Rice's theorem

“<u>every</u> question about the [external] behavior of Turing machines is undecidable”

Rice's theorem

“There is no hope in trying to understand [arbitrary] computer programs”

Does this program halt? Does this program's behavior match the specification? Do these two programs agree on all inputs?

Notation

TM

- L(M) = "the language recognized by M"
  = "the decision problem that M solves"

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

Undecidability Reductions involving TMs

- Show that the following language undecidable:

  $L_{nonempty}$ = { <M> | M is a TM and L(M) ≠ ∅ }

- Recipe: show $L_{accept} \leq L_{nonempty}$
  - Write an algorithm that decides L_{accept}, using a (hypothetical) subroutine for $L_{nonempty}$

$L_{nonempty} \leq L_{accept}$?

showing $L_{accept} \leq L_{nonempty}$

$L_{nonempty} = \{ \langle M \rangle \mid L(M) \neq \emptyset \}$

Algo for L_{accept}

hard-coded constants

- On input <M',x'>: *// "does M' accept x'?"*
  - Write down (but **don't execute**) source code:

    M* = "$\boxed{\text{On input z: ignore z, just run } \underline{\text{M'}} \text{ on } \underline{\text{x'}}}$"

    string manipulation
  - If <M*> ∈ $L_{nonempty}$ then say yes, else say no

    call hypothetical subroutine for $L_{nonempty}$

If M' accepts x'
then M* accepts all inputs
⇒ subroutine says yes
⇒ we say yes

If M' doesn't accept x'
then M* accepts <u>no</u> inputs
⇒ we say <u>no</u>

Be careful:

- (Unnamed) reduction algorithm: it solves L_{accept}, taking advantage of a subroutine for $L_{nonempty}$
- <M',x'>: a generic instance of L_{accept}, and input to our (unnamed) reduction algorithm
- M*: a TM whose source code is generated on the fly by our reduction algorithm
  - Source code of M* depends on M' and x'
  - Hence the **behavior** of M* depends on M' and x'
- z: generic input to M*

L_{finite} = { <M> | L(M) finite } is undecidable

Solves L_{accept}

reduction Algo:

- On input <M',x'>   *// "does M' accept x'?"*
  - Write down (but **don't execute**) source code:

    M* = " $\boxed{\text{On input z : ignore z, run M' on x'}}$ "   hard-coded in M* from input

  - If <M*> ∈ L_{finite} then say _no_, else say _yes_

hypothetical subroutine

WANT:

If M' accepts x', M* should accept infinite language

If not, M* should accept finite language

Achieved:

If M' accepts' : $L(M^*) = \{0,1\}^*$

If M' doesnt $L(M^*) = \varnothing$

$$L_{primes} = \{ \langle M \rangle \mid L(M) = PRIMES \}$$

is undecidable

- On input <M',x'>   *// "does M' accept x'?"*
  - Write down (but **don't execute**) source code:

M* = " On input z :
 If M' accepts x' AND z is prime
 accept
 else reject "

  - If <M*> ∈ L_{primes} then say yes, else say no

L(M*) = PRIMES

WANT:

If M' accepts x' ⇒ M* accepts exactly primes PRIMES

If M' doesn't ⇒ M* doesn't accept

GET:

If M' accepts x' ⇒ M* accepts only primes

If M' doesn't ⇒ M* accepts no strings

L_{primes} = { <M> | L(M) = PRIMES }

- On input <M',x'> *// "does M' accept x'?"*
  - Write down (but **don't execute**) source code:

    M* = " On input z :
    If M' accepts x' AND z is prime
    accept
    else reject "

  - If <M*> ∈ L_{primes} then say yes, else say no

If M' accepts x' :
M* equivalent to
If ~~true~~ AND z is prime
...

else:
M* never enter
if-branch

Rice's Theorem

- **Theorem:** { <M> | L(M) has property P } is undecidable if P is **nontrivial**:
  - there is an M_Y with $L(M_Y)$ having property P
  - there is an M_N with $L(M_N)$ **not** having property P

- In other words, given encoding of M:
  - Can't decide whether L(M) is empty
  - Can't decide whether some special $x \in L(M)$
  - Can't decide whether L(M) is finite
  - Can't decide anything interesting about L(M)

Rice's Theorem proof

- Show: L_P = { <M> | L(M) has property P } undecidable

  let M_N = TM that rejects everything

  wlog M_N doesn't have property P

  M_Y does have property P

- On input <M',x'> *// "does M' accept x'?"*
  - Write down (but don't execute) source code:

    M* = " On input z :
    if M' accepts x' AND M_Y accepts z
    else ~~reject~~ accept "

  - If <M*> ∈ L_P then say _____, else say _____

Rice's Theorem proof

- On input <M',x'>   *// "does M' accept x'?"*
  - Write down (but don't execute) source code:

    M* = " On input z :
    if M' accepts x' AND M_y accepts z
    accept
    else reject "

  - If <M*> ∈ L_P then say yes, else say no

Idea: If M' accepts x' ⇒ M* acts just like M_y
⇒ L(M*) has property P
⇒ we say yes

If not ⇒ M* acts just like M_N ⇒ doesn't have property P

Some technical issues about reductions

- **Properties:**
  - If A ≤ B and B decidable then A decidable, too
  - If A ≤ B and A **un**decidable then B **un**decidable

- **What about this???**
  - If A ≤ B and B recognizable then A recognizable
  - If A ≤ B and A **un**recognizable then B **un**recognizable

Some technical issues

- **Counterexample to:** If A ≤ B and B recognizable then A recognizable

Constraining the reduction

- A $\leq_T$ B, “Turing reduction”, “Cook reduction”:
  - Algorithm for A uses subroutine for B in arbitrary way
- A $\leq_m$ B, “many-one reduction”, “Karp reduction”:
  - Algorithm for A only calls B subroutine once, as a “tail call”

- **Property:** If A $\leq_m$ B and B recognizable then A recognizable

Constraining the reduction

- **Property:** If A $\leq_m$ B and B recognizable then A recognizable

- **Corollary:** $\sim L_{accept}$ is not recognizable

- **Recipe:** to show that L is not recognizable, just show $\sim L_{accept} \leq_m L$

Extended Rice's Theorem

- **Theorem:** { <M> | L(M) has property P } is not even recognizable if P is **monotone**:
  - there is an M_Y with $L(M_Y)$ having property P
  - there is an M_N with $L(M_N)$ **not** having property P
  - $L(M_Y) \subseteq L(M_N)$
- So, the following are unrecognizable:
  - Given M, is L(M) finite?
  - Given M, is L(M) a regular language?
  - Given M, is L(M) empty?

Extended Rice's Theorem

- **Main idea 1:** Use $\leq_m$ so we can say something about [un]recognizability

- **Main idea 2:** given <M,x>, design M* so that:
  - If M accepts x, then M* behaves like M_N
  - Otherwise M* behaves like M_Y
  - $M \in \sim L_{accept} \Leftrightarrow$ M doesn't accept x
    $\Leftrightarrow$ M* behaves like M_Y
    $\Leftrightarrow M^* \in L_P$