

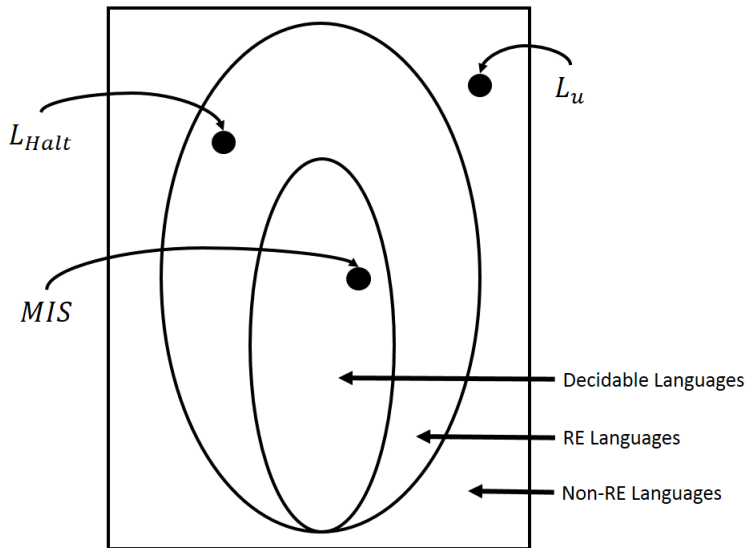
CSL853: Complexity Theory

Ragesh Jaiswal, CSE, IIT Delhi

Complexity Diagram

Complexity Diagram

Our current view of complexity classes



The complexity class P

Complexity Class P

- A *complexity class* is a set of functions that can be computed within a given resource.
- We will be mainly interested in language recognition or *boolean* function computation.

Definition (The class DTIME)

Let $T : \mathbb{N} \rightarrow \mathbb{N}$ be some function. We let $\text{DTIME}(T(n))$ be the set of all Boolean functions that are computable in $c \cdot T(n)$ -time for some constant $c > 0$.

Definition (The class P)

$$P = \bigcup_{c \geq 1} \text{DTIME}(n^c).$$

The complexity class NP

- Verifying solution of a problem is easier than coming up with a solution.

Definition (The class NP)

A language is in NP if there exists a polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ and a polynomial-time TM M such that for every $x \in \{0, 1\}^*$,

$$x \in L \leftrightarrow \exists u \in \{0, 1\}^{p(|x|)} \text{ s.t. } M(x, u) = 1$$

If $x \in L$ and $u \in \{0, 1\}^{p(|x|)}$ satisfy $M(x, u) = 1$, then we call u a *certificate* for x (w.r.t. the language L and machine M).

Definition (The class NP)

A language is in NP if there exists a polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ and a polynomial-time TM M such that for every $x \in \{0, 1\}^*$,

$$x \in L \leftrightarrow \exists u \in \{0, 1\}^{p(|x|)} \text{ s.t. } M(x, u) = 1$$

If $x \in L$ and $u \in \{0, 1\}^{p(|x|)}$ satisfy $M(x, u) = 1$, then we call u a *certificate* for x (w.r.t. the language L and machine M).

- Few Examples:
 - Independent set.
 - Travelling salesperson
 - Subset sum
 - Linear programming

Definition (The class NP)

A language is in NP if there exists a polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ and a polynomial-time TM M such that for every $x \in \{0, 1\}^*$,

$$x \in L \leftrightarrow \exists u \in \{0, 1\}^{p(|x|)} \text{ s.t. } M(x, u) = 1$$

If $x \in L$ and $u \in \{0, 1\}^{p(|x|)}$ satisfy $M(x, u) = 1$, then we call u a *certificate* for x (w.r.t. the language L and machine M).

- So, why is this class of languages called **Non-deterministic Polynomial Time**.

- So, why is this class of languages called **Non-deterministic Polynomial Time**.
 - There is an alternate characterisation in terms of Non-deterministic Turing Machines.
- Non-deterministic Turing Machine (NDTM): Same as a deterministic Turing Machine except that the transition function maps a state and a tape symbol to a “set” of (state, tape symbol, direction) triples (assume that we talk about single tape machines). Acceptance is defined in a similar manner as we defined for NFAs

Complexity Class NP

- Non-deterministic Turing Machine (NDTM): Same as a deterministic Turing Machine except that the transition function maps a state and a tape symbol to a “set” of (state, tape symbol, direction) triples (assume that we talk about single tape machines). Acceptance is defined in a similar manner as we defined for NFAs

Definition (NTIME)

For every function $T : \mathbb{N} \rightarrow \mathbb{N}$ and $L \subseteq \{0, 1\}^*$, we say that $L \in \text{NTIME}(T(n))$ if there is a constant $c > 0$ and a $c \cdot T(n)$ -time NDTM M such that for every $x \in \{0, 1\}^*$, $x \in L \leftrightarrow M(x) = 1$.

Definition (The class NP)

$\text{NP} = \bigcup_{c \in \mathbb{N}} \text{NTIME}(n^c)$.

Complexity Class NP

Definition (The class NP)

A language is in NP if there exists a polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ and a polynomial-time TM M such that for every $x \in \{0, 1\}^*$,

$$x \in L \leftrightarrow \exists u \in \{0, 1\}^{p(|x|)} \text{ s.t. } M(x, u) = 1$$

If $x \in L$ and $u \in \{0, 1\}^{p(|x|)}$ satisfy $M(x, u) = 1$, then we call u a *certificate* for x (w.r.t. the language L and machine M).

Definition (The class NP)

$$\text{NP} = \bigcup_{c \in \mathbb{N}} \text{NTIME}(n^c).$$

- Claim: Above two definitions are equivalent.

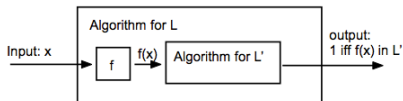
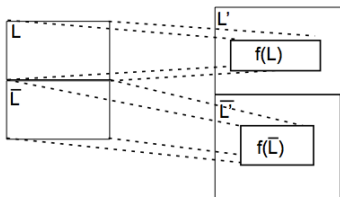
Reduction and NP-completeness

Reduction and NP-completeness

Many-one reduction or Karp-reduction

Definition (Many-one reduction or Karp reduction)

We say that a language $A \subseteq \{0,1\}^*$ is polynomial-time Karp reducible to a language $B \subseteq \{0,1\}^*$ (sometimes shortened to just “polynomial-time reducible” or “many-one reducible”) denoted by $A \leq_p B$ if there is a polynomial-time computable function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ such that for every $x \in \{0,1\}^*$, $x \in A$ if and only if $f(x) \in B$.



Source: Arora-Barak

Reduction and NP-completeness

Many-one reduction or Karp-reduction

Definition (Many-one reduction or Karp reduction)

We say that a language $A \subseteq \{0,1\}^*$ is polynomial-time Karp reducible to a language $B \subseteq \{0,1\}^*$ (sometimes shortened to just “polynomial-time reducible” or “many-one reducible”) denoted by $A \leq_p B$ if there is a polynomial-time computable function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ such that for every $x \in \{0,1\}^*$, $x \in A$ if and only if $f(x) \in B$.

Definition (NP-hard and NP-complete)

We say that a language B is NP-hard if $A \leq_p B$ for every $A \in \text{NP}$. We say that B is NP-complete if B is NP-hard and $B \in \text{NP}$.

Reduction and NP-completeness

NP-hard and NP-complete

Definition (NP-hard and NP-complete)

We say that a language B is NP-hard if $A \leq_p B$ for every $A \in \text{NP}$. We say that B is NP-complete if B is NP-hard and $B \in \text{NP}$.

Theorem

The following are true:

- 1 If $A \leq_p B$ and $B \leq_p C$, then $A \leq_p C$.
- 2 If language A is NP-hard and $A \in \text{P}$, then $\text{P} = \text{NP}$.
- 3 If language A is NP-complete, then $A \in \text{DP}$ if and only if $\text{P} = \text{NP}$.

Reduction and NP-completeness

NP-hard and NP-complete

Definition (NP-hard and NP-complete)

We say that a language B is NP-hard if $A \leq_p B$ for every $A \in \text{NP}$.
We say that B is NP-complete if B is NP-hard and $B \in \text{NP}$.

- Does there exist an NP-complete language?

Reduction and NP-completeness

NP-hard and NP-complete

Definition (NP-hard and NP-complete)

We say that a language B is NP-hard if $A \leq_p B$ for every $A \in \text{NP}$. We say that B is NP-complete if B is NP-hard and $B \in \text{NP}$.

- Does there exist an NP-complete language?

Theorem (An NP-complete language)

The following language is NP-complete

$$\text{TMSAT} = \{ \langle \alpha, x, 1^n, 1^t \rangle : \exists u \in \{0, 1\}^n \text{ s.t. } M_\alpha \text{ outputs 1} \\ \text{on input } \langle x, u \rangle \text{ within } t \text{ steps} \}$$

where M_α denotes the TM represented by the string α .

Reduction and NP-completeness

NP-hard and NP-complete

Theorem (Cook-Levin Theorem)

Let SAT be the language of all satisfiable Boolean formulae, then SAT is NP-complete.

End