CS 611: Theory of Computation

Hongmin Li

Department of Computer Science
California State University, East Bay

Review: Decision Problems and Languages

- A decision problem requires checking if an input (string) has some property.

Review: Decision Problems and Languages

- A decision problem requires checking if an input (string) has some property. Thus, a decision problem is a function from `strings` to `boolean`.

Review: Decision Problems and Languages

- A decision problem requires checking if an input (string) has some property. Thus, a decision problem is a function from `strings` to `boolean`.
- A decision problem is represented as a formal language consisting of those strings (inputs) on which the answer is "yes".

Review: Recursive Enumerability

- A Turing Machine on an input w either (halts and) accepts, or (halts and) rejects, or never halts.

Review: Recursive Enumerability

- A Turing Machine on an input w either (halts and) accepts, or (halts and) rejects, or never halts.
- The language of a Turing Machine M, denoted as $L(M)$, is the set of all strings w on which M accepts.

Review: Recursive Enumerability

- A Turing Machine on an input w either (halts and) accepts, or (halts and) rejects, or never halts.
- The language of a Turing Machine M, denoted as $L(M)$, is the set of all strings w on which M accepts.
- A language L is recursively enumerable/Turing recognizable if there is a Turing Machine M such that $L(M) = L$.

Review: Decidability

- A language L is decidable if there is a Turing machine M such that $L(M) = L$ and M halts on every input.

Today:

Review: Decidability

- A language L is decidable if there is a Turing machine M such that $L(M) = L$ and M halts on every input.
- Thus, if L is decidable then L is recursively enumerable.

Today:

Review: Decidability

- A language L is decidable if there is a Turing machine M such that $L(M) = L$ and M halts on every input.
- Thus, if L is decidable then L is recursively enumerable.
- We have seen decision procedures for automata and grammars: $A_{DFA}, A_{NFA}, EQ_{DFA}, A_{CFG}, E_{CFG}$ are decidable; but $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ is Turing-recognizable.

Today:

Review: Decidability

- A language L is decidable if there is a Turing machine M such that $L(M) = L$ and M halts on every input.
- Thus, if L is decidable then L is recursively enumerable.
- We have seen decision procedures for automata and grammars: $A_{DFA}, A_{NFA}, EQ_{DFA}, A_{CFG}, E_{CFG}$ are decidable; but $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ is Turing-recognizable.

Today:

- A_{TM} is undecidable

Review: Decidability

- A language L is decidable if there is a Turing machine M such that $L(M) = L$ and M halts on every input.
- Thus, if L is decidable then L is recursively enumerable.
- We have seen decision procedures for automata and grammars: $A_{DFA}, A_{NFA}, EQ_{DFA}, A_{CFG}, E_{CFG}$ are decidable; but $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ is Turing-recognizable.

Today:

- A_{TM} is undecidable
- The diagonalization method

Review: Decidability

- A language L is decidable if there is a Turing machine M such that $L(M) = L$ and M halts on every input.
- Thus, if L is decidable then L is recursively enumerable.
- We have seen decision procedures for automata and grammars: $A_{DFA}, A_{NFA}, EQ_{DFA}, A_{CFG}, E_{CFG}$ are decidable; but $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ is Turing-recognizable.

Today:

- A_{TM} is undecidable
- The diagonalization method
- $\overline{A_{TM}}$ is Turing-unrecognizable

Review: Decidability

- A language L is decidable if there is a Turing machine M such that $L(M) = L$ and M halts on every input.
- Thus, if L is decidable then L is recursively enumerable.
- We have seen decision procedures for automata and grammars: $A_{DFA}, A_{NFA}, EQ_{DFA}, A_{CFG}, E_{CFG}$ are decidable; but $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ is Turing-recognizable.

Today:

- A_{TM} is undecidable
- The diagonalization method
- $\overline{A_{TM}}$ is Turing-unrecognizable
- The reducibility method

Review: Decidability

- A language L is decidable if there is a Turing machine M such that $L(M) = L$ and M halts on every input.
- Thus, if L is decidable then L is recursively enumerable.
- We have seen decision procedures for automata and grammars: $A_{DFA}, A_{NFA}, EQ_{DFA}, A_{CFG}, E_{CFG}$ are decidable; but $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ is Turing-recognizable.

Today:

- A_{TM} is undecidable
- The diagonalization method
- $\overline{A_{TM}}$ is Turing-unrecognizable
- The reducibility method
- Other undecidable languages

Undecidability

Definition

A language L is undecidable if L is not decidable.

Undecidability

Definition

A language L is undecidable if L is not decidable. Thus, there is no Turing machine M that halts on every input and $L(M) = L$.

Undecidability

Definition

A language L is undecidable if L is not decidable. Thus, there is no Turing machine M that halts on every input and $L(M) = L$.

- This means that either L is not recursively enumerable. That is there is no turing machine M such that $L(M) = L$, or

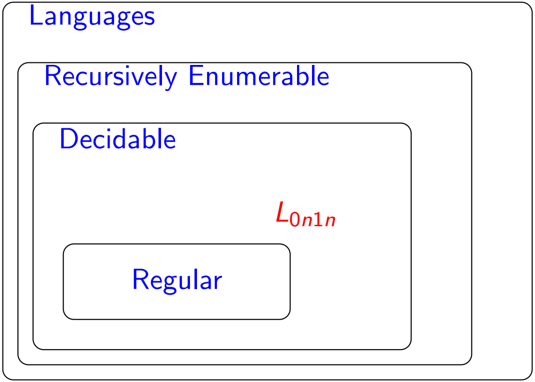
Undecidability

Definition

A language L is undecidable if L is not decidable. Thus, there is no Turing machine M that halts on every input and $L(M) = L$.

- This means that either L is not recursively enumerable. That is there is no turing machine M such that $L(M) = L$, or
- L is recursively enumerable but not decidable. That is, any Turing machine M such that $L(M) = L$, M does not halt on some inputs.

Big Picture



Relationship between classes of Languages

Diagonalization

Recall: Acceptance Problem for TMs
$A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$
Today's Theorem: A_{TM} is not decidable.
Proof uses the diagonalization method, let's briefly introduce that.

- Start with question: How to compare the relative sizes of infinite sets?
- Cantor (1890s) had the following idea:
  Defn: Say that set A and B have the same size if there is a one-to-one and onto function $f : A \rightarrow B$.
- Informally, two sets have the same size if we can pair up their members.
- This definition works for finite sets. Apply it to infinite sets too.

Diagonalization

Defn: A set is countable if it is finite or it has the same size as natural number set N.

Let $N = \{1, 2, 3, \dots\}$, and $Z = \{\dots, -2, -1, 0, 1, 2, \dots\}$

Let $Q^+ = \{\frac{m}{n} | m, n \in N\}$

Show N and Q^+ have the same size

- Integer number set Z is countable;
- Real number set R is uncountable;
- All languages set $\mathcal{L}$ is uncountable;

$\mathbb{Q}^+$	1	2	3	4	...
1	1/1	1/2	1/3	1/4	
2	2/1	2/2	2/3	2/4	...
3	3/1	3/2	3/3	3/4	
4	4/1	4/2	4/3	4/4	
⋮		⋮			

$\mathbb{N} \to \mathbb{Q}^+$

n	$f(n)$
1	1/1
2	2/1
3	1/2
4	3/1
5	3/2
6	2/3
7	1/3
⋮	⋮

$\mathbb{N} \to \mathbb{Z}$

n	$f(n)$
1	0
2	-1
3	1
4	-2
5	2
6	-3
7	3
⋮	⋮

Diagonalization

Defn: A set is countable if it is finite or it has the same size as natural number set N.

- Real number set R is uncountable;

Let $\mathbb{R}$ = all real numbers (expressible by infinite decimal expansion)

Theorem: $\mathbb{R}$ is uncountable

Proof by contradiction via diagonalization: Assume $\mathbb{R}$ is countable

So there is a 1-1 correspondence $f : \mathbb{N} \to \mathbb{R}$

n	$f(n)$
1	2.718281828…
2	3.141592653…
3	0.000000000…
4	1.414213562…
5	0.142857242…
6	0.207879576…
7	1.234567890…
⋮	⋮

Diagonalization

Demonstrate a number $x \in \mathbb{R}$ that is missing from the list.

$$x = 0.\overset{\neq 7}{8}\overset{\neq 4}{5}\overset{\neq 0}{1}\overset{\neq 2}{6}\overset{\neq 5}{1}\overset{\neq 9}{8}\overset{\neq 8}{2}\ldots$$

differs from the n^{th} number in the n^{th} digit
so cannot be the n^{th} number for any n.

Hence x is not paired with any n. It is missing from the list.

Therefore f is not a 1-1 correspondence.

Diagonalization

Defn: A set is countable if it is finite or it has the same size as natural number set N.

- Real number set R is uncountable;

Let $\mathbb{R}$ = all real numbers (expressible by infinite decimal expansion)

Theorem: $\mathbb{R}$ is uncountable

Proof by contradiction via diagonalization: Assume $\mathbb{R}$ is countable

So there is a 1-1 correspondence $f : \mathbb{N} \to \mathbb{R}$

n	$f(n)$
1	2.718281828…
2	3.141592653…
3	0.000000000…
4	1.414213562…
5	0.142857242…
6	0.207879576…
7	1.234567890…
⋮	⋮

Diagonalization

Demonstrate a number $x \in \mathbb{R}$ that is missing from the list.

$\neq 7 \; \neq 4 \; \neq 0 \; \neq 2 \; \neq 5 \; \neq 9 \; \neq 8$

$$x = 0.8516182\ldots$$

differs from the n^{th} number in the n^{th} digit so cannot be the n^{th} number for any n.

Hence x is not paired with any n. It is missing from the list.

Therefore f is not a 1-1 correspondence.

Diagonalization

Defn: A set is countable if it is finite or it has the same size as natural number set N.

- All languages set $\mathcal{L}$ is uncountable;

Let $\mathcal{L}$ = all languages

Corollary 1: $\mathcal{L}$ is uncountable

Proof: There's a 1-1 correspondence from $\mathcal{L}$ to $\mathbb{R}$ so they are the same size.

Observation: $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$ is countable.

Let $\mathcal{M}$ = all Turing machines

Observation: $\mathcal{M}$ is countable.

Because $\{\langle M\rangle \mid M \text{ is a TM}\} \subseteq \Sigma^*$.

Σ^*	$\{\ \varepsilon,$	0,	1,	00,	01,	10,	11,	000,	.
$A \in \mathcal{L}$	$\{$	0,		00,	01,				.
$f(A)$	.0	1	0	1	1	0	0	0	.

Corollary 2: Some language is not decidable.

Because there are more languages than TMs.

We will show some specific language A_{TM} is not decidable.

The Universal Language A_{TM} not decidable

Proof.

Suppose (for contradiction) A_{TM} is decidable. Then there is a TM H that decides A_{TM}, which means it always halts and $L(H) = A_{\mathrm{TM}}$.

So H on $\langle M, w\rangle = \begin{cases} \text{Accept, if M accept w} \\ \text{Reject, if not} \end{cases}$

Use H to construct TM D as follows:

D = On input $\langle M\rangle$
    Run H on input $\langle M, \langle M\rangle\rangle$
    Output ''yes'' if H rejects (M rejects $\langle M\rangle$)
    Output ''no'' if H accepts (M accepts $\langle M\rangle$)

D accepts $\langle M\rangle$ iff M doesn't accept $\langle M\rangle$.
D accepts D iff D doesn't accept $\langle D\rangle$. Contradiction. $\square$

Why this proof a Diagonalization?

Proof (contd).

All TM Descriptions $\longrightarrow$

TMs $\downarrow$

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\langle M_j \rangle$	$\cdots$
M_1	N	N	N	N	N	N	N	
M_2	N	N	N	N	N	N	N	
M_3	Y	N	Y	N	Y	Y	Y	
M_4	N	Y	N	Y	Y	N	N	
$\cdots$	N	Y	N	Y	Y	N	N	
D	N	N	Y	N	Y	?	Y	

$\square$

Why this proof a Diagonalization?

Proof (contd).

All TM Descriptions $\longrightarrow$

TMs $\downarrow$

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\langle M_j \rangle$	$\cdots$
M_1	N	N	N	N	N	N	N	
M_2	N	N	N	N	N	N	N	
M_3	Y	N	Y	N	Y	Y	Y	
M_4	N	Y	N	Y	Y	N	N	
$\cdots$	N	Y	N	Y	Y	N	N	
D	N	N	Y	N	Y	?	Y	

□

Why this proof a Diagonalization?

Proof (contd).

All TM Descriptions $\longrightarrow$

TMs $\downarrow$

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\langle M_j \rangle$	$\cdots$
M_1	$\boxed{\text{N}}$	N	N	N	N	N	N	
M_2	N	$\boxed{\text{N}}$	N	N	N	N	N	
M_3	Y	N	$\boxed{\text{Y}}$	N	Y	Y	Y	
M_4	N	Y	N	$\boxed{\text{Y}}$	Y	N	N	
$\cdots$	N	Y	N	Y	$\boxed{\text{Y}}$	N	N	
D	N	N	Y	N	Y	$\boxed{?}$	Y	

□

The Universal Language A_{TM} not decidable

Proof.

Suppose (for contradiction) A_{TM} is decidable. Then there is a TM H that decides A_{TM}, which means it always halts and $L(H) = A_{\mathrm{TM}}$.

So H on $\langle M, w\rangle = \begin{cases} \text{Accept, if M accept w} \\ \text{Reject, if not} \end{cases}$

Use H to construct TM D as follows:

D = On input $\langle M\rangle$
    Run H on input $\langle M, \langle M\rangle\rangle$
    Output ''yes'' if H rejects (M rejects $\langle M\rangle$)
    Output ''no'' if H accepts (M accepts $\langle M\rangle$)

The Universal Language A_{TM} not decidable

Proof.

Suppose (for contradiction) A_{TM} is decidable. Then there is a TM H that decides A_{TM}, which means it always halts and $L(H) = A_{\mathrm{TM}}$.

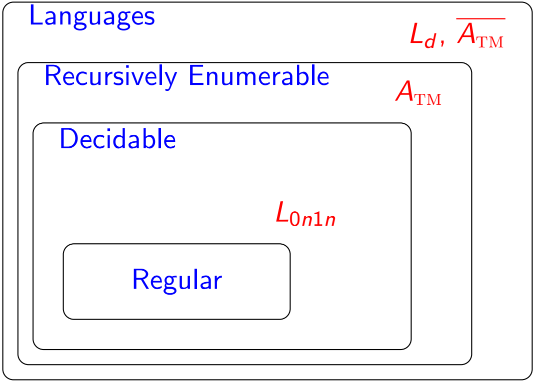
So H on $\langle M, w\rangle = \begin{cases} \text{Accept, if M accept w} \\ \text{Reject, if not} \end{cases}$

Use H to construct TM D as follows:

D = On input $\langle M\rangle$
    Run H on input $\langle M, \langle M\rangle\rangle$
    Output ''yes'' if H rejects (M rejects $\langle M\rangle$)
    Output ''no'' if H accepts (M accepts $\langle M\rangle$)

Observe that $L(D) = L_d = \{M \mid M \notin L(M)\}$!, L_d is not r.e. $\square$

A more complete Big Picture



Reductions

A reduction is a way of converting one problem into another problem such that a solution to the second problem can be used to solve the first problem. We say the first problem reduces to the second problem.

Reductions

A reduction is a way of converting one problem into another problem such that a solution to the second problem can be used to solve the first problem. We say the first problem reduces to the second problem.

- Informal Examples: Measuring the area of rectangle reduces to measuring the length of the sides

Reductions

A reduction is a way of converting one problem into another problem such that a solution to the second problem can be used to solve the first problem. We say the first problem reduces to the second problem.

- Informal Examples: Measuring the area of rectangle reduces to measuring the length of the sides; Solving a system of linear equations reduces to inverting a matrix

Reductions

A reduction is a way of converting one problem into another problem such that a solution to the second problem can be used to solve the first problem. We say the first problem reduces to the second problem.

- Informal Examples: Measuring the area of rectangle reduces to measuring the length of the sides; Solving a system of linear equations reduces to inverting a matrix
- The problem L_d reduces to the problem A_{TM} as follows: "To see if $w \in L_d$ check if $\langle w, w \rangle \in A_{\mathrm{TM}}$."

Undecidability using Reductions

Proposition

Suppose L_1 reduces to L_2 and L_1 is undecidable. Then L_2 is undecidable.

Undecidability using Reductions

Proposition

Suppose L_1 reduces to L_2 and L_1 is undecidable. Then L_2 is undecidable.

Proof Sketch.

Suppose for contradiction L_2 is decidable. Then there is a M that always halts and decides L_2. Then the following algorithm decides L_1

Undecidability using Reductions

Proposition

Suppose L_1 reduces to L_2 and L_1 is undecidable. Then L_2 is undecidable.

Proof Sketch.

Suppose for contradiction L_2 is decidable. Then there is a M that always halts and decides L_2. Then the following algorithm decides L_1

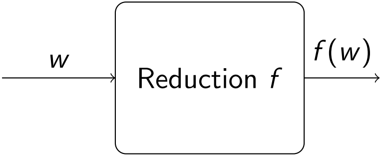
- On input w, apply reduction to transform w into an input w' for problem 2
- Run M on w', and use its answer.

Schematic View

$\xrightarrow{w}$

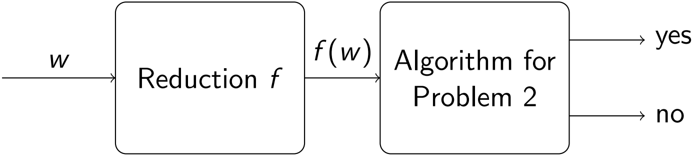
Reductions schematically

Schematic View



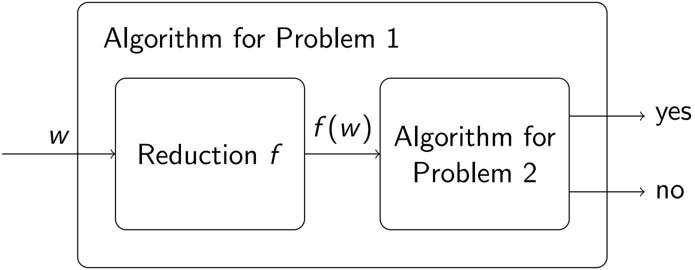
Reductions schematically

Schematic View



Reductions schematically

Schematic View



Reductions schematically

The Halting Problem

Use our knowledge that A_{TM} is undecidable to show other problems are undecidable.

Proposition

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

Proof.

Proof by contradiction, showing that A_{TM} is reducible to : Assume that HALT is decidable and show that A_{TM} is decidable (false!). Let TM R decide HALT. Construct TM S deciding A_{TM}.

```
S = On input ⟨M, w⟩
Use R to test if M on w halts.  If not, reject.
Simulate M on w until it halts (as guaranteed by R).
If M has accepted then accept.
```

Proposition

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

Proof.

Proof by contradiction, showing that A_{TM} is reducible to HALT: Assume that HALT is decidable and show that A_{TM} is decidable (false!).
Let TM R decide HALT. Construct TM S deciding A_{TM}.

```
S = On input ⟨M, w⟩
Use R to test if M on w halts.  If not, reject.
Simulate M on w until it halts (as guaranteed by R).
If M has accepted then accept.
If M has rejected then reject.
```

TM S decides A_{TM}, a contradiction. Therefore HALT is undecidable.

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 $A \leq_m B$.

Convention

In this course, we will drop the adjective "mapping" or "many-one", and simply talk about reductions and reducibility.

Reductions and Recursive Enumerability

Proposition

If $A \leq_m B$ and B is recursively enumerable then A is recursively enumerable.

Reductions and Recursive Enumerability

Proposition

If $A \leq_m B$ and B is recursively enumerable then A is recursively enumerable.

Proof.

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

Reductions and Recursive Enumerability

Proposition

If $A \leq_m B$ and B is recursively enumerable then A is recursively enumerable.

Proof.

Let f be the reduction from A to B and let M_B be the 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 does and reject if M_B rejects
```

□

Reductions and non-r.e.

Corollary

If $A \leq_m B$ and A is not recursively enumerable then B is not recursively enumerable.

Reductions and Decidability

Proposition

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

Reductions and Decidability

Proposition

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

Proof.

Let M_B be the Turing machine deciding B and let f be the reduction. Then the algorithm deciding A, on input w, computes $f(w)$ and runs M_B on $f(w)$. $\square$

Reductions and Decidability

Proposition

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

Proof.

Let M_B be the Turing machine deciding B and let f be the reduction. Then the algorithm deciding A, on input w, computes $f(w)$ and runs M_B on $f(w)$. $\square$

Corollary

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