CSC363 Tutorial #3

Decidable and Recognizable sets

February 1, 2023

Things covered in this tutorial

Things covered in this tutorial

- ⋆ What's a language?

Things covered in this tutorial

- ⋆ What's a language?
- ⋆ What's a decidable language? What's a recognizable language?

Things covered in this tutorial

- ⋆ What's a language?
- ⋆ What's a decidable language? What's a recognizable language?
- ⋆ How many synonyms are there for "decidable" and "recognizable"?

Things covered in this tutorial

- ⋆ What's a language?
- ⋆ What's a decidable language? What's a recognizable language?
- ⋆ How many synonyms are there for "decidable" and "recognizable"?
- ⋆ How do I show that something is decidable or recognizable?

Things covered in this tutorial

- ⋆ What's a language?
- ⋆ What's a decidable language? What's a recognizable language?
- ⋆ How many synonyms are there for "decidable" and "recognizable"?
- ⋆ How do I show that something is decidable or recognizable?
- ⋆ What's an enumerator?
- ⋆ Can I get a hint for A2?

Things covered in this tutorial

- ⋆ What's a language?
- ⋆ What's a decidable language? What's a recognizable language?
- ⋆ How many synonyms are there for "decidable" and "recognizable"?
- ⋆ How do I show that something is decidable or recognizable?
- ⋆ What's an enumerator?
- ⋆ Can I get a hint for A2? No, but you may cite these slides for your homework. (You still have to prove everything.)[1]

[1] Citation: Paul "sjorv" Zhang. "Sussy Tutorial #3. Decidable and Recognizable Sets.". EXTRACT_MONEY_FROM_STUDENTS Publishing, 2023.

Turing Machines: Review

Turing Machines: Review

Let T be a Turing machine. Recall that given a string input $w \in \Sigma^*$, one of the following will happen:

Turing Machines: Review

Let T be a Turing machine. Recall that given a string input $w \in \Sigma^*$, one of the following will happen:

- ⋆ $T(w)$ accepts, and outputs whatever is left on the tape.

Turing Machines: Review

Let T be a Turing machine. Recall that given a string input $w \in \Sigma^*$, one of the following will happen:

- $\star$ $T(w)$ accepts, and outputs whatever is left on the tape.
- $\star$ $T(w)$ rejects, and outputs whatever is left on the tape.

Turing Machines: Review

Let T be a Turing machine. Recall that given a string input $w \in \Sigma^*$, one of the following will happen:

- ⋆ $T(w)$ accepts, and outputs whatever is left on the tape.
- ⋆ $T(w)$ rejects, and outputs whatever is left on the tape.
- ⋆ $T(w)$ loops.

Turing Machines: Review

Let T be a Turing machine. Recall that given a string input $w \in \Sigma^*$, one of the following will happen:

- ⋆ $T(w)$ accepts, and outputs whatever is left on the tape.
- ⋆ $T(w)$ rejects, and outputs whatever is left on the tape.
- ⋆ $T(w)$ loops.

Church-Turing Thesis:
Anything that your computer can do, so can a Turing machine.

Turing Machines: Review

Let T be a Turing machine. Recall that given a string input $w \in \Sigma^*$, one of the following will happen:

- $\star$ $T(w)$ accepts, and outputs whatever is left on the tape.
- $\star$ $T(w)$ rejects, and outputs whatever is left on the tape.
- $\star$ $T(w)$ loops.

Church-Turing Thesis:
Anything that your computer can do, so can a Turing machine.

For this tutorial, we will use the Church-Turing Thesis to write pseudocode instead of low-level TMs.

Languages

If you still remember from CSC263, recall what a *language* is.

Languages

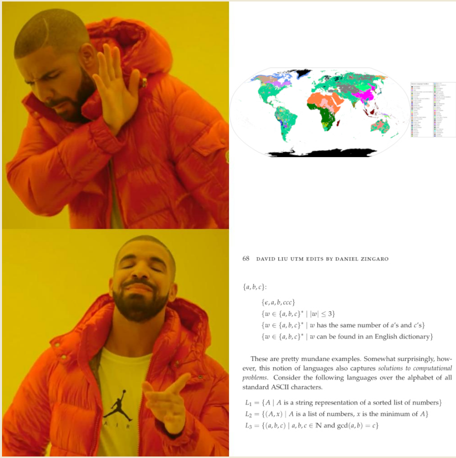
If you still remember from CSC263, recall what a *language* is.

Let Σ be an alphabet. Then Σ^* is the set of all finite strings using characters from Σ.

Languages

If you still remember from CSC263, recall what a *language* is.

Let Σ be an **alphabet**. Then Σ^* is the set of all finite strings using characters from Σ. A **language** is any subset of Σ^*.



Decidable and Recognizable

Let $L \subseteq \Sigma^*$ be a language.

Decidable and Recognizable

Let $L \subseteq \Sigma^*$ be a language. We say L is decidable if there is a Turing machine M (called the decider) such that for any input $s \in \Sigma^*$:

Decidable and Recognizable

Let $L \subseteq \Sigma^*$ be a language. We say L is decidable if there is a Turing machine M (called the decider) such that for any input $s \in \Sigma^*$:

- $\star$ If $s \in L$, then M accepts s.

Decidable and Recognizable

Let $L \subseteq \Sigma^*$ be a language. We say L is decidable if there is a Turing machine M (called the decider) such that for any input $s \in \Sigma^*$:

- ⋆ If $s \in L$, then M accepts s.
- ⋆ If $s \notin L$, then M rejects s.

Decidable and Recognizable

Let $L \subseteq \Sigma^*$ be a language. We say L is decidable if there is a Turing machine M (called the decider) such that for any input $s \in \Sigma^*$:

- ⋆ If $s \in L$, then M accepts s.
- ⋆ If $s \notin L$, then M rejects s.

Decidable and Recognizable

Let $L \subseteq \Sigma^*$ be a language. We say L is decidable if there is a Turing machine M (called the decider) such that for any input $s \in \Sigma^*$:

- ⋆ If $s \in L$, then M accepts s.
- ⋆ If $s \notin L$, then M rejects s.

We say L is recognizable if there is a Turing machine M (called the recognizer) such that for any input $s \in \Sigma^*$:

- ⋆ If $s \in L$, then M accepts s.
- ⋆ If $s \notin L$, then M rejects or loops on s.

Decidable and Recognizable

Let $L \subseteq \Sigma^*$ be a language. We say L is decidable if there is a Turing machine M (called the decider) such that for any input $s \in \Sigma^*$:

- ⋆ If $s \in L$, then M accepts s.
- ⋆ If $s \notin L$, then M rejects s.

We say L is recognizable if there is a Turing machine M (called the recognizer) such that for any input $s \in \Sigma^*$:

- ⋆ If $s \in L$, then M accepts s.
- ⋆ If $s \notin L$, then M rejects or loops on s.

Note the difference! M always has to halt in order to be a decider.

Decidable and Recognizable

Let $L \subseteq \Sigma^*$ be a language. We say L is decidable if there is a Turing machine M (called the decider) such that for any input $s \in \Sigma^*$:

- ⋆ If $s \in L$, then M accepts s.
- ⋆ If $s \notin L$, then M rejects s.

Example:

Decidable and Recognizable

Let $L \subseteq \Sigma^*$ be a language. We say L is decidable if there is a Turing machine M (called the decider) such that for any input $s \in \Sigma^*$:

- $\star$ If $s \in L$, then M accepts s.
- $\star$ If $s \notin L$, then M rejects s.

Example: Let $\Sigma = \{0, 1\}$, and $L = \{0^n 1^n : n \in \mathbb{N}\}$.[2] Show that L is decidable.

[2] That is, $L = \{\epsilon, 01, 0011, 000111, \ldots\}$.

Decidable and Recognizable

Let $L \subseteq \Sigma^*$ be a language. We say L is decidable if there is a Turing machine M (called the decider) such that for any input $s \in \Sigma^*$:

- ⋆ If $s \in L$, then M accepts s.
- ⋆ If $s \notin L$, then M rejects s.

Example: Let $\Sigma = \{0, 1\}$, and $L = \{0^n 1^n : n \in \mathbb{N}\}$.[2] Show that L is decidable.

Proof. The following is the pseudocode of a decider M for L:

[2] That is, $L = \{\epsilon, 01, 0011, 000111, \ldots\}$.

Decidable and Recognizable

Let $L \subseteq \Sigma^*$ be a language. We say L is decidable if there is a Turing machine M (called the decider) such that for any input $s \in \Sigma^*$:

- $\star$ If $s \in L$, then M accepts s.
- $\star$ If $s \notin L$, then M rejects s.

Example: Let $\Sigma = \{0, 1\}$, and $L = \{0^n 1^n : n \in \mathbb{N}\}$.[2] Show that L is decidable.

Proof. The following is the pseudocode of a decider M for L:

```
M(w):
  n = length(w)
  if n is odd:
    reject
  for i in 0 to (n/2 - 1):
    if w[n/2] != 0 or w[n/2 + i] != 1:
      reject
  accept
```

[2] That is, $L = \{\epsilon, 01, 0011, 000111, \ldots\}$.

Decidable and Recognizable

Decidable languages are also known as recursive languages.

Decidable and Recognizable

Decidable languages are also known as recursive languages.

Here are some synonyms for recognizable:

- ⋆ Listable.
- ⋆ Recursively enumerable (r.e.).
- ⋆ Computably enumerable (c.e.).
- ⋆ Partially decidable.
- ⋆ Σ^0_1.

Worksheet time!

Try doing Exercise -1. Here's the definition of *decidable* again:

Let $L \subseteq \Sigma^*$ be a language. We say L is decidable if there is a Turing machine M (called the decider) such that for any input $s \in \Sigma^*$:

Worksheet time!

Try doing Exercise -1. Here's the definition of *decidable* again:

Let $L \subseteq \Sigma^*$ be a language. We say L is decidable if there is a Turing machine M (called the decider) such that for any input $s \in \Sigma^*$:

- ⋆ If $s \in L$, then M accepts s.
- ⋆ If $s \notin L$, then M rejects s.

If you are done, try Exercise 0 as well.

Decidable Languages

Here are some more examples of decidable languages:

[3] Recall from CSC236 that regular language is a language that is decidable by a DFA.

Decidable Languages

Here are some more examples of decidable languages:

- ⋆ Any regular language.[3]

[3] Recall from CSC236 that regular language is a language that is decidable by a DFA.

Decidable Languages

Here are some more examples of decidable languages:

- ⋆ Any regular language.[3]
- ⋆ $L = \{w : \text{Someone in this room has a credit card with number } w\}$.

[3] Recall from CSC236 that regular language is a language that is decidable by a DFA.

Decidable Languages

Here are some more examples of decidable languages:

- ⋆ Any regular language.[3]
- ⋆ $L = \{w : \text{Someone in this room has a credit card with number } w\}$.
- ⋆ In fact, any finite language L is decidable!

[3]Recall from CSC236 that regular language is a language that is decidable by a DFA.

Decidable Languages

Here are some more examples of decidable languages:

- ⋆ Any regular language.[3]
- ⋆ $L = \{w : \text{Someone in this room has a credit card with number } w\}$.
- ⋆ In fact, any finite language L is decidable!
- ⋆ $L = \{w : w \text{ is a grammatically correct English sentence}\}$.

[3] Recall from CSC236 that regular language is a language that is decidable by a DFA.

Decidable Languages

Here are some more examples of decidable languages:

- ⋆ Any regular language.[3]
- ⋆ $L = \{w : \text{Someone in this room has a credit card with number } w\}$.
- ⋆ In fact, any finite language L is decidable!
- ⋆ $L = \{w : w \text{ is a grammatically correct English sentence}\}$.
- ⋆ $L = \{w : w \text{ is a valid solution to the P vs NP problem}\}$.

[3] Recall from CSC236 that regular language is a language that is decidable by a DFA.

Decidable Languages

Here are some more examples of decidable languages:

- ⋆ Any regular language.[3]
- ⋆ $L = \{w : \text{Someone in this room has a credit card with number } w\}$.
- ⋆ In fact, any finite language L is decidable!
- ⋆ $L = \{w : w \text{ is a grammatically correct English sentence}\}$.
- ⋆ $L = \{w : w \text{ is a valid solution to the P vs NP problem}\}$.

and (infinitely) many more!

[3] Recall from CSC236 that regular language is a language that is decidable by a DFA.

Decidable Languages

Here are some more examples of decidable languages:

- ⋆ Any regular language.[3]
- ⋆ $L = \{w : \text{Someone in this room has a credit card with number } w\}$.
- ⋆ In fact, any finite language L is decidable!
- ⋆ $L = \{w : w \text{ is a grammatically correct English sentence}\}$.
- ⋆ $L = \{w : w \text{ is a valid solution to the P vs NP problem}\}$.

and (infinitely) many more!

Are there any non-decidable languages?

[3] Recall from CSC236 that regular language is a language that is decidable by a DFA.

Decidable Languages

Are there any non-decidable languages?

[4] Recall that a Diophantine equation is a multivariable equation like $x^3 + 3xyz - w^{420} = 2$.

Decidable Languages

Are there any non-decidable languages? Yes.

$$L = \{p : p \text{ is a Diophantine equation that has a natural solution}\}$$

is not decidable![4]

[4] Recall that a Diophantine equation is a multivariable equation like $x^3 + 3xyz - w^{420} = 2$.

Decidable Languages

Are there any non-decidable languages? Yes.

$$L = \{p : p \text{ is a Diophantine equation that has a natural solution}\}$$

is not decidable![4]

Source: trust me bro.[5]

[4] Recall that a Diophantine equation is a multivariable equation like $x^3 + 3xyz - w^{420} = 2$.

[5] You can look up "Hilbert's tenth problem".

Recognizable Languages

$$L = \{p : p \text{ is a Diophantine equation that has a natural solution}\}$$

is not decidable![6]

[6] Recall that a Diophantine equation is a multivariable equation like

Recognizable Languages

$$L = \{p : p \text{ is a Diophantine equation that has a natural solution}\}$$

is not decidable![6] L is recognizable though.

[6] Recall that a Diophantine equation is a multivariable equation like

Recognizable Languages

$$L = \{p : p \text{ is a Diophantine equation that has a natural solution}\}$$

is not decidable![6] L is recognizable though.

Source:

```
M(p):
  if p isn't a valid Diophantine equation:
    reject
  n = number of variables in p
  s = 0
  while True:
    for all x1, x2, ..., xn
    with x1 + x2 + ... + xn = s:
      if (x1, x2, ... xn) is a solution to p:
        accept
      s += 1
```

[6] Recall that a Diophantine equation is a multivariable equation like

Recognizable Languages

For example, if p is the equation $3x^5 - xy + y^2 = 3$, $M(p)$ will:

Recognizable Languages

For example, if p is the equation $3x^5 - xy + y^2 = 3$, $M(p)$ will:

- ⋆ Check if $x = 0, y = 0$ is a solution. If not,
- ⋆ Check if $x = 0, y = 1$ is a solution. If not,
- ⋆ Check if $x = 1, y = 0$ is a solution. If not,
- ⋆ Check if $x = 0, y = 2$ is a solution. If not,
- ⋆ Check if $x = 1, y = 1$ is a solution. If not,
- ⋆ Check if $x = 2, y = 0$ is a solution. If not,
- ⋆ Check if $x = 0, y = 3$ is a solution. If not,
- ⋆ Check if $x = 1, y = 2$ is a solution. If not,
- ⋆ Check if $x = 2, y = 1$ is a solution. If not,
- ⋆ Check if $x = 3, y = 0$ is a solution. If not,
- ⋆ Check if $x = 0, y = 4$ is a solution. If not,
- ⋆ Check if $x = 1, y = 3$ is a solution. If not,
- ⋆ Check if $x = 2, y = 2$ is a solution. If not,
- ⋆ Check if $x = 3, y = 1$ is a solution. If not,
- ⋆ Check if $x = 4, y = 0$ is a solution. If not,
- ⋆ Check if $x = 0, y = 5$ is a solution. If not,
- ⋆ Check if $x = 1, y = 4$ is a solution. If not,

Recognizable Languages

If $3x^5 - xy + y^2 = 3$ has a natural solution $(x, y) \in \mathbb{N}^2$, $M(p)$ will

Recognizable Languages

If $3x^5 - xy + y^2 = 3$ has a natural solution $(x, y) \in \mathbb{N}^2$, $M(p)$ will eventually accept.

Recognizable Languages

If $3x^5 - xy + y^2 = 3$ has a natural solution $(x, y) \in \mathbb{N}^2$, $M(p)$ will eventually accept.

If $3x^5 - xy + y^2 = 3$ has no natural solutions, $M(p)$ will

Recognizable Languages

If $3x^5 - xy + y^2 = 3$ has a natural solution $(x, y) \in \mathbb{N}^2$, $M(p)$ will eventually accept.

If $3x^5 - xy + y^2 = 3$ has no natural solutions, $M(p)$ will run forever, i.e. loop.

Recognizable Languages

Question: Why couldn't have $M(p)$ done the following sequence of checks, with p being the equation $3x^5 - ky + y^2 = 3$?

- ⋆ Check if $x = 0, y = 0$ is a solution. If not,
- ⋆ Check if $x = 0, y = 1$ is a solution. If not,
- ⋆ Check if $x = 0, y = 2$ is a solution. If not,
- ⋆ Check if $x = 0, y = 3$ is a solution. If not,
- ⋆ Check if $x = 0, y = 4$ is a solution. If not,
- ⋆ Check if $x = 0, y = 5$ is a solution. If not,
- ⋆ Check if $x = 0, y = 6$ is a solution. If not,
- ⋆ Check if $x = 0, y = 7$ is a solution. If not,
- ⋆ Check if $x = 0, y = 8$ is a solution. If not,
- ⋆ Check if $x = 0, y = 9$ is a solution. If not,
- ⋆ Check if $x = 0, y = 10$ is a solution. If not,
- ⋆ Check if $x = 0, y = 11$ is a solution. If not,
- ⋆ Check if $x = 0, y = 12$ is a solution. If not,
- ⋆ Check if $x = 0, y = 13$ is a solution. If not,
- ⋆ Check if $x = 0, y = 14$ is a solution. If not,
- ⋆ Check if $x = 0, y = 15$ is a solution. If not,

Recognizable Languages

Another recognizable but not decidable language:

Recognizable Languages

Another recognizable but not decidable language:

- $\star$ $L = \{M\#x : M \text{ is a TM that halts on } x\}$.

Recognizable Languages

Another recognizable but not decidable language:

- $\star$ $L = \{M\#x : M$ is a TM that halts on $x\}$. This is known as the **Halting problem**.

Recognizable Languages

Another recognizable but not decidable language:

- ⋆ $L = \{M\#x : M \text{ is a TM that halts on } x\}$. This is known as the **Halting problem**.

Some non-recognizable languages:

Recognizable Languages

Another recognizable but not decidable language:

- ⋆ $L = \{M\#x : M \text{ is a TM that halts on } x\}$. This is known as the **Halting problem**.

Some non-recognizable languages:

- ⋆ $L = \{M\#x : M \text{ is a TM that loops on } x\}$.

Recognizable Languages

Another recognizable but not decidable language:

- ⋆ $L = \{M\#x : M \text{ is a TM that halts on } x\}$. This is known as the **Halting problem**.

Some non-recognizable languages:

- ⋆ $L = \{M\#x : M \text{ is a TM that loops on } x\}$.
- ⋆ $L = \{M : M \text{ is a TM that halts on all inputs}\}$.

Enumerators (possibly useful for assignment!)

Enumerators (possibly useful for assignment!)

Let L be a language.

Enumerators (possibly useful for assignment!)

Let L be a language. An **enumerator** for L is a program that accepts no input and:

Enumerators (possibly useful for assignment!)

Let L be a language. An **enumerator** for L is a program that accepts no input and:

- ⋆ Eventually prints out w for every $w \in L$ (possibly with duplicates).

Enumerators (possibly useful for assignment!)

Let L be a language. An **enumerator** for L is a program that accepts no input and:

- ⋆ Eventually prints out w for every $w \in L$ (possibly with duplicates).
- ⋆ Never prints out anything other than strings in L.

Enumerators (possibly useful for assignment!)

Let L be a language. An enumerator for L is a program that accepts no input and:

- ⋆ Eventually prints out w for every $w \in L$ (possibly with duplicates).
- ⋆ Never prints out anything other than strings in L.

Example: the following program is an enumerator for the language $\{0^n 1^n : n \in \mathbb{N}\}$.

```
def enum():
  n = 0
  while True:
    print('0'*n + '1'*n)
    n += 1
```

Enumerators (possibly useful for assignment!)

Let L be a language. An enumerator for L is a program that accepts no input and:

- ⋆ Eventually prints out w for every $w \in L$ (possibly with duplicates).
- ⋆ Never prints out anything other than strings in L.

Example: the following program is an enumerator for the language $\{0^n 1^n : n \in \mathbb{N}\}$.

```
def enum():
  n = 0
  while True:
    print('0'*n + '1'*n)
    n += 1
```

Try running `enum.py` on your computer!

Enumerators (possibly useful for assignment!)

Here's something that might be useful. Remember to cite!

Enumerators (possibly useful for assignment!)

Here's something that might be useful. Remember to cite!

Theorem (Sipser 3.21)

A language L is recognizable if and only if it has an enumerator.

Enumerators (possibly useful for assignment!)

Here's something that might be useful. Remember to cite!

Theorem (Sipser 3.21)

A language L is recognizable if and only if it has an enumerator.

Proof. First, suppose L has an enumerator `enum`. Define M as follows:

```
M(x):
  run enum in the background
  while True:
    if enum has printed x:
      accept
```

Enumerators (possibly useful for assignment!)

Here's something that might be useful. Remember to cite!

Theorem (Sipser 3.21)

A language L is recognizable if and only if it has an enumerator.

Proof. First, suppose L has an enumerator `enum`. Define M as follows:

```
M(x):
  run enum in the background
  while True:
    if enum has printed x:
      accept
```

⋆ If $x \in L$, then x is eventually printed by the enumerator, and $M(x)$ accepts.

Enumerators (possibly useful for assignment!)

Here's something that might be useful. Remember to cite!

Theorem (Sipser 3.21)

A language L is recognizable if and only if it has an enumerator.

Proof. First, suppose L has an enumerator `enum`. Define M as follows:

```
M(x):
  run enum in the background
  while True:
    if enum has printed x:
      accept
```

- ⋆ If $x \in L$, then x is eventually printed by the enumerator, and $M(x)$ accepts.
- ⋆ If $x \notin L$, then x is never printed, and $M(x)$ loops.

Enumerators (possibly useful for assignment!)

Here's something that might be useful. Remember to cite!

Theorem (Sipser 3.21)

A language L is recognizable if and only if it has an enumerator.

Proof. First, suppose L has an enumerator `enum`. Define M as follows:

```
M(x):
  run enum in the background
  while True:
    if enum has printed x:
      accept
```

- ⋆ If $x \in L$, then x is eventually printed by the enumerator, and $M(x)$ accepts.
- ⋆ If $x \notin L$, then x is never printed, and $M(x)$ loops.

Thus M recognizes L.

Enumerators (possibly useful for assignment!)

Here's something that might be useful. Remember to cite!

Theorem (Sipser 3.21)

A language L is recognizable if and only if it has an enumerator.

Proof. Now, suppose L has a recognizer M. Define `enum` as follows:

Enumerators (possibly useful for assignment!)

Here's something that might be useful. Remember to cite!

Theorem (Sipser 3.21)

A language L is recognizable if and only if it has an enumerator.

Proof. Now, suppose L has a recognizer M. Define `enum` as follows:

Enumerators (possibly useful for assignment!)

Here's something that might be useful. Remember to cite!

Theorem (Sipser 3.21)

A language L is recognizable if and only if it has an enumerator.

Proof. Now, suppose L has a recognizer M. Define `enum` as follows:

```
enum():
  n = 0
  while True:
    for all strings w of length <= n:
        run M(w)
        if M(w) accepts:
            print(w)
    n += 1
```

Question: The above is not a valid enumerator of L. Why?

Enumerators (possibly useful for assignment!)

Here's something that might be useful. Remember to cite!

Theorem (Sipser 3.21)

A language L is recognizable if and only if it has an enumerator.

Proof. Now, suppose L has a recognizer M. Define `enum` as follows:

```
enum():
  n = 0
  while True:
    for all strings w of length <= n:
        run M(w) for n steps
        if M(w) accepts:
            print(w)
    n += 1
```

Task:

- ⋆ Given any $w \notin L$, Why does `enum()` never print out w?
- ⋆ Given any $w \in L$, Why does `enum()` eventually print out w?

Useful trick

```
enum():
  n = 0
  while True:
    for all strings w of length <= n:
        run M(w) for n steps
        if M(w) accepts:
            print(w)
    n += 1
```

This idea of "running for *n* steps, then increasing the maximum time allowed and trying again" is very useful in CSC363!