Patterns of Reducability

Turing computable functions

- A function $\Sigma^* \rightarrow \Sigma^*$ is a computable function if some Turing Machine M, in every input w, halts with just f(w) on its tape.

Polynomial time function

- A function f: $\Sigma^* \rightarrow \Sigma^*$ is a polynomial time computable function if some polynomial time TM, M, exists that halts with just F(w) on its tape, when started on any input w.

Mapping reducability

- A language A is mapping reducable to language B, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \rightarrow \Sigma^*$, where for every $w \in \Sigma^*$,

$$w \in A \Leftrightarrow F(w) \in B$$

Polynomial time reducability

- A language, A, is polynomial time mapping reducible (or simply polynomial time reducible) to a language, B, written $A \leq_p B$,if a polynomial time computable function f: : $\Sigma^* \rightarrow \Sigma^*$ exists, where for every w,

$$w \in A \iff f(w) \in B$$

- The function f is called the polynomial time reduction of A to B

Decidability Theorems

1. A $\leq_m$ B and B is decidable then A is decidable

2. A $\leq_m$ B and A is undecidable, then B is undecidable

Recognizability Theorems

- A $\leq_m$ B and B is Turing recognizable then A is Turing recognizable

- A $\leq_m$ B and A is not Turing recognizable then B is not Turing recognizable
  - Typically we let A be $\underline{A_{TM}}$ the complement of A_{TM}

Definition

- A language B is NP-complete if it satisfies 2 conditions
  1. B is n NP, and
  2. Every A in NP is polynomial time reducable to B

     Forall $A \in NP$. $A \leq_p B$

P or NP-complete Theorems

- To show a language is in P
  - $A \leq_p B$ and $B \in P$ then $A \in P$

- To Show a language is NP-complete
  - If B is NP-complete and $B \leq_p C$, for $C \in NP$, then C in NP-complete
  - The most common "B" is the language boolean satifiability