The Halting Problem

• Definition of the "Halting Problem":
  \( \text{HALT} = \{ <M, x> : \text{TM } M \text{ halts on input } x \} \)

• HALT is recursively enumerable.
  – proof?

• Is HALT decidable?

Theorem: HALT is not decidable (undecidable).

Proof:
– Suppose TM H decides HALT
  – Define new TM H': on input <M>
    • if H accepts <M, <M>> then loop
    • if H rejects <M, <M>> then halt

  – consider H' on input <H'>:
    • if it halts, then H rejects <H', <H'>>, which implies it cannot halt
    • if it loops, then H accepts <H', <H'>> which implies it must halt
  – contradiction.
So far…

• Can we exhibit a natural language that is non-RE?

RE and co-RE

**Theorem:** a language $L$ is decidable if and only if $L$ is RE and $L$ is co-RE.

Proof:

(⇒) we already know decidable implies RE
  - if $L$ is decidable, then complement of $L$ is decidable by flipping accept/reject.
  - so $L$ is in co-RE.

RE and co-RE

**Theorem:** a language $L$ is decidable if and only if $L$ is RE and $L$ is co-RE.

Proof:

(⇐) we have TM $M$ that recognizes $L$, and TM $M'$ recognizes complement of $L$.
  - on input $x$, simulate $M, M'$ in parallel
  - if $M$ accepts, accept; if $M'$ accepts, reject.

A natural non-RE language

**Theorem:** the complement of HALT is not recursively enumerable.

Proof:

- we know that HALT is RE
- suppose complement of HALT is RE
- then HALT is co-RE
- implies HALT is decidable. Contradiction.

Summary

Main point: some problems have no algorithms, HALT in particular.
Reductions

• Given a new problem NEW, want to determine if it is easy or hard
  – right now, easy typically means decidable
  – right now, hard typically means undecidable

• One option:
  – prove from scratch that the problem is decidable, or
  – prove from scratch that the problem is undecidable (dream up a diag. argument)

Reductions are one of the most important and widely used techniques in theoretical Computer Science.

• especially for proving problems “hard”
  – often difficult to do “from scratch”
  – sometimes not known how to do from scratch
  – reductions allow proof by giving an algorithm to perform the transformation

Example reduction

• How could we use procedure that decides \( A_{TM} \) to decide HALT?
  – given input to HALT: \( <M, w> \)

• Some things we can do:
  – check if \( <M, w> \in A_{TM} \)
  – construct another TM \( M' \) and check if \( <M', w> \in A_{TM} \)

Example reduction

• Try to prove undecidable:
  \( A_{TM} = \{<M, w> : M \text{ accepts input } w\} \)
  – We know this language is undecidable:
  \( HALT = \{<M, w> : M \text{ halts on input } w\} \)

• Idea:
  – suppose \( A_{TM} \) is decidable
  – show that we can use \( A_{TM} \) to decide HALT
  – conclude HALT is decidable. Contradiction.

Example reduction

• Deciding HALT using a procedure that decides \( A_{TM} \) (“reducing HALT to \( A_{TM} \”).
  – on input \( <M, w> \)
  – check if \( <M, w> \in A_{TM} \)
    • if yes, the \( M \) halts on \( w \); ACCEPT
    • if no, then \( M \) either rejects \( w \) or it loops on \( w \)
  – construct \( M' \) by swapping \( q_{accept}/q_{reject} \) in \( M \)
  – check if \( <M', w> \in A_{TM} \)
    • if yes, then \( M' \) accepts \( w \), so \( M \) rejects \( w \); ACCEPT
    • if no, then \( M \) neither accepts nor rejects \( w \); REJECT
Example reduction

- Preceding reduction proved:

**Theorem**: $A_{TM}$ is undecidable.

Proof (recap):
- suppose $A_{TM}$ is decidable
- we showed how to use $A_{TM}$ to decide HALT
- conclude HALT is decidable. Contradiction.

Another example

- Try to prove undecidable:
  \[ E_{TM} = \{<M> : L(M) = \emptyset\} \]
- which problem should we reduce from?
  - HALT = \{<M, w> : M halts on input w\}
  - $A_{TM} = \{<M, w> : M accepts input w\}$
- Some things we can do:
  - check if $<M> \in E_{TM}$
  - construct another TM $M'$ and check if $<M'> \in E_{TM}$

Another example

- We are given input $<M, w>$
- We want to use a procedure that decides $E_{TM}$ to decide if $<M, w> \in A_{TM}$

  Idea:
  - check if $<M> \in E_{TM}$
  - if not?
    - helpful if could make $M$ reject everything except possibly $w$.

Another example

- Preceding reduction proved:

  **Theorem**: $E_{TM}$ is undecidable.

  Proof (recap):
  - suppose $E_{TM}$ is decidable
  - we showed how to use $E_{TM}$ to decide $A_{TM}$
  - conclude $A_{TM}$ is decidable. Contradiction.

Example reduction

- We proved
  \[ A_{TM} = \{<M, w> : M accepts input w\} \]
  undecidable, by reduction from
  \[ HALT = \{<M, w> : M halts on input w\} \]
- We proved
  \[ E_{TM} = \{<M> : L(M) = \emptyset\} \]
  undecidable by reduction from $A_{TM}$
Definition of reduction

• Can you reduce co-HALT to HALT?

• We know that HALT is RE
• Does this show that co-HALT is RE?
  – recall, we showed co-HALT is not RE

• our current notion of reduction cannot distinguish complements