# Computational Problems, Mapping Reduction

CSE 105 Week 8 Discussion

# Deadlines and Logistics

- Test 2 next week (week 9)
- Do review quizzes on PrairieLearn
- HW 6 due 12/3/24 at 5pm (week 10)
- Project due 12/11/24 11am (final week)

# Multiple descriptions

**Describing Turing machines** (Sipser p. 185) To define a Turing machine, we could give a

- **Formal definition**: the 7-tuple of parameters including set of states, input alphabet, tape alphabet, transition function, start state, accept state, and reject state; or, $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$
- **Implementation-level definition**: English prose that describes the Turing machine head movements relative to contents of tape, and conditions for accepting / rejecting based on those contents.
- **High-level description**: description of algorithm (precise sequence of instructions), without implementation details of machine. As part of this description, can "call" and run another TM as a subroutine.

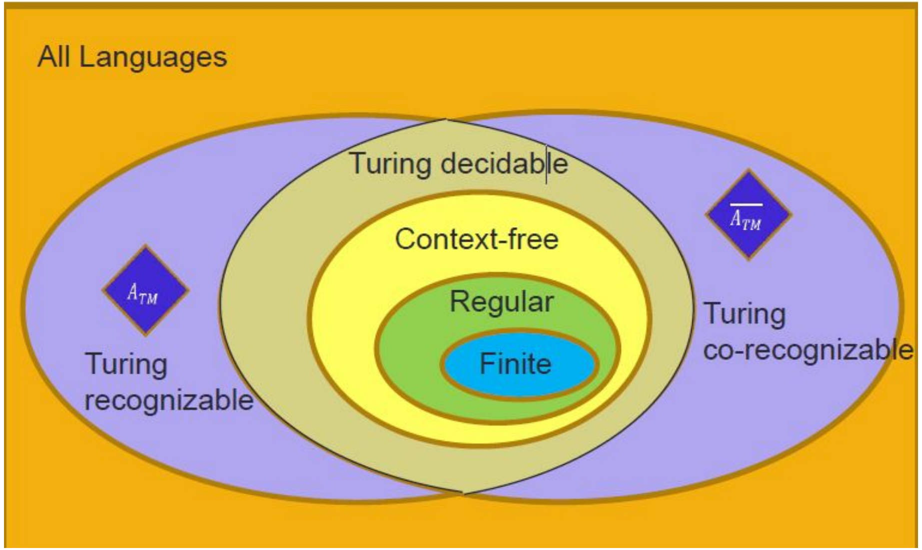
# Multiple descriptions

**Describing Turing machines** (Sipser p. 185) To define a Turing machine, we could give a

- **Formal definition**: the 7-tuple of parameters including set of states, input alphabet, tape alphabet, transition function, start state, accept state, and reject state; or, $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$
- **Implementation-level definition**: English prose that describes the Turing machine head movements relative to contents of tape, and conditions for accepting / rejecting based on those contents.
- **High-level description**: description of algorithm (precise sequence of instructions), without implementation details of machine. As part of this description, can "call" and run another TM as a subroutine.

# Properties of languages

1. Regular
    a. Recognized by a DFA/NFA
    b. Described by a regex
2. Context free
    a. Recognized by a PDA
    b. Generated by a CFG
3. (Turing) Decidable
    a. Can be decided by a Tm
4. (Turing) Recognizable
    a. Can be recognized by a Tm



# Algorithm computation

## Church-Turing Thesis

**Anything** that is **computable** is computable with a **Turing machine** because any method of computation using finite time and finite resources will be **equally expressive** to that of a Turing machine.

# Vocabulary check

1. Are all decidable languages recognizable?
2. If language A is recognizable and language B is decidable, is |A| > |B|
3. If M is a Turing machine, what is <M>?

# Representations of algorithms

To decide these problems, we need to represent the objects of interest as **strings**

For inputs that aren't strings, we have to **encode the object** (represent it as a string) first

To define TM M:

"On input w …

1. ..
2. ..
3. …

**Notation:**
**<O>** is the **string** that represents (encodes) the object O
$\langle O_1, \ldots, O_n \rangle$ is the single string that represents the list of objects $O_1, \ldots, O_n$

# Turing Decidable Languages

Recap : Turing decidable languages are closed under complementation

# Turing Decidable Languages - Recap

1. If a language is decidable if and only if it is co-recognizable and recognizable.
2. If two languages over a fixed alphabet are turing-decidable, then their union is decidable as well
3. If two languages over a fixed alphabet are turing-recognizable, then their union is recognizable as well

# Computational Problems

# Computational problems

| | | |
|---|---|---|
| **Acceptance problem** | | |
| ... for DFA | $A_{DFA}$ | $\{\langle B, w\rangle \mid B$ is a DFA that accepts input string $w\}$ |
| ... for NFA | $A_{NFA}$ | $\{\langle B, w\rangle \mid B$ is a NFA that accepts input string $w\}$ |
| ... for regular expressions | $A_{REX}$ | $\{\langle R, w\rangle \mid R$ is a regular expression that generates input string $w\}$ |
| ... for CFG | $A_{CFG}$ | $\{\langle G, w\rangle \mid G$ is a context-free grammar that generates input string $w\}$ |
| ... for PDA | $A_{PDA}$ | $\{\langle B, w\rangle \mid B$ is a PDA that accepts input string $w\}$ |
| **Language emptiness testing** | | |
| ... for DFA | $E_{DFA}$ | $\{\langle A\rangle \mid A$ is a DFA and $L(A) = \emptyset\}$ |
| ... for NFA | $E_{NFA}$ | $\{\langle A\rangle \mid A$ is a NFA and $L(A) = \emptyset\}$ |
| ... for regular expressions | $E_{REX}$ | $\{\langle R\rangle \mid R$ is a regular expression and $L(R) = \emptyset\}$ |
| ... for CFG | $E_{CFG}$ | $\{\langle G\rangle \mid G$ is a context-free grammar and $L(G) = \emptyset\}$ |
| ... for PDA | $E_{PDA}$ | $\{\langle A\rangle \mid A$ is a PDA and $L(A) = \emptyset\}$ |
| **Language equality testing** | | |
| ... for DFA | $EQ_{DFA}$ | $\{\langle A, B\rangle \mid A$ and $B$ are DFAs and $L(A) = L(B)\}$ |
| ... for NFA | $EQ_{NFA}$ | $\{\langle A, B\rangle \mid A$ and $B$ are NFAs and $L(A) = L(B)\}$ |
| ... for regular expressions | $EQ_{REX}$ | $\{\langle R, R'\rangle \mid R$ and $R'$ are regular expressions and $L(R) = L(R')\}$ |
| ... for CFG | $EQ_{CFG}$ | $\{\langle G, G'\rangle \mid G$ and $G'$ are CFGs and $L(G) = L(G')\}$ |
| ... for PDA | $EQ_{PDA}$ | $\{\langle A, B\rangle \mid A$ and $B$ are PDAs and $L(A) = L(B)\}$ |

# Computational problems for Turing machines

| | | |
|---|---|---|
| **Acceptance problem** for Turing machines | $A_{TM}$ | $\{\langle M, w\rangle \mid M$ is a Turing machine that accepts input string $w\}$ |
| **Language emptiness testing** for Turing machines | $E_{TM}$ | $\{\langle M\rangle \mid M$ is a Turing machine and $L(M) = \emptyset\}$ |
| **Language equality testing** for Turing machines | $EQ_{TM}$ | $\{\langle M_1, M_2\rangle \mid M_1$ and $M_2$ are Turing machines and $L(M_1) = L(M_2)\}$ |

What is $A_{TM}$ ?

A. A Turing machine whose input is codes of TMs and strings.

B. A set of pairs of TMs and strings.

C. A set of strings that encode TMs and strings.

D. Not well defined.

E. I don't know.

# $A_{TM}$ is recognizable but undecidable

- $A_{TM}$ is Turing recognizable
  - We can define a Turing machine that recognizes $A_{TM}$
- $A_{TM}$ is **not** Turing decidable
  - Proof by contradiction (diagonalization proof)

Define the TM N = "On input <M,w>:

1. Simulate M on w.
2. If M accepts, accept. If M rejects, reject."

Which of the following statements is true?

A. N decides $A_{TM}$
B. N recognizes $A_{TM}$
C. N always halts
D. More than one of the above.
E. I don't know

# $A_{TM}$ is recognizable but undecidable

- $A_{TM}$ is Turing recognizable
  - We can define a Turing machine that recognizes $A_{TM}$
- $A_{TM}$ is **not** Turing decidable
  - Proof by contradiction (diagonalization proof)

**Proof**: Suppose **towards a contradiction** that there is a Turing machine that decides $A_{TM}$. We call this presumed machine $M_{ATM}$.

Define a **new** Turing machine using the high-level description:

$D =$" On input $\langle M \rangle$, where $M$ is a Turing machine:

1. Run $M_{ATM}$ on $\langle M, \langle M \rangle \rangle$.
2. If $M_{ATM}$ accepts, reject; if $M_{ATM}$ rejects, accept."

What is the result of the computation of $D$ on $\langle D \rangle$?

# $A_{TM}$ is recognizable but undecidable

- $A_{TM}$ is recognizable.
- $A_{TM}$ is not decidable.
- $\overline{A_{TM}}$ is not recognizable.
- $\overline{A_{TM}}$ is not decidable.

# $A_{TM}$ is recognizable but undecidable

- $A_{TM}$ is recognizable.
- $A_{TM}$ is not decidable.
- $\overline{A_{TM}}$ is not recognizable.
- $\overline{A_{TM}}$ is not decidable.

- A language being **decidable** means "we can say **both yes and no** answers about string **membership** in a language in **finite** time."
- A language being **recognizable** means "we can say yes about string membership in a language in **finite** time."

- Anytime we can prove that a set is undecidable yet recognizable, its complement will be **unrecognizable**.

# Closure claims

## RQ8.5. Closure and nonclosure

Recall the definitions: A language $L$ over an alphabet $\Sigma$ is called **recognizable** if there is some Turing machine $M$ such that $L = L(M)$. A language $L$ over an alphabet $\Sigma$ is called **co-recognizable** if its complement, defined as $\Sigma^* \setminus L = \{x \in \Sigma^* \mid x \notin L\}$, is Turing-recognizable. A language $L$ over an alphabet $\Sigma$ is called {\bf unrecognizable} if there is no Turing machine that recognizes it.

Select all and only true statements below.

- [ ] The class of unrecognizable languages is closed under complementation.
- [ ] The class of recognizable languages is closed under complementation.
- [ ] The class of decidable languages is closed under complementation.
- [ ] The class of undecidable languages is closed under complementation.

# Mapping Reduction

# Motivation

- We want to leverage our previous results of language properties
- Thus, we want to relate the “difficulty level” of one problem to another

If problem X is no harder than problem Y
…and if Y is **decidable**
…then X must also be **decidable**

If problem X is no harder than problem Y
…and if X is **undecidable**
…then Y must also be **undecidable**

“Problem X is no harder than problem Y” means
“Can convert questions about membership in X to questions about membership in Y”

# Mapping reduction & computable functions

Definition: $A$ is **mapping reducible to** $B$ means there is a computable function $f : \Sigma^* \to \Sigma^*$ such that *for all* strings $x$ in $\Sigma^*$,

$$x \in A \qquad \text{if and only if} \qquad f(x) \in B.$$

Notation: when $A$ is mapping reducible to $B$, we write $A \leq_m B$.

*Intuition:* $A \leq_m B$ means $A$ is no harder than $B$, i.e. that the level of difficulty of $A$ is less than or equal the level of difficulty of $B$. **"Can convert questions about membership in A to questions about membership in B"**

**Computable functions**

Definition: A function $f : \Sigma^* \to \Sigma^*$ is a **computable function** means there is some Turing machine such that, for each $x$, on input $x$ the Turing machine halts with exactly $f(x)$ followed by all blanks on the tape

# Computable functions example

**Computable functions**

Definition: A function $f : \Sigma^* \to \Sigma^*$ is a **computable function** means there is some Turing machine such that, for each $x$, on input $x$ the Turing machine halts with exactly $f(x)$ followed by all blanks on the tape

## Define a Turing machine that computes the following function:

The function that maps strings that are not the codes of NFAs to the empty string and that maps strings that code NFAs to the code of a DFA that recognizes the language recognized by the NFA produced by the macro-state construction from Chapter 1.

# “No harder than”?

Which of the following statements are true?

A. $\{0^i1^j \mid i,j >= 0\}$ is no harder than $A_{TM}$
B. $A_{TM}$ is no harder than itself
C. $A_{DFA}$ is no harder than $\{ww \mid w$ is a string over $\{0,1\}\}$
D. $EQ_{DFA}$ is no harder than $A_{DFA}$
E. All of the above

# Mapping reduction practice

## RQ8.10. Properties of mapping reductions

Recall that **mapping reduction** is defined in section 5.3: For languages $A$ and $B$ over $\Sigma$, we say that the problem $A$ mapping reduces to $B$ means there is a computable function $f : \Sigma^* \to \Sigma^*$ such that for all $x \in \Sigma^*$, $x \in A$ iff $f(x) \in B$. A computable function that makes the iff true is said to witness the mapping reduction from $A$ to $B$.

Select all and only the true statements below.

- [ ] For all languages $A$ and $B$, if $A$ mapping reduces to $B$ then $B$ mapping reduces to $A$.
- [ ] Every language mapping reduces to its complement.
- [ ] $\Sigma^*$ mapping reduces to every nonempty language over $\Sigma$.
- [ ] Every decidable language mapping reduces to $\emptyset$.