every k $0 \leq k \leq n$ Consider [All strings that (R_{ij}^k) lead M from q_i to q_j passing through only state in $\{q_1, q_2, \dots q_k\}$ (never above k! start is ok)

intuition: if k raise to n, then problem solved

Base case: $k=0$, q to q' without going through any intermediate step (# of step ≤ 1)



need to prove from k to k+1, we have R_{ij}^{k-1} $\forall ij$

$R_{ij}^k = R_{ij}^{k-1} + R_{ik}^{k-1} R_{kk}^{k-1*} R_{kj}^{k-1}$ ↳ k is last visit

↑ before that all intermediate states ≤ k−1 (loop)

DFA | NFA | NFA/ε | Regexp

subset construction → all NFA are simulated with DFA

dynamic programming → start with DFA and try to express by RegR

- Not regular: sets in Σ^* cannot expressed by a DFA. to prove non-regular ⇒ Pumping Lemma

Pumping lemma for regular language.

for every regular language L, accepted by DFA, there exist an integer $n \geq 1$ such that, for every $x \in L$, with length $|x| \geq n$, there exist a decomposition $x = uvw$ such that $\begin{cases} |uv| \leq n \\ |v| \geq 1 \end{cases}$, $\forall i \geq 0$ $uv^iw \in L$

if L is regular, there is $n \geq 1$, $x \in L$ $|x| \geq n$ (pick any $x \in L$) win the game by showing L is nonregular

$L = \{a^n b^n \mid n \geq 1\}$

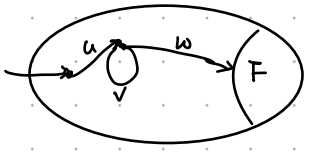
① for every $n \geq 1$, pick $a^n b^n \in L$

② for every decomposition of $x = uvw$ with $|v| \geq 1$, $|uv| \leq n$ v must be a

the resulting uv^iw for $i = 2$ not in L

Proof of Pumping Lemma

- prove L is regular so $\exists$ DFA M $L = L(M)$
- Let M has n states $\forall x \in L(M)$ $|x| \geq n$
- x will see $n+1$ occurance of states ⇒ same state repeated
- $x = uvw$ and as v is looping, after $v \to v^i$ $x \in L$ for $i \geq 0$



Strong form of Pumping Lemma

$\forall$ reg L $\exists n \geq 1$ $\forall x = x_1x_2x_3 \in L$ with $|x_2| \geq n$ $\exists$ decomp $x_2 = uvw$, $|v| \geq 1$ s.t. $\forall i \geq 0$, $x_1uv^iwx_3 \in L$

$L = \{w \mid w$ has same # of 0's and 1's$\}$

if $\forall n \geq 1$ consider 0^n1^n, $\forall$ decomposition with $|uv| \leq n$ $|v| \geq 1$, it must be that both $u, v \in 0^*$ $v \in 0^+$, then uv^2w has more 0 than 1

another way: $L_1 \cap L_2$ are also regular if L_1 L_2 regular

ex $L_1 = L$ $L_2 = 0^*1^*$ $L_1 \cap L_2 = 0^n1^n$

$\{ww \mid w \in (0+1)^*\}$

$\forall n \geq 1$ consider $0^n1^n0^n1^n$ $\forall$ decomposition uvw with $|uv| \leq n$ $|v| \geq 1$, v must be in 0

then $uv^2w \Rightarrow 0^{n+1}1^n0^n1^n \notin L$

$\{1^{n^2} \mid n \geq 1\}$ $(n+1)^2 - n^2 = 2n+1 \to \infty$ as $n \to \infty$

$\forall n \geq 1$ pick 1^{n^2}, $\forall$ decomposition uvw $|uv| \leq n$ $|v| \geq 1$ $i = 2 \to 1^{n^2+|v|}$ $|v| \leq n$, $|v|$ need to be at least $2n+1$

$L = \{a^n \mid n$ is a prime$\} = \{a^2, a^3, a^5, a^7, a^{11}, a^{13}, \dots\}$ $\pi(N) \approx \frac{N}{\ln N}$: n-th prime $\approx n \ln n$

we will pick $p > n$ since infinite many prime, $\forall$ decomposition uvw $|uv| \leq n$ $|v| \geq 1$

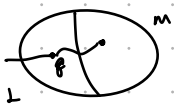
$a^{p+k \cdot j}$ $k = |v|$ need to show $p + kj$ is not prime j could be $p \to p + kj = p + kp = p(k+1)$

pitfall: $L = \{w \in 1^*2^*, \#_1(w) \neq \#_2(w)\}$

$\forall n \geq 1$, pick $1^n2^{n+n!}$ for decomposition $x = uvw$, v must be 1

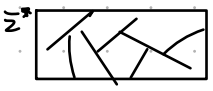
$x' = 1^{n-|v|}1^{|v| \cdot i}2^{n+n!} = 1^{n+|v|\cdot(i-1)}2^{n+n!}$ for $i = 1 + \frac{n!}{|v|}$, $x' \notin L$

Equivalence relation



R_M $x \underset{M}{\equiv} y$ iff $\delta(q_0, x) = \delta(q_0, y)$

R_L $x \underset{R_L}{\equiv} y$ iff $\forall z \in \Sigma^*$ $xz \in L \Leftrightarrow yz \in L$



finite piece ⇒ since finite states.

if 2 points are from same pieces, then $R_M = R_L$

equalence relation

① identity

② transform $x = y$ $y = x$

③ $x = z$ $y = z$ $x = y$

Myhill - Nerode Theorem

Def DFA from R_L { the space are set of equalent class of R_L, Q

$\delta([x], a) = [xa]$

$q_0 = [\varepsilon]$ $\delta([\varepsilon], 0) = 0$

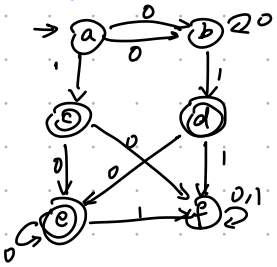
$F = \{[x] \mid x \in L\}$

↳ not working since Fis not finite!

L is regular iff # of equivalent class is finite.

↳ unique DFA exist for regular language

↳ equivalence class of R_L paitition Σ^* in to chunks



$C_A = (00)^*$ $C_b = 0(00)^*$ or $(00)^*0$

$C_{cd} = 0(00)^*1$ $C_e = (00)^*1$

$C_e = (00)^*10 + (00)^*010)0^*$ cf $(C_e + C_d + C_e)10 + 1)^*$

$= 0^*1 00^*$

Minimizing DFA

① make every pair $(p\ q)$ $p \in F$ $q \notin F$

② $\forall a \in \Sigma$ $\forall (p, q)$ not marked yet, if $(\delta(p,a), \delta(q,a))$ is already marked, mark it.

↳ if witness exist to distinguish 2 states

↳ repeat, stop when go through all unmark without adding.

- prove correctness of terminate

→ marks should be separated by induction

→ not marked are related R_L since xz, yz one in final state. the other not

if they are separated by empty string, they should be marked in first process

↳ so distinguisher is not ε

↳ x ends in q

y ends in q' (q, q') unmarked at the end

Induction: $\forall z \in \Sigma^*$ $|z| = n$ $xz \in L \Leftrightarrow yz \in L$

$\Rightarrow \forall z' \in \Sigma^*$ $|z| \leq n+1$ $z' = az$ $|z| = n$ $\delta(q,a), \delta(q',a)$ cannot be distinguished by length n.

x, y integer $d \mid n$ $(0, 1, 2 \dots n-1)$

$\{0, d, 2d, \dots (\frac{n}{d}-1)d = n-d\}$ they all have n/d elements, have d classes

$\{1, d+1, d+2, \dots (n-d+1)\}$

↳ take any a is relatively prime to n $a \perp n$

$\{a\ \%\ n, a^1 \bmod n, \dots a^t, a^s \equiv a^t \bmod n\}$ ← $\exists x \cdot (a^s \equiv a^t \bmod n) \equiv 1$

$a^s \equiv a^t \bmod n$ $a^{s-1} = a^{t-1} \bmod n$ $t > 0$

$\exists x, y$ relatively prime

$x \cdot a - y \cdot n = 1$ $x \cdot a \equiv 1 \bmod n$

def $x \equiv y$ if $\exists i$ $x = y \cdot a^i \bmod n$

for $x, y \perp n$ $\{x, x \cdot a, x \cdot a^2, \dots x \cdot a^{s-1}\}$ if y is in set it is equivalence as x

$s \mid \Phi(n)$ = # of integer $k \perp n$

if n is prime $\Phi(p) = p - 1$ $a \in \{1, 2 \dots p-1\}$ $a^s \equiv 1$ when $s \mid p-1 \cdot k \cdot s$

$a^{p-1} = (a^s)^k \equiv 1 \bmod p$ Fermat theorem

Chinese remainder theorem

p, q are distinct primes. $\Phi(pq) = (p-1)(q-1)$

PRIME is not Regular

- pick prime $p = V(q) > 2^n$ $|p| > n$
- $x = uvw$ $|uv| \le n$ $|v| \ge 1$ $v(u) \cdot 2^{|v|+|w|} + v(v) \cdot 2^{|w|} + v(w)$ ⇒ value of string

pump ↳ $v(x)' = v(u) \cdot 2^{i|v|+|w|} + v(v) \cdot 2^{|w|} \cdot (1 + 2^{|v|} + \dots + 2^{|v|(i-1)}) + v(w)$

$= v(u) \cdot 2^{|w|} \cdot (2^p)^{|v|}$ ↳ $\frac{\alpha^p - 1}{\alpha - 1}$

$v(w) + v(v) \cdot 2^{|w|} \cdot 1 + v(u)\ 2^{|w|+|v|} \equiv \alpha(p) = x = p$. which is a prime $\equiv 0 \bmod p$

- find something bigger than p but divisible by p ⇒ PRIME is not regular

RSA : p, q are distinct primes, $x \in \cdot Z^*_{p \cdot q}$

Enc : select a key e relatively prime to $\Phi(p \cdot q)$

To send x: compute $x^e \bmod pq$

Dec : receiver has decrypt key d (secret) s.t. $e \cdot d \equiv 1 \bmod \Phi(pq)$

take $x^e \to (x^e)^d = x^{1+m \cdot (p-1)(q-1)} = x$

Context free language

$\{0^n 1^n \mid n \geq 1\}$ is not regular language, but Context free

$S \to 0S1 \mid 01 \mid \varepsilon$

{ Production / substitution rules : Symbol (variable) & String (terminals) separated by arrow
  Starting variable : first line left most

Derivation: write down start variable, substitute until no variable remain

↳ ∀ strings generated in this way ⇒ language of grammar $L(G_1)$ → language of grammar G_1

Any language can be generated by context free grammar ⇒ CFL

Context-free Grammar is a 4-tuple (V, Σ, R, S)

1. V is a finite set called variables
2. Σ is a finite set, disjoint from V, terminals
3. R is finite set of rules, each rule : variable → string of variable and terminal
4. $S \in V$, start variable.

$R \to P$ P: finite set of predictions
A prediction is a rule of form $A \to \alpha \in (V \cup \Sigma)^*$

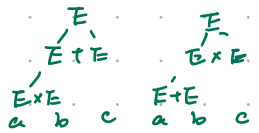
DFA → CFG

① variable R_i for each state q_i
② Rule $R_i \to aR_j$ if $\delta(q_i, a) = q_j$
③ $R_i \to \varepsilon$ if $q_i \in F$
④ R_0 as start if q_0

Ambiguity: a grammar can generate the same string in more than one ways / 2 diff derivation tree

↳ some CFL can only be generated by ambiguous grammar ⇒ inherently ambiguous

$E \to a \mid b \mid E+E \mid c \mid E \times E$



$E \to E+T \mid T$
$T \to T \times F \mid F$
$F \to (E) \mid a \mid b$

proof of un-ambiguity

def $y \sqsubseteq x$ prefix $\underbrace{\quad x \quad}_{\exists z}$ $y \sqsubseteq x$ for $y = \varepsilon / x = y$ included : proper prefix $\varepsilon \neq y \sqsubsetneq x$

lemma ① $E \overset{*}{\Rightarrow} x \in \Sigma^*$ $\Sigma = \{a, b, c, \#, \ldots, (,)\}$
then $|x| \geq 1$ & $N_c(x) = N_)(x)$ & $\forall y \sqsubseteq x$ $N_c(y) \geq N_)(y)$ ⇒ left ≥ right

② $T \overset{*}{\Rightarrow} x \in \Sigma^*$
then $|x| \geq 1$ & $N_c(x) = N_)(x)$ embedded in sequence
unique : $\forall y \sqsubseteq x$ $N_c(y) \geq N_)(y)$ & if $x = \sqcup + \sqcup$ then $N_c(y) > N_)(y)$

③ if $F \overset{*}{\Rightarrow} x \in \Sigma^*$
unique : if $|x| \geq 1$ $\forall y \sqsubset x$ $N_c(y) > N_c(y)$

And in all 3 cases, x doesn't start so end in * and +

proof : by induction on i ⇒ (steps of derivation)
$F \overset{i}{\Rightarrow} x$ if $|x| \geq 1$
first step must be $F \Rightarrow (E) \overset{i-1}{\Rightarrow} (y) = x$
by induction what could be produced in E is balanced
$T \overset{i}{\Rightarrow} x$ if $|x| \geq 1$
T can go $T \times F$ or F .
↳ ....

Theorem: This G_1 is unambiguous

proof: Let $T \overset{*}{\Rightarrow} \wedge$, if $|x| = 1$ $F \to a \mid b$ game over. unique
suppose $|x| > 1$ must use $F \to (E)$ front step: $F \Rightarrow (E) \overset{i-1}{\Rightarrow} (y) = x$ & suppose $F \overset{j}{\Rightarrow} x$ ∴ unique
Let $T \overset{*}{\Rightarrow} x$ ($T \overset{i}{\Rightarrow} x$) if $|x|=1 \to T \Rightarrow F \overset{i-1}{\Rightarrow} x$ (a, b) if $|x| > 1$, $T \Rightarrow T * F \overset{i-1}{\Rightarrow} x$ = $\underbrace{\quad y * x \quad}_{\exists z}$ y: proper prefix, $N_c y > N_) y$
$\Rightarrow F \overset{i-1}{\Rightarrow} x$ ↳ but it is from T, $N_c y = N_) y$

contradiction ⇑

Pushdown automata like NFA with stack

- can write symbols on stack and read them back later
- stack has unlimited space
- Deterministic and nondeterministic pushdown automata are not equivalent in power.

A pushdown automata is a 6 tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where Q, Σ, Γ, F are finite sets.

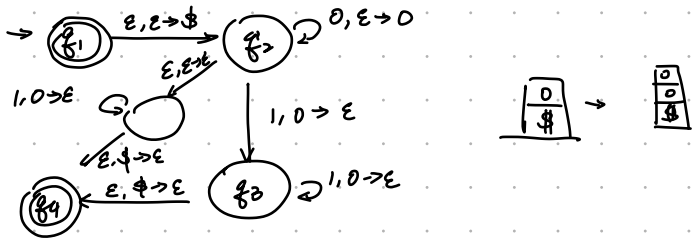
1. Γ is stack alphabet · stack is a finite string but with unlimited space.

$\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to P(Q \times \Gamma_\varepsilon)$

push: $\delta(q, a, \varepsilon) \Rightarrow (q', B)$ pop: $\delta(q, a, A) \Rightarrow (q', \varepsilon)$

A language is CFL iff some pushdown automata recognize it

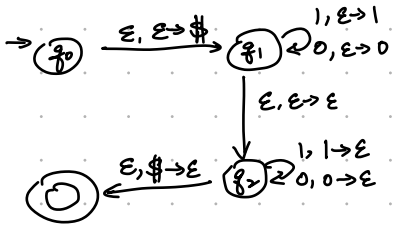
PDA for $\{0^n 1^n \mid n \geq 0\}$



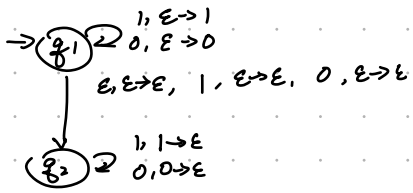
Def L(P): $\{w \in \Sigma^* \text{, s.t. } \exists \text{ a sequence of legal move from } q_0 \text{ consuming all symbols of } w \text{ \& end in } q \in F\}$

↳ or, end up with an empty stack.

$\{ww^R \mid w \in \{0,1\}^*\}$



$\{w 0 w^R \mid w \in \{1\}^*\}$



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

$S \to S_1 \mid S_2$

$S_1 \to S_1 c \mid S_1'$

$S_1' \to a S_1' b \mid \varepsilon$

$S_2 \to a S_2 c \mid S_2'$

$S_2' \to S_2' b \mid \varepsilon$

Chomsky Normal Form

In a c.f.g. G of every production is of the form

$A \to BC$ $S \to \varepsilon$ A, B, C are not necessary distinct

$A \to a$

Define a graph where vertices are variables

A → B put arrow A → B

↳ find  no cycle

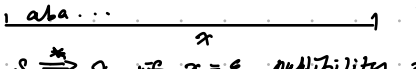
If a variable A can produce ε, A is called nullable. $A \overset{*}{\Rightarrow} \varepsilon$

Decide the set of all $V' \subseteq V$ s.t. $A \in V' \Leftrightarrow A$ is nullable.

- Given a grammar G & $x \in \Sigma^*$, whether $x \in L(G)$ decidable ⇒ PDA is not deterministic

$L(G) = L(G')$ G' in Chomsky N. F

Dynamic program algorithm:



# of contigous subsequence : $1 \leq i \leq j \leq n$

$S \overset{*}{\Rightarrow} x$ if $x = \varepsilon$ nullibility of S ⇒ decidable

Suppose $x \neq \varepsilon$ $n = |x| \geq 1$

n rounds : $j = 1, 2, \ldots n$

length j contigous subsequence

at j-th round, I'll connect all variables that can derive x_i s $x_{i+1} \ldots x_{i+j-1}$

for i = 1 to n do: $V_{i,1} = \{A : A \to a \in P \text{ \& } i\text{th symbol of } x \text{ is } a\}$

for j = 2 to n do:

$S \to AB \mid BC$

$A \to BA \mid a$

$B \to CC \mid b$

$C \to AB \mid a$

```
for i = 1 to n-j+1
  begin Vij = ∅
  for k=1 to j-1
    break up all possibilities of substringing length < j see if there is a rule
    Vij = Vij ∪ {A | A→BC ∈ P & B ∈ Vik}
```

? → BA, BC ⇒ ba

j \ i	1	2	3	4	5
1	B	AC	AC	B	AC
2	A,S				
3	∅	ac, B		B	

B

BC

SA

∅ SAC

∃ AC s is there $x \in L(G)$

1. Union $L(G_1) \cup L(G_2)$ is CFL
2. $L(\bar{G})$ is generally not CFL ↪ de Morgan's law
3. Intersection is not generally CFL ⇒ 2 stack might be conflict, example: $a^n b^n c^n$ is not CFL
4. Concatenation is CFL ⇒ add a new start variable
   ↳ * is also CFL
5. if G is c.f.g and M is DFA, $L(G) \cap L(M)$ is c.f.l. M doesn't have a stack
6. ↓ (set diff) $L(G) \setminus L(M)$ is c.f.l ⇒ $L(G) \cap \overline{L(M)}$

$L(G) \setminus L(M)$: $L(G) \cap \overline{L(M)}$

proof: we can construct a PDA, the PDA will process input and track states of DFA & follow stack operations of PDA

Pumping Lemma to CFL

for every cfl L ∃ an integer $n \geq 1$, for every $x \in L$ $|x| \geq n$

∃ a decomposition of $x = uvwyz$ ① $|v,y| \neq 0$ ② $|vwy| \leq n$

$\forall i \geq 0$ $uv^i w y^i z \in L$

if $\{a^n b^n c^n\}$ c.f.l

∃ $N \geq 1$ satisfy that PL. for cfl L. and yet condition $a^N b^N c^N \in L$ had length $\geq N$

for any decomposition $a^N b^N c^N = uvwyz$ have $|v.y| \geq 1$ $|vwy| \leq N$

↳ each part of v & y cannot contain more than one type of symbol, else $i = 2$ gives $x \notin a^* b^* c^*$

v and y contains at most 2 type of symbols

↳ at least 1 type is not present, uv^2wy^2z: the missing one type is not increasing, but the other 2 are increasing

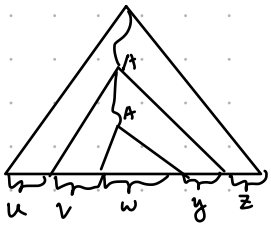
prove pumping lemma

Assume $L = L(G)$ G in CNF

A is a child of A (appear in a path)
↳ if path length is > # of variable
↳ the path exist if leaf is big enough

By G being CNF, the tree is a binary tree while every "internal node" has a children (>1 step from leaf)
if # of leaves > $2^{|V|}$ → # of variable ⇒ some path from root to just above leaf have length > # of variable

some variable appears at least twice



v and y cannot be both ε as $|v,y| \neq 0$

proving $|vwy| \leq n$ since there is same variable appearing twice under "A"

Turing Machine

A TM is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$, Q, Σ, Γ are all finite sets

1. Q is the set of states
2. Σ is the input alphabet, no $\sqcup$
3. Γ is the tape alphabet, $\sqcup \in \Gamma$, $\Sigma \subseteq \Gamma$
4. $\delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
5. q_0 is the start state $q_0 \in Q$, q_{acc} is accept state q_{rej} is reject state, $q_{acc} \neq q_{rej}$

- C_1 yields C_2 if C_1 goto C_2 in 1 step.
  - left move: uaq_ibv yields uq_jacv on $\delta(q_i, b) = (q_j, c, L)$
  - right move: uaq_ibv yields $uacq_jv$ on $\delta(q_i, b) = (q_j, c, R)$

  if on the head: left: q_ibv yields q_jcv

  right: $bvq_i = bvq_i\sqcup$ yields $bvcq_j$ on $\delta(q_i, \sqcup) = (q_j, c, R)$

ID IR⊢ ID₂

↳ $\Gamma^* Q \Gamma^*$

tape is one-side infinite

uq_iac has next step iff $u \neq \varepsilon$

↓

$u'd$, u' could $= \varepsilon$, $d \in \Gamma$

$\vdash^*$: 0 or more step

- Turing Machine have finite input with infinite tape length (the # of '$\sqcup$' at right is infinite)

Turing Machine accepting: $L(M) = \{w \in \Sigma^* \mid q_0 w \sqcup\sqcup / q_0 w \vdash^* u\, q_{accept}\, v$ for some $q_{acc} \in Q_{acc}\}$

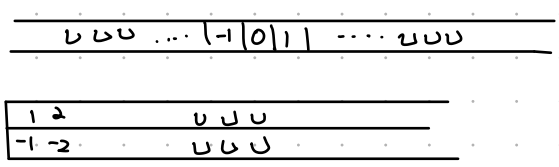
Not finite definition:

A set S is NOT finite iff there is 1-1 correspondence of S & a proper subset $S' \subsetneq S$

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

$\{*, 3, 4, 5, \dots\}$

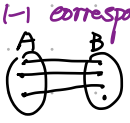
- For a turing machine, one-way finite is the same as 2 way finite



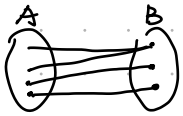
- just adjust the alphabet

Any set S that there is a 1-1 correspondce with $\mathbb{N}$ is called countable infinite

- into: $f: A \to B$



onto



- cardinality: size of a set

A set is countable if it's either finite or has the same cardinality with $\mathbb{N}$

Uncountable infinity cannot put into 1-1 correspondence with set of $\mathbb{N}$ - "too large"

- the set $\mathbb{R}$ is uncountable ⇒ prove by diagonal

for a set $\mathbb{R}$ we can have a list of all r in $\mathbb{R}$ in range $[0, 1]$

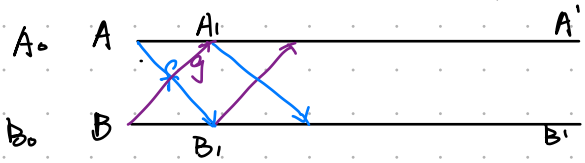
r_1 0. 3 4 5 ... alter the bit on diagnoal line and form a new number

r_2 0. 4 7 8 ...

r_3 0. 2 5 1 ...

r_4 0. 1 7 9 3 ...

Cardinality : $\exists$ 1 - 1 correspond A & B $|A| = |B|$



$B_{i+1} = f(A_i)$

$A_{i+1} = g(B_i)$

$A' = \bigcap_{i \geq 0} A_i$ if $A_1 = A_0$ g does $\exists$ $A_1 \neq A_0$ $B_1 \neq B_0$

$B' = \bigcap_{i \geq 0} B_i$ $B_1 = B_0$ f does $\exists$

Suppose $x \in A'$ so $\exists$ must be in one of A_i

For $x \in A = A_0$ if $x \notin A'$ $\exists$ max$i$ $x \in A_i - A_{i+1}$ $x \neq A_{i+1}$

if x is even, do f : map x under f to go forward

if x is odd, do g^{-1} go backward

if $x \in A'$, take $f(x) \to B'$

- say a set has linearly / total order (for $\exists x \neq y$, either $x<y$ or $x>y$)

A linear order is called a well-order if following are true

- every non-empty set $S \subseteq A$, $S \neq \emptyset$, S has a minimumelt. ⇒ must have a smallest / base case

↳ not forall $\mathbb{N}$, but for all positive number, also for rational number → $(0,1)$ not work but $[0,1]$ works

order $0, 1, 2, 3 \ldots \omega$

$1+\omega = \omega$, $\omega+1$, $\omega+\omega$, ω^{ω} .....

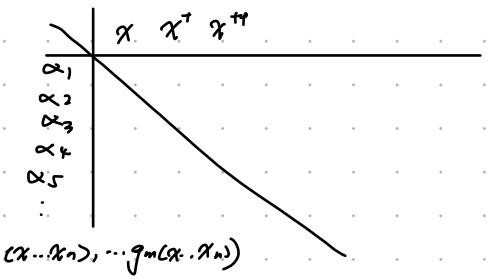
$\beta_0(x) = x+1$ $\beta_1(x) = 2+x$ $\beta_2(x) = 2 \cdot x$ $\beta_3(x) = 2^x$ $\beta_4(x) = 2^{2^{2^{\cdot^{\cdot^{\cdot}}}}} 3$ $\beta_5(x) = 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}$ (x 2's)

$\beta_1(x^+) = \beta_0(\beta_1(x))$ $\beta_2(x^+) = \beta_1(\beta_2(x))$ .... $\beta_4(x^+) = \beta_3(\beta_4(x))$ ... induction

↗ permutatively recursively function

Acumen function

↳ $A(x) = \beta_x(x)$ → diagnol function



Primitive recursive functions

1. $\lambda x.\,[x+1]$ success function x^+
2. $\lambda x_1 \ldots x_n\,[m]$ $\forall n, m$ inf
3. $\lambda x_1 \ldots x_n\,[x_i]$ $\forall n \geq 1$ $1 \leq i \leq n$ "identity function"
4. Composition if $g_1 \ldots g_n$ h are "primitive" then so is $f(x_1 \ldots x_n) = h(g_1(x_1 \ldots x_n), \ldots g_m(x_1 \ldots x_n))$

Acumen is not captured by scheme table ⇒ we cannot enmerate all computable things

↳ but Turing machine are countable. Total functions will never work ⇒ DIAG on primitive recursion

so Church add μ operator

$f(x_1, \ldots x_n) = \mu z$ (minimum z such that) $[g(x_1 \ldots x_n, z) = 0]$

find smallest counter example. if not exist, z is undefined (will not halt)

Thm: the set of (partial) recursive functions. using 1-6 is exactly the same as Turing Machine computable.

Turing Decidability & Recognizability

Σ^* : set of finite length string, total # of string infinite.

Def a subset $L \subseteq \Sigma^*$ is Turing recognizable r.e.

- $\exists$ TM. M s.t. $L = L(M)$, for $w \notin L$, M will NOT ACCEPT (might reject, might loop)

↳ for L^c is complement of L, L^c either reject or loop in M, so we wont know if $L(M') = L^c$

Def a subset $L \subseteq \Sigma^*$ is Turing decidable (recursive)

$\exists$ TM. M "total" $L(M) = L$ if on every input $w \in \Sigma^*$, $M(w)$ either { halt on q_{acc} / halt on q_{rej} } in finite steps.

Thm: L is recursive iff both L and L^c are r.e

proof: $\Rightarrow$ we can have a decier for L $\Rightarrow$ implies L is r.e

flip the accept and reject $\Rightarrow$ L^c is r.e.

$\Leftarrow$ for M_1, $L(M_1) = L$ M_2, $L(M_2) = L^c$

Construct a M_3: on input w

run one step on M_1 and one step on M_2

if M_1 accept, accept

if M_2 accept, reject

- TM can be encoded uniquely into integer by ascii code.

M_i [q_1] # [q_2] .... $\Rightarrow$ integer

↳ you can enumerate all TM code of M_i $\langle M_i \rangle$

	ε	0	1	2	3	4	...
M_1		rej					
M_2				acc			
⋮							

- $A_{TM} = \{\langle M_i, w \rangle \mid M_i \text{ accept } w\}$

Assume A_{TM} is decidable, TM M^* is decider

① Design a TM M on input $x_i = \langle M_i \rangle$

feeds $\langle M_i, \langle M_i \rangle \rangle$ to M^*, if { M^* accept, M loop / M^* reject, M accept }

② When M on input $\langle M \rangle$, what happens:

↳ feed $\langle M, \langle M \rangle \rangle$ to M^*, if { M^* accept, M loop $\Rightarrow$ but $\langle M, \langle M \rangle \rangle \in A_{TM}$, M should stop / M^* reject, M accept $\Rightarrow$ but $\langle M, \langle M \rangle \rangle \notin A_{TM}$, M should loop and never halt }

↳ contridiction.

- Church-Turing Thesis: Anything that can be computed is computable by TM. Also state the limitation.
  - Turing Machine algorithm = λ calculus functions
  - Defines intuitive notion of algorithms

Examples: $A_{DFA} = \{\langle B, w \rangle \mid B \text{ is DFA accept } w\}$ A_{DFA} decidable

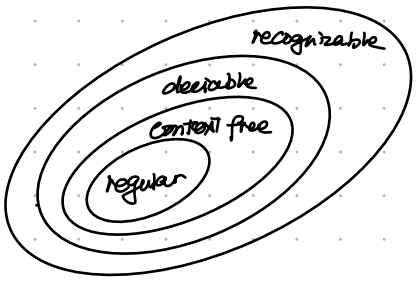
CFG is decidable. proof: for Chomsky form, any derivation of w within $2n-1$ step

↳ for decider D, D run every derivation within $2n-1$ step.

if $n = 0$, list all derivations with one step

$E_{DFA} = \{\langle B \rangle \mid B \text{ is a DFA } L(B) = \emptyset\}$ · mark stated, make all incoming link from marked state, check if q_{acc} marked

Undecidability



A_{TM} is not decidable ⇒ partial decider exists. r.e but not recursive

- HALT problem

$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ on } w \text{ eventually halts}\}$

Thm: $HALT_{TM}$ is not recursive / decidable.

proof: suppose we can have a decider D for $HALT_{TM}$
then we can construct a decider M' for A_{TM}

M' = on input $\langle\langle M\rangle, x\rangle$,
run D on $(\langle M\rangle, x)$, if reject, reject
if accept, run M on x.

↳ if we can decide $HALT_{TM}$, we can decide A_{TM} ⇒ contradiction

- $\bar{A}_{TM}$ is not recognizable otherwise A_{TM}, $\bar{A}_{TM}$ both r.e. A_{TM} will be recursive.

Reducibility: A reduction is a way of converting one problem to another problem in such a way that a solution to the second problem can be used to solve the first problem.

$E\emptyset$ is undecidable: suppose $E\emptyset$ is decidable and have decider D, we can have following

TM M_1 = "on input x:
if $x \neq w$, reject; otherwise run M on w"

Then we can construct a decider for A_{TM}:

M' = "on input $\langle\langle M\rangle w\rangle$:
run D on $\langle M_1\rangle$, if accept, reject, if reject, accept."

Since M_1 only accept w if M accept w, if M don't accept w $M_1 \in E\emptyset$

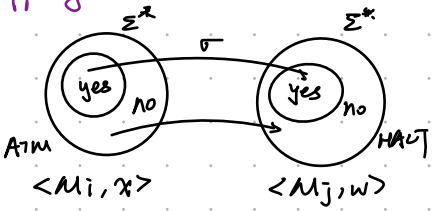
$REGULAR_{TM} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$

prove: suppose $REGULAR_{TM}$ is decidable with D. we can construct following
then we can construct a M' for A_{TM}

M' = on input $(\langle M\rangle, w)$, construct M_2 = "on input x, if x is in form 0^n1^n, accept, otherwise run M on w"
if D accept M_2, accept.

if M accept w then M_2 accept all other strings.

Mapping Reduce:



σ: a computable function in finite step
M_j: on input w, ignore w
Run x on M_i, if accept, halt
if loop/rej, loop

⇒ if you can decide HALT you can decide A_{TM}

Rice's theorem

$\{\langle M_i\rangle \mid P(L(M_i))\}$ is undecidable for non-trivial property of recursively enumerable set

- P is about the language but not TM.

proof: P on r.e set is non-trivial iff $\exists M_1, M_2$ s.t. $P(L(M_1))$, $\neg P(L(M_2))$

Suppose M_1 s.t. $L(M_1) = \emptyset$ and $\emptyset \notin P$
M_2 s.t. $L(M_2) \in P$

$\bar{P}$ is not decidable: if $\langle M_i, w\rangle \in A_{TM}$ then $L(M_{\text{construct}}) = L(M_j) \in P$
$\notin A_{TM}$ then $L(M_{\text{construct}}) = \emptyset \notin P$

proof Rice's theorem:

① Define P
② Prove P
③ Prove if $L(M_1) = L(M_2)$ they both $\in P$

note: if P is trivial ⇒ might be decidable

P is expressed by PCP ⇒ r.e., trivial P
$\bar{P}$: not r.e. ⇒ trivial P

Post Correspondence Problem

Given input $\{[\frac{t_1}{b_1}], [\frac{t_2}{b_2}] \dots [\frac{t_n}{b_n}]\}$ $t_i, b_i \in \Gamma^*$, question: is there a match?

PCP is undecidable : we need to prove this by introducing MPCP

Modified PCP · same instance , question : is there a sequence must start on the first $i_1 \dots i_k \in \{1, \dots n\}$ s.t. $i_1 = 1$ & $t_{i_1} \cdots t_{i_k} = b_{i_1} \cdots b_{i_k}$

Proof : $A_{TM} \leq$ MPCP , we can construct MPCP from $M, w \Rightarrow$

$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$

M will work as follows : for input from $\langle M, w \rangle$

1. $[\frac{t_1}{b_1}] = [\frac{\#}{\#q_0 w_1 \dots w_n \#}]$
2. $\forall a, b \in \Gamma$ $\forall q, q' \in Q$ $q \neq q_{rej}$, if $\delta(q, a) = (q', b, R)$, then put $[\frac{qa}{bq'}]$ in P , put one $\exists a, b$
3. $\forall a, b, c \in \Gamma$, $\forall q, q' \in Q$, $q \neq q_{rej}$, if $\delta(q, a) = (q', b, L)$, then put $[\frac{cqa}{q'cb}]$ in P, put on $\exists c$, for $\exists a, b$
4. $\forall x \in \Gamma$, put $[\frac{x}{x}]$ in P $\Rightarrow$ everything in Γ
5. put $[\frac{\#}{\#}]$ in P
6. if the turing machine reach q_{acc}, put $[\frac{a\, q_{acc}}{q_{acc}}], [\frac{q_{acc}\, a}{q_{acc}}]$ $\forall a \in \Gamma$
7. $[\frac{q_{acc}\, \#\#}{\#}]$

To prove MPCP $\leq$ PCP if PCP solved then MPCP also

let $[\frac{t_i}{b_i}]$ be instance in MPCP

$t_1 = abcd \dots d \Rightarrow *a*b*c*d* \dots *d$

$b_1 = abcd \dots d \Rightarrow *a*b*c*d* \dots *d*$

def $*t$: put $*$ in front of any element in t

$t*$: put $*$ in front of any element in t , also right after

instance of MPCP $\rightarrow \{\frac{*t_1}{*b_1*}, \frac{*t_i}{b_i*}\}$ $i \geq 1$ $\Rightarrow$ the only solution must start with $\frac{*t_1}{*b_1*}$

- Turing decidable is closed under
  - union · concat · star · complementation · intersection

Recognizable closed under

- union : execute M_1, M_2 individually $\Rightarrow$ at least one accept
- concat : divide each input into 2 part non-deterministically and run
- star : divide w in non-deterministically into $w_1 \dots w_n$
- intersection : execute both
- homomorphism : on inputs consider all w $h(w) = s$, run M on w , using merging to interleave with other M
- if $A \leq_M B$, B is regular , A could be not regular
- E_{TM} is not reducible to A_{TM} since $\overline{A_{TM}}$ is not r.e. if $E_{TM} \leq_P A_{TM}$ then $\overline{E_{TM}} \leq_P A_{TM}$
- if A r.e. $A \leq_M \overline{A}$, then A is decidable
- A r.e. iff $A \leq_M A_{TM}$
- $A_{TM} \leq_M \overline{A_{TM}}$
- if $A \leq_M B$. A is not r.e. B is not r.e.
- CFG $\Rightarrow$ PCP to prove $L(G) \cap L(H) \neq \emptyset$

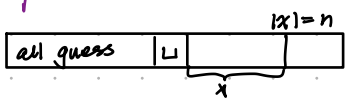
P : $L \in P$ iff $\exists$ TM on any input x, $|x| = n$, M on x halts in $\leq p(n)$ steps where p is some polynomial

$x \in L$ $L = L(M) \Rightarrow$ M halts and accept on x in $P(|x|)$ steps

$x \notin L$ $L = L(M) \Rightarrow$ M halts and reject on x in $P(|x|)$ steps.

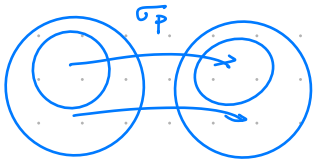
(poly time recursive)

NP : $\exists$ TM M with 2 way infinite type such that $\exists$ input x, $|x| = n$. if $x \in L$, $\exists$ a guess to let me verify that M accepts x (a guess is an assignment encoded in each cell to the left cell in TM M)
the guess y, $|y| \leq p(n)$ s.t. $M(y \sqcup x)$ are in $P(n)$ stop and accept.
if $x \notin L$. $\forall$ guess y, $|y| \leq p(n)$ $M(y \sqcup x)$ are in $P(n)$ stop and reject.



· NP : can be verify in poly-time ; poly-time version of r.e.

$\leq_M^P$ is poly-time version



NP-Complete : A decision problem is NP-Complete if $L \subseteq \Sigma^*$

① $L \in NP$

② $\forall$ NP set L' $L' \leq_M^P L$

SAT $x_1, x_2 \ldots x_n$ variables $C_1 C_2 \ldots C_n$ each C_i is a clause

↳ disjuction $\vee$ with variable / negation is there an assignment of $x_1, x_2 \ldots x_n$, it takes $x_1 \ldots x_n \to \{0,1\}$
that every clause satisfy $\Phi|_f = C_1 \wedge C_2 \wedge \ldots C_m|_f = \text{True}$
Φ is a conjuction of disjuction of literals variable or its negation

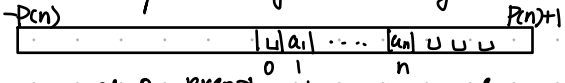
Cook-Levin : every NP set $L(M)$ is reducible p-time to SAT

proof : a poly-time reduction for every $L(M)$ to SAT

$M = (Q, \Sigma, \Gamma, \sqcup, q_{acc}, q_{rej}, \delta)$ x, $n = |x|$ $x = a_1 a_2 \ldots a_n$ $a_i \in \Sigma$

variables : In time stamp i, are you in state j (true or false)

time stamp 0 :



time stamp 1 : same as 0, except not in q_0 $\to$ $\delta(q_0, a_1) = (q', a', L|R)$

⋮ in $\leq P(n)$ steps

accept : after accept, all states goes to accept state

variable Q_{ij} $0 \leq i \leq p(n)$ $0 \leq j \leq r$ $Q = \{q_0, q_1, q_2, q_3 \ldots q_r\}$ (q_1 = acc, q_2 = rej)

$|Q| = r+1$ $|\Gamma| = s+1$ $\Gamma = \{a_0 = \sqcup, a_1 \ldots a_s\}$

① Q_{ij} = true iff at time i, M is at state j

② H_{ij} → at time i, where is my leading head (on tape) $-p(n) \leq j \leq P(n)+1$

③ A_{ijk} at time i, at time j, contain same symbol a_k $k = 0 \ldots s$

Constrains: $Q_{i0} \vee Q_{i1} \ldots \vee Q_{ir}$ is true $\Rightarrow$ you must be in a state at time i

and $\neg(Q_{i0} \wedge Q_{i1})$ is true → $\overline{Q_{i0}} \vee \overline{Q_{i1}} \ldots \overline{Q_{i0}} \vee \overline{Q_{ir}}$ $\overline{Q_{i1}} \vee \overline{Q_{i2}} \ldots \overline{Q_{i1}} \vee \overline{Q_{ir}}$ $\Rightarrow$ for $\exists$ $\overline{Q_{ij}} \vee \overline{Q_{ij}}$ is true ($0 \leq i \leq P(n)$) ($0 \leq j < j' \leq r$) $[\overline{Q_{ij}} \vee \overline{Q_{ij'}}]$ is true

↳ you are in exactly one state at a time

and

$H_{i,-p(n)} \vee \ldots H_{i,p(n)+1} \Rightarrow$ at any time i, in one of the leading position (on tape)

$\bigwedge_{-p(n) \leq j \leq j' \leq p(n)+1} [\overline{H_{ij}} \vee \overline{H_{ij'}}]$ ⋮ at all time, all cells contains exactly one symbol of Γ (same apply to A)

↳ after all, for input $a_i, a_{i+1} \ldots a_{in}$ $a_{ig} \in \Sigma$, the cell position $A_{0,1,1}$ at time 0, cell 1, have a_{i1} symbol

$A_{0,i,i_2} \wedge \ldots A \ldots A_{0,n+1,0} \wedge A_{0,0,0} \wedge \ldots A_{0,n,i_n} \ldots$

Assign $A_{0,-1}, \ldots A_{0,-p(n)}$ to make $\wedge Q_{p(n),1} \wedge Q_{0,0}$ → If at time i, not in j, $\overline{H_{ij}} \wedge A_{ije} \to A_{i+1,j,e}$

$[H_{ij} \vee \overline{A_{ij,e}} \vee A_{i+1,j,e}]$

if $x \in L$, $\exists$ an computation of M on x of length $p(n)$ or less, satisfies all the clause in C = all restrictions

- $L \in NP$ iff $\exists$ p-time predicate A, $x \in L \Leftrightarrow \exists y, |y| \le poly(|x|)$, $A(x,y)$ holds P: problem solvable in poly-time

Decision problem: a decision problem consists a set of instance D and a subset $Y \subseteq D$ of yes-instances.

SAT is NP-Complete: every problem in NP is poly-time reducible to SAT

① SAT $\in NP$

② $\exists L \in NP$, $L \le_M^P SAT$

- A generic NP set $\le_M^P$ SAT $\le_M^P$ 3SAT

3SAT: SAT that accept every clause has 3 literals

proof: convert SAT to 3SAT

$k=1$ for $C=\{x_1\}$ $C' = (x_1 \vee y_1 \vee y_2) \wedge (x_1 \vee \bar{y_1} \vee \bar{y_2}) \wedge (x_1 \vee \bar{y_1} \vee y_2) \wedge (x_1 \vee y_1 \vee \bar{y_2})$

$k=2$ for $C=\{x_1 \vee x_2\}$ $C' = (x_1 \vee x_2 \vee \bar{z}) \wedge (x_1 \vee x_2 \vee \bar{z})$

$k>3$ for $C=\{x_1 \vee \bar{x_3} \vee x_4 \dots\}$ $C' = (x_1 \vee \bar{x_3} \vee z_1) \wedge (\bar{z_1} \vee x_4 \vee z_2) \wedge (\bar{z_2} \vee x_5 \vee \bar{z_3}) \dots$

∴ 3SAT we can have a poly-time algorithm to solve it.

Reducing decision problem to optimizing problem: use the decision problem as a oracle, according to input size do binary search and run decision problem.

- $A \le_P^M B$ ⇒ if we have a poly-time algorithm decide B, then we can have a … decide A

$SAT \le_P^M$ 3SAT $\le_P^M$ Vertex Cover $\equiv_P^M$ IND $\equiv_P^M$ Clique

$\le_P^M$ 3-D Matching $\le_P^M$ 3XC $\le_P^M$ XC

$\le_P^M$ HAM $\le_P^M$ TSP

- P is closed under union, concatenation, complement, star, intersection

Union: check both decider and if either accept

concat: separate input in n ways and run both decider

complement: reverse accept / reject

star: on input assume $y_1, y_2 \dots y_n \in \Sigma$

1) if $y = \varepsilon$ accept and halt
2) init $T[i,j] = 0$ for $i \le j$
3) for $i = 1$ to n
   - Run M on y_i, if $y_i \in L$ set $T[i,j] = 1$
4) for $k = 2$ to n, for $i = 1$ to $n-k+1$
   a) $j = i+k-1$
   b) Run M on $y_i \dots y_j$ and set $T[i,j] = 1$
   c) for $l = i$ to $j-1$, set $T[i,j] = 1$ if $T[i,j] = 1$ and $T[i,l] = 1$

Accept if $T[1,n] = 1$

- NP is closed under union and concatenation, star, intersection

union: verify in both verifier

concatenation: split S in $S = S_1 S_2$ O(n) on non-deterministic decider

star: for each $w = w_1 \dots w_k$ non-deterministically guess certificate, verify all.

- if P = NP then every $A \in P$ except $A = \emptyset$ and $A = \Sigma^*$ is NP-Complete.
- $A \notin P$ and $B \in P$ A is not necessarily $\le_P^m$ B as B could be trivial and A non-trivial

Vertex-Cover

input: $G = (V, E)$, k

question: is there a subset $S \subseteq V$, $|S| \leq k$ s.t. $\forall e = \{x, y\}, e \in E$, either x or $y \in S$

from $3SAT \leq_p^M VC$

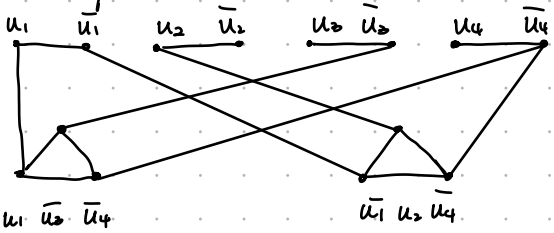
for the instance of 3SAT, $X = \{x_1, x_2, \dots x_n\}$ $C = \{c_1, c_2, \dots c_m\}$

create following graph:

for each variable x_i, have (x_i) —— $(\bar{x}_i)$, S need to contain at least one of $x_i, \bar{x}_i$

for each clause c_i, have a triangle  representing 3 literals

connect corresponding variable



to have a VC, must have 2 edge in 1 triange and a out edge (must be 1 true in 1 clause)

↳ $2m + n = k$, the resulting Graph have $3m + 2n$ vertex, $k = 2m + n$

Independent Set

input: $G = (V, E)$, k

question: is there a subset $S \subseteq V$, $|S| \geq k$ s.t. $\forall e = \{x, y\} \in E$, x, y don't both in S

from $VC \leq_m^P IND$

for the same graph, find a VC for $|S| - k$, take $V - S$ of VC $\Rightarrow$ IND

Clique

input: a graph $G = (V, E)$, k

question: Is there a clique of at least size k; clique: $S \subseteq V$ is called a clique if every vertices of S, all edge exist

every vertex is connected to every other vertex by edge

$IND \leq_p^M$ Clique

create a new graph and reverse the edge, find IND.

- A Matching $M \subseteq E$ st. no 2 edges are incident (share a vertex)

for $|X| = |Y| = n$ $G = (X \cup Y, E \subseteq X \times Y)$ A perfect matching is $|M| = n$

for $\forall\ S \subseteq X$ (left side) $T(S)$ be their neighbors $= \{ y \in Y,\ (S, y) \in E$ for some $s \in S \}$

The problem of 2DM can be solved $\Rightarrow$ if G has a perfect matching

if G has a perfect matching, then $|T(S)| \geq |S|\ \forall S \subseteq V$, if $S \subseteq V$ $|T(S)| < |S| \rightarrow$ not matching

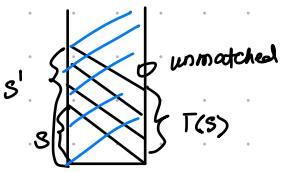
proof: Augmenting path: start / end with non-matching edge $\Rightarrow$ 2 vertex not matched $\Rightarrow$ increase matching size by exchanging.

we can give a P time algorithm to find a PM, prove there is no PM for $|T(S)| < |S|$

① start with a perfect matching M (initially $\emptyset$)

② enumrate all unmatched in left sets and neighbors $T(S)$ matched

the algorithm shall halt in n vertices.



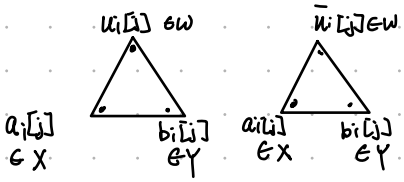
3DM

Input: X, Y, W $|X| = |Y| = |W| = n$ & a set of triples (x, y, w) $x \in X$ $y \in Y$ $w \in W$

question: is there a subset T' of T s.t. every $v \in X \cup Y \cup W$ has exactly one of them. $|T| = n$

prove $3SAT \leq_p^m 3DM$

design the gadget

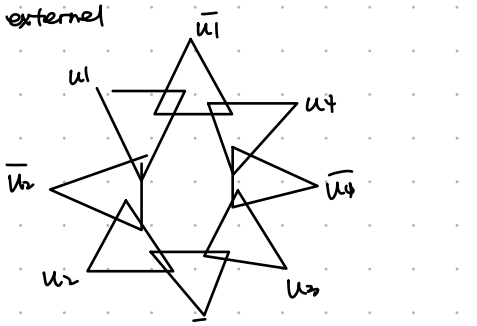


have $m \times 2$ gadget

$N = nm - m$

bottom 2 internel
top 1 external

$1 \leq j \leq m$ ← # of clause



3XC: Exact Cover

input: a finite set X with $|X| = 3q$ and a collection C of 3 element subsets of X

question: does C contains an exact cover for X, a subcollection $C' \subseteq C$ st. $\forall X$ occur exactly one member of C'.

regard 3DM as unordered subset of $W \cup X \cup Y$, matchings in 3DM $\rightarrow$ XC

HAM: Hamiltonian circuit / path

Input : a graph $G = (V, E)$

A HAM circuit $(v_{ij}, v_{ij+1} \in E,\ i \le j < n,\ n+1 \equiv 1\ (n))$ is a cycle that visits every vertex exactly once and return to the beginning

A HAM path is the circuit without going back to beginning

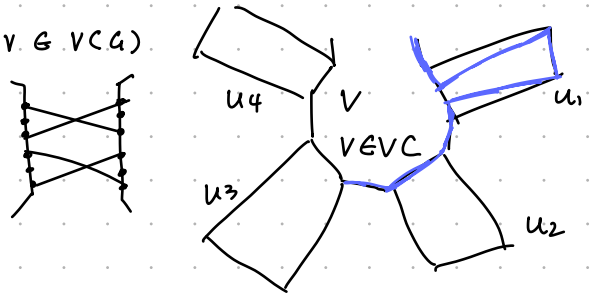
Eulerian circuit : instead of traveling vertex, traverse all edge. exact once.

↳ iff { $\forall v \in V$ degree(v) = even ⇒ has in-out pair
is a fully connected graph

↳ P-time solvable : just go on as you go, when you stuck at some place ⇒ must be odd
if finished to start without getting all edges: revise your step!
↳ the place you haven't visit is also a Eulerian circuit.

prove $VC \le_p^m$ HAM-C

- Start with a VC instance $G = (V, E)$, k
- Construct a graph G' every vertex get replaced with



each tour visit some part of the graph
(inside a VC)

if $u_1 \notin VC$
$u_2 \in VC \Rightarrow$ the edge will be take care by VC

The resulting graph ⇒
have k selected vertex s_1 s_2 ... s_k
s_1 ... s_k have connection into the chain
↳ k round of bus tour.

s_1 ⟶ selected vertex cover v entry → travel around
⟶ exit from v and start from s_2 ⇒ ....
⟶ start at s_k → exit at s_k → go to s_1

TSP:

Input : $G = (V, E)$ & $w : E \to \mathbb{R}_+$, k

question : start from some vertex, and visit every vertex at least once and return to the starting vertex
s.t. weight $\le k$ / total weight is minimized

HAM-C $\le_p^m$ TSP

- If HAM-C exist, TSP ⇒ has minimum.

For a given TSP graph, set all $e \in E$ as $w = 1$, ask if ∃ TSP for cost n ⇒ give HAM-C

U-HAMPATH is also NP-Complete, HAMP $\le_p^m$ UHAMP

input : Directed graph G and start point s end point t

↳ for each $v \in V$ except s and t, replace by a triple $(v^{in}, v^{mid}, v^{out})$
$s = s^{out}$ $t = t^{in}$

① connect edges v^{mid} with v^{in} and v^{out}

② connect v^{out} with z^{in} if edge in G is $v \to z$

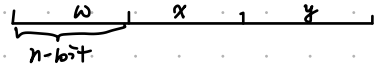
Subset Sum Problem

Input: A set of positive integers $S = \{n_1, n_2, \dots n_m\}$ & a target T

Question: is there a subset of S that sums to T (a positive integer)

↳ Input length: # bits to specify the problem $\sum_i^m \lceil \log(n_i) \rceil + \lceil \log(T+1) \rceil$

prove NP-Complete by 3DM $\leq_m^p$ Subset Sum



- Design number of $3n$ bits. Bits position labeled by elements $W \cup X \cup Y$
- each triple 3DM instance (w_i, x_j, y_k)

  let $N(w_i, x_j, y_k)$ = binary #, have exactly one at the positions correspond to w_i x_j y_k

  $= 2^{k-1} + 2^{n+j-1} + 2^{2n+i-1}$

↳ solution for 3DM: all element in w, x, y can be triples without overlaping

solution for Subset Sum $T = 2^{3n}-1$

↳ but, it wont handle carry in/out ⇒ we now add buffers

Now, consider binary #'s that are $3 \cdot n \cdot$ # of tuples.

n fields: each field have # of tuples many bit positions and one 1 in least significant bit in fields for 1 tuple.

Target: one 1 in each field (at least significant bit of level)

on the other side: if there is other than ↓ 1, it cannot go out of buffer

Attempt on poly-time algorithm: N is not n, N can be exp

- if SubSet Sum is in unary, then size of input = length
  - ↳ if binary, $\log x$ needed to represent input value x
  - ↳ compare to N in binary, unary is 2^N

↳ more than exponent time to reduce SubSet Sum to 3DM

↳ Unary SubSet ∈ P

① Compute Sum of input

② Check if the Sum of # $\geq S$, if no, reject

③ run DP on input