Consider the empty input halting problem:

11/30/18

$$H' = \{ w_M \mid w_M \text{ encodes a Turing machine } M \text{ that halts when started with empty input} \}$$

<u>Prop:</u> H' is not T-D.

<u>Pf:</u> Suppose for contradiction there is a machine A that decides H'. We use A to construct a machine B that decides the halting language H.

B takes an encoding $w_{M'}$ of a machine M' and an encoding w_v of input v ~~to~~ for M'. B will ~~chang~~ use $w_{M'}$ and w_v to construct an encoding of a machine $M''(v)$ that first write v on the tape and then run M'. Then B feeds $w_{M''(v)}$ as input to A.

If A decides H', then certainly B will decide H.

Since we know H is not T-D this is a contradiction to the existence of A, so H' is not T-D. □

Let ~~H~~ F be the language

$$F = \left\{ w_M \;\middle|\; \begin{array}{l} M \text{ encodes a T-M such} \\ \text{that the language accepted} \\ \text{by } M \text{ is finite} \end{array} \right\}$$

<u>Prop</u>: F is not T-D.

<u>Pf</u>: Suppose F is T-D and A is a TM that decides ~~B~~ F. Construct a machine B ~~H~~ (from A) that decides H as follows.

B takes as input $w_{M'}$ and w_v encoding a TM M' and a string v. B then modifies $w_{M'}$ to create an encoding of $M''(v)$ that first erases its ~~emp~~ input, then write v on the tape, and run M', ~~It~~ with $M''(v)$ accepting if M' halts. Then it feeds $w_{M''(v)}$ to A.

If M' halts on v, then $M''(v)$ accepts every string. If M' doesn't halt on v, $M''(v)$ doesn't accept anything; it won't halt on any input. Hence, if $w_{M'} w_v$ is in H, then $M''(v)$ is infinite, so when $w_{M''(v)}$ is fed to A, A will say "no". If $w_{M'} w_v$ is not in H, A will say "yes". Reversing the answers, ~~A solves th~~ B ~~solves~~ decides the halting problem.

This is a contradiction, so F is not T-D. $\square$

It turns out there is Rice's Theorem: any nontrivial question about TMs (or about T-E languages) is undecidable.

I ~~ha~~ won't tell you exactly what this means. [It's proved by showing a proof like those above can be construc

Unrestricted grammars:

This is a grammar where, in our production rules, the stuff on the left can be anything, not just a single variable as in CF grammars.

E.g. S is start:

$$S \longrightarrow ABxy$$

$$B\cancel{X} \longrightarrow yz$$

$$AB \longrightarrow BxAy$$

$$B\cancel{X} \longrightarrow Ayzx$$

$$B\cancel{y} \longrightarrow BxAy$$

Thm: ~~Unrestrict~~ A language can be gen. by an unrestricted grammar if and only if it is T-E.

One direction is "easy" - ~~if~~ let L be the ~~grammar g~~ language generated by a grammar.

We can construct a TM that

1. get all the results of applying a production to S.

2. "Print out" any that are all terminals (i.e. no variables)

3. Get all the results of applying a production to a string gen. in step 1.

4. "Print out" any that are all terminals

5. etc.

⋮