CS301: Theory of Computation
Sections 4.1 & 4.2 - Decidable Languages and Undecidability

Daniel White

Gettysburg College
Computer Science

Spring 2024

The **acceptance problem** for DFAs asks if a TM exists which can determine whether a given DFA accepts a given input string.

Theorem 4.1 (pg. 194)

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

Proof (high-level). On input $\langle B, w\rangle$:

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

exercise: Discuss some implementation details of the TM used in the proof above. *e.g.* encoding and tape management

Define A_{NFA} and A_{REX} in a similar way for NFAs and regular expressions, respectively.

Theorems 4.2 & 4.2 (pgs. 195-196)

A_{NFA} and A_{REX} are decidable languages.

Proof. We first present a TM N that decides A_{NFA}. Instead of designing N to simulate any given NFA, we first have N ...

exercise: Finish this proof.

Another fundamental class of problem is **emptiness testing**. Given a DFA A, does the recognized language $L(A)$ contain any member?

Theorem 4.4 (pg. 196)

$E_{\text{DFA}} = \{\langle A \rangle \mid A \text{ is a DFA and } L(A) = \varnothing\}$ is a decidable language.

Proof. On input $\langle A \rangle$:

1. Mark the start state of A.
2. Repeat until no new states get marked: Mark any state that has ...
3. ...

exercise: Finish this proof.

homework: 4.10

Theorem 4.8 (pg. 198)

E_{CFG} is a decidable language.

Proof. We will test whether the start variable can generate a string of terminals by determining if every variable can generate a string of terminals. On input $\langle G \rangle$:

1. Mark all terminals in G.
2. Repeat until no new variables get marked: Mark any variable A where ...

exercise: Finish this proof.

homework: 2.18 & 4.14

One of the most philosophically important theorems in the theory of computation: *some problems are algorithmically unsolvable.*

Theorem 4.11 (pg. 202)

A_{TM} is Turing-recognizable, but undecidable.

Proof. Define the TM U so that on input $\langle M, w \rangle$:

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

Note that while U (the so-called *universal Turing machine*) recognizes A_{TM}, it does not decide A_{TM}. This is not proof that A_{TM} is undecidable though ...

By way of contradiction, suppose A_{TM} is decidable and let H be a decider which recogizes A_{TM}. In other words,

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

We further define a decider D so that on input of TM M:

1. Run H on input $\langle M, \langle M\rangle\rangle$, then
2. output the opposite of H's output.

Finally, we evaluate $D(\langle D\rangle)$ ...

exercise: Finish this proof.

homework: Know this proof.

Definition (pgs. 202-203)

Two sets are said to have the same **cardinality**, or size, if there exists a bijection between them.

exercise: Argue that this definition is consistent for finite sets.
exercise: Prove that $\mathbb{N}$ has the same size as $\{2, 4, \ldots\}$.

Definition 4.14 (pg. 203)

A set is **countable** if either it is finite or has the same size as $\mathbb{N}$.

exercise: Prove that $\mathbb{Q}$ is countable.

Definition (pg. 204)

A set is **uncountable** if it is not countable.

Theorem 4.17 (pg. 205)

$\mathbb{R}$ is uncountable.

Proof. By way of contradiction, suppose $\mathbb{R}$ is countable. Then the real numbers are indexable; *i.e.* we can label the reals as r_1, r_2, r_3, and so on ...

exercise: Finish this proof.
exercise: Prove that the set B of all infinite binary strings is uncountable.

Corollary 4.18 (pg. 206)

The set of languages that are not Turing-recognizable is uncountable.

Proof. Fix an alphabet Σ and consider any fixed ordering of the set Σ^* as $\Sigma^*_{\text{ord}} = [w_0, w_1, \ldots]$ For each infinite binary string $\beta \in B$, $\beta = \beta_0\beta_1\beta_2\ldots$, define $L(\beta)$ to be the subset of Σ^* defined as

$$L(\beta) = \{w_i \mid w_i \in \Sigma^*_{\text{ord}} \text{ and } \beta_i = 1\}.$$

Note that, in this way, the set B of infinite binary strings exhaustively (and exactly) generates all languages over Σ. That is, the languages over Σ are in one-to-one correspondence with B, and so must be *uncountable* . . .

exercise: Finish this proof.

$\boxed{\text{homework: Thm 4.22 \& Cor 4.23}}$