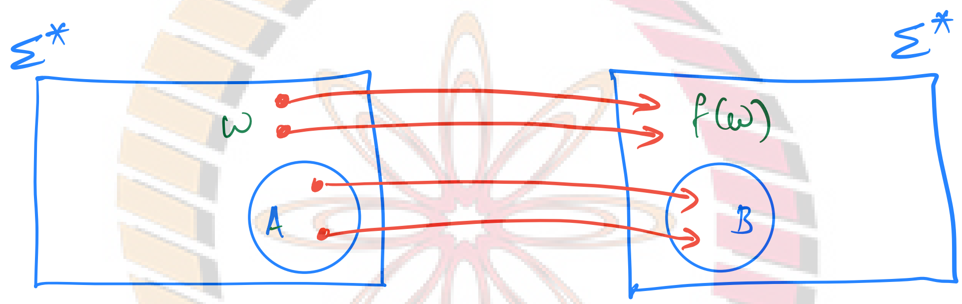
Reductions

Reductions $\rightarrow$ Converting one problem into another.

What can we infer from reductions / conversions?

Let A, B be languages over Σ.



Reduction from A to B: Is a function $f: \Sigma^* \rightarrow \Sigma^*$

such that Strings in $A \longrightarrow$ Strings in B

Strings not in $A \longrightarrow$ Strings not in B.

So we can infer statements like:

$A \leq_m B$ and B is decidable $\Rightarrow$ A is decidable

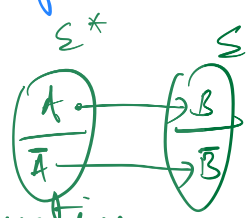
$A \leq_m B$ and A is undecidable $\Rightarrow$ B is undecidable.

Def 5.17: A function $f : \Sigma^* \to \Sigma^*$ is a <u>computable function</u>, if $\exists M$ such that on input w, M computes $f(w)$, writes it and halts.

Def 5.20: language A is <u>mapping (many-one) reducible</u> to language B (denoted $A \leq_m B$) if there is a computable function $f : \Sigma^* \to \Sigma^*$, such that for all $w \in \Sigma^*$,

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

The function f is called the <u>reduction</u> of A to B.



<u>Remarks</u>: (1) $A \leq_m B \iff \bar{A} \leq_m \bar{B}$

The reduction f from A to B is also a reduction from $\bar{A}$ to $\bar{B}$.

(2) There are two things to check:

(a) $w \in A \iff f(w) \in B \quad \forall w$

(b) Is f computable?

(3) Suppose f is a reduction from A to B. Then f^{-1} need not be a reduction from B to A. f^{-1} may not be defined.

Examples: (1) Recall the proof that A_{NFA} is decidable

Given NFA N and string w, we converted N to an equivalent DFA M. Then we used the A_{DFA} decider on $\langle M, w \rangle$.

→ $\langle N, w \rangle \in A_{NFA} \iff \langle M, w \rangle \in A_{DFA}$

→ The NFA to DFA conversion process was computable.

(2) We saw EQ_{DFA} and E_{DFA}.

$$EQ_{DFA} = \{ \langle A, B \rangle \mid A, B \text{ are DFA's}, L(A) = L(B) \}$$

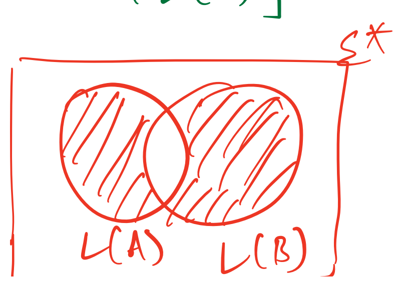
$$E_{DFA} = \{ \langle C \rangle \mid C \text{ is a DFA}, L(C) = \phi \}$$

To show EQ_{DFA} is decidable, we did this:

→ Given $\langle A, B \rangle$, we constructed a DFA C such that

$$L(C) = [L(A) \cap \overline{L(B)}] \cup [\overline{L(A)} \cap L(B)]$$

→ Run E_{DFA} decider on $\langle C \rangle$.



$$L(A) = L(B) \iff L(C) = \phi.$$

$$\langle A, B \rangle \in EQ_{DFA} \iff \langle C \rangle \in E_{DFA}.$$

$$EQ_{DFA} \leq_m E_{DFA}.$$

Since E_{DFA} is decidable, EQ_{DFA} is decidable.

---

(3) $ALL_{CFG} = \{ \langle G \rangle \mid G \text{ is a CFG}, \ L(G) = \Sigma^* \}$

$EQ_{CFG} = \{ \langle G_1, G_2 \rangle \mid G_1, G_2 \text{ are CFG}, \ L(G_1) = L(G_2) \}$

We will see how to reduce ALL_{CFG} to EQ_{CFG}.

* Given $\langle G \rangle$, construct a CFG H that generates all strings in Σ^*.

Note: $L(H) = \Sigma^*$ $H: S \to aS \mid \varepsilon \quad \forall a \in \Sigma.$

* $\langle G \rangle \in ALL_{CFG} \iff L(G) = \Sigma^* = L(H)$

$\iff \langle G, H \rangle \in EQ_{CFG}.$

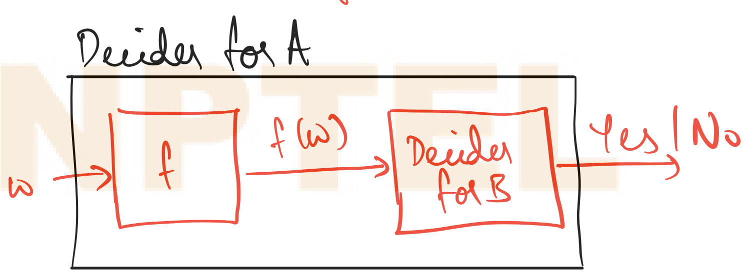
So $ALL_{CFG} \le_m EQ_{CFG}$.

Fact: ALL_{CFG} is undecidable.

If EQ_{CFG} was decidable, this implies that we can reduce the ALL_{CFG} instance to EQ_{CFG} and then use EQ_{CFG} decider to decide ALL_{CFG}. This is a contradiction. Hence EQ_{CFG} is undecidable.

Theorem 5.22: If $A \le_m B$ and B is decidable, then A is decidable.

Proof: Combine the reduction function computer and decider for B to get a decider for A.



Corollary 5.23: If $A \le_m B$ and A is undecidable, then B is undecidable.

Theorem 5.28: If $A \leq_m B$ and B is Turing recognizable, then A is Turing recognizable.

Corollary 5.29: If $A \leq_m B$, and A is not Turing recognizable, then B is not Turing recognizable.