COURSE: THEORY OF AUTOMATA COMPUTATION

TOPICS TO BE COVERED

- Using Reduction to prove properties

USING REDUCTION TO PROVE R.E.

Theorem: If L_2 is R.E., and $L_1 \leq L_2$, then L_1 is also R.E.

Proof:

Let L_1 and L_2 be languages over Σ, $L_1 \leq L_2$, and L_2 be R.E.

Because L_2 is R.E, there is a TM T_2 accepting L_2.

Because $L_1 \leq L_2$, there is a TM T_1 computing a function f such that $w \in L_1 \leftrightarrow f(w) \in L_2$.

Using Reduction to Prove R.E.

Construct a TM $T=T_1 \rightarrow T_2$. We show that T accepts L_1.

- If $w \in L_1$, T_1 in T computes $f(w) \in L_2$ and T_2 in T accepts f(w). Thus, T accepts w.
- If $w \notin L_1$, T_1 in T computes $f(w) \notin L_2$ and T_2 in T does not accept (f(w)). Thus, T does not accept w.

Thus, L_1 is also R.E.

Using Reduction to Prove Non-R.E.

Collorary:

If L_1 is not recursively enumerable, and $L_1 \leq L_2$, then L_2 is not recursively enumerable.

Using Reduction to Prove Co-R.E.

Theorem: If L_2 is co-R.E., and $L_1 \leq L_2$, then L_1 is also co-R.E.

Proof:

Let L_1 and L_2 be languages over Σ, $L_1 \leq L_2$, and L_2 be co-R.E.

Because L_2 is co-R.E, $\overline{L_2}$ is R.E.

Because $L_1 \leq L_2$, $\overline{L_1} \leq \overline{L_2}$. Then, $\overline{L_1}$ is R.E.

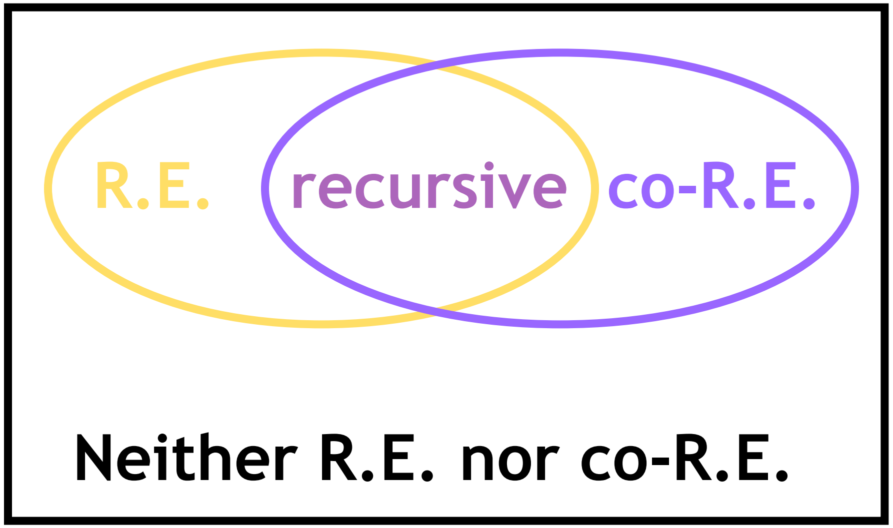
Thus, L_1 is co-R.E.

USING REDUCTION TO PROVE NON-CO-R.E.

Collorary:

If L_1 is not co-R.E., and $L_1 \leq L_2$, then L_2 is not co-R.E.

ANOTHER WAY TO PROVE UNDECIDABILITY

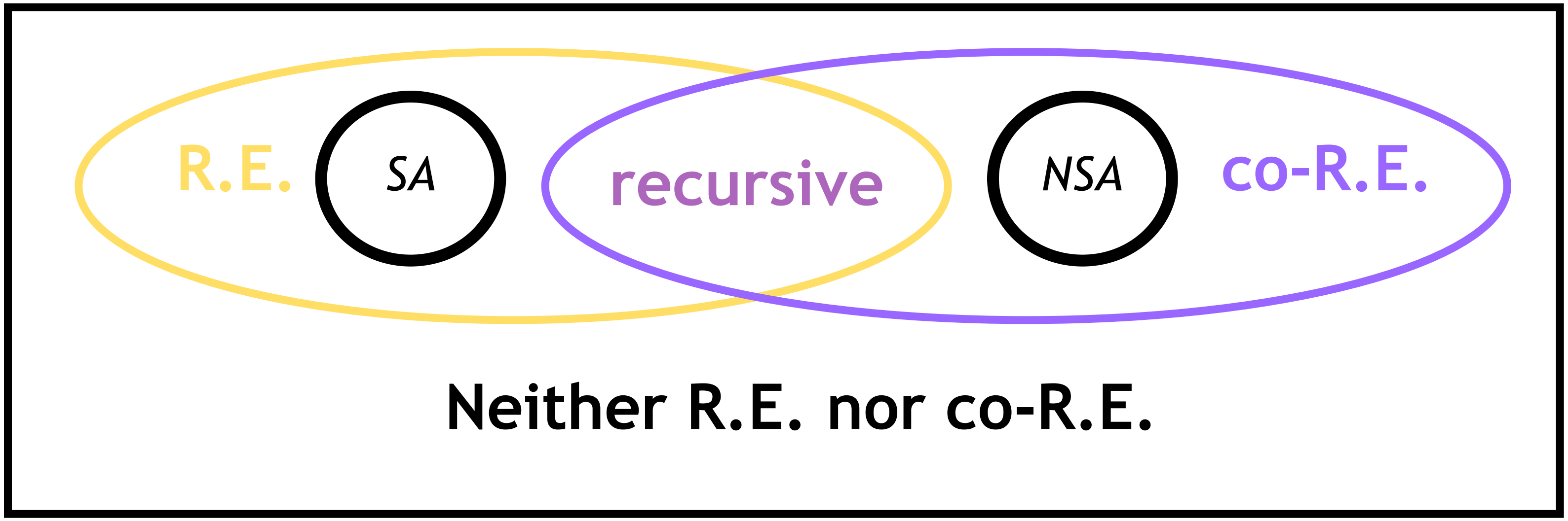


Let L1≤L2.
If L1 is not recursive / R.E. / co-R.E.,
then L2 is not recursive / R.E. / co-R.E.

To prove a language L is not recursive:

1. Guess where L is (not R.E. or not co-R.E.)
2. Choose another non-recursive language R which is of the same type
3. Show R ≤ L.

ANOTHER WAY TO PROVE UNDECIDABILITY



To prove a language L is not recursive:

1. Guess where L is (not R.E. or not co-R.E.)
2. If L is not R.E., then show NSA $\leq$ L.
3. If L is not co-R.E., then show SA $\leq$ L.

GUESS IF IT'S REC., R.E., CO-R.E., OR NEITHER

Given a TM T,

- does T get to state q on blank tape? *R.E., not co-R.E.*
- does T accept ε? *R.E., not co-R.E.*
- does T output 1? *Neither*
- does T accept everything? *Neither*
- is L(T) finite? *Neither*

PROBLEM OF ACCEPTING AN EMPTY STRING

- We will prove that the problem if a TM accepts an empty string is undecidable.
- This problem is corresponding to the following language.
  - Acceptε = {e(M)| M is a TM accepting ε}
- Thus, we will prove that Acceptε is not recursive.

ACCEPTε IS NOT RECURSIVE.

Proof:

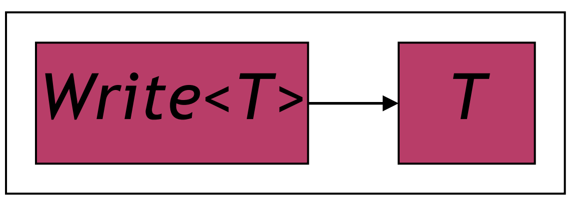
(Guess Acceptε is in R.E., but not co-R.E.)

- Show SA $\leq$ Acceptε

(We want a Turing-computable f $\underline{\text{n}}$ f(<T>)=<M> such that

  - T accepts e(T) $\rightarrow$ M accepts ε
  - T does not accept e(T) $\rightarrow$ M does not accept ε

- Let f(T)=M is a TM that first writes e(T) after its input and then runs T.
- M writes e(T) after its input. If its input is ε, T has e(T) as input.



M

ACCEPTε IS NOT CO-R.E.

Verify that T accepts e(T) $\leftrightarrow$ M accepts ε

M writes e(T) and lets T run. If the input of M is ε:

- when T accepts e(T), M accepts ε.
- when T doesn't accept e(T), then M doesn't accept ε.

ACCEPTε IS NOT CO-R.E.

Next, we show that there is a TM TF computing f.

TF works as follows:

- changes the start state of T in e(T) to a new state
- add e(*Write<T>*), make its start state the start state of TF, and make the transition from its halt state to T's start state.

Then, SA $\leq$ Acceptε.

Then, Acceptε is not co-R.E, and is not recursive.

HALTING PROBLEM

- Problem
  - Given a Turing machine T and string z, does T halt on z?
  - Given a program P and input z, does P halt on z?
- Language
  - Halt = $\{w \in \Sigma^* \mid w = e(T)e(z)$ for a Turing machine T halting on z$\}$.
  - Halt = {<T,z>| T is a Turing machine halting on z}.

HALTING PROBLEM IS UNDECIDABLE

Proof:

Let Halt = {<T,z>| T is a Turing machine halting on z}.

(Guess Halt is in R.E., but not co-R.E.)

- Show SA $\leq$ Halt

(We want a Turing-computable f $\underline{n}$ $f(<T_1>)=<T_2, z>$ such that

  - T_1 accepts $e(T_1) \rightarrow T_2$ halts on z
  - T_1 does not accept $e(T_1) \rightarrow T_2$ does not halt on z

Then, a possible function is $f(<T>) = <T, e(T)>$ because T accepts $e(T) \leftrightarrow$ T halts on $e(T)$.)

HALTING PROBLEM IS UNDECIDABLE

- Let $f(X) = X \cdot e(X)$. f is Turing-computable because there is a TM that can write an encoding of an input string after the string itself.
- If $f(<T>)=<T>\cdot e(<T>)$, then T accepts e(T) $\leftrightarrow$T halts on e(T).
- Then, SA $\leq$ Halt, and Halt is not co-R.E. Thus, Halting problem is undecidable.

SOME OTHER UNDECIDABLE PROBLEMS

- FINITE

  Given a TM T, is L(T) finite?

Guess FINITE is neither R.E. nor co-R.E.

- To assure L(T) is finite, we need to run T on all possible input and count if T accepts a finite number of strings.
- To assure L(T) is infinite, we need to run T on all possible input and count if T accepts an infinite number of strings.

FINITE IS NOT RECURSIVE

Let FINITE={<T>| T is a TM such that L(T) is finite.}

Guess FINITE is neither R.E. nor co-R.E.

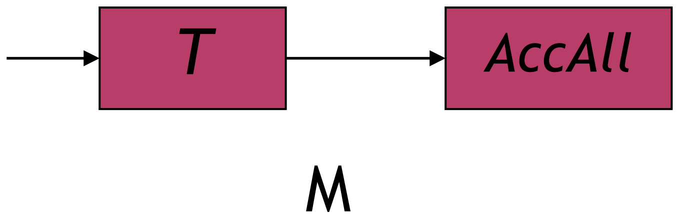
Choose NSA which is not co-R.E. to show that NSA$\leq$FINITE.

We want to find a Turing-computable function f such that $\langle T\rangle \in NSA \leftrightarrow f(\langle T\rangle)=M \in FINITE$

$\langle T\rangle \in NSA \rightarrow$ M accepts $\varnothing$, and thus L(M) is finite.

$\langle T\rangle \notin NSA \rightarrow$ M accepts Σ^*, and thus L(M) is infinite.

Then, let M=f(<T>) be a TM that runs T on its input, and accepts everything if T halts.

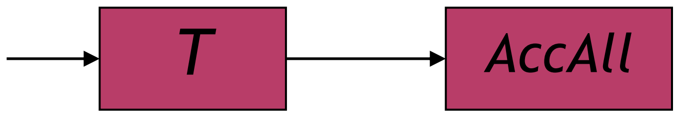


FINITE IS NOT RECURSIVE

Now, we will show that $<T> \in NSA \leftrightarrow <M> \in FINITE$

If $<T> \in NSA$, then T does not accept $<T>$. Then, M does not get to start *AccAll*. Thus, M accepts nothing and L(M) is finite.

If $<T> \notin NSA$, then T accepts $<T>$. Then, M gets pass T, and accept everything. Thus, M accepts everything and L(M) is infinite.



M

f is Turing-computable.
Thus, $NSA \leq FINITE$.
Since NSA is not Recursive, neither is FINITE.

CHECKLIST

- Prove a language is recursive, R.E., or co-R.E.
- Prove closure properties of these classes of languages
- Prove properties of reduction
- Prove a language is not recursive, not R.E., or not co-R.E.
- Prove a problem is decidable
- Prove a problem is undecidable