non-RE languages

**Theorem:** there exist languages that are not Recursively Enumerable.

Proof outline:
- the set of all TMs is countable
- the set of all languages is uncountable
- the function \( L : \{\text{TMs}\} \to \{\text{languages}\} \) cannot be onto

non-RE languages

**Lemma:** the set of all TMs is countable.

Proof:
- each TM \( M \) can be described by a finite-length string \( <M> \)
- can enumerate these strings, and give the natural bijection with \( \mathbb{N} \)

non-RE languages

**Lemma:** the set of all languages is uncountable

Proof:
- fix an enumeration of all strings \( s_1, s_2, s_3, \ldots \) (for example, lexicographic order)
- a language \( L \) is described by its characteristic vector \( \chi_L \), whose \( i \)-th element is 0 if \( s_i \) is not in \( L \) and 1 if \( s_i \) is in \( L \)

non-RE languages

- suppose the set of all languages is countable
- list characteristic vectors of all languages according to the bijection \( f \):

\[
\begin{array}{c|c}
\text{Index} & \chi_L \\
\hline
1 & 0101010... \\
2 & 1010011... \\
3 & 1100001... \\
4 & 0100011... \\
\end{array}
\]
non-RE languages
– suppose the set of all languages is countable
– list characteristic vectors of all languages
  according to the bijection \( f \):
  \[
  \begin{array}{c|c}
  n \quad f(n) & x \\
  \hline
  1 & 0101010101... \\
  2 & 101010011010... \\
  3 & 110001010101... \\
  4 & 010011010101... \\
  \vdots & \vdots \\
  \end{array}
  \]
  where \( p \) digit \( \neq \) digit of \( f(i) \)
  \( x \) cannot be in the list!
  therefore, the language with characteristic vector \( x \) is not in the list
  \[ x = 1101... \]
  where \( i \)th digit \( \neq i \)th digit of \( f(i) \)
  \( x \) cannot be in the list!
  therefore, the language with characteristic vector \( x \) is not in the list
  \[
  \begin{array}{c}
  \text{set} \ x = 1101... \\
  \end{array}
  \]
  \[
  \begin{array}{c}
  \text{where} \ p \ \text{digit} \ \neq \ \text{digit of} \ f(i) \\
  \end{array}
  \]
  \[
  \begin{array}{c}
  \text{x cannot be in the list!} \\
  \end{array}
  \]
  \[
  \begin{array}{c}
  \text{therefore, the language with characteristic vector} \ x \ \text{is not} \\
  \text{in the list} \\
  \end{array}
  \]
  \[
  \begin{array}{c}
  \text{where} \ i \ \text{th digit} \ \neq \ i \ \text{th digit of} \ f(i) \\
  \end{array}
  \]
  \[
  \begin{array}{c}
  \text{x cannot be in the list!} \\
  \end{array}
  \]
  \[
  \begin{array}{c}
  \text{therefore, the language with characteristic vector} \ x \ \text{is not} \\
  \text{in the list} \\
  \end{array}
  \]

So far...
\[
\begin{array}{c}
\text{regular languages} \\
\text{context free languages} \\
\text{decidable} \\
\text{RE} \\
\end{array}
\]
\[
\begin{array}{c}
\text{decidable} \\
\text{all languages} \\
\end{array}
\]
\[
\begin{array}{c}
\text{undecidable} \\
\text{RE} \\
\end{array}
\]
\[
\begin{array}{c}
\text{set} \ x = 1101... \\
\end{array}
\]

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

- The complement of a RE language is called a co-RE language

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

- The complement of a RE language is called a co-RE language

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)

A better option:
- to prove NEW is decidable, show how to transform it into a known decidable problem OLD so that solution to OLD can be used to solve NEW.
- to prove NEW is undecidable, show how to transform a known undecidable problem OLD into NEW so that solution to NEW can be used to solve OLD.

Called a reduction

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

- Try to prove un decidable:
  - \( A_{TM} = \{<M, w>: M \text{ accepts input } w\} \)
- We know this language is undecidable:
  - \( \text{HALT} = \{<M, w>: M \text{ halts on input } w\} \)
- Idea:
  - suppose \( A_{TM} \) is decidable
  - show that we can use \( A_{TM} \) to decide HALT
  - conclude HALT is decidable. Contradiction.

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 \(<M', w> \in A_{TM}\)

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.