# CS21
# Decidability and Tractability

Lecture 16

February 9, 2022



## Outline

- Beyond RE and co-RE
- Recursion Theorem
- Gödel Incompleteness Theorem



## Beyond RE and co-RE

- We saw (by a counting argument) that there is *som*… that i… or co-RE.
- We will prove this for a natural language:

  $EQ_{TM} = \{<M_1, M_2> : L(M_1) = L(M_2)\}$
- Recall:
  - $A_{TM}$ is undecidable, but RE
  - co-$A_{TM}$ is undecidable, but coRE

Therefore, not in co-RE

Therefore, not in RE



## Beyond RE and co-RE

**<u>Theorem</u>**: $EQ_{TM}$ is neither RE nor coRE.

**Proof:**

- not RE:
  - reduce from co-$A_{TM}$ (i.e. show co-$A_{TM} \leq_m EQ_{TM}$)
  - what should f(<M, w>) produce?
- not co-RE:
  - reduce from $A_{TM}$ (i.e. show $A_{TM} \leq_m EQ_{TM}$)
  - what should f(<M, w>) produce?



## Beyond RE and co-RE

**Proof** ($A_{TM} \leq_m EQ_{TM}$)

- f(<M, w>) = $<M_1, M_2>$ described below:

TM $M_1$: on input x,
- accept

TM $M_2$: on input x,
- simulate M on input w
- accept if M accepts w

- YES maps to YES?

  $<M, w> \in A_{TM} \Rightarrow L(M_1) = \Sigma^*$ and $L(M_2) = \Sigma^*$ $\Rightarrow f(<M, w>) \in EQ_{TM}$
- NO maps to NO?

  $<M, w> \notin A_{TM} \Rightarrow L(M_1) = \Sigma^*$ and $L(M_2) = \emptyset$ $\Rightarrow f(<M, w>) \notin EQ_{TM}$



## Beyond RE and co-RE

**Proof** (co-$A_{TM} \leq_m EQ_{TM}$)

- f(<M, w>) = $<M_1, M_2>$ described below:

TM $M_1$: on input x,
- reject

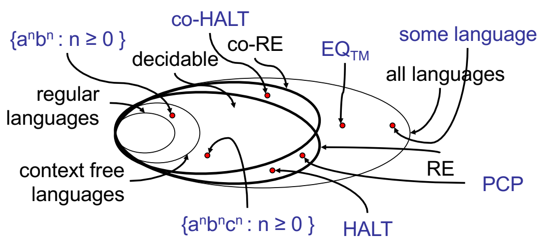
TM $M_2$: on input x,
- simulate M on input w
- accept if M accepts w

- YES maps to YES?

  $<M, w> \in$ co-$A_{TM}$ $\Rightarrow L(M_1) = \emptyset$ and $L(M_2) = \emptyset$ $\Rightarrow f(<M, w>) \in EQ_{TM}$
- NO maps to NO?

  $<M, w> \notin$ co-$A_{TM}$ $\Rightarrow L(M_1) = \emptyset$ and $L(M_2) = \Sigma^*$ $\Rightarrow f(<M, w>) \notin EQ_{TM}$

## Summary






## The Recursion Theorem

- A very useful, and non-obvious, capability of Turing Machines:
  - in the course of computation, can print out a description of itself!
- how is this possible?
  - an example of a program that prints out self:

  Print two copies of the following, the 2nd one in quotes:
  “Print two copies of the following, the 2nd one in quotes:”



## The Recursion Theorem

- Why is this useful?
- Example: slick proof that $A_{TM}$ undecidable
  - assume TM M decides $A_{TM}$
  - construct machine M’ as follows:

on input x,
- obtain own description <M’>
- run M on input <M’, x>
- if M rejects, accept; if M accepts, reject.

if M’ on input x:
- accepts, then M rejects <M’, x>, but then M’ does not accept!
- rejects, then M accepts <M’, x>, but then M’ accepts!



## The Recursion Theorem

- Lemma: there is a computable function

  $q:\Sigma^* \rightarrow \Sigma^*$

  such that q(w) is a description of a TM $P_w$ that prints out w and then halts.
- Proof:
  - on input w, construct TM $P_w$ that has w hard-coded into it; output <$P_w$>



## The Recursion Theorem

- Warm-up: produce a TM SELF that prints out its own description.
- Two parts:
  - Part A:
    - output a description of B
    - pass control to B.
  - Part B:
    - prepend a description of A
    - done



## The Recursion Theorem

- Part A:
  - output a description of B
  - pass control to B.
- Part B:
  - prepend a description of A
  - done

Recall: q(w) is a description of a TM $P_w$ that prints out w and then halts.

Note: <A> = q(<B>)

A
- output <B>

B
- read contents of tape
- apply q to it
- prepend* result to tape

*combine with description on tape to produce a complete TM

## The Recursion Theorem

Note: <A> = q(<B>)

A

- output <B>

B

- read contents of tape
- apply q to it
- prepend result to tape

Recall: q(w) is a description of a TM $P_w$ that prints out w and then halts.

– watch closely as TM AB runs:
– A runs. Tape contents: <B>
– B runs. Tape contents: q(<B>)<B> ) <AB>
– AB is our desired machine SELF.



## The Recursion Theorem

**Theorem**: Let T be a TM that computes fn:

$t: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$

There is a TM R that computes the fn:

$r: \Sigma^* \rightarrow \Sigma^*$

defined as r(w) = t(w, <R>).

- This allows "obtain own description" as valid step in TM program
  – first modify TM so that it takes an additional input (that is own description); use at will



## The Recursion Theorem

**Theorem**: Let T be a TM that computes fn:

$t: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$

There is a TM R that computes the fn:

$r: \Sigma^* \rightarrow \Sigma^*$

defined as r(w) = t(w, <R>).

Proof outline: TM R has 3 parts

Part A: output description of BT
Part B: prepend description of A
Part "T": run TM T



## The Recursion Theorem

Proof details: TM R has 3 parts

Part A: output description of BT
- <A> = q(<BT>)

Part B: prepend description of A
- read contents of tape — <BT>
- apply q to it — q(<BT>) = <A>
- prepend to tape — <ABT>

Part "T": run TM T
- $2^{nd}$ argument on tape is description of R