CS 383

Lecture 22 – Mapping reductions

Stephen Checkoway

Fall, 2023

Review of decidable languages

- Context-free languages (and thus regular)

Review of decidable languages

- Context-free languages (and thus regular)
- Acceptance problems
  - A_{DFA}
  - A_{NFA}
  - A_{REX}
  - A_{CFG}

Review of decidable languages

- Context-free languages (and thus regular)
- Acceptance problems
  - A_{DFA}
  - A_{NFA}
  - A_{REX}
  - A_{CFG}
- Emptiness problems
  - E_{DFA}
  - E_{CFG}

Review of decidable languages

- Context-free languages (and thus regular)
- Acceptance problems
  - A_{DFA}
  - A_{NFA}
  - A_{REX}
  - A_{CFG}
- Emptiness problems
  - E_{DFA}
  - E_{CFG}
- Equivalence problems
  - EQ_{DFA}

Review of undecidable languages

- The diagonal language $\textsc{Diag} = \{\langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin L(M)\}$

Review of undecidable languages

- The diagonal language $\textsc{Diag} = \{\langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin L(M)\}$
- A_{TM}

Review of undecidable languages

- The diagonal language $\textsc{Diag} = \{\langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin L(M)\}$
- $A_{\textsf{TM}}$
- $\textsc{Halt}_{\textsf{TM}}$

Review of undecidable languages

- The diagonal language $\textsc{Diag} = \{\langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin L(M)\}$
- A_{TM}
- $\textsc{Halt}_{\mathsf{TM}}$
- E_{TM}

Review of undecidable languages

- The diagonal language $\textsc{Diag} = \{\langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin L(M)\}$
- A_{TM}
- $\textsc{Halt}_{\mathsf{TM}}$
- E_{TM}
- ALL_{CFG}

Review of undecidable languages

- The diagonal language $\textsc{Diag} = \{\langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin L(M)\}$
- A_{TM}
- $\textsc{Halt}_{\mathsf{TM}}$
- E_{TM}
- ALL_{CFG}
- EQ_{CFG}

Review of undecidable languages

- The diagonal language $\textsc{Diag} = \{\langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin L(M)\}$
- A_{TM}
- $\textsc{Halt}_{\mathsf{TM}}$
- E_{TM}
- ALL_{CFG}
- EQ_{CFG}
- EQ_{TM}

Review of undecidable languages

- The diagonal language $\textsc{Diag} = \{\langle M \rangle \mid M$ is a TM and $\langle M \rangle \notin L(M)\}$
- A_{TM}
- $\textsc{Halt}_{\mathsf{TM}}$
- E_{TM}
- ALL_{CFG}
- EQ_{CFG}
- EQ_{TM}
- $\textsc{Regular}_{\mathsf{TM}}$

Turing recognizable (RE) and co-Turing-recognizable (coRE)

Recall, L is decidable iff L is RE and coRE

Language	RE	coRE
A_{DFA}	✔	✔
E_{DFA}	✔	✔
EQ_{DFA}	✔	✔
A_{CFG}	✔	✔
E_{CFG}	✔	✔
EQ_{CFG}	✖	✔
$\textsc{Diag}$	?	?
A_{TM}	✔	✖
$\textsc{Halt}_{\text{TM}}$	?	?
E_{TM}	✖	✔
EQ_{TM}	?	?
$\textsc{Regular}_{\text{TM}}$	?	?

Reductions

Recall that A reduces to B (written $A \leq B$) means
"If B is decidable, then A is decidable"

Reductions

Recall that A reduces to B (written $A \leq B$) means
"If B is decidable, then A is decidable"

We used reductions to

1. prove that languages are decidable ("good-news reductions")
2. prove that languages are not decidable ("bad-news reductions")

Reductions

Recall that A reduces to B (written $A \leq B$) means
"If B is decidable, then A is decidable"

We used reductions to

1. prove that languages are decidable ("good-news reductions")
2. prove that languages are not decidable ("bad-news reductions")

We were able to determine that some languages aren't RE by showing that they're coRE but not decidable

Similarly, we proved some languages aren't coRE by showing that they're RE but not decidable

Reductions

Recall that A reduces to B (written $A \leq B$) means
"If B is decidable, then A is decidable"

We used reductions to

1. prove that languages are decidable ("good-news reductions")
2. prove that languages are not decidable ("bad-news reductions")

We were able to determine that some languages aren't RE by showing that they're coRE but not decidable

Similarly, we proved some languages aren't coRE by showing that they're RE but not decidable

Reductions alone were not sufficient; we need a *stronger* notion of reduction

Computable functions

A function $f : \Sigma^* \to \Sigma^*$ is a computable function if there is some TM M such that when M is run on w, M halts with $f(w)$ on the tape (and nothing else)

This is similar to a decider in that M cannot loop, but there's no notion of accepting or rejecting a string, M just computes a function

Examples of computable functions

- Arithmetic: $\langle k, m, n \rangle \mapsto \langle k \cdot m - 67n \rangle$ where $k, m, n \in \mathbb{Z}$
  The corresponding TM performs the arithmetic and then copies the result to the beginning of the tape and clears the rest

Examples of computable functions

- Arithmetic: $\langle k, m, n\rangle \mapsto \langle k \cdot m - 67n\rangle$ where $k, m, n \in \mathbb{Z}$
  The corresponding TM performs the arithmetic and then copies the result to the beginning of the tape and clears the rest
- Converting a grammar to CNF: $\langle G\rangle \mapsto \langle G'\rangle$ where $L(G) = L(G')$ and G' is in CNF
  The corresponding TM performs the conversion to CNF algorithm

Examples of computable functions

- Arithmetic: $\langle k, m, n \rangle \mapsto \langle k \cdot m - 67n \rangle$ where $k, m, n \in \mathbb{Z}$
  The corresponding TM performs the arithmetic and then copies the result to the beginning of the tape and clears the rest
- Converting a grammar to CNF: $\langle G \rangle \mapsto \langle G' \rangle$ where $L(G) = L(G')$ and G' is in CNF
  The corresponding TM performs the conversion to CNF algorithm
- Constructing new TMs: $\langle M, w \rangle \mapsto \langle M' \rangle$ where M' is the TM that ignores its input and runs M on w

Examples of computable functions

- Arithmetic: $\langle k, m, n\rangle \mapsto \langle k \cdot m - 67n\rangle$ where $k, m, n \in \mathbb{Z}$
  The corresponding TM performs the arithmetic and then copies the result to the beginning of the tape and clears the rest
- Converting a grammar to CNF: $\langle G\rangle \mapsto \langle G'\rangle$ where $L(G) = L(G')$ and G' is in CNF
  The corresponding TM performs the conversion to CNF algorithm
- Constructing new TMs: $\langle M, w\rangle \mapsto \langle M'\rangle$ where M' is the TM that ignores its input and runs M on w
- Constructing multiple TMs: $\langle M\rangle \mapsto \langle M, M'\rangle$ where M' is a TM such that $L(M') = \Sigma^*$

Examples of computable functions

- Arithmetic: $\langle k, m, n \rangle \mapsto \langle k \cdot m - 67n \rangle$ where $k, m, n \in \mathbb{Z}$
  The corresponding TM performs the arithmetic and then copies the result to the beginning of the tape and clears the rest
- Converting a grammar to CNF: $\langle G \rangle \mapsto \langle G' \rangle$ where $L(G) = L(G')$ and G' is in CNF
  The corresponding TM performs the conversion to CNF algorithm
- Constructing new TMs: $\langle M, w \rangle \mapsto \langle M' \rangle$ where M' is the TM that ignores its input and runs M on w
- Constructing multiple TMs: $\langle M \rangle \mapsto \langle M, M' \rangle$ where M' is a TM such that $L(M') = \Sigma^*$

Anything that a TM can do without looping, including running deciders, is permissible

Examples of computable functions

- Arithmetic: $\langle k, m, n \rangle \mapsto \langle k \cdot m - 67n \rangle$ where $k, m, n \in \mathbb{Z}$
  The corresponding TM performs the arithmetic and then copies the result to the beginning of the tape and clears the rest
- Converting a grammar to CNF: $\langle G \rangle \mapsto \langle G' \rangle$ where $L(G) = L(G')$ and G' is in CNF
  The corresponding TM performs the conversion to CNF algorithm
- Constructing new TMs: $\langle M, w \rangle \mapsto \langle M' \rangle$ where M' is the TM that ignores its input and runs M on w
- Constructing multiple TMs: $\langle M \rangle \mapsto \langle M, M' \rangle$ where M' is a TM such that $L(M') = \Sigma^*$

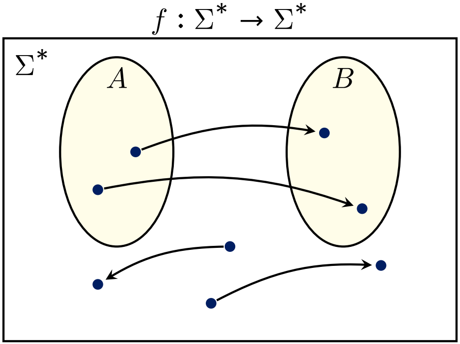
Anything that a TM can do without looping, including running deciders, is permissible

If the form of the input is wrong (e.g., if the TM is expecting $\langle M, w \rangle$ but gets something else), then it clears the tape and halts (i.e., outputs ε)

Mapping reducibility

Language A is mapping reducible to language B, written $A \leq_m B$, if there exists a computable function $f : \Sigma^* \to \Sigma^*$ such that for each $w \in \Sigma^*$,

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



f maps elements of A to elements of B
f maps elements of $\overline{A}$ to elements of $\overline{B}$

Mapping instances of problems to instances of other problems

Consider the problems

1. Is the string w recognized by the PDA P?
2. Is the string x generated by the CFG G?

We express both of these as languages, A_{PDA} and A_{CFG}, respectively

An instance of the first problem is the (representation of the) pair $\langle P, w\rangle$ and an instance of the second problem is $\langle G, x\rangle$

A mapping reduction $A \leq_{\mathrm{m}} B$ takes an instance of problem A and maps it to an instance of problem B such that the solution to the latter gives the solution to the former

E.g., $\langle P, w\rangle \mapsto \langle G, w\rangle$ where $L(G) = L(P)$ is a computable mapping and $\langle P, w\rangle \in A_{\mathsf{PDA}} \iff \langle G, w\rangle \in A_{\mathsf{CFG}}$ so $A_{\mathsf{PDA}} \leq_{\mathrm{m}} A_{\mathsf{CFG}}$

Question 1

Is $A_{\mathsf{CFG}} \leq_{\mathrm{m}} A_{\mathsf{PDA}}$?

Question 1

Is $A_{\mathsf{CFG}} \leq_{\mathrm{m}} A_{\mathsf{PDA}}$?

Yes. The mapping $\langle G, w \rangle \mapsto \langle P, w \rangle$ where $L(P) = L(G)$ is computable because the CFG to PDA conversion is a simple algorithm.

As before, $\langle G, w \rangle \in A_{\mathsf{CFG}} \iff \langle P, w \rangle \in A_{\mathsf{PDA}}$

Question 2

Is $A_{\mathsf{DFA}} \leq_{\mathrm{m}} A_{\mathsf{CFG}}$?

Question 2

Is $A_{\mathsf{DFA}} \leq_{\mathrm{m}} A_{\mathsf{CFG}}$?

Yes. We can convert a DFA to an equivalent CFG; i.e., $\langle M, w\rangle \mapsto \langle G, w\rangle$ where $L(G) = L(M)$ is computable and clearly $\langle M, w\rangle \in A_{\mathsf{DFA}} \iff \langle G, w\rangle \in A_{\mathsf{CFG}}$

Question 3

Is $A_{\mathsf{CFG}} \leq_{\mathrm{m}} A_{\mathsf{DFA}}$?

Question 3

Is $A_{\text{CFG}} \leq_{\mathrm{m}} A_{\text{DFA}}$?

Perhaps counterintuitively, yes!

Remember, A_{CFG} is decidable so we can use the decider R for it when constructing our mapping
T = "On input $\langle G, w \rangle$,

1. Run R on $\langle G, w \rangle$
2. If R accepts, let M be the 1-state DFA such that $L(M) = \Sigma^*$
3. If R rejects, let M be the 1-state DFA such that $L(M) = \varnothing$
4. Output $\langle M, \varepsilon \rangle$"

This won't loop because R is a decider.

If $\langle G, w \rangle \in A_{\text{CFG}}$, then $L(M) = \Sigma^*$ so $\langle M, \varepsilon \rangle \in A_{\text{DFA}}$

If $\langle G, w \rangle \notin A_{\text{CFG}}$, then $L(M) = \varnothing$ so $\langle M, \varepsilon \rangle \notin A_{\text{DFA}}$

Mapping reductions are a stronger form of reduction

What we've called a reduction up until now is also called a Turing reduction

Theorem

If $A \leq_{\mathrm{m}} B$*, then* $A \leq B$*. In other words, if* $A \leq_{\mathrm{m}} B$ *and* B *is decidable, then* A *is decidable*

How can we prove this?

Mapping reductions are a stronger form of reduction

What we've called a reduction up until now is also called a Turing reduction

Theorem

If $A \le_{\mathrm{m}} B$, then $A \le B$. In other words, if $A \le_{\mathrm{m}} B$ and B is decidable, then A is decidable

How can we prove this?

Proof.

Let R be a decider for B and let $f : \Sigma^* \to \Sigma^*$ be the mapping reduction.

D = "On input w,

1. Compute $f(w)$
2. Run R on $f(w)$ and if R accepts, then *accept*; otherwise *reject*"

f is computable and R is a decider so D is a decider.

If $w \in A$, then $f(w) \in B$ so R and thus D will accept

If $w \notin A$, then $f(w) \notin B$ so R and thus D will reject □

Using mapping reductions to show languages are undecidable

Just like with Turing reductions, we have a simple corollary:

Theorem

If $A \leq_{\mathrm{m}} B$ and A is undecidable, then B is undecidable

We typically use this fact by giving a TM that computes the mapping reduction

T = "On input $\langle$an instance of problem $A\rangle$,

1. Construct an instance of problem B
2. Output $\langle$the instance of problem $B\rangle$"

Rather than accept or reject, the TM T corresponding to the mapping outputs the result

Example: $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$

Show that $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$ by giving a TM T that computes the mapping
How do we do this?

Example: $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$

Show that $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$ by giving a TM T that computes the mapping
How do we do this?

T = "On input $\langle M \rangle$,

Example: $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$

Show that $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$ by giving a TM T that computes the mapping
How do we do this?

T = "On input $\langle M \rangle$,

1. Build TM M' such that $L(M') = \varnothing$

Example: $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$

Show that $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$ by giving a TM T that computes the mapping
How do we do this?

T = "On input $\langle M \rangle$,

1. Build TM M' such that $L(M') = \varnothing$
2. Output $\langle M, M' \rangle$"

Example: $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$

Show that $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$ by giving a TM T that computes the mapping
How do we do this?

T = "On input $\langle M \rangle$,

1. Build TM M' such that $L(M') = \varnothing$
2. Output $\langle M, M' \rangle$"

Note that $\langle M \rangle$ is an instance of E_{TM} and $\langle M, M' \rangle$ is an instance of EQ_{TM}

Example: $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$

Show that $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$ by giving a TM T that computes the mapping
How do we do this?

T = "On input $\langle M \rangle$,

1. Build TM M' such that $L(M') = \emptyset$
2. Output $\langle M, M' \rangle$"

Note that $\langle M \rangle$ is an instance of E_{TM} and $\langle M, M' \rangle$ is an instance of EQ_{TM}

We need to show that T doesn't loop and that $\langle M \rangle \in E_{\mathsf{TM}}$ iff $\langle M, M' \rangle \in EQ_{\mathsf{TM}}$

Example: $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$

Show that $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$ by giving a TM T that computes the mapping
How do we do this?

T = "On input $\langle M \rangle$,

1. Build TM M' such that $L(M') = \varnothing$
2. Output $\langle M, M' \rangle$"

Note that $\langle M \rangle$ is an instance of E_{TM} and $\langle M, M' \rangle$ is an instance of EQ_{TM}

We need to show that T doesn't loop and that $\langle M \rangle \in E_{\mathsf{TM}}$ iff $\langle M, M' \rangle \in EQ_{\mathsf{TM}}$

Neither steps 1 nor 2 loop, so T doesn't loop

Next, we have a chain of iff

$$\langle M \rangle \in E_{\mathsf{TM}} \iff L(M) = \varnothing \iff L(M) = L(M') \iff \langle M, M' \rangle \in EQ_{\mathsf{TM}}$$

Example: $A_{\mathsf{TM}} \leq_{\mathrm{m}} \textsc{Halt}_{\mathsf{TM}}$

This one is more tricky: Given $\langle M, w \rangle$ (an instance of A_{TM}), we need to construct $\langle M', w \rangle$ such that M accepts w iff M' halts on w

How can we do this?

Example: $A_{\mathsf{TM}} \leq_{\mathrm{m}} \textsc{Halt}_{\mathsf{TM}}$

This one is more tricky: Given $\langle M, w\rangle$ (an instance of A_{TM}), we need to construct $\langle M', w\rangle$ such that M accepts w iff M' halts on w

How can we do this?

T = "On input $\langle M, w\rangle$,

1. Construct a new TM M' = 'On input x,
   1. Run M on x
   2. If M accepts, then *accept*
   3. If M rejects, then *loop*'
2. Output $\langle M', w\rangle$"

Example: $A_{\mathsf{TM}} \leq_{\mathrm{m}} \textsc{Halt}_{\mathsf{TM}}$

This one is more tricky: Given $\langle M, w\rangle$ (an instance of A_{TM}), we need to construct $\langle M', w\rangle$ such that M accepts w iff M' halts on w

How can we do this?

T = "On input $\langle M, w\rangle$,

1. Construct a new TM M' = 'On input x,
   1. Run M on x
   2. If M accepts, then *accept*
   3. If M rejects, then *loop*'
2. Output $\langle M', w\rangle$"

Constructing the TM M' can't loop so T can't loop

If $\langle M, w\rangle \in A_{\mathsf{TM}}$, then M accepts w so M' accepts and thus halts on w so $\langle M', w\rangle \in \textsc{Halt}_{\mathsf{TM}}$

If $\langle M, w\rangle \notin A_{\mathsf{TM}}$, then either M rejects or loops on w and in either case, M' loops on w [why?] so $\langle M', w\rangle \notin \textsc{Halt}_{\mathsf{TM}}$

Example: $EQ_{\mathsf{CFG}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$

How do we show this?

Example: $EQ_{\mathsf{CFG}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$

How do we show this?

T = "On input $\langle G_1, G_2 \rangle$,

1. Construct TM M_1 s.t. $L(M_1) = L(G_1)$ (we can use the decider for A_{CFG} to do this)
2. Construct TM M_2 s.t. $L(M_2) = L(G_2)$
3. Output $\langle M_1, M_2 \rangle$"

Now what?

Example: $EQ_{\mathsf{CFG}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$

How do we show this?

T = "On input $\langle G_1, G_2 \rangle$,

1. Construct TM M_1 s.t. $L(M_1) = L(G_1)$ (we can use the decider for A_{CFG} to do this)
2. Construct TM M_2 s.t. $L(M_2) = L(G_2)$
3. Output $\langle M_1, M_2 \rangle$"

Now what?

T can't loop because it's just constructing two TMs

Since $L(G_i) = L(M_i)$, $\langle G_1, G_2 \rangle \in EQ_{\mathsf{CFG}} \iff L(G_1) = L(G_2) \iff L(M_1) = L(M_2) \iff \langle M_1, M_2 \rangle \in EQ_{\mathsf{TM}}$

Mapping reductions between RE languages

Theorem
If $A \leq_{\mathrm{m}} B$ *and* B *is Turing-recognizable, then* A *is Turing-recognizable.*

How do we prove this?

Mapping reductions between RE languages

Theorem

If $A \leq_{\mathrm{m}} B$ and B is Turing-recognizable, then A is Turing-recognizable.

How do we prove this? Same construction as for the decidable case.

Proof.

Let R be a TM such that $L(R) = B$ and $f : \Sigma^* \to \Sigma^*$ be the computable mapping.
Build TM M to recognize A:
M = "On input w,

1. Run R on $f(w)$. If R accepts, then *accept*; if R rejects, then *reject*"

Now we just need to show that $L(M) = A$

$$w \in A \iff f(w) \in B \iff R \text{ accepts } f(w) \iff M \text{ accepts } w. \qquad \square$$

Proving that a language is not RE

Theorem

If $A \leq_{\mathrm{m}} B$ and A is not Turing-recognizable, then B is not Turing-recognizable

Why?

Proving that a language is not RE

Theorem

If $A \leq_{\mathrm{m}} B$ *and* A *is not Turing-recognizable, then* B *is not Turing-recognizable*

Why?

Proof.

If B were RE, then by the previous theorem, A would be RE. □

Mapping reduction between complements

Theorem

If $A \leq_{\mathrm{m}} B$, then $\overline{A} \leq_{\mathrm{m}} \overline{B}$ with the reduction given by the same mapping.

We just use the fact that if f is the computable mapping, then $w \in A \iff f(w) \in B$

Mapping reduction between complements

Theorem

If $A \leq_{\mathrm{m}} B$, then $\overline{A} \leq_{\mathrm{m}} \overline{B}$ with the reduction given by the same mapping.

We just use the fact that if f is the computable mapping, then $w \in A \iff f(w) \in B$

Proof.

Let f be the mapping reduction from A to B. Then

$$w \in \overline{A} \iff w \notin A \iff f(w) \notin B \iff f(w) \in \overline{B}. \qquad \square$$

coRE

Theorem

If $A \leq_{\mathrm{m}} B$ and B is co-Turing-recognizable, then A is co-Turing-recognizable.

Why?

coRE

Theorem

If $A \leq_{\mathrm{m}} B$ and B is co-Turing-recognizable, then A is co-Turing-recognizable.

Why?

Proof.

By the previous theorem, $\overline{A} \leq_{\mathrm{m}} \overline{B}$.

Since B is coRE, $\overline{B}$ is RE and thus $\overline{A}$ is RE. Therefore, A is coRE. □

Not coRE

Theorem

If $A \leq_{\mathrm{m}} B$ and A is not co-Turing-recognizable, then B is not co-Turing-recognizable.

Proof.

If B were coRE, then A would be coRE by the previous theorem. □

Recapitulate our results

A and B are languages and $A \leq_{\mathrm{m}} B$.

Good-news reductions

- If B is decidable, then A is decidable
- If B is RE, then A is RE
- If B is coRE, then A is coRE

Bad-news reductions

- If A is not decidable, then B is not decidable
- If A is not RE, then B is not RE
- If A is not coRE, then B is not coRE

Example

Show $A_{\mathsf{TM}} \leq_{\mathrm{m}} \overline{E_{\mathsf{TM}}}$

Example

Show $A_{\mathsf{TM}} \leq_{\mathrm{m}} \overline{E_{\mathsf{TM}}}$

We need to give a TM that takes as input an instance of A_{TM} and outputs an instance of $\overline{E_{\mathsf{TM}}}$

Example

Show $A_{\mathsf{TM}} \leq_{\mathrm{m}} \overline{E_{\mathsf{TM}}}$

We need to give a TM that takes as input an instance of A_{TM} and outputs an instance of $\overline{E_{\mathsf{TM}}}$

T = "On input $\langle M, w\rangle$,

1. Construct TM M_w = 'On input x,
    1. Ignore x and run M on w. If M accepts, then *accept*; if M rejects, then *reject*'
2. Output $\langle M_w\rangle$"

This is clearly computable (i.e., T doesn't loop)

Now we just need to show that $\langle M, w\rangle \in A_{\mathsf{TM}}$ iff $\langle M_w\rangle \in \overline{E_{\mathsf{TM}}}$

Example

Show $A_{\mathsf{TM}} \leq_{\mathrm{m}} \overline{E_{\mathsf{TM}}}$

We need to give a TM that takes as input an instance of A_{TM} and outputs an instance of $\overline{E_{\mathsf{TM}}}$

T = "On input $\langle M, w \rangle$,

1. Construct TM M_w = 'On input x,
   1. Ignore x and run M on w. If M accepts, then *accept*; if M rejects, then *reject*'
2. Output $\langle M_w \rangle$"

This is clearly computable (i.e., T doesn't loop)

Now we just need to show that $\langle M, w \rangle \in A_{\mathsf{TM}}$ iff $\langle M_w \rangle \in \overline{E_{\mathsf{TM}}}$

If $\langle M, w \rangle \in A_{\mathsf{TM}}$, then M accepts w so $L(M_w) = \Sigma^*$ and thus $\langle M_w \rangle \in \overline{E_{\mathsf{TM}}}$

If $\langle M, w \rangle \notin A_{\mathsf{TM}}$, then M doesn't accept w so $L(M_w) = \varnothing$ and thus $\langle M_w \rangle \notin \overline{E_{\mathsf{TM}}}$

One missing detail

What happens if the input to our T does not have the form $\langle M, w \rangle$?

One missing detail

What happens if the input to our T does not have the form $\langle M, w \rangle$?

We said it outputs ε but that's actually a problem; why?

One missing detail

What happens if the input to our T does not have the form $\langle M, w \rangle$?

We said it outputs ε but that's actually a problem; why?

$\varepsilon \in \overline{E_{\mathsf{TM}}}$

We need to modify T:
T = "On input w,

1. If w isn't of the form $\langle M, w \rangle$, then output $\langle M' \rangle$ where $L(M') = \varnothing$
2. Otherwise, construct M_w = 'On input x,
   1. Run M on w. If M accepts, then *accept*; if M rejects, then *reject*'
3. Output $\langle M_w \rangle$"

Now strings that don't have the appropriate form for A_{TM} are mapped to something that's not in $\overline{E_{\mathsf{TM}}}$

Example

We showed that $A_{\mathsf{TM}} \leq E_{\mathsf{TM}}$ when we proved that E_{TM} is undecidable; show that $A_{\mathsf{TM}} \nleq_{\mathrm{m}} E_{\mathsf{TM}}$

How do we show this?

Example

We showed that $A_{\mathsf{TM}} \leq E_{\mathsf{TM}}$ when we proved that E_{TM} is undecidable; show that $A_{\mathsf{TM}} \not\leq_{\mathrm{m}} E_{\mathsf{TM}}$
How do we show this?

By contradiction. Assume that $A_{\mathsf{TM}} \leq_{\mathrm{m}} E_{\mathsf{TM}}$. We previously showed that E_{TM} is coRE so therefore A_{TM} is coRE. But this is a contradiction because we also proved that A_{TM} is *not* coRE

Languages that are neither RE nor coRE

So far, we've seen languages like A_{TM} that are RE but not coRE and languages like E_{TM} that are coRE but not RE

It's reasonable to ask if a language must be either RE or coRE. The answer is no

Languages that are neither RE nor coRE

So far, we've seen languages like A_{TM} that are RE but not coRE and languages like E_{TM} that are coRE but not RE

It's reasonable to ask if a language must be either RE or coRE. The answer is no

The language EQ_{TM} is neither RE nor coRE

To prove this, we want to find two languages A and B such that $A \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$ and $B \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$ where A is not RE and B is not coRE

EQ_{TM} is not RE

We already showed $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$ and E_{TM} is not RE so EQ_{TM} is not RE

EQ_{TM} is not coRE

This one is a bit trickier. Let's mapping reduce A_{TM} to EQ_{TM}

How do we do this?

EQ_{TM} is not coRE

This one is a bit trickier. Let's mapping reduce A_{TM} to EQ_{TM}

How do we do this?

T = "On input $\langle M, w \rangle$,

1. Construct TM M_1 = 'On input x,
   1. If $x \neq w$, then *reject*
   2. Run M on w. If M accepts, then *accept*; if M rejects, then *reject*'
2. Construct TM M_2 = 'On input x,
   1. If $x = w$, then *accept*; otherwise *reject*'
3. Output $\langle M_1, M_2 \rangle$"

EQ_{TM} is not coRE

This one is a bit trickier. Let's mapping reduce A_{TM} to EQ_{TM}
How do we do this?

T = "On input $\langle M, w\rangle$,

1. Construct TM M_1 = 'On input x,
   1. If $x \neq w$, then *reject*
   2. Run M on w. If M accepts, then *accept*; if M rejects, then *reject*'
2. Construct TM M_2 = 'On input x,
   1. If $x = w$, then *accept*; otherwise *reject*'
3. Output $\langle M_1, M_2\rangle$"

If $\langle M, w\rangle \in A_{\mathsf{TM}}$, then M accepts w so $L(M_1) = \{w\}$. If $\langle M, w\rangle \notin A_{\mathsf{TM}}$, then M does not accept w so $L(M_1) = \varnothing$

Regardless of M, the language of M_2 is $L(M_2) = \{w\}$.

Thus $\langle M, w\rangle \in A_{\mathsf{TM}}$ iff $\langle M_1, M_2\rangle \in EQ_{\mathsf{TM}}$

Question 4

Is there a RE language A such that $EQ_{\mathsf{TM}} \leq_{\mathrm{m}} A$? Why or why not?

Question 4

Is there a RE language A such that $EQ_{\mathsf{TM}} \leq_{\mathrm{m}} A$? Why or why not?

No. EQ_{TM} is not RE, so any A such that $EQ_{\mathsf{TM}} \leq_{\mathrm{m}} A$ is also not RE

Question 5

Is there a coRE language B such that $B \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$? Why or why not?

Question 5

Is there a coRE language B such that $B \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$? Why or why not?

Yes. We showed $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$ and E_{TM} is coRE

Question 6

If C is a language and $EQ_{\mathsf{TM}} \leq_{\mathrm{m}} C$, what can we conclude about C?

Question 6

If C is a language and $EQ_{\mathsf{TM}} \leq_{\mathrm{m}} C$, what can we conclude about C?

C is neither RE nor coRE

Question 7

True or false: If $D \leq E$, then $D \leq_{\mathrm{m}} E$.

Question 7

True or false: If $D \leq E$, then $D \leq_{\mathrm{m}} E$.

False. $A_{\mathsf{TM}} \leq E_{\mathsf{TM}}$ but $A_{\mathsf{TM}} \not\leq_{\mathrm{m}} E_{\mathsf{TM}}$

Question 8

Tricky! If $F \leq_{\mathrm{m}} \Sigma^*$, what can we conclude about F?

Question 8

Tricky! If $F \leq_{\mathrm{m}} \Sigma^*$, what can we conclude about F?

$F = \Sigma^*$. Let f be the mapping. Then $w \in F \iff f(w) \in \Sigma^*$

For any language other than Σ^*, there's some string x not in the language but then $f(x) \notin \Sigma^*$; but every string is in Σ^*

Question 9

Tricky! If $\Sigma^* \leq_{\mathrm{m}} G$, what can we conclude about G?

Question 9

Tricky! If $\Sigma^* \leq_{\mathrm{m}} G$, what can we conclude about G?

We know $G \neq \varnothing$.

Since every string $w \in \Sigma^*$ needs to be mapped to an element of G, G cannot be empty

Updated table

Before today's lecture

Language	RE	coRE
A_{DFA}	✓	✓
E_{DFA}	✓	✓
EQ_{DFA}	✓	✓
A_{CFG}	✓	✓
E_{CFG}	✓	✓
EQ_{CFG}	✗	✓
DIAG	?	?
A_{TM}	✓	✗
HALT_{TM}	?	?
E_{TM}	✗	✓
EQ_{TM}	?	?
$\text{REGULAR}_{\text{TM}}$	?	?

Now

Language	RE	coRE
A_{DFA}	✓	✓
E_{DFA}	✓	✓
EQ_{DFA}	✓	✓
A_{CFG}	✓	✓
E_{CFG}	✓	✓
EQ_{CFG}	✗	✓
DIAG	?	?
A_{TM}	✓	✗
HALT_{TM}	?	✗
E_{TM}	✗	✓
EQ_{TM}	✗	✗
$\text{REGULAR}_{\text{TM}}$	?	?

$\textsc{Halt}_{\textsf{TM}}$ is RE

It's easy to show that $\textsc{Halt}_{\textsf{TM}}$ is RE

1. Construct a TM that recognizes $\textsc{Halt}_{\textsf{TM}}$
   H = "On input $\langle M, w\rangle$,
   1. Run M on w. If M halts, then *accept*"

$\textsc{Halt}_{\mathsf{TM}}$ is RE

It's easy to show that $\textsc{Halt}_{\mathsf{TM}}$ is RE

1. Construct a TM that recognizes $\textsc{Halt}_{\mathsf{TM}}$
   H = "On input $\langle M, w \rangle$,
   1. Run M on w. If M halts, then *accept*"
2. Mapping reduce $\textsc{Halt}_{\mathsf{TM}}$ to A_{TM}
   T = "On input $\langle M, w \rangle$,
   1. Construct TM M' = 'On input x,
      1. Run M on x. If M halts, then *accept*'
   2. Output $\langle M', w \rangle$"

Turning a Turing reduction into a mapping reduction

If the Turing reduction $A \leq B$ looks like:

Let R decide B and construct TM M to decide A:
M = "On input w,

1. Construct some instance w' of B
2. Run R on w' and if R accepts, then *accept*; otherwise *reject*"

Turning a Turing reduction into a mapping reduction

If the Turing reduction $A \leq B$ looks like:

Let R decide B and construct TM M to decide A:
M = "On input w,

1. Construct some instance w' of B
2. Run R on w' and if R accepts, then *accept*; otherwise *reject*"

then we can turn that into a mapping reduction

T = "On input w,

1. Construct some instance w' of B
2. Output w'"

Turning a Turing reduction into a mapping reduction

If the Turing reduction $A \leq B$ looks like:

Let R decide B and construct TM M to decide A:
M = "On input w,

1. Construct some instance w' of B
2. Run R on w' and if R accepts, then *accept*; otherwise *reject*"

then we can turn that into a mapping reduction

T = "On input w,

1. Construct some instance w' of B
2. Output w'"

Note that R must be used exactly one time and M accepts iff R accepts

$\textsc{Regular}_{\textsf{TM}}$ is not coRE

We can turn our reduction $A_{\textsf{TM}} \leq \textsc{Regular}_{\textsf{TM}}$ into a mapping reduction $A_{\textsf{TM}} \leq_{\mathrm{m}} \textsc{Regular}_{\textsf{TM}}$

$\textsc{Regular}_{\mathsf{TM}}$ is not coRE

We can turn our reduction $A_{\mathsf{TM}} \leq \textsc{Regular}_{\mathsf{TM}}$ into a mapping reduction $A_{\mathsf{TM}} \leq_{\mathrm{m}} \textsc{Regular}_{\mathsf{TM}}$

T = "On input $\langle M, w \rangle$,

1. Construct TM M' = 'On input x,
   1. If $x = 0^n 1^n$ for some n, then *accept*
   2. Otherwise, run M on w and if M accepts, then *accept*; if M rejects, then *reject*'
2. Output $\langle M' \rangle$"

$\textsc{Regular}_{\textsf{TM}}$ is not coRE

We can turn our reduction $A_{\textsf{TM}} \le \textsc{Regular}_{\textsf{TM}}$ into a mapping reduction $A_{\textsf{TM}} \le_{\mathrm{m}} \textsc{Regular}_{\textsf{TM}}$

T = "On input $\langle M, w \rangle$,

1. Construct TM M' = 'On input x,
   1. If $x = 0^n 1^n$ for some n, then *accept*
   2. Otherwise, run M on w and if M accepts, then *accept*; if M rejects, then *reject*'
2. Output $\langle M' \rangle$"

$$\langle M, w \rangle \in A_{\textsf{TM}} \iff L(M') = \Sigma^* \iff L(M') \text{ is regular} \iff \langle M' \rangle \in \textsc{Regular}_{\textsf{TM}}$$

$A_{\textsf{TM}}$ is not coRE, so $\textsc{Regular}_{\textsf{TM}}$ is not coRE

$\text{REGULAR}_{\text{TM}}$ is not RE

We could reduce from E_{TM}, but it's simpler to reduce from $\overline{A_{\text{TM}}}$

T = "On input s,

1. If $s \neq \langle M, w \rangle$ for some TM M and input w, let M' be a TM such that $L(M') = \varnothing$
2. Otherwise, construct TM M' = 'On input x,
   1. If $x \neq 0^n 1^n$ for some n, then *reject*
   2. Run M on w and if M accepts, then *accept*; if M rejects, then *reject*'
3. Output $\langle M' \rangle$"

Three cases

$\textsc{Regular}_{\mathsf{TM}}$ is not RE

We could reduce from E_{TM}, but it's simpler to reduce from $\overline{A_{\mathsf{TM}}}$

T = "On input s,

1. If $s \neq \langle M, w\rangle$ for some TM M and input w, let M' be a TM such that $L(M') = \varnothing$
2. Otherwise, construct TM M' = 'On input x,
   1. If $x \neq 0^n 1^n$ for some n, then *reject*
   2. Run M on w and if M accepts, then *accept*; if M rejects, then *reject*'
3. Output $\langle M'\rangle$"

Three cases

1. If $s \in \overline{A_{\mathsf{TM}}}$ but $s \neq \langle M, w\rangle$, then $L(M) = \varnothing$ and $\langle M'\rangle \in \textsc{Regular}_{\mathsf{TM}}$

$\textsc{Regular}_{\mathsf{TM}}$ is not RE

We could reduce from E_{TM}, but it's simpler to reduce from $\overline{A_{\mathsf{TM}}}$

T = "On input s,

1. If $s \neq \langle M, w\rangle$ for some TM M and input w, let M' be a TM such that $L(M') = \varnothing$
2. Otherwise, construct TM M' = 'On input x,
   1. If $x \neq 0^n 1^n$ for some n, then *reject*
   2. Run M on w and if M accepts, then *accept*; if M rejects, then *reject*'
3. Output $\langle M'\rangle$"

Three cases

1. If $s \in \overline{A_{\mathsf{TM}}}$ but $s \neq \langle M, w\rangle$, then $L(M) = \varnothing$ and $\langle M'\rangle \in \textsc{Regular}_{\mathsf{TM}}$
2. If $s = \langle M, w\rangle \in \overline{A_{\mathsf{TM}}}$, then $w \notin L(M)$ so $L(M') = \varnothing$ and $\langle M'\rangle \in \textsc{Regular}_{\mathsf{TM}}$

$\textsc{Regular}_{\text{TM}}$ is not RE

We could reduce from E_{TM}, but it's simpler to reduce from $\overline{A_{\text{TM}}}$

T = "On input s,

1. If $s \neq \langle M, w \rangle$ for some TM M and input w, let M' be a TM such that $L(M') = \emptyset$
2. Otherwise, construct TM M' = 'On input x,
   1. If $x \neq 0^n 1^n$ for some n, then *reject*
   2. Run M on w and if M accepts, then *accept*; if M rejects, then *reject*'
3. Output $\langle M' \rangle$"

Three cases

1. If $s \in \overline{A_{\text{TM}}}$ but $s \neq \langle M, w \rangle$, then $L(M) = \emptyset$ and $\langle M' \rangle \in \textsc{Regular}_{\text{TM}}$
2. If $s = \langle M, w \rangle \in \overline{A_{\text{TM}}}$, then $w \notin L(M)$ so $L(M') = \emptyset$ and $\langle M' \rangle \in \textsc{Regular}_{\text{TM}}$
3. If $s \notin \overline{A_{\text{TM}}}$, then $s = \langle M, w \rangle$ and $w \in L(M)$. In this case, $L(M') = \{0^n 1^n \mid n \geq 0\}$ so $\langle M' \rangle \notin \textsc{Regular}_{\text{TM}}$

Since $\overline{A_{\text{TM}}}$ is not RE, $\textsc{Regular}_{\text{TM}}$ is not RE

Updated table

Before today's lecture

Language	RE	coRE
A_{DFA}	✔	✔
E_{DFA}	✔	✔
EQ_{DFA}	✔	✔
A_{CFG}	✔	✔
E_{CFG}	✔	✔
EQ_{CFG}	✖	✔
DIAG	?	?
A_{TM}	✔	✖
$\text{HALT}_{\mathsf{TM}}$	?	?
E_{TM}	✖	✔
EQ_{TM}	?	?
$\text{REGULAR}_{\mathsf{TM}}$	?	?

Now

Language	RE	coRE
A_{DFA}	✔	✔
E_{DFA}	✔	✔
EQ_{DFA}	✔	✔
A_{CFG}	✔	✔
E_{CFG}	✔	✔
EQ_{CFG}	✖	✔
DIAG	?	?
A_{TM}	✔	✖
$\text{HALT}_{\mathsf{TM}}$	✔	✖
E_{TM}	✖	✔
EQ_{TM}	✖	✖
$\text{REGULAR}_{\mathsf{TM}}$	✖	✖