Remember $A_{TM}=\{<M,w>| M \text{ accepts } w\}$ is undecidable

- When $M$ does not accept $w$ cannot decide if its because it will eventually reject or loop

**Reduction**: converting one problem $A$ into another problem $B$, where we can use solver for $B$ to solve $A$

- Also $A$ clearly cannot be “harder” than $B$, so if $B$ is “decidable” then so is $A$.

**Standard reduction**:

- **Assume language $L$** of interest is decidable by $R$
- Show that solving $L$ means we can solve $A_{TM}$
  - By mapping any instance of $A_{TM}$ into $L$
  - Thus if $R$ exists, then we can construct a $TM$ $S$ so that $A_{TM}$ is decidable
  - But this is impossible, so no such $R$ can exist

pp. 215-227. *Undecidable Language Problems* (Sec. 5.1)
• \( \text{HALT}_{\text{TM}} = \{<M,w>| \ M \text{ is a TM that halts on } w\} \)

• (p. 216) **Theorem 5.1.** \( \text{HALT}_{\text{TM}} \) **is undecidable**
  • Proof by contradiction. Assume \( \text{HALT}_{\text{TM}} \) is decidable by \( R \)
  • Build a decider for \( A_{\text{TM}} \)
    • Given \( <M,w> \) instance from \( A_{\text{TM}} \), pass unchanged to \( R \)
    • If \( R \) finds \( M \) halts on \( w \), \( R \) halts and accepts
    • If \( R \) finds \( M \) doesn’t halt on \( w \), \( R \) halts and rejects

  ![Diagram](image.png)

  • Construct TM \( S \) to decide \( A_{\text{TM}} \) from \( R \) as follows
    • Run \( R \) on \( <M,w> \)
    • If \( R \) rejects, reject (we know \( M \) loops on \( w \))
    • If \( R \) accepts (we know \( M \) halts on \( w \)):
      • Simulate \( M \) on \( w \) until it halts
      • If \( M \) accepts \( w \) then \( S \) accepts
      • If \( M \) rejects \( w \), then \( S \) rejects
    • If \( R \) exists, then \( S \) as constructed above decides \( A_{\text{TM}} \)

  • **But \( A_{\text{TM}} \) is undecidable, so \( R \) cannot exist**
• **$E_{TM} = \{<M> | M \text{ is a TM and } L(M) = \emptyset\}$**

• (p. 217) **Theorem 5.2 $E_{TM}$ is undecidable**

  • Assume $R$ decides $E_{TM}$, i.e. given $<M>$ as input, $R$
    • accepts if $L(M)$ is empty
    • rejects if $L(M)$ is not

• Use $R$ to construct a $S$ that decides $A_{TM}$ as follows

  • Given any $<M,w>$, first convert $M$ to $M_1$ as follows
    • On any input $x$, if $x \neq w$, $M_1$ rejects
    • If $x = w$, run $M$ on $w$ and accept if $M$ does
    • Only string $M_1$ can possibly accept is $w$

  • Now define $S$ on an input $<M,w>$ as follows
    • Construct $M_1$ from $M$
    • Run $R$ on $<M_1>$ (We are assuming $R$ exists)
    • If $R$ accepts (i.e. $L(M) = \emptyset$), $S$ rejects ($w$ not in $L(M)$)
    • else if $R$ rejects ($L(M_1)$ not empty), $S$ accepts
      • $w$ accepted by $M$
    • If $R$ were decider for $E_{TM}$, then $S$ is a decider for $A_{TM}$
• (p. 218) \( \text{REGULAR}^\text{TM} = \{ <M> | M \text{ a TM & } L(M) \text{ is regular} \} \)

• **Theorem 5.3 REGULAR\(^\text{TM}\) is undecidable**
  
  • Assume REGULAR\(^\text{TM}\) is decidable by some TM R
    
    • Given some M, R accepts if \( L(M) \) is regular
    
    • R rejects if \( L(M) \) is NOT regular
  
  • Construct S from R as decider for \( A^\text{TM} = \{ <M,w> \} \) as follows
    
    • Take M from its input <\( M,w > \> and modify M to \( M_2 \) that
      
      • recognizes non-regular language \( \{ 0^n1^n | n \geq 0 \} \) if M does not accept \( w \)
      
      • recognizes regular language \( \Sigma^* \) if M accepts \( w \)
      
      • \( M_2 \) constructed ONLY for purpose of feeding its description into assumed decider R for REGULAR\(^\text{TM}\)
    
    • Run R on \( <M_2> \)
      
      • If R accepts, then \( <M_2> \) recognizes a regular language
        
        • Which means M accepts \( w \)
      
      • If R rejects, then \( M_2 \) recognizes a non-reg language
        
        • Which means that M does not accept \( w \)
      
    • Which makes R a decider for \( A^\text{TM} \)
(p. 219 & Prob. 5.28) Rice’s Theorem:

Let P be any property of the language of a TM

\[ L_P = \{ <M> | M \text{ a TM such that } L(M) \text{ has property } P \} \]

- \( L_P \) contains some but not all TMs
- Whenever \( L(M_1) = L(M_2) \), \( <M_1> \in L_P \) iff \( <M_2> \in L_P \)

- Thus \( L_P \) is undecidable

Above proved undecidability from \( A_{TM} \)

- but other undecidable languages such as \( E_{TM} \) usable

\[ EQ_{TM} = \{ <M_1, M_2> | M_1, M_2 \text{ TMs, and } L(M_1) = L(M_2) \} \]

(p. 220) Theorem 5.4 \( EQ_{TM} \) is undecidable

- Assume TM R decides \( EQ_{TM} \)
- Construct S to decide \( E_{TM} \) (not \( A_{TM} \)) as follows:
  - On input \( <M> \) to \( E_{TM} \)
  - Run R on \( <M,M_1> \) where \( M_1 \) a TM that rejects all inputs
  - If R accepts (i.e. \( M \) matches machine with empty language), then S accepts (\( L(M) \) is empty)
  - If R rejects (\( M\neq M_1 \)) then S rejects (\( M \) accepts something)
- If R exists we now have in S a decider for \( E_{TM} \)
- Not possible, so R cannot exist
• (p. 220) Reductions via Computational Histories

• **Accepting Computational History** of M given w
  - Sequence of configurations $C_1, \ldots, C_l$ where
    - $C_1$ is start, $C_l$ is accepting, and $C_i$ legally follows from $C_{i-1}$
    - Remember a configuration $= ua q_i bv$, $b$ under tape head
    - Note this is finite in length

• **Rejection Computational History** is similar

• (p. 221) **Linear Bounded Automata (LBA)**
  - TM with finite tape
  - Cannot move off of original tape: Off left or into “blanks”

• (p. 222) **Lemma 5.8. Assume** M is an LBA with exactly $q$ states & $g$ symbols in $\Gamma$. There are exactly $q^n g^n$ possible configurations of tape of length $n$.

• $A_{\text{LBA}} = \{<M, w> | \, M \text{ an LBA that accepts } w\}$

• (p. 222) **Theorem 5.9** $A_{\text{LBA}}$ is decidable
  - Have decider L keep track of each configuration that M enters while processing $w$
  - If we ever enter same configuration a 2nd time, reject
    - This is after at most $q^n g^n$ steps of simulating M
  - If M accepts, L accepts
  - If M rejects, L rejects
• (p. 223) \( E_{LBA} = \{<M>| M \text{ an LBA where } L(M) \text{ is empty}\} \)

• **Theorem 5.10** \( E_{LBA} \) is undecidable
  • Assume TM \( R \) decides \( E_{LBA} \)
  • (p. 224) Construct an LBA \( B \) that recognizes all accepting computational histories for \( M \) on \( w \)
    • If \( M \) accepts \( w \), \( L(B) = 1 \) string
    • If \( M \) does not accept \( w \), then \( L(B) \) is empty
  • Given \( <M,w> \) \( B \) constructs all valid histories as strings separated by \#s
  • Construct \( S \) to decide \( A_{TM} \) as follows
    • Construct LBA \( B \) from \( <M,w> \)
    • Run \( R \) on \( <B> \)
    • If \( R \) rejects, \( S \) accepts
    • If \( R \) accepts, \( S \) rejects

• (p. 5.13) **Theorem 5.12** Likewise \( \text{ALL}_{CFG} = \{<G>| G \text{ is CFG where } L(G) = \Sigma^* \} \) is undecidable
PCP: POST CORRESPONDENCE PROBLEM

- Consider a set of dominoes with 2 strings on each
- A match: list of dominoes where concatenated string on top is same as concatenated string on bottom
  - Repetitions allowed
- PCP: Given a set of dominoes, is there a match?
  - Can use duplicates
  - Try Exercise 5.3 p. 239
- PCP is undecidable (see book for proof details)
  - Reduction from $A_{TM}$ via accepting histories
  - Given any $<M,w>$ build a matching PCP instance
  - IF PCP is decidable, so is $A_{TM}$