Mapping Reductions
Announcements

• Casual CS Dinner for Women Studying Computer Science: **Thursday, March 7** at **6PM** in **Gates 219**!

• RSVP through the email link sent out earlier today.
Announcements

• All Problem Set 6's are graded, will be returned at end of lecture.

• Problem Set 7 due right now, or due at Thursday at 12:50PM with a late day.
  • Please submit no later than 12:50PM; we're hoping to get solutions posted then. This is a hard deadline.

• Problem Set 8 out, due next Monday, March 11 at 12:50PM.
  • Explore the limits of computation!
Recap from Last Time
There is a TM $M$ where $M$ accepts $w$ iff $w \in L$. There is a TM $M$ where $M$ rejects $w$ iff $w \notin L$. What's out here?
A Repeating Pattern
\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

For \( A_{TM} \):

1. Construct the string \( \langle M, \varepsilon \rangle \).
2. Run \( R \) on \( \langle M, \varepsilon \rangle \).
3. If \( R \) accepts \( \langle M, \varepsilon \rangle \), then \( H \) accepts \( \langle M, \varepsilon \rangle \).
4. If \( R \) rejects \( \langle M, \varepsilon \rangle \), then \( H \) rejects \( \langle M, \varepsilon \rangle \).
From $\overline{A_{TM}}$ to $L_D$

$H = "On input $\langle M \rangle$:"

- Construct the string $\langle M, \langle M \rangle \rangle$.
- Run $R$ on $\langle M, \langle M \rangle \rangle$.
- If $R$ accepts $\langle M, \langle M \rangle \rangle$, then $H$ accepts $\langle M, \langle M \rangle \rangle$.
- If $R$ rejects $\langle M, \langle M \rangle \rangle$, then $H$ rejects $\langle M, \langle M \rangle \rangle."$
From $\text{HALT}$ to $\text{A}_{\text{TM}}$}

$H = "\text{On input } \langle M, w \rangle:\n\cdot \text{ Build } M \text{ into } M' \text{ so } M' \text{ loops when } M \text{ rejects.}\n\cdot \text{ Run } D \text{ on } \langle M', w \rangle.\n\cdot \text{ If } D \text{ accepts } \langle M', w \rangle, \text{ then } H \text{ accepts } \langle M, w \rangle.\n\cdot \text{ If } D \text{ rejects } \langle M', w \rangle, \text{ then } H \text{ rejects } \langle M, w \rangle."
The General Pattern

$H = \text{"On input } w:\"
\cdot \text{ Transform the input } w \text{ into } f(w).$
\cdot \text{ Run machine } R \text{ on } f(w).$
\cdot \text{ If } R \text{ accepts } f(w), \text{ then } H \text{ accepts } w.$
\cdot \text{ If } R \text{ rejects } f(w), \text{ then } H \text{ rejects } w.$"
Reductions

• Intuitively, problem $A$ reduces to problem $B$ iff a solver for $B$ can be used to solve problem $A$. 

\[
\begin{align*}
L_D & \quad \text{Can be converted to} \quad \overline{A_{TM}} \\
\text{Can be used to solve} & \quad \overline{A_{TM}}
\end{align*}
\]
Reductions

- Intuitively, problem A reduces to problem B iff a solver for B can be used to solve problem A.
Reductions

• Intuitively, problem $A$ reduces to problem $B$ iff a solver for $B$ can be used to solve problem $A$.

• Reductions can be used to show certain problems are “solvable:”

  If $A$ reduces to $B$ and $B$ is “solvable,” then $A$ is “solvable.”

• Reductions can be used to show certain problems are “unsolvable:”

  If $A$ reduces to $B$ and $A$ is “unsolvable,” then $B$ is “unsolvable.”
Defining Reductions

- A reduction from $A$ to $B$ is a function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ such that

  For any $w \in \Sigma_1^*$, $w \in A$ iff $f(w) \in B$
Defining Reductions

- A reduction from $A$ to $B$ is a function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ such that

  For any $w \in \Sigma_1^*$, $w \in A$ iff $f(w) \in B$

- Every $w \in A$ maps to some $f(w) \in B$.
- Every $w \notin A$ maps to some $f(w) \notin B$.
- $f$ does not have to be injective or surjective.
Computable Functions

- Not all mathematical functions can be computed by Turing machines.
- A function $f : \Sigma_1^* \to \Sigma_2^*$ is called a **computable function** if there is some TM $M$ with the following behavior:

  "On input $w$:
  
  Compute $f(w)$ and write it on the tape.
  Move the tape head to the start of $f(w)$.
  Halt."
Mapping Reductions

- A function \( f : \Sigma_1^* \rightarrow \Sigma_2^* \) is called a \textbf{mapping reduction} from \( A \) to \( B \) iff
  - For any \( w \in \Sigma_1^* \), \( w \in A \) iff \( f(w) \in B \).
  - \( f \) is a computable function.
- Intuitively, a mapping reduction from \( A \) to \( B \) says that a computer can transform any instance of \( A \) into an instance of \( B \) such that the answer to \( B \) is the answer to \( A \).
Compute $f(w)$

$H = \text{"On input } w: \text{"}

- Transform the input $w$ into $f(w)$.
- Run machine $R$ on $f(w)$.
- If $R$ accepts $f(w)$, then $H$ accepts $w$.
- If $R$ rejects $f(w)$, then $H$ rejects $w$."

$H$ accepts $w$ iff $R$ accepts $f(w)$ iff $f(w) \in B$ iff $w \in A$
Mapping Reducibility

• If there is a mapping reduction from language \( A \) to language \( B \), we say that language \( A \) is **mapping reducible** to language \( B \).

• Notation: \( A \leq_{M} B \) iff language \( A \) is mapping reducible to language \( B \).

• Note that we reduce *languages*, not *machines*.

• Interesting exercise: Show \( \leq_{M} \) is reflexive and transitive, but not antisymmetric.
$A \leq_M B$
Why Mapping Reducibility Matters

- **Theorem**: If $B \in \mathbb{R}$ and $A \leq_{M} B$, then $A \in \mathbb{R}$.
- **Theorem**: If $B \in \text{RE}$ and $A \leq_{M} B$, then $A \in \text{RE}$.
- **Theorem**: If $B \in \text{co-RE}$ and $A \leq_{M} B$, then $A \in \text{co-RE}$.

- *Intuitively*: $A \leq_{M} B$ means “$A$ is not harder than $B$.”
Why Mapping Reducibility Matters

- **Theorem**: If $A \notin \mathbb{R}$ and $A \leq^M B$, then $B \notin \mathbb{R}$.
- **Theorem**: If $A \notin \text{RE}$ and $A \leq^M B$, then $B \notin \text{RE}$.
- **Theorem**: If $A \notin \text{co-RE}$ and $A \leq^M B$, then $B \notin \text{co-RE}$.

- *Intuitively*: $A \leq^M B$ means “$B$ is at least as hard as $A$.”
Why Mapping Reducibility Matters

\[ A \leq_{M} B \]

If this one is "easy" (R, RE, co-RE)...

... then this one is "easy" (R, RE, co-RE) too.
Why Mapping Reducibility Matters

If this one is “hard” (not R, not RE, or not co-RE)...

\[ A \leq^M B \]

...then this one is “hard” (not R, not RE, or not co-RE) too.
Using Mapping Reductions
Revisiting our Proofs

• Consider the language

\[ L = \{ \langle M \rangle \mid M \text{ is a TM and } M \text{ accepts } \varepsilon \} \]

• We have already proven that this language is \textbf{RE} by building a TM for it.

• Let's repeat this proof using mapping reductions.

• Specifically, we will prove

\[ L \leq_{\text{TM}} A_{\text{TM}} \]
\[ L = \{ \langle M \rangle \mid M \text{ is a TM and } M \text{ accepts } \varepsilon \} \]

- To prove \( L \leq_M A_{\text{TM}} \), we will need to find a computable function \( f \) such that
  \[ \langle M \rangle \in L \iff f(\langle M \rangle) \in A_{\text{TM}} \]

- Since \( A_{\text{TM}} \) is a language of TM/string pairs, let's assume \( f(\langle M \rangle) = \langle N, w \rangle \) for some TM \( N \) and string \( w \) (which we'll pick later):
  \[ \langle M \rangle \in L \iff \langle N, w \rangle \in A_{\text{TM}} \]

- Substituting definitions:
  \[ M \text{ accepts } \varepsilon \iff N \text{ accepts } w \]

- Choose \( N = M, w = \varepsilon \). So \( f(\langle M \rangle) = \langle M, \varepsilon \rangle \).
One Interpretation of the Reduction

\[ H = \text{“On input } \langle M \rangle \text{:} \]
\[ \begin{align*}
  &\cdot \text{ Run machine } R \text{ on } \langle M, \varepsilon \rangle. \\
  &\cdot \text{ If } R \text{ accepts } \langle M, \varepsilon \rangle, \text{ then } H \text{ accepts } w. \\
  &\cdot \text{ If } R \text{ rejects } \langle M, \varepsilon \rangle, \text{ then } H \text{ rejects } w. \\
\end{align*} \]

\[ H \text{ accepts } \langle M \rangle \iff R \text{ accepts } \langle M, \varepsilon \rangle \iff M \text{ accepts } \varepsilon \iff \langle M \rangle \in L \]
Theorem: $L \in \text{RE}$.
Proof: We will prove that $L \leq_A A_{\text{TM}}$. Since $A_{\text{TM}} \in \text{RE}$, this proves $L \in \text{RE}$ as well.

Consider the function $f(\langle M \rangle) = \langle M, \varepsilon \rangle$. We state without proof that this function is computable and claim that $f$ is a mapping reduction from $L$ to $A_{\text{TM}}$. To see this, note that $f(\langle M \rangle) = \langle M, \varepsilon \rangle \in A_{\text{TM}}$ iff $M$ accepts $\varepsilon$ iff $\langle M \rangle \in L$, so $\langle M \rangle \in L$ iff $f(\langle M \rangle) \in A_{\text{TM}}$.

Since $f$ is a mapping reduction from $L$ to $A_{\text{TM}}$, we have $L \leq_M A_{\text{TM}}$, and thus $L \in \text{RE}$. ■
What Did We Prove?

Machine $H$:

$H = \text{“On input } \langle M \rangle: \text{“}$

- Run machine $R$ on $\langle M, \varepsilon \rangle$.
- If $R$ accepts $\langle M, \varepsilon \rangle$, then $H$ accepts $w$.
- If $R$ rejects $\langle M, \varepsilon \rangle$, then $H$ rejects $w$.”

$H$ accepts $\langle M \rangle$ iff $R$ accepts $\langle M, \varepsilon \rangle$ iff $M$ accepts $\varepsilon$ iff $\langle M \rangle \in L$.
Interpreting Mapping Reductions

- If $A \leq^M B$, there is a known construction to turn a TM for $B$ into a TM for $A$.
- When doing proofs with mapping reductions, you do not need to show the overall construction.
- You just need to prove that
  - $f$ is a computable function, and
  - $w \in A$ iff $f(w) \in B$. 
Another Mapping Reduction
$L_D$ and $\overline{A}_{TM}$

- Earlier, we proved $\overline{A}_{TM} \notin \text{RE}$ by proving that
  
  If $\overline{A}_{TM} \in \text{RE}$, then $L_D \in \text{RE}$.

- The proof constructed this TM, assuming $R$ was a recognizer for $\overline{A}_{TM}$.

\[
H = \text{“On input } \langle M \rangle: \\
\quad \cdot \text{Construct the string } \langle M, \langle M \rangle \rangle. \\
\quad \cdot \text{Run } R \text{ on } \langle M, \langle M \rangle \rangle. \\
\quad \cdot \text{If } R \text{ accepts } \langle M, \langle M \rangle \rangle, \text{ then } H \text{ accepts } \langle M \rangle. \\
\quad \cdot \text{If } R \text{ rejects } \langle M, \langle M \rangle \rangle, \text{ then } H \text{ rejects } \langle M \rangle."
\]

- Let's do another proof using mapping reductions.
\[ L_D \leq_{M} \overline{A_{TM}} \]

- To prove that \( \overline{A_{TM}} \notin \text{RE} \), we will prove \( L_D \leq_{M} \overline{A_{TM}} \).

- By our earlier theorem, since \( L_D \notin \text{RE} \), we have that \( \overline{A_{TM}} \notin \text{RE} \).

- Intuitively: \( \overline{A_{TM}} \) is “at least as hard” as \( L_D \), and since \( L_D \notin \text{RE} \), this means \( \overline{A_{TM}} \notin \text{RE} \).
\( L_D \leq_M \overline{A}_{TM} \)

- Goal: Find a computable function \( f \) such that
  \[ \langle M \rangle \in L_D \iff f(\langle M \rangle) \in \overline{A}_{TM} \]

- Simplifying this using the definition of \( L_D \)
  \[ M \text{ does not accept } \langle M \rangle \iff f(\langle M \rangle) \in \overline{A}_{TM} \]

- Let's assume that \( f(\langle M \rangle) \) has the form \( \langle N, w \rangle \) for some TM \( N \) and string \( w \). This means that
  \[ M \text{ does not accept } \langle M \rangle \iff \langle N, w \rangle \in \overline{A}_{TM} \]

  \[ M \text{ does not accept } \langle M \rangle \iff N \text{ does not accept } w \]

- If we can choose \( w \) and \( N \) such that the above is true, we will have our reduction from \( L_D \) to \( \overline{A}_{TM} \).

- Choose \( N = M \) and \( w = \langle M \rangle \).
One Interpretation of the Reduction

Machine $H$

- Compute $f$ on input $\langle M \rangle$.
- Run machine $R$ on $\langle M, \langle M \rangle \rangle$.
- If $R$ accepts $\langle M, \langle M \rangle \rangle$, then $H$ accepts $w$.
- If $R$ rejects $\langle M, \langle M \rangle \rangle$, then $H$ rejects $w$.

$H$ accepts $\langle M \rangle$ iff $R$ accepts $\langle M, \langle M \rangle \rangle$ iff $M$ does not accept $\langle M \rangle$ iff $\langle M \rangle \in L_D$.
**Theorem:** $\overline{A}_{TM} \notin \text{RE}$.  

**Proof:** We exhibit a mapping reduction $f$ from $L_D$ to $\overline{A}_{TM}$.

Consider the function $f$ defined as follows:

$$f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$$

We claim that $f$ can be computed by a TM and omit the details from this proof. We will prove that $\langle M \rangle \in L_D$ iff $f(\langle M \rangle) \in \overline{A}_{TM}$. Note that $f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$, so $f(\langle M \rangle) \in \overline{A}_{TM}$ iff $\langle M, \langle M \rangle \rangle \in \overline{A}_{TM}$. By definition of $\overline{A}_{TM}$, $\langle M, \langle M \rangle \rangle \in \overline{A}_{TM}$ iff $\langle M \rangle \notin \mathcal{L}(M)$. Finally, note that $\langle M \rangle \notin \mathcal{L}(M)$ iff $\langle M \rangle \in L_D$. Thus $f(\langle M \rangle) \in \overline{A}_{TM}$ iff $\langle M \rangle \in L_D$, so $f$ is a mapping reduction from $L_D$ to $\overline{A}_{TM}$.

Since $f$ is a mapping reduction from $L_D$ to $\overline{A}_{TM}$, we have $L_D \leq_M \overline{A}_{TM}$. Since $L_D \notin \text{RE}$ and $L_D \leq_M \overline{A}_{TM}$, this means $\overline{A}_{TM} \notin \text{RE}$, as required. ■
Another Example of Mapping Reductions
A More Elaborate Reduction

- Since $\overline{A_{TM}} \notin \text{RE}$, there is no algorithm for determining whether a TM will not accept a given string.
- Could we check instead whether a TM never accepts a string?
- Consider the language

$$L_e = \{ \langle M \rangle \mid M \text{ is a TM and } M \text{ never accepts} \}$$

- How “hard” is $L_e$? Is it $\text{R}$, $\text{RE}$, co-$\text{RE}$, or none of these?
Building an Intuition

- Before we even try to prove how “hard” this language is, we should build an intuition for its difficulty.

- $L_e$ is probably not in $\text{RE}$, since if we were convinced a TM never accepted, it would be hard to find positive evidence of this.

- $L_e$ is probably in $\text{co-RE}$, since if we were convinced that a TM did accept some string, we could exhaustively search over all strings and try to find the string it accepts.

- Best guess: $L_e \in \text{co-RE} - \text{R}$. 
\[ \overline{A_{TM}} \leq_M L_e \]

- We will prove that \( L_e \notin \text{RE} \) by showing that \( \overline{A_{TM}} \leq_M L_e \).
  (This also proves \( L_e \notin \text{R} \)).

- We want to find a function \( f \) such that
  \[ \langle M, w \rangle \in \overline{A_{TM}} \iff f(\langle M, w \rangle) \in L_e \]

- Since \( L_e \) is a language of TM descriptions, let's assume
  \( f(\langle M, w \rangle) = \langle N \rangle \) for some TM \( N \). Then
  \[ \langle M, w \rangle \in \overline{A_{TM}} \iff \langle N \rangle \in L_e \]

- Expanding out definitions, we get
  \[ M \text{ doesn't accept } w \text{ iff } N \text{ doesn't accept any strings} \]

- How do we pick the machine \( N \)?
The Reduction

- Find a TM $N$ such that $N$ does not accept any strings iff $M$ does not accept $w$.
- **Key idea:** Build $N$ such that running $N$ on any input runs $M$ on $w$.
- Here is one choice of $N$:

  $$N = \text{"On input } x:\$$
  $$\text{Ignore } x.$$  
  $$\text{Run } M \text{ on } w.$$  
  $$\text{If } M \text{ accepts } w, \text{ then } N \text{ accepts } x.$$  
  $$\text{If } M \text{ rejects } w, \text{ then } N \text{ rejects } x.$$  
- Notice that $N$ “amplifies” what $M$ does on $w$:
  - If $M$ does not accept $w$, $N$ does not accept anything.
  - If $M$ does accept $w$, $N$ accepts everything.
The Reduction

\[ \langle M, w \rangle \xrightarrow{\text{Construct } N \text{ from } \langle M, w \rangle} \langle N \rangle \xrightarrow{\text{Machine for } L_e} \]

This is a recognizer for \( \overline{A_{TM}} \)!
The Reduction

Simulate $M$ on $w$ (Ignored)

Construct $N$ from $\langle M, w \rangle$

$\langle M, w \rangle$ $\rightarrow$ $\langle N \rangle$ $\rightarrow$ Machine for $L_e$

How is this step possible?

$\chi$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$
Justifying \( N \)

- Notice that our machine \( N \) has the machine \( M \) and string \( w \) built into it!

- This is different from the machines we have constructed in the past.

- How do we justify that it's possible for some TM to construct a new TM at all?

\( N = \text{“On input } x: \text{ Ignore } x. \text{ Run } M \text{ on } w. \text{ If } M \text{ accepts } w, \text{ accept. If } M \text{ rejects } w, \text{ reject.”} \)
$N = \text{"On input } x:\text{ }
\cdot \text{Ignore } x.
\cdot \text{Run } M \text{ on } w.
\cdot \text{If } M \text{ accepts } w, \text{ then } N \text{ accepts } x.
\cdot \text{If } M \text{ rejects } w, \text{ then } N \text{ rejects } x.\text{"}

Hypothetically, assume that $w$ is the string $11011$.
The Takeaway Point

• Turing machines can embed TMs inside of other TMs.

• TMs of the following form are legal:

\[ H = \text{"On input } \langle M, w \rangle, \text{ where } M \text{ is a TM:}\]
  \[ \quad \text{• Construct } N = \text{"On input } x:\]
  \[ \quad \text{• Do something with } x.\]
  \[ \quad \text{• Run } M \text{ on } w.\]
  \[ \quad \text{• } \ldots\]
  \[ \quad \text{• Do something with } N." \]
Theorem: $\overline{A}_{TM} \leq_M L_e$.

Proof: We exhibit a mapping reduction from $\overline{A}_{TM}$ to $L_e$.

For any TM/string pair $\langle M, w \rangle$, let $f(\langle M, w \rangle) = \langle N \rangle$, where $\langle N \rangle$ is defined in terms of $M$ and $w$ as follows:

$$N = \text{"On input } x:\n\text{Ignore } x.\n\text{Run } M \text{ on } w.\n\text{If } M \text{ accepts } w, \text{ then } N \text{ accepts } x.\n\text{If } M \text{ rejects } w, \text{ then } N \text{ rejects } x."$$

We state without proof that $N$ is computable. We further claim that $\langle M, w \rangle \in \overline{A}_{TM}$ iff $f(\langle M, w \rangle) \in L_e$. To see this, note that $f(\langle M, w \rangle) = N \in L_e$ iff $N$ does not accept any strings. We claim that $N$ does not accept any strings iff $M$ does not accept $w$. To see this, note that $M$ does not accept $w$ iff $M$ loops on $w$ or $M$ rejects $w$. By construction, if $M$ loops on $w$, then $N$ loops on all strings, and if $M$ rejects $w$, then $N$ rejects all strings. Thus $N$ does not accept any strings iff $M$ does not accept $w$. Finally, $M$ does not accept $w$ iff $\langle M, w \rangle \in \overline{A}_{TM}$. Thus $\langle M, w \rangle \in \overline{A}_{TM}$ iff $f(\langle M, w \rangle) \in L_e$, so $f$ is a mapping reduction from $\overline{A}_{TM}$ to $L_e$, and so $\overline{A}_{TM} \leq_M L_e$, as required. □
Recitation Sections
The Limits of Computability

CO-RE
- $\overline{\text{HALT}}$
- $\overline{\text{ONES}}$
- $L_D$
- $\overline{A_{TM}}$
- $L_e$

R
- $\text{ADD}$
- $\text{SEARCH}$

RE
- $\overline{L_D}$
- $\overline{L_e}$
- $\text{HALT}$
- $\text{ONES}$
- $A_{TM}$

What's out here?
**RE ∪ co-RE is Not Everything**

- Using the same reasoning as the first day of lecture, we can show that there must be problems that are neither RE nor co-RE.
- There are more sets of strings than TMs.
- There are more sets of strings than twice the number of TMs.
- What do these languages look like?
An Extremely Hard Problem

- Recall: All regular languages are also RE.
- This means that some TMs accept regular languages and some TMs do not.
- Let $\text{REGULAR}_{TM}$ be the language of all TM descriptions that accept regular languages:

$$\text{REGULAR}_{TM} = \{ \langle M \rangle \mid L(M) \text{ is regular} \}$$

- Is $\text{REGULAR}_{TM} \in \text{R}$? How about $\text{RE}$? How about co-$\text{RE}$?
Building an Intuition

• If you were convinced that a TM had a regular language, how would you mechanically verify that?

• If you were convinced that a TM had a nonregular language, how would you mechanically verify that?

• Both of these seem difficult, if not impossible. Chances are \textsc{REGULAR}_{\textsc{TM}} is neither \textsc{RE} nor \textsc{co-RE}. 
REGULAR\textsubscript{TM} \not\in \text{RE}

- It turns out that REGULAR\textsubscript{TM} is unrecognizable, meaning that there is no computer program that can confirm that another TM's language is regular!
- To do this, we'll do a reduction from $L_D$ and prove that $L_D \leq_M \text{REGULAR}_\text{TM}$. 
\[ L_D \leq_M \text{REGULAR}_{\text{TM}} \]

- We want to find a computable function \( f \) such that
  \[
  \langle M \rangle \in L_D \quad \iff \quad f(\langle M \rangle) \in \text{REGULAR}_{\text{TM}}.
  \]
- We need to choose \( N \) such that \( f(\langle M \rangle) = \langle N \rangle \) for some TM \( N \). Then
  \[
  \langle M \rangle \in L_D \quad \iff \quad f(\langle M \rangle) \in \text{REGULAR}_{\text{TM}}
  \]
  \[
  \langle M \rangle \in L_D \quad \iff \quad \langle N \rangle \in \text{REGULAR}_{\text{TM}}
  \]
  \[
  \langle M \rangle \notin \mathcal{L}(M) \quad \iff \quad \mathcal{L}(N) \text{ is regular.}
  \]
- Question: How do we pick \( N \)?
\[
L_D \leq_M \text{REGULAR}_{TM}
\]

• We want to construct some \( N \) out of \( M \) such that
  • If \( \langle M \rangle \in \mathcal{L}(M) \), then \( \mathcal{L}(N) \) is not regular.
  • If \( \langle M \rangle \notin \mathcal{L}(M) \), then \( \mathcal{L}(N) \) is regular.

• One option: choose two languages, one regular and one nonregular, then construct \( N \) so its language switches from regular to nonregular based on whether \( \langle M \rangle \notin \mathcal{L}(M) \).
  • If \( \langle M \rangle \in \mathcal{L}(M) \), then \( \mathcal{L}(N) = \{ 0^n1^n \mid n \in \mathbb{N} \} \)
  • If \( \langle M \rangle \notin \mathcal{L}(M) \), then \( \mathcal{L}(N) = \emptyset \)
The Reduction

• We want to build $N$ from $M$ such that
  • If $\langle M \rangle \in \mathcal{L}(M)$, then $\mathcal{L}(N) = \{ 0^n1^n \mid n \in \mathbb{N} \}$
  • If $\langle M \rangle \notin \mathcal{L}(M)$, then $\mathcal{L}(N) = \emptyset$
• Here is one way to do this:
  
  $N =$ “On input $x$:
  
  If $x$ does not have the form $0^n1^n$, reject.
  Run $M$ on $\langle M \rangle$.
  If $M$ accepts, accept $x$.
  If $M$ rejects, reject $x$.”
Theorem: $L_D \leq_M \text{REGULAR}_{\text{TM}}$.

Proof: We exhibit a mapping reduction from $L_D$ to $\text{REGULAR}_{\text{TM}}$.

For any TM $M$, let $f(\langle M \rangle) = \langle N \rangle$, where $N$ is defined in terms of $M$ as follows:

\[ N = \text{"On input } x:\]
\[ \text{If } x \text{ does not have the form } 0^n1^n, \text{ then } N \text{ rejects } x. \]
\[ \text{Run } M \text{ on } \langle M \rangle. \]
\[ \text{If } M \text{ accepts } \langle M \rangle, \text{ then } N \text{ accepts } x. \]
\[ \text{If } M \text{ rejects } \langle M \rangle, \text{ then } N \text{ rejects } x. \]

We claim $f$ is computable and omit the details from this proof. We further claim that $\langle M \rangle \in L_D$ iff $f(\langle M \rangle) \in \text{REGULAR}_{\text{TM}}$. To see this, note that $f(\langle M \rangle) = \langle N \rangle \in \text{REGULAR}_{\text{TM}}$ iff $\mathcal{L}(N)$ is regular. We claim that $\mathcal{L}(N)$ is regular iff $\langle M \rangle \notin \mathcal{L}(M)$. To see this, note that if $\langle M \rangle \notin \mathcal{L}(M)$, then $N$ never accepts any strings. Thus $\mathcal{L}(N) = \emptyset$, which is regular. Otherwise, if $\langle M \rangle \in \mathcal{L}(M)$, then $N$ accepts all strings of the form $0^n1^n$, so we have that $\mathcal{L}(N) = \{0^n1^n \mid n \in \mathbb{N}\}$, which is not regular. Finally, $\langle M \rangle \notin \mathcal{L}(\langle M \rangle)$ iff $\langle M \rangle \in L_D$. Thus $\langle M \rangle \in L_D$ iff $f(\langle M \rangle) \in \text{REGULAR}_{\text{TM}}$, so $f$ is a mapping reduction from $L_D$ to $\text{REGULAR}_{\text{TM}}$. Therefore, $L_D \leq_M \text{REGULAR}_{\text{TM}}$. □