Mapping Reductions

Definition

A function $f : \Sigma^* \to \Sigma^*$ is computable if there is some Turing Machine M that on every input w halts with $f(w)$ on the tape.

Mapping Reductions

Definition

A function $f : \Sigma^* \to \Sigma^*$ is computable if there is some Turing Machine M that on every input w halts with $f(w)$ on the tape.

Definition

A mapping/many-one reduction from A to B is a computable function $f : \Sigma^* \to \Sigma^*$ such that

$$w \in A \text{ if and only if } f(w) \in B$$

Mapping Reductions

Definition

A function $f : \Sigma^* \to \Sigma^*$ is computable if there is some Turing Machine M that on every input w halts with $f(w)$ on the tape.

Definition

A mapping/many-one reduction from A to B is a computable function $f : \Sigma^* \to \Sigma^*$ such that

$$w \in A \text{ if and only if } f(w) \in B$$

In this case, we say A is mapping/many-one reducible to B, and we denote it by $A \leq_m B$.

Mapping Reductions

Definition

A function $f : \Sigma^* \to \Sigma^*$ is computable if there is some Turing Machine M that on every input w halts with $f(w)$ on the tape.

Definition

A reduction (a.k.a. mapping reduction/many-one reduction) from a language A to a language B is a computable function $f : \Sigma^* \to \Sigma^*$ such that

$$w \in A \text{ if and only if } f(w) \in B$$

In this case, we say A is reducible to B, and we denote it by $A \leq_m B$.

Reductions and Recursive Enumerability

Proposition

If $A \leq_m B$ and B is r.e., then A is r.e.

Proof.

Let f be a reduction from A to B and let M_B be a Turing Machine recognizing B. Then the Turing machine recognizing A is

```
On input w
    Compute f(w)
    Run M_B on f(w)
    Accept if M_B accepts, and reject if M_B rejects
```

□

Corollary

If $A \leq_m B$ and A is not r.e., then B is not r.e.

Reductions and Decidability

Proposition

If $A \leq_m B$ and B is decidable, then A is decidable.

Proof.

Let f be a reduction from A to B and let M_B be a Turing Machine *deciding* B. Then a Turing machine that decides A is

```
On input w
    Compute f(w)
    Run M_B on f(w)
    Accept if M_B accepts, and reject if M_B rejects
```

□

Corollary

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

The Halting Problem

Proposition

The language $HALT = \{\langle M, w\rangle \mid M \text{ halts on input } w\}$ *is undecidable.*

Proof.

Recall $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid w \in L(M)\}$ is undecidable. Will give reduction f to show $A_{\mathrm{TM}} \leq_m$ HALT $\implies$ HALT undecidable.

Let $f(\langle M, w\rangle) = \langle N, w\rangle$ where N is a TM that behaves as follows:

```
On input x
    Run M on x
    If M accepts then halt and accept
    If M rejects then go into an infinite loop
```

N halts on input w if and only if M accepts w.

The Halting Problem

Proposition

The language HALT $= \{\langle M, w\rangle \mid M$ *halts on input* $w\}$ *is undecidable.*

Proof.

Recall $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid w \in L(M)\}$ is undecidable. Will give reduction f to show $A_{\mathrm{TM}} \leq_m$ HALT $\implies$ HALT undecidable.
Let $f(\langle M, w\rangle) = \langle N, w\rangle$ where N is a TM that behaves as follows:

```
On input x
    Run M on x
    If M accepts then halt and accept
    If M rejects then go into an infinite loop
```

N halts on input w if and only if M accepts w. i.e., $\langle M, w\rangle \in A_{\mathrm{TM}}$ iff $f(\langle M, w\rangle) \in$ HALT $\square$

Emptiness of Turing Machines

Proposition

The language $E_{\mathrm{TM}} = \{M \mid L(M) = \emptyset\}$ *is not decidable.*

Note: in fact, E_{TM} is not recognizable.

Emptiness of Turing Machines

Proposition

The language $E_{\mathrm{TM}} = \{M \mid L(M) = \emptyset\}$ *is not decidable.*

Note: in fact, E_{TM} is not recognizable.

Proof.

Recall $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid w \in L(M)\}$ is undecidable.

Emptiness of Turing Machines

Proposition

The language $E_{\mathrm{TM}} = \{M \mid L(M) = \emptyset\}$ *is not decidable.*

Note: in fact, E_{TM} is not recognizable.

Proof.

Recall $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid w \in L(M)\}$ is undecidable. For the sake of contradiction, suppose there is a decider B for E_{TM}.

Emptiness of Turing Machines

Proposition

The language $E_{\mathrm{TM}} = \{M \mid L(M) = \emptyset\}$ *is not decidable.*

Note: in fact, E_{TM} is not recognizable.

Proof.

Recall $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid w \in L(M)\}$ is undecidable. For the sake of contradiction, suppose there is a decider B for E_{TM}. Then we first transform $\langle M, w\rangle$ to $\langle M_1\rangle$ which is the following:

```
On input x
    If x ≠ w, reject
    else run M on w , and accept if M accepts w
```

, and accept if B rejects $\langle M_1\rangle$, and rejects if B accepts $\langle M_1\rangle$.

Emptiness of Turing Machines

Proposition

The language $E_{\mathrm{TM}} = \{M \mid L(M) = \emptyset\}$ *is not decidable.*

Note: in fact, E_{TM} is not recognizable.

Proof.

Recall $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid w \in L(M)\}$ is undecidable. For the sake of contradiction, suppose there is a decider B for E_{TM}. Then we first transform $\langle M, w\rangle$ to $\langle M_1\rangle$ which is the following:

```
On input x
    If x ≠ w, reject
    else run M on w , and accept if M accepts w
```

, and accept if B rejects $\langle M_1\rangle$, and rejects if B accepts $\langle M_1\rangle$.

Then we show that (1) if $\langle M, w\rangle \in A_{\mathrm{TM}}$, then accept, and (2) $\langle M, w\rangle \in A_{\mathrm{TM}}$, then reject. (how?)

Emptiness of Turing Machines

Proposition

The language $E_{\mathrm{TM}} = \{M \mid L(M) = \emptyset\}$ *is not decidable.*

Note: in fact, E_{TM} is not recognizable.

Proof.

Recall $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid w \in L(M)\}$ is undecidable. For the sake of contradiction, suppose there is a decider B for E_{TM}. Then we first transform $\langle M, w\rangle$ to $\langle M_1\rangle$ which is the following:

```
On input x
    If x ≠ w, reject
    else run M on w , and accept if M accepts w
```

, and accept if B rejects $\langle M_1\rangle$, and rejects if B accepts $\langle M_1\rangle$.

Then we show that (1) if $\langle M, w\rangle \in A_{\mathrm{TM}}$, then accept, and (2) $\langle M, w\rangle \in A_{\mathrm{TM}}$, then reject. (how?) This implies A_{TM} is decidable, which is a contradiction. □

Checking Regularity

Proposition

The language $REGULAR = \{M \mid L(M) \text{ is regular}\}$ *is undecidable.*

Checking Regularity

Proposition

The language $REGULAR = \{M \mid L(M) \text{ is regular}\}$ *is undecidable.*

Proof.

We give a reduction f from A_{TM} to REGULAR.

Checking Regularity

Proposition

The language $REGULAR = \{M \mid L(M) \text{ is regular}\}$ is undecidable.

Proof.

We give a reduction f from A_{TM} to REGULAR. Let $f(\langle M, w\rangle) = N$, where N is a TM that works as follows:

```
On input x
    If x is of the form 0^n1^n then accept x
    else run M on w and accept x only if M does
```

Checking Regularity

Proposition

The language REGULAR $= \{M \mid L(M)$ *is regular*$\}$ *is undecidable.*

Proof.

We give a reduction f from A_{TM} to REGULAR. Let $f(\langle M, w\rangle) = N$, where N is a TM that works as follows:

```
On input x
    If x is of the form 0^n1^n then accept x
    else run M on w and accept x only if M does
```

If $w \in L(M)$ then $L(N) =$

Checking Regularity

Proposition

The language $REGULAR = \{M \mid L(M) \text{ is regular}\}$ *is undecidable.*

Proof.

We give a reduction f from A_{TM} to REGULAR. Let $f(\langle M, w\rangle) = N$, where N is a TM that works as follows:

```
On input x
    If x is of the form 0^n1^n then accept x
    else run M on w and accept x only if M does
```

If $w \in L(M)$ then $L(N) = \Sigma^*$.

Checking Regularity

Proposition

The language $REGULAR = \{M \mid L(M)$ *is regular*$\}$ *is undecidable.*

Proof.

We give a reduction f from A_{TM} to REGULAR. Let $f(\langle M, w \rangle) = N$, where N is a TM that works as follows:

```
On input x
    If x is of the form 0^n1^n then accept x
    else run M on w and accept x only if M does
```

If $w \in L(M)$ then $L(N) = \Sigma^*$. If $w \notin L(M)$ then $L(N) =$

Checking Regularity

Proposition

The language $REGULAR = \{M \mid L(M) \text{ is regular}\}$ *is undecidable.*

Proof.

We give a reduction f from $A_{\rm TM}$ to REGULAR. Let $f(\langle M, w\rangle) = N$, where N is a TM that works as follows:

```
On input x
    If x is of the form 0^n1^n then accept x
    else run M on w and accept x only if M does
```

If $w \in L(M)$ then $L(N) = \Sigma^*$. If $w \notin L(M)$ then $L(N) = \{0^n1^n \mid n \geq 0\}$.

Checking Regularity

Proposition

The language $REGULAR = \{M \mid L(M) \text{ is regular}\}$ is undecidable.

Proof.

We give a reduction f from A_{TM} to REGULAR. Let $f(\langle M, w\rangle) = N$, where N is a TM that works as follows:

```
On input x
    If x is of the form 0^n1^n then accept x
    else run M on w and accept x only if M does
```

If $w \in L(M)$ then $L(N) = \Sigma^*$. If $w \notin L(M)$ then $L(N) = \{0^n1^n \mid n \geq 0\}$. Thus, $\langle N\rangle \in$ REGULAR if and only if $\langle M, w\rangle \in A_{\mathrm{TM}}$ □

Checking Equality

Proposition

$EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$ *is not r.e.*

Checking Equality

Proposition

$EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$ *is not r.e.*

Proof.

We will give a reduction f from E_{TM} (assume that we know E_{TM} is R.E.) to $\mathrm{EQ}_{\mathrm{TM}}$.

Checking Equality

Proposition

$EQ_{\mathrm{TM}} = \{\langle M_1, M_2 \rangle \mid L(M_1) = L(M_2)\}$ *is not r.e.*

Proof.

We will give a reduction f from E_{TM} (assume that we know E_{TM} is R.E.) to $\mathrm{EQ}_{\mathrm{TM}}$. Let M_1 be the Turing machine that on any input, halts and rejects

Checking Equality

Proposition

$EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$ *is not r.e.*

Proof.

We will give a reduction f from E_{TM} (assume that we know E_{TM} is R.E.) to $\mathrm{EQ}_{\mathrm{TM}}$. Let M_1 be the Turing machine that on any input, halts and rejects i.e., $L(M_1) = \emptyset$. Take $f(M) = \langle M, M_1\rangle$.

Checking Equality

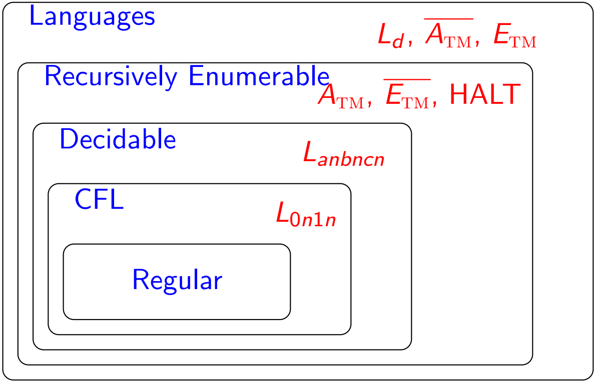
Proposition

$EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$ *is not r.e.*

Proof.

We will give a reduction f from E_{TM} (assume that we know E_{TM} is R.E.) to $\mathrm{EQ}_{\mathrm{TM}}$. Let M_1 be the Turing machine that on any input, halts and rejects i.e., $L(M_1) = \emptyset$. Take $f(M) = \langle M, M_1\rangle$. Observe $M \in E_{\mathrm{TM}}$ iff $L(M) = \emptyset$ iff $L(M) = L(M_1)$ iff $\langle M, M_1\rangle \in \mathrm{EQ}_{\mathrm{TM}}$. □

Big Picture . . . again



Big Picture ... again