Computability

Xin Li

lixints@cs.jhu.edu
Topics

• Universal Turing Machine

• Diagonalization and computability

• Reductions
TM vs General Computer

• So far all the TMs we have seen are special purpose (i.e., special algorithms)

• A modern computer is quite general and can handle different tasks.

• How does that compare to a TM?

• Answer: Universal TM.
Universal TM

- A TM can also be represented by a (binary) string.

- Thus a TM can also be used as an input/data to another TM.

- This blurs the distinction between software, hardware and data.

- Brings up the idea of a universal TM (first observed by Turing himself).
Universal TMs

There exists one universal TM that can simulate any other TM.
Universal TMs

- **Thm:** There exists a TM $U$ s.t. $\forall x, \alpha \in \{0, 1\}^*, U(x, \alpha) = M_\alpha(x)$, where $\alpha$ represents the TM $M_\alpha$. Moreover, if $M_\alpha$ halts on $x$ with $T$ steps, then $U(x, \alpha)$ halts within $C T \log T$ steps. $C$ is a constant that only depends on $M_\alpha$.

- **Proof with $C T^2$ time.**

- **First assume** $M$ has a single tape and uses input alphabet $\{0, 1\}$.

- **By previous theorems, the running time becomes $C' T^2$.**
Universal TMs

- U will use 5 tapes.

- Simulate one step of M:
  - Read current state and symbol, apply M’s transition function, implement the changes. This takes constant time.

- So total simulation time $C T^2$
Recognizing vs Deciding

- Say a TM $M$ recognizes the language $L$ if the following holds:
  
  \[ w \in L \iff M \text{ accepts } w \]

- However, $M$ may not halt on some inputs, e.g., if $w \notin L$.

- Prefer TMs that halt on all inputs, such TMs are called deciders.

- If $M$ recognizes the language $L$ and $M$ is a decider, say $M$ **decides** $L$.

  \[ w \in L : M \text{ accepts}; \ w \notin L : M \text{ rejects} \]
Decidable languages
Recognizable languages?
All languages?
Question

• Are there Boolean functions that cannot be computed by any TM?

• Are there languages that cannot be decided/recognized by any TM?

• Are there languages that can be recognized but cannot be decided by any TM?
A Detour to Set Theory

- Countable set vs uncountable set

- Say a set $S$ is countable if it is finite or there is a bijection between $N$ and $S$.

- Otherwise, say $S$ is uncountable.
Examples

N (the set of natural numbers): countable

Z (the set of integers): countable

Q (the set of rational numbers): countable

R (the set of real numbers): uncountable
Breakout Question

• Is the set of all TMs countable?

• Is the set of all (binary) languages countable?
Conclusion

• *Is the set of all TMs countable? Yes*

• *Is the set of all (binary) languages countable? No*

• *There exists a language that cannot be decided/recognized by any TM.*
Topics

• Computability

• Reductions

• Complexity
A specific example

• Assumption: any binary string represents a TM (map invalid encoding to a trivial TM, e.g., one that rejects all inputs).

• Define the Boolean function $UC: \{0,1\}^* \rightarrow \{0,1\}$ as follows.

  For any input $\alpha$, if $M_\alpha(\alpha)=1$ (accepts), then $UC(\alpha)=0$ (rejects).

  Otherwise, if $M_\alpha(\alpha)=0$ (rejects or loop forever), then $UC(\alpha)=1$ (accepts).

$M_\alpha$ is the TM represented by string $\alpha$. 
Claim

• UC is uncomputable/the language is undecidable.

• Proof: assume FSOC that a TM M computes UC, let [M] be its binary representation.

• Consider UC([M]) and M([M]).

  By assumption, we have UC([M])=M([M]).

  By construction, we have UC([M])≠M([M]).
Another example

• The language $\text{HALT} = \{<M, \alpha> \mid M \text{ halts on } \alpha\}$, i.e., deciding if a TM $M$ halts on input $\alpha$.

• Claim: HALT is undecidable.

• Proof: assume FSOC that a TM $M_{\text{HALT}}$ that decides HALT, construct another TM $M_{\text{UC}}$ that computes UC.
Construction of $M_{UC}$ using $M_{HALT}$

- Idea: to compute UC, can run $M_\alpha$ on $\alpha$ and flip the output.

- What is the problem?
  $M_\alpha$ may not halt on $\alpha$

- Use $M_{HALT}$ to check that.

- $M_{UC}$: run $M_{HALT}(\alpha, \alpha)$. If output 0 (does not halt), $M_{UC}$ outputs 1. Otherwise (halt) run $M_\alpha$ on $\alpha$ and flip the output
Another example

• The language $A_{TM}=\{<M, \alpha> \mid TM \ M \ accepts \ \alpha\}$. 

• Claim: $A_{TM}$ is undecidable.

• Proof: assume FSOC that a TM $M_A$ that decides $A_{TM}$, construct another TM $M_{UC}$ that computes UC.

  $M_{UC}$: On input $\alpha$, run $M_A(\alpha, \alpha)$. If $M_A$ outputs 1, $M_{UC}$ outputs 0; If $M_A$ outputs 0, $M_{UC}$ outputs 1.
A close look

$M_{UC}$: On input $\alpha$, run $M_A(\alpha, \alpha)$. If $M_A$ outputs 1, $M_{UC}$ outputs 0; If $M_A$ outputs 0, $M_{UC}$ outputs 1.

What are we doing here?

We constructed a function $f$ that maps the inputs to UC to the inputs to $A_{TM}$: $f(\alpha)=<\alpha, \alpha>$

Property: $UC(\alpha)=1$ iff $A_{TM}(<\alpha, \alpha>)=0$.

Such a function is called a reduction.
Reduction

• Intuitively, given two problems A and B, we reduce A to B so that to solve A, we can first solve B and use the solution to solve A.

• Examples in real life: reduce calculation to building a calculator, reduce traveling to building buses, ships, airplanes…

• Examples in mathematics: reduce solving polynomial equations to finding the formula for the roots of the equations.
Mapping Reduction

• Say a language $A$ is mapping reducible to a language $B$, written $A \leq_m B$, if

  \[ \exists \text{ a computable function } f : \Sigma^* \rightarrow \Sigma^* \text{ s.t. } \forall w, w \in A \iff f(w) \in B \]

if and only if

never cross the boundary
Properties of mapping reduction

- If $A \leq_m B$ then

- If $B$ is decidable, then $A$ is also decidable.

- If $A$ is undecidable, then $B$ is also undecidable.

- To prove a language $B$ is undecidable, find a known undecidable language $A$ and show $A \leq_m B$. 
Example

• Define the language $E_{TM}=\{<M> \mid TM \text{ M accepts nothing}\}$.

• Claim: $E_{TM}$ is undecidable.

• Proof: we pick $A_{TM}$ and show that $A_{TM} \leq_{m} E_{TM}$ or $A_{TM} \leq_{m}$ the complement of $E_{TM}$
More Examples

• Define the language $EQ_{TM} = \{<M_1, M_2> | M_1$ and $M_2$ are TMs and $L(M_1) = L(M_2)\}$.

• Claim: $EQ_{TM}$ is undecidable.

• Proof: we pick $E_{TM}$ and show that $E_{TM} \leq_m EQ_{TM}$.

  Given input $<M>$ to $E_{TM}$, map it to a pair of input $<M, M'>$ to $EQ_{TM}$.
  Define $M'$ to be the TM that rejects everything.
  Thus, $<M> \in E_{TM}$ iff $<M, M'> \in EQ_{TM}$.
More Examples

• There are many other undecidable languages.

• E.g., Hilbert’s 10th problem (posed at ICM 1900)

• Given a Diophantine Equation (a polynomial equation with integer coefficients and a finite number of variables), decide if it has an integer solution.
Looking Back

- There are unrecognizable/undecidable languages.

- Recognizable but undecidable?

- \( \text{HALT}=\{<M, \alpha> \mid M \text{ halts on } \alpha \}, \) i.e., deciding if a TM \( M \) halts on input \( \alpha \).

- \( A_{TM}=\{<M, \alpha> \mid TM \text{ M accepts } \alpha \} \).
Specific Examples of Unrecognizable Languages

• **Thm:** A language $A$ is decidable iff both $A$ and its complement are recognizable.

• **Easy direction:** $A$ is decidable $\Rightarrow$ both $A$ and its complement are recognizable.

• **Harder direction:** both $A$ and its complement are recognizable $\Rightarrow$ $A$ is decidable.
Proof

• Suppose A is recognized by TM $M_1$ and its complement is recognized by TM $M_2$.

• Build another TM $M$ that decides A.

• Idea: run input $w$ on $M_1$, if accepts, A accepts. Otherwise run $M$ on $M_2$, if accepts, A rejects.

• Problem: may not halt on $w$.

• Solution: run $M_1$ and $M_2$ on $w$ in parallel until one halts.
Implication:

• We know HALT and $A_{TM}$ are undecidable but recognizable.

• So their complements are unrecognizable.

• Remark: the argument of diagonalization works as long as the machine can be represented as a finite string.

• There will always be unrecognizable/undecidable languages even if more powerful machines appear in the future.