CSCI 1010
Models of Computation

Lecture 19
Limits to Language Recognition
Overview

• Universal TMs, again
• Definitions of decidable and recursively enumerable (r.e.) languages.
• A first r.e. but not decidable language, $L_1$.
• The role of reductions.
• Examples of undecidable problems.
• Rice’s Theorem
Encoding Turing Machines

• **Goal:** Define a **canonical representation** $\rho(M)$ of a TM $M$ so that a **universal TM** can simulate it.
  
  – Encode states, $Q$; inputs letters $\Gamma$, blank $\beta$; moves $\{L,R\}$ as integers in set $\{0,1,2,\ldots, |Q|+|\Gamma|+2\}$.
  
  – Encode a transition $\delta(p,y) = (q,z)$ as 4-tuple $(p,y,q,z)$ and transition function $\delta$ as string $\rho(M)$ of 4-tuples.

  – Represent both “(” and “)” by 10 and comma by 01.
  
  – Convert integers into $k$-bit binary numbers. Then, replace 0 by 00 & 1 by 11 for $k = \lceil \log (|Q|+|\Gamma|+2) \rceil$.
Encoding Turing Machines

• Example: (1,2,3,4)(2,1,5,3) encoded in 3-bit binary as

• 10 000011 01 001100 01 001111 01 110000 10
10 001100 01 000011 01 110011 01 001111 10
– Spaces added for clarity
Universal TM

- Let universal TM $U$ have alphabet $\{0, 1, \underline{0, 1}\}$.
- To simulate $M$, place $\rho(M)$ left-adjusted on tape of $U$ with the blank $\beta$ to its left. Place input $w$ to its right.
- Use $0, 1$ to mark head position in $\rho(M)$ and in $w$.
- Simulate by bouncing between $\rho(M)$ and in $w$.
- $U$ simulates $M$ on $w$. $U$ halts iff $M$ halts; gives same result.
Definitions

• Language $L$ is **recursively enumerable** (r.e.) if there is a TM that accepts strings in $L$ by halting in an accept state and does not accept strings not in $L$. (*It may not halt on strings not in $L$.*)

• Language $L$ is **decidable** if both $L$ and $\overline{L}$ are r.e. That is, there is an algorithm $A$ (*a halting TM*) that accepts only strings in $L$. We say that $A$ **decides** $L$. 
Two Decidable Languages

• Every regular language is decidable.

• \{ρ(M) | M is a DFSM and L(M) = ∅\}
  – Do a BFS of graph of M to see if an accepting state can be reached. If not, L(M) = ∅.
  – Time is polynomial in |ρ(M)|.
A Language that is Not R.E.

• Let $M_i$ be $i$th TM (each TM is a string $\rho(M)$).
• Let $w_i$ be $i$th binary string.

\[ L_1 = \{ w_i \mid \text{not accepted by } M_i \} \]

• Put 1 in $(i,j)$ in matrix if $w_i$ not accepted by $M_j$
• $w$ is in $L_1$ iff it corresponds to a 1 on the diagonal.
Diagonalization

• **Theorem** $L_1$ is not recursively enumerable.

• **Proof** Let $w_i$ and $M_i$ denote $i$th binary string and $i$th TM. Assume $L_1$ is r.e. and TM $M_k$ recognizes it. If $w_k \in L_1$, $M_k$ accepts it, which means $w_k \notin L_1$. But if $w_k \notin L_1$, then $M_k$ does not accepts it, which means $w_k \in L_1$.

• Thus, $w_k \in L_1$ iff $w_k \notin L_1$, a contradiction. Thus, $L_1$ cannot be recursively enumerable.
R.E. but Not Decidable Languages

• **Theorem** The complement of a decidable language is decidable.

• **Proof** If $L$ is decidable, there is a TM $M(L)$ that halts on all inputs and recognizes $L$. It has one halt state. Change the accepting halt state to rejecting. Make all other halt states, which are rejecting, accepting. Put this TM into standard form by adding one new halt state. Make all new accepting halt states move to that new halt state on all inputs.
An R.E. Language Not Decidable

• **Theorem** The following language is r.e. but not decidable.

\[ L_2 = \{ w_i \mid w_i \text{ is accepted by } M_i \} \]

• **Proof** The complement of \( L_2 \) is \( L_1 \) which is not r.e. To show \( L_2 \) is r.e., construct a TM \( M^* \) that given \( w_i \), uses enumeration to find \( M_i \). \( M^* \) then simulates \( M_i \) on \( w_i \) using a universal TM. By definition, \( M^* \) accepts exactly those \( w_i \) in \( L_2 \).
Reducibility

• Let $L_1$ and $L_2$ be languages over $\Sigma_1$ and $\Sigma_2$. $L_1$ is **reducible** to $L_2$ ($L_1 \rightarrow_{TM} L_2$) if there is an **algorithm** $A$ that translates each $w$ in $\Sigma_1^*$ into a $z$ in $\Sigma_2^*$ such that $w \in L_1$ if and only if $z \in L_2$. $A$ is also called a reduction.

  – **Note**: Reductions used with $\textbf{NP}$-complete languages are $\textbf{P}$-time algorithms. Here they are generally not $\textbf{P}$-time.

  – We use reductions to show that if $L_2$ can be decided, so can $L_1$. If $L_1$ cannot be decided, the same holds for $L_2$. 
Reducibility

• **Theorem** Let $L_1 \rightarrow_{TM} L_2$. If $L_2$ is decidable, $L_1$ is decidable. If $L_1$ is undecidable, $L_2$ is also undecidable.

• **Proof** Let $A$ be the algorithm reducing $L_1$ to $L_2$. If $L_2$ is decidable, a halting TM $M_b$ recognizes it. Let TM $M_a$ do following: On input $w$, $M_a$ uses $A$ to produce $z$ which it passes to $M_b$. If $M_b$ accepts $w$, $M_a$ accepts. If $M_b$ rejects it, the same holds for $M_a$. Thus, $M_a$ decides $L_1$.

• Let $L_1$ be undecidable. If we assume that $L_2$ is decidable, from the above, $L_1$ is decidable, a contradiction. Thus, $L_2$ cannot be decidable.
Halting is Undecidable

• Halting problem (language)
  \[ L_H = \{ [ρ(M), x] \mid M \text{ halts on } x \} \]

• **Theorem**: \( L_H \) is r.e. but not decidable.

• **Proof** \( L_H \) is r.e. because, if \([ρ(M), x]\) is given to a universal TM \( U \), \( U \) will halt and accept it.
Halting is Undecidable

• **Proof (cont.)** To show $L_H$ not decidable, assume $M_H$ exists to decide it and show this implies that $L_1$ is decidable, which it is not. We show $L_1$ is reducible to $L_H$.

$$L_1 = \{ w_i \mid w_i \text{ is not accepted by } M_i \}$$
Halting is Undecidable

• Proof (cont.)
  \[ L_1 = \{ w_i \mid w_i \text{ is not accepted by } M_i \} \]
  • We reduce \( L_1 \) to \( L_H \).
  • Using \( M_H \) construct \( M' \) to decide \( L_1 \). On \( x \), \( M' \) uses its work tapes to generate \( w_1 \) and \( \rho(M_1) \), \( w_2 \) and \( \rho(M_2) \), etc. until it finds \( w_i = x \). Each \( \rho(M_i) \) is string of 4-tuples \((p, y, q, z)\)

• \( M' \) simulates \( M_H \) on \([\rho(M_i), x]\) to determine if \( M_i \) halts on \( x \). If \( M_i \) does not halt, \( M' \) accepts \( x \). If \( M_i \) does halt on \( x \), \( M' \) simulates \( M_i \) on \( x \) and rejects if it accepts and accepts if it rejects. Thus, \( M' \) recognizes \( L_1 \). Contradiction!
Halting Problem Not Decidable
A Direct Proof

\[ L_H = \{[\rho(M), x] \mid M \text{ halts on } x \} \]

- **Theorem** \( L_H \) is r.e. but not decidable.
- **Proof** Assume that \( M_H \) decides \( L_H \). Given \([\rho(M), x]\), \( M_H \) halts and reports whether \([\rho(M), x]\) is in \( L_H \) or not.

- Construct \( H_1 \) which, given \([\rho(M), x]\), simulates \( M_H \) on \([\rho(M), x]\). If \( M_H \) does not accept (\( M \) does not halt on \( x \)), let \( H_1 \) accept. If \( M_H \) does accept, \( H_1 \) enters an infinite loop.

- Finally, design \( H_2 \). On input \( w \), \( H_2 \) rejects if \( w \) is not a TM description. Otherwise, \( H_2 \) simulates \( H_1 \) on \([w, w]\).

- What happens when \( H_2 \) is given \( w = \rho(H_2) \)?
Halting Problem is Not Decidable

• On input $\rho(H_2)$, $H_2$ simulates $H_1$ on $[\rho(H_2), \rho(H_2)]$. Thus, $H_1$ simulates $M_H$ on $[\rho(H_2), \rho(H_2)]$.
  - If $M_H$ says that $H_2$ halts on $\rho(H_2)$, then $H_1$ runs forever which means $H_2$ runs forever, a contradiction.
  - If $M_H$ says that $H_2$ does not halt on $\rho(H_2)$, then $H_1$ halts which means that $H_2$ halts, another contradiction.

• It follows that the assumption that $L_H$ was decidable is incorrect.
Empty Set Recognition is Undecidable

\[ L_E = \{ \rho(M) \mid L(M) = \emptyset \} \]

• **Theorem:** \( L_E \) is not decidable.

• **Proof** Assume \( M_E \) exists deciding \( L_E \). Then show \( L_H \) can be decided, contradiction. We show \( L_H \) is reducible to \( L_E \).

• Given a TM \( M \) (i.e. \( \rho(M) \)) and a string \( w \), our goal is to determine if \( M \) halts on \( w \). Design a TM \( T_1[M,w] \) that on input \( x \) loops if \( x \not\in w \) and simulates \( M \) on \( w \) otherwise. \( T_1[M,w] \) either does not halt in which case \( L(T_1[M,w]) = \emptyset \) or it does halt and its language is not empty.

• To decide \( L_H \), given \( \rho(M) \) and \( w \), create \( T_1[M,w] \) and then pass it to \( M_E \) which decides \( L_H \). Contradiction.
A Non-Computable Function

• Let $t(M, w)$ be the number of steps $M$ takes to halt on input $w$ or -1 if $M$ does not halt.
• $t(M, w)$ cannot be computable because that would allow one to solve the Halting problem.
• Why is that?
Rice’s Theorem

• It says that no algorithm exists to determine from a description of a TM whether or not the language it accepts falls into any proper subset of the r.e. languages.

• \( \text{RE} \) denotes the r.e. languages over the alphabet \( \mathcal{B} \). Let \( \mathcal{C} \) be a proper subset of \( \text{RE} \) and let

\[
\mathcal{L}_c = \{ \rho(M) \mid L(M) \in \mathcal{C} \}
\]

be the TMs whose languages are in \( \mathcal{C} \).
Rice’s Theorem

• **Theorem (Rice)** Let \( C \) be a non-trivial set of r.e. languages. That is, \( C \subseteq \text{RE} \) but \( C \neq \text{RE} \), \( C \neq \emptyset \). Then \( L_C \) is not decidable.

• **Proof** By contradiction. Assume \( M_C \) decides \( L_C \). We show this implies \( L_H \) is decidable.

• (a) If \( B^* \) in \( C \), let \( L \) be a **language** in \( \text{RE} \) – \( C \). (b) If \( B^* \) not in \( C \), let \( L \) be in \( C \). There is such an \( L \) because \( C \) is a proper subset of \( \text{RE} \).

• Let \( M_L \) recognize \( L \).
Rice’s Theorem

• If $M_C$ decides $\mathcal{L}_C$, we construct $M_H$ deciding $L_H$. For arbitrary TM $M$ and input $w$, construct $T_2[M,w]$:  
  - On input $x$, $T_2[M,w]$ in parallel a) simulates $M_L$ on $x$ and b) simulates $M$ on $w$, by alternating steps between them.
• If $M_L$ accepts $x$ or $M$ halts on $w$, $T_2[M,w]$ halts and accepts. If $M_L$ rejects $x$, continue simulation of $M$ on $w$. Accept if $M$ halts on $w$.

• $T_2[M,w]$ accepts every $x \in L$. It accepts some $x \notin L$ iff $M$ halts on $w$. Thus, $L(T_2[M,w]) = L$ if $M$ does not halt on $w$ and $L(T_2[M,w]) = \mathcal{B}^*$ if $M$ does halt on $w$.

• Use $M_C$ on $T[M,w]$ to decide halting or realize $M_H$. Contradiction!
Application of Rice’s Theorem

• **Theorem** Let $\text{REGULAR} = \{\rho(M) \mid L(M) \text{ is regular}\}$. REGULAR is not decidable.

• **Proof** The class of regular languages is a non-empty subset of the set of r.e. languages. The result follows from Rice’s Theorem.
Is It Possible to Detect Infected Code?

• Consider the following piece of code
  
f();
  Infect_code for f();

• Is the code infected?
  – Yes, if f() halts.
  – No, if f() does not halt.

• Any tester for infected code must determine whether or not a function (a TM) halts!
Summary

• Universal TMs, again
• Definitions of decidable and recursively enumerable (r.e.) languages.
• A first r.e. but not decidable language, $L_1$.
• The role of reductions.
• Examples of undecidable problems.
• Rice’s Theorem