Automata

def: **Deterministic Finite Automata**

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

$Q :=$ finite set of states
$\Sigma :=$ finite alphabet
$\delta := Q \times \Sigma \to Q$, Transition Function
$q_0 :=$ start state
$F := \subseteq Q$, set of accepting states

def: **NFA**

Like DFA but $\delta := Q \times (\Sigma \cup \{\varepsilon\}) \to P(Q)$, where $P(Q) = \{s | s \subset Q\}$

def: **Regular Expressions**

RegEx of Σ is:

1. $e \in \Sigma$
2. ϕ, empty set
3. ε, empty string
4. $R = R_1 \vee R_2$
5. $R = R_1 || R_2$
6. $R = R_1^*$

def: **Context Free Grammars**

$G = (V, \Sigma, R, s)$

$V :=$ Variables
$\Sigma :=$ Terminals
$R :=$ set of rules on the terminal
$s \in V :=$ starting variable

def: **Pushdown Automata**

Alternative to CFL.
$P = (Q, \Sigma, \Gamma, \delta, q_0, F)$

$Q :=$ finite set of states
$\Sigma :=$ input alphabet
$\Gamma :=$ stack alphabet
$\delta := Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to Q \times \Gamma_\varepsilon$, transition function

def: **Turing Machine**

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

$Q :=$ set of states
$\Gamma :=$ alphabet of tape
$\Sigma := \subset \Gamma$ alphabet of input
$\delta := Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
$q_{acc} :=$ M accepts if it visits
q_{rej} M rejects if it visits

thm: **CFL-PDA equivalence**

A language is CFL iff $\exists$ PDA s.t $\mathcal{L}(P) = L$

Automata Language

def: **Language**

$\mathcal{L}(M) = \{\ x \in \Sigma^* \ \mid\ M(x) \text{ accepts}\ \}$

def: **Regular**

L regular iff $L = \mathcal{L}(M)$ for some DFA M.

def: **Equivalent Relation**

$x \sim_L x' \to \forall z \in \Sigma^*, xz \in L \Longleftrightarrow x'z \in L$

def: **Context Free Language**

L is CFL if $\exists$ CFG s.t $\mathcal{L}(G) = L$

def: **Chomsky Normal Form**

Every rule in the form:
$A \to BC$
$A \to a$

thm: **Regular Closure**

1. Closed under union
2. Closed under intersection
3. Closed under complement
4. Closed under concatenation
5. Closed under *
6. Closed under reverse

thm: **Pumping Lemma**

if L is regular, then $\exists p \in \mathbb{N}$ s.t ,
if $w \in L$ with length $\geq p$, then $w = xyz$ s.t :

1. $|y| \geq 1$
2. $xy^k z \in L$
3. $|xy| \leq p$

thm: **Myhill-Nerode**

L is regular iff the number of equivalency classes in $\tilde{L}$ is finite.
Number of equivalency classes equals the minimum number of states a DFA of L must have.

thm: **CFG**

if CFG G in normal form and $w \in \mathcal{L}(G)$, then w's parsing tree is of length $\leq 2n - 1$

thm: **Pumping Lemma for CFL**

if L is CFL, then $\exists p \in \mathbb{N}$ s.t
if $w \in L, |w| \geq p$, then $w = uvxyz$ s.t :

1. $|vy| > 0$
2. $|vxy| \leq p$
3. $uv^i xy^i z \in L, \forall i \in \mathbb{N}_0$

thm: **CFL closure**

1. if L_1 CFL and L_2 CFL, then $L_1 \vee L_2$ CFL
2. if L_1 CFL and L_2 regular, then $L_1 \wedge L_2$ CFL

Decidability

def: **Recursively Enumerable**

L is Recursively Enumerable if there exists a Turing Machine M such that $\mathcal{L}(M) = L$

def: **Decider**

M that halts at all inputs.

def: **Decidable**

L is decidable if there exists a decider M such that $L = \mathcal{L}(M)$.

def: **Enumerator**

TM has output tape and can only print.
M is enumerator for L if it only outputs all words in L.

def: $R\mathcal{E}$

Set of all Recursively Enumerable languages.

def: **Co-**$R\mathcal{E}$

Co-$R\mathcal{E} = \{ \ L \ | \ \overline{L} \in R\mathcal{E} \ \}$

def: R

Set of all decidable languages.

def: **Encoding** $\langle\rangle$

Can encode anything into a string of 1s and 0s.

def: **Computable Function**

f is computable if there exists a TM M such that, for all inputs $w \in \Sigma^*, M(w)$ alts with only $f(w)$ on tape.

def: **Mapping Reduction**

$f : \Sigma^* \to \Sigma^*$ is a mapping reduction from languages A to B if:

1. f is computable
2. $x \in A \iff f(x) \in B$

If so, then $A \leq_m B$

thm: $R\mathcal{E}$ **Enumerator**

$L \in R\mathcal{E} \iff L$ has an enumerator

thm: **R**

$R = R\mathcal{E} \wedge$ Co-$R\mathcal{E}$

thm: **Rice's Theorem**

Let $P \subset R\mathcal{E}$ be nontrivial, meaning that $P \neq \emptyset$ nor $R\mathcal{E}$.

$$L = \{ \ \langle M \rangle \ | \ \mathcal{L}(M) \in P \ \}$$

is not decidable

thm: **Closure under reduction**

Let $A \leq_m B$

1. $A \notin R\mathcal{E} \to B \notin R\mathcal{E}$
2. $A \notin$ Co-$R\mathcal{E} \to B \notin$ Co-$R\mathcal{E}$
3. $A \notin R \to B \notin R$

Decidability Problems

def: **Acceptance Problem**

$A_M = \{ \ (\langle M \rangle, x) \ | \ x \in \mathcal{L}(M) \ \}$

def: **Halting Problem**

$\text{Halt}_{TM} = H_{TM} = \{ \ (\langle M \rangle, x) \ | \ M(x) \text{ halts} \ \}$

def: **Empty**$_{TM}$ $= E_{TM}$

$\{ \ \langle M \rangle \ | \ \mathcal{L}(M) = \phi \ \}$

def: **EQ**$_{TM}$

$\{ \ (\langle M_1 \rangle, \langle M_2 \rangle) \ | \ \mathcal{L}(M_1) = \mathcal{L}(M_2) \ \}$

def: **Busy Beaver Function**

a_n = number of TMs with at most n states. $bb(n)$ = max number of steps in the computation of $M_i(\epsilon)$

thm: $A_{DFA} \in R$

thm: $A_{CFG \in R}$

thm: $A_{TM} \in R\mathcal{E}, \notin \text{Co-}R\mathcal{E}$

thm: $H_{TM} \in R\mathcal{E}, \notin \text{Co-}R\mathcal{E}$

thm: $\text{Empty}_{TM} \notin R$

thm: $\text{EQ}_{TM} \notin R$

thm: $bb(n)$ **is not computable**

If it were, $\text{Halt}_{TM} \in R$ because it gives us the step in which we can halt.

Runtime

def: **Big O notation**

$f(n) = O(g(n))$ if $\exists c > 0, n > 0$ such that:
$f(n) \leq c \cdot g(n), \forall n \in \mathbb{N}$

def: **TM Runtime**

M runs in $t(n)$ if $\forall x$ of length $\leq n$,
$M(x)$ halts after at most $t(n)$ steps.

def: **Verifier**

TM V is verifier if
$L = \{ x \in \Sigma^* \mid \exists w \in \Sigma^*, V(x, w) = 1 \}$
V is poly-time if it runs in poly with respect to $|x|$

def: **DTIME($t(n)$)**

$\{ L \mid \exists M$ such that $\mathcal{L}(M) = L \wedge M$ runs in $O(t(n)) \}$

def: **Poly-Time Computable**

f is poly-time computable if $\exists$ TM M that runs in poly($|x|$) and $M(x)$ has $f(x)$ on tape when it halts

def: **Poly-time Reduction Map**

f is a poly-time reduction map from A to B if:

1. f is poly-time computable
2. if $x \in A$, then $f(x) \in B$
3. if $x \notin A$, teh $f(x) \notin B$

If f exists, then $A \leq_p B$

def: **P**

$\bigcup_{c=1}^{\infty} \text{DTIME}(n^c)$

def: **NP**

$NP = \bigcup_{c=1}^{\infty} \text{NTIME}(n^c)$
$L \in NP \iff L$ has poly-time verifier

def: **NP-Hard**

$A \in$ NP-Hard if $\forall L \in NP, L \leq_p A$

def: **NP-Complete**

$A \in$ NP-Complete if $A \in$ NP-Hard and $A \in$ NP

thm: **Multi-tape to single run-time**

$\exists$ single-tape universal TM U that, given $(\langle M \rangle, x)$ where M is multi-tape and $M(x)$ runs in $t(n)$, U simulates $M(x)$ in $O(|\langle M \rangle|^2 \cdot t(n)^2)$

thm: **Time Hierarchy Theorem**

Can solve more with more time
$t_2(n) \geq n^2 t_1(n)^2 \rightarrow \text{DTIME}(t_1(n)) \not\subseteq \text{DTIME}(t_2(n))$

thm: **P closed under $\leq_p$**

$A \leq_p B$ and $B \in P \rightarrow A \in P$

thm: **NP-Hard closed under $\leq_p$**

$A \leq_p B$ and $A \in$ NP-Hard $\rightarrow B \in$ NP-Hard

NP Languages

def: **3-CNF**

An logic expression consisting of AND of clauses.
Clauses are 3 variables with the OR/negation operator.

def: **3-SAT**

$\{ \langle \phi \rangle \mid \phi \text{ is satisfiable} \}$

def: **Vertex-Cover**

$C \subseteq V(G)$ is a vertex-cover if
$\forall (u,v) \in E(G), u \in C$ or $v \in C$

def: **Independent-Set**

$I \subseteq V(G)$ is independent-set if
$\forall u, v \in I, (u,v) \notin E(G)$

def: **Knapsack**

$\{ (\{a_1, ..., a_k\}, \{b_1, ..., b_k\}), B, t \mid \exists I \subseteq [k]$ s.t.
$\sum_{i \in I} a_i \geq t \wedge \Sigma_{i \in I} b_i \leq B \}$

def: **Dominating Set**

A set D is dominating *in* G if every vertex in $V(G)$ is either in D or adjacent to some vertex in D.

def: **Clique**

A subset of vertices that create a complete graph.

NP-Complete

thm: **3-SAT in NP-Complete**

$\{ \langle \phi \rangle \mid \phi \text{ is 3-CNF and } \exists \text{ assignment } x \text{ s.t } \phi(x) = 1 \}$

thm: **Subset-Sum in NP-Complete**

$\{ (A, T) \mid \exists S \subseteq A \text{ s.t } \sum_{s \in S} s = t \}$

thm: A_{NP} **in NP-Complete**

$\{ (\langle M \rangle, x, 1^k) \mid \exists w \text{ s.t } M(x, w) = 1 \text{ in } \leq k \text{ steps} \}$

thm: **Independent-Set in NP-Complete**

$\{ (\langle G \rangle, k) \mid G \text{ has IS of size } \geq k \}$

thm: **Partition in NP-Complete**

$\{ S \mid S \text{ is a set and } \exists T \subseteq S \text{ s.t } \sum_{x \in T} x = \sum_{y \in S \setminus T} \}$

thm: **Vertex-Cover in NP-Complete**

$\{ (\langle G \rangle, k) \mid G \text{ has a vertex-cover of size } \leq k \}$

thm: **Dominating Set in NP-Complete**

$\{ (\langle G \rangle, k) \mid G \text{ has a dominating set of size } \leq k \}$

thm: **Clique in NP-Complete**

$\{ (\langle G \rangle, k) \mid G \text{ has a clique of size } \geq k \}$

NP

thm: Knapsack in NP