Theory of Computation
Decidability

Dimitris Diochnos
School of Computer Science
University of Oklahoma

Outline

Table of Contents

Introduction

Objective

Explore the limits of algorithmic solvability

Why study solvability?

- Useful to know when a problem is unsolvable
  $\Rightarrow$ Need to modify the problem if we need some algorithmic solution
- Cultural
  $\Rightarrow$ Gain better perspective on computation, what computers can and cannot do

Table of Contents

Decidable Problems Concerning Regular Languages

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

The language A_{DFA} expresses the **acceptance problem** for DFAs.
This language contains all the encodings of all DFAs together with strings that the DFAs accept.

Decidable Problems Concerning Regular Languages

Theorem 1

The language

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

is a decidable language.

Proof Idea.

We need a TM M that decides A_{DFA}.
$M =$ "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 non-accepting state, **reject**."

□

Decidable Problems Concerning Regular Languages

Theorem 2

The language

$$A_{NFA} = \{\langle B, w\rangle \mid B \text{ is an NFA that accepts input string } w\}$$

is a decidable language.

Proof Idea.

1. Use the powerset construction and transform the given (assuming valid) NFA N to a DFA C.
2. Then use the TM M from the previous theorem (Theorem 1) and decide the input $\langle C, w\rangle$.
3. Return the answer that we obtain from M. □

Decidable Problems Concerning Regular Languages

Theorem 3

The language

$$A_{REX} = \{\langle R, w\rangle \mid R \text{ is a regular expression that generates string } w\}$$

is a decidable language.

Proof Idea.

1. Convert R to an NFA A using the conversion that we have seen in class.
2. Now use the TM N from Theorem 2 and return the answer obtained on input $\langle A, w\rangle$. □

Decidable Problems Concerning Regular Languages

Theorem 4

The language

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

is a decidable language.

Proof Idea.

Create a TM T that propagates marking labels.
$T =$ "on input $\langle A \rangle$ where A is a DFA:

1. Mark the start state of A
2. Repeat until no new states get marked:
   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**."

□

Decidable Problems Concerning Regular Languages

Theorem 5

The language

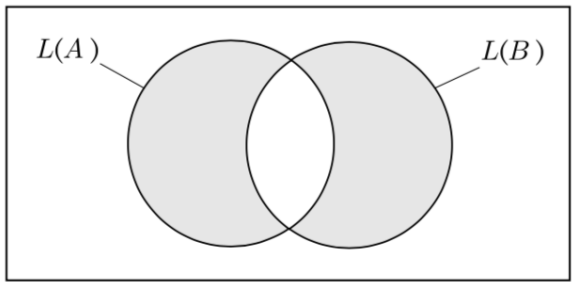
$$EQ_{DFA} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$$

is a decidable language.

Proof Idea.

Construct a DFA C that accepts the symmetric difference.

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



Decidable Problems Concerning Context-Free Languages

Theorem 6

The language

$$A_{CFG} = \{\langle G, w\rangle \,|\, G \text{ is a CFG that generates string } w\}$$

is a decidable language.

Proof Idea.

(It does not work to try all possible derivations, as there is an infinite number of such derivations.)
Every CFG in CNF will generate w in $2n-1$ steps, where $|w| = n$ and provided w can be generated by G.
$\Rightarrow$ So finitely many such derivations need to be checked
$\Rightarrow$ If some derivation generates w, **accept**; otherwise **reject**. □

Decidable Problems Concerning Context-Free Languages

Remarks:

- The above problem is related to compiling programming languages
- There is a more (much more) efficient way of solving the problem (using dynamic programming; CYK algorithm)
- Everything we say about CFGs also applies to PDAs

Decidable Problems Concerning Context-Free Languages

Theorem 7

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

is a decidable language.

Proof Idea.

For each variable determine if we can generate some string.
$\Rightarrow$ Start from terminals and mark them
$\Rightarrow$ Then mark variables that can yield such terminals
$\Rightarrow$ Then mark variables that can yield the previous variables
$\Rightarrow$ etc
In other words, if $A \rightarrow U_1 U_2 \ldots U_k$ is a rule of G, mark A when **each** of $U_1 U_2 \ldots U_k$ has been marked.
So in the end, if the start variable is not marked, **accept**.
otherwise **reject**. □

Decidable Problems Concerning Context-Free Languages

What about

$$EQ_{CFQ} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}?$$

Turns out that it is **not** decidable.
CFLs are not closed under complementation or intersection.

Decidable Problems Concerning Context-Free Languages

Theorem 8

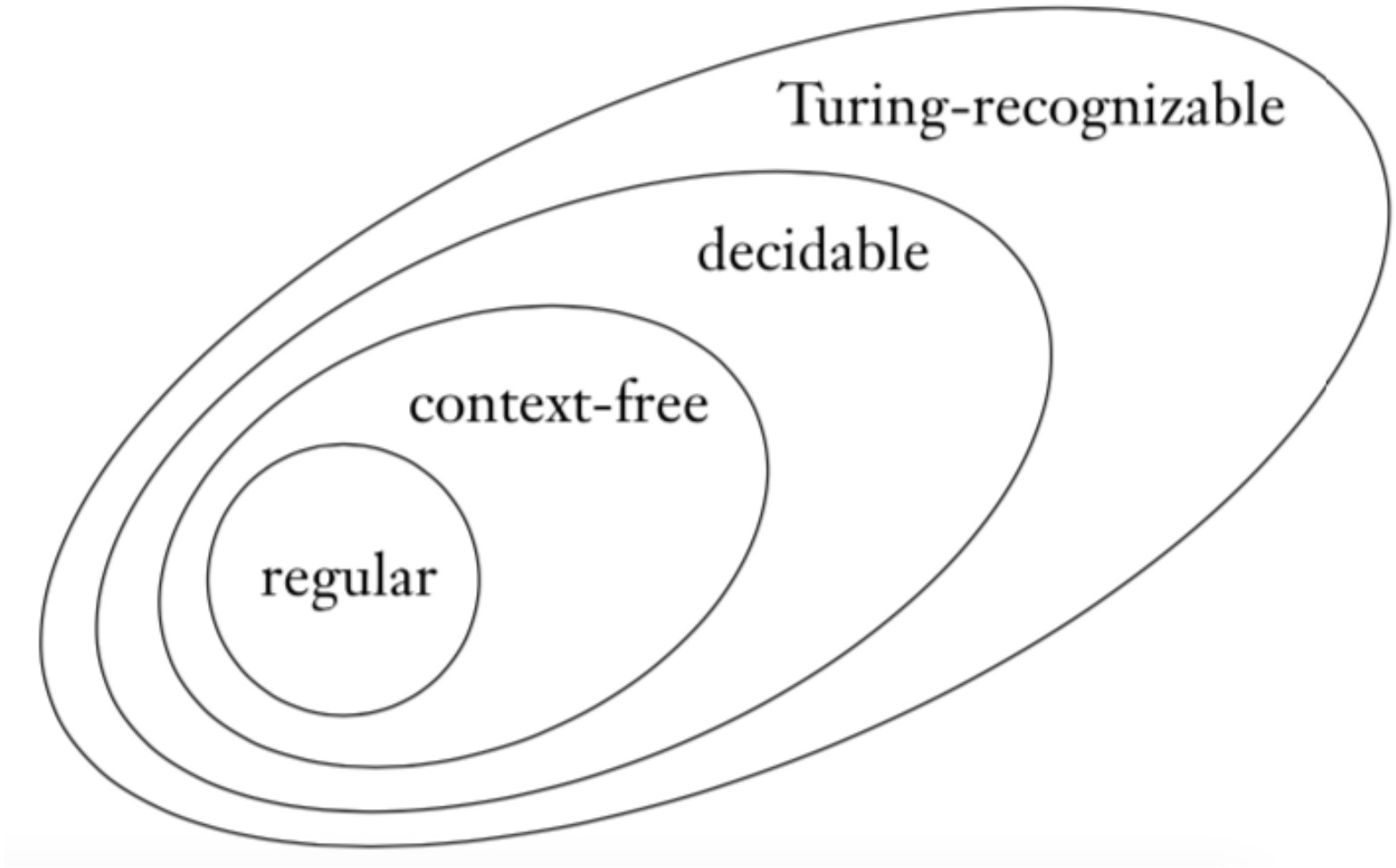
Every context-free language is decidable.

Proof Idea.

Use Theorem 6.
In other words, let G be a CFG for language A.
Then, on input w, use a TM M_G that feeds $\langle G, w \rangle$ into the TM S of Theorem 6.
Return whatever S returns. □

Note that simulating a non-deterministic PDA is a bad idea, because some branches of the computation may go on forever ⇒ We do not have a decider.
So, instead, first convert the PDA into a CFG G.

Decidable Problems Concerning Context-Free Languages



The relationship among classes of languages

Table of Contents

Equal Cardinality

Equal cardinality of infinite sets:

Create a bijection (correspondence) between the two sets A and B.

f is one-to-one: denoted as $f : A \rightarrowtail B$ and means that $f(a) = f(b) \Leftrightarrow a = b$

f is onto: denoted as $f : A \twoheadrightarrow B$ and means $(\forall b \in B)(\exists a \in A)[f(a) = b]$

f is bijection (correspondence): denoted as $f : A \rightarrowtail\!\!\!\!\twoheadrightarrow B$ and means that every element of A maps to a unique element of B and vice versa

For example $\mathcal{N} = \{1, 2, 3. . .\}$ and Even $= \{2, 4, 6, 8, . . .\}$ have the same cardinality.

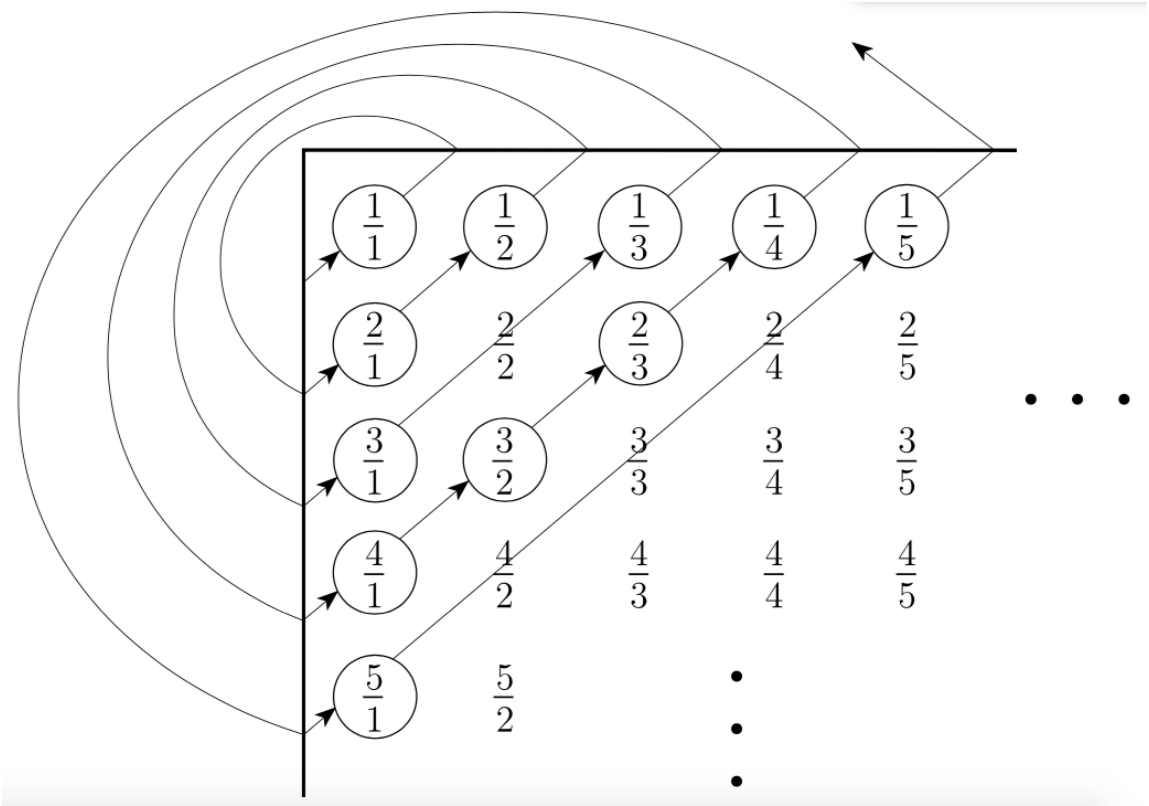
n	$f(n)$
1	2
2	4
3	6
$\vdots$	$\vdots$

Countable Set

Countable Set:

Finite set or has same cardinality as $\mathcal{N}$

For example $|Q| = |\mathcal{N}|$



Countable Set

Theorem 9

$\mathcal{R}$ is uncountable

Proof.

Idea: Diagonalization

n	$f(n)$
1	$3.14159\ldots$
2	$55.55555\ldots$
3	$0.12345\ldots$
4	$0.50000\ldots$
$\vdots$	$\vdots$

Construct a number that does not appear anywhere in the bijection, by selecting a decimal digit in each position that is different from the respective digit in $f(n)$ (e.g. $x = 0.4641\ldots$) $\Rightarrow$ contradict the existence of f. $\square$

Table of Contents

The Acceptance Problem

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

Theorem 10

A_{TM} is Turing-recognizable.

Proof Idea.

$U =$ "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 its accept state, **accept**;
   If M ever enters its reject state, **reject**."

U is an example of the **universal Turing machine** because it is capable of simulating any other Turing machine from the description of that machine. (Such a machine was first used by Turing) □

The Acceptance Problem

Theorem 11

A_{TM} is undecidable.

Proof (to be cont.)

Towards contradiction, assume A_{TM} is decidable, with the Turing machine H being a decider for A_{TM}.

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

Proof (cont.)

Now look at the following TM:

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

1. Run H on input $\langle M, \langle M \rangle \rangle$
2. Output the opposite of what H outputs."

So,

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

What about $D(\langle D \rangle)$?
We have

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

Contradiction either way $\Rightarrow$ H **cannot exist!** □

Turing Unrecognizable Languages

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

Theorem 12

A language L is decidable $\Rightarrow$ L is Turing recognizable and co-Turing recognizable

Proof (to be cont.)

($\Rightarrow$) L decidable. Then L and $\overline{L}$ are both Turing recognizable.
(The complement $\overline{L}$ of a decidable language L is also decidable)

Turing Unrecognizable Languages

Proof (cont.)

($\Leftarrow$) Let M_1 recognize L and M_2 recognize $\overline{L}$. Now consider the following TM M.

$M =$ "On input w:

1. Run both M_1 and M_2 on input w in parallel
2. If M_1 accepts, accept;
   if M_2 accepts, reject"

For every string $w \in \Sigma^*$, we have either $w \in L$ or $w \in \overline{L} \Rightarrow$
Either M_1 or M_2 accepts $w \Rightarrow M$ always halts with an answer $\Rightarrow$
M is a decider for $L \Rightarrow L$ is decidable. □

Turing Unrecognizable Languages

Corollary 13

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

Proof.

If $\overline{A_{TM}}$ was recognizable $\overset{\text{Theorem 12}}{\Longrightarrow}$ A_{TM} is decidable $\Rightarrow$ Contradicts the undecidability of A_{TM} (Theorem 11) $\square$