Halting Problem

对于任一程序，我们无法设计一个算法判断它是否会在有限时间内停机.

假设存在 willHalt(foo) 函数可以判断.

```
void foo() {
    if ( willHalt(foo) ) {
        while (true) {} // Endless loop.
    }
    return;
}
```

接下来，如果我们想知道 `foo()` 是否会停机，就会执行 `willHalt(foo)`。然而在 `foo()` 内部也有一个 `willHalt(foo)`，如果它认为 `foo()` 会停机，则构造一个死循环；而如果它认为 `foo()` 不会停机，则选择让它立刻停机，于是这里就产生了矛盾。

Turing Machine

Deterministic Turing Machine：下一步做什么由当前状态决定.

Non ... ：在多项时间内一直走在正确的方向.

The problem is NP if we can prove any solution is true in polynomial time.

eg：Hamilton Cycle：找一个包含所有 vertex 的 cycle. NP.

⇒ 找一个不含 Hamilton Cycle 的 graph：非 NP，但是 P.

$P \subseteq NP$. $NP \nsubseteq P$

NP Complete Problem (NPC)

Every problem in NP can polynomial (多项式时间归约) reduce to NP complete problem.

在 polynomial time 内将 A 转化为 B. ⇒ $A \leq_p B$ (A不比B难).

⇔ $\exists f()$ in polynomial time

s.t. $\forall x \in A$, $f(x) \in B$, and $\forall f(y) \in B$, $y \in A$

Traveling Salesman Problem.

已知 Hamilton Cycle 是一个 NPC 问题.

TSP：给定一个完全图，求一个 Hamilton Cycle，且路径长 $\le k$.   每对顶点之间都有连接.

⇒ 证明 NPC：① 证 TSP 为 NP

② 证明 Hamilton Cycle 可 reduce 为 TSP.

① ✓.

② HCP：$G(V,E)$，无边权

TSP：$G'(V',E')$ 完全图，有边权，$\sum v_i \le K$

将 $G(V,E)$ 中的 V 边权设为 1，无边处连接并设边权为 0.

完全图边数 $\frac{n(n-1)}{2}$ → polynomial

在 G 上找 Hamilton Cycle ⇔ 在 $G' = f(G)$ 上找 $k=0$ 的 TSP

∴ $HCP \le_p TSP$

⇒ TSP ∈ NPC.

A Formal Language Framework. 形式化语言框架.

SHORTED-PATH Algorithm.

$I = \{ \langle G, u, v \rangle : G = (V,E) \text{ is an undired graph}, u, v \in G \}$.

$S = \{ \langle u, w_1, w_2, \cdots, w_k, v \rangle : \langle u, w_1 \rangle, \cdots, \langle w_k, v \rangle \in E \}$

For every $i \in I$, SHORTED-PATH $(i) = s \in S$.

对 decision PATH：$I = \{ \langle G, u, v, k \rangle : \ \cdots \ \}$，$S = \{0, 1\}$.

Formal-language Theory

Σ：一组有限的符号 Σ 上的一种语言 L：用 Σ 中符号组成.

如：$L = \{ x \in \Sigma^* : Q(x) = 1 \}$ $\Sigma = \{0,1\}$

ε: empty string　$\emptyset$: empty language

Σ上所有字符串之语言: Σ^*.　L之补集: $\Sigma^* - L$

L_1和L_2之串联: $L = \{x_1 x_2 \mid x_1 \in L_1, x_2 \in L_2\}$ → concatenation.

Kleene star of language L:

L之闭包: $L^* = \{\varepsilon\} \cup L_1 \cup L_2 \cup \cdots \cup L_k \cup \cdots$

(L_k表示L与自身串联k次)

Algorithm A

1. accepts a string $x \in \{0,1\}^*$ if $A(x) = 1$
2. rejects - - - $x \in \{0,1\}^*$ if $A(x) = 0$
3. decides language L if L中任何 string x 满足 $A(x) = 1$. accepted

   L外 - - - - - - - $A(x) = 0$. rejected.

$P = \{L \subseteq \{0,1\}^*$: there exists an Algorithm A that decides L in polynomial time$\}$

Verification Algorithm.

2-argument algorithm: input string x + binary string y → certification

4. 若存在 certification y 使 $A(x,y) = 1$, 则称A certificate x.

描述语言: $L = \{x \in \{0,1\}^*$: $\exists\, y \in \{0,1\}^*$ 使 $A(x,y) = 1\}$.

〖Example〗 For SAT

$x = (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4)$

Certificate: $y = \{x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 1\}$

language L 是 NP iff

$\exists$ A (2 input, polynomial time) 和 c, s.t.

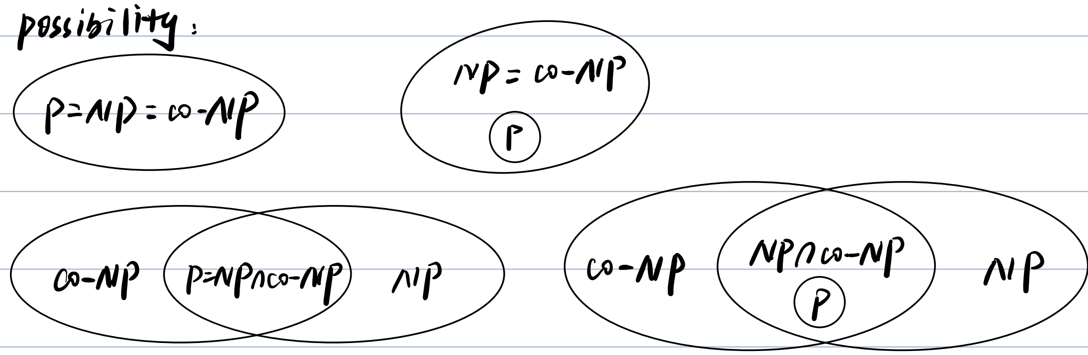
$L = \{x \in \{0,1\}^*$: $\exists\, y$, $|y| = O(|x|^c)$ s.t. $A(x,y) = 1\}$.

(A verifies language L in polynomial time)

complexity class co-NP

co-NP = { L | L ∈ NP and $\bar{L}$ ∈ NP }

4 possibility:



Vertex Cover Problem

1. **Clique Problem**

无向图 G=(V,E), integer K.

G是否包含一个完全子图(clique), 要求至少有K个顶点.

clique = { <G,K> : G is a graph with a clique of size K }.

2. **Vertex Cover Problem** (TSP问题抽象化)

无向图 G=(V,E). integer K.　　(未all vertex, 对一个edge 可以只覆盖一个vertex)

G是否包含一个子图, 满足: ① 有至多K条边, ② cover G中所有边之端点 (至少一个)

vertex-cover = { <G,K> : G has a vertex cover of size K }.

① 先证明是NP

对 ∀x = <G,K>, 取 V' ⊆ V 作 certification y.

????　Verification algorithm: check if |V'| = K　　=> O(N³) ??

一定有 $A(x,y)=1$. | check 每一 edge (u,v), 端点 u 或 v 在 V' 中

② 证明 clique $\le_p$ vertex-cover

我看不动了