RICE’S THEOREM

Let L be a language over Turing machines. Assume that L satisfies the following properties:

1. For TMs $M_1$ and $M_2$, if $M_1 = M_2$ then $M_1 \in L \iff M_2 \in L$
2. There are TMs $M_1$ and $M_2$, such that $M_1 \in L$ and $M_2 \notin L$

Then $L$ is undecidable

$FIN_{TM} = \{M \mid M \text{ is a TM and } L(M) \text{ is finite}\}$

RICE’S THEOREM

Let L be a language over Turing machines. Assume that L satisfies the following properties:

1. For TMs $M_1$ and $M_2$, if $M_1 = M_2$ then $M_1 \in L \iff M_2 \in L$
2. There are TMs $M_1$ and $M_2$, such that $M_1 \in L$ and $M_2 \notin L$

Then $L$ is undecidable

$FIN_{TM} = \{M \mid M \text{ is a TM and } L(M) \text{ is finite}\}$

RICE’S THEOREM

Let L be a language over Turing machines. Assume that L satisfies the following properties:

1. For TMs $M_1$ and $M_2$, if $M_1 = M_2$ then $M_1 \in L \iff M_2 \in L$
2. There are TMs $M_1$ and $M_2$, such that $M_1 \in L$ and $M_2 \notin L$

Then $L$ is undecidable

$FIN_{TM} = \{M \mid M \text{ is a TM and } L(M) \text{ is finite}\}$

RICE’S THEOREM

Let L be a language over Turing machines. Assume that L satisfies the following properties:

1. For TMs $M_1$ and $M_2$, if $M_1 = M_2$ then $M_1 \in L \iff M_2 \in L$
2. There are TMs $M_1$ and $M_2$, such that $M_1 \in L$ and $M_2 \notin L$

Then $L$ is undecidable

$FIN_{TM} = \{M \mid M \text{ is a TM and } L(M) \text{ is finite}\}$

$REG_{TM} = \{M \mid M \text{ is a TM and } L(M) \text{ is regular}\}$

FINTM = $\{M \mid M$ is a TM and $L(M)$ is finite}
RICE'S THEOREM

Let \( L \) be a language over Turing machines. Assume that \( L \) satisfies the following properties:

1. For TMs \( M_1 \) and \( M_2 \), if \( M_1 \equiv M_2 \) then \( M_1 \in L \Leftrightarrow M_2 \in L \)

2. There are TMs \( M_1 \) and \( M_2 \) such that \( M_1 \in L \) and \( M_2 \notin L \)

Then \( L \) is undecidable

Proof: Will show:

\( A_{TM} \) is mapping reducible to \( L \)

Problem

Let \( S = \{ M : M \) is a TM with the property:

for all \( w \), \( M(w) \) accepts implies \( M(w^R) \) accepts\}

\( S \) is undecidable.
ATM = \{ (M,w) | M \text{ is a TM that accepts string } w \} \\
HALT_{TM} = \{ (M,w) | M \text{ is a TM that halts on string } w \} \\
E_{TM} = \{ M | M \text{ is a TM and } L(M) = \emptyset \} \\
REG_{TM} = \{ M | M \text{ is a TM and } L(M) \text{ is regular} \} \\
EQ_{TM} = \{(M, N) | M, N \text{ are TMs and } L(M) = L(N)\} \\
ALL_{PDA} = \{ P | P \text{ is a PDA and } L(P) = \Sigma^* \} \\

ALL UNDECIDABLE

Where is Rice’s Theorem Applicable?

Which are SEMI-DECIDABLE?

“The recursion theorem is just like tennis. Unless you’re exposed to it at age five, you’ll never become world class.”

- Juris Hartmanis (Turing Award 1993)

(Note: Juris didn’t see the recursion theorem until he was in his 20’s….)

Recursion Theorem says:
A Turing machine can obtain its own description, and compute with it.

We can use the operation:
“Obtain your own description”
in pseudocode!

Given a computable t, we can get a computable r such that r(w) = t(<R>,w) where <R> is a description of r.

INSIGHT: T (or t) is really R (or r)

Theorem: A_{TM} is undecidable

Proof (using the Recursion Theorem):
Assume H decides A_{TM}
Construct machine R such that on input w:
1. Obtains its own description <R>
2. Runs H on (<R>, w) and flips the output

Running R on input w always does the opposite of what H says it should!
Theorem: \( A_{TM} \) is undecidable

Proof (using the Recursion Theorem):
Assume \( H \) decides \( A_{TM} \)
Let \( T_H(x, w) \) =
  Reject if \( H(x, w) \) accepts
  Accept if \( H(x, w) \) rejects

(Here \( x \) is viewed as a code for a TM)
By the Recursion Theorem, there is a TM \( R \) such that:
\[ R(w) = T_H(<R>, w) = \begin{cases} 
  \text{Reject} & \text{if } H(<R>, w) \text{ accepts} \\
  \text{Accept} & \text{if } H(<R>, w) \text{ rejects}
\end{cases} \]
Contradiction!

---

**THE FIXED-POINT THEOREM**

Theorem: Let \( f: \Sigma^* \to \Sigma^* \) be a computable function. There is a TM \( R \) such that \( f(<R>) \) describes a TM that is equivalent to \( R \).

Proof: Let \( T_f(x, w) = G(w) \) where \( <G> = f(x) \)
(Here \( f(x) \) is viewed as a code for a TM)
By the Recursion Theorem, there is a TM \( R \) such that:
\[ R(w) = T_f(<R>, w) = G(w) \]
Hence \( R = G \) where \( <G> = f(<R>) \), ie \( <R> \) “\( \equiv \)” \( f(<R>) \)
So \( R \) is a fixed point of \( f \)!
THE FIXED-POINT THEOREM

Theorem: Let \( f : \Sigma^* \rightarrow \Sigma^* \) be a computable function. There is a TM \( R \) such that \( f(<R>) \) describes a TM that is \textit{equivalent} to \( R \).

Examples:
1. For any 1-1 computable enumeration of \( \Sigma^* \) (or Gödel numbering of TMs), there will always be a TM \( R \) that is equivalent to its successor in the enumeration.
2. Let a virus flip the first bit of each word \( w \) in \( \Sigma^* \) (or in each TM). Then there is a TM \( R \) that “remains uninfected”.

THE RECURSION THEOREM

Theorem: Let \( T \) be a Turing machine that computes a function \( t : \Sigma^* \times \Sigma^* \rightarrow \Sigma^* \). Then there is a Turing machine \( R \) that computes a function \( r : \Sigma^* \rightarrow \Sigma^* \), where for every string \( w \),

\[
    r(w) = t(<R>, w)
\]

To Start: Need to show how to construct a TM that computes its own description.

Lemma: There is a computable function \( q : \Sigma^* \rightarrow \Sigma^* \), where for any string \( w \), \( q(w) \) is the\textit{ description} of a TM \( P_w \) that on any input, prints out \( w \) and then accepts.

A TM SELF THAT PRINTS <SELF>

\[
\begin{array}{c}
\text{TM B, on any input } <M>, \\
\text{prints the code for a TM that on any input} \end{array}
\]

What about B on input <B>?
**The Recursion Theorem**

Theorem: Let $T$ be a Turing machine that computes a function $t : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$. There is a Turing machine $R$ that computes a function $r : \Sigma^* \rightarrow \Sigma^*$, where for every string $w$,

$$r(w) = t(<R>, w)$$

**Proof:**

$$\begin{align*}
(a,b) & \rightarrow T \rightarrow t(a,b) \\
(w) & \rightarrow R \rightarrow t(<R>, w)
\end{align*}$$
Proof:
\[(a,b) \rightarrow t(a,b)\]
\[<M> \rightarrow B\]
\[w \rightarrow P_{<M>} \rightarrow B\]
\[<R> = ???\]
\[w' \rightarrow R\]

Proof:
\[(a,b) \rightarrow t(a,b)\]
\[<M> \rightarrow B\]
\[w \rightarrow P_{<M>} \rightarrow M\]
\[<R> = ???\]
\[w' \rightarrow P_{<M>} \rightarrow BT\]

Proof:
\[(a,b) \rightarrow t(a,b)\]
\[<M> \rightarrow B\]
\[w \rightarrow P_{<M>} \rightarrow M\]
\[<R> = ???(P_{<BT>})\]
\[w' \rightarrow P_{<BT>} \rightarrow BT\]

Proof:
\[(a,b) \rightarrow t(a,b)\]
\[<M> \rightarrow B\]
\[w \rightarrow P_{<M>} \rightarrow M\]
\[<R> = ???(P_{<BT>})\]
\[w' \rightarrow P_{<BT>} \rightarrow BT\]

Proof:
\[(a,b) \rightarrow t(a,b)\]
\[<M> \rightarrow B\]
\[w \rightarrow P_{<M>} \rightarrow M\]
\[<R> = ???(P_{<BT>}T)\]
\[w' \rightarrow P_{<BT>} \rightarrow R\]

Proof:
\[(a,b) \rightarrow t(a,b)\]
\[<M> \rightarrow B\]
\[w \rightarrow P_{<M>} \rightarrow M\]
\[<R> = ???(P_{<BT>})\]
\[w' \rightarrow P_{<BT>} \rightarrow BT\]
**THE RECURSION THEOREM**

Theorem: Let $T$ be a Turing machine that computes a function $t : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$. There is a Turing machine $R$ that computes a function $r : \Sigma^* \rightarrow \Sigma^*$, where for every string $w$,

$$r(w) = t(<R>, w)$$

$$(a,b) \xrightarrow{T} t(a,b)$$

$$w \xrightarrow{R} t(<R>, w)$$

Read Chapter 6.1 and 6.3 for next time