FA/B $Halt_{TM}$

CHAPTER 2.3

Pumping lemma for context-free language:

If A is a context-free language, then there is a number *p* (the pumping length) where, if *s* is any string in A of length at least *p*, then *s* may be divided into 5 pieces s = uvxyz satisfying the condition:

1. For each i >= 0, $uv^ixy^iz \in A$
2. |vy| > 0
3. |vxy| <= p

How to prove a language is not context-free

Example 2.36 (page 128) https://www.youtube.com/watch?v=AdfE0lcGaJs

CHAPTER 3

Formal definition of Turing Machine.

A Turing Machine is a 7-tuple, (Q, Σ, Γ, δ, q_0, q_{accept}, q_{reject}):

1. Q is a finite set of states
2. Σ is the input alphabet not containing the blank symbol
3. Γ is the tape alphabet $\sqcup \in \Gamma$, 空 [illegible] $\Sigma \subseteq \Gamma$
4. δ: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function
5. q_0 is the start state
6. q_{accept} is the accept state
7. q_{reject} is the reject state, where $q_{accept} \neq q_{reject}$

Σ does not contain the blank symbol, so the first blank appearing on the tape marks the end of the input.

Configuration of the Turing Machine: A setting of current state, current tape contents, current head location

→ uqv configuration means: current state is q, current tape contents us uv, current head location is the first symbol of v.

Configuration C1 yields configuration C2 if the Turing machine can legally go from C1 to C2 in a single step.

Start configuration: q_0w

Accepting configuration: q_{accept}

Rejecting configuration: q_{reject}

Halting configurations: accepting and rejecting configurations.

A Turing machine M accepts input w if a sequence of configurations $C_1, C_2, C_3, \ldots, C_k$ exists where:

1. C_1 is the start configuration of M on input w
2. Each C_i yields C_{i+1} and
3. C_k is an accepting configuration

The language of M, or the **language recognized by M**, L(M), is the collection of strings that M accepts.

Call a language **Turing-recognizable** if some Turing machine recognizes it.
Three possible outcomes for a Turing machine are **accept**, **reject** and **loop**
Loop means that machine simply does not halt
Deciders are Turing machines that halt on all inputs.
A decider that recognizes some language is also said to **decide** that language.
Call a language **Turing-decidable** if some Turing machine decides it.
Every decidable language is Turing-recognizable.
How to design an algorithm for a Turing machine:
→ Example 3.7 (page 171)
How to write the sequence of configurations for a Turing machine:
→ Figure 3.8 (page 172)

Variants of the Turing machine model: the alternative definitions of Turing machines
Robustness: invariance to certain changes in the definition → the original model and its reasonable variants all have the same power
Multitape Turing machine: is an ordinary Turing machine with several tapes.

- $\delta: Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, S\}^k$
- k: the number of tapes

Nondeterministic Turing machine: a Turing machine that may proceed according to several possibilities.

- $\delta: Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{L, R\})$
- Every nondeterministic Turing machine has an equivalent deterministic Turing machine.

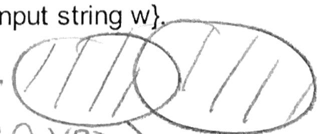
Enumerator: is a Turing machine with an attached printer.
HILBERT'S PROBLEM
Polynomial is a sum of terms, where each **term** is a product of certain variables and a constant, called a **coefficient**.
Root: is an assignment of values to its variables so that the value of the polynomial is 0.
Integral root: is a root where all the variables are assigned integer values.
Church -Turing thesis: the intuitive notion of algorithms equals the Turing machine algorithms.
To describe a Turing machine algorithm:

- Formal description: give details on the Turing machine's states, transition functions and so on.
- Implementation description: use English to describe the way that a Turing machine moves its head and the way it stores data on its tape.
- High-level description: use English to describe an algorithm, ignoring the implementation details

CHAPTER 4

ADFA: is a language expressed as the acceptance problem for DFAs of testing whether a particular deterministic finite automaton accepts a given string
ADFA = {⟨B, w⟩| B is a DFA that accepts input string w}.

Symmetric Difference :



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

A_{DFA} is a decidable language.
$EQ_{DFA} = \{\langle A, B\rangle \mid A$ and B are DFAs and $L(A) = L(B)\}$.
EQ_{DFA} is a decidable language
$A_{NFA} = \{\langle B, w\rangle \mid B$ is an NFA that accepts input string $w\}$.
$A_{REX} = \{\langle R, w\rangle \mid R$ is a regular expression that generates string $w\}$.
$A_{CFG} = \{\langle G, w\rangle \mid G$ is a CFG that generates string $w\}$.
$E_{CFG} = \{\langle G\rangle \mid G$ is a CFG and $L(G) = \emptyset\}$. → it's a decidable language
$EQ_{CFG} = \{\langle G, H\rangle \mid G$ and H are CFGs and $L(G) = L(H)\}$.
Every context free language is decidable.

Regular language → Context-free → Decidable → Recognizable

$A_{TM} = \{\langle M, w\rangle \mid M$ is a TM that accepts $w\}$. → **undecidable**
One-to-one (injective): never maps 2 different elements to the same place, $f(a) \neq f(b)$ whenever $a \neq b$
Onto (surjective): for every $b \in B$, there's an $a \in A$ such that $f(a) = b$
Correspondence (bijective): both one-to-one and onto
A and B are the **same size** if there is a one-to-one, onto function $f: A \to B$
A set A is **countable** if either it is finite or it has the same size as N
The set R of real numbers is **uncountable**.
Each language $A \in L$ has a unique sequence in B.The ith bit of that sequence is a 1 if $s_i \in A$ and is a 0 if s_i not $\in A$, which is called the **characteristic sequence** of A
co-Turing-recognizable: the language that is the complement of a Turing-recognizable language.
A language is **decidable** iff it is Turing-recognizable and co-Turing-recognizable.

CHAPTER 5

Reducibility: the primary method for proving that problems are computationally unsolvable.
A **reduction** is a way of converting one problem to another problem in such a way that a solution to the second problem can be used to solve the first problem.
$HALT_{TM} = \{\langle M, w\rangle \mid M$ is a TM and M halts on input $w\}$. → undecidable
$E_{TM} = \{\langle M\rangle \mid M$ is a TM and $L(M) = \emptyset\}$. → undecidable
$REGULAR_{TM} = \{\langle M\rangle \mid M$ is a TM and $L(M)$ is a regular language$\}$ → undecidable
$EQ_{TM} = \{\langle M_1, M_2\rangle \mid M_1, M_2$ are TMs and $L(M_1) = L(M_2)\}$ → undecidable
Accepting computation history: a sequence of configurations, where C1 is the start of configuration of M on w, C(l) is an accepting configuration of M, and each Ci legally follows from C(i-1) according to the rules of M.
Rejecting computation history: The same, except Cl is a rejecting configuration.
Linear bounded automaton: a restricted type of Turing machine wherein the tape head isn't permitted to move off the portion of the tape containing the input. It has limited amount of memory.
$A_{LBA} = \{\langle M, w\rangle \mid M$ is an LBA that accepts string $w\}$. → Decidable

A_{LBA} is decidable.

Lemma 5.8: Let M be an LBA with q states and g symbols in the tape alphabet. There are exactly qng^n distinct configurations of M for a tape of length n.

$E_{LBA} = \{\langle M\rangle |$ M is an LBA where $L(M) = \emptyset\}$.

E_{LBA} is undecidable.

$ALL_{CFG} = \{\langle G\rangle |$ G is a CFG and $L(G) = \Sigma^*\}$.