Theoretical Computer Science (Bridging Course)

Decidability

Gian Diego Tipaldi

Topics Covered

- Investigation on what is solvable
- Decidable languages
- The halting problem

Decidable Problems

- Acceptance problem:
  - Decide if a string is accepted
- Equivalence problem:
  - Decide if two automata are equivalent
- Emptiness test problem:
  - Decide if the accepted language is empty

- Applied to DFA,NFA,RE,PDA,CFG,TM,…

Acceptance problem for DFAs

- Decide is a DFA accept a string w
- Express it as a language A_{DFA}

$$A_{DFA} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts } w\}$$

- B is the encoding of a DFA
- Testing acceptance is equivalent to language membership

Acceptance problem for DFAs

- Use a decider to test membership
- Idea: Decider will simulate B and accept/reject if B accepts/rejects w
- Encoding of $\mathrm{B} = (Q, \Sigma, \delta, q_o, F)$ and w

q0	**q1**	**q2**	**q3**	**#**	**0**	**1**	**#**	**<table>**	**…**
#	**q0**	**#**	**q2**	**q3**	**#**	**#**	**0**	**1**	**1**
0	**1**	**1**	**1**	**0**	**#**	**q0**			

Acceptance problem for DFAs

- Use a decider to test membership
- Idea: Decider will simulate B and accept/reject if B accepts/rejects w
- Encoding of $\mathrm{B} = (\boxed{Q}, \Sigma, \delta, q_o, F)$ and w

q0	q1	q2	q3	#	0	1	#	<table>	…
#	q0	#	q2	q3	#	#	0	1	1
0	1	1	1	0	#	q0			

Acceptance problem for DFAs

- Use a decider to test membership
- Idea: Decider will simulate B and accept/reject if B accepts/rejects w

- Encoding of $\mathrm{B} = (Q,\boxed{\Sigma,}\delta,q_o,F)$ and w

q0	**q1**	**q2**	**q3**	**#**	**0**	**1**	**#**	**<table>**	**…**
#	**q0**	**#**	**q2**	**q3**	**#**	**#**	**0**	**1**	**1**
0	**1**	**1**	**1**	**0**	**#**	**q0**			

Acceptance problem for DFAs

- Use a decider to test membership
- Idea: Decider will simulate B and accept/reject if B accepts/rejects w

- Encoding of $\mathrm{B} = (Q, \Sigma, \boxed{\delta,} q_o, F)$ and w

q0	**q1**	**q2**	**q3**	**#**	**0**	**1**	**#**	**<table>**	**...**
#	**q0**	**#**	**q2**	**q3**	**#**	**#**	**0**	**1**	**1**
0	**1**	**1**	**1**	**0**	**#**	**q0**			

Acceptance problem for DFAs

- Use a decider to test membership
- Idea: Decider will simulate B and accept/reject if B accepts/rejects w

- Encoding of $B = (Q, \Sigma, \delta, \boxed{q_o}, F)$ and w

q0	**q1**	**q2**	**q3**	**#**	**0**	**1**	**#**	**<table>**	**…**
#	**q0**	**#**	**q2**	**q3**	**#**	**#**	**0**	**1**	**1**
0	**1**	**1**	**1**	**0**	**#**	**q0**			

Acceptance problem for DFAs

- Use a decider to test membership
- Idea: Decider will simulate B and accept/reject if B accepts/rejects w

- Encoding of $\mathrm{B} = (Q, \Sigma, \delta, q_o, \boxed{F})$ and w

q0	**q1**	**q2**	**q3**	**#**	**0**	**1**	**#**	**<table>**	**…**
#	**q0**	**#**	**q2**	**q3**	**#**	**#**	**0**	**1**	**1**
0	**1**	**1**	**1**	**0**	**#**	**q0**			

Acceptance problem for DFAs

- Use a decider to test membership
- Idea: Decider will simulate B and accept/reject if B accepts/rejects w
- Encoding of $\mathrm{B} = (Q, \Sigma, \delta, q_o, F)$ and $\boxed{\text{w}}$

q0	**q1**	**q2**	**q3**	**#**	**0**	**1**	**#**	**<table>**	**...**
#	**q0**	**#**	**q2**	**q3**	**#**	**#**	**0**	**1**	**1**
0	**1**	**1**	**1**	**0**	**#**	**q0**			

Acceptance problem for DFAs

- Use a decider to test membership
- Idea: Decider will simulate B and accept/reject if B accepts/rejects w

- Encoding of $\mathrm{B} = (Q, \Sigma, \delta, q_o, F)$ and w

q0	**q1**	**q2**	**q3**	**#**	**0**	**1**	**#**	**<table>**	**…**
#	**q0**	**#**	**q2**	**q3**	**#**	**#**	**0**	**1**	**1**
0	**1**	**1**	**1**	**0**	**#**	**q0**			

Acceptance problem for DFAs

<u>**Theorem 4.1**</u>:
A_{DFA} is a decidable language.

Proof:
M= "On input $\langle B, w \rangle$, where B is a DFA and w is a string:

1. Simulate B on input w.
2. If the simulation ends in an accept state, *accept.* If it ends in a nonaccepting state, *reject*."

Acceptance problem for NFAs

<u>Theorem 4.2</u>:

A_{NFA} is a decidable language.

Proof:

N = "On input $\langle B, w \rangle$, where B is a NFA and w is a string:

1. Convert the NFA in an equivalent DFA C
2. Run the previous machine on input $\langle C, w \rangle$
3. If the simulation ends in an accept state, *accept*. If it ends in a nonaccepting state, *reject*."

Acceptance problem for REs

<u>Theorem 4.3</u>:

A_{RE} is a decidable language.

Proof:

P = "On input $\langle B, w \rangle$, where B is a RE and w is a string:

1. Convert the RE in an equivalent NFA A
2. Run the machine N on input $\langle A, w \rangle$
3. If N accepts, *accept.* If N rejects, *reject*."

Emptiness problem for DFAs

- Decide is a DFA accept the empty language
- Express it as a language E_{DFA}

$$E_{DFA} = \{ \langle A \rangle \mid A \text{ is a DFA for which } L(A) = \emptyset \}$$

- A is the encoding of a DFA

Emptiness problem for DFAs

<u>Theorem 4.4</u>:
E_{DFA} is a decidable language.

Proof idea:
A DFA accepts some string iff it is possible to reach an accept state using valid transitions. Construct a TM T similar to the one for connected graphs

Emptiness problem for DFAs

<u>**Theorem 4.4**</u>:

E_{DFA} is a decidable language.

Proof:

T = "On input $\langle A \rangle$, where A is a DFA:

1. Mark the start state of A
2. Repeat until no new state is marked
   1. Mark a state if it has a transition to it coming from any other marked state
3. If no accept state is marked, *accept.* Otherwise, *reject*."

Equivalence Problem for DFAs

<u>Theorem 4.5</u>:

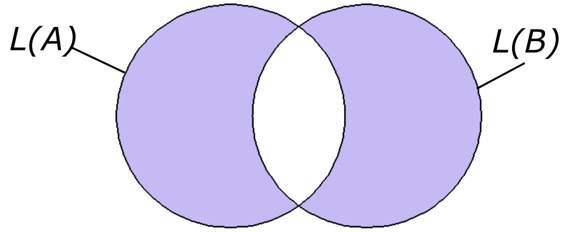
$EQ_{DFA} = \{ \langle A, B \rangle \mid A, B \in \text{DFA}, \ L(A) = L(B) \}$

is a decidable language.

Proof idea:

Prove the symmetric difference is empty.

$$L(C) = \left(\overline{L(B)} \cap L(A) \right) \cup \left(\overline{L(A)} \cap L(B) \right)$$



The symmetric difference of $L(A)$ and $L(B)$

Equivalence Problem for DFAs

Theorem 4.5:
EQ_{DFA} is a decidable language.

Proof:
F = "On input $\langle A, B \rangle$, where A, B are DFA:

1. Construct the DFA C for the symmetric difference language (closure property)
2. Run TM T from Theorem 4.4
3. If T accepts, *accept.* If T rejects, *reject*."

Acceptance Problem for CFGs

<u>Theorem 4.7</u>:

$A_{CFG} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates } w \}$ is a decidable language.

Proof idea:

Rely on the fact that if G is in CNF, then any derivation of w has length at most 2|w| + 1.

Also consider that there are only finitely many derivations of length less than n.

Acceptance Problem for CFGs

<u>Theorem 4.7</u>:

A_{CFG} is a decidable language.

Proof:

S = "On input $\langle G, w \rangle$, where B is a CFG and w is a string:

1. Convert G to a CNF
2. List all derivations with k=2n-1 steps, n is the length of w. if n=0 consider k=1.
3. If any of them generates w, *accept.* If not, *reject*."

Emptiness Problem for CFGs

Theorem 4.7:

$E_{CFG} = \{ \langle G \rangle \mid G \text{ is a CFG for which } L(G) = \emptyset \}$

is a decidable language.

Proof idea:

Determine for each variable whether that variable is capable to generate a string of terminals

Emptiness Problem for CFGs

Theorem 4.7:

E_{CFG} is a decidable language.

Proof:

R = "On input $\langle G \rangle$, where G is a CFG:

1. Mark all terminal symbols in G
2. Repeat until no new variable is marked
    1. Mark any variable A if G contains $A \rightarrow U_1 \dots U_k$ and $U_1, \dots, U_k$ have all been marked
3. If the start symbol is marked, *accept.* If not, *reject*."

Equivalence Problem for CFGs

$$EQ_{CFG} = \{\langle G, H \rangle | G, H \in \ CFGs, L(G) = L(H)\}$$

- Is it decidable?

Equivalence Problem for CFGs

$$EQ_{CFG} = \{\langle G, H\rangle | G, H \in \ CFGs, L(G) = L(H)\}$$

- ~~Is it decidable?~~ **NO!**

- We cannot use the same method as for DFAs, since context free languages are not closed under complementation nor intersection.

Decidability of CFLs

Theorem 4.9:

Every context free language is decidable.

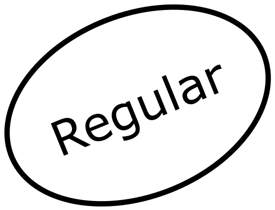
Proof:

Let G be a CFG for the language

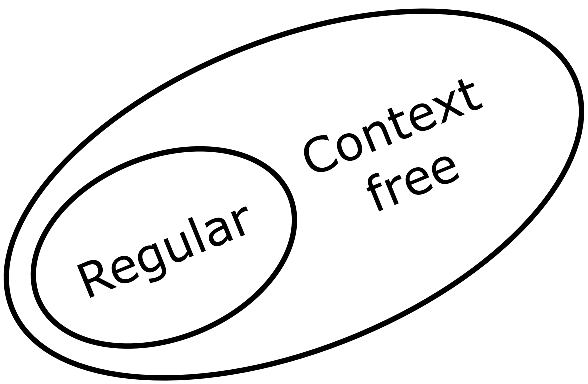
M_G = "On input w:

1. Run the TM S on input $\langle G, w \rangle$
2. If S accepts, *accept;* If S rejects, *reject*."

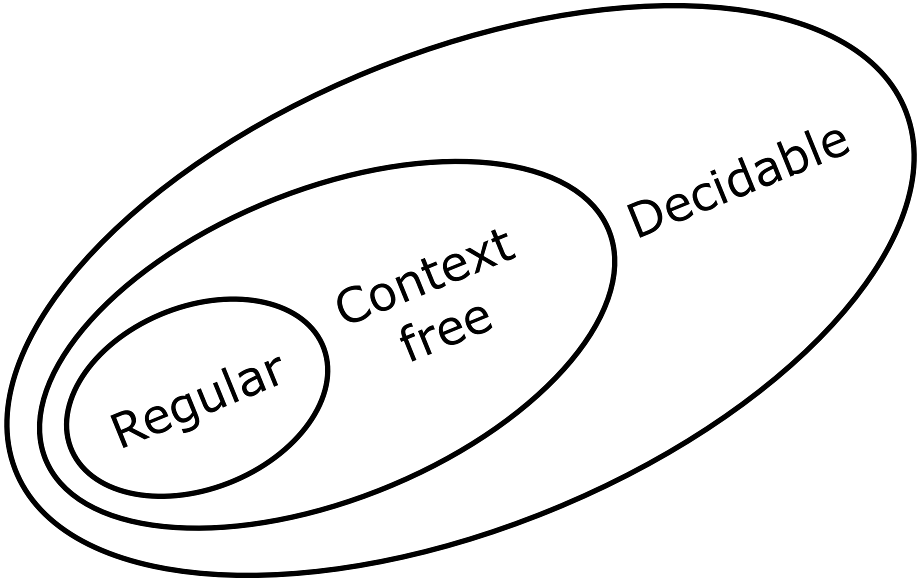
Classes of Languages



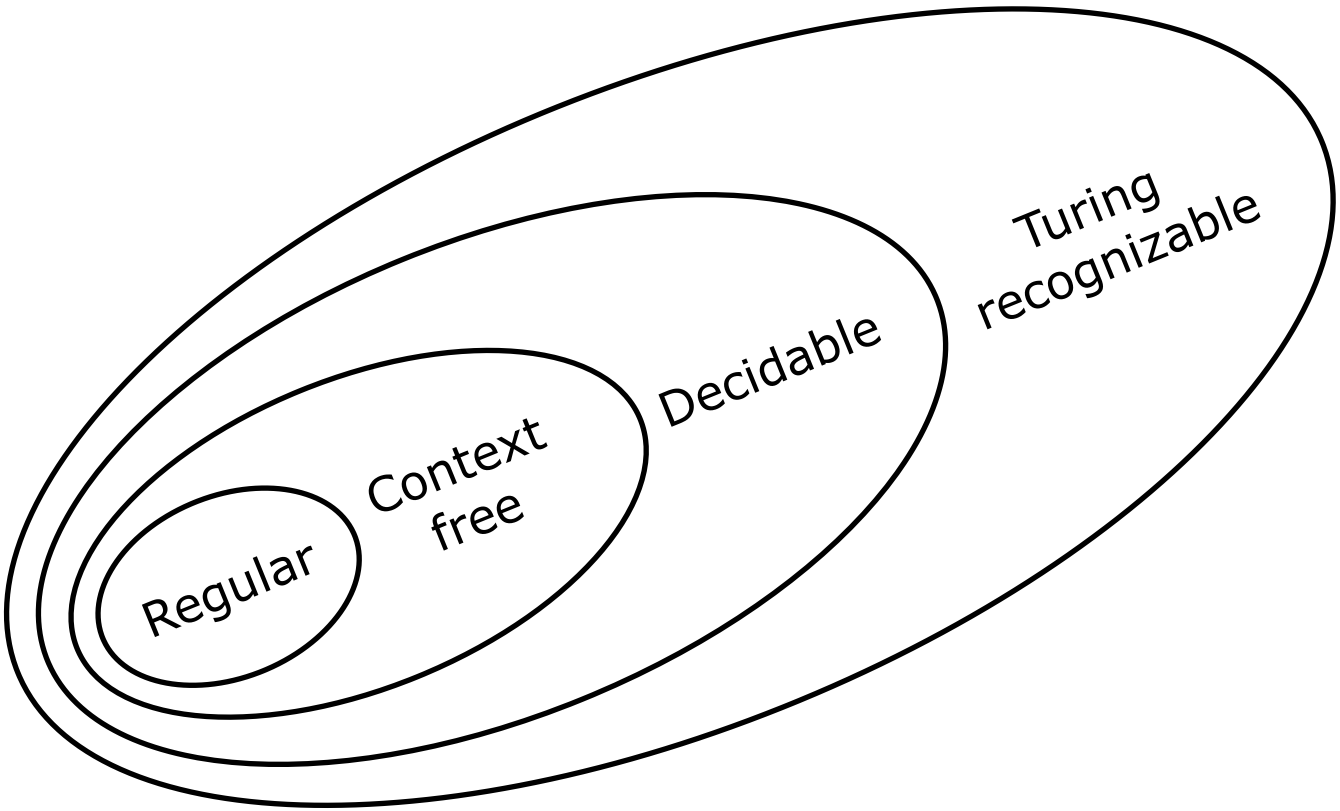
Classes of Languages



Classes of Languages



Classes of Languages



The Halting Problem

- Some problems are unsolvable

- Famous example: the halting problem

Philosophical implication:
Computers are fundamentally limited

The Halting Problem

$$A_{TM} = \{\, \langle M, w \rangle \mid M \text{ is a TM that accepts } w \,\}$$

Theorem 4.11:

A_{TM} is undecidable

Observation: A_{TM} is Turing recognizable, thus recognizer are more powerful than deciders.

The Halting Problem

$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts } w \}$$

The following machine recognizes it

U = "On input $\langle M, w \rangle$, where M is a TM and w is a string

1. Simulate M on input w
2. If M ever enters the accept state, *accept;* if M ever enters the reject state, *reject.*"

Diagonalization

- Developed by G. Cantor in 1873
- Concerns the measure of infinite sets
- Used to prove the halting problem

- Do sets A and B have the same size?
- If finite, one can count the elements
- How about infinite sets?

Diagonalization

- Consider possible pairings from A to B

- Consider the function $f: A \rightarrow B$
  - f is injective: $f(a) \neq f(b)$, whenever $a \neq b$
  - f is surjective: $\forall\, b \in B, \exists a \in A: f(a) = b$
  - f is bijective if injective and surjective

- A and B have same size if $\exists\, f$ bijective
- A set is countable if size at most of $\mathbb{N}$

Example – Even Numbers

- Consider this two sets

$$A = \{ a \mid a \in \mathbb{N} \}, B = \{ b \mid b/2 \in \mathbb{N} \}$$

- Consider the function $f\colon A \rightarrow B$

$$f(n) = 2n$$

- f is a bijective function, therefore A and B have the same size and B is countable

Example – Rational Numbers

- Consider the set of rational numbers
$$\mathbb{Q} = \left\{ \frac{m}{n} \mid m, n \in \mathbb{N} \right\}$$

- $\mathbb{Q}$ seems much larger than $\mathbb{N}$

Example – Rational Numbers

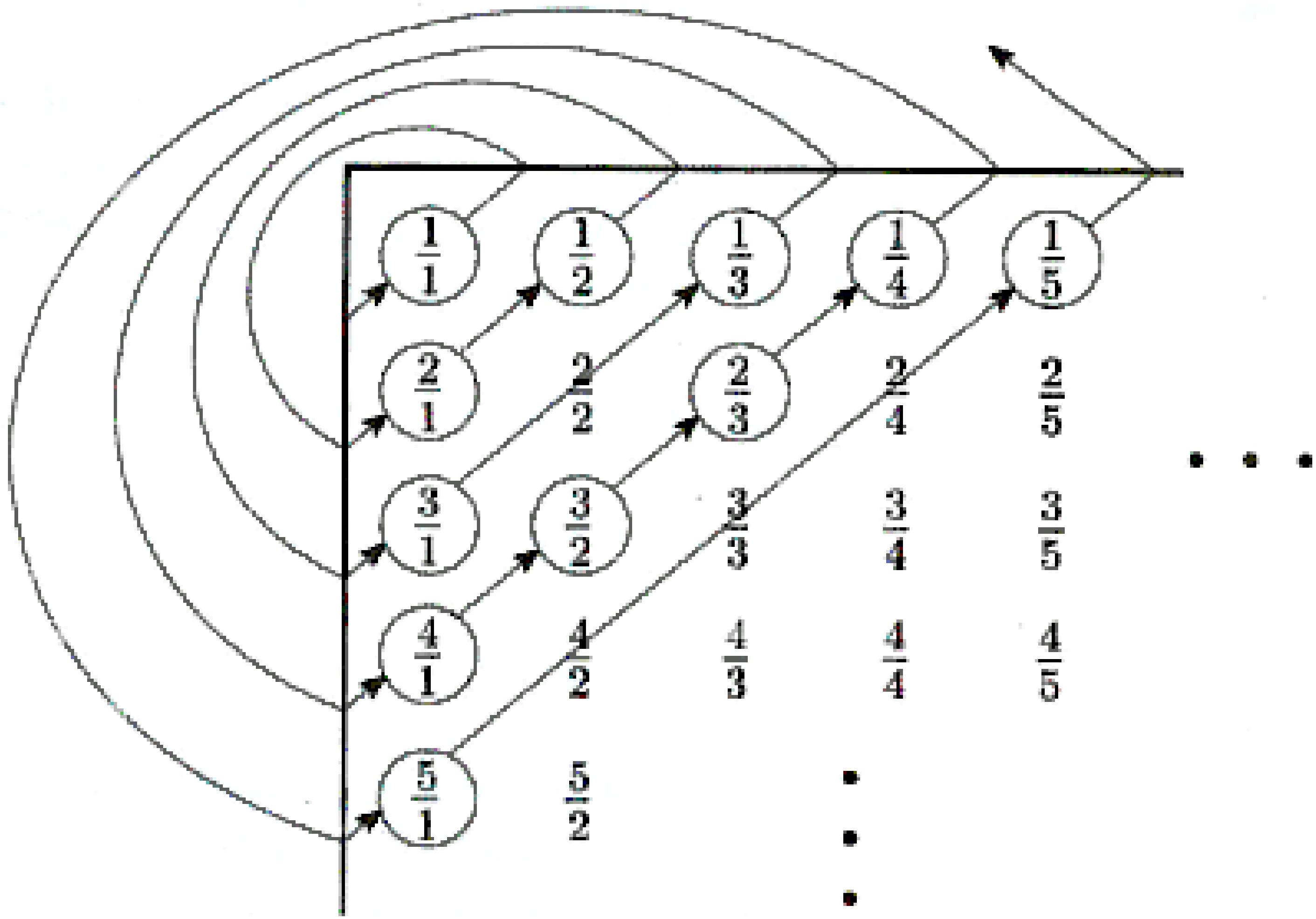
- Consider the set of rational numbers

$$\mathbb{Q} = \left\{ \frac{m}{n} \mid m, n \in \mathbb{N} \right\}$$

- $\mathbb{Q}$ seems much larger than $\mathbb{N}$, but…

- $\mathbb{Q}$ is countable

Example – Rational Numbers



Example – Real Numbers

- Consider the set of real numbers $\mathbb{R}$
- $\pi = 3.1415926 \ldots$
- $\sqrt{2} = 1.4142135 \ldots$
- Is it countable?

Example – Real Numbers

<u>Theorem 4.17:</u>

$\mathbb{R}$ is uncountable

Proof (by contradiction):

Assume that there is an f from $\mathbb{N}$ to $\mathbb{R}$. Find an x in $\mathbb{R}$ that $x \neq f(n)\ \forall\, n \in \mathbb{N}$. x is a number between 0 and 1. The first digit is different than the first decimal of f(1), the second is different than the second decimal of f(2) and so on…

Example – Real Numbers

<u>Theorem 4.17:</u>

$\mathbb{R}$ is uncountable

Proof (by contradiction):

n	$f(n)$
1	$3.\underline{1}414\ldots$
2	$5.5\underline{6}7\ldots$
3	$0.88\underline{8}888\ldots$
…	…

Example – Real Numbers

<u>Theorem 4.17:</u>

$\mathbb{R}$ is uncountable

Proof (by contradiction):

n	$f(n)$
1	$3.\underline{1}414\ldots$
2	$5.5\underline{6}7\ldots$
3	$0.88\underline{8}888\ldots$
…	…

x = 0.275…

Example – Real Numbers

<u>Theorem 4.17:</u>

$\mathbb{R}$ is uncountable

Proof (by contradiction):

n	$f(n)$
1	$3.\underline{1}414\ldots$
2	$5.5\underline{6}7\ldots$
3	$0.88\underline{8}888\ldots$
…	…

$x = 0.275\ldots$

So, $x \neq f(n)$ for all n

What About Turing Machines?

- The set of all strings Σ^* is countable
  - Similar to rational numbers
  - Go from small strings to bigger
- We can encode a TM as a string
- The set of TM is countable

Uncountable Languages

Lemma:
The set B of infinite binary strings is uncountable

Proof:
Proceed as with the real numbers. Find an infinite string whose first symbol is different than the first symbol on the first string and so on…

Uncountable Languages

Lemma:
The set L of all languages is uncountable

Proof:
Define a correspondence between L and B. Take the set of all strings Σ^* . Encode a language A with a binary string χ_A, where 1 means a string belongs to A.

$$
\begin{array}{lcccccccccccc}
\Sigma^* = \{ & \varepsilon, & 0, & 1, & 00, & 01, & 10, & 11, & 000, & 001, & \ldots & \} & ; \\
A = \{ & & 0, & & 00, & 01, & & & 000, & 001, & \ldots & \} & ; \\
\chi_A = & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & \ldots & . &
\end{array}
$$

Turing Recognizable Languages

Theorem 4.18:

Some languages are not Turing recognizable

Proof:

The set of TMs is countable, while the set of all language is not. Therefore, there is no correspondence between the set of languages and the set of TMs.

The Halting Problem

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM that accepts } w\}$$

<u>Theorem 4.11:</u>

A_{TM} is undecidable

Proof (by contradiction):
Assume A_{TM} is decidable and H is a decider for that. Build another decider D that contradicts the hypothesis.
Hint: Use H to define D.

The Halting Problem

Proof (by contradiction):

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

1. Run H on $\langle M, \langle M \rangle \rangle$
2. Accept if H rejects and vice versa"

We have

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

The Halting Problem

Proof (by contradiction):

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

1. Run H on $\langle M, \langle M \rangle \rangle$
2. Accept if H rejects and vice versa"

We have (on its own encoding)

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

The Halting Problem

Proof (by contradiction):

D = “On input $\langle M \rangle$, where M is a TM:

1. Run H on $\langle M, \langle M \rangle \rangle$
2. Accept if H rejects and vice versa”

We have (on its own encoding)

$$D(\langle D \rangle) = \begin{cases} accept \text{ if } D \ldots \text{accept } \langle D \rangle \\ \ldots D \text{ accepts } \langle D \rangle \end{cases}$$

Contradiction

The Halting Problem

Where is the diagonalization?

The Halting Problem

Where is the diagonalization?

	<M1>	<M2>	<M3>	<M4>	...	<D>	...
M1	*accept*		*accept*			*accept*	
M2	*accept*	*accept*	*accept*	*accept*		*accept*	
M3					...		...
M4	*accept*		*accept*			*accept*	
⋮		⋮			⋱		

The Halting Problem

Where is the diagonalization?

	<M1>	<M2>	<M3>	<M4>	...	<D>	...
M1	*accept*	*reject*	*accept*	*reject*		*accept*	
M2	*accept*	*accept*	*accept*	*accept*		*accept*	
M3	*reject*	*reject*	*reject*	*reject*	...	*reject*	...
M4	*accept*	*reject*	*accept*	*reject*		*accept*	
⋮		⋮			⋱		

The Halting Problem

Where is the diagonalization?

	<M1>	<M2>	<M3>	<M4>	...	<D>	...
M1	*accept*	*reject*	*accept*	*reject*		*accept*	
M2	*accept*	*accept*	*accept*	*accept*		*accept*	
M3	*reject*	*reject*	*reject*	*reject*	...	*reject*	...
M4	*accept*	*reject*	*accept*	*reject*		*accept*	
⋮		⋮			⋱		

The Halting Problem

Where is the diagonalization?

	<M1>	<M2>	<M3>	<M4>	...	<D>	...
M1	*accept*	*reject*	*accept*	*reject*		*accept*	
M2	*accept*	*accept*	*accept*	*accept*		*accept*	
M3	*reject*	*reject*	*reject*	*reject*	...	*reject*	...
M4	*accept*	*reject*	*accept*	*reject*		*accept*	
⋮		⋮			⋱		
D	*reject*	*reject*	*accept*	*accept*		?	
⋮		⋮					⋱

Non Recognizable Languages

- Are there languages that are not even Turing recognizable?

Non Recognizable Languages

- Are there languages that are not even Turing recognizable?

- Yes!

- We need something harder than A_{TM}

Co-Turing Recognizable

A language is co-Turing recognizable if it is the complement of a Turing recognizable language

Theorem 4.22:

A language is decidable iff it is Turing recognizable and co-Turing recognizable

Co-Turing Recognizable

Theorem 4.22:

A language is decidable iff it is Turing recognizable and co-Turing recognizable

Proof (forward):
If A is decidable, then the complement of A is decidable. A decidable language is also Turing recognizable.

Co-Turing Recognizable

<u>Theorem 4.22</u>:

A language is decidable iff it is Turing recognizable and co-Turing recognizable

Proof (backward):

M1,M2 are the recognizer of A, co-A.

M = "On input w:

1. Run M1 and M2 in parallel.
2. If M1 accepts, *accept*. If M2 accepts, *reject*"

Non Recognizable Languages

Corollary:

$\overline{A_{TM}}$ is not Turing recognizable

Proof:

We know that A_{TM} is Turing recognizable.

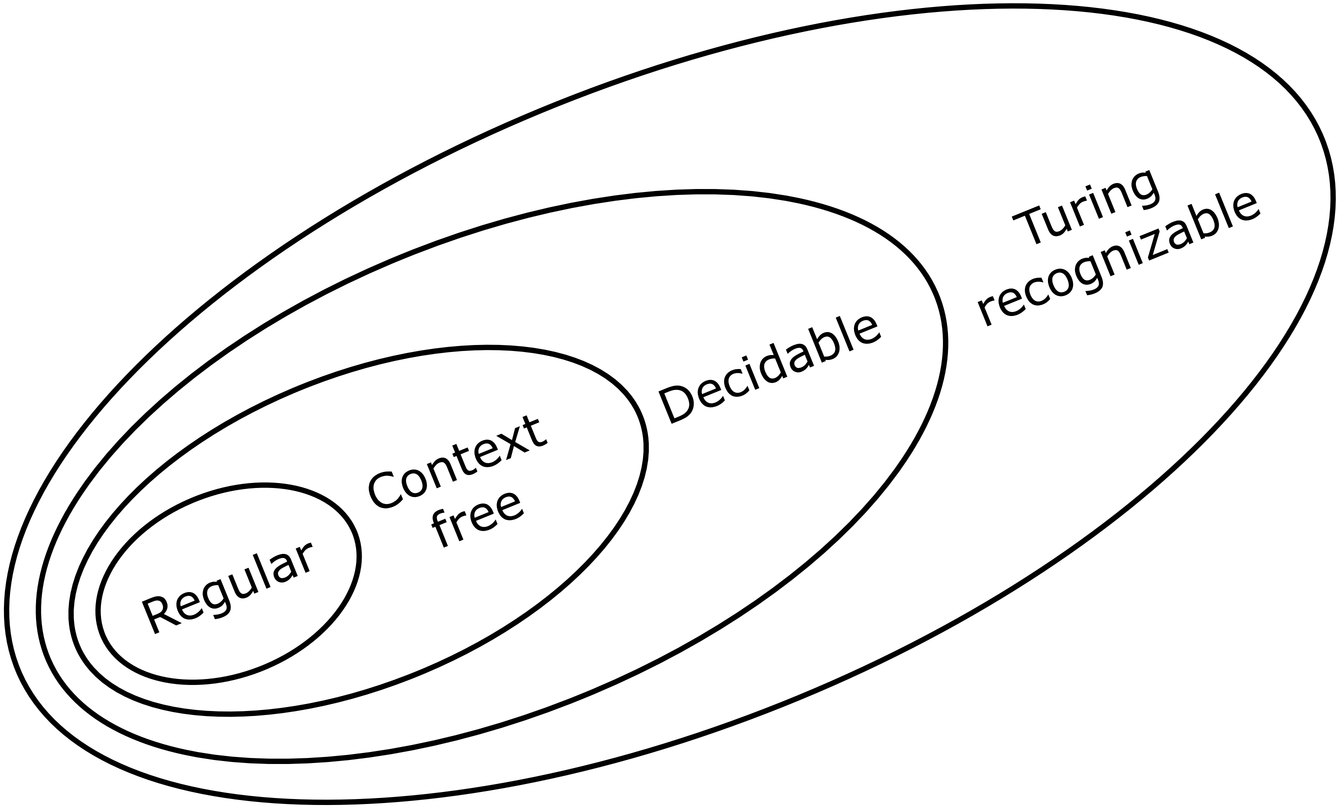
Assume $\overline{A_{TM}}$ is also Turing recognizable.

Then A_{TM} would be decidable.

Contradiction: Theorem 4.11 tells us not!

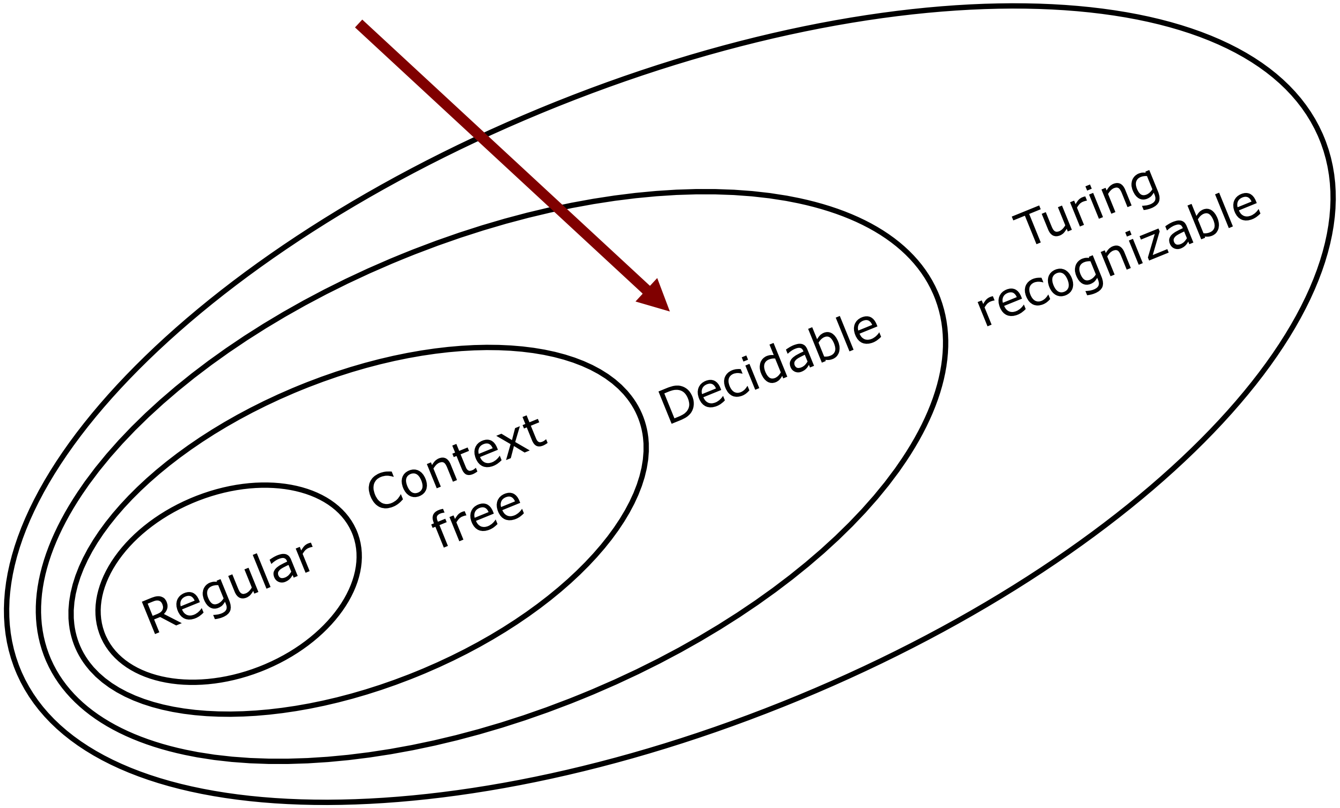
Example Exam Question

Classes of Languages



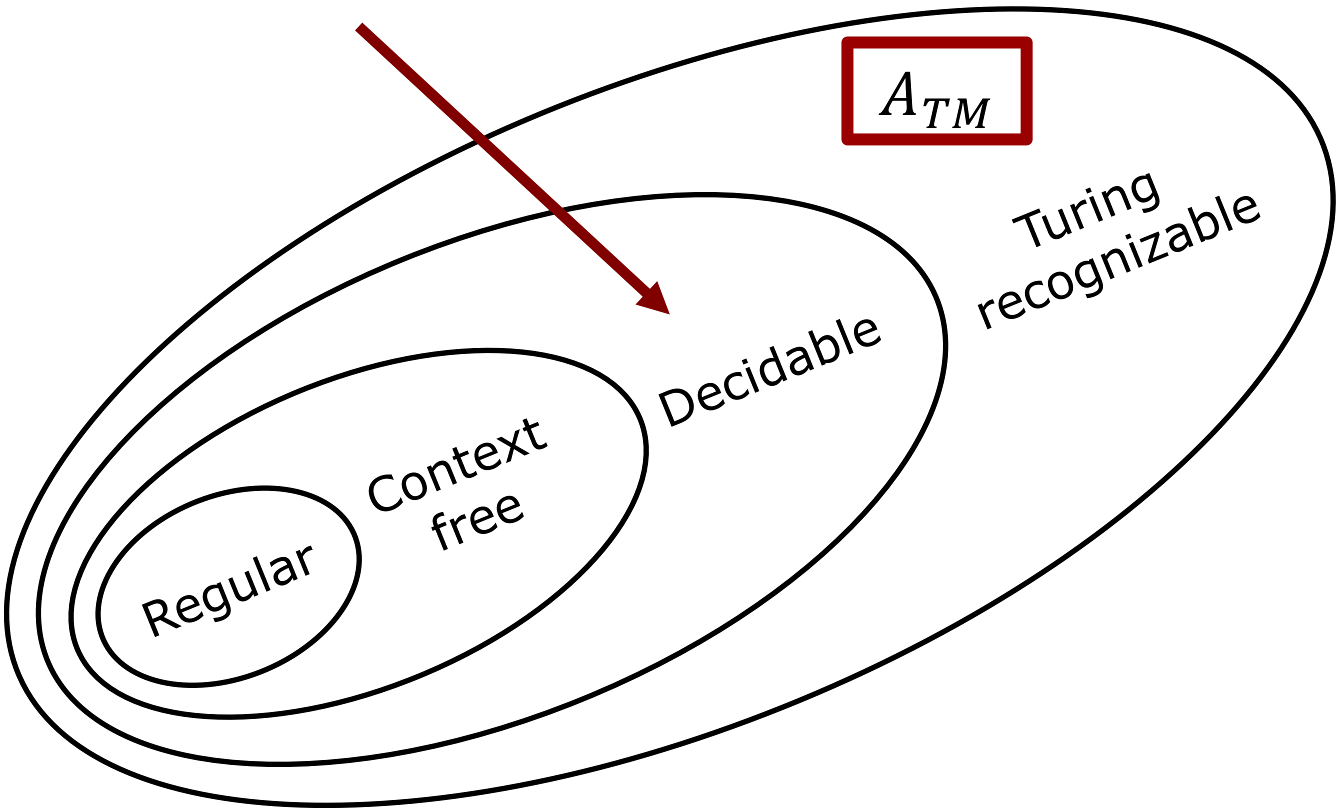
Classes of Languages

$$\boxed{\begin{array}{c} A_{DFA}, A_{NFA}, A_{RE}, \\ A_{CFG}, E_{DFA}, E_{CFG}, EQ_{DFA} \end{array}}$$



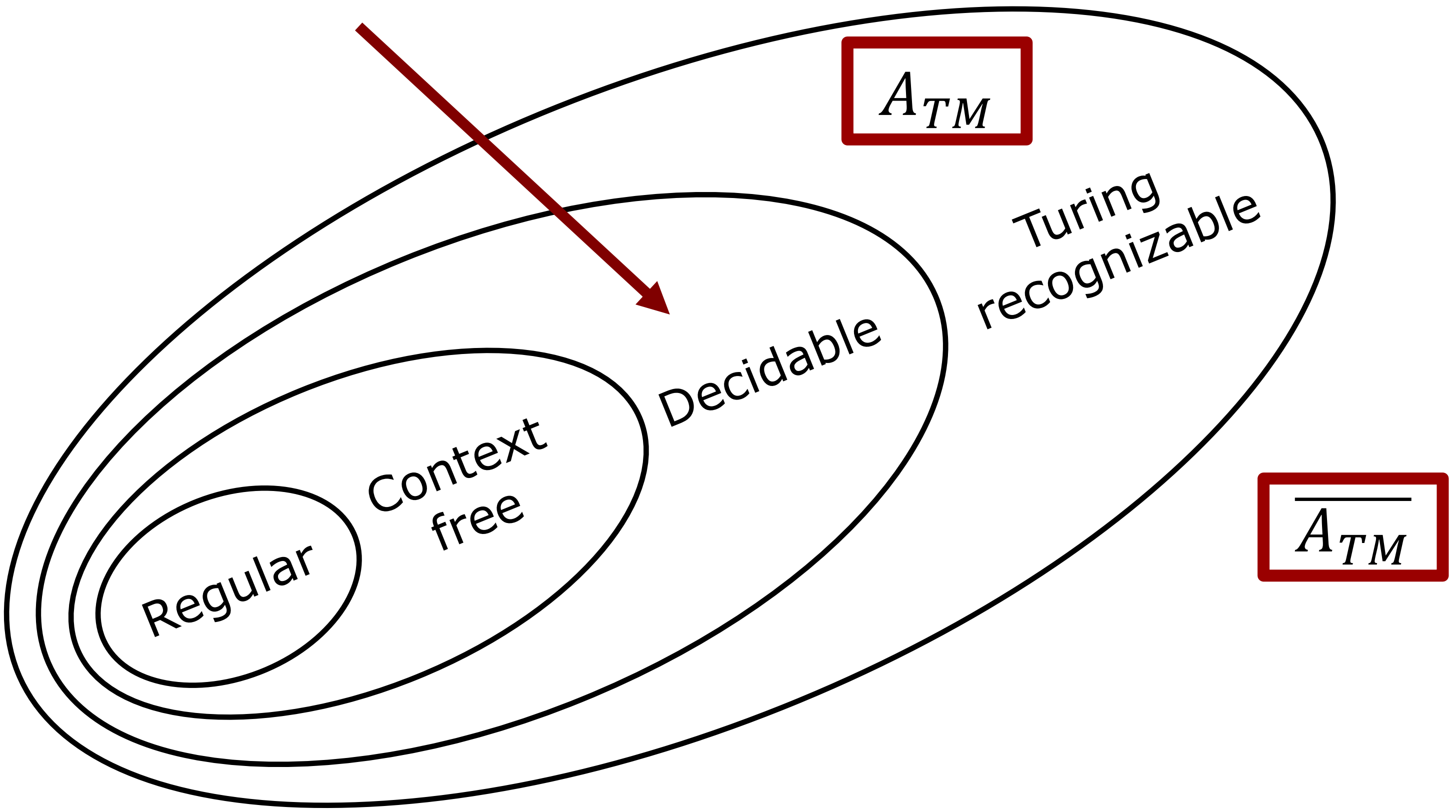
Classes of Languages

$A_{DFA}, A_{NFA}, A_{RE},$
$A_{CFG}, E_{DFA}, E_{CFG}, EQ_{DFA}$



Classes of Languages

$A_{DFA}, A_{NFA}, A_{RE},$
$A_{CFG}, E_{DFA}, E_{CFG}, EQ_{DFA}$



Summary

- Decidable problems
  - Acceptance
  - Emptiness test
  - Equivalence
- The halting problem
- Digaonalization