CS 3313
Foundations of Computing:

Lab 8

Decidable vs Undecidable problems

Algorithm = Turing machine that halts on all inputs (always halts)

- Decision problem: the answer is "Yes" or "No"

A problem is undecidable if there is no algorithm (Turing machine that always halts) that solves the problem

- Problem = language
- How do we show a problem is undecidable – need to prove the problem is undecidable

A problem is decidable if there is an algorithm (Turing machine that always halt) to solve the problem

- How do we show a problem is solvable – provide an algorithm that solves the problem
- *Key observation: the algorithm can be deterministic or non-deterministic when we are trying to prove it is solvable/decidable*

Decidable Problems

A problem is *decidable or recursive* if there is an algorithm to answer it

- Recall: An "algorithm," formally, is a TM that halts on all inputs, accepted or not

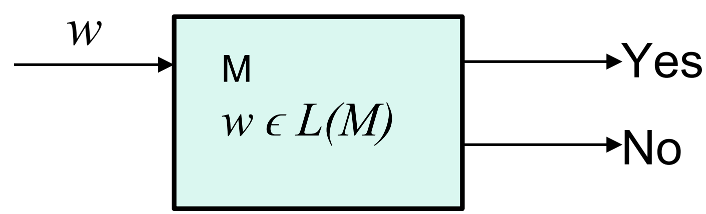
Otherwise, the problem is *undecidable*.

Language is *Turing-recognizable or recursively enumerable* if it is accepted by a TM

- TM halts and accepts if the string is in the language
- However, TM may not halt if the string is not in the language

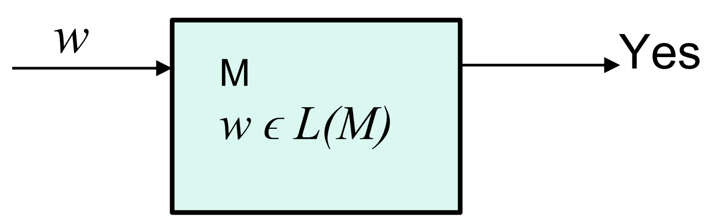
Recall Definitions

Decidable Language: A language L is recursive if there is a Turing machine that accepts the language and ***halts on all inputs***

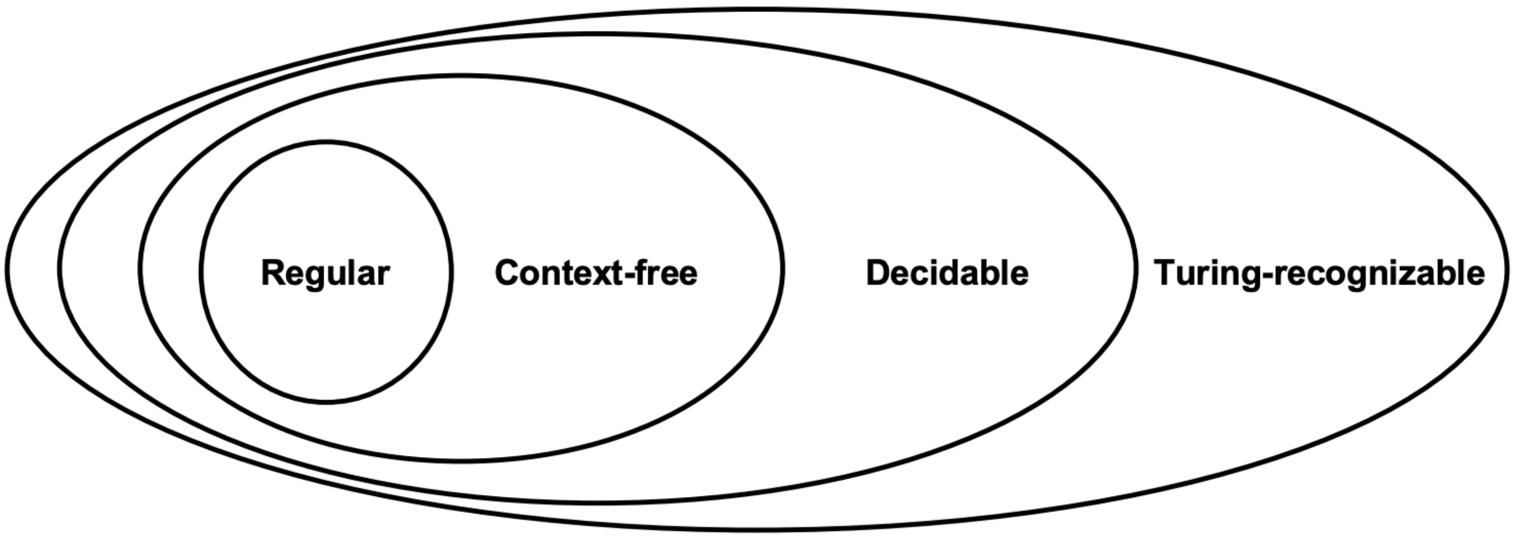


Turing-recognizable Language: if there is a Turing machine that accepts the language by *halting when the input string is in the language*

- The machine may or may not halt if the string is not in the language



Recall the Relationships Among Language Classes



Recall Proof that L_{TM} is Undecidable

$$A_{TM} = \{\langle M, \mathrm{w}\rangle \mid M \text{ is a TM and } M(\mathrm{w}) = 1\}$$

- Assume that A_{TM} is decided by TM H

$$H(\langle \mathrm{M}, \mathrm{w}\rangle) = \begin{cases} \text{accept} & \text{if } M \text{ accepts w} \\ \text{reject} & \text{if } M \text{ does not accept w} \end{cases}$$

- Use H to build a TM D that checks whether a TM M accepts its own description, and then does the opposite:

  On Input $\langle M\rangle$, where M is a TM

  1. Run H on input $\langle M, \langle M\rangle\rangle$
  2. Output the opposite of what H outputs

$$\mathrm{D}(\langle \mathrm{M}\rangle) = \begin{cases} \text{accept} & \text{if } M \text{ does not accept } \langle M\rangle \\ \text{reject} & \text{if } M \text{ accepts } \langle M\rangle \end{cases}$$

- Now consider what happens if we run D on $\langle D\rangle$

$$D(\langle D\rangle) = \begin{cases} \text{accept} & \text{if } D \text{ does not accept } \langle D\rangle \\ \text{reject} & \text{if } D \text{ accepts } \langle D\rangle \end{cases}$$

Decidability…and Reducibility proof technique

Reducibility of a problem A to problem B

Given two problems A and B, problem A is reducible to problem B if an algorithm for solving B can be used to solve problem A

- Therefore, solving A cannot be harder than solving B
- *If A is undecidable and A is reducible to B, then B is undecidable*

Idea: If you had a black box that can solve instances of B, can you solve instances of A using calls to this Black box?

- The black box is the assumed Algorithm for B

Crucial step in the proof is the reduction "algorithm"

- This process should be an "algorithm" – i.e., a TM that always halts

Example: Proof that the halting problem is undecidable

$HALT_{\mathrm{TM}} = \{ <M,w> \mid M \text{ halts on } w \}$

Given any input and any machine, will the machine terminate or run forever?

Assume algorithm B for HALT

Reducibility algorithm R ($HALT_{\mathrm{TM}}$ reducible to A_{TM}):

- Run B(<M,w>), if it rejects then reject – M does not halt on w
- *Otherwise Run M(w) and output what it outputs*
- *This algorithm R decides* A_{TM}

Exercise 1: L = { ⟨*M*⟩ | *M* is a TM and L(*M*)=∅ }

$$L = \{ \langle M \rangle \mid M \text{ is a TM and } L(M)=\emptyset \}$$

Given a Turing machine *M*, does *M* accept any input?

- (i.e., does *M* accept the empty set)

Exercise 2: $L = \{\langle M_1, M_2 \rangle \mid L(M_1) \subseteq L(M_2)\}$ is Undecidable

Given any two Turing machines $M_{1,}$ M_2 is the language accepted by M_1 a subset of language accepted by M_2?

- Hint: Reduce to Exercise 1