- You know the drill now: Find students around you to form a ***small group***; use ***all resources*** to help to solve the problems; ***discuss*** your idea with other group member and ***write down*** your own solutions; raise your hand and pull the ***course staffs*** to help; ***submit*** your writeup through Gradescope in *24 hours*.

Our topic for this working session is *undecidable problems*.

Unlike the fooling set construction for regular languages, our strategy to show that certain problems cannot be computed by any Turing machines is by proof of contradiction: Assume that such magical machine exists (along with rainbows and unicorns), we use such machine to either derive a contradiction by feeding its source code back to itself (maybe with some slight modifications), or to solve other existing problems that we already know are undecidable. (The second approach is called a *reduction*, which we will study more in depth in the coming weeks.)

One consequence of the existence of undecidable problems is that it is not always safe to write "if some program exhibits a specific behavior, then ..." in your pseudocode, because you might not have a Turing machine checking that behavior.

Small tip: A Turing machine M ***decides*** a language L if M *halts* and always output the right answer. (We say L is ***decided*** by the machine M. A language is ***undecidable*** if there are no Turing machine that computes that language.) It is important to demostrate that your machine halts on every input possible if you claim the machine solves a problem.

Example. Prove that there is no Turing machine that decides the language

$$\textsc{HaltItself} := \left\{ \langle M \rangle \mid M \text{ is a Turing machine that halts on input } \langle M \rangle \right\}.$$

Solution: Assume for contradiction that some Turing machine M_H decides HALTITSELF. Construct a wrapper function INFLOOP around M_H: Given an input $\langle M \rangle$, INFLOOP feed the input to M_H.

- If M_H accepts, then INFLOOP goes into an infinite loop.
- If M_H rejects, then INFLOOP accepts.

To derive a contradiction, we feed the source code $\langle$INFLOOP$\rangle$ as an input to INFLOOP itself. We consider the following two cases.

- If INFLOOP accepts $\langle$INFLOOP$\rangle$, then M_H must have rejected $\langle$INFLOOP$\rangle$, which implies that INFLOOP does not halt on input $\langle$INFLOOP$\rangle$ by definition that M_H decides HALTITSELF. This is a contradiction.
- If INFLOOP does not halt on input $\langle$INFLOOP$\rangle$, then M_H must have accepted $\langle$INFLOOP$\rangle$, which implies that INFLOOP halts on input $\langle$INFLOOP$\rangle$ by definition that M_H decides HALTITSELF. This is also a contradiction.

Consequently, no M_H can exist that decides HALTITSELF.

Prove that all following languages are undecidable.

1. Prove that there is no Turing machine that decides the language

$$\textsc{NotAcceptItself} := \Big\{ \langle M \rangle \mid M \text{ is a Turing machine which does not accept } \langle M \rangle \Big\}.$$

2. Prove that there is no Turing machine that decides the language

$$\textsc{Accept} := \Big\{ \langle M, w \rangle \mid M \text{ is a Turing machine and } w \in L(M) \Big\}.$$

To think about later: (No submissions needed)

3. Prove that there is no Turing machine that decides the language

$$\textsc{AcceptItself} := \Big\{ \langle M \rangle \mid M \text{ is a Turing machine which accepts } \langle M \rangle \Big\}.$$

4. Prove that there is no Turing machine that decides the language

$$\textsc{NeverAccept} := \Big\{ \langle M \rangle \mid \text{Turing machine } M \text{ never accepts any } w \Big\}.$$

Conceptual question: Wait. Assuming *we* are Turing machines, trying to write a proof for the above practice problems. By the Curry–Howard correspondence, proofs are programs. As we write the solution, we are writing a program checking the correctness of the problem statement. But didn't *we* write the sentence "if machine M rejects input $\langle M \rangle$ ..." several times?