Reducibility

We have

- established the Turing Machine as the Gold Standard of computability . . .
- seen examples of solvable problems . . .
- and one problem, A_{TM}, that is computationally unsolvable.

Now we look at other computationally unsolvable problems.

Reducibility

Example:

- Finding your way around a city reduces to ...
- obtaining a city map.

Reducibility

Always involves two problems A and B.

Definition: If A reduces to B, then any solution of B solves A.

Remark: This definition says nothing about solving A or B alone.

Examples

Reductions:

- Traveling Boston to Paris . . .
- buying plane ticket . . .
- earning the money for that ticket . . .
- finding a job.

Examples

Reductions:

- Measuring area of rectangle . . .
- measuring lengths of sides.

Also:

- Solving system of linear equations . . .
- inverting a matrix.

Reducibility

If A is reducible to B, then

- A cannot be harder than B
- if B is decidable, so is A.
- if A is undecidable and reducible to B, then B is undecidable.

Undecidable Problems

We have already established that A_{TM} is undecidable.

Here is a related problem.

$$H_{\mathsf{TM}} = \{\langle M, w\rangle | M \text{ is a TM and } M \text{ halts on input } w\}$$

Question: How is this language different from A_{TM}?

Undecidable Problems

$$H_{\mathsf{TM}} = \{\langle M, w\rangle | M \text{ is a TM and } M \text{ halts on input } w\}$$

Theorem: H_{TM} is undecidable.

Basic idea:

- By contradiction.
- Assume H_{TM} is decidable.
- Let R be a TM that decides H_{TM}.
- Use R to construct S, a TM that decides A_{TM}.
- Deciding A_{TM} is reduced to H_{TM}.
- since deciding A_{TM} is undecidable, so is deciding H_{TM}.

Undecidable Problems

Theorem: H_{TM} is undecidable.

Assume R decides H_{TM}.

Define S as follows.

On input $\langle M, w \rangle$,

- Run R on $\langle M, w \rangle$.
- If R rejects, *reject*.
- If R accepts, simulate M on w until it halts.
- If M accepted, *accept*; otherwise *reject*.

Undecidable Problems

Does a TM accept any string at all?

$$E_{\mathsf{TM}} = \{\langle M, w\rangle | M \text{ is a TM and } L(M) = \emptyset\}$$

Theorem: E_{TM} is undecidable.

Basic idea:

- By contradiction.
- Assume E_{TM} is decidable.
- Let R be a TM that decides E_{TM}.
- Use R to construct S, a TM that decides A_{TM}.

Undecidable Problems

$$E_{\text{TM}} = \{\langle M, w\rangle | M \text{ is a TM and } L(M) = \emptyset\}$$

Idea: When S receives input $\langle M, w\rangle$, it calls R with input M.

- If R accepts, then *reject*, because M does not accept w.
- But what if R rejects?

Idea: let's modify M.

Undecidable Problems

$$E_{\mathsf{TM}} = \{\langle M, w\rangle | M \text{ is a TM and } L(M) = \emptyset\}$$

Define M_1: on input x,

1. if $x \neq w$, *reject*.

2. if $x = w$, run M on w and *accept* if M does.

M_1 either

- accepts w, or
- accepts nothing

Undecidable Problems

Define M_1: on input x,

1. if $x \neq w$, *reject*.
2. if $x = w$, run M on w and *accept* if M does.

Question: Can a TM construct M_1 from M?

Answer: Yes, because we need only add extra states to perform test.

Undecidable Problems

$$E_{\mathsf{TM}} = \{\langle M, w\rangle | M \text{ is a TM and } L(M) = \emptyset\}$$

Theorem: E_{TM} is undecidable.

Define S as follows.

On input $\langle M, w\rangle$, where M is a TM and w a string,

- Construct M_1 from M and w.
- Run R on input $\langle M_1\rangle$
- if R accepts, *reject*; if R rejects, *accept*.

Undecidable Problems

Does a TM accept a regular language?

$$R_{\text{TM}} = \{\langle M, w\rangle | M \text{ is a TM and } L(M) \text{ is regular}\}$$

Theorem: R_{TM} is undecidable.

- By contradiction.
- Assume E_{TM} is decidable.
- Let R be a TM that decides E_{TM}.
- Use R to construct S, a TM that decides A_{TM}.

But how?

Undecidable Problems

$$R_{\mathsf{TM}} = \{\langle M, w\rangle | M \text{ is a TM and } L(M) \text{ is regular}\}$$

Modify M so that the resulting language accepts a regular language if and only if M accepts w.

Design M_2 so that

- if M does not accept w, then M_2 accepts non-regular $\{0^n 1^n | n \geq 0\}$
- if M accepts w, then M_2 accepts regular Σ^*.

Undecidable Problems

From M and w define M_2:

On input x,

1. If x has the form $0^n 1^n$, *accept*.
2. Otherwise, run M on input w and *accept*if M accepts w.

Claim:

- If M does not accept w, then M_2 accepts $\{0^n 1^n | n \geq 0\}$.
- If M accepts w, then M_2 accepts Σ^*.

Undecidable Problems

$$R_{\mathsf{TM}} = \{\langle M, w\rangle | M \text{ is a TM and } L(M) \text{ is regular}\}$$

Theorem: R_{TM} is undecidable.

Define S:

On input $\langle M, w\rangle$,

1. Construct M_2 from M and w.

2. Run R on input $\langle M_2\rangle$.

3. If R accepts, *accept*; if R rejects, *reject*.

Undecidable Problems

Are two TMs equivalent?

$\mathsf{EQ}_{\mathsf{TM}} = \{\langle M_1, M_2\rangle | M_1, M_2$ are TMs and $L(M_1) = L(M_2)\}$

Theorem: $\mathsf{EQ}_{\mathsf{TM}}$ is undecidable.

We are getting tired of reducing everything to deciding A_{TM}.

Let's try reduction to E_{TM}.

Undecidable Problems

$\mathsf{EQ}_{\mathsf{TM}} = \{\langle M_1, M_2\rangle | M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

Theorem: $\mathsf{EQ}_{\mathsf{TM}}$ is undecidable.

Idea:

- E_{TM} is the problem of testing whether a TM language is empty.
- $\mathsf{EQ}_{\mathsf{TM}}$ is the problem of testing whether two TM languages are the same.
- If one of these two TM languages happens to be empty, then we are back to E_{TM}.
- So E_{TM} is a special case of $\mathsf{EQ}_{\mathsf{TM}}$.

The rest is easy.

Undecidable Problems

$\text{EQ}_{\text{TM}} = \{\langle M_1, M_2\rangle | M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

Theorem: EQ_{TM} is undecidable.

Let M_{NO} be the TM:

1. *reject*.

Let R decide EQ_{TM}.

Let S be:

On input $\langle M\rangle$:

1. Run R on input $\langle M, M_{\text{NO}}\rangle$.
2. If R accepts, *accept*; if R rejects, *reject*.

Bucket of Undecidable Problems

Same techniques prove undecidability of

- Does a TM accept a decidable language?
- Does a TM accept a context-free language?
- Does a TM accept a finite language?
- Does a TM halt on all inputs?
- Is there an input string that causes a TM to traverse all its states?

Rice's Theorem

By now, some of you may have become cynical and embittered.

- Like, been there, done that, bought the T-shirt.
- Looks like *any* non-trivial property of TMs is undecidable.

Rice's Theorem

By now, some of you may have become cynical and embittered.

- Like, been there, done that, bought the T-shirt.
- Looks like *any* non-trivial property of TMs is undecidable.

That is correct.

Rice's Theorem

Theorem: If $\mathcal{C}$ is a proper non-empty subset of the set of enumerable languages, then it is undecidable whether for a given TM M, $L(M)$ is in $\mathcal{C}$.

(See Papadimitriou page 62.)

Rice's Theorem

Theorem: If $\mathcal{C}$ is a proper non-empty subset of the set of enumerable languages, then it is undecidable whether for a given TM M, $L(M)$ is in $\mathcal{C}$.

Proof by reduction to H_{TM} (does M halt on input x?).

- Assume R decides $\mathcal{C}$.
- Use R to implement S, which decides H_{TM}.

continued . . .

Rice's Theorem

- Assume WLOG that $\emptyset \notin \mathcal{C}$.
- Otherwise, look at $\overline{\mathcal{C}}$, also proper and non-empty.
- Also, there exists some $L \in \mathcal{C}$ accepted by some M_L.

continued . . .

Rice's Theorem

Given M and x, we will construct M_0 such that:

- If M accepts x, then $L(M_0) \in \mathcal{C}$.
- If M does not accept x, then $L(M_0) \notin \mathcal{C}$.

Deciding whether $M_0 \in \mathcal{C}$ is the same as deciding whether $\langle M, x \rangle \in H_{\text{TM}}$!

Rice's Theorem

Given M and x, construct M_0 such that:

- If M accepts x, then $L(M_0) \in \mathcal{C}$.
- If M does not accept x, then $L(M_0) \notin \mathcal{C}$.

On input y:

1. Run M on input x.

2. If M halts, then run M_L on y.

   1. if M_L accepts, *accept*, and

   2. if M_L rejects, *reject*.

Recall that M_L accepts some language in $\mathcal{C}$.

Rice's Theorem

Theorem: If $\mathcal{C}$ is a proper non-empty subset of the set of enumerable languages, then it is undecidable whether for a given TM M, $L(M) \in \mathcal{C}$.

Assume that TM R decides $\mathcal{C}$.

Construct S that decides H_{TM}.

On input $\langle M, x \rangle$,

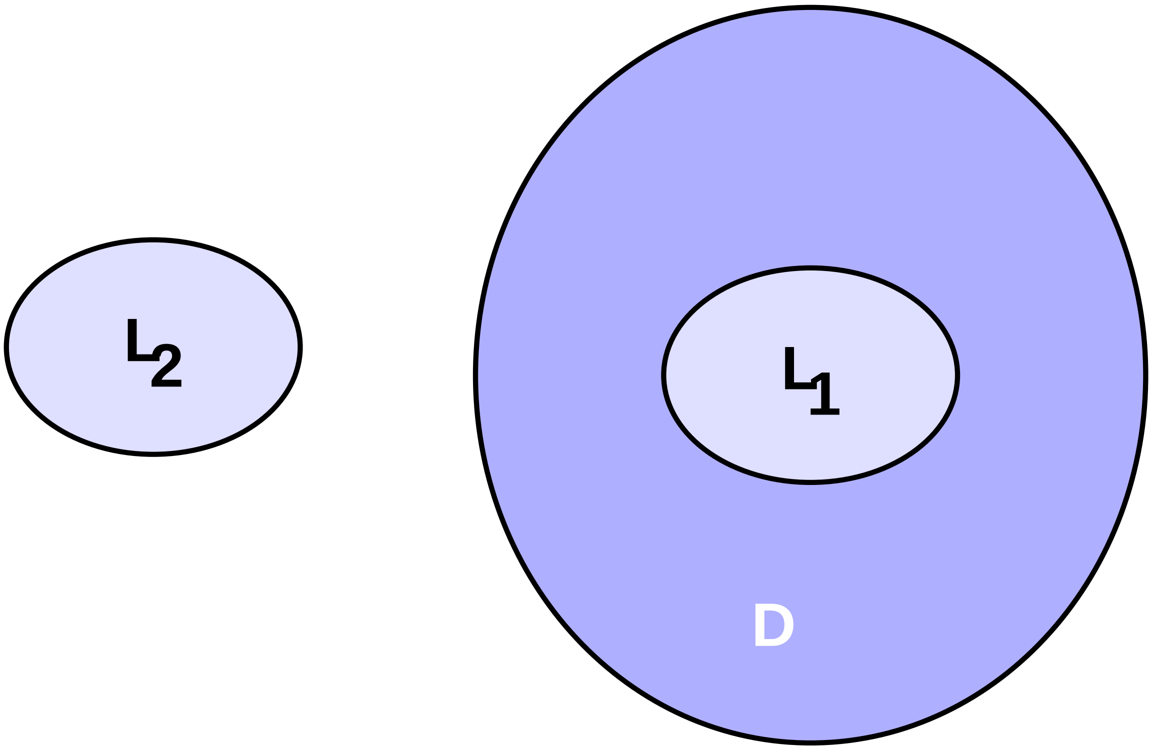
1. Construct M_0 from M and x, as above.
2. Run R on input M_0.
3. If R accepts, *accept*, if R rejects, *reject*.

Recursive Inseparability

Two disjoint languages L_1 and L_2 are *recursively inseparable* if there is no decidable language D such that

- $L_1 \cap D = \emptyset$, and
- $L_2 \subset D$.

Example of recursively separable languages:



Recursive Inseparability

A_{TM} and $\overline{A_{\mathsf{TM}}}$ are a trivial example.

Why?

Are there non-trivial examples?

Recursive Inseparability

Define

$$A_{\text{yes}} = \{\langle M\rangle | M \text{ is a TM that accepts } \langle M\rangle\}$$

and

$$A_{\text{no}} = \{\langle M\rangle | M \text{ is a TM that halts and rejects } \langle M\rangle\}$$

Theorem: A_{yes} and A_{no} are recursively inseparable.

Recursive Inseparability

Theorem: A_{yes} and A_{no} are recursively inseparable.

Proof: By contradiction.

- Let D be a decidable language that separates them.
- Assume $A_{\text{no}} \subset D$ and $D \cap A_{\text{yes}} = \emptyset$.
- Let M_D be the TM that decides D
- What does M_D do with input $\langle M_D \rangle$?
- It must halt. Why?
- If M_D accepts $\langle M_D \rangle$:
  - $\langle M_D \rangle \in A_{\text{yes}}$
  - $\langle M_D \rangle \notin D$
  - so M_D rejects $\langle M_D \rangle$.
- If M_D rejects $\langle M_D \rangle$:
  - $\langle M_D \rangle \in A_{\text{no}}$
  - $\langle M_D \rangle \in D$
  - so M_D accepts $\langle M_D \rangle$.

Recursive Inseparability

Define

$$B_{\text{yes}} = \{\langle M\rangle | M \text{ is a TM that accepts } \varepsilon\}$$

and

$$B_{\text{no}} = \{\langle M\rangle | M \text{ is a TM that halts and rejects } \varepsilon\}$$

Theorem: B_{yes} and B_{no} are recursively inseparable.

Proof by reduction and contradiction.

Recursive Inseparability

Theorem: B_{yes} and B_{no} are recursively inseparable.

By reduction and contradiction.

- Assume B_{yes} and B_{no} can be separated by E, decided by TM M_E.
- For TM M, define M': On any input,
  1. run M on input $\langle M \rangle$.
  2. if M accepts, *accept*; if M rejects, *reject*;
- Define N: On input M,
  1. construct description of M'.
  2. run M_E on $\langle M' \rangle$.
  3. if M_E accepts, *accept*; if M_E rejects, *reject*;
- Claim:
  - N is a decider. Why?
  - So N decides a language D.
  - D separates A_{yes} and A_{no}.