## Week 6 at a glance

### Textbook reading: Chapter 3

Before Monday, Page 165-166 Introduction to Section 3.1.
Before Wednesday, Example 3.9 on page 173.
Before Friday, Page 184-185 Terminology for describing Turing machines.
Week 7 Monday: No class, in observance of Veterans Day. For Week 7 Wednesday: Introduction to Chapter 4.

**We will be learning and practicing to:**

- Clearly and unambiguously communicate computational ideas using appropriate formalism. Translate across levels of abstraction.
  - Use precise notation to formally define the state diagram of a Turing machine
  - Use clear English to describe computations of Turing machines informally.
    - **Motivate the definition of a Turing machine**
    - **Trace the computation of a Turing machine on given input**
    - **Describe the language recognized by a Turing machine**
    - **Determine if a Turing machine is a decider**
    - **Given an implementation-level description of a Turing machine**
    - **Use high-level descriptions to define and trace Turing machines**
    - **Apply dovetailing in high-level definitions of machines**
  - Give examples of sets that are recognizable and decidable (and prove that they are).
    - **State the definition of the class of recognizable languages**
    - **State the definition of the class of decidable languages**
    - **State the definition of the class of co-recognizable languages**
- Know, select and apply appropriate computing knowledge and problem-solving techniques. Reason about computation and systems.
- Describe and prove closure properties of classes of languages under certain operations.
  - **Apply a general construction to create a new Turing machine from an example one.**
  - **Formalize a general construction from an informal description of it.**
  - **Use general constructions to prove closure properties of the class of decidable langugages.**
  - **Use general constructions to prove closure properties of the class of recognizable langugages.**

**TODO:**

Review Quiz 6 on PrairieLearn (http://us.prairielearn.com), complete by Sunday 11/11/2024

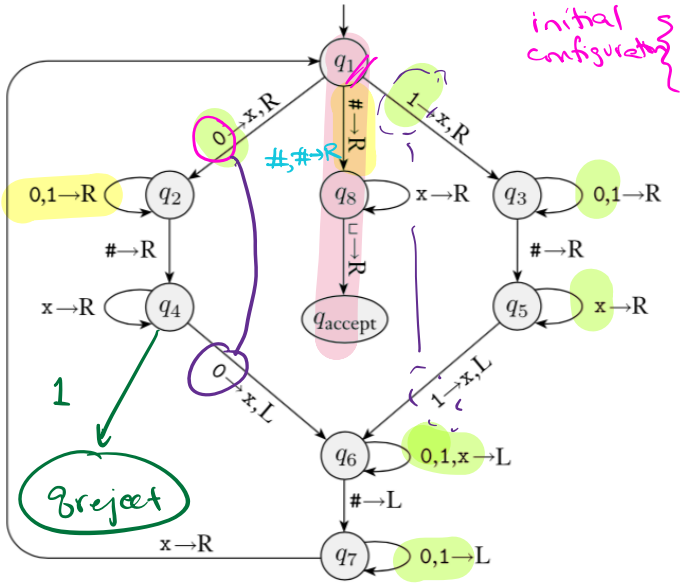
Homework 4 submitted via Gradescope (https://www.gradescope.com/), due Tuesday 10/12/2024

CC BY-NC-SA 2.0 Version November 1, 2024 (1)

# Monday: Descriptions of Turing machines

*Sipser Figure 3.10*

**Conventions in state diagram of TM**: $b \to R$ label means $b \to b, R$ and all arrows missing from diagram represent transitions with output $(q_{reject}, \sqcup, R)$




0→R
1→R
i.e.
0→0,R
1→1,R

Computation on input string 01#01

input

initial configuration

| $q_1$↓ | | | | | blank | blank |
|---|---|---|---|---|---|---|
| 0 | 1 | # | 0 | 1 | ⊔ | ⊔ |
| $q_2$↓ | | | | | | |
| X | 1 | # | 0 | 1 | ⊔ | ⊔ |
| $q_2$↓ | | | | | | |
| X | 1 | # | 0 | 1 | ⊔ | ⊔ |
| $q_4$↓ | | | | | | |
| X | 1 | # | 0 | 1 | ⊔ | ⊔ |
| $q_6$↓ | | | | | | |
| X | 1 | # | X | 1 | ⊔ | ⊔ |
| $q_7$↓ | | | | | | |
| X | 1 | # | X | 1 | ⊔ | ⊔ |
| $q_7$↓ | | | | | | |
| X | 1 | # | X | 1 | ⊔ | ⊔ |
| $q_3$↓ | | | | | | |
| X | X | # | X | 1 | ⊔ | ⊔ |
| $q_5$↓ | | | | | | |
| X | X | # | X | 1 | ⊔ | ⊔ |
| $q_5$↓ | | | | | | |
| X | X | # | X | 1 | ⊔ | ⊔ |
| $q_6$↓ | | | | | | |
| X | X | # | X | X | ⊔ | ⊔ |
| $q_6$↓ | | | | | | |
| X | X | # | X | X | ⊔ | ⊔ |
| $q_7$↓ | | | | | | |
| X | X | # | X | X | ⊔ | ⊔ |
| $q_1$↓ | | | | | | |
| X | X | # | X | X | ⊔ | ⊔ |
| $q_8$↓ | | | | | | |
| X | X | # | X | X | ⊔ | ⊔ |
| $q_8$↓ | | | | | | |
| X | X | # | X | X | ⊔ | ⊔ |
| $q_8$↓ | | | | | | |
| X | X | # | X | X | ⊔ | ⊔ |
| $q_{accept}$↓ | | | | | | |
| X | X | # | X | X | ⊔ | ⊔ |

Tape

Notice 01#1 is rejected.

$\Sigma = \{0, 1, \#\}$

$\Gamma = \{0, 1, \#, X, \sqcup\}$

Implementation level description of this machine:

> Zig-zag across tape to corresponding positions on either side of # to check whether the characters in these positions agree. If they do not, or if there is no #, reject. If they do, cross them off.
>
> Once all symbols to the left of the # are crossed off, check for any un-crossed-off symbols to the right of #; if there are any, reject; if there aren't, accept.

The language recognized by this machine is

$$\{w\#w \mid w \in \{0,1\}^*\}$$

Not context-free!

CC BY-NC-SA 2.0 Version November 1, 2024 (2)

High-level description of this machine is

"On input w

1. for each index $i=0,\ldots$

1a. compare $w_i$ and $w_{|w|+1/2}$

1b. if they're equal increment i and continue.
if not, reject w.

2. if $|w|$ is even, reject.

3. if $w_{|w|+1/2}$ is not #, reject.

4. Accept "

*Recall:* High-level descriptions of Turing machine algorithms are written as indented text within quotation marks. Stages of the algorithm are typically numbered consecutively. The first line specifies the input to the machine, which must be a string.

*Extra practice*

Computation on input string 01#1

| $q_1 \downarrow$ | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | # | 1 | ␣ | ␣ | ␣ |

CC BY-NC-SA 2.0 Version November 1, 2024 (3)

A language $L$ is **recognized by** a Turing machine $M$ means

$L(M)$

$L(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w \}$

$= \{ w \in \Sigma^* \mid$ computation of $M$ on $w$ halts after finitely many steps by entering $q_{accept} \}$

A Turing machine $M$ **recognizes** a language $L$ means

$L = L(M)$.

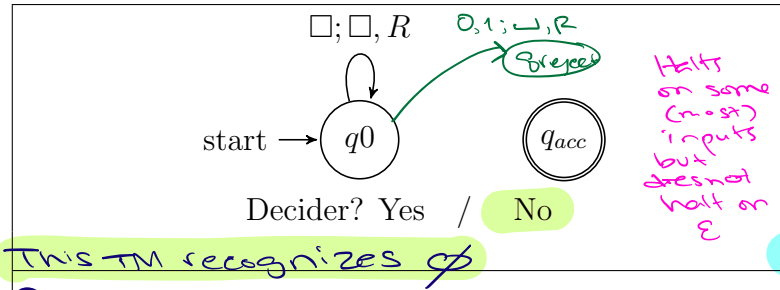
A Turing machine $M$ is a **decider** means for each input $w \in \Sigma^*$, the computation of $M$ on $w$ enters $q_{accept}$ or $q_{reject}$ in finitely many steps, i.e. computation of $M$ on $w$ halts.

A language $L$ is **decided by** a Turing machine $M$ means $M$ is a decider and $L(M) = L$. Equivalently, each string in $L$ is accepted by $M$ and each string not in $L$ is rejected by $M$.

A Turing machine $M$ **decides** a language $L$ means

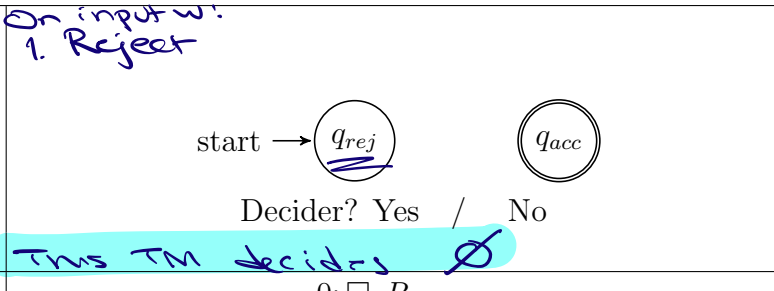
$L$ is decided by $M$.

Fix $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, \sqcup\}$ for the Turing machines with the following state diagrams:



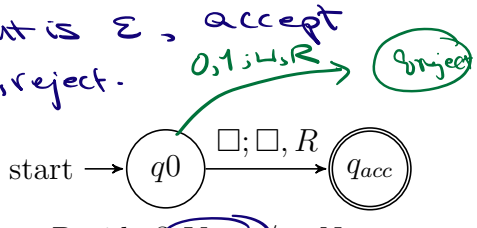

Halts on some (most) inputs but does not halt on ε

This TM recognizes ∅



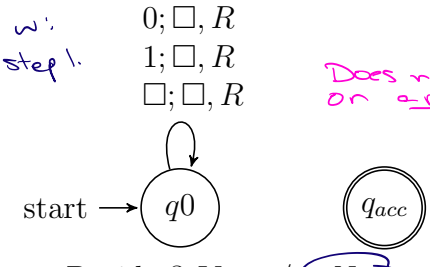

This TM decides ∅

On input w:
If input is ε, accept
otherwise, reject.




This TM decides {ε}

On input w:
1. Go to step 1.




Does not halt on any input.

This TM recognizes ∅

CC BY-NC-SA 2.0 Version November 1, 2024 (4)

## Wednesday: Recognizable and decidable languages

A **Turing-recognizable** language is a set of strings that is the language recognized by some Turing machine. We also say that such languages are recognizable.

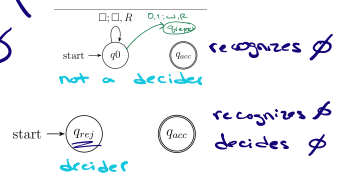
A **Turing-decidable** language is a set of strings that is the language recognized by some decider. We also say that such languages are decidable.

An **unrecognizable** language is a language that is not Turing-recognizable.

An **undecidable** language is a language that is not Turing-decidable.

Examples of languages over $\Sigma = \{0, 1\}$

$\emptyset$




$\Sigma^*$ is recognizable and decidable → $q_{acc}$

$\{w \in \{0,1\} \mid |w| \text{ is a multiple } 3\}$ regular

DFA ⇝ TM so Every regular lang is decidable and hence recognizable.

$\{ww^R \mid w \in \{0,1\}^*\}$ nonregular context-free

[Ch 2] ⇝ Every context-free language is decidable and hence recognizable

**True** or ~~**False**~~: Any decidable language is also recognizable.

Consider $L$ language, assume decidable.
so it is the language recognized by some decider, $M_L$.
But $M_L$ is a Turing machine so $L$ is recognized by some Turing machine, i.e. is recognizable.

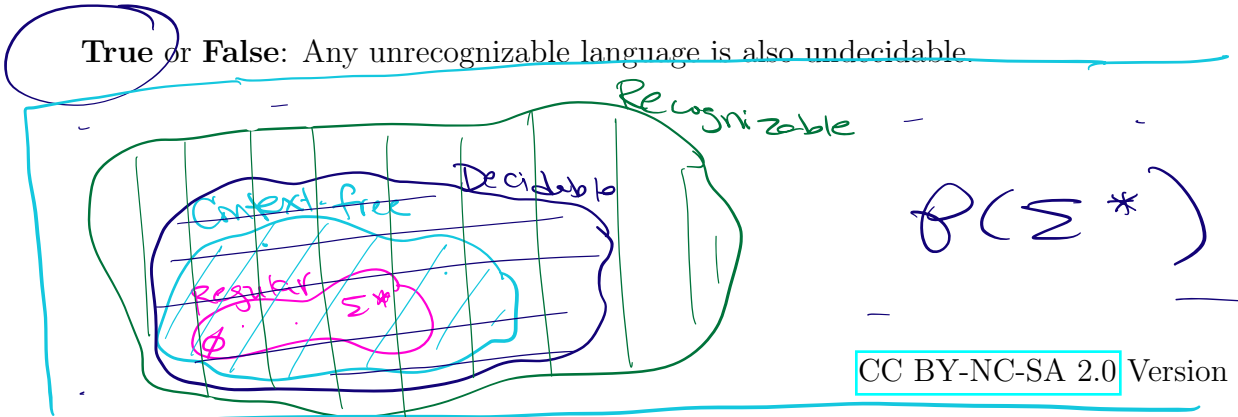
**True** or **False**: Any recognizable language is also decidable.

?

**True** or **False**: Any undecidable language is also unrecognizable.

?

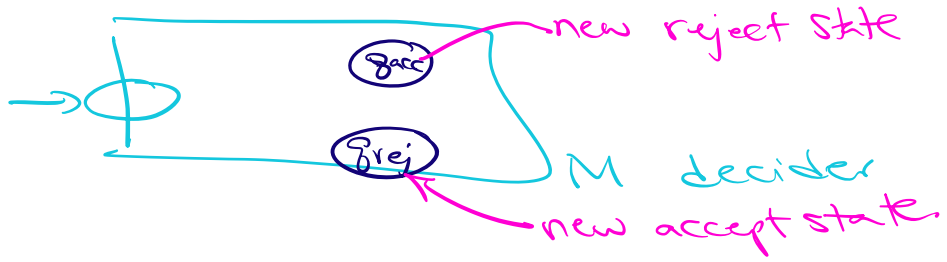
**True** or **False**: Any unrecognizable language is also undecidable.




CC BY-NC-SA 2.0 Version November 1, 2024 (5)

**True** or **False**: The class of Turing-decidable languages is closed under complementation.

Schematic




Using formal definition:

Given $M=(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

Define new TM $M_{new}=(Q, \Sigma, \Gamma, \delta, q_0, q_{reject}, q_{accept})$

and claim $L(M_{new}) = \overline{L(M)}$

Using high-level description:

Given M a decider and let $L(M)=L$

Define new TM

$M_{new}$ = "On input $x$.

1. Run M on $x$. Finitely many steps because M is a decider
2. If M accepts $x$, reject. One step.
3. If M rejects $x$, accept. One step.

Notice (1) $M_{new}$ is decider and (2) $L(M_{new})=\overline{L}$.

**Church-Turing Thesis** (Sipser p. 183): The informal notion of algorithm is formalized completely and correctly by the formal definition of a Turing machine. In other words: all reasonably expressive models of computation are equally expressive with the standard Turing machine.

CC BY-NC-SA 2.0 Version November 1, 2024 (6)

"Yes" in finite time

"Yes" in finite time AND "No" in finite time

## Friday: Closure for the classes of recognizable and decidable languages

Definition: 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.

Examples of co-recognizable languages

"No" in finite time

$\Sigma^*$ : complement is $\emptyset$, which is recognizable

$\emptyset$ : complement is $\Sigma^*$, which is recognizable

any regular language: complement is also regular, therefore decidable, therefore recognizable.

any decidable language: complement is also decidable therefore recognizable.

**Theorem** (Sipser Theorem 4.22): A language is Turing-decidable if and only if both it and its complement are Turing-recognizable.

"Yes" "No"

**Proof, first direction:** Suppose language $L$ is Turing-decidable. WTS that both it and its complement are Turing-recognizable.

Let $L$ be an arbitrary decidable language. There is TM $M_L$ that is a decider and $L(M_L) = L$.

WTS ① $L$ is recognizable, because we have witness TM $M_L$ which by assumption $L(M_L) = L$, as required.

WTS ② $\overline{L}$ is recognizable, because $L$ is decidable and class of decidable languages is closed under complementation $\overline{L}$ is decidable, hence recognizable ∎

**Proof, second direction:** Suppose language $L$ is Turing-recognizable, and so is its complement. WTS that $L$ is Turing-decidable.

$L$ is co-recognizable

Let $L$ be an arbitrary language that is both recognizable and co-recognizable. By definition, this means there are $M$ TM with $L(M) = L$ and TM $M_c$ with $L(M_c) = \overline{L}$. Need to define a TM that is a decider and recognizes $L$.

Consider $D$ = "On input $w$.
1. Run $M$ on $w$. If accepts, accept.
2. Run $M_c$ on $w$. If accepts, reject."

$L(D) = L$ ?

$D$ decider ?

Notation: The complement of a set $X$ is denoted with a superscript $c$, $X^c$, or an overline, $\overline{X}$.

CC BY-NC-SA 2.0 Version November 1, 2024 (7)

Case ① $w \in L$. Then M halts and accepts w.

Tracing D on w. In step 1, run M on w, and since M accepts w, so does D ✓.

Case ②a. $w \notin L$ and M rejects w.

Tracing D on w, in step 1, we run M on w, halt after finitely many step (by case assumption) since M rejects w. Since M doesn't accept, go to step 2 and run $M_c$ on w. which will accept because $L(M_c) = L$ and $w \notin L$. D rejects ✓

Case ②b. $w \notin L$ and M loops on w.

Tracing D on w, in step 1, run M on w by case assumption this subroutine doesn't halt

ie. D loops on w.

Fix: limit number of steps each of M or $M_c$ can run before it cedes control

New definition

$D_{new}$ = "On input w.

1. For $n = 1, 2, 3, \ldots$
2. Run M on w for at most n steps. If M halts before n steps and accepts, accept. If M halts before n steps, and rejects, reject.
3. Run $M_c$ on w for at most n steps. If $M_c$ halts before n steps and accepts, reject. If $M_c$ halts before n steps and rejects, accept."

Proof of correctness: Claim Drew decides L. Consider arbitrary string w.

Case ① $w \in L$. WTS Drew accepts w.

By assumption, M halts on w (and accepts) in finitely many, say $t_M$, many steps.

Case ①a Mc halts on (and rejects) w in fewer than $t_M$ many steps, say $t_c$. Then in the for loop iteration of Drew $n = t_c$, in step 3, Mc will halt and reject, so Drew accepts, as required.

Case ①b Mc doesn't halt on w within $t_M$ steps. Then Drew's computation on w continues until $n = t_M$, at which point, in step 2, M will halt and accept, so Drew accepts, as required.

Case ② $w \notin L$ WTS Drew rejects w.

By assumption, $M_c$ halts on w (and accepts) in finitely many, say $t_c$, many steps.

Case ②a M halts on (and rejects) w in fewer than $t_c$ many steps, say $t_M$. Then in the for loop iteration of Drew $n = t_M$, in step 2, M will halt and reject, so Drew rejects, as required.

Case ②b M doesn't halt on w within $t_c$ steps. Then Drew's computation on w continues until $n = t_c$, at which point, in step 3, $M_c$ will halt and accept, so Drew rejects, as required.

The class of decidable languages is closed under union

**Claim**: If two languages (over a fixed alphabet $\Sigma$) are Turing-decidable, then their union is as well.

**Proof**:

Let $L_1, L_2$ be arbitrary decidable languages. There are $D_1, D_2$ deciders such that $L(D_1) = L_1$ and $L(D_2) = L_2$. Define the new Turing machine:

"D = On input w

1. Run $D_1$ on w.
   If accepts, accept.
   If rejects, go to step 2.
2. Run $D_2$ on w.
   If accepts, accept.
   If rejects, rejects."

Proof of correctness left as exercise.

WTS for each string w, if $w \in L_1 \cup L_2$ then D accepts w and if $w \notin L_1 \cup L_2$ then D rejects w.

CC BY-NC-SA 2.0 

# The class of recognizable languages is closed under union

**Claim**: If two languages (over a fixed alphabet $\Sigma$) are Turing-recognizable, then their union is as well.

**Proof**:

~~M = "On input w~~
~~1. Run $M_1$ on w.~~
~~If accepts, accept.~~
~~If rejects, go to step 2.~~
~~<maybe $M_1$ loops on w... uh oh!>~~
~~2 Run $M_2$ on w.~~
~~If accepts, accept.~~
~~If rejects, rejects.~~

Let $L_1, L_2$ be arbitrary recognizable languages.
There are Turing machines $M_1$, $M_2$ with $L(M_1) = L_1$ and $L(M_2) = L_2$.
Define the new Turing machine:
M = "On input w

1. For $n = 1, 2, 3, \ldots$.
2. Run $M_1$ on w for at most n steps.
   If $M_1$ accepts w within these n steps, accept.
   if not, go to step 3.
3. Run $M_2$ on w for at most n steps.
   If $M_2$ accepts w within these n steps, accept.
   if not, go to step 3. "

Proof of correctness left as exercise.
WTS for each string w, if $w \in L_1 \cup L_2$ then D accepts w and if $w \notin L_1 \cup L_2$ then D does not accept w.

CC BY-NC-SA 2.0 