Computability

A Turing Machine (TM) is a mathematical model of computation.

\[ T = (Q, \Sigma, \Gamma, \delta, q_0, \{ \text{start, end} \}) \]

- \( Q \) is a finite set of states
- \( \Sigma \) is a finite alphabet
- \( \Gamma \) is a finite tape alphabet
- \( \delta \) is a transition function
- \( q_0 \) is the initial state
- \( \{ \text{start, end} \} \) are special states

A TM accepts a string \( x \) if it reaches a special state \( \text{accept} \) or \( \text{reject} \) on input \( x \).

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

We encode tape configurations of a TM with 3 pieces of info:
1. current state
2. symbol scanned
3. tape

The computation of \( M \) on input \( w \) is a series of configs \( C_0, C_1, \ldots \) (where \( C_0 = q_0 \) w)

\( M \) halts iff it reaches a halting config = \{ accepting config \}

Define: \( M \) is a decider iff it halts for all \( w \in \Sigma^* \)

\( M \) is an enumerator if it has a write-only tape (no input) and only points at characters

- \( \text{L ESD iff } 3 M, L = L(M) \)
- \( \text{L ED iff } 3 M, L = L(M) \) \( \forall M \text{ is a decider} \)

**Example:** \( \text{PAL} = \{ \omega \omega \mid \omega \in \{0,1\}^* \} \subseteq \Sigma \)

**Church-Turing Thesis (1936)**

Any 'algorithm' can be simulated by a TM.
Def: $P = \{L \mid L \in \text{TM}(\Sigma^*), 3c(\Sigma^*, T_M(m) \in O(m^2), L = \overline{L(M)}\}$

$SD = \{L \mid L \text{ semi}-decidable \}$

$D = \{L \mid L \text{ decidable } \}$

$\overline{SD} = \{L \mid L \not\in SD \land \overline{L \in SD} \}$

$\Sigma^*$

Cardinality of power set: all possible languages from fixed finite alphabet $\Sigma^*$

$\langle TM, \text{input} \rangle$: encoding of a TM in a language

$A_{TM} = \{\langle M, \omega \rangle \mid M \text{ is TM and } M \text{ accepts } \omega \}$

$All_{TM} = \{\langle M \rangle \mid M \text{ is TM and } L(M) = \Sigma^*\}$

$E_{TM} = \{\langle M \rangle \mid M \text{ is TM and } L(M) = \emptyset\}$

$H_{TM} = \{\langle M \rangle \mid M \text{ halts on } \omega\}$

$Diag = \{\langle M \rangle \in L(M)\}$

Closure:

$D$ is closed under $\land$, $\lor$ and $\neg$.

$SD$ is closed under $\land$, $\lor$ but not $\neg$. 

For example:

$A_{TM} = \{\langle M, w \rangle \mid M \text{ is TM and } M \text{ accepts } w\}$

$All_{TM} = \{\langle M \rangle \mid M \text{ is TM and } L(M) = \Sigma^*\}$

$E_{TM} = \{\langle M \rangle \mid M \text{ is TM and } L(M) = \emptyset\}$

$H_{TM} = \{\langle M \rangle \mid M \text{ halts on } \omega\}$

$Diag = \{\langle M \rangle \in L(M)\}$

Closure:

$D$ is closed under $\land$, $\lor$ and $\neg$.

$SD$ is closed under $\land$, $\lor$ but not $\neg$. 

Further discussion on the properties of these sets and their relationships to computability theory.
$\text{DIAG} = \{ \langle M \rangle \mid M \text{ is a TM } \land \langle M \rangle \notin \chi(M) \}$

Then $\text{DIAG}$ is not semi-decidable.

**Proof:** Suppose $\chi(M_0) = \text{DIAG}$

By contradiction: if $M_0$ accepts $\langle M_0 \rangle$ then $\langle M_0 \rangle \in \text{DIAG}$.

Corollary: $A_{TM}$ is not decidable.

$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM } \land M \text{ accepts } w \}$

$A_{TM}$ decidable $\implies$ $\text{DIAG}$ decidable.

**Church-Turing Thesis:**

No computer program solves $\text{DIAG}, A_{TM}$.

$\text{HALT}_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w \}$

Then $\text{HALT}_{TM}$ is not decidable.

**Proof:** Assume $\text{HALT}_{TM}$ is decidable.

Given $\langle M, w \rangle$, we can determine whether $M$ accepts $w$.

**Algorithm:**

1. Run $\text{HALT}_{TM}$ on $\langle M, w \rangle$.
2. If $M$ does not halt, accept $\langle M, w \rangle$.
3. If $M$ halts on input $w$, accept if $M$ accepts $w$.

Subject of decidable sets is not necessarily decidable.
Recall $A = \{ x \in \Sigma^* \mid x \notin \overline{A} \}$

Then $A$ is decidable iff $A \land \overline{A}$ are semi-dec.

Proof:

$\Rightarrow$

$\subseteq$ suppose $A \land \overline{A}$ are semi-dec

$\Rightarrow$ let $A = \mathcal{L}(M_1), \overline{A} = \mathcal{L}(M_2)$

so design a decider $M_3$ as such:

- on input $x$:
  - run $M_1, M_2$ on input $x$
  - accept if $M_1$ accepts $x$
  - reject if $M_2$ rejects $x$

Corollary $\overline{A_{TM}}$ is not semi-dec

Proof: be $\overline{A_{TM}}$ is dec $A$ is semi-dec.

Def $D = \{ A \subseteq \Sigma^* \mid A$ is decidable $\}$

SD = _______ semi-dec?

![Diagram showing $\Sigma^+$ with SD, $\overline{SD}$, and uncountable sets]

$R(x, y)$ is a relation on strings of pairs

$R: \Sigma^* \times \Sigma^* \rightarrow \{0, 1\}$

$\mathcal{L}_R = \{ <x, y> \mid R(x, y) = 1 \}$

Then $A \in SD$ iff $\exists R(x, y)$ decidable s.t.

$\forall x \in \Sigma^*, x \notin A \Rightarrow \exists y, R(x, y)$

$y$ is a certificate showing $x \in \overline{A}$

E.g. $A_{TM} = \{ \langle M, w \rangle \mid M$ accepts $w \}$

$R(x, y)$: suppose $x = \langle M, w \rangle$
Let \( y = c_0, c_1, \ldots, c_T \) be the computation of \( M \) on input \( w \). \( D \) must be an accepting config of \( M \) on input \( w \). Thus \( R(M, w, D) \) holds iff \( y \) codes an accepting config of \( M \) on input \( w \) (\( R \) is decidable).

If \( x \) is not of the form \( x = \langle M, w \rangle \) where \( M \) is a TM and \( w \in \Sigma^* \Rightarrow R(x, y) \)
\[
\begin{align*}
& x \in \text{TA}n \iff \exists y : R(xy) = 0 \\
& x \in \langle M, \_ \rangle
\end{align*}
\]

The certificate theorem

Let \( A \subseteq \Sigma^* \), \( A \in \text{SD} \) iff \( \exists x \) a decidable relation \( R(x, y) \) s.t. \( x \in A \Leftrightarrow x \in \text{TA} \Rightarrow \exists y : R(x, y) \)

When \( R(x, y) \) holds, we say \( y \) is a certificate that proves \( x \in A \).

1. \( \leq \) proof

Suppose that we have a decidable \( R \) satisfying \( x \in \text{TA} \iff \exists y : R(x, y) \)

We will construct a machine \( M \) such that \( A = L(M) \)

\[
\begin{align*}
\Sigma^* & = \{ w_0, w_1, \ldots \} \\
M \text{ on input } x = & \\
\text{for } i = 0 \text{ to } \infty & \\
\text{if } R(x, y) \text{ holds, then accept} & \\
\text{end for} &
\end{align*}
\]

2. \( \geq \) proof

Suppose \( A = L(M) \), for some TM \( M \). For any input \( x \), let \( C_0, C_1, C_2, \ldots, C_T \) be an accepting computation of \( M \) on \( x \).
Let \( y = \langle C_0, C_1, \ldots, C_t \rangle \) be our cerif.

Let \( R(x, y) \) holds iff \( y \) is an accepting computation of \( M \) on \( x \).

Then \( R \) is decidable remark the cerif. He gives another way of proving that a language is in SD.

\[ E_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset \} \]

Is \( E_{TM} \in SD ? \)

Let \( y = \langle C_0, C_1, \ldots, C_t \rangle \) be a cerif where \( C_0, C_t \) is an accept. comp. of \( M \) on \( x \),

Is \( E_{TM} \notin SD ? \) no use reduction

**Reducibility**

\[ A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \} \]

\[ \text{Diag} = \{ \langle M \rangle \mid M \text{ and } M \text{ is a TM} \} \]

\[ \langle M \rangle \notin \text{Diag} \Rightarrow \langle M \rangle \notin \text{Diag} \]

\[ \langle M \rangle \notin \text{Diag} \Rightarrow \langle M \rangle \notin \text{Diag} \]

\[ \langle M, \langle M \rangle \rangle \in A_{TM} \]

**Computable functions**

A function \( f : \Sigma^* \to \Sigma^* \) is comp if \( \exists \text{ a TM } H \) s.t. \( \forall y \in \Sigma^* \), \( H \) halts on input \( y \) and

Many-one reducibility (mapping reduc. in Sipser)

**Def.** Let \( L_1, L_2 \subseteq \Sigma^* \). We say \( L_1 \) is many-one reducible
to \( L_2 \), \( L_1 \leq_m L_2 \), if \( f \) a comp. func. s.t. \( f(x) \in L_2 \)

\[ \forall x \in \Sigma^*, x \in L_1 \implies f(x) \in L_2 \]

Historically, \( A \) and \( \Sigma \) came from the fact it was easier to reverse \( A \) and \( E \) on hypercubes.

Claim \( \text{DIAG} \leq_m A_{TM} \ f(\langle M \rangle) \)

define \( f(\langle M \rangle) = \langle \langle H \rangle, \langle M \rangle \rangle \)

Theorem \( L_1, L_2 \subset \Sigma^* \). Assume \( L_1 \leq_m L_2 \)

1. \( L_1 \leq_m L_2 \)
2. \( L_2 \notin D \implies L_1 \notin D \)
3. \( L_2 \notin D \implies L_1 \notin D \)

To prove -

Proof

1. \( L_1 \leq_m L_2 \implies \exists f : \Sigma^* \to \Sigma^*, x \in L_1 \implies f(x) \in L_2 \)

2. \( L_2 \notin D \implies L_1 \notin D \)

Assume \( L_2 \notin D \). Then call \( M_2 \) a TM s.t. \( L(M_2) = L_2 \) and \( M_2 \) always halts.

Since \( f \) is comp. there is a TM that compiles \( f \).

Define \( M_1 \) a follows:

- on input \( x \), run \( M \) on \( x \) to compute \( f(x) \)
- run \( M_2 \) on \( f(x) \), and accept or reject as \( M_2 \) does.

Clearly \( M_1 \) always halts and \( L(M_1) = L_1 \)

\[ \implies L_1 \leq D \]
3. Define $M_1: \text{on } x$, run $M$ to compute $f(x)$.
   - run $M_2$ on $f(x)$
   - if $M_2$ accepts, accept
      - rej

(similar to 2 besides $M_2$ might not halt)

Clearly $A(M_1) = 2$.

**Example** 

$\text{Dia} \leq_m A_{TM}$

- $\text{Dia} \not\in D \Rightarrow \text{Dia} \not\in D \Rightarrow A_{TM} \not\in D$
- $\text{Dia} \not\in SD \Rightarrow \text{Dia} \not\in SD \Rightarrow \text{Dia} \not\in A_{TM}$

Hence (by 3), $A_{TM} \not\in SD$

6. $\text{HB} = \{<M> | M \text{ is a TM and M halts on blank input} \}$

**Claim:** $\text{HB} \in SD \land \text{HB} \notin D$

**Proof:** We will show: $A_{TM} \leq_m \text{HB}$

Let $x \in \Sigma^*$, we assume $x = <M, w>$ where $M'$ is a TM.

Define $f(x) = <M'>$ where $M'$ works as follows:

- $M'$ write $w$ on the tape
- Run $M$ on $w$
- if $M$ halts accept, $M'$ accepts

Therefore, $M'$ loops forever.

Clearly $f$ is computable.

Also: $M$ accepts $w \Rightarrow M'$ halts $\Rightarrow <M'> \in \text{HB}$

$<M, w> \notin A_{TM} \Rightarrow M'$ doesn't halt

$<M', w> \notin \text{HB}$

**Corollary:** $\text{HB} \in SD$

**Proof:** Suppose $\text{HB} \in SD$, hence since $\text{HB} \in SD$

we have $\text{HB} \in \text{ED}$, contradict.
Claim \( E_{TM} = \{ \langle M \rangle | M \text{ is a TM and } L(M) = \emptyset \} \)

\( \emptyset \cap E_{TM} \cap SD \cap \overline{SD} \)

Proof. \( \emptyset \) show that: \( \overline{HB} \leq_m E_{TM} \) (or \( \leq_m \))

Define \( f(\langle M \rangle) = \langle M' \rangle \) where:

on input \( x \):
- if \( x \) is not the blank tape, rej.
- if \( x \) is the blank tape, run \( M \)
  on \( x \) and accept if it halts.

Hence: \( \langle M \rangle \in \overline{HB} \Rightarrow M \text{' accepts the blank tape } \)

\( \Rightarrow M' \in E_{TM} \)

\( \Rightarrow f(\langle M \rangle) \in E_{TM} \)

Example \( A_{OB} \) = set of all C programs \( \langle P \rangle \) s.t. \( P \) causes an array out of bound, res.

Recall

\[ \text{coSD} = \{ A \subseteq \Sigma^* | A \notin SD \} \]

\[ \text{diag} A_{TM} \cap \overline{HB} \cap E_{TM} = D \]

\[ \text{diag} \overline{A_{TM}} \cap \overline{HB} \cap E_{TM} \]

Theorem

1. Let \( A, B, C \subseteq \Sigma^* \). \( A \leq_m B \leq_m C \Rightarrow A \leq_m C \)
2. Let \( A \subseteq \Sigma^* \). \( A \notin \text{SD} \land A \notin \overline{A} \Rightarrow A \in \text{D} \)

Example of (1): \( A_{TM} \leq_m \overline{HB} \leq_m \overline{E_{TM}} \)
proof (1)

Assume A ≤_m B via computable f: \Sigma^* \to \Sigma^*

and \lambda \leq_m C

gof is computable:

\forall \text{TMs} \ 	ext{that comp f and g}

\forall x \in A \iff f(x) \in B \iff g(f(x)) \in C

\iff gof(x) \in C

proof (2)

A \leq_m \overline{A} \iff \overline{A} \leq_m A

\forall \text{words of TMs}

A \in ESD \land \overline{A} \leq_m A \iff \overline{A} \in ESD

A \in ESD \land \overline{A} \in ESD \iff A \in E

Define \ T_m = \{ \langle M \rangle | M \ is \ a \ TM \ and \ M \ is \ a \ decider \}

Claim \ \overline{T_{ESD}} \land \overline{T_{ESD}}

proof @ show that \ \overline{HB} \leq_m T_T \iff \overline{HB} \leq_m \overline{T_T}

define: \ f(\langle M \rangle) = \langle M' \rangle \ where

\{ \begin{align*}
\text{if input} & \text{ is blank} \ M' \ rejects \\
\text{if } x & \text{ is blank and } M \ on \ it
\end{align*}

\iff <M> \in NB \iff M \ halts on blank tape

\iff M' \ halts on all inputs

\iff <M'> \in T_T

@ show that \ \overline{HB} \leq_m \overline{T_T}

define: \ f(\langle M \rangle) = \langle M' \rangle \ where \ \{ \begin{align*}
\text{if } M \ halts \ within \ x \ steps, \ M' \ loops \\
\text{if } x \ 	ext{steps, } M' \ loops
\end{align*}

on input x; \ \Rightarrow \ x \text{ steps, } M' \ 	ext{loops} \}

\iff \overline{M} \ halts on blank tape

\iff \overline{M} \ halts on blank tape

\iff \overline{M} \ halts on blank tape
Thus: \( M \) doesn't halt on blank tape \( \Rightarrow M' \) halts on all input
\[ \Rightarrow (M') \in T_{TM} \]
\[ \Rightarrow \overline{(M')} \notin \overline{T_{TM}} \]
\[ \Rightarrow M \text{ halts on blank tape in } T \text{ steps} \]
\[ \Rightarrow (M') \text{ doesn't halt} \quad \forall x, |x| \geq 1 \]
\[ \Rightarrow M' \text{ is not total} \]
\[ \Rightarrow (M') \in \overline{T_{TM}} \]

Define \( \text{All}_{TM} = \{ \langle M \rangle | \forall x, \langle M \rangle \in \overline{T_{TM}} \} \)

Claim: \( \text{All}_{TM} \notin \text{SD} \) \( \land \overline{\text{All}_{TM}} \notin \text{SD} \)

proof: show \( T_{TM} \subseteq \text{All}_{TM} \)

define \( f(\langle M \rangle) = \langle M' \rangle \) where \( M' \) on \( x \)

accepts or rejects

- run \( M \) on \( x \)
  - if \( M \) halts, then \( M' \) accepts
  - otherwise, \( M' \) rejects

then: \( \langle M \rangle \in \overline{T_{TM}} \) \( \Rightarrow \exists \forall x \in \text{input}, M \text{ halt} \)
\[ \Rightarrow \overline{\text{All}_{TM}} \]
\[ \Rightarrow M' \in \text{All}_{TM} \]

\[ \Rightarrow \overline{\text{All}_{TM}} \notin \text{SD} \]

since \( T_{TM} \subseteq \text{All}_{TM} \), we have \( T_{TM} \subseteq \overline{\text{All}_{TM}} \)

since

Define \( \text{EQ}_{TM} = \{ \langle M \rangle | \} \)

\[ \text{Define \( \text{EQ}_{TM} = \{ \langle M \rangle | \} \} \]
Chomsky Hierarchy

<table>
<thead>
<tr>
<th>rec. enum.</th>
<th>TM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSL</td>
<td>PDA = NPDA</td>
</tr>
<tr>
<td>CFL</td>
<td>FS = NFSA</td>
</tr>
</tbody>
</table>

Finite automaton:

\[ M = \{ \Sigma, Q, \delta, q_0, F \} \]

At each step, head moves right one step. \[ \delta(q, a) = q' \]

\( M \) accepts \( w \) iff it reaches \( FF \) after scanning last symbol.

\[ \text{REG} = \{ L(N) \mid M \text{ is a FSA} \} \]

\( \text{REG} \neq \text{CFL} \)

E.g. PAL \# REG (pumping lemma)

NFSA can be converted to FSA w/ exp run of states.

[Box: REG is closed under \( U \), \( \cap \), -]

Pushdown automaton:

\[ M = \{ \Sigma, Q, S, F \} \]

\[ \delta(q, a, b) = (q', \text{pop} or \text{pop}) \]

CFL = \( \{ L(M) \mid M \text{ is a PDA} \} \)

DCFL \( \neq \) CFL \( \land \) DCFL = \( \{ L(M) \mid M \text{ is a DPDA} \} \)

DCFL is closed under \( - \) but not \( U, \cap \)

Proof: take opposite DPDA

\[ \neg \neg \neg : L_{ab} \cap L_{\neg ab} \in \text{DCFL} \]

\[ L_{\neg ab} \cap L_{\neg ab} = L_{\neg ab} \cap \neg L_{ab} \in \text{CFL} \]

U: De Morgan \( w \) - and \( \neg \)
All\( \text{CFG} = \{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \Sigma^* \} \)

**Claim:** All\( \text{CFG} \not\in \text{SD} \)

**Proof:** show \( \text{HP} \leq_m \text{All\( \text{CFG} \)} \)

- Given a TM \( M \), construct a CFG \( G \) s.t. \( M \) halts on a blank tape iff \( L(G) \not= \Sigma^* \)
- Idea: design \( G \) s.t. \( L(G) \) is the set of all strings which do not code a halting of \( M \) on a blank tape.

Let \( C_1, \ldots, C_t \) be a halting comp. \( \psi \), where \( C_i = x \# y \# \# \#
\)

\( L_1 = 1 \# \) and last config one ok (REG)

\( L_2 = \text{even} \) trans: one ok (DCFL)

\( L_3 = \text{odd} \) trans: \( \Sigma^* \) (DCFL)

\( L_1 \cap L_2 \cap L_3 = \begin{cases} \emptyset & \text{if } M \text{ loops} \\ \{ \text{halting comp} \} & \text{if } M \text{ halts} \end{cases} \)

\( L_1 \cup L_2 \cup L_3 \) is \( \Sigma^* \) (DCFL)

---

Most of the line, when a model of computer accepting lang is closed under its unary be we can easily charge accept. Set of sets \( S, \{ S \} > 2 \) states. Russel: consider a set with...
All CFG

Given a TM \( M \), find a CFG \( G \) s.t. \( M \) doesn't halt on a blank tape iff \( L(G) = \Sigma^* \)

Find \( \epsilon' \) over \( \Sigma' = \Sigma \cup \{\#\} \)
Use ASCII codes for \( \epsilon' \)s in \( \Sigma' \)

\[ \text{comp} = \{ L_1 \cap L_2 \cap L_3 \} \text{ s.t.} \]
\[ L_1 \cap L_2 \cap L_3 = \{ \emptyset \text{ if } M \text{ doesn't halt} \}
\]
\[ \text{or } \{ \Sigma \} \text{ if } \epsilon \]

With ASCII codes, \( \text{comp} \leq \Sigma^* \)

Comprehensibility redux

Decidability relies on our model of (universal) comp: TM

Church-Turing Thesis -

Undecidable pts

Diophantine equations

Axiomatizing arithmetic - Gödel's incompleteness

Use fact that \( \overline{\text{HB-ESD}} \)

\( \psi(x) \) asserts \( x \) codes a halt. comp

Give a TM ... \( \psi(x) \) assert codes a halt. comp

Use \( \overline{\psi(x)} \)

If every have such statement has a finite proof then \( \overline{\text{HB-ESD}} \)

Entscheidungs pb

Church & Turing

S. Cook

Google for spelling
WCTime complexity: Let $M$ be a TM decider.

\[ T_M(n) = \max \{t_M(x) \mid x \in \{0,1\}^n\} \]

Def
\[ \text{TIME}(1(n)) = \{A \subseteq \Sigma^* \mid \exists N \text{ a TM with } L(N) = A \text{ and } T_M(n) \leq O(1(n))\} \]

Def
\[ P = \bigcup_k \text{TIME}(1^n) \]

E.g. $\text{SQ} = \{\langle a^n \rangle \mid a \in \{0,1\}, a^2 \text{ in binary}\}$

$\text{SQ} \in P \rightarrow$ Newlow's method: $x_{m+1} = \frac{1}{2}(x_m + \frac{a}{x_m})$

Def $\text{NP} = \{\langle x, y \rangle \mid \exists R(x, y) \text{ poly} p(n)$

E.g. $\text{HAMPATH} = \{\langle G, s, t \rangle \mid G \text{ is an undirected graph with a Hamiltonian Path from } s \text{ to } t\}$

$\text{CLIQUE} = \{\langle G, k \rangle \mid$

Def $\langle G, k \rangle$ is $k$-colourable if there is a function $c: V \rightarrow \{1, \ldots, k\}$ s.t. if $(u, v) \in E$, then $c(u) \neq c(v)$.
Every $NP$-complete problem $A$ has an associated search problem $A$-search.

Let $A \in NP$, so $x \in A \Rightarrow \exists y \left( |y| \leq p(|x|) \text{ and } R(x, y) \right)$

$A$-search

- instance: $x \in \Sigma^*$
- output: $y \in \Sigma^*$ s.t. $|y| \leq p(|x|) \land R(x, y)$ or 'no' if no such $y$ exists

Let $P_1, P_2$ be problems (search or decision)

$P_1 \leq P_2$ iff 3 polytime alg which solves $P_1$ which can ask questions to an 'oracle' which solves $P_2$ (do not count time required by $P_2$)

Theorem: $A \in NP$-complete then $A$-search $\leq A$

e.g. $HAMPATH$-search $\leq HAMPATH$

Here's the reducing algorithm:

input: $<G, s, t>$
oracle: boolean procedure $HP(H)$ which solves $HAMPATH$
alg: if $HP(G) = 0$ then output 'no'$
for $i = 1 \ldots m$
if $HP(H - e_i) \land e_i \not\in G$
then $H \leftarrow H - e_i$
Connectness: invariant $G$ has a HAMPATH from $s$ to $t$ using every edge in $E_{HAMP}(G)$

Def $A \leq_p B$ (Karp reducible)

If $\exists$ polytime func $f : \Sigma^* \to \Sigma^*$ s.t. $x \in A \iff f(x) \in B$

Note 1: $A \leq_p B \implies A \leq_{\text{m}} B$

$A \leq_p B \implies A \leq_{\text{p}} B$

Def $A \in \text{NP-hard}$ iff $\exists B \in \text{NP}, B \leq_p A$

Def $A \in \text{NPC}$ iff $A \in \text{NP} \land A$ is NP-hard

Lemma 1: $A \leq_p B \land B \in \text{P} \implies A \in \text{P}$

$A \leq_{\text{m}} B \land B \in \text{D} \implies A \in \text{D}$

$\text{FP}$ a class of polytime computable functions

Corollary: $A \in \text{NPC} \land A \in \text{P} \implies P = \text{NP}$

---missed lecture---
\(A \text{ is } \text{NP-land} \iff B \leq_p A \land B \in \text{ENP} \)

\(A \in \text{NPC} \iff A \text{ is } \text{NP-land} \land A \in \text{ENP} \)

\text{Cook-Levin's Theorem:} \quad \text{SAT} \in \text{NP}

\( \text{SAT} = \{ \langle \varphi \rangle \mid \varphi \text{ is boolean formula. SAT} \} \)

\text{Lemma:} \quad A \in \text{NPH} \land A \leq_p B \implies B \in \text{NPH}

"A is hard and A is reducible to B, so B must be hard."

\text{Proof:} \quad \leq_p \text{ is transitive}

\(\text{Def:} \quad \text{CNF} = \{ \langle \varphi \rangle \mid \varphi \text{ is in CNF} \} \)

\(k \text{CNF} = \{ \langle \varphi \rangle \mid \varphi \text{ is in CNF, } s.t. \text{ each clause has at most } k \} \)

\(\neg k \text{SAT} = \{ \langle \varphi \rangle \mid \varphi \text{ is in } k \text{CNF } \land \varphi \text{ is SAT} \} \)

Then \(3 \text{SAT} \in \text{NPC} \)

show \(\text{SAT} \leq_p 3 \text{SAT} \)

\text{Proof: Idea:} \quad \text{introduce a new literal (negate some variables)}

for every sub-formula of \(\varphi\)

\(\text{such that for every such } \alpha: \quad \alpha = (H \lor V \land \neg A)\), construct a

\(D_\alpha \equiv x_\alpha \leftrightarrow (x_H \lor x_V \land \neg A)\)

... Then define \(\varphi' = D_\alpha \land D_B \ldots\)

\(\text{Proof:} \quad \text{show } \varphi \leftrightarrow \varphi' \)

\(\text{sup } \forall \text{ sat } \varphi', \text{ for every this form } \forall \text{ sat}

\alpha = x_\alpha \land \neg \varphi \equiv \varphi \text{ under } \gamma \)

\(\Rightarrow \text{sup } \forall \text{ sat } \varphi' \quad \text{then there is an extension } \gamma' \text{ s.t. } \gamma' \text{ sat } \varphi'\)
\[ \text{IND-SET} = \{ \langle G, k \rangle \mid G \text{ is undirected with IS of size } k \} \]

\[ \text{Def: } V' \subseteq V \text{ is IS of } G = (V, E) \iff u, v \in V' \Rightarrow (u, v) \notin E \]

\[ \text{In ISENPC} \]

- Show \( 3\text{SAT} \leq_m \text{IND-SET} \) (Cook 1971)

**Proof.** Assume \( \Phi \) has exactly 3 literals per clause \( \forall \Phi \in \text{CNF} \)

\[ \Phi = \bigwedge C_1 \wedge \ldots \wedge C_m, C_i = (l_{i,1} \lor l_{i,2} \lor l_{i,3}) \]

\[ \forall \phi \in \Phi, \phi \text{ contains exactly 3 literals} \]

\[ V_\Phi = \{ \langle i, j \rangle \mid 1 \leq i \leq m, 1 \leq j \leq 3 \} \]

\[ \overline{t_{ij}} \text{ occurrences of literals in } \Phi \]

\[ |V_\Phi| = 3m \]

Let \( k_{\Phi} = m \)

\[ E_1 = \{ \langle i, j \rangle, \langle i, k \rangle \mid 1 \leq i < m, 1 \leq j, k \leq 3 \} \]

\[ E_2 = \{ \langle i, j \rangle, \langle k, i \rangle \mid 1 \leq i, j, k \leq 3 \} \]

\[ E_\Phi = E_1 \cup E_2 \]

\[ G_\Phi = (V_\Phi, E_\Phi) \text{ has an } \text{IS} \text{ of size } m \text{ iff } \Phi \text{ is SAT} \]

\( \Rightarrow \) if \( \Phi \) SAT

\( \Leftarrow \)

- Suppose \( G_\Phi \) has an IS \( V' \) of size \( m \). Then \( V' \) must have exactly 1 literal per clause.

- Choose \( \Gamma \) to make all these literals \( 1 \).

\[ \text{CLIQUE} = \{ \langle G, k \rangle \mid G \text{ has clique of size } k \} \]

\[ \text{In CLIQUE-ENPC} \]

**Proof.** CLIQUE-ENP since given a \( \langle G, k \rangle \) check that \( |V'| = k \) in polytime

\( 2 \) show \( \text{IND-SET} \leq_m \text{CLIQUE} \)
Suppose \( V' \subseteq V \). Then \( V' \) is an \( \mathbf{IS} \) of \( G \) if and only if \( V' \) is a \text{CLIQUE} in the complement of \( G \), i.e., \( \overline{G} \).

\[
\overline{G} = \{ (V, \overline{E}) \mid (u, v) \in \overline{E} \iff u \neq v, (u, v) \notin E \}
\]

**VERTEX-COVER** = \( \{\langle G, k \rangle \mid G \) is an undirected graph with a vertex cover of size \( k \} \)

\( V' \subseteq V \) is a VC for \( G \) if and only if \( \forall (u, v) \in E \exists u \in V' \forall v \in V' \)

Then **VERTEX-COVER \( \leq \text{p} \) SAT**

show **IND-SET \( \leq \text{p} \) VERTEX-COVER**

Let \( \forall u \in V' \forall v \notin V' \)

**HAMPATH** = \( \{\langle G, s, t \rangle \mid G \) is a directed graph with a hamiltonian path from \( s \) to \( t \} \)

Then **HAMPATH \( \leq \text{p} \) SAT**

proof sipser

Claim **HAMPATH \( \leq \text{p} \) HAMPATH**

**IDEA** for each \( u \in V \) \( u \sim u \)

Back to **Cook–Levin**

**SAT \( \leq \text{p} \) SAT**

\( \forall L \in \mathbf{NP}, L \leq \text{p} \) SAT

proof

Assume \( A \in \mathbf{NP} \)

Given \( x \in \Sigma^* \), find \( y \) in polytime s.t. \( x \in A \iff y \in \text{SAT} \)

By def, \( A \in \mathbf{NP} \) \( \Rightarrow \) 3 polytime rela. \( R(x, y) \land \text{poly } p(n) \)

\( x \in A \iff \exists y \ (|y| \leq p(|x|) \land R(x, y)) \)
Let $M$ a TM which accepts $R(x, y)$ in time $T(n)$, $n = \log y$.

Input to $M$ is $x, y$, assume wlog that $\log y = \rho(12x)$.

$x \notin A$ iff $3y_1 y_2 \ldots y_{\rho(\log x)}$ s.t. comp $C_0 \ldots C_{\rho(\log x)}$ of $M$ (with $c_0$)

and $C_{\rho(\log x)}$ is an accept state with $T(m) = c_{\infty}$ (for some $c, k$).

Variables in $\Psi_{\infty}$ are such that describe $\Omega_{\infty}$:

- state set $Q = \{q_0 \ldots q_{\rho(\log x)}\}$ for each $C_t$
- head pos for each $t$
- tape contents for each $t$

\[ \Psi_{\infty} = \Psi_1 \wedge \Psi_2 \wedge \Psi_3 \wedge \Psi_4 \wedge \ldots \wedge \Psi_7 \]

Define:

\[ \text{Unique} (P_1, \ldots, P_k) = \text{exactly 1 of } P_1, \ldots, P_k \text{ is true} \]

\[ \text{Unique}(P_1, \ldots, P_k) \equiv \bigwedge_{i \neq j} \overline{(P_i \lor P_j)} \]

\[ \Psi_1 = \bigwedge_{t=0}^T \text{Unique}(q_{0t}, \ldots, q_{tt}) \]

\[ \Psi_2 = \bigwedge_{t=0}^T \text{Unique}(h_{0t}, \ldots, h_{tt}) \]

\[ \Psi_3 = \bigwedge_{t=0}^T \text{Unique}(z_{jt}, u_{jt}, b_{jt}) \]

E.g., initial config:

$0110b y_1 \ldots y_{\rho(\log x)} \Rightarrow \Psi_4 = q_{00} \wedge z_{00} \wedge u_{10} \wedge u_{20} \wedge z_{30} \wedge u_{40} \wedge z_{50} \wedge \ldots$
\[ \psi_5 \equiv q_f \text{ - end i wan accept state} \]

\[ \psi_6 \equiv \bigwedge_{i=0}^{T-1} \bigwedge_{j=0}^{T-1} \left( \Gamma_{i,j} \rightarrow (2i+1, b_{i,j}) \wedge (2j+1, \Gamma_{i,j+1}) \right) \]

- all tape symbols are unchanged except
the scanned square -

\[ \psi_f - \psi_i - \text{transition function changes} \]

\[ \delta(q_{s,f}, 1) = (q_{s,f}, 0, R) \]

\[ q_{s,f+1} \equiv 2 \cdot i + \Gamma_{i,j+1} \]

**Self-reducibility**

\[ \text{AENPC } \Rightarrow \text{ A-search } 5 \text{ A } = \{ x \mid \exists y, |y| \leq \rho(|x|), V_A(x, y) = \text{yes} \} \]

**poly alg**

\[ C \in \mathbb{E} \]

while \( V_A(x, C) = \text{"no"} \)

if \( C \) is a prefix of a certif

\[ C \subseteq C \]

else \( C \subseteq C_0 \)

**renew C**

**Theorem**

\[ 3 \text{COLOR} = \{ (G) \mid G \text{ is 3colorable} \} \]

show \( 3 \text{SAT } \equiv \ 3 \text{COLOR} \)

\[ x, y, z \rightarrow U - F \]

\[ x \sim \bar{x} \]

\[ y \sim \bar{y} \]

\[ z \sim \bar{z} \]
Space Complexity

Let $M$ be a TM, $x \in \mathbb{Z}^+$, $S_M(x)$: number of tape squares scanned in the computation of $M$ on $x$.

$$S_M(x) = \max\{S_M(x') \mid |x'| = n\}$$

$$\text{SPACE}(f(n)) = \text{DSPACE}(f(n)) = \{A \in \Sigma^* \mid S_M(A) \in O(f(n))\}$$

**Def** $\text{PSPACE} = \bigcup_k \text{DSPACE}(n^k)$

**Thm** $\text{DTIME}(f) \subseteq \text{DSPACE}(f)$ proof obvious

$\Rightarrow P \subseteq \text{PSPACE}$

**Thm** $\text{NP} \subseteq \text{PSPACE}$

Proof: $A \in \text{NP}$ iff $x \in A \iff \exists y, 1^y \leq p(m) \land R(x, y)$

Since $1^y \leq p(m)$, we can use brute force to find $y$.

$\Rightarrow \text{NP} \subseteq \text{PSPACE}$
Space Complexity

TQBF

- quantified Bool. form.

QBF \( \phi \) is a sentence if \& only if it has no free vars.

TQBF = \{ \phi \mid \phi \text{ is a true QBF sentence} \}

Claim SAT \leq_p TQBF

given sat \( \psi(x_1, \ldots, x_m) \), construct \( \psi' = 3x_1 \ldots x_m, \phi(x_1 \ldots x_m) \)

Then TQBF is PSPACE-complete.

Def Truth of a QBF (in prenex form)

\[ \psi(x_1 \ldots x_m) = Q_1 x_1 \ldots Q_m x_m, \phi(x_1 \ldots x_m) \]

Induc

Base: \( n = 0 \), \( \psi(\emptyset) = \psi(\text{wwf}(0, 1, \lambda, v)) \)

I. step: case 1: \( Q_1 = 3 \)

\[ \implies \psi = Q_2 \cdot Q_{m+1} x_{m+1} \cdot \phi(0, x_2 \ldots x_m) \]

\[ \lor \psi(1, x_2 \ldots x_m) \]

, case 2: \( Q_1 = A \)

\[ \implies \psi = \ldots \land \ldots \]

Since we can build such an alg. that will eliminate quantifiers every time, we only need linear space with brute force (EXPTIME)

\[ \implies \text{TQBF} \in \text{L, PSPACE} \]
Show $\text{TQBF} \in \text{PSPACE}$ hand:

Let $\text{AEPSPACE}$, $A = \mathcal{L}(M)$

Now $x$ has a comp using space $O(n^k)$

given $x \in \Sigma^*$, we must construct in poly-time

a QBF sentence $q_x$ s.t. $M$ accepts $x$ iff $q_x$ is true.

$\text{Def } L = \text{DSPACE}(\log n)$; $\text{NL} = \text{NPSpace}(\log n)$

---

**Path** in NL

proof: How imp $\langle c, s, t \rangle$

it guesses $s = v_1, v_2, \ldots, v_k = t$

checks $(v_i, v_{i+1}) \in E$, $1 \leq i < k$

Note: we may assume a log-space TM has finitely many symbols

---

No certificate theorem for NL

Since you would get a certificate for all

of NP. —Cook
Claim \( \text{PATH} \in \text{NL} \)-complete under \( \leq_L \)

\( \text{L} = \text{NL} \iff \text{PATH} \in \text{L} \)

Claim \( \text{L} \subseteq \text{P} \)

Let \( M \) be a logspace TM. \( \text{conf} \text{ of } M \) on \( \langle x \rangle \), \( |x| = n \)

\[ \text{conf} = \langle j, q, u \rangle \]

leads to \( c = \langle j, q, u \rangle \) on work tape

\[ 0 \leq j \leq n+1 \implies 3 \cdot O(m^k) \text{ possible configurations} \]

**Def Logspace reductions**

\[ \text{FL} = \{ f : 2^k \rightarrow 2^k \mid f \text{ comp in } O(\log) \text{ space} \} \]

note model has extra W-only output

\[ f \in \text{FL} \implies |f(x)| = O(1) \text{ for some } k \]

Because \( M \) must halt in time \( O(n^k) \)

E.g.

\[ f_+ (x, y) = x + y \quad f_\times (x, y) = x \cdot y \]

**Def** \( A \leq_L B \iff \exists f \in \text{FL} x \in A \iff f(x) \in B \quad (\forall x \in 2^k) \)

Claim \( A \leq_L B \implies A \leq_p B \quad \text{be } \text{FL} \subseteq \text{P} \)

Claim \( \leq_L \) is transitive \( \text{be } \text{FL is closed under composition} \)

**Proof** \( f \circ f \in \text{FL} \quad f \circ f \in \text{FL} \)

**Side note:** probabilistic TM

\( \text{BPP} = \text{P} \text{ for } \text{pTM} \)

\( \text{BPNP} = \text{NP} \text{ for } \text{pTM} \) (hard to think about?) probdist

\( \text{BPNP} = \text{AM} \) def Arthur-Merlin protocol \( \text{DTM w/ random tape} \)
Savitch's theorem

- D. Cook, 1st PhD student, 1970

\[ \text{NL} \leq \text{DSPACE} \left( \log^2 n \right) \]

Suffices to show \( \text{PATH} \in \text{DSPACE} \left( \log^2 n \right) \)

(since \( \text{PATH} \) is \( \text{NL} \)-complete, i.e. \( \text{AENL} \Rightarrow \text{A} \subseteq \text{PATH} \))

Lemma: \( \text{A} \subseteq \text{B} \subseteq \text{DSPACE} \left( \log^2 n \right) \Rightarrow \text{A} \in \text{DSPACE} \left( \log^2 n \right) \)

Proof:

Fact: if \( G \) is an undirected graph, \( G \) has a path of length at most \( 2l \) from \( u \) to \( v \)

iff \( 3 \in \text{EN} \), \( G \) has \( 2 \) paths from \( u \) to \( w \) and \( w \) to \( v \), each of length \( \leq l \).

Idea: use \( D \& \text{FC} \), \( \text{W} = \log \text{n} \)

Alg.

Boîl procedure \( \text{Path} \left( G, u, v, l \right) \)

holds iff \( G \) has \( \text{Path} \left( u, v, \leq 2l \right) \)

if \( l = 0 \)

and \( u = v \), then \( \text{ACCEPT} \)

else for \( w \in \text{V} \)

if \( \text{Path} \left( G, u, w, l-1 \right) \) and

\( \text{Path} \left( G, w, v, l-1 \right) \) then \( \text{ACCEPT} \)

end for

\( \text{REJECT} \)

Space analysis: nesting depth \( : l \)

- space per call \( : O \left( \log \left( V^l \right) \right) \)

- total space \( : O \left( l \cdot \log^2 V \right) \)

\( \Rightarrow \) To solve \( \text{PATH} \), call \( \text{Path} \left( G, s, t, \lceil \log_2 m \rceil \right) \)
Def. $f(n)$ is space constructible iff $1^{\leq f(n)} \in \text{space } O(f(n))$.

- $f(n)$ can be written in unary/binary.
- If $f(n)$ is space constructible, then $f(n)$ can be written in space $O(f(n))$.
  - $(n)_{\text{bin}} \in O(\log n)$

General Savitch

$f(n)$ is space constructible ($\exists f(n) \geq \log_2 n$) $\implies$ NSPACE($f(n)$) $\subseteq$ DSPACE($f^2(n)$)

1st method: Assume $A \in \text{NSPACE}(f(n))$.

- Let $A = \mathcal{L}(M)$, $M$ is a nondet $O(f(n))$ TM.
- Use the configuration graph for $M$ on $x$.

2nd method: Padding argument:
  - def: $\text{pad}(w, 1) = \{w \# 1 \mid \exists z \geq 2$ s.t. $\text{pad}(A, f) = \{\text{pad}(w, f(|w|) \mid w \in A\}$

Assume $f(n)$ is sp. cons. $A \in \text{NSPACE}(\log n)$.

- $B = \text{pad}(A, 2f(n)) \in \text{NL}$
- By special Savitch, $B \in \text{DSPACE}(\log^2 n)$

Thus: $A \in \text{DSPACE}(f(n)^2)$
Recursion Theorem

\[ \exists q : \Sigma^* \rightarrow \Sigma^* \quad \text{such that outputs } w \]

Q: on \( w \), print \( \langle P_w \rangle \) where \( P_w : \text{on } x, \text{reset } x, \text{halt} \)

\[ \text{SELF: } \begin{array}{c}
\begin{array}{c}
A \\
B \rightarrow \top
\end{array}
\end{array} \]

\[ \begin{array}{c}
\text{SELF = } \quad \begin{array}{c}
\text{utilizes } A \text{ points to } \langle B \rangle \text{ and } \langle A \rangle \\
\text{B computes } q(\langle B \rangle) = \langle A \rangle \\
\text{paradox}
\end{array}
\end{array} \]

Build a \( T \) that can compute \( w \) as its description (avoiding selfref)

\[ \begin{array}{c}
\text{R: } \begin{array}{c}
\begin{array}{c}
A \\
B \rightarrow T
\end{array}
\end{array}
\end{array} \]

\[ \begin{array}{c}
\text{R: } \text{print } x
\end{array} \]

\[ \begin{array}{c}
\text{B: } \text{compute } q(\langle B \rangle) = \langle A \rangle \\
\text{paradox}
\end{array} \]

Space Hierarchy

For any space cons. \( f : \mathbb{N} \rightarrow \mathbb{N} \), \( \exists A \) decidable in \( O(f(n)) \) space but not decidable in \( o(f(n)) \) space

**Proof Idea:**

- Build \( D \) using diagonalization
- \( D \) on input \( \langle M \rangle \): \( D \) runs \( M \) on \( \langle M \rangle \) within bound
  - If \( M \) halts within \( f(n) \), do the opposite of \( M \)
  - If \( M \) uses none space, do not halt
Technical issues

1. \( M \) runs in \( o(f(n)) \) space may use none than \( f(n) \) space for small \( n \) when asymptotic bad kicks-in -
   - accept \( \langle M \rangle 1^k \) as input

2. \( M \) might loop forever
   - if it runs in \( o(f(n)) \) space, it will use at most \( 2^o(f(n)) \) time - keep a counter

D) on \( w \): if \( w \neq \langle M \rangle 1^k \) for some \( M, k \)
   - let \( n = |w| \), since \( f(n) \) is space cons.
     - mark off \( f(n) \) space. If later stops
       - simulate \( M \) on \( w \) w/ tape-compress reject
         - iff \( M \) accepts on counter > 2 \( f(n) \)
         - else accept.

Corollary ① (if space cons. \( \forall g \in o(f) \implies DSPACE(g) \neq DSPACE(f) \))

② \( \forall \varepsilon_1, \varepsilon_2, 0 \leq \varepsilon_1, \varepsilon_2 \implies DSPACE(m^{\varepsilon_1}) \neq DSPACE(m^{\varepsilon_2}) \)

③ \( NC \neq PSPACE \)
   - proof by Savitch. \( NC \subseteq DSPACE(\log^2 n) \)
     - By SETH, \( DSPACE(\log^2 n) \neq DSPACE(n) \)

④ \( PSPACE \neq EXPSPACE \)
Question 3

We show that \( \text{SET-PARTITION} \) is NP-complete.

1. Given a certificate for \(<S_1, \ldots, S_m, k> \in \text{SET-PARTITION}\), a subset \( S \subseteq \{1, \ldots, m\} \) of size \( k \) such that the sets \( S_i \) corresponding to elements \( i \in S \) are pairwise disjoint. Clearly, we can verify in polynomial each \(|C| \times (|C| - 1)\) pair for empty intersection, and that \(|C| = k\). Hence \( \text{SET-PARTITION} \in \text{NP} \).

2. show \( \text{IND-SET} \leq^P_m \text{SET-PARTITION} \)

Given a graph \( G = (V, E) \), we build \( n = |V| \) sets \( S_i \) (\( 1 \leq i \leq m \)), such that each corresponds to a node in \( G \), and contains every edge incident to that node.

Thus, \( G \) will have an independent set of size \( k \) iff we have \( k \) nodes (each corresponding to an \( S_i \)) such that no two nodes share an edge (every \( S_i \) pair is disjoint).

By (1) and (2), \( \text{SET-PARTITION} \) is NP-complete.
Question 4

We show $\mu \text{HAMPATH} \leq^p \mu \text{HAMCYCLE}$.

Given a graph $G = (V, E)$ and $s, t \in V$, we construct a graph $G' = (V', E')$ where $V' = V \cup \{v\}$ ($v \notin V$ is an extra node) and $E' = E \cup \{v, s\} \cup \{v, t\}$.

Hence, if $G$ has a Hamiltonian path from $s \rightarrow t$, then the new graph $G'$ has a Hamiltonian cycle from $s \rightarrow v \rightarrow s$.

The converse holds since $v$ is only connected to $s$ and $t$, hence every Hamiltonian cycle has to include $(s,v)$ and $(v,t)$, thus there must be a Hamiltonian path from $s \rightarrow t$.

10/10 Correct.