Theory of Computation
Mapping Reducibility

Arjun Chandrasekhar

Mapping Reducibility

Mapping Reducibility

- When we defined ‘reducibility’, we gave an informal definition

Mapping Reducibility

- When we defined ‘reducibility’, we gave an informal definition
- We will give a mathematically precise definition of what it means for one problem to be reducible to another

Computable Function

Computable Function

- So far we have considered machines that take in an input string and output ACCEPT or REJECT

Computable Function

- So far we have considered machines that take in an input string and output ACCEPT or REJECT
- We can also construct machines that take an input and produce an output

Computable Function

- So far we have considered machines that take in an input string and output ACCEPT or REJECT
- We can also construct machines that take an input and produce an output
- Let $f : \Sigma^* \to \Sigma^*$ be a function that takes a string as input and produces another string as output

Computable Function

- So far we have considered machines that take in an input string and output ACCEPT or REJECT
- We can also construct machines that take an input and produce an output
- Let $f : \Sigma^* \to \Sigma^*$ be a function that takes a string as input and produces another string as output
- We say f is a **computable function** if some Turing machine M *computes* f

Computable Function

- So far we have considered machines that take in an input string and output ACCEPT or REJECT
- We can also construct machines that take an input and produce an output
- Let $f : \Sigma^* \to \Sigma^*$ be a function that takes a string as input and produces another string as output
- We say f is a **computable function** if some Turing machine M *computes* f
  - For every input w, M halts and leaves $f(w)$ on the tape, nothing else

Mapping Reducibility

Mapping Reducibility

- Let $A, B \subseteq \Sigma^*$ be formal languages

Mapping Reducibility

- Let $A, B \subseteq \Sigma^*$ be formal languages
- Suppose $f : \Sigma^* \to \Sigma^*$ is a computable function, and $w \in A \Leftrightarrow f(w) \in B$

Mapping Reducibility

- Let $A, B \subseteq \Sigma^*$ be formal languages
- Suppose $f : \Sigma^* \to \Sigma^*$ is a computable function, and $w \in A \Leftrightarrow f(w) \in B$
- We say A is **mapping reducible** to B

Mapping Reducibility

- Let $A, B \subseteq \Sigma^*$ be formal languages
- Suppose $f : \Sigma^* \to \Sigma^*$ is a computable function, and $w \in A \Leftrightarrow f(w) \in B$
- We say A is **mapping reducible** to B
  - We denote this $A \leq_M B$

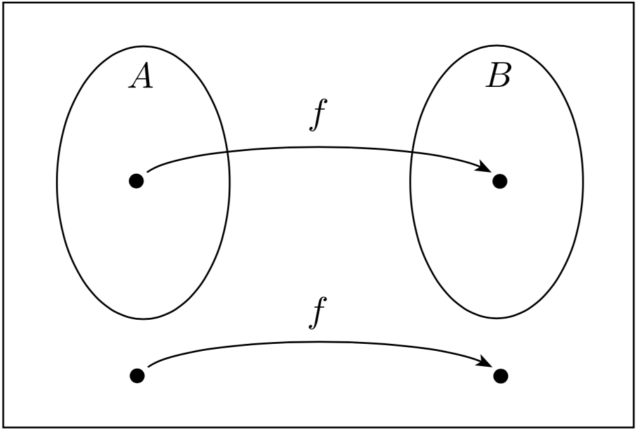
Mapping Reducibility

- Let $A, B \subseteq \Sigma^*$ be formal languages
- Suppose $f : \Sigma^* \to \Sigma^*$ is a computable function, and $w \in A \Leftrightarrow f(w) \in B$
- We say A is **mapping reducible** to B
  - We denote this $A \leq_M B$
  - We say f is a **reduction** from A to B

Mapping Reducibility

"<u>YES maps to YES</u>"

"<u>NO maps to NO</u>"



Mapping vs. Turing Reducibility

Mapping vs. Turing Reducibility

Turing Reducibility:

Mapping vs. Turing Reducibility

Turing Reducibility:

- $A \leq_T B$

Mapping vs. Turing Reducibility

Turing Reducibility:

- $A \leq_T B$
- M_A can call M_B as a subroutine any number of times

Mapping vs. Turing Reducibility

Turing Reducibility:

- $A \leq_T B$
- M_A can call M_B as a subroutine any number of times
- M_A can call M_B at <u>any</u> point in its computation

Mapping vs. Turing Reducibility

Turing Reducibility:

- $A \leq_T B$
- M_A can call M_B as a subroutine any number of times
- M_A can call M_B at <u>any</u> point in its computation

Mapping Reducibility

Mapping vs. Turing Reducibility

Turing Reducibility:

- $A \leq_T B$
- M_A can call M_B as a subroutine any number of times
- M_A can call M_B at <u>any</u> point in its computation

Mapping Reducibility

- $A \leq_M B$

Mapping vs. Turing Reducibility

Turing Reducibility:

- $A \leq_T B$
- M_A can call M_B as a subroutine any number of times
- M_A can call M_B at any point in its computation

Mapping Reducibility

- $A \leq_M B$
- M_A can use M_B as a subroutine exactly once

Mapping vs. Turing Reducibility

Turing Reducibility:

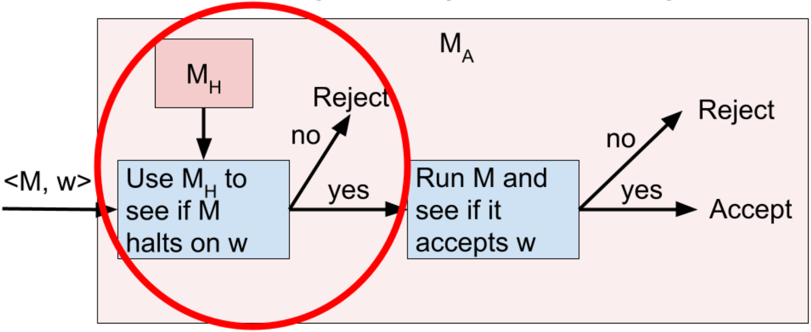
- $A \leq_T B$
- M_A can call M_B as a subroutine any number of times
- M_A can call M_B at any point in its computation

Mapping Reducibility

- $A \leq_M B$
- M_A can use M_B as a subroutine exactly once
- M_A can only call M_B at the very last step in the computation

Non-Mapping Reductions

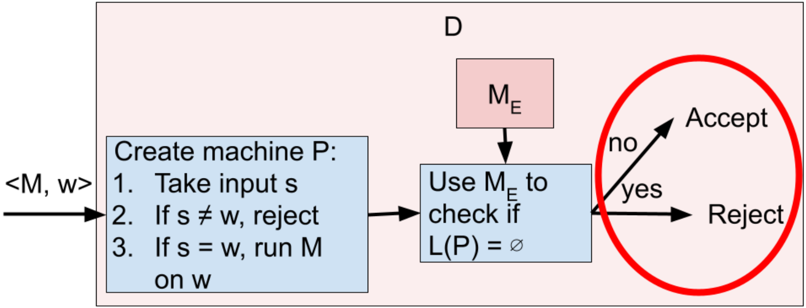
$$A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$$
$$HALT = \{\langle M, w\rangle \mid M \text{ halts on } w\}$$



M_H subroutine is used prior to the last step

Non-Mapping Reductions

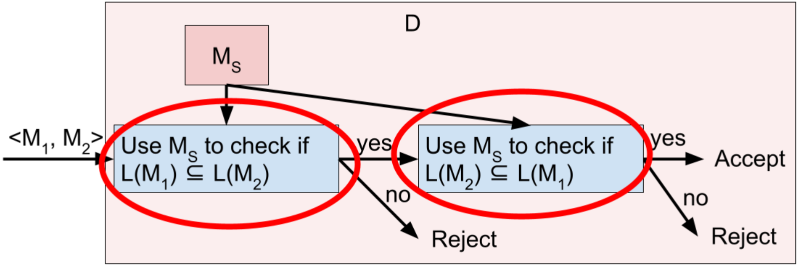
A_{TM} = {<M, w> | M accepts w}
E_{TM} = {<M> | L(M) = ∅}



"Yes maps to No"
"No maps to Yes"

Non-Mapping Reductions

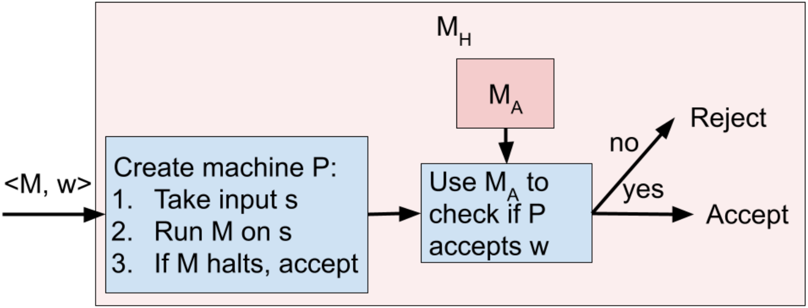
$$EQ_{TM} = \{<M_1, M_2> \mid L(M_1) = L(M_2)\}$$
$$SUB_{TM} = \{<M_1, M_2> \mid L(M_1) \subseteq L(M_2)\}$$



M_S subroutine is used more than once

HALT $\leq_M$ A_{TM}

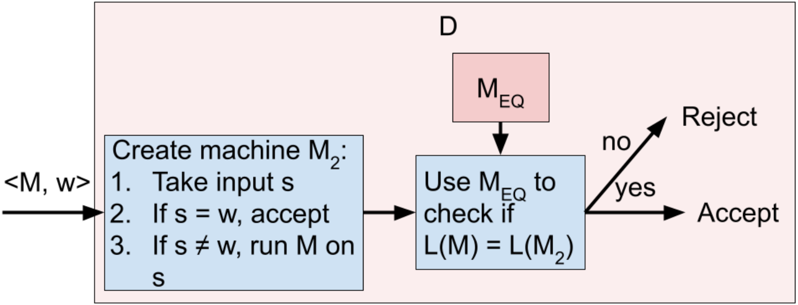
HALT = {<M, w> | M halts on w}
A_{TM} = {<M, w> | M accepts w}



Reduction: f(<M, w>) ↦ <P, w>
<M, w> ∈ HALT ⇔ f(<M, w>) ∈ A_{TM}

$A_{TM} \leq_M EQ_{TM}$

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$$
$$EQ_{TM} = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$$

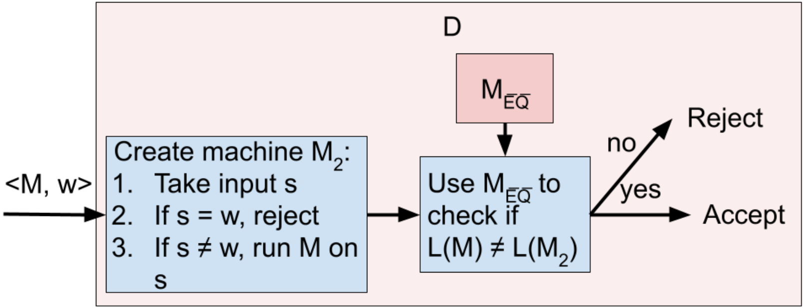


Reduction: $f(\langle M, w\rangle) \mapsto \langle M, M_2\rangle$

$\langle M, w\rangle \in A_{TM} \Leftrightarrow f(\langle M, w\rangle) \in EQ_{TM}$

$A_{TM} \leq_M \overline{EQ_{TM}}$

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$$
$$\overline{EQ}_{TM} = \{\langle M_1, M_2\rangle \mid L(M_1) \neq L(M_2)\}$$



Reduction: $f(\langle M, w\rangle) \mapsto \langle M, M_2\rangle$

$$\langle M, w\rangle \in A_{TM} \Leftrightarrow f(\langle M, w\rangle) \in \overline{EQ}_{TM}$$

Mapping Reducibility

Theorem: The following four statements are true:

Mapping Reducibility

Theorem: The following four statements are true:

1. If $A \leq_M B$ and B is decidable, then A is decidable

Mapping Reducibility

Theorem: The following four statements are true:

1. If $A \leq_M B$ and B is decidable, then A is decidable
2. If $A \leq_M B$ and B is recognizable, then A is recognizable

Mapping Reducibility

Theorem: The following four statements are true:

1. If $A \leq_M B$ and B is decidable, then A is decidable
2. If $A \leq_M B$ and B is recognizable, then A is recognizable
3. If $A \leq_M B$ and A is undecidable, then B is undecidable

Mapping Reducibility

Theorem: The following four statements are true:

1. If $A \leq_M B$ and B is decidable, then A is decidable
2. If $A \leq_M B$ and B is recognizable, then A is recognizable
3. If $A \leq_M B$ and A is undecidable, then B is undecidable
4. If $A \leq_M B$ and A is unrecognizable, then B is unrecognizable

Mapping Reducibility

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

Mapping Reducibility

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

- There is a computable function $f : \Sigma^* \to \Sigma^*$ such that $w \in A \Leftrightarrow f(w) \in B$

Mapping Reducibility

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

- There is a computable function $f : \Sigma^* \to \Sigma^*$ such that $w \in A \Leftrightarrow f(w) \in B$
- There is a machine M_B that decides B

Mapping Reducibility

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

- There is a computable function $f : \Sigma^* \to \Sigma^*$ such that $w \in A \Leftrightarrow f(w) \in B$
- There is a machine M_B that decides B
- Construct a machine M_A to decide A

Mapping Reducibility

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

- There is a computable function $f : \Sigma^* \to \Sigma^*$ such that $w \in A \Leftrightarrow f(w) \in B$
- There is a machine M_B that decides B
- Construct a machine M_A to decide A
    1. M_A takes w as input

Mapping Reducibility

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

- There is a computable function $f : \Sigma^* \to \Sigma^*$ such that $w \in A \Leftrightarrow f(w) \in B$
- There is a machine M_B that decides B
- Construct a machine M_A to decide A
  1. M_A takes w as input
  2. Compute $f(w)$

Mapping Reducibility

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

- There is a computable function $f : \Sigma^* \rightarrow \Sigma^*$ such that $w \in A \Leftrightarrow f(w) \in B$
- There is a machine M_B that decides B
- Construct a machine M_A to decide A
  1. M_A takes w as input
  2. Compute $f(w)$
  3. Run M_B on $f(w)$

Mapping Reducibility

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

- There is a computable function $f : \Sigma^* \rightarrow \Sigma^*$ such that $w \in A \Leftrightarrow f(w) \in B$
- There is a machine M_B that decides B
- Construct a machine M_A to decide A
  1. M_A takes w as input
  2. Compute $f(w)$
  3. Run M_B on $f(w)$
     3.1 If M_B accepts $f(w)$, then M_A accepts w

Mapping Reducibility

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

- There is a computable function $f : \Sigma^* \to \Sigma^*$ such that $w \in A \Leftrightarrow f(w) \in B$
- There is a machine M_B that decides B
- Construct a machine M_A to decide A
    1. M_A takes w as input
    2. Compute $f(w)$
    3. Run M_B on $f(w)$
        - 3.1 If M_B accepts $f(w)$, then M_A accepts w
        - 3.2 Otherwise M_A rejects w

Mapping Reducibility

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

- There is a computable function $f : \Sigma^* \to \Sigma^*$ such that $w \in A \Leftrightarrow f(w) \in B$
- There is a machine M_B that decides B
- Construct a machine M_A to decide A
  1. M_A takes w as input
  2. Compute $f(w)$
  3. Run M_B on $f(w)$
     - 3.1 If M_B accepts $f(w)$, then M_A accepts w
     - 3.2 Otherwise M_A rejects w
- M_A accepts $w \Leftrightarrow M_B$ accepts $f(w) \Leftrightarrow f(w) \in B \Leftrightarrow w \in A$

Mapping Reducibility

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

- There is a computable function $f : \Sigma^* \to \Sigma^*$ such that $w \in A \Leftrightarrow f(w) \in B$
- There is a machine M_B that decides B
- Construct a machine M_A to decide A
  1. M_A takes w as input
  2. Compute $f(w)$
  3. Run M_B on $f(w)$
     - 3.1 If M_B accepts $f(w)$, then M_A accepts w
     - 3.2 Otherwise M_A rejects w
- M_A accepts $w \Leftrightarrow M_B$ accepts $f(w) \Leftrightarrow f(w) \in B \Leftrightarrow w \in A$
- f is computable, and M_B is a decider, so M_A will always halt. Thus, M_A decides A

Mapping Reducibility

Theorem: If $A \leq_M B$ and B is recognizable, then A is recognizable

- Let M_B recognize B
- Let f be the reduction from A to B
- M_A recognizes A as follows:
  1. M_A takes input w
  2. Compute $f(w)$
  3. Run M_B on $f(w)$
     3.1 If M_B accepts $f(w)$, M_A accepts w
     3.2 If M_B does not accept $f(w)$, M_A will not accept w

Mapping Reducibility

Theorem: If $A \leq_M B$ and B is recognizable, then A is recognizable

- Let M_B recognize B
- Let f be the reduction from A to B
- M_A recognizes A as follows:
  1. M_A takes input w
  2. Compute $f(w)$
  3. Run M_B on $f(w)$
     3.1 If M_B accepts $f(w)$, M_A accepts w
     3.2 If M_B does not accept $f(w)$, M_A will not accept w
- M_A accepts $w \Leftrightarrow M_B$ accepts $f(w) \Leftrightarrow f(w) \in B \Leftrightarrow w \in A$

Mapping Reducibility

Theorem: If $A \leq_M B$ and B is recognizable, then A is recognizable

- Let M_B recognize B
- Let f be the reduction from A to B
- M_A recognizes A as follows:
  1. M_A takes input w
  2. Compute $f(w)$
  3. Run M_B on $f(w)$
     - 3.1 If M_B accepts $f(w)$, M_A accepts w
     - 3.2 If M_B does not accept $f(w)$, M_A will not accept w
- M_A accepts $w \Leftrightarrow M_B$ accepts $f(w) \Leftrightarrow f(w) \in B \Leftrightarrow w \in A$
- Thus M_A recognizes A

Mapping Reducibility

Theorem: If $A \leq_M B$ and A is undecidable then B is undecidable

Mapping Reducibility

Theorem: If $A \leq_M B$ and A is undecidable then B is undecidable

- AFSOC B is decidable

Mapping Reducibility

Theorem: If $A \leq_M B$ and A is undecidable then B is undecidable

- AFSOC B is decidable
- Then A is decidable

Mapping Reducibility

Theorem: If $A \leq_M B$ and A is undecidable then B is undecidable

- AFSOC B is decidable
- Then A is decidable
- But A is undecidable! This is a contradiction, and we conclude that B is not decidable.

Mapping Reducibility

Theorem: If $A \leq_M B$ and A is unrecognizable then B is unrecognizable

Mapping Reducibility

Theorem: If $A \leq_M B$ and A is unrecognizable then B is unrecognizable

- AFSOC B is recognizable

Mapping Reducibility

Theorem: If $A \leq_M B$ and A is unrecognizable then B is unrecognizable

- AFSOC B is recognizable
- Then A is recognizable

Mapping Reducibility

Theorem: If $A \leq_M B$ and A is unrecognizable then B is unrecognizable

- AFSOC B is recognizable
- Then A is recognizable
- But A is unrecognizable! This is a contradiction, and we conclude that B is not recognizable.

Mapping Reducibility

Theorem: If $A \leq_M B$ then $\overline{A} \leq_M \overline{B}$

Mapping Reducibility

Theorem: If $A \leq_M B$ then $\overline{A} \leq_M \overline{B}$

- There is a computable function f such that $w \in A \Leftrightarrow f(w) \in B$

Mapping Reducibility

Theorem: If $A \leq_M B$ then $\overline{A} \leq_M \overline{B}$

- There is a computable function f such that $w \in A \Leftrightarrow f(w) \in B$
- $w \notin A \Leftrightarrow f(w) \notin B$

Mapping Reducibility

Theorem: If $A \leq_M B$ then $\overline{A} \leq_M \overline{B}$

- There is a computable function f such that $w \in A \Leftrightarrow f(w) \in B$
- $w \notin A \Leftrightarrow f(w) \notin B$
  - $w \in \overline{A} \Leftrightarrow f(w) \in \overline{B}$

$\mathrm{EQ}_{\mathrm{TM}}$

Theorem: $\mathrm{EQ}_{\mathrm{TM}}$ is neither recognizable nor co-recognizable

$\mathrm{EQ}_{\mathrm{TM}}$ is not recognizable

$\mathrm{EQ}_{\mathrm{TM}}$ is not recognizable

- $\mathrm{A}_{\mathrm{TM}} \leq_M \overline{\mathrm{EQ}_{\mathrm{TM}}}$, therefore $\overline{\mathrm{A}_{\mathrm{TM}}} \leq_M \mathrm{EQ}_{\mathrm{TM}}$

$\mathrm{EQ}_{\mathrm{TM}}$ is not recognizable

- $\mathrm{A}_{\mathrm{TM}} \leq_M \overline{\mathrm{EQ}_{\mathrm{TM}}}$, therefore $\overline{\mathrm{A}_{\mathrm{TM}}} \leq_M \mathrm{EQ}_{\mathrm{TM}}$
- $\overline{\mathrm{A}_{\mathrm{TM}}}$ is not recognizable

$\mathrm{EQ}_{\mathrm{TM}}$ is not recognizable

- $\mathrm{A}_{\mathrm{TM}} \leq_M \overline{\mathrm{EQ}_{\mathrm{TM}}}$, therefore $\overline{\mathrm{A}_{\mathrm{TM}}} \leq_M \mathrm{EQ}_{\mathrm{TM}}$
- $\overline{\mathrm{A}_{\mathrm{TM}}}$ is not recognizable
- Therefore $\mathrm{EQ}_{\mathrm{TM}}$ is not recognizable

$\overline{\mathrm{EQ}_{\mathrm{TM}}}$ is not recognizable

$\overline{\mathrm{EQ}_{\mathrm{TM}}}$ is not recognizable

- $\mathrm{A}_{\mathrm{TM}} \leq_M \mathrm{EQ}_{\mathrm{TM}}$, therefore $\overline{\mathrm{A}_{\mathrm{TM}}} \leq_M \overline{\mathrm{EQ}_{\mathrm{TM}}}$

$\overline{\mathrm{EQ}_{\mathrm{TM}}}$ is not recognizable

- $\mathrm{A}_{\mathrm{TM}} \leq_M \mathrm{EQ}_{\mathrm{TM}}$, therefore $\overline{\mathrm{A}_{\mathrm{TM}}} \leq_M \overline{\mathrm{EQ}_{\mathrm{TM}}}$
- $\overline{\mathrm{A}_{\mathrm{TM}}}$ is not recognizable

$\overline{\mathrm{EQ}_{\mathrm{TM}}}$ is not recognizable

- $\mathrm{A}_{\mathrm{TM}} \leq_M \mathrm{EQ}_{\mathrm{TM}}$, therefore $\overline{\mathrm{A}_{\mathrm{TM}}} \leq_M \overline{\mathrm{EQ}_{\mathrm{TM}}}$
- $\overline{\mathrm{A}_{\mathrm{TM}}}$ is not recognizable
- Therefore $\overline{\mathrm{EQ}_{\mathrm{TM}}}$ is not recognizable