

Theory of Computation

Chapter 4 & 5

Decidable Languages
&
Undecidable Languages



School of Engineering | Computer Science

Alonzo Church

1903-1995



- Mathematician and logician
- Co-creator of Church-Turing thesis
 - Provability concept, problems exist that computers **cannot** solve
- Church's theorem (first order logic undecidable)
- Creator of lambda calculus
 - Church and Turing proved Church's lambda calculus and Turing's Turing machine were equivalent
- Supervised Turing's PhD work at Princeton

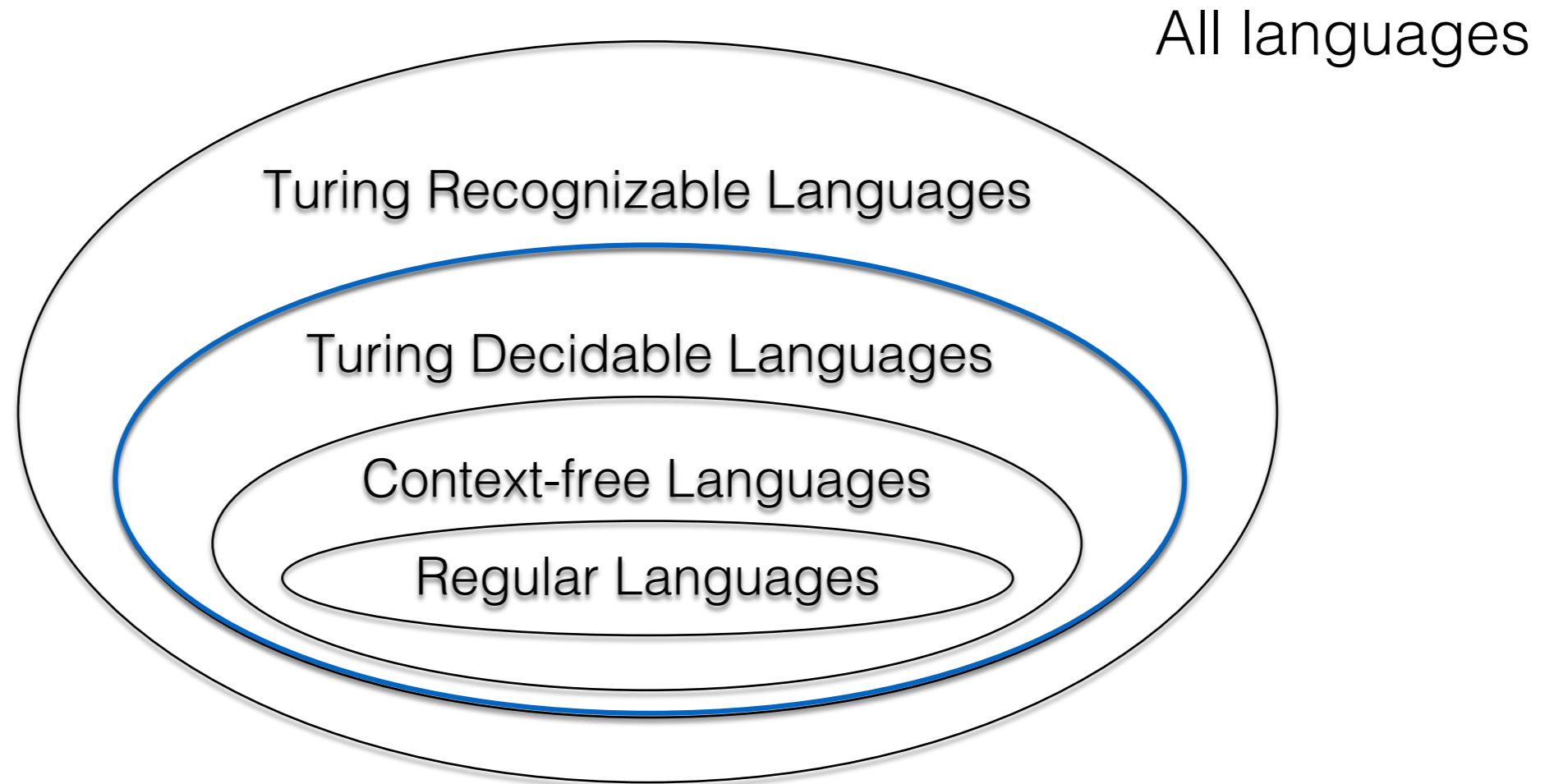
Why Turing Machines?

- Why are Turing Machines still relevant?
 - Church-Turing Thesis: Every physically realizable computation device can be simulated by a Turing Machine. (This has not been proven, but it is a general belief.)
 - The strong Church-Turing thesis says a TM can simulate all other computational models efficiently (with polynomial overhead)
 - Essentially, the set of “computable” problems is fully captured by Turing Machines

Decidability

- Decidable problems: Problems where we can construct an algorithm to solve it in finite time. A Turing Machine will halt on every input with an accept or reject (Turing Decidable)
- Undecidable problems: Problems where we cannot construct an algorithm that can solve it in finite time. A Turing Machine cannot solve these problems.
- Semi-decidable problems: Problems where the Turing Machine accepts and halts, but also rejects or loops forever (Turing Recognizable)

Decidability



Decidable Languages 4.1

- Theorem 4.4: Prove E_{DFA} is a decidable language.
 $E_{\text{DFA}} = \{ \langle A \rangle \mid A \text{ is a DFA and } L(A) = \emptyset \}$
- $E_{\text{DFA}} \Rightarrow$ emptiness testing for the language of a finite automaton.
- We determine whether or not a finite automaton accepts any string at all.
- Proof:
 - Remember how DFA's work. What is considered acceptance? Having a path from the start node to an accept node.

Decidable Languages 4.1

- Theorem 4.4 Proof: Let M be a TM to decide E_{DFA}
 - “On input $\langle A \rangle$:
 1. Mark the start state of A
 2. Repeat the following until no new states get marked:
 - a. Mark any state that has a transition coming into it from any state that is already marked
 3. If no accept state is marked, accept; otherwise reject”
 - This algorithm / TM will always halt and has a clear accept / reject values, therefore Turing Decidable

Breadth-
first search



Decidable Languages 4.1

- E_{DFA} is a Decidable Language
- Why is this?
 - Finite automatas are finite and countable. We can easily count the states and know we will reach a conclusion

Undecidability

- Fact: Almost all problems must be undecidable by any system that involves programming.
- A problem is the membership of a string in a language
- The number of different languages over any alphabet of more than one symbol is not countable
 - There is no way to assign integers to the languages such that every language has an integer, and every integer is assigned to one language
- On the other hand, programs, being finite strings over a finite alphabet are countable, thus there are infinitely fewer programs as there are problems

Countable Sets

- Let's look at a problem: How do we prove two sets are the same size?
- Finite Sets: Map every element from one set to exactly one other element on the other set, one-to-one and onto function
 - One-to-one: never maps two different elements in one set to the same element in the other set, $f(a) \neq f(b)$, when $a \neq b$
 - Onto: for every $b \in B$ there is an $a \in A$ such that $f(a) = b$
- If you have both, then have one-to-one correspondence and have a bijection

Countable Sets, cont.

- Let's look at a problem: How do we prove two sets are the same size?
- Infinite Sets:
 - Ex 4.13: Let $\mathbb{N} = \{1, 2, 3, \dots\}$ be the set of natural numbers. Let $N_2 = \{2, 4, 6, \dots\}$ (even elements of \mathbb{N})
 - Claim: $|\mathbb{N}| = |N_2|$ (the size of both is the same)
 - Proof Idea: Consider the bijection of $f: \mathbb{N} \rightarrow N_2$. Every element from \mathbb{N} will have a unique pair with N_2 and it will be one-to-one and on-to, therefore, these two sets are the same size.

Countable Sets

- Let's look at a problem: How do we prove two sets are the same size?
 - Infinite Sets:
 - It does not seem right that the two sets we saw were the same size. You would think that N_2 would be smaller than N . This brings us to countability.
 - Countable: Set B is countable if one of the following hold:
 1. $|B|$ is finite (finite size) or
 2. $|B| = |\mathbb{N}|$ (infinite size, but same size as the set of natural numbers)
 - If these do not hold then B is considered uncountable.

Countable Sets

- Ex.1.5: Let \mathbb{Q}^+ be the set of non-negative rational numbers.
 - Is $|\mathbb{Q}^+| = |\mathbb{N}|$?
- Proof Idea: We can list out the rational numbers as fractions by changing the numerator or denominator

1/1	1/2	1/3	1/4	1/5	...
2/1	2/2	2/3	2/4	2/5	...
3/1	3/2	3/3	3/4	3/5	...
4/1	4/2	4/3	4/4	4/5	...
5/1	5/2	5/3	5/4	5/5	...
...

We can now map
the bijection

Countable Sets

- Ex.1.5: Let \mathbb{Q}^+ be the set of non-negative rational numbers.

- Is $|\mathbb{Q}^+| = |\mathbb{N}|$?

We can now map the bijection.
Do a breadth-first search diagonally through the set and map the elements to \mathbb{N}

1 1/1	3 1/2	5 1/3	9 1/4	11 1/5	...
2 2/1	2/2	8 2/3	2/4	2/5	...
4 3/1	7 3/2	3/3	3/4	3/5	...
6 4/1	4/2	4/3	4/4	4/5	...
10 5/1	5/2	5/3	5/4	5/5	...
...

Skip duplicates: 2/2, 4/2, 3/3, 2/4, ...

This ensures one-to-one, so:

$$f(1) = 1/1, f(2) = 2/1,$$

$$f(3) = 1/2, \dots$$

Yes, \mathbb{Q}^+ is countable.

Uncountable Sets

- Are all infinite sets countable? What about real numbers \mathbb{R} ?
- Theorem 4.17: \mathbb{R} is uncountable.
 - Let \mathbb{R} = set of real numbers
 - Claim: $|\mathbb{R}| \neq |\mathbb{N}|$, \mathbb{R} is uncountable.

Uncountable Sets

- Proof by Diagonalization and Contradiction using Cantor's Diagonalization Argument:
 - Assume that \mathbb{R} is countable, therefore we can list every element of \mathbb{R} without missing any elements. Call this list L.
 - Suppose L looks like: (arbitrarily chosen numbers)

0.00110854...	Let's construct an element x which is in \mathbb{R} , but is not in L, thus proving the contradiction. This is easy to do since real numbers go on to infinity in both directions (significant digits and thus in actual numbers)
0.11987654...	
0.22889615...	
0.33985724...	

Uncountable Sets

- Proof by Diagonalization and Contradiction:
 - Suppose L looks like: (arbitrarily chosen numbers)
0.00110854...
0.11987654...
0.2289615...
0.33985724...
Let's construct an element x which is in \mathbb{R} , but is not in L . We do this by going diagonally through L 's list, see the numbers underlined.
 - We can pick a number that is not the same as what is in L or at the diagonal of L (underlined above) such as $x = 0.25621$. The elements of this number are not the same as the underlined elements in L , which were 0.0188...
 - We can keep redefining x to be unequal to L 's diagonal.

Uncountable Sets

- Proof by Diagonalization and Contradiction:
 - Suppose L looks like: (arbitrarily chosen numbers)
0.00110854...
0.11987654...
0.22889615...
0.33985724...

• The number we picked, $x = 0.25621$ is not the same as the the diagonal, 0.0188...

• So the i th digit of x is created to be different from every i th digit of L_i , therefore $x \neq L_i$ for all i . This means L is not a complete list and not countable.

Uncountable Sets – Undecidable Languages

- How do uncountable sets relate to Turing Machines?
 - Each Turing Machine recognizes one language.
 - Each Turing Machine is represented by a finite string, so the number of Turing Machines is countable.
 - The set $\{0, 1\}^*$ is uncountable, and, since all languages as we defined them are included in this set, there are many more languages than can be recognized by a Turing Machine.
 - Most languages are undecidable languages.

Undecidable Languages

- Barber's Paradox: The barber is the “one who shaves all those, and those only, who do not shave themselves.”
So, who shaves the barber?
- Claim: This is a contradiction. Here are the possibilities:
 - The barber cannot shave himself as he only shaves those who do not shave themselves. If he shaves himself, he is no longer the barber.
 - If he does not shave himself, he is in the group that would be shaved by the barber so he must shave himself.
- These scenarios are impossible; thus, we have a contradiction.

Undecidable Languages

- Barber's Paradox: The barber is the “one who shaves all those, and those only, who do not shave themselves.” So, who shaves the barber?
- We will use the same trick to prove Turing Machines are undecidable. We will feed a Turing Machine to itself to prove the contradiction.

Undecidable Languages

- Theorem 4.11: Look at the problem of determining whether a Turing Machine accepts a given input string called $A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \}$
- Claim: A_{TM} is undecidable
- Proof by Contradiction: Assume there exists a TM H such that $H(\langle M, w \rangle) = \{ \text{accepts if } M \text{ accepts } w, \text{ rejects if } M \text{ does not accept } w \}$
- Now construct a new TM D that uses H as a subroutine. D calls H to determine what M does when the input to M is its own description $\langle M \rangle$. Once D determines what M does, it will do the opposite.

Undecidable Languages

- Theorem 4.11: $A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \}$ Claim: A_{TM} is undecidable
- Proof by Contradiction: $H(\langle M, w \rangle) = \{ \text{accepts if } M \text{ accepts } w, \text{ rejects if } M \text{ does not accept } w \}$
- $D =$ “On input $\langle M \rangle$, where M is a TM
 1. Run H on $\langle M, \langle M \rangle \rangle$
 2. Output the opposite of what H outputs
 - If H accepts, reject
 - If H rejects, accept

Input to M is
its own
description as
a string $\langle M \rangle$

Undecidable Languages

- $D =$ “On input $\langle M \rangle$, where M is a TM
 1. Run H on $\langle M, \langle M \rangle \rangle$
 2. Output the opposite of what H outputs
 - If H accepts, reject
 - If H rejects, accept
- $D(\langle M \rangle) = \{\text{accepts if } M \text{ does not accept } \langle M \rangle, \text{ rejects if } M \text{ accepts } \langle M \rangle\}$
- $D(\langle D \rangle) = \{\text{accepts if } D \text{ does not accept } \langle D \rangle, \text{ rejects if } D \text{ accepts } \langle D \rangle\}$
- This is a contradiction, which means A_{TM} is undecidable. However, A_{TM} is recognizable.

Input to M is
its own
description as
a string $\langle M \rangle$

Undecidable Languages

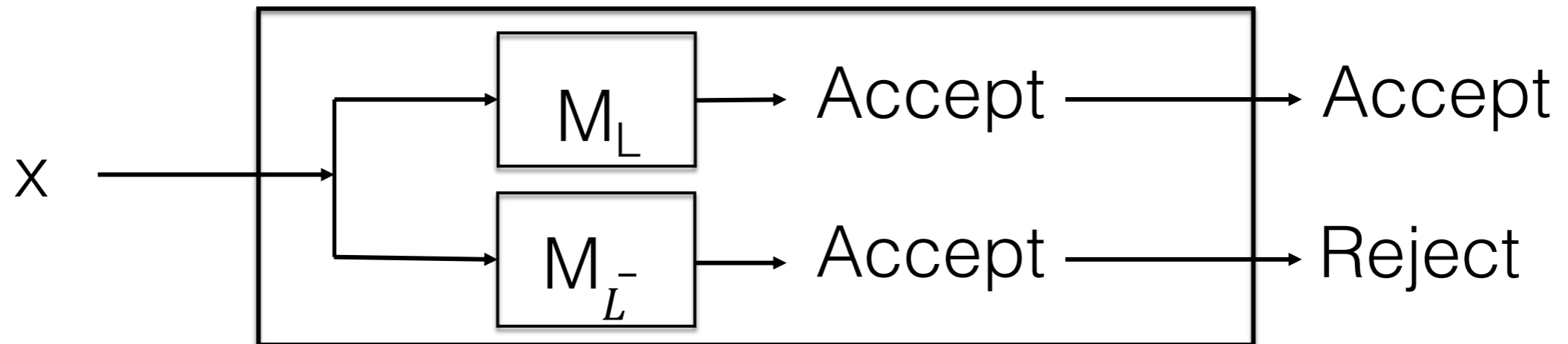
- Theorem 4.11: $A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \}$ Claim: A_{TM} is undecidable
- Proof by Contradiction: $H(\langle M, w \rangle) = \{ \text{accepts if } M \text{ accepts } w, \text{ rejects if } M \text{ does not accept } w \}$
- Here is why it is a contradiction:
 - H accepts $\langle M, w \rangle$ exactly when M accepts w
 - D rejects $\langle M \rangle$ exactly when M accepts $\langle M \rangle$
 - D rejects $\langle D \rangle$ exactly when D accepts $\langle D \rangle$ (This cannot happen. D cannot both reject $\langle M \rangle$ and $\langle D \rangle$.)

Undecidable Languages

- 4.22 Claim: A language L is decidable if and only if L and \bar{L} are Turing Recognizable
- If we know L is Turing Recognizable, and if \bar{L} is Turing Recognizable as well, then L must be decidable.
- This implies that for all x we can decide in finite time whether $x \in L$ or $x \in \bar{L}$. ($\Sigma^* = L \cup \bar{L}$)
 - Assume L and \bar{L} are Turing Recognizable
 - Given $x \in L$, there exists a TM, M_L that halts and accepts
 - Given $x \in \bar{L}$, there exists a TM, $M_{\bar{L}}$ that halts and accepts
 - Therefore, we take turns simulating one step of M_L and $M_{\bar{L}}$ in parallel. One machine must halt and accept since $\Sigma^* = L \cup \bar{L}$

Undecidable Languages

- 4.22 Claim cont.: A language L is decidable if and only if L and \bar{L} are Turing Recognizable



- We take turns simulating one step of M_L and $M_{\bar{L}}$ in parallel. One machine must halt and accept since $\Sigma^* = L \cup \bar{L}$
 - If M_L accepts, we accept ($x \in L$)
 - If $M_{\bar{L}}$ accepts, we reject ($x \notin L$)

Undecidable Languages

- Remember the language from above, $A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \}$. This language is undecidable, but Turing Recognizable.
- What does this tell us about $\overline{A_{TM}}$?
 - We know that A_{TM} is Turing Recognizable.
 - If $\overline{A_{TM}}$ was also Turing Recognizable, A_{TM} would be decidable.
 - Since we proved that A_{TM} is not decidable above, $\overline{A_{TM}}$ must not be Turing Recognizable. $\overline{A_{TM}}$ is undecidable.

Try It

1. Prove that the set L of all infinite sequences over $\{0, 1\}$ are uncountable using the proof by diagonalization.
2. Prove that \mathbb{N}^2 where $\mathbb{N}^2 = \{1, 4, 9, 16, 25, \dots\}$ is countable.

Try It

1. Prove that the set L of all infinite sequences over $\{0, 1\}^*$ are uncountable using the proof by diagonalization.
 - If we try to construct all the infinite sequences L such as:
$$\begin{array}{l} \underline{0}10101\dots \\ 1\underline{0}1010\dots \\ 11\underline{1}010\dots \\ \dots \end{array}$$
 - We can pick a number that is not the same as what is in $\{0, 1\}^*$ or at the diagonal of $\{0, 1\}^*$ (underlined above) such as $x = 0111000$. The elements of this number are not the same as the underlined elements in L , which were $001\dots$
 - This shows that $x \notin L$, and therefore all infinite sequences over $\{0, 1\}^*$ are uncountable.

Try It

2. Prove that \mathbb{N}^2 where $\mathbb{N}^2 = \{1, 4, 9, 16, 25, \dots\}$ is countable.
- We can list out the \mathbb{N}^2 numbers and match them to the set of \mathbb{N} , thus $|\mathbb{N}^2| = |\mathbb{N}|$

1 – 1

4 – 2

9 – 3

16 – 4

25 – 5

...