Queens' College Cambridge

Complexity Theory

Alistair O'Brien

Department of Computer Science

May 16, 2021

Contents

1 Background

1.1 Asymptotic notation

Definition 1.1.1. (O-notation) For a given function $g : \mathbb{N} \to \mathbb{N}$, we denote $O(g(n))$ as the set of functions

$$O(g(n)) = \{f(n) : \exists N, b > 0. \forall n > N. 0 \leq f(n) \leq bg(n)\}.$$

- **Notation**: We often write functions as $f(n)$ (*as opposed to the correct notation:* $f : \mathbb{N} \to \mathbb{N}$)
- **Macro convention**:
  - A set in a formula represents an anonymous function in the set.
  - For example

    $$f(n) = n^3 + O(n^2) \iff \exists g(n) \in O(n^2). f(n) = n^3 + g(n).$$

    and

    $$n^2 + O(n) = O(n^2) \iff \forall f(n) \in O(n). \exists g(n) \in O(n^2). n^2 + f(n) = g(n).$$

Definition 1.1.2. (**Ω-notation**) For a given function $g : \mathbb{N} \to \mathbb{N}$, we denote $\Omega(g(n))$ as the set of functions

$$O(g(n)) = \{f(n) : \exists N, a > 0. \forall n > N. 0 \leq ag(n) \leq f(n)\}.$$

Definition 1.1.3. (**Θ-notation**) For a given function $g : \mathbb{N} \to \mathbb{N}$, we denote $\Theta(g(n))$ as the set of functions

$$\Theta(g(n)) = \{f(n) : \exists N, a, b > 0. \forall n > N. 0 \leq ag(n) \leq f(n) \leq bg(n)\}.$$

- Note that

  $$\Theta(g(n)) = \{f(n) : f(n) \in \Omega(g(n)) \wedge f(n) \in O(g(n))\}.$$

  Hence $\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$. So $\Theta(g(n)) \subset O(g(n))$ and $\Theta(g(n)) \subset \Omega(g(n))$

Theorem 1.1.1. For all functions $f,\ g, h : \mathbb{N} \to \mathbb{N}$, we have

(i) *Transitivity*:

$$\begin{aligned}
f(n) \in \Theta(g(n)) \wedge g(n) \in \Theta(h(n)) &\implies f(n) \in \Theta(h(n)) \\
f(n) \in O(g(n)) \wedge g(n) \in O(h(n)) &\implies f(n) \in O(h(n)) \\
f(n) \in \Omega(g(n)) \wedge g(n) \in \Omega(h(n)) &\implies f(n) \in \Omega(h(n))
\end{aligned}$$

(ii) *Reflexivity*:

$$\begin{gathered}
f(n) \in \Theta(f(n)) \\
f(n) \in O(f(n)) \\
f(n) \in \Omega(f(n))
\end{gathered}$$

(iii) *Symmetry*:

$$f(n) \in \Theta(g(n)) \iff g(n) \in \Theta(f(n)).$$

(iv) *Transpose Symmetry*:

$$f(n) \in O(g(n)) \iff g(n) \in \Omega(f(n))$$

2 Turing Machines

2.1 Turing Machines

Definition 2.1.1. (**Turing Machines**) A k-tape Turing machine M on Σ is the 5-tuple $(Q, \Sigma, q_0, \delta, H)$:

(i) Q is a finite set of *states*

(ii) Σ is a finite alphabet, disjoint from Q

(iii) $q_0 \in Q$ is the initial state.

(iv) $\delta : (Q \times \mathcal{O}(\Sigma)^k) \to Q \times (\mathcal{O}(\Sigma) \times \{\leftarrow, -, \rightarrow\})$ is the transition function.

(v) $H \subseteq Q$ is the set of *halting states*

- **Notation**:
  - The set of k-tape Turing machines over Σ is denoted TM^k_Σ.
  - The set of directions $\mathsf{Direction} = \{\leftarrow, -, \rightarrow\}$.
  - The set of actions $\mathsf{Act}_\Sigma = \mathcal{O}(\Sigma) \times \mathsf{Direction}$.
- **Idea**: Reasoning about *labelled states* (e.g. `acc, rej`) $\implies$ *labelled machines*
- Tape Classifications (related to Flynn's Tax):
  - $n = 0$ defines the set of `Nil` machines
  - $n = 1$ defined the set of SISD machines
  - $n > 1$ defines the set of MIMD machines

Definition 2.1.2. (**Labelled Machines**) A labelled k-tape Turing machine M_L w/ labels on L is the pair $M_L = (M, \mathsf{lab})$ where $M = (Q, \Sigma, q_0, \delta, H)$ is a k-tape *unlabelled* Turing machine and $\mathsf{lab} : Q \to L$ is the *labelling function* of M.

- **Notation**: The set of k-tape labelled Turing machines over Σ, L is denoted $\mathsf{TM}^k_\Sigma(L)$.

Definition 2.1.3. (**Tape**) A tape τ on Σ is the pair $\tau = (u, v)$ where $u \in \rhd\Sigma^*, v \in \Sigma^*$, and $\rhd$ denote the left end-of-tape markers.

If $u = wa$, then a is the current character under the read/write head.

- **Notation**: Tape_Σ denotes the set of tapes on Σ

Definition 2.1.4. (**Tape Operators**) The tape movement operator $\texttt{move} : \mathsf{Direction} \times \mathsf{Tape}_\Sigma \to \mathsf{Tape}_\Sigma$:

$\texttt{move}(\leftarrow, (\rhd, v)) = (\rhd, v)$
$\texttt{move}(\leftarrow, (ua, v)) = (u, av)$
$\texttt{move}(\rightarrow, (u, av)) = (ua, v)$
$\texttt{move}(\rightarrow, (u, \varepsilon)) = (u, \varepsilon)$
$\texttt{move}(-, \tau) = \tau$

The left/right tape operators $\texttt{left}, \texttt{right} : \mathsf{Tape}_\Sigma \to \Sigma^*$

$\texttt{left}(\rhd, v) = \varepsilon$
$\texttt{left}(\rhd ua, v) = u$

The tape write operator $\texttt{write} : \mathsf{Tape}_\Sigma \times \mathcal{O}(\Sigma) \to \mathsf{Tape}_\Sigma$

$\texttt{write}(\tau, \emptyset) = \tau$
$\texttt{write}(\tau, \lfloor a \rfloor) = (\rhd\texttt{left}(\tau)a, \texttt{right}(\tau))$

The current symbol function $\texttt{current} : \mathsf{Tape}_\Sigma \to \mathcal{O}(\Sigma)$ is defined by

$\texttt{current}(ua, v) = \lfloor a \rfloor$
$\texttt{current}(\_) = \emptyset$

The contents function $\texttt{contents} : \mathsf{Tape}_\Sigma \to \Sigma^*$ is defined by

$\texttt{contents}(\rhd u, v) = uv$

- The empty type is $\tau_0 = (\rhd, \varepsilon)$. The initial tape w/ contents $u \in \Sigma^*$ is $\tau_0(u) = (\rhd, u)$.

Definition 2.1.5. (**Configuration**) A k-tape Turing machine *configuration* c for $M = (Q, \Sigma, q_0, \delta, H)$ is the tuple $c = (q, \tau_1, \ldots, \tau_k)$ where:

– The current state $q \in Q$

– The ith tape $\tau_i \in \mathsf{Tape}_\Sigma$.

The set of configurations for M is denoted $\mathscr{C}(M)$.

Definition 2.1.6. (**Transition Relation**) The transition function for Σ tapes Tape_Σ w/ action $\mathsf{act} \in \mathsf{Act}_\Sigma$, denoted $\xrightarrow{\mathsf{act}}: \mathsf{Tape}_\Sigma \rightharpoonup \mathsf{Tape}_\Sigma$ is defined by

$$\frac{\tau_2 = \mathsf{move}(D, \mathsf{write}(\tau_1, a))}{\tau_1 \xrightarrow{(a,D)} \tau_2}$$

The transition relation for n-tape TM $M = (Q, \Sigma, q_0, \delta, H)$, denoted $\longrightarrow_M: \mathscr{C}(M) \rightharpoonup \mathscr{C}(M)$ is inductively defined by

$$\frac{\mathsf{current}(\tau_i) = a_i \qquad \delta(q, a_i) = (q', \mathsf{act}_1, \ldots, \mathsf{act}_n) \qquad \tau_i \xrightarrow{\mathsf{act}_i} \tau_i'}{(q, \tau_1, \ldots, \tau_n) \longrightarrow_M (q', \tau_1', \ldots, \tau_n')} \; [q \notin H]$$

- **Notation**: $\longrightarrow_M^*$ denotes a sequence of transitions, the reflexive transitive closure of $\longrightarrow_M$.

Theorem 2.1.1. (**Determinisim**)

$$\begin{aligned}
&\forall M \in \mathsf{TM}_\Sigma^k. \\
&\quad \forall c, c_0, c_1 \in \mathscr{C}(M). \\
&\quad\quad c \longrightarrow_M c_0 \land c \longrightarrow_M c_1 \implies c_0 = c_1
\end{aligned}$$

Definition 2.1.7. (**Computation**) A *computation* of a Turing machine M is a sequence of transitions (infinite or finite)

$$c_0 \longrightarrow_M c_1 \longrightarrow_M \cdots,$$

where $c_0 \in \mathscr{C}(M)$ is the *initial* configuration.

- A configuration $c = (q, \tau_1, \ldots, \tau_n)$ is halting if $q \in H$. Note that $c \not\longrightarrow_M \iff c$ is halting.

Definition 2.1.8. (**Halting Computation**) A *halting computation* of a k-tape Turing machine $M = (Q, \Sigma, q_0, \delta, H)$, denoted $(u_1, \ldots, u_\ell) \Downarrow_M v$, where $\Downarrow_M \colon (\Sigma^*)^\ell \rightharpoonup \Sigma^*$, and $\ell < k$, defined by

$$(u_1, \ldots, u_\ell) \Downarrow_M^q \mathsf{contents}(\tau_k) \iff (q_0, \tau_0(u_1), \ldots, \tau_0(u_\ell), \ldots, \tau_0) \longrightarrow_M^* (q, \tau_1, \ldots, \tau_k) \not\longrightarrow .$$

- Halting computation define the I/O convention: kth tape = output and first ℓ tapes = input.

Theorem 2.1.2. (**MIMD and SISD Reduction**)

$$\begin{aligned} &\forall M \in \mathsf{TM}_\Sigma^k . \exists N \in \mathsf{TM}_\Sigma^1 . \\ &\qquad \forall u, v \in \Sigma^* . u \Downarrow_M v \iff u \Downarrow_N v \end{aligned}$$

2.1.1 Computable Functions and Languages

Definition 2.1.9. (**Computable**) A function $f \in \mathcal{P}\left[\Sigma^* \rightharpoonup \Sigma^*\right]$ is Turing computable iff there exists a k-tape Turing machine $M \in \mathsf{TM}_\Sigma^k$ s.t

$$\forall u, v \in \Sigma^* . u \Downarrow_M v \iff f(u) = v$$

Definition 2.1.10. (**Encodable**) A set S is encodable on the finite alphabet Σ_S if there exists a bijection $[\![\cdot]\!]_S : S \to \Sigma_S^*$.

- All encodings for representations of mathematical objects are *polynomially related* (w/ the exception of unary encoding $\implies$ pseudo-polynomial complexity)
- If S is encodable on Σ, we must extend Σ (denoted Σ_+) w/ additional start, stop symbols

Definition 2.1.11. The extended alphabet Σ_+ of Σ be defined as:

$$\begin{aligned} \Sigma_+ \ &\texttt{::=}\ \texttt{Start} \mid \texttt{Stop} \\ &\mid\ a \in \Sigma \end{aligned}$$

Let $\mathsf{Extend} : \Sigma \to \Sigma^+$ define a retraction.

- Extended alphabet Σ_+ is used to define tape contents of *encodable* sets

Definition 2.1.12. (**Containment**) Let S be an encodable set on Σ. The containment relation $\simeq: S \longleftrightarrow \Sigma_+^*$ is defined by

$$x \simeq w \iff u \in \Sigma^*.w = u\mathsf{Start}\,[\![x]\!]_S\,\mathsf{Stop}$$

The containment relation $\simeq: S \longleftrightarrow \mathsf{Tape}_{\Sigma_+}$ is defined by

$$x \simeq \tau \iff x \simeq \mathsf{contents}(\tau)$$

Definition 2.1.13. (**(Natural) Computable**) A function $f \in \mathcal{P}\left[\mathbb{N}^n \rightharpoonup \mathbb{N}\right]$ is Turing computable iff there exists a Turing machine $M \in \mathsf{TM}_\Sigma^k$ s.t

$$\begin{aligned} &\forall x_1, \ldots, x_n, y \in \mathbb{N}. \\ &\quad u \Downarrow_M v \wedge [x_1, \ldots, x_n] \simeq u \wedge y \simeq v \iff f(x_1, \ldots, x_n) = y \end{aligned}$$

Definition 2.1.14. (**Decidable**) A Turing machine M decides $\mathcal{L} \subseteq \Sigma^*$ iff the characteristic function $\chi_{\mathcal{L}} : \Sigma^* \to \{0, 1\}$ is computable.

A language $\mathcal{L} \subseteq \Sigma^*$ is *decidable* iff there exists a Turing machine M that decides $\mathcal{L}$.

Definition 2.1.15. (**Acceptable**) A language $\mathcal{L} \subseteq \Sigma^*$ is *acceptable* iff the partial function $\mathsf{acc}_{\mathcal{L}} : \Sigma^* \rightharpoonup \{1\}$ s.t

$$\forall x \in \Sigma^*.x \in \mathcal{L} \iff \mathsf{acc}_{\mathcal{L}}(x) = 1$$

The language $\mathcal{L}(M) \subseteq \Sigma^*$ that is *accepted* by the labelled Turing machine $M \in \mathsf{TM}_\Sigma^k(\{\_, \mathsf{acc}, \mathsf{rej}\})$ is defined by

$$\mathcal{L}(M) = \{u \in \Sigma^* : \exists v \in \Sigma^*.u \Downarrow_M^q v \wedge \mathsf{lab}(q) = \mathsf{acc}\}$$

- Acceptable is a *weaker* property that decidable (since acceptable may have a non-terminating computation for $u \notin \mathcal{L}$) $\implies$ Hierarchy of languages

Definition 2.1.16. (**Recursive and Recursively Enumerable Languages**) Let $\mathcal{L} \subseteq \Sigma^*$ be a formal language:

(i) $\mathcal{L}$ is *recursively enumerable* iff $\mathcal{L}$ is acceptable

(ii) $\mathcal{L}$ is *recursive* iff $\mathcal{L}$ is decidable.

Theorem 2.1.3. For all $\mathcal{L} \subseteq \Sigma^*$:

$$\mathcal{L} \text{ is recursive} \implies \mathcal{L} \text{ is recursively enumerable}$$

2.1.2 Time and Space Complexity

Definition 2.1.17. (**Running Time**) The *running* time of a k-tape Turing machine $M \in \mathsf{TM}_\Sigma^k$ on $u_1, \ldots, u_\ell \in \Sigma^*$ is $t_M(u_1, \ldots, u_\ell) = k \in \mathbb{N}$ iff

$$\begin{aligned}&\exists q \in Q, (\tau_i) \in \mathsf{Tape}_\Sigma.\\ &\qquad (q_0, \tau_0(u_1), \ldots, \tau_0(u_\ell), \ldots, \tau_0) \longrightarrow_M^k (q, \tau_1, \ldots, \tau_k) \not\longrightarrow\end{aligned}$$

For non-terminating computations, $t_M(u_1, \ldots, u_\ell) = \infty$.

Definition 2.1.18. (**Time Bound**) The time bound of a k-tape Turing machine $M \in \mathsf{TM}_\Sigma^k$ is $f : \mathbb{N} \to \mathbb{N}$ iff

$$\forall u_1, \ldots, u_\ell \in \Sigma^*. t_M(u_1, \ldots, u_\ell) \leq f\left(\sum_{i=1}^{\ell} |u_i|\right)$$

- Asymptotic notation is used to denote *asymptotic time bounds* (or *time complexity*) of TMs.

Definition 2.1.19. (**Time Complexity**) The time complexity of a k-tape Turing machine $M \in \mathsf{TM}_\Sigma^k$ is $O(g(n))$ s.t the time bound of M, $f(n) \in O(g(n))$.

- **Problem**: Space complexity is defined on space used (excluding input) $\implies$ we require a Turing machine w/ I/O conventions.

Definition 2.1.20. (**Turing Machine with I/O**) A k-tape Turing Machine $M = (Q, \Sigma, q_0, \delta, H)$ w/ I/O is a k-tape Turing Machine M that satisfies:

(i) The first ℓ-tapes are *read-only* where $\ell < k$.

$$\begin{aligned}&\forall 1 \leq i \leq \ell.\\ &\quad \forall q, q' \in Q, (a_i), (a_i') \in \mathcal{O}(\Sigma), (D_i) \in \mathsf{Direction}.\\ &\qquad \delta(q, a_1, \ldots, a_i, \ldots, a_k) = (q', (a_1', D_1), \ldots, (a_i', D_i), \ldots, (a_k', D_k)) \implies a_i = a_i'\end{aligned}$$

(ii) The last tape is *write-only* (cannot move to the left)

$$\begin{aligned}&\forall q, q' \in Q, (a_i), (a_i') \in \mathcal{O}(\Sigma), (D_i) \in \mathsf{Direction}\\ &\quad \delta(q, a_1, \ldots, a_k) = (q', (a_1', D_1), \ldots, (a_k', D_k)) \implies D_k \neq \leftarrow\end{aligned}$$

- **Notation**: The set of k-tape Turing Machines w/ I/O is denoted $\mathsf{IO\_TM}^k_\Sigma$

Theorem 2.1.4. (**IO Equivalence**)

$$\forall M \in \mathsf{TM}^k_\Sigma . \exists N \in \mathsf{IO\_TM}^{k+\ell+1}_\Sigma$$
$$\forall (u_i), v \in \Sigma^* . (u_1, \ldots, u_\ell) \Downarrow_M v \iff (u_1, \ldots, u_\ell) \Downarrow_N v$$

Definition 2.1.21. (**Space**) For a k-tape Turing machine $M \in \mathsf{IO\_TM}^k_\Sigma$ w/ I/O on input $u_1, \ldots, u_\ell \in \Sigma^*$, the *space used* is $s_M(u_1, \ldots, u_\ell) \in \mathbb{N}$ is defined as

$$s_M(u_1, \ldots, u_\ell) = \sum_{i=\ell+1}^{k-1} |\mathsf{contents}(\tau_i)|$$

where

$$\exists q \in Q, (\tau_i) \in \mathsf{Tape}_\Sigma .$$
$$(q_0, \tau_0(u_1), \ldots, \tau_0(u_\ell), \ldots, \tau_0) \longrightarrow^*_M (q, \tau_1, \ldots, \tau_k) \not\longrightarrow$$

Definition 2.1.22. (**Space Bound**) The space bound of a Turing machine M is $f : \mathbb{N} \to \mathbb{N}$ iff

$$\forall u_1, \ldots, u_\ell \in \Sigma^* . t_M(u_1, \ldots, u_\ell) \leq f\left(\sum_{i=1}^{\ell} |u_i|\right)$$

- Asymptotic space complexity is the asymptotic space bound of a TM.

Definition 2.1.23. (**Time Complexity**) The space complexity of a k-tape Turing machine $M \in \mathsf{IO\_TM}^k_\Sigma$ is $O(g(n))$ s.t the space bound of M, $f(n) \in O(g(n))$.

- **Convention**: Generally denote the space complexity of TM $M \in \mathsf{TM}^k_\Sigma$ w/ the space complexity of it's equivalent TM $M' \in \mathsf{IO\_TM}^{k+\ell+1}_\Sigma$

2.2 Non-Deterministic Turing Machines

Definition 2.2.1. (**Non-Deterministic Turing Machines**) A non-deterministic k-tape Turing machine N is the 5-tuple $N = (Q, \Sigma, q_0, \Delta, H)$:

(i) Q, Σ, q_0 and H are as defined in ??

(ii) $\Delta : (Q \times \mathcal{O}(\Sigma)^k) \to \mathcal{P}(Q \times (\mathcal{O}(\Sigma) \times \{\leftarrow, -, \rightarrow\})^k)$ is the transition *relation*.

- **Notation**: The set of k-tape NTMs over Σ is denoted NTM_Σ^k
- Equivalent definitions for non-deterministic Turing machine configurations (See ??)

Definition 2.2.2. (**Transition Relation**) The transition relation for n-tape NTM $N = (Q, \Sigma, q_0, \Delta, H)$, denoted $\longrightarrow_N$: $\mathscr{C}(N) \longleftrightarrow \mathscr{C}(N)$ is inductively defined by

$$\frac{\mathsf{current}(\tau_i) = a_i \qquad (q', \mathsf{act}_1, \dots, \mathsf{act}_n) \in \Delta(q, a_i) \qquad \tau_i \xrightarrow{\mathsf{act}_i} \tau_i'}{(q, \tau_1, \dots, \tau_n) \longrightarrow_N (q', \tau_1', \dots, \tau_n')} \; [q \notin H]$$

- **Note**: Determinism (theorem ??) no-longer holds for NTMs $\implies$ *execution tree of configurations*: IMAGE

  A *computation tree*.
- Equivalent definitions for computations, computable, etc.

2.2.1 Time and Space Complexity

- Time and space bounds are equivalently defined. See ??

Theorem 2.2.1. For all $N \in \mathsf{NTM}_\Sigma^k$, that accepts $L(N)$ w/ $f(n)$ time bound, there exists a deterministic Turing machine $M \in \mathsf{TM}_\Sigma^n$ s.t $L(N) = L(M)$ w/ time bound $O(c^{2f(n)})$ for some $n \in \mathbb{N}, c > 1$.

- Deterministic simulation of NTM is performed by a breadth-first search of the computation tree.

3 Complexity Classes

3.1 Complexity Classes

- **Idea**: Classify languages via bounds, specified by
  - A model of computation
  - A resource (time, space, # cores, etc)
  - A bound on the resource.

Definition 3.1.1. (**Complexity Class**) A complexity class $\mathcal{C}$ is the set $\mathcal{C} \subseteq \mathcal{P}(\Sigma^*)$, where $\mathcal{L} \in \mathcal{C}$ is decidable in some model of computation w/ a bound on a given resource.

- Complexity classes define a collection of *computationally similar* problems $\implies$ reasoning about classes

3.1.1 Reductions

- **Idea**: Solving problems via *reductions*.

Definition 3.1.2. (**Reduction**) A reduction of $\mathcal{L}_1 \subseteq \Sigma_1^*$ to $\mathcal{L}_2 \subseteq \Sigma_2^*$ is a Turing computable function $f : \Sigma_1^* \to \Sigma_2^*$ s.t

$$\forall x \in \Sigma_1^*. x \in \mathcal{L}_1 \iff f(x) \in \mathcal{L}_2$$

- A reduction $f : \Sigma_1^* \to \Sigma_2^*$ reduces $\mathcal{L}_1$ to $\mathcal{L}_2 \implies$ if $\mathcal{L}_2$ is *decidable* then $\mathcal{L}_1$ must be.
- **Notation**: $f : \mathcal{L}_1 \to \mathcal{L}_2$ denotes a reduction from $\mathcal{L}_1$ to $\mathcal{L}_2$

Lemma 3.1.1. For all reductions $f : \mathcal{L}_1 \to \mathcal{L}_2$,

$$\mathcal{L}_2 \text{ is decidable} \implies \mathcal{L}_1 \text{ is decidable.}$$

Proof. Let $\mathcal{L}_1 \subseteq \Sigma_1^*, \mathcal{L}_2 \subseteq \Sigma_2^*$ be arbitrary. Let $f : \mathcal{L}_1 \to \mathcal{L}_2$ be an arbitrary $\mathcal{L}_1$ to $\mathcal{L}_2$ reduction.

Let us assume that $\mathcal{L}_2$ is decidable. Hence $\chi_{\mathcal{L}_2} : \Sigma_2^* \to \{0, 1\}$ is computable. By definition ??,

$$\forall x \in \Sigma_1^*.\chi_{\mathcal{L}_1}(x) = 1 \iff \chi_{\mathcal{L}_2}(f(x)) = 1.$$

Hence $\chi_{\mathcal{L}_1} = \chi_{\mathcal{L}_2} \circ f$. By theorem ??, $\chi_{\mathcal{L}_1}$ is computable. Hence $\mathcal{L}_1$ is decidable. □

Corollary 3.1.0.1. For all reductions $f : \mathcal{L}_1 \to \mathcal{L}_2$,

$$\mathcal{L}_1 \text{ is undecidable} \implies \mathcal{L}_2 \text{ is undecidable.}$$

- Corollary ?? provides a method for proving whether a $\mathcal{L}$ is undecidable:
  - Determine a reduction $f : H \to \mathcal{L}$ where H is the *halting problem* (see section ??)

Definition 3.1.3. (**Hardness Relation**) The *hardness* relation $\leq: \mathcal{P}(\Sigma^*) \leftrightarrow \mathcal{P}(\Sigma^*)$ defined by

$$\mathcal{L}_1 \leq \mathcal{L}_2 \iff \exists \text{ reduction } f : \mathcal{L}_2 \to \mathcal{L}_1$$

We say that $\mathcal{L}_2$ is at least as *hard* as $\mathcal{L}_1$.

Theorem 3.1.1. (**Preorder of Hardness**) $\leq: \mathcal{P}(\Sigma^*) \leftrightarrow \mathcal{P}(\Sigma^*)$ is a preorder.

Definition 3.1.4. (**$\mathcal{C}$-Reductions**) A reduction $f : \mathcal{L}_1 \to \mathcal{L}_2$ is a $\mathcal{C}$-reduction if the language $\{x\_y : f(x) = y\} \in \mathcal{C}$. Similarly, for the Hardness relation, denoted $\leq_{\mathcal{C}}$.

Definition 3.1.5. (**Closure of $\mathcal{C}$-Reductions**) $\mathcal{C}$ is closed under $\mathcal{C}$-reductions iff

$$\forall \mathcal{L}_1, \mathcal{L}_2 \subseteq \Sigma^*.\mathcal{L}_1 \leq_{\mathcal{C}} \mathcal{L}_2 \wedge \mathcal{L}_1 \in \mathcal{C} \implies \mathcal{L}_2 \in \mathcal{C}$$

- All classes $\mathcal{C}$ of interest are closed under $\mathcal{C}$-reductions.

3.1.2 Completeness

- **Idea**: Hardness relation $\implies$ ordering on decision problems in a class.

Definition 3.1.6. (**Completeness**) For $\mathcal{L} \in \mathcal{C}$, $\mathcal{L}$ is $\mathcal{C}$-complete iff

$$\forall \mathcal{L}' \in \mathcal{C}. \mathcal{L} \leq \mathcal{L}'$$

Definition 3.1.7. (**Hardness**) A language $\mathcal{L} \subseteq \Sigma^*$ is $\mathcal{C}$-hard iff

$$\forall \mathcal{L}' \in \mathcal{C}. \mathcal{L} \leq \mathcal{L}'$$

- Hardness and Completeness allow reasoning about the *hierarchy* of classes.
- Complete problems also "determine" the hardness of a class

3.2 Polynomial Time

- Deterministic Turing machines $\implies$ deterministic time and space complexity classes

Definition 3.2.1. (**Time Complexity Class**) Let $f : \mathbb{N} \to \mathbb{N}$. $\mathsf{TIME}(f(n)) \subseteq \mathcal{P}(\Sigma^*)$ is the complexity class (set of languages) s.t $\mathcal{L} \in \mathsf{TIME}(f(n)) \iff \mathcal{L}$ is decidable by a TM w/ time bound $O(f(n))$.

Definition 3.2.2. (**Space Complexity Class**) Let $f : \mathbb{N} \to \mathbb{N}$. $\mathsf{SPACE}(f(n)) \subseteq \mathcal{P}(\Sigma^*)$ is the complexity class (set of languages) s.t $\mathcal{L} \in \mathsf{SPACE}(f(n)) \iff \mathcal{L}$ is decidable by a TM w/ space bound $O(f(n))$.

Definition 3.2.3. (**Polynomial Time Complexity Class**) The polynomial time complexity class P is defined by

$$\mathsf{P} = \bigcup_{k=1}^{\infty} \mathsf{TIME}(n^k)$$

- P defines the set of *feasibly* computable languages.

Definition 3.2.4. (**Tractable and Intractable**) For $\mathcal{L} \subseteq \Sigma^*$, if $\mathcal{L} \in \mathsf{P}$, then $\mathcal{L}$ is said to be *tractable*. Conversely, if $\mathcal{L} \notin \mathsf{P}$, then $\mathcal{L}$ is *intractable*

- **Note**: Definitions of tractability and intractability *are not always* sound (e.g. worst-case exponential algorithms are used in practice).

Definition 3.2.5. (**Polynomial Time Reductions**) A reduction $f : \mathcal{L}_1 \to \mathcal{L}_2$ is a *polynomial time reduction* iff the TM M that computes f has a polynomial time bound $O(n^k)$.

Lemma 3.2.1. P is closed under P-reductions.

3.2.1 Problems

3.2.1.1 Reachability

Definition 3.2.6. (**Reachability**) The reachability problem is defined as the *decision problem*:

Given a directed graph $G = (V, E)$ and vertices $u, v \in V$, determined whether $u \longrightarrow^* v$.

Denoted $\mathsf{Reachability}(G, u, v)$.

- $\mathsf{Reachability}$ is a *decision problem*
- **Solution**: BFS / DFS. See Algorithms IA notes
- **Complexity**: $O(|V|^2)$

3.2.1.2 Max Flow

Definition 3.2.7. (**Network**) A network $N = (V, E, s, t, c)$ is a directed graph $G = (V, E)$ w/ vertices $s, t \in V$ denoting the source and sink respectively, and $c : E \to \mathbb{N}$ denoting the *capacity function*.

Definition 3.2.8. (**Flow**) A flow f is a function $f : E \to \mathbb{N}$ in the network N satisfying:

(i) Capacity: $\forall (u, v) \in E . f(u, v) \leq c(u, v)$

(ii) Conservation:

$$\forall v \in V \setminus \{s, t\} . \sum_{u:(u,v) \in E} f(u, v) = \sum_{w:(v,w) \in E} f(v, w)$$

Definition 3.2.9. (**Value**) The value of a flow f in N is defined as

$$\begin{aligned}\text{value}(f) &= \sum_{u:(s,u)\in E} f(s,u) - \sum_{v:(v,s)\in E} f(v,s) \\ &= \sum_{u:(u,t)\in E} f(u,t) - \sum_{v:(t,v)\in E} f(t,v)\end{aligned}$$

Definition 3.2.10. (**Max Flow**) The max flow problem is defined as the *optimization problem*:

Given a network N, determined a flow f s.t

$$f = \arg\max_{\text{flow } f} \text{value}(f)$$

Denoted $\mathsf{MaxFlow}(N)$.

- **Note**: $\mathsf{MaxFlow}$ is an *optimization problem*. However, all optimization problems have an equivalent *decision problem*

Definition 3.2.11. (**Bounded Max Flow**) The bounded max flow problem is defined as the *decision problem*:

Given a network N and bound k, determine whether

$$\max_{\text{flow } f} \text{value}(f) \leq k$$

Denoted $\mathsf{MaxFlow}(N, k)$.

Definition 3.2.12. (**Cut**) A cut $(S, \overline{S})$ on the network N is a *partition* on V, s.t $s \in S$ and $t \in \overline{S}$. The value of a cut $(S, \overline{S})$ is defined by

$$\text{value}(S, \overline{S}) = \sum_{\substack{u\in S, v\in\overline{S} \\ (u,v)\in E}} c(u,v)$$

Theorem 3.2.1. (**Max Flow, Min Cut Theorem**) For all flows f and cuts $(S, \overline{S})$ on N:

$$\text{value}(f) \leq \text{value}(S, \overline{S})$$

Proof. Let $N = (V, E, s, t, c)$ be an arbitrary network. Let f and $(S, \overline{S})$ be arbitrary flows and cuts on N.

Let us extend c and f, to be defined on $V^2 \to \mathbb{N}$ s.t

$$\forall (u, v) \notin E . c(u, v) = f(u, v) = 0$$

So we have

$$\begin{aligned}
\text{value}(f) &= \sum_{u \in V} f(s, u) - \sum_{u \in V} f(u, s) \\
&= \sum_{v \in S} \left(\sum_{u \in V} f(v, u) - \sum_{u \in V} f(u, v) \right) && \text{Conversation} \\
&= \sum_{v \in S} \sum_{u \in S} f(v, u) + \sum_{v \in S} \sum_{u \in \overline{S}} f(v, u) \\
&\quad - \sum_{v \in S} \sum_{u \in S} f(u, v) - \sum_{v \in S} \sum_{u \in \overline{S}} f(u, v) \\
&= \sum_{v \in S} \sum_{u \in \overline{S}} f(v, u) - \sum_{v \in S} \sum_{u \in \overline{S}} f(u, v) \\
&\leq \sum_{v \in S} \sum_{u \in \overline{S}} f(v, u) && f(u, v) \geq 0 \\
&\leq \sum_{v \in S} \sum_{u \in \overline{S}} c(v, u) && \text{Capacity} \\
&= \text{value}(S, \overline{S})
\end{aligned}$$

□

- **Observation**: A flow f is *not optimal* $\iff$ there exists a flow f' s.t $\Delta f = f' - f$ has the value $\text{value}(\Delta f) \geq 0$.

  Problem: Augmenting flow Δf is negative on some edges $\implies$ *backwards edges*

  Definition 3.2.13. (**Residual Network**) The residual network $N(f)$ of a flow f on $N = (V, E, s, t, c)$ is defined as $N(f) = (V, E', s, t, c')$ where

  $$\begin{aligned}
  E' = E &\setminus \{(u, v) \in E : f(u, v) = c(u, v)\} \\
  &\cup \left\{(u, v) \in V^2 : (v, u) \in E \wedge f(v, u) > 0\right\}
  \end{aligned}$$

and

$$c'(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E \\ f(v,u) & (u,v) \in E' \setminus E \end{cases}$$

So by definition: f is optimal $\iff$ there is no positive flow Δf in $N(f)$

- **Solution**:
  1. Determining a positive flow Δf consists of determining a path in $N(f) \implies$ reduces to Reachability ($O(|V|^2)$)
  2. Compute flow $f \leftarrow f + \Delta f$
  3. Repeat 1, 2 until no such flow Δf exists.
- **Complexity**:
  - Finding the augmenting flow Δf: $O(|V|^2)$
  - Each flow adds at least 1 to value(f). Defined $F = \max_{\text{flow } f} \text{value}(f)$ $\implies O(F)$ iterations
  - $O(|V|^2 F)$

3.2.1.3 Matching

Didn't really appreciate the hetro and cis normativity of this chapter...

Definition 3.2.14. (**Bipartite Graph**) A Bipartite graph B is the tuple $B = (U, V, E)$ where $U \cap V \neq \emptyset$ and $E \subseteq U \times V$.

Definition 3.2.15. (**Matching**) A matching on B is a set $M \subseteq E$ satisfying

$$\forall (u,v), (u',v') \in M. u \neq u' \land v \neq v'$$

- A matching defines a *bijection* between subsets of U, V.

Definition 3.2.16. (**Matching**) The matching problem is defined as the decision problem:

Given a Bipartite graph B, determine whether a matching M exists

Denoted $\mathsf{Matching}(B)$.

- **Solution**:
  - Reduction from $\mathsf{Matching}(B)$ to $\mathsf{MaxFlow}(N, n)$ with network $N = (\{s, t\} \cup U \cup V, E', s, t, c)$ where

$$\begin{aligned} E' &= \{(s, u) : u \in U\} \cup E \cup \{(v, t) : v \in V\} \\ c &= \mathbf{1} \end{aligned}$$

    and $n = \max\{|U|, |V|\}$

- **Complexity**: $O(n^3)$.

3.2.2 co-P

3.3 Non-Deterministic Polynomial Time

- Non-Deterministic Turing machines $\implies$ non-deterministic time and space complexity classes

Definition 3.3.1. (**NTime Complexity Class**) Let $f : \mathbb{N} \to \mathbb{N}$. $\mathsf{NTIME}(f(n)) \subseteq \mathcal{P}(\Sigma^*)$ is the complexity class (set of languages) s.t $\mathcal{L} \in \mathsf{NTIME}(f(n)) \iff \mathcal{L}$ is decidable by a NTM w/ time bound $O(f(n))$.

Definition 3.3.2. (**NSpace Complexity Class**) Let $f : \mathbb{N} \to \mathbb{N}$. $\mathsf{NSPACE}(f(n)) \subseteq \mathcal{P}(\Sigma^*)$ is the complexity class (set of languages) s.t $\mathcal{L} \in \mathsf{NSPACE}(f(n)) \iff \mathcal{L}$ is decidable by a NTM w/ space bound $O(f(n))$.

Definition 3.3.3. (NP **Complexity Class**) The non-deterministic polynomial time complexity class NP is defined by

$$\mathsf{NP} = \bigcup_{k=1}^{\infty} \mathsf{NTIME}(n^k)$$

Lemma 3.3.1. NP is closed under P-reductions.

Definition 3.3.4. (NP **Complexity Class**) *Equivalently*, a language $\mathcal{L} \subseteq \Sigma^*$ is in $\mathsf{NP} \iff$

$$\mathcal{L} = \{x \in \Sigma^*. \exists y \in \Sigma^*. R(y, x)\},$$

where $R : \Sigma^* \longleftrightarrow \Sigma^*$ satisfying:

- R is decidable in polynomial time by the TM M
- R is *polynomially balanced*:

$$\forall y, x \in \Sigma^*. R(y, x) \implies |y| \le p(|x|),$$

where p is some polynomial.

- y is the *certificate* of $x \in \mathcal{L}$

3.3.1 Propositional Logic $\mathcal{L}_0$ Problems

3.3.1.1 SAT

Definition 3.3.5. (**SAT**) The satisfiability problem is defined defined as the decision problem:

Given a proposition $\psi \in \mathcal{L}_0$, determine whether ψ is satisfiable.

Denoted $\mathsf{SAT}(\psi)$.

- **Note**: VAL reduces to SAT by $\vDash \psi \iff \neg\psi$ is unsatisfiable.
- **Solution**:
  - Consider $2^{|\llbracket\psi\rrbracket_P|}$ possible interpretations (by coincidence lemma I)
  - $O(|\llbracket\psi\rrbracket_P|^2)$ algorithm for determining $\mathcal{T}\llbracket\psi\rrbracket_{\mathcal{I}} = 1$.
- **Complexity**: $O(2^{|\llbracket\psi\rrbracket_P|}|\llbracket\psi\rrbracket_P|^2)$.

Lemma 3.3.2. $\mathsf{SAT} \in \mathsf{NP}$.

Theorem 3.3.1. SAT is NP-complete.

Proof. We show that

$$\forall \mathcal{L} \in \mathsf{NP}. \mathcal{L} \le \mathsf{SAT}$$

Let $\mathcal{L} \in \mathsf{NP}$ be arbitrary. By definition, there exists a NTM $M = (Q, \Sigma, q_0, \Delta, H)$ that decides $\mathcal{L}$ w/ polynomial time bound $f(|x|) \in O(|x|^k)$ for some $k \ge 0$.

We wish to define a reduction $f : \mathcal{L} \to \mathsf{SAT}$. Let $x \in \Sigma^*$ be arbitrary. Define the set of propositional symbols $\Omega(x) \subseteq \Sigma_P$ as:

$S_{i,q}$	will be true if M is in state q at step i	$i \leq \lvert x\rvert^k, q \in Q$
$T_{i,j,a}$	will be true if tape τ is at position j w/ current a at step i	$i, j \leq \lvert x\rvert^k, a \in \mathcal{O}(\Sigma)$
$H_{i,j}$	will be true if head is at position j at step i	$i, j \leq \lvert x\rvert^k$

Note that $|\Omega(x)| = |Q||x|^k + |\Sigma||x|^{2k} + |x|^{2k}$.

The proposition $f(x) = \psi$ is defined by the conjunction of:

$S_{1,q_0} \wedge H_{1,1}$	Computation starts in state q_0 at position 1
$\bigwedge_i \bigwedge_j \left(H_{i,j} \to \bigwedge_{k \neq j} \neg H_{i,j} \right)$	Tape position cannot be in 2 positions at once
$\bigwedge_i \bigwedge_q \left(S_{i,q} \to \bigwedge_{q' \neq q} \neg S_{i,q'} \right)$	Cannot be in 2 states at once
$\bigwedge_i \bigwedge_j \bigwedge_a \left(T_{i,j,a} \to \bigwedge_{a' \neq a} \neg T_{i,j,a'} \right)$	Cannot have 2 symbols in the same position at once
$\bigwedge_{j \leq \lvert x \rvert} T_{1,j,\lfloor x_j \rfloor} \wedge \bigwedge_{\lvert x \rvert < j} T_{1,j,\emptyset}$	Initially, tape contains $\lvert x \rvert$ (and nothing else)
$\bigwedge_i \bigwedge_j \bigwedge_{j' \neq j} \bigwedge_a (H_{i,j} \wedge T_{i,j',a}) \to T_{i+1,j',a}$	Only the tape changes under the head
$\bigwedge_i \bigwedge_j \bigwedge_a \bigwedge_q (H_{i,j} \wedge S_{i,q} \wedge T_{i,j,a})$ $\to \bigvee_{\Gamma} (H_{i+1,j'} \wedge S_{i+1,q'} \wedge T_{i+1,j,a'})$ where $\Gamma = \{(q', j', a') : (q', a', D) \in \Delta(q, a) \wedge j' = [\![j]\!]_D\}$.	Transitions
$\bigvee_{q \in \mathsf{lab}(\mathsf{acc})} \bigvee_i S_{i,q}$	Halts at some point.

We now show that

$$x \in \mathcal{L} \iff f(x) \in \mathsf{SAT}$$

($\implies$). Let us assume that $x \in \mathcal{L}$, so we have

$$\exists q_{\mathsf{acc}} \in \mathsf{lab}(\mathsf{acc}), u \in \Sigma^*.(q_0, \tau_0(x)) \longrightarrow_M^{|x|^k} (q_{\mathsf{acc}}, u)$$

Let $c_i = (q_i, \tau_i)$ denote the ith configuration in the above computation. Let us define the interpretation $\mathcal{I} \in \Sigma_{\mathcal{I}}$ s.t

$$\begin{aligned} \mathcal{I}(S_{i,q_i}) &= 1 \\ \mathcal{I}(T_{i,j,a}) &= (a = \tau_i[j]) \\ \mathcal{I}(H_{i,j}) &= \mathsf{index}(\tau_i) \end{aligned}$$

where

```
index(▷, _) = 1
index(▷u, _) = |u| + 1
```

By definition of the tape semantics, Δ, and the computation, we have $\mathcal{T}\,[\![\psi]\!]_{\mathcal{I}} = 1$.

($\impliedby$). Let us assume that $f(x) = \psi \in \mathsf{SAT}$. Then there exists $\mathcal{I} \in \Sigma_{\mathcal{I}}$ s.t $\mathcal{T}\,[\![\psi]\!]_{\mathcal{I}} = 1$. We define the computation $c_i = (q_i, \tau_i)$ s.t

$$\begin{aligned} q_i &= q & \mathcal{I}(S_{i,q}) &= 1 \\ \tau_i &= (\triangleright a_1 \dots a_j, a_{j+1} \dots a_m) & \mathcal{I}(T_{i,k,a_{k-1}}) &= 1, \mathcal{I}(H_{i,j-1}) = 1 \end{aligned}$$

By (2), (3), (4), q_i and τ_i are unique. By (1), $c_0 = (q_0, \tau_0(x))$. By (5) and (6), the computation $c_0 \longrightarrow_M c_1 \longrightarrow \dots$ satisfies $\longrightarrow_M$, and by (7),

$$\exists q_{\mathsf{acc}} \in \mathsf{lab}(\mathsf{acc}), u \in \Sigma^*.(q_0, \tau_0(x)) \longrightarrow_M^* (q_{\mathsf{acc}}, u)$$

Hence since M decides $\mathcal{L}$, we have $x \in \mathcal{L}$. □

- **Note**: ψ encodes M and x. Constructing ψ is done in polynomial time.

3.3.1.2 CNF

Definition 3.3.6. (**CNF**) The CNF satisfiability problem is defined as the decision problem:

Given proposition $\psi = \bigwedge_i \bigvee_j \ell_{ij} \in \mathcal{L}_0^{CNF}$, determine whether ψ is satisfiable.

- Translation function $[\![\cdot]\!]^{CNF} : \mathcal{L}_0 \to \mathcal{L}_0^{CNF}$ defines reduction to CNF satisfiability.
- **Problem**: Reduction is *not polynomial* but exponential (by distributive law).
- **Solution**: Convert (1), (2), ..., (7) directly into CNF. Resulting in polynomial increases.

Lemma 3.3.3. CNF is NP-complete.

Proof. (**Sketch**) Polynomial reduction from SAT to CNF. Closure and transitivity of $\leq_{\mathsf{P}} \implies \mathsf{CNF}$ is NP-complete. □

Definition 3.3.7. (**3 CNF**) A proposition $\psi \in \mathcal{L}_0$ is said to be in *3 CNF* iff $\forall C_i \in [\![\psi]\!]_\Delta . |C_i| \leq 3$.

The set of 3 CNF propositions is denoted $\mathcal{L}_0^{3CNF}$

Definition 3.3.8. (**CNF**) The 3CNF satisfiability problem is defined as the decision problem:

Given proposition $\psi = \bigwedge_i \bigvee_{j\leq 3} \ell_{ij} \in \mathcal{L}_0^{3CNF}$, determine whether ψ is satisfiable.

Lemma 3.3.4. $\mathsf{3CNF}$ is NP-complete.

Proof. (*Sketch*) We have the following reduction $f : \mathsf{CNF} \to \mathsf{3CNF}$:

$$f(\bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i} \ell_{ij}) = \bigwedge_{i=1}^{n} \left((\ell_{i1} \vee \ell_{i2} \vee P_{i2}) \wedge \bigwedge_{j=2}^{m-3} (\neg P_{ij} \vee \ell_{i(j+1)} \vee P_{i(j+1)}) \wedge (\neg P_{i(m-2)} \vee \ell_{i(m-1)} \vee \ell_{im}) \right)$$

where $P_{ij} \notin [\![\psi]\!]_P$.

Note that ψ is satisfiable $\iff f(\psi)$ is satisfiable. We also note that polynomial increase in propositional symbols occurs due to $f \implies$ polynomial reduction.

□

3.3.2 Graph Problems

3.3.2.1 Independent Set

Definition 3.3.9. (**Independent Set**) For a undirected graph $G = (V, E)$, a set $I \subseteq V$ is said to be *independent* $\iff$

$$\forall u, v \in I.(u, v) \notin E.$$

- Every graph $G = (V, E)$ w/ $|V| \geq 1$ has a *trivial* independent set: $I = \{v\}$ w/ $v \in V$ (assuming no loops).

Definition 3.3.10. ($\mathsf{Independent}$) The independent set problem is defined as the optimization problem:

$$\text{Given } G = (V, E), \text{ determine}$$

$$I = \underset{\text{independent I}}{\arg\max} |I|$$

Denoted $\mathsf{Independent}(G)$

Definition 3.3.11. (**Bounded** $\mathsf{Independent}$) The bounded independent set problem is defined as the decision problem:

$$\text{Given } G = (V, E) \text{ and } k, \text{ determine whether}$$

$$\max \{|I| : \text{independent } I\} = k$$

Denoted $\mathsf{Independent}(G, k)$

- **Solution**: $2^{|V|}$ subsets of V. $O(|V|^2)$ method of determining whether $I \subseteq V$ is independent.
- **Complexity**: $O(2^{|V|}|V|^2)$.

Lemma 3.3.5. $\mathsf{Independent} \in \mathsf{NP}$

Theorem 3.3.2. $\mathsf{Independent}$ is NP-complete.

Proof. We proceed by defining a polynomial reduction from $\mathsf{3CNF}$ to $\mathsf{Independent}$, $f : \mathsf{3CNF} \to \mathsf{Independent}$, where

$$f(\bigwedge_{i=1}^{n} \bigvee_{j=1}^{3} \ell_{ij}) = (G, n)$$

where $G = (V, E)$ and

$$\begin{aligned} V &= \{v_{ij} : 1 \leq i \leq n, 1 \leq j \leq 3\} \\ E &= \{(v_{ij}, v_{ik}) : 1 \leq i \leq n, 1 \leq j < k \leq 3\} \\ &\quad \cup \{(v_{ij}, v_{kl}) : i \neq k \wedge \ell_{ij} \equiv \neg \ell_{kl}\} \end{aligned}$$

We now show that

$$\forall \psi \in \Sigma^*. \psi \in \mathsf{3CNF} \iff f(\psi) \in \mathsf{Independent}$$

Let $\psi \in \Sigma^*$ be arbitrary.
($\implies$). Let us assume that $\psi = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{3} \ell_{ij} \in \mathsf{3CNF}$. Hence there exists interpretation $\mathcal{I} \in \Sigma_{\mathcal{I}}$ s.t

$$\begin{aligned} & \mathcal{T}\left[\!\left[\psi\right]\!\right]_{\mathcal{I}} = 1 \\ \iff & \forall 1 \leq i \leq n. \mathcal{T}\left[\!\left[\vee_{j=1}^{3} \ell_{ij}\right]\!\right]_{\mathcal{I}} = 1 \\ \iff & \forall 1 \leq i \leq n. \exists 1 \leq j \leq 3. \mathcal{T}\left[\!\left[\ell_{ij}\right]\!\right]_{\mathcal{I}} = 1 \end{aligned}$$

Let us introduce the witnesses $1 \leq j_1, \ldots, j_n \leq 3$ s.t $\mathcal{T}\left[\!\left[\ell_{ij_i}\right]\!\right] = 1$. Define the set $I = \{v_{ij_i} : 1 \leq i \leq n\} \subseteq V$. We have $|I| = n$. So we wish to show that

$$\forall v_{ij}, v_{kl} \in I. (v_{ij}, v_{kl}) \notin E$$

Let $v_{ij}, v_{kl} \in I$ be arbitrary. We proceed by contradiction. Let us assume that $(v_{ij}, v_{kl}) \in E$. We have the following cases (by definition of E):

- $i = k$. Let us assume that $i = k$. A contradiction! By definition of I.
- $i \neq k \wedge \ell_{ij} \equiv \neg \ell_{kl}$. So we have

$$\begin{aligned} & \mathcal{T}\left[\!\left[\ell_{ij}\right]\!\right] = 1 \\ \iff & \mathcal{T}\left[\!\left[\neg \ell_{kl}\right]\!\right] = 1 \\ \iff & \mathcal{T}\left[\!\left[\ell_{kl}\right]\!\right] = 0 \end{aligned}$$

  A contradiction! Since $v_{kl} \in I$ and $T\left[\!\left[\ell_{kl}\right]\!\right] = 1$ by the definable property of I.

($\Longleftarrow$). Let us assume that $f(\psi) = (G, n) \in \mathsf{Independent}$, where $\psi = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{3} \ell_{ij}$ and $G = (V, E)$.

Hence there exists an independent set $I \subseteq V$ w/ $|I| = n$. Let us define the interpretation $\mathcal{I} \in \Sigma_{\mathcal{I}}$ s.t

$$\mathcal{T}\,[\![\ell_{ij}]\!]_{\mathcal{I}} = 1 \iff v_{ij} \in I$$

We wish to show that $\mathcal{T}\,[\![\psi]\!]_{\mathcal{I}} = 1$:

$$\begin{aligned} &\mathcal{T}\,[\![\psi]\!]_{\mathcal{I}} = 1 \\ \iff &\forall 1 \le i \le n. \mathcal{T}\left[\!\!\left[\vee_{j=1}^{3} \ell_{ij}\right]\!\!\right]_{\mathcal{I}} = 1 \\ \iff &\forall 1 \le i \le n. \exists 1 \le j \le 3. \mathcal{T}\,[\![\ell_{ij}]\!]_{\mathcal{I}} = 1 \end{aligned}$$

TODO

$\square$

3.3.2.2 Clique

Definition 3.3.12. (**Clique**) For a graph $G = (V, E)$, a set $X \subseteq V$ of vertices is a *clique* $\iff$

$$\forall u, v \in X. (u, v) \in E.$$

Definition 3.3.13. (Clique) The clique problem is defined as a optimization problem:

$$\text{Given a graph } G\text{, determine}$$

$$X = \underset{\text{clique } X}{\arg\max} \, |X|$$

Denoted $\mathsf{Clique}(G)$

Definition 3.3.14. (**Bounded** Clique) The bounded clique problem is defined as a decision problem:

$$\text{Given a graph } G \text{ and } k\text{, determine whether}$$

$$\max \{|X| : \text{clique } X\} \ge k$$

Denoted $\mathsf{Clique}(G, k)$

- **Solution**: $2^{|V|}$ subsets of V. $O(|V|^2)$ method of determining whether $X \subseteq V$ is a clique.
- **Complexity**: $O(2^{|V|}|V|^2)$.

Lemma 3.3.6. $\mathsf{Clique} \in \mathsf{NP}$

Theorem 3.3.3. Clique is NP-complete

Proof. Suffices to prove that $\mathsf{Independent} \leq_{\mathsf{P}} \mathsf{Clique}$. Define the reduction $f : \mathsf{Independent} \to \mathsf{Clique}$ s.t

$$f(G, k) = (\overline{G}, k)$$

where $G = (V, E)$, $\overline{G} = (V, \overline{E})$ and

$$\overline{E} = \left\{(u, v) \in V^2 : (u, v) \notin E\right\}.$$

We have

$$\begin{aligned}
&(G, k) \in \mathsf{Independent} \iff (\overline{G}, k) \in \mathsf{Clique} \\
&\iff \exists I \subseteq V.|I| = k \land (\forall u, v \in I.(u, v) \notin E) \iff \exists X \subseteq V.|X| = k \land \left(\forall u, v \in X.(u, v) \in \overline{E}\right) \\
&\iff \exists I \subseteq V.|I| = k \land (\forall u, v \in I.(u, v) \notin E) \iff \exists X \subseteq V.|X| = k \land (\forall u, v \in X.(u, v) \notin E)
\end{aligned}$$

By α-equivalence, f is a reduction. f is a polynomial reduction w/ time bound $O(|V|^2)$. So we have $\mathsf{Independent} \leq_{\mathsf{P}} \mathsf{Clique}$. □

3.3.2.3 Graph Colorability

Definition 3.3.15. (**k-colorable**) For a $G = (V, E)$, G is k-colorable if there exists a *coloring*

$$\chi : V \to [0, k-1],$$

s.t

$$\forall (u, v) \in E.\chi(u) \neq \chi(v)$$

Definition 3.3.16. (k-$\mathsf{Colorable}$) The k-$\mathsf{Colorable}$ problem is defined as a decision problem:

> Given a graph $G = (V, E)$, determine whether G is k-colorable.

Denoted $k\text{-}\mathsf{Colorable}(G)$

- **Solution**: Determine $k^{|V|}$ possible functions $\chi : V \to [1, k]$. $O(|V|^2)$ algorithm for determining whether χ is a k-coloring.
- **Complexity**: $O(k^|V||V|^2)$.
- $2\text{-}\mathsf{Colorable} \in \mathsf{P}$ (determine whether G is Bipartite)

Theorem 3.3.4. $3\text{-}\mathsf{Colorable}$ is NP-complete.

Proof. (*Sketch*) Suffices to prove $\mathsf{3CNF} \leq_{\mathsf{P}} 3\text{-}\mathsf{Colorable}$. Define the reduction $f : \mathsf{3CNF} \to 3\text{-}\mathsf{Colorable}$ s.t

$$f(\bigwedge_{i=1} \bigvee_{j=1}^{3} \ell_{ij}) = G,$$

where $G = (V, E)$ and

$$\begin{aligned} V &= \{a\} \cup \bigcup_{P \in [\![\psi]\!]_P} \{v_P, v_{\neg P}\} \cup \{v_{ij} : 1 \leq i \leq n, 1 \leq j \leq 3\} \\ E &= \bigcup_{P \in [\![\psi]\!]_P} \{(a, v_P), (a, v_{\neg P}), (v_P, v_{\neg P})\} \\ &\quad \cup \left\{(v_{ij}, v_{\ell_{ij}}) : 1 \leq i \leq n, 1 \leq j \leq 3\right\} \end{aligned}$$

We now show that $\forall \psi \in \Sigma^*. \psi \in \mathsf{3CNF} \iff f(\psi) \in 3\text{-}\mathsf{Colorable}$. Let $\psi \in \Sigma^*$ be arbitrary.
($\implies$). Let us assume that $\psi = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{3} \ell_{ij} \in \mathsf{3CNF}$. So there exists an interpretation $\mathcal{I} \in \Sigma_{\mathcal{I}}$ s.t $\mathcal{T}\,[\![\psi]\!]_{\mathcal{I}} = 1$. Define the coloring $\chi : V \to \{0, 1, 2\}$ s.t

$$\begin{aligned} \chi(a) &= 2 \\ \chi(v_P) &= \begin{cases} 1 & \text{if } \mathcal{I}(P) = 1 \\ 0 & \text{otherwise} \end{cases} \\ \chi(v_{\neg P}) &= \begin{cases} 1 & \text{if } \mathcal{I}(P) = 0 \\ 0 & \text{otherwise} \end{cases} \\ \chi(v_{i1}), \chi(v_{i2}), \chi(v_{i3}) &= 0, 1, 2 \qquad \text{wlog } \mathcal{T}\,[\![\ell_{i1}]\!] = 1, \mathcal{T}\,[\![\ell_{i2}]\!] = 0 \end{aligned}$$

χ is trivially a coloring.
($\Longleftarrow$). Let us assume that $f(\psi) = G \in \textsf{3-Colorable}$. Without loss of generality (by renaming), there exists a coloring $\chi : V \to \{0, 1, 2\}$ s.t $\chi(a) = 2$. Hence $\{\chi(v_P), \chi(v_{\neg P})\} = \{0, 1\}$.

Define the interpretation $\mathcal{I} \in \Sigma_{\mathcal{I}}$ s.t

$$\mathcal{I}(P) = 1 \iff \chi(v_P) = 1$$
$$\mathcal{I}(P) = 0 \iff \chi(v_{\neg P}) = 1$$

We wish to show that $\mathcal{T}\,[\![\psi]\!]_{\mathcal{I}} = 1$. It suffices to show that $\forall 1 \leq i \leq n. \exists 1 \leq j \leq 3. \chi(v_{ij}) = 1$.

Let $1 \leq i \leq n$ be arbitrary. We proceed by contradiction. Let us assume that $\forall 1 \leq j \leq 3. \chi(v_{ij}) \neq 1 \in \{0, 2\}$. So we have $\chi(v_{i1}), \chi(v_{i2}), \chi(v_{i3}) \in \{0, 2\}$, so by the Pigeonhole principle, there exists $1 \leq j < j' \leq 3$ s.t $\chi(v_{ij}) = \chi(v_{ij'})$. A contradiction! □

3.3.2.4 Hamiltonian Graph

Definition 3.3.17. (**Hamiltonian Cycles and Graphs**) For a graph $G = (V, E)$, a Hamiltonian cycle is a $v \longrightarrow^*_{\textsf{path}} v$ path for some $v \in V$ and $[\![v \longrightarrow^*_{\textsf{path}} v]\!]_V = V$.

A graph G is *Hamiltonian* if it contains a *Hamiltonian cycle*.

Definition 3.3.18. ($\textsf{Hamiltonian}$) The Hamiltonian graph problem is defined as the *decision problem*:

Given a graph G, determine whether G is Hamiltonian.

Denoted $\textsf{Hamiltonian}(G)$.

- **Solution**: Generate $O(|V|!)$ permutations of vertices. Verify permutation is a Hamiltonian cycle in $O(|V|)$ time.
- **Complexity**: $O(|V|!|V|)$

Lemma 3.3.7. $\textsf{Hamiltonian} \in \textsf{NP}$

Theorem 3.3.5. $\textsf{Hamiltonian}$ is $\textsf{NP}$-complete.

3.3.2.5 TSP

Definition 3.3.19. (**Travelling Salesperson Problem**) The travelling salesperson problem is defined as the *optimization problem*:

Given a graph $G = (V, V \times V)$ w/ symmetric cost function $c : V \times V \to \mathbb{N}$ and $V = \{v_1, \ldots, v_n\}$. Define the cost of a permutation $\pi : V \to V$ as

$$C(\pi) = c(\pi(v_1), \pi(v_n)) + \sum_{i=1}^{n-1} c(\pi(v_i), \pi(v_{i+1}))$$

Determine a permutation $\pi : V \to V$ s.t $\pi = \arg\min_{\pi} C(\pi)$.

Denoted $\mathsf{TSP}(G, c)$.

- Optimization problem $\to$ bounded decision problem

Definition 3.3.20. (**Bounded TSP**) The bounded TSP is defined as the *decision problem*:

Given a graph $G = (V, E)$ w/ symmetric cost function $c : V \times V \to \mathbb{N}$, and $k \in \mathbb{N}$, determine

$$C(\mathsf{TSP}(G, c)) \leq k$$

Denoted $\mathsf{TSP}(G, c, k)$

- **Complexity**:
  - Upper bound: Brute force: Check all permutations $O(n!)$
  - Lower bound: $\Omega(n \log n)$

Lemma 3.3.8. $\mathsf{TSP} \in \mathsf{NP}$

Theorem 3.3.6. TSP is NP-complete.

Proof. TODO REDUCTION FROM HAM TO TSP □

3.3.3 Sets and Numbers

3.3.3.1 Tripartite Matching

Definition 3.3.21. A matching on disjoint X, Y, Z s.t $|X| = |Y| = |Z| = n$ is a set $M \subseteq X \times Y \times Z$ satisfying

$$\forall (x, y, z), (x', y', z') \in M. x \neq x' \wedge y \neq y' \wedge z \neq z',$$

and $|M| = n$.

- **Note**: Generalization of $\mathsf{Matching}$ (see section ??)

Definition 3.3.22. (**Tripartite Matching**) The tripartite matching problem is defined as the decision problem:

$$\text{Given disjoint } X, Y, Z \text{ and } M\text{, is } M \text{ a matching}$$

Denoted $\mathsf{3Matching}(X, Y, Z, M)$

Lemma 3.3.9. $\mathsf{3Matching} \in \mathsf{NP}$

Theorem 3.3.7. $\mathsf{3Matching}$ is NP-complete.

Proof. Suffices to show that $\mathsf{3CNF} \leq_{\mathsf{P}} \mathsf{3Matching}$, by defining a polynomial reduction $f : \mathsf{3CNF} \to \mathsf{3Matching}$, where

$$f\left(\bigwedge_{i=1}^{n} \bigvee_{j=1}^{3} \ell_{ij}\right) = (X, Y, Z)$$

where $[\![\psi]\!]_P = \{P_1, \ldots, P_m\}$ and

$$\begin{aligned}
X = &\bigcup_{P \in [\![\psi]\!]_P} \{x_{P1}, \ldots, x_{Pn}\} \\
&\cup \{x_{C_1}, \ldots, x_{C_n}\} \cup \underbrace{\{x'_1, \ldots, x'_{n(m-1)}\}}_{\text{Dummy elements}} \\
Y = &\bigcup_{P \in [\![\psi]\!]_P} \{y_{P1}, \ldots, y_{Pn}\} \\
&\cup \{y_{C_1}, \ldots, y_{C_n}\} \cup \underbrace{\{y'_1, \ldots, y'_{n(m-1)}\}}_{\text{Dummy elements}} \\
Z = &\bigcup_{P \in [\![\psi]\!]_P} \{z_{P1}, \ldots, z_{Pn}, \overline{z}_{P1}, \ldots, \overline{z}_{Pn}\}
\end{aligned}$$

Dummy elements required since $|Z| = 2mn$ and (without dummy elements) $|X| = |Y| = mn + n$. However $|X| = |Y| = |Z|$ is required.

Define the set N s.t

$$N = \bigcup_{P \in [\![\psi]\!]_P} \left\{(x_{Pi}, y_{Pi}, z_{Pi}), (x_{Pi}, y_{P(i+1)}, \overline{z}_{Pi}) : 1 \leq i < n\right\} \cup \left\{(x_{Pn}, y_{Pn}, z_{Pn}), (x_{Pn}, y_{P1}, z_{Pn})\right\}$$
$$\cup \left\{(x_{C_i}, y_{C_i}, z_{Pi}) : 1 \leq i \leq n, P \in C_i\right\} \cup \left\{(x_{C_i}, y_{C_i}, \overline{z}_{Pi}) : 1 \leq i \leq n, \neg P \in C_i\right\}$$
$$\cup \underbrace{\left\{(x'_i, y'_j, z) : z \in Z, 1 \leq i, j \leq n(m-1)\right\}}_{\text{Dummy variables}}$$

We wish to show $\forall \psi \in \Sigma^*. \psi \in \mathsf{3CNF} \iff f(\psi) \in \mathsf{3Matching}$. Let $\psi \in \Sigma^*$ be arbitrary.

($\implies$). Let us assume that $\psi = \bigwedge_{i=1}^n \bigvee_{j=1}^3 \ell_{ij} \in \mathsf{3CNF}$. Hence an interpretation $\mathcal{I} \in \Sigma_{\mathcal{I}}$ s.t $\mathcal{T}[\![\psi]\!]_{\mathcal{I}} = 1$.

We wish to find $M \subset N$ s.t M is a matching. We have

$$M = \left\{(x_{Pi}, y_{P(i+1)}, \overline{z}_{Pi}) : 1 \leq i \leq n, \mathcal{I}(P) = 1\right\} \cup \left\{(x_{Pi}, y_{P(i+1)}, z_{Pi}) : 1 \leq i \leq n, \mathcal{I}(P) = 0\right\}$$
$$\cup \left\{(x_{C_i}, y_{C_i}, z_{Pi}) : 1 \leq i \leq n, P \in C_i, \mathcal{I}(P) = 1\right\}$$
$$\cup \left\{(x_{C_i}, y_{C_i}, \overline{z}_{Pi}) : 1 \leq i \leq n, P \in C_i, \mathcal{I}(P) = 0\right\}$$

(*wlog only a true literal in* C_i *for all* $1 \leq i \leq n$).

M is a matching and $|M| = mn + mn = 2mn$.

($\impliedby$). TODO □

3.3.3.2 Set Covering

Definition 3.3.23. (**Set Covering**) For a family $\mathcal{F} = \{S_1, \ldots, S_n\} \subseteq \mathcal{P}(\mathscr{U})$, a family $\mathcal{G} \subseteq \mathcal{F}$ *covers* $\mathscr{U}$ $\iff$ $\bigcup \mathcal{G} = \mathsf{U}$.

Definition 3.3.24. ($\mathsf{SetCovering}$) The set covering problem is defined as the decision problem:

> Given a family $\mathcal{F} \subseteq \mathcal{P}(\mathscr{U})$, a universe $\mathscr{U}$, and $1 \leq k \leq |\mathcal{F}|$. Determine whether there exists $\mathcal{G} \subseteq \mathcal{F}$ s.t $\mathcal{G}$ covers $\mathscr{U}$ and $|\mathcal{G}| = k$.

Denoted $\mathsf{SetCovering}(\mathcal{F}, \mathscr{U}, k)$.

Lemma 3.3.10. $\mathsf{SetCovering} \in \mathsf{NP}$

Definition 3.3.25. (3SetCovering) The 3 set covering problem is defined as the decision problem:

Given a family $\mathcal{F} \subseteq \{S \in \mathcal{P}(\mathscr{U}) : |S| = 3\}$, a universe $\mathscr{U}$ w/ $|\mathscr{U}| = 3n$.
Determine whether there exists $\mathcal{G} \subseteq \mathcal{F}$ s.t $\mathcal{G}$ covers $\mathscr{U}$ and $|\mathcal{G}| = n$.

Denoted $\mathsf{3SetCovering}(\mathcal{F}, \mathscr{U})$.

Theorem 3.3.8. 3SetCovering is NP-complete.

Proof. (*Sketch*) Suffices to show 3Matching $\leq_{\mathsf{P}}$ 3SetCovering, with reduction

$$f(X, Y, Z, M) = (\{\{x, y, z\} : (x, y, z) \in M\}, X \cup Y \cup Z).$$

□

Theorem 3.3.9. SetCovering is NP-complete.

Proof. (*Sketch*) Suffices to show 3SetCovering $\leq_{\mathsf{P}}$ SetCovering, with reduction

$$f(\mathcal{F}, \mathscr{U}) = (\mathcal{F}, \mathscr{U}, |\mathscr{U}|/3).$$

□

3.3.3.3 Knapsack and Pseudo-polynomial Time

Definition 3.3.26. (Knapsack) The knapsack problem is defined as the constrained optimization problem:

Given n items w/ value $v_i \in \mathbb{N}$ and weight $w_i \in \mathbb{N}$, and bounds W and V, determine whether

$$\sum_{i=1}^{n} v_i I_i \geq V$$

$$\sum_{i=1}^{n} w_i I_i \leq W$$

where $I_i = 0$ if the ith item is used.

Denoted $\mathsf{Knapsack}(n, \mathbf{v}, \mathbf{w}, V, W)$

Theorem 3.3.10. Knapsack is NP-complete.

Proof. (*Sketch*) Suffices to show that $\mathsf{3SetCovering} \leq_{\mathsf{P}} \mathsf{Knapsack}$. Let us define the reduction

$$f(\mathcal{F}, \mathscr{U}) = (m, \mathbf{v}, \mathbf{w}, V, W),$$

where $\mathcal{F} = \{S_1, \ldots, S_m\}$, $\mathscr{U} = \{1, \ldots, 3n\}$ and

$$v_i = w_i = \sum_{j \in S_i} (m+1)^{j-1}$$
$$V = W = \sum_{j=0}^{3n-1} (m+1)^j = (m+1)^{3n} - 1$$

Idea: items represent bit vectors and values and weights of sets S_i are $m+1$ base integers (required to avoid carry bits). Union of bit vectors is addition $\implies$ a set covers $\mathscr{U} \iff$ bit vector of $111\ldots1_{m+1}$. □

Theorem 3.3.11. $\mathsf{Knapsack}$ has a time bound of $O(nW)$.

- **Note**: This complexity doesn't reflect the input size: $O(n \log W)$. Hence the complexity is *exponential*.

3.3.4 co-NP

Definition 3.3.27. (co-NP) We define $\mathsf{co\text{-}NP} = \overline{\mathsf{NP}}$. Equivalently, $\mathcal{L} \in \mathsf{co\text{-}NP} \iff$

$$\mathcal{L} = \{x \in \Sigma^*. \forall y \in \Sigma^*. \neg R(y, x)\}.$$

- **Note**: co-NP problems require checking *all polynomial certificates falsify R.*

Theorem 3.3.12. $\mathsf{P} \subseteq \mathsf{co\text{-}NP}$.

Proof. Since $\mathsf{P} = \mathsf{co\text{-}P}$, and $\mathsf{P} \subseteq \mathsf{NP}$, hence $\mathsf{P} = \mathsf{co\text{-}P} \subseteq \mathsf{co\text{-}NP}$ □

- Hence $\mathsf{P} \subseteq \mathsf{NP} \cap \mathsf{co\text{-}NP}$.

IMAGE

Theorem 3.3.13.

$\forall \mathcal{L}_1 \subseteq \Sigma_1^*, \mathcal{L}_2 \subseteq \Sigma_2^*. \exists f : \mathcal{L}_1 \to \mathcal{L}_2$ reduction $\implies f : \overline{\mathcal{L}_1} \to \overline{\mathcal{L}_2}$ is a reduction

Proof. Let $\mathcal{L}_1 \subseteq \Sigma_1^*, \mathcal{L}_2 \subseteq \Sigma_2^*$ be arbitrary. Let us assume there exists a reduction $f : \mathcal{L}_1 \to \mathcal{L}_2$. Let x be arbitrary. We have

$$\begin{aligned} x \in \overline{\mathcal{L}_1} &\iff x \notin \mathcal{L}_1 \\ &\iff f(x) \notin \mathcal{L}_2 \\ &\iff f(x) \in \overline{\mathcal{L}_2} \end{aligned}$$

Hence $f : \overline{\mathcal{L}_1} \to \overline{\mathcal{L}_2}$.

□

Corollary 3.3.13.1. If $\mathcal{L}$ is NP-complete, then $\overline{\mathcal{L}}$ is $\mathsf{co\text{-}NP}$-complete.

- $\overline{\mathsf{SAT}}$ is $\mathsf{co\text{-}NP}$-complete.

3.3.4.1 Primality

Definition 3.3.28. $\mathsf{Comp} = \{x \in 1\{0,1\}^* : x \text{ is composite}\}$ and $\mathsf{Prime} = \{x \in 1\{0,1\}^* : x \text{ is prime}\}$.

- Comp is in NP:
  - Certificate is the factorization (not necessarily the prime factorization). Divisor length is bounded by $O(\log \sqrt{x})$

  Hence Prime is in $\mathsf{co\text{-}NP}$.

Theorem 3.3.14. $\mathsf{Comp} \in \mathsf{NP}$. $\mathsf{Prime} \in \mathsf{co\text{-}NP}$.

Proof. Algorithm for determine whether n is composite:

- Search the range $[1, \sqrt{n}]$ for a *factor*.
- The factor is the certificate y (of length $O(\log_2 \sqrt{n})$)
- $R(y, x) = y \mid x$. $O(|y|^2)$ time complexity. Deterministic.

So by definition ??, $\mathsf{Comp} \in \mathsf{NP}$. Hence by definition ??, $\mathsf{Prime} = \overline{\mathsf{Comp}} \in \mathsf{co\text{-}NP}$. □

Theorem 3.3.15. $p \in \mathbb{P} \iff$ there exists $1 < i < p$ s.t $i^{p-1} \equiv 1 \pmod p$ and $i^{\frac{p-1}{q}} \not\equiv 1 \pmod p$ for all prime divisors q of p.

Theorem 3.3.16. $\mathsf{Prime} \in \mathsf{NP}$.

Proof. Determining $a^b \pmod p$:

- Compute $a^2, a^4, \ldots, a^{2^{\log_2 b}}$ on modulo p, which requires $\log_2 b$ multiplications.
- **Complexity**: $O(\log_2^3 b)$.

So we may determine $i^{p-1} \equiv 1 \pmod p$ in $O(\log_2^3 p)$.
Determining $i^{\frac{p-1}{q}} \not\equiv 1 \pmod p$:

- Requires prime divisors $\mathbf{q} = (q_1, \ldots, q_k)$ of $p - 1$
- Uses algorithm to determine $i^{\frac{p-1}{q_j}} \pmod p$ for all $q_j \in \mathbf{q}$.
- **Complexity**: $O(k \log_2^3 p) = O(\log_2^4 p)$

Certificate:

$$C(p) = (i, \{(q_i, C(q_i)) : q_i \in \mathbf{q}\})$$

w/ $|C(p)| \leq 4 \log_2^2 p$ (strong induction on p). □

3.4 Function Classes

- **Problem**: Complexity classes only reason about *decidable* languages. Not about *functions*.

Definition 3.4.1. (**Witness Function**) A witness function for $\mathcal{L} \subseteq \Sigma^*$ is a function $f : \Sigma^* \to \mathcal{O}(\Sigma^*)$ such that:

– If $x \in \mathcal{L}$, then $f(x) = \lfloor y \rfloor$ s.t $R(y, x)$ where y is the certificate of x

– Otherwise $f(x) = \emptyset$.

- **Note**: This definition applies for all $\mathcal{L}$ (not just NP)

Definition 3.4.2. The function class of a complexity class $\mathcal{C}$ is the class $\mathsf{F}\mathcal{C} = \{\text{witness } f_{\mathcal{L}} : \mathcal{L} \in \mathcal{C}\}$.

- $\implies$ FP and FNP.

3.4.1 Reductions

TODO

3.4.2 Problems

3.4.2.1 Factorization

Definition 3.4.3. (**Factorization Problem**) The factorization problem is the witness function:

$$f(n) = (2, k_1; 3, k_2; \ldots; p_m, k_m),$$

s.t $n = \prod_i p_i^{k_i}$.

- Factorization is FNP since the witness function is for a language $\mathsf{Comp} \in \mathsf{NP}$.

3.4.3 Cryptography

- **Problem**: 2 parties (Alice and Bob) wish to send a message x w/out a third party (Eve) knowing x.
- **Solution**: Encryption:
  - Alice and Bob define $E, D : \Sigma^* \times \Sigma^* \to \Sigma^*$ and $e, d \in \Sigma^*$ s.t

$$\forall x \in \Sigma^*. D(E(x, e), d) = x$$

- $D, E \in \mathsf{P}$ for the cryptographic system to be *tractable*.
- **Types**:
  - Private-key: e, d are only known to A and B respectively.
  - Public-key: e is public and d is only known to B.

Example 3.4.1. (**XOR Encryption**) Define $D, E = \oplus$ and $d = e$ (known as a *one-time pad*). Relies on the identity $(x \oplus e) \oplus e = x$.

Provably secure if Eve doesn't know e: since $e = x \oplus y$.

3.4.3.1 One-way Functions

- **Problem**: For public-key systems: The function that maps $E(x, e) \mapsto x$ (w/out d) $\notin \mathsf{FP} \implies \mathsf{FNP}$.
- Hence public-key systems rely on $\mathsf{FNP} \neq \mathsf{FP}$.

Definition 3.4.4. (**One-way Functions**) A function $f : \Sigma^* \to \Sigma^*$ is a *one-way function* $\iff$:

1. f is injective
2. $\forall x \in \Sigma^*. \exists k \in \mathbb{N}. |x|^{1/k} \leq |f(x)| \leq |x|^k$.
3. $f \in \mathsf{FP}, f^{-1} \notin \mathsf{FP}$

- Proving f is a one-way function relies on assuming $\mathsf{P} \neq \mathsf{NP}$. So *security* of f is shown by proving: $\mathsf{P} \neq \mathsf{NP} \implies f$ is one-way. e.g. RSA

Definition 3.4.5. (**Unambiguous**) A non-deterministic Turing machine $M \in \mathsf{NTM}_\Sigma^k$ is *unambiguous* $\iff$

$$\exists!(q, \tau_1, \ldots, \tau_k) \in \mathscr{C}(M).(q_0, \tau_0(u_1), \ldots, \tau_0(u_\ell), \ldots, \tau_0) \longrightarrow_M^* (q, \tau_1, \ldots, \tau_k) \wedge \mathsf{lab}(q) = \mathsf{acc}$$

Definition 3.4.6. (UP) $\mathcal{L} \in \mathsf{UP}$ iff $\mathcal{L}$ is accepted by an unambiguous NTM. Equivalently, $\mathcal{L} \in \mathsf{UP}$ iff

$$\mathcal{L} = \{x \in \Sigma^*. \exists y \in \Sigma^*. R(y, x)\},$$

where

- R is polynomial time computable and balanced.
- $\forall x, y_1, y_2 \in \Sigma^*. R(y_1, x) \wedge R(y_2, x) \implies y_1 = y_2$. Injective.

Lemma 3.4.1. $\mathsf{P} \subseteq \mathsf{UP} \subseteq \mathsf{NP}$

Theorem 3.4.1. $\mathsf{UP} = \mathsf{P} \iff$ there are no one-way functions.

Proof. We proceed by contradiction.
($\implies$). Let us assume there exists a one-way function $f : \Sigma^* \to \Sigma^*$. Define

$$\mathcal{L}_f = \{(x, y) : \exists z \in \Sigma^*. z \leq x \wedge f(z) = y\},$$

where $\leq$ is a lexicographical ordering.

It suffices to show that $\mathcal{L}_f \in \mathsf{UP} \setminus \mathsf{P}$. We first show that $\mathcal{L}_f \in \mathsf{UP}$:

- $R(z, (x, y)) = z \leq x \wedge f(z) = y$ is computable in polynomial time and polynomially balanced by definition of f (??).
- R is injective since f is injective

So by definition ??, $\mathcal{L}_f \in \mathsf{UP}$. We now show that $\mathcal{L}_f \notin \mathsf{P}$. Let us assume that $\mathcal{L}_f \in \mathsf{P}$. Then it follows that we may compute f^{-1} in polynomial time:

- By definition ??, given $y \in \Sigma^*$ s.t $f(z) = y \implies |z|^{1/k} \leq |y| \leq |z|^k$ for some $k \in \mathbb{N}$.
- Use a binary search procedure w/ time bound of $(\log_2 y)^k$

Hence $f^{-1} \in \mathsf{FP}$. A contradiction! So we are done.
($\Longleftarrow$). Let us assume that $\mathsf{UP} \neq \mathsf{P}$. Hence there exists $\mathcal{L} \in \mathsf{UP} \setminus \mathsf{P}$, where U is the unambiguous NTM accepting $\mathcal{L}$.

Define $f_U : \Sigma^* \to \Sigma^*$:

$$f_U(x) = \begin{cases} 1y & \text{if } x \text{ encodes the accepting computation of } U \text{ w/ input } y \\ 0x & \text{otherwise} \end{cases}$$

f_U is injective since U is unambiguous. $f_U \in \mathsf{FP}$ (assuming a polynomial encoding between x and computations). $|f_U(x)| \leq |x|^k$ since U has a polynomial running time. We now show that $f_U^{-1} \notin \mathsf{FP}$. Assume that $f_U^{-1} \in \mathsf{FP}$. Hence $\mathcal{L} \in \mathsf{P}$. A contradiction!

$\square$

3.5 Space Complexity

- **Idea**: Reason about relations between time and space complexity classes

3.5.1 Classes

Definition 3.5.1. (**Space Complexity Class**) Let $f : \mathbb{N} \to \mathbb{N}$. $\mathsf{SPACE}(f(n)) \subseteq \mathcal{P}(\Sigma^*)$ is the complexity class (set of languages) s.t $\mathcal{L} \in \mathsf{SPACE}(f(n)) \iff \mathcal{L}$ is decidable by a TM w/ space bound $O(f(n))$.

Definition 3.5.2. (**NSpace Complexity Class**) Let $f : \mathbb{N} \to \mathbb{N}$. $\mathsf{NSPACE}(f(n)) \subseteq \mathcal{P}(\Sigma^*)$ is the complexity class (set of languages) s.t $\mathcal{L} \in \mathsf{NSPACE}(f(n)) \iff \mathcal{L}$ is decidable by a NTM w/ space bound $O(f(n))$.

- See definition ?? for space bound (relies on TM's w/ I/O)

Definition 3.5.3. (**Logarithmic Space Complexity**) The logarithmic space complexity class LSPACE is defined by

$$\mathsf{LSPACE} = \mathsf{SPACE}(\log n).$$

Similarly, the non-deterministic logarithmic space complexity class $\mathsf{NLSPACE}$ is

$$\mathsf{NLSPACE} = \mathsf{NSPACE}(\log n).$$

Definition 3.5.4. (**Polynomial Space Complexity**) The polynomial space complexity class PSPACE is defined by

$$\mathsf{PSPACE} = \bigcup_{k=1}^{\infty} \mathsf{SPACE}(n^k).$$

The non-deterministic polynomial space complexity class $\mathsf{NPSPACE}$ is

$$\mathsf{NPSPACE} = \bigcup_{k=1}^{\infty} \mathsf{NSPACE}(n^k).$$

3.5.2 Inclusions

- **Idea**: Prove $\subseteq$ relations between complexity classes:

$$\mathsf{LSPACE} \subseteq \mathsf{NLSPACE} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} \subseteq \mathsf{NPSPACE}.$$

Lemma 3.5.1. (**Deterministic Completement Lemma**) For all $\mathcal{L} \subseteq \Sigma^*$:

$$\mathcal{L} \in \mathsf{TIME}(f(n)) \implies \overline{\mathcal{L}} \in \mathsf{TIME}(f(n))$$
$$\mathcal{L} \in \mathsf{SPACE}(f(n)) \implies \overline{\mathcal{L}} \in \mathsf{SPACE}(f(n))$$

Corollary 3.5.0.1. $\mathsf{LSPACE} = \mathsf{co\text{-}LSPACE}$, $\mathsf{P} = \mathsf{co\text{-}P}$ and $\mathsf{PSPACE} = \mathsf{co\text{-}PSPACE}$.

- **Problem**: Functions $f : \mathbb{N} \to \mathbb{N}$ may not be computable (in the complexity class they describe) $\implies$ difficult for constructions on bounded computation.

- **Solution**: *Proper function* (or *constructible functions*).

Definition 3.5.5. (**Proper Function**) A function $f : \mathbb{N} \to \mathbb{N}$ is a *proper function* iff:

(i) f is non-decreasing: $\forall n \in \mathbb{N}. f(n) \leq f(n+1)$.

(ii) f is computable w/ a $O(|x| + f(|x|))$ time bound and $O(f(|x|))$ space bound, where $|x| = |\, [\![n]\!]_{\mathbb{N}} \,|$ (the length of the encoding).

Lemma 3.5.2. For all $f, g : \mathbb{N} \to \mathbb{N}$, if f, g are constructible, then

1. $f + g, f \times g$ and 2^f are constructible
2. $f \circ g$ is constructible iff $\forall n \in \mathbb{N}. f(n) > n$.

Proof. (*Sketch*) Relies on re-computing f determining symbol a at index i on output tape to obtain output w/out affecting space bound. TODO □

Theorem 3.5.1. (**The Inclusion Theorem**) For all constructible functions $f : \mathbb{N} \to \mathbb{N}$:

(i) $\mathsf{SPACE}(f(n)) \subseteq \mathsf{NSPACE}(f(n))$

(ii) $\mathsf{TIME}(f(n)) \subseteq \mathsf{NTIME}(f(n))$

(iii) $\mathsf{NTIME}(f(n)) \subseteq \mathsf{SPACE}(f(n))$

(iv) $\mathsf{NSPACE}(f(n)) \subseteq \mathsf{TIME}(k^{\log n + f(n)})$

Proof. Let $f : \mathbb{N} \to \mathbb{N}$ be an arbitrary constructible function.

(i), (ii) (i) and (ii) are trivial.

(iii) Let $\mathcal{L} \in \mathsf{NTIME}(f(n))$. Let $N \in \mathsf{NTM}^k_\Sigma$ be the NTM that decides $\mathcal{L}$ in $f(n)$ time. The computation tree $\mathscr{T}$ of N is given by: IMAGE

where $b = \max\left\{|S| : S \in \overrightarrow{\Delta}(Q, \mathcal{O}(\Sigma)^k)\right\}$ is branching factor of the computation tree.

A bounded depth-first search on $\mathscr{T}$ is computable w/ space bound of $d = f(n)$ (See AI supervision work). Hence $\mathcal{L} \in \mathsf{SPACE}(f(n))$.

(iv) Let $\mathcal{L} \in \mathsf{NSPACE}(f(n))$. Let $N \in \mathsf{NTM}^k_\Sigma$ be the NTM that decides $\mathcal{L}$ in $f(n)$ space.

The configuration graph of N w/ input $x \in \Sigma^*$ is a graph $G = (V, E)$ inductively defined by

$$\frac{}{(q_0, \tau_0(x), \tau_0, \dots, \tau_0) \in V}$$

$$\frac{c \in V \qquad c \longrightarrow_N c'}{c' \in V, (c, c') \in E}$$

M accepts $x \iff$ an accepting configuration is *reachable* from c_0. We note that $|V| \leq |\mathscr{C}_x(N)|$, where $|\mathscr{C}_x(N)|$ is the number of possible configuration of N w/ input x:

$$|Q| \times \underbrace{|x|}_{\text{position on input}} \times \Big[\underbrace{(|\Sigma| + 1)^{f(|x|)}}_{\text{tape contents}} \times \underbrace{f(|x|)}_{\text{position on tape}} \Big]^{k-2}$$

Note that $\mathsf{Reachability}$ has a time complexity of $O(|V|^2)$ (see section ??). Hence $\mathcal{L}$ is decidable in

$$\begin{aligned} O\left(\left\{|Q| \times |x| \times [(|\Sigma|+1)^{f(|x|)} \times f(|x|)]^{k-2}\right\}^2\right) &= O\left(\left\{|x|[c^{\log_c |x| + f(|x|)}]^{k-2}\right\}^2\right) \\ &= O\left(c^{\log_c |x| + f(|x|)}\right) \end{aligned}$$

So we have $\mathcal{L} \in \mathsf{TIME}(c^{\log n + f(n)})$.

□

Theorem 3.5.2. ($\mathsf{NLSPACE}$ **Reachability**) $\mathsf{Reachability} \in \mathsf{NLSPACE}$

Proof. $\mathsf{Reachability} \in \mathsf{NLSPACE}$ w/ the following algorithm:

```
i ← index_of(u); // current index
while (true) {
    if (V[i] = v) return acc;

    non-deterministically determine index j (log|V| bits);
    if ((V[i], V[j]) ∉ E) return rej;
    i ← j;
}
```

Requires $2 \log |V|$ space. Hence $\mathsf{Reachability} \in \mathsf{NLSPACE}$. □

3.5.3 Savitch's Theorem

Theorem 3.5.3. (**Savitch's Theorem**) $\mathsf{Reachability} \in \mathsf{SPACE}(\log^2 |V|)$

Proof. $\mathsf{Reachability} \in \mathsf{SPACE}(\log^2 |V|)$ w/ the following algorithm:

```
path(u, v, i) {
    if (i = 1 ∧ u ≠ v ∧ (u,v) ∉ E) return false;
    if ((u,v) ∈ E ∨ u = v) return true;

    for (w ∈ V) {
        if (path(u, w, ⌊i/2⌋)
            ∧ path(w, v, ⌈i/2⌉))
            return true;
    }
}
```

The above algorithm computes `path(`u`, `v`, `i`)`, which defines the predicate: there is a path $u \longrightarrow^k v$ where $k \leq i$. Each stack frame of `path` may be defined by $4 \log |V|$ bits (3 for parameters + 1 for w). The maximum recursion depth is given by $\log |V| \implies$ space bound of $4 \log^2 |V|$

`path(`u`, `v`, `$\lceil \log |V| \rceil$`)` defines reachability.
Hence $\mathsf{Reachability} \in \mathsf{SPACE}(\log^2 |V|)$. □

Corollary 3.5.3.1. For constructible functions $f : \mathbb{N} \to \mathbb{N}$ s.t $\forall n \in \mathbb{N}. f(n) \geq \log n$,

$$\mathsf{NSPACE}(f(n)) \subseteq \mathsf{SPACE}(f(n)^2).$$

Proof. Let $f : \mathbb{N} \to \mathbb{N}$ be an arbitrary constructible function. Let us assume that $\forall n \in \mathbb{N}. f(n) \geq \log n$.

Let $\mathcal{L} \in \mathsf{NSPACE}(f(n))$ be defined as in theorem ??, proof of (iv). Hence $\mathcal{L}$ is decidable in

$$O(\log^2 \left(c^{\log_c |x| + f(|x|)}\right)) = O((\log_c |x| + f(|x|))^2) = O(f(|x|)^2).$$

So $\mathcal{L} \in \mathsf{SPACE}(f(n)^2)$. □

Lemma 3.5.3. $\mathsf{PSPACE} = \mathsf{NPSPACE}$

Proof. By Corollary ??, we have

$$\begin{aligned}\mathsf{NPSPACE} &= \bigcup_{k=1}^{\infty} \mathsf{NSPACE}(n^k)\\ &\subseteq \bigcup_{k=1}^{\infty} \mathsf{SPACE}(n^{2k})\\ &\subseteq \mathsf{PSPACE}\end{aligned}$$

Since $\mathsf{PSPACE} \subseteq \mathsf{NPSPACE}$, then we have $\mathsf{PSPACE} = \mathsf{NPSPACE}$. □

Theorem 3.5.4. (**The Immerman-Szelepscenyi Theorem**) For a given $G = (V, E)$ and vertex $v \in V$, $|\{u \in V : v \longrightarrow^* u\}|$, the number of vertices reachable from v, is computable by a NTM w/ a $\log |V|$ space bound.

Corollary 3.5.4.1. For all constructible functions $f : \mathbb{N} \to \mathbb{N}$ w/ $\forall n \in \mathbb{N}. f(n) \geq \log n$,

$$\mathsf{NSPACE}(f(n)) = \mathsf{co\text{-}NSPACE}(f(n)).$$

3.5.4 Hierarchy Theorem

- **Problem**: Theorems so far prove $\subseteq$ but not $\subset$.

Definition 3.5.6. (**Time Bounded Halting Problem**) Let $f : \mathbb{N} \to \mathbb{N}$ be a constructible function s.t $\forall n \in \mathbb{N}. f(n) \geq n$. The *time bounded halting problem* H_f is defined as

$$H_f = \{(\llbracket M \rrbracket, x) : M \in \mathsf{TM}_\Sigma \text{ accepts } x \text{ after } f(|x|) \text{ transitions}\},$$

where $\llbracket \cdot \rrbracket : \mathsf{TM}^k_\Sigma \to \Sigma^*_+$ is an encoding function for Turing machines.

Lemma 3.5.4. $H_f \in \mathsf{TIME}(f(n)^3)$.

Proof. (*Sketch*) Define a Turing machine U_f that decides H_f:

- Computes $f(|x|)$ from input $(\llbracket M \rrbracket_M, x)$. Takes $O(f(n))$ time (by definition of constructible function).
- Universal Turing machine U simulates each transition of M in time $O(\ell_M k_M^2 f(|x|))$ w/ k_M is # of strings of M, ℓ_M is the length of the description of each state and symbol.

  $k_M, \ell_M = O(\log |\llbracket M \rrbracket_M|)$.

  Hence each step takes $O(f^2(|x|))$ time.

So $H_f \in \mathsf{TIME}(f(n)^3)$. $\square$

Lemma 3.5.5. $H_f \notin \mathsf{TIME}(f(\lfloor n/2 \rfloor))$.

Proof. We proceed by contradiction. Suppose $H_f \in \mathsf{TIME}(f(\lfloor n/2 \rfloor))$.

Let $M \in \mathsf{TM}_\Sigma^k$ be the TM that decides H_f in $f(\lfloor n/2 \rfloor)$ time.

Define the TM $N \in \mathsf{TM}_\Sigma^k$ which decides

$$\mathcal{L} = \{[\![M]\!]_M : ([\![M]\!]_M, [\![M]\!]_M) \notin H_f\}.$$

By existence of M, N has the time bound of $f(\lfloor (2n+3)/2 \rfloor) = f(n+1)$ (copies input $[\![M]\!]_M$, adds (,) (3 symbols)). Note that

$$\begin{aligned}
& [\![N]\!]_M \in \mathcal{L} \\
\iff & ([\![N]\!]_M, [\![N]\!]_M) \notin H_f \\
\iff & N \text{ doesn't accept } [\![N]\!]_M \text{ after } f(n) \text{ transitions} \\
\iff & [\![N]\!] \notin \mathcal{L}
\end{aligned}$$

A contradiction!

$\square$

Theorem 3.5.5. (**Time Heirarchy Theorem**) For all $f : \mathbb{N} \to \mathbb{N}$ s.t f is constructible and $\forall n \in \mathbb{N}. f(n) \geq n$,

$$\mathsf{TIME}(f(n)) \subset \mathsf{TIME}(f(2n+1)^3).$$

Corollary 3.5.5.1. $\mathsf{P} \subset \mathsf{EXP}$