Undecidability via Reductions

office hours:

Me: MWF@3
Ni: TR@4

Image via Flickr user visitfingerlakes,

Recap:

- Language L is <u>Turing-recognizable</u> if there exists Turing machine M:
  - $x \in L \Rightarrow$ M accepts x
  - $x \notin L \Rightarrow$ M doesn't accept x (rejects or runs forever)

- L is <u>Turing-decidable</u> if there exists Turing machine M:
  - $x \in L \Rightarrow$ M accepts x
  - $x \notin L \Rightarrow$ M rejects x

  } halt on all inputs

Theorem (Turing 1936)

- L_{acc} = { <M,x> | M is a TM that accepts x } is **recognizable** but not **decidable**

  – Recognizable? ⇒ easy

  just run M on x, see what it does (simulation of M may not finish)

  – Undecidable? ⇒ diagonalization

Reduction Example:

- L_{halt} = { <M,x> | M is a TM that halts on x } is undecidable
  - Can do diagonalization again (traumatic)
  - **Better:** use the fact that we already proved L_{acc} undecidable

Idea: Proof by contradiction

Suppose L_{halt} is decidable

⇒ there is an algo that decides L_{halt}

☆ [I claim we can use this algo as subroutine to decide L_{acc}] ☆

That would be contradiction since L_{acc} known undecidable

$L_{acc} = \{$ - - - M accepts x $\}$

L_{halt} = { <M,x> | M is a TM that halts on x } is undecidable

Goal: write algo that decides L_{acc}, using (hypothetical) subroutine for L_{halt}.

MyAlgo

on input $\langle M,x \rangle$:

// want to know, does M accept x?

call subroutine on $\langle M,x \rangle$

// tells me whether M halts on x

if answer = yes

  run M on x

  if M accepts, say yes

    else say no

else say no

ANALYSIS:

$\langle M,x \rangle \in L_{acc} \Leftrightarrow M$ accepts x

$\Rightarrow$ M halts on x

$\Rightarrow$ subroutine says yes, MyAlgo will run M on x, observe that it accepts

$\Rightarrow$ MyAlgo says yes

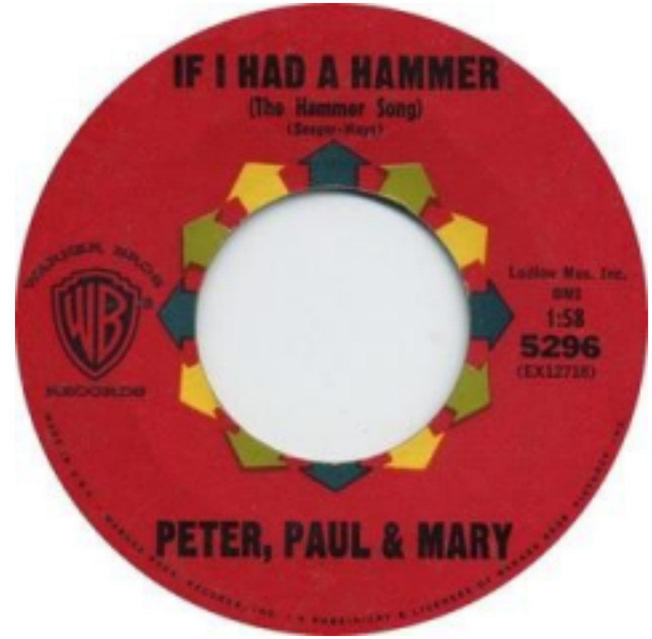
$\langle M,x \rangle \notin L_{acc} \Leftrightarrow M$ doesn't acc x

(2 cases) $\Rightarrow$ MyAlgo says no

Reductions: the heart of CS517

- If I had a subroutine for this problem …

  … I'd be able to solve this other problem



- If I had a subroutine for L_{halt}, I could use it to solve L_{acc}
  - But L_{acc} is undecidable, so subroutine for L_{halt} is impossible!

"A reduces to B"

Reductions: the heart of CS517

A, B are decision problems

- **Definition:** A ≤ B means: "A can be solved using a (hypothetical) subroutine for B"
- **Properties:**

  Ex: $L_{acc} \leq L_{halt}$

  – If A ≤ B and B decidable then A decidable, too
  – If A ≤ B and A **un**decidable then B **un**decidable
- Interpretation of A **≤** B:
  – "[solving] B is at least as hard as [solving] A"
  – "[solving] A is no harder than [solving] B"

Reductions Recipe

- Suppose you want to show that some language/problem L is undecidable

show known undecidable $\leq$ problem you want to show undecidable

---

write algo that solves known undecidable problem, using subroutine for new problem

(Much) Harder example

$L(M) = \{x \mid M \text{ accepts } x\}$

- L_{empty} = { <M> | M is a TM and L(M) = ∅ } is undecidable

"given M, is it true that M doesn't accept any x?"

Want to show: $L_{acc} \leq L_{empty}$

Want to show algorithm that solves L_{acc} using subroutine for L_{empty}

source code of M^* hard-codes M & x

(Much) Harder example

info about global behavior of M

- L_{empty} = { <M> | M is a TM and $L(M) = \varnothing$ } is undecidable

MyAlgo:

on input $\langle M, x \rangle$:

// want to know: does M accept x?

write down, but don't execute

M^*: on input z: ignore z, run M on x

call subroutine on M^*

return opposite of subroutine result

Idea: write down source code of M^*, designed so that

M accepts $x \Rightarrow L(M^*) \neq \varnothing$

M doesn't accept $x \Rightarrow L(M^*) = \varnothing$

$\underbrace{M \text{ accepts } x \,/\, M \text{ doesn't accept } x}_{\text{what I care about}} \qquad \underbrace{L(M^*) \neq \varnothing \,/\, L(M^*) = \varnothing}_{\text{what subroutine can tell me}}$

(Much) Harder example

- L_{empty} = { <M> | M is a TM and L(M) = ∅ } is undecidable

MyAlgo:

on input ⟨M,x⟩:

// want to know: does M accept x?

write down, but don't execute

M*:
on input z:
ignore z, run M on x

call subroutine on M*

return opposite of subroutine result

Analysis:

If M accepts x, then M* accepts all strings

⇒ $L(M^*) = \{0,1\}^* \neq \emptyset$

⇒ subroutine says no

⇒ MyAlgo says yes

If M doesn't accept x, then M* doesn't accept anything

⇒ $L(M^*) = \emptyset$

⇒ MyAlgo says no