Euclid's Algo (GCD)

```
if (y = 0) return x
if (y = 1) return 1
return Euclid(y, x mod y)
```

Potential (Leaky Bucket):

① Unit of time

② How to measure start $S_0 \geq 0$

③ Show $S_{t+1} < S_t$

** find good potential function **

Size of Input = $O(\log x)$

Master Theorem:

$T(n) = k\,T(n/b) + O(n^d)$

k: # of subproblems

b: size of subproblem

n^d: time to "merge"

$$T(n) = \begin{cases} O(n^d) & k < b^d \\ O(n^d \log n) & k = b^d \\ O(n^{\log_b k}) & k > b^d \end{cases}$$

$T(n) = k\,T(n/b) + O(n^d \log^w n)$

$$T(n) = \begin{cases} O(n^d \log^w n) & k/b^d < 1 \\ O(n^d \log^{w+1} n) & k/b^d = 1 \\ O(n^{\log_b k}) & k/b^d > 1 \end{cases}$$

Closest Pair (2D):

» Sort points by x-coord

» find closest pair in each half

» $\delta = \min(\delta_1, \delta_2)$

» To find close red/blue pair: slide $\delta \times 2\delta$ rectangle down (for each P ∈ filtered_in_δ_strip compare to 7 lower points)

» $T(n) = 2T(n/2) + O(n) = O(n \log n)$

Karatsuba Multiplication:

» $x = a \cdot 10^{n/2} + b$; $y = c \cdot 10^{n/2} + d$

» $xy = ac \cdot 10^n + (ad+bc) \cdot 10^{n/2} + bd$

» ac ; bd ; $ad+bc = (a+b)(c+d) - ac - bd$

» $T(n) = 3T(n/2) + O(n) \rightarrow O(n^{\log_2 3})$

» // Do 3 mults instead of 4!

Longest Increasing Subsequence

- Sequence of numbers, $S_i < S_{i+1}$
- $|LIS(X)|$ = LIS of array of nums X
- Can't just directly compute - solve a related subproblem
  - $L(X)$ = |LIS that includes last num|
- $L(X) = |X|$ if $|X| \leq 1$
  $= \max(L(x')) + 1$, $x' := x[\text{end}] < x[\text{end}]$
- Find all prev entries that are smaller, take max of those $L(x)$'s, add 1

Big-O Definitions + Proofs

$f(n) = O(g(n)) \rightarrow |f(n)| \leq M|g(n)|$, $\forall n > n_0$

$f(n) = \Omega(g(n)) \rightarrow |f(n)| \geq M|g(n)|$, $\forall n > n_0$

$f(n) = \Theta(g(n)) \rightarrow f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))$

$f(n) = o(g(n)) \rightarrow |f(n)| < M|g(n)|$, $\forall n > n_0$

$f(n) = \omega(g(n)) \rightarrow |f(n)| > M|g(n)|$, $\forall n > n_0$

Proving $f(n) = O(g(n))$:

- Master Thm
- $\lim_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| < c$ for $c \in \mathbb{R}$ (L'Hôpital: $\lim \frac{f(n)}{g(n)} = \lim \frac{f'(n)}{g'(n)}$)

Note: $\frac{d}{dx} a^x \rightarrow \ln(a)\,a^x$

- Find witnesses M, n_0

Prove $f(n) \neq O(g(n))$:

- Use Master Thm to convert
- $\lim_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| = \infty$ (diverging not enough)
- Show $g(x) = o(f(x)) \Leftrightarrow \lim_{n\to\infty} \left|\frac{g(n)}{f(n)}\right| = 0$
- Argue 203-style: $\forall M, n_0, \exists n > n_0$ s.t. $|f(n)| \geq M|g(n)|$ (Construct n value)

- Alphabet Σ: non-empty, finite set of symbols
- Σ^n: $\Sigma \times \Sigma \times \Sigma$, set of all strings of length n from Σ
- $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$
- Language L: $L \subseteq \Sigma^*$

DFA: $\{Q, \Sigma, \delta, q_0, F\}$

→ Q: set of states

→ Σ: alphabet

→ $\delta: Q \times \Sigma \rightarrow Q$ transition function

→ $q_0 \in Q$: initial state

→ $F \subseteq Q$: accepting state(s)

DFA M "accepts" x iff ends in a final state ∈F after reading x

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

If L_1, L_2 are regular languages:

- $\Sigma^* - L$ ($\bar{L}$)
- L^*
- L^R (reverse) — flip transitions?
- $L_1 \cup L_2$, $L_1 \cap L_2$

are all regular languages

No DFA decides $\{0^n 1^n \mid n \geq 0\}$

It is possible to decide if 2 DFAs decide same language

Church-Turing Thesis

- Every problem computable by mechanical device is computable by a Turing Machine
- Two models of computation equivalent if one can simulate the other (prove both ways)

Reg Exes:
*: 0 or more
+: 1 or more
[A-Z]: all uppercase letters

Chomsky's Hierarchy

Regular Languages ↔ Finite Automata

Context-Free Ls ↔ Pushdown Automata

Context-Sensitive Ls ↔ Linear-Bounded Aut.

Recursively-enumerable ↔ Turing (recognizable) Ls Machines

Turing Machines

- Read-Only TM ≡ DFA (Rabin + Scott)
- Consists Of:
  - → 1D infinite tape of cells — initially contains input, then blanks (⊥)
  - → tape head: can be moved L/R initially @ leftmost cell reads symbol under head and writes from tape alphabet $\Gamma \supseteq \{\Sigma \cup \perp\}$
  - → controlled by state machine — accepts/rejects on special states
- $M = (Q, \Gamma, \Sigma, \delta, q_{start}, q_{accept}, q_{reject})$

Q: set of states Γ: tape alphabet

Σ: input alphabet (Σ, ⊥, and more (?))

$\delta: (Q - q_A, q_R) \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ transition function

- M accepts x ⇔ ends @ q_{accept}

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

M recognizes $L(M)$

- M also DECIDES $L(M)$ if it halts on all inputs; ∴, it accepts all $x \in L(M)$ and rejects $x \notin L(M)$

CH (1878): No set S exists such that $|\mathbb{N}| < |S| < |\mathbb{R}|$

Paul Cohen (1963): CH independent of classical set theory axioms

Dynamic Programming:

① Top-Down Recursive (inefficient if overlapping subproblems; space-efficient)

② Top-Down Memoization (faster time, more space, only computes needed subs)

③ Bottom-Up Table (faster time, more space, may calculate unnecessary subproblems)

Longest Common Subsequence:

```
LCS(X[1...m], Y[1...n]):
  if(m==0 || n==0)
    return 0
  if (X[m] == Y[n])
    return 1 + LCS(X[1→m-1], Y[1→n-1])
  return max{ LCS(X[1→m], Y[1→n-1]),
              LCS(X[1→m-1], Y[1→n]) }
```

(Do using table)

Recurrence Relations:

- Rod-Cut

→ given rod of length n, list P of prices where P[i] is price of length i, find max profit of cutting rod + selling pieces

» $R(0) = 0$; $R(1) = P[1]$

» $R(n) = \max_{1 \leq k \leq j} \{P(k) + R(j-k)\}$

- Edit-Distance:

→ Transform string A into B via substituting, deleting, inserting characters (cost s, d, i)

→ Find minimum-cost of transforming A to B

» Let A = [1, ..., m] ; B = [1, ..., n]

» If A[m] = B[n]:
  return ED(A[1→m-1], B[1→n-1])

» Else:
  $\min\{ c_1 + ED(A[1\to m], B[1\to n-1]),$
  $c_2 + ED(A[1\to m-1], B[1\to n-1]),$
  $c_3 + ED(A[1\to m-1], B[1\to n]) \}$

Proving Not All Binary Languages Decidable

- Let ALL be set of all binary languages; D be decidable binary languages
- Show that $|D| = |\mathbb{N}| < |ALL| = |\mathbb{R}|$ (countably vs. uncountably ∞)
- Recall: ∞ set countable if bijection to ℕ (can list in some order)
- $\{0,1\}^*$ is countable — just list LEXICOGRAPHICALLY (length, then alphabetical)
- Similarly, all valid C++ programs countable (lexicographical ordering)
- As C++ programs ≡ Turing Machines ⇔ Languages, |D| is thus COUNTABLE.
- ALL is uncountable — consider listing every possible language out in some table. Rows ≡ language #; columns ≡ $\{0,1\}^*$; entries: include/not include

	ε	0	1	00	01	…
0	N	Y	Y	N	N	
1	Y	N	Y	N	N	
2	Y	Y	Y	Y	Y	
3	N	N	N	N	N	
⋮						

$\bar{L}$ then differs from row i of the table in the i^{th} spot every time. Thus, we have a a contradiction. No such table can exist, so ALL is UNCOUNTABLE.

→ Aside: similar argument to show $|\mathcal{P}(S)| > |S|$ (power set goes ↑)

- So, as ALL uncountable and D countable, ALL − D uncountable.

Proof that L_{DIAG} is Undecidable (Languages that don't accept themselves)

- Let S be language of all valid Turing Machine encodings
- $L_{DIAG} = \{\langle M\rangle \in S \mid \langle M\rangle \notin L(\langle M\rangle)\}$ (M does not accept itself)
- Claim: For any valid TM M^*, $L(M^*) \neq L_{DIAG}$.
- Suppose $L_{DIAG} = L(M^*)$. Then, $\langle M^*\rangle$ itself either ∈ or ∉ L_{DIAG}
  - Case 1: $M^* \in L_{DIAG}$
    Then, $M^* \notin L(M^*)$. But $L(M^*) = L_{DIAG}$, so contradiction!
  - Case 2: $M^* \notin L_{DIAG}$
    Then, $M^* \in L(M^*)$. But $L(M^*) = L_{DIAG}$, so contradiction!
- Thus, no such M^* exists and L_{DIAG} is thus undecidable.

L_{ACC} Undecidable (T.R.)

- Let $L_{ACC} = \{\langle M, x\rangle$: TM M accepts $x\}$
  "Universal" TM that can simulate any other.
- Show $L_{DIAG} \leq_T L_{ACC}$
- Suppose for sake of contradiction a decider D exists for L_{ACC}.
- Then, construct D' for L_{DIAG}:
  » if input x ∉ M → reject ← invalid input
  » return $!D(\langle M, \langle M\rangle\rangle)$
- This is a decider because D is decider.
- If D accepts, then M accepts $\langle M\rangle$ — so D' rejects.
- Similarly, if D rejects, then M rejects $\langle M\rangle$ — so D' accepts.
- This creates contradiction — created decider for undecidable language, so ∄ D.

L_{HALT} is Undecidable

- $L_{HALT} = \{\langle M, x\rangle$: M halts on $x\}$
- Show $L_{ACC} \leq_T L_{HALT}$
- ∃ decider D for L_{HALT}, construct decider D' for L_{ACC}

On input $\langle M, x\rangle$:
  if $D(M, x)$ = reject → reject
  else return $M(x)$

- As L_{ACC} undecidable, so is L_{HALT}

** Reminders **

All-Pairs Shortest Path (Floyd-Warshall) — DP

- Given directed, weighted graph on n vertices $\{1, \ldots, n\}$, calculate shortest path $d(i,j)$ ∀ (i,j) s.t. $i \neq j$. Assume: no negative cycles
- Idea: Recursively compute $d^l(i,j)$ — shortest path using vertices that are in range $[1, \ldots, l]$. This ensures dependent entries done
- If $l = 0$, $d^l(i,j) = w(i,j)$ [could be ∞ if no edge]
- Else: $d^l(i,j) = \min\{ d^{l-1}(i,l-1) + d^{l-1}(l-1,j),\ d^{l-1}(i,j)\}$
- $d(i,j) = d^n(i,j)$
- Bottom-Up — 3D Table (l, i, j)
  » for ∀ i, j: $T[0,i,j] = w(i,j)$
  » for $l = 1 \rightarrow n$ (works b/c entire $l-1$ filled already)
    for ∀ i, j:
    $T(l,i,j) = \min\{T(l-1,i,j),\ T(l-1,i,l) + T(l-1,l,j)\}$

Kruskal's Minimum Spanning Tree (Greedy)

- G is weighted, connected, undirected graph (V, E)
- Spanning Tree of G: acyclic, connected subgraph ($e \in E$) containing all vertices
- Find minimum-weight spanning tree of G
- Kruskal's (Greedy):
  Let T = ∅ (empty graph)
  Let E = sorted (E) — increasing order of weight
  For $e \in E$:
    if T + e is acyclic: T ← T + e
  return T
- Proof: Kruskal finds MST
  » Suppose T' is MST. Show that T' can be transformed to T (still optimal)
  » Let $e_1, e_2, \ldots e_{n-1}$ be edges of T (sorted, in order of addition)
  » Proof by induction:
    » Base Case: $T(e_1 \rightarrow e_1) = T'(e_1 \rightarrow e_1)$
    » IS: Suppose $e_1 \rightarrow e_k$ OK. Consider e_{k+1} ($k+1^{st}$ in T)
    » if $(e_{k+1} \in T')$, we are good.
    » else: adding e_{k+1} to T' creates a cycle C. Then, ∃ $f \in T'$ on cycle C
      Remove f and swap w/ e_{k+1}. This is optimal b/c as f ∉ T,
      Weight(f) ≥ Weight(e_{k+1}).

Longest Palindromic Subsequence

- Given string S, determine length of longest palindromic subsequence of S
- Insight: Use LPS[i,j] for substring from i to j (inclusive)

$$LPS[i,j] = \begin{cases} 1 & i == j \\ LPS[i+1, j-1] + 2 & S[i] == S[j] \\ \max(LPS[i, j-1], LPS[i+1, j]) & \text{otherwise} \end{cases}$$

0-1 Knapsack (DP)

- Given Weights $W = (w_1, w_2, \ldots, w_n)$ and Values $V = (v_1, v_2, \ldots, v_n)$
- Have knapsack with weight capacity C — pick subset of items $(1 \rightarrow n)$ such that $\sum w_i \leq C$ and $\sum v_i$ is maximized
- Subproblem: $K(i, c)$:= value when only consider first i items, capacity C
  » If $c = 0$ || $i = 0$ → $K(i,c) = 0$
  » Elif $w_i > C$ → $K(i,c) = K(i-1, c)$
  » Else → $K(i,c) = \max\{K(i-1, c),\ K(i, C - w_i)\}$

stone in n × C table
iterate i's first, then C's

Balancing Weights (HW Group Problem)

- Given $n \geq 1$ positive integer weights $w_1, w_2, \ldots, w_n$. Find subset $A \subseteq \{1, \ldots, n\}$ s.t. $|\sum_{i \in A} w_i - \sum_{i \notin A} w_i|$ is minimum.
- Insight: reduce to the knapsack(?) problem: if $k = \sum w_i$, reach $\lfloor k/2 \rfloor$.
- Maintain $\lfloor k/2 \rfloor \times n$ memo of bools (!)
  » $A(i, w) =$ 1 if $w = 0$; 0 if $i = 0 \wedge w \neq 0$
    $A(i-1, w)$ || $A(i, w - w_i)$ otherwise
- Iterate through $A(n, ?)$ column from $\lfloor k/2 \rfloor$ up and find first 1.
- Return $k - 2w$ [weight thats 1]

$L_{E\text{-}ACC}$ is Undecidable

- $L_{E\text{-}ACC}$: $\{\langle M\rangle \mid M$ is TM accepting ε$\}$
- Show $L_{ACC} \leq_T L_{E\text{-}ACC}$
- Let D decide $L_{E\text{-}ACC}$; construct D'
- D': on input $\langle M, x\rangle$:
  M_1 = "on input y:
    if y == ε, return M(x)
    else, reject"
  return $D(\langle M_1, \varepsilon\rangle)$
- D' is a decider because D is a decider, so always terminates. Furthermore, if M accepts x, $D(\langle M_1, \varepsilon\rangle$ accepts and rejects otherwise. So, D' decides L_{ACC}

L_{EQ} (Autograder) Undecidable

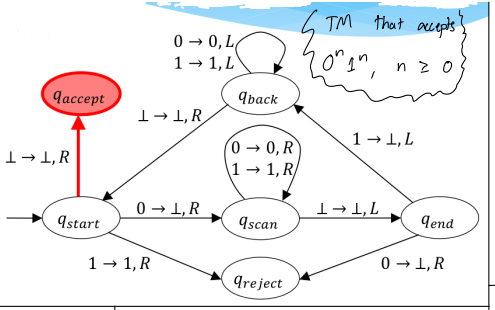
- $L_{EQ} := \{\langle M_1, M_2\rangle : L(M_1) = L(M_2)\}$
  (M_1, M_2 equivalent Turing Machines)
- We will show $L_{ACC} \leq_T L_{EQ}$
- Let D decide L_{EQ}. Construct D':
  D': on input $\langle M, x\rangle$
    Let TM M_1: "on all input, ACCEPT"
    Let TM M_2: "on all input,
      return M(x)"
    return $D(\langle M_1, M_2\rangle)$

Cantor's Diagonalization to show $|S| < |\mathcal{P}(S)|$

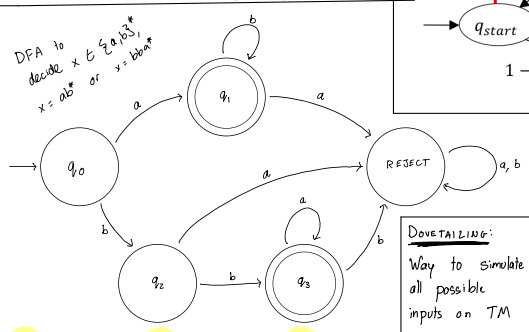
- Consider any function $S \rightarrow \mathcal{P}(S)$ (maps elements in set to elems in power set)
- Let DIAG = $\{x \in S \mid x \notin f(x)\}$
  - all elements of S s.t. x not contained in corresponding subset of S
  - DIAG itself a subset of S
- Claim: $f(x^*) \neq$ DIAG ∀ $x^* \in S$ — by contradiction.
- Assume $f(x^*) =$ DIAG. Then $x^* \in$ DIAG or $x^* \notin$ DIAG
- If $x^* \in$ DIAG, then $x^* \notin f(x^*) \rightarrow x^* \notin$ DIAG — CONTRADICTION
- If $x^* \notin$ DIAG, then $x^* \in f(x^*) \rightarrow x^* \in$ DIAG — CONTRADICTION
- So, ∞ DIAG ∉ $f(x^*)$ ∀$x^* \in S$, $|S| < |2^S|$

Proof that 2-tape TM ≡ 1-tape

- 2-tape can simulate 1-tape by ignoring second tape
- 1-tape can simulate 2-tape as follows:
  - → parity determines 1^{st} / 2^{nd} tape
  - → remember head thru special symbols $0'$, $1'$, $\perp'$
  - → to change state: find first head, modify then find 2^{nd} head, modify



DFA to decide $x \in \{a,b\}^*$ s.t. $x = ab^*$ or $x = bba^*$



Dovetailing: Way to simulate all possible inputs on TM

Order inputs lexicographically

① Simulate one step of x_0
② Simulate 1 step for x_0, x_1
③ Simulate 1 step x_0, x_1, x_2
- - -

Similar to alternating 2 machines, but works for infinitely many machines (countable)

Properties of Decidable Languages

- If L is decidable, $\bar{L}$ is decidable
- If L_1, L_2 decidable, $L_1 \cap L_2$ decidable
- If L_1, L_2 decidable, $L_1 \cup L_2$ decidable
- If L decidable, $L \leq_T L'$ for all L'
- $\forall L,\ L \leq_T \bar{L}$

Properties of Undecidable Languages

- If L undecidable, $\bar{L}$ also undecidable
- For undecidable L_1, L_2, $(L_1 \cap L_2)$ not necessarily undecidable (consider: $L_1 \cap L_2 = \emptyset$)
- Similarly, $L_1 \cup L_2$ not necessarily undecidable (consider: $L_2 = \bar{L_1}$)

- If $A \leq_T B$, $B \leq_T C \to A \leq_T C$

Recognizability

- L RECOGNIZABLE if $L = L(M)$ for some M
  - M may loop on on input
- All decidable languages are recognizable
- If L_1, L_2 recognizable, so is $L_1 \cup L_2$
  - » Alternately simulate steps of M_1, M_2
  - » accept as soon as one accepts
- Similarly, $L_1 \cap L_2$ is recognizable
- If BOTH L and $\bar{L}$ recognizable, then L is decidable
  - » alternately simulate recognizers — one MUST eventually accept
- COROLLARY: if L undecidable, then at least one of $L, \bar{L}$ unrecognizable
- L is CO-RECOGNIZABLE if $\bar{L}$ is recognizable
  - → Decidable L is both recognizable, corecognizable
  - → Countably many recognizable languages b/c countably many programs

Table of Languages

Name	Defn.	D	R	C-R
L_{DIAG}	$\langle M \rangle$: $\langle M \rangle \notin L(M)$	X	X	✓
L_{ACC}	$\langle M,x \rangle$: M accepts x	X	✓	X
L_{HALT}	$\langle M,x \rangle$: M halts on x	X	✓	X
$L_\emptyset$	$\langle M \rangle$: $L(M) = \emptyset$	X	X	✓
L_{EQ}	$\langle M_1, M_2 \rangle$: $L(M_1) = L(M_2)$	X	X	✓
$L_{\varepsilon\text{-HALT}}$	$\langle M \rangle$: M halts on ε	X	✓	X

(a) $L_1 = \{\langle M_1, M_2 \rangle : M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) \cap L(M_2) \neq \emptyset\}$.

Solution: The language L_1 is undecidable. We will prove this via Turing reduction with $L_\emptyset \leq_T L_1$. Assume that we have some decider D for L_1. We will construct a Turing machine T that decides $L_\emptyset$ using D as follows:

```
T(⟨M⟩)
1: Construct a Turing machine M' as follows:
     M'(w)
     1: Accept.
2: Run D(⟨M', M⟩)
3: if D accepts ⟨M', M⟩ then
4:     Reject
5: else
6:     Accept
```

(b) $L_2 = \{\langle M \rangle : M \text{ is a TM that loops on the input string "eecs376"}\}$.

Solution: The language L_2 is undecidable. We will prove this via Turing reduction with $L_{HALT} \leq_T L_2$. Assume that we have some decider D_2 for L_2. We will construct a Turing machine D_{HALT} that decides L_{HALT} using D_2 as follows:

```
D_HALT(⟨M⟩, x)
1: Construct a Turing machine M' as follows:
     M'(w)
     1: Run M(x)
     2: Accept
2: Run D_2(M')
3: if D_2 accepts M' then
4:     Reject
5: else
6:     Accept
```

Note that the Turing machine M' we construct have $L(M') = \Sigma^*$ when M halts on x and it will loop on all the strings if M loops on x.

$\forall(\langle M \rangle, x) \in L_{HALT} \implies M$ halts on $x \implies L(M') = \Sigma^* \implies M'$ accepts the string "eecs376" $\implies D_2$ rejects $M' \implies D_{HALT}$ accepts $(\langle M \rangle, x)$.
$\forall(\langle M \rangle, x) \notin L_{HALT} \implies M$ loops on $x \implies M'$ loops on all the strings, including the string "eecs376" $\implies D_2$ accepts $M' \implies D_{HALT}$ rejects $(\langle M \rangle, x)$.
Since D_{HALT} decides L_{HALT}, we have the Turing reduction $L_{HALT} \leq_T L_2$. Then we can know L_2 is undecidable as L_{HALT} is undecidable.

Proof: L_{DIAG} is unrecognizable

- $L_{DIAG} = \{\langle M \rangle$ s.t. $\langle M \rangle \notin L(M)\}$
- $\overline{L_{DIAG}}$ recognizer:
  - On input $x = \langle M \rangle$ for some TM M
    - return $M(\langle M \rangle)$;
    - If $x \neq \langle M \rangle$: accept
- As L_{DIAG} undecidable and $\overline{L_{DIAG}}$ recognizable, L_{DIAG} must be unrecognizable

Proof: $L_\emptyset$ is UNRECOGNIZABLE

- $L_\emptyset = \{\langle M \rangle \mid M \text{ is TM}, L(M) = \emptyset\}$
  - All TMs that accept no input
- $L_\emptyset$ is undecidable:
  - $L_{DIAG} \leq_T L_\emptyset$ (Let D decide $L_\emptyset$)
  - D': On input $\langle M \rangle$
    - M_1 = "return $M(\langle M \rangle)$ on all input"
    - return $D(\langle M_1 \rangle)$
- $\bar{L}_\emptyset$ is recognizable
  - On input $\langle M \rangle$:
    - simulate $M(x)$ for all x by DOVETAILING
    - accept as soon as M accepts some x
- Thus, $L_\emptyset$ must be unrecognizable.

Kolmogorov Complexity

- KC of a binary string x is length of shortest program that outputs x
- Complexity independent of programming language
- Implication: $\exists$ incompressible string

Proving Undecidability using Rice's

① Find semantic property $\mathbb{P}$ s.t. $L = L_{\mathbb{P}}$
② Show $\mathbb{P}$ is non-trivial ($L_1 \in \mathbb{P}$, $L_2 \notin \mathbb{P}$)
③ Conclude L is undecidable

Ex: $L = \{\langle M \rangle$: M accepts all strings of length less than 376$\}$

Let S_{376} be set of all strings less than length 376. Then let $\mathbb{P}$ be the set of all languages L^* s.t. $S_{376} \subseteq L^*$. Then, if $\langle M \rangle \in L$, M accepts all s s.t. $s \notin S_{376}$. As $S_{376} \subseteq L(M)$, $L(M) \in \mathbb{P}$ and thus $\langle M \rangle \in L_{\mathbb{P}}$.
Furthermore, $\Sigma^* \in \mathbb{P}$ and $\emptyset \notin \mathbb{P}$, so $\mathbb{P}$ is non-trivial (both recognizable).
So, by Rice's, L is undecidable.

Rice's Theorem

- For any non-trivial property $\mathbb{P}$ (such that there exists recognizable $A \in \mathbb{P}$ and $B \notin \mathbb{P}$), $L_{\mathbb{P}} = \{\langle M \rangle \mid L(M) \in \mathbb{P}\}$ is undecidable.
  - → Only decidable when $\mathbb{P}$ includes all/no langs
  - → $\mathbb{P}$ is a PROPERTY — just a set of languages
  - → For any non-trivial set of languages, impossible to decide whether a given TM recognizes a language in that set
- Proof: Show $L_{HALT} \leq_T L_{\mathbb{P}}$
  Let D decide $L_{\mathbb{P}}$; Construct D' for L_{HALT}

Case 1: $\emptyset \notin \mathbb{P}$ ($\mathbb{P}$ does not include empty language)

D' on input $\langle M, x \rangle$
Let M_1 = "On input w:
simulate $M(x)$
if $M(x)$ halts:
return $M_A(x)$, $A \in \mathbb{P}$"
Return $D(M_1)$

So, $L(M_1) = A$ ($\in \mathbb{P}$) if halts, and $\emptyset$ ($\notin \mathbb{P}$) if not.

Case 2: $\emptyset \in \mathbb{P}$

Some proof, substitute M_A for M_B, $!D(M_1)$

So, if M halts → $L(M_1) = B$ & $\notin \mathbb{P}$
if M loops → $L(M_1) = \emptyset \in \mathbb{P}$

Implications:

Any non-trivial question about semantics (meaning, what it computes) is undecidable

Behavior of a program nearly impossible

Example: $L = \{\langle M \rangle \mid$ TM M writes a $\perp$ to its tape on input $\varepsilon\}$ is undecidable!

- $L_{HALT} \leq L$: On input $\langle M, x \rangle$: Build a new TM M' from M that simulates M on x, which writes $\bar{\perp}$ instead of $\perp$ and only writes $\perp$ right before halting. Run decider for L on $\langle M' \rangle$.

Prove Decidability w/ Rice's

① Find $\mathbb{P}$ s.t. $L = L_{\mathbb{P}}$ ($\forall M \in L$, $L(M) \in \mathbb{P}$)
② Show $\mathbb{P}$ is trivial
③ Conclude L decidable.

Example 4.7. Let $L = \{\langle M \rangle : L(M) \subseteq \Sigma^*\}$. Show L is decidable by applying Rice's Theorem.

Solution: Observe that $L = \{\langle M \rangle : L(M) \subseteq \Sigma^*\}$. All languages (not just the recognizable languages) are a subset of Σ^*, so every machine should be a part of this set.
What we need to do is construct a $\mathbb{P}$ such that $L_{\mathbb{P}}$ is the set of all machines. If we take $\mathbb{P}$ to be the set of all languages U, then we should have $L_{\mathbb{P}} = \{\langle M \rangle : L(M) \in U\}$. To show $L_{\mathbb{P}} = L$

$$\langle M \rangle \in L \iff L(M) \subseteq \Sigma^* \iff L(M) \in U \iff \langle M \rangle \in L_{\mathbb{P}}.$$

Then we need to show $\mathbb{P}$ is trivial. Every language (not just the recognizable languages) is an element U, so all recognizable languages satisfy (or are contained in) $\mathbb{P}$.
Then by Rice's Theorem, L is decidable.