1 Prerequisite Definitions

Alphabets Σ, and Γ are finite nonempty sets of symbols.

A *string* is a finite sequence of zero or more symbols from an alphabet.

$\Sigma^\star$ is the set of all strings over alphabet Σ.

ε is the empty string and cannot be in Σ.

A *problem* is a mapping from strings to strings.

A *decision problem* is a problem whose output is yes/no (or often accept/reject).

A decision problem be thought of as the set of all strings for which the function outputs "accept".

A *language* is a set of strings, so any set $S \subseteq \Sigma^\star$ is a language, even $\emptyset$. Thus, decision problems are equivalent to languages.

2 Regular Languages

$L(M)$ is the language accepted by machine M.

A deterministic finite automaton is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite set of states,
- Σ is an alphabet,
- $\delta : Q \times \Sigma \to Q$ is a transition function describing its transitions and labels,
- $q_0 \in Q$ is the starting state, and
- $F \subseteq Q$ is a set of accepting states.

If δ is not fully specified, we assume an implicit transition to an *error state*.

A deterministic finite automaton M accepts input string $w = w_1 w_2 \ldots w_n$ ($w_i \in \Sigma^\star$) if there exists a sequence of states $r_0, r_1, r_2, \ldots, r_n$ ($r_i \in Q$) such that

- $r_0 = q_0$,
- for all $i \in \{1, \ldots, n\}$, $r_i = \delta(r_{i-1}, w_i)$, and
- $r_n \in F$.

$r_0, r_1, r_2, \ldots, r_n$ are the sequence of states visited during the machine's computation.

A non-deterministic finite automaton is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where

- Q, Σ, q_0, F are the same as a deterministic finite automaton's, and
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$.

A non-deterministic finite automaton accepts the string $w = w_1 w_2 \ldots w_n$ ($w_i \in \Sigma^\star$) if there exist a string $y = y_1 y_2 \ldots y_m$ ($y_i \in (\Sigma \cup \{\varepsilon\})^\star$) and a sequence $r = r_0, r_1, \ldots, r_n$ ($r_i \in Q$) such that

- $w = y_1 \circ y_2 \circ \cdots \circ y_m$ (i.e. y is w with some ε inserted),
- $r_0 = q_0$,
- for all $i = \{1, \ldots, m\}$, $r_i \in \delta(r_{i-1}, q_i)$, and
- $r_m \in F$.

The ε-*closure* for any set $S \subseteq Q$ is denoted $E(S)$, which is the set of all states in Q that can be reachable by following any number of ε-transition.

Theorem 1. *A non-deterministic finite automaton can be converted to an equivalent deterministic finite automaton.*

A *regular language* is any language accepted by some finite automaton. The set of all regular languages is called the *class of regular languages*.

Theorem 2. *Regular languages are closed under*

- *Concatenation* $L_1 \circ L_2 = \{x \circ y : x \in L_1 \text{ and } y \in L_2\}$. *Note:* $L_1 \not\subseteq L_1 \circ L_2$.
- *Union* $L_1 \cup L_2 = \{x : x \in L_1 \text{ or } x \in L_2\}$.
- *Intersection* $L_1 \cap L_2 = \{x : x \in L_1 \text{ and } x \in L_2\}$.
- *Complement* $\overline{L} = \Sigma^\star \setminus L = \{x : x \notin L\}$.
- *Star* $L^\star = \{x_1 \circ x_2 \circ \cdots \circ x_k : x_i \in L \text{ and } k \geq 0\}$.

R is a regular expression if R is

- $a \in \Sigma$,
- ε,
- $\emptyset$,
- $R_1 \cup R_2$, or $R_1 | R_2$,
- $R_1 \circ R_2$, or $R_1 R_2$,
- $R_1^\star$,
- Shorthand: $\Sigma = (a_1 | a_2 | \ldots | a_k)$, $a_i \in \Sigma$,

where R_i is a regular expression.

Identities of Regular Languages

- $\emptyset \cup R = R \cup \emptyset = R$
- $\emptyset \circ R = R \circ \emptyset = \emptyset$
- $\varepsilon \circ R = R \circ \varepsilon = R$
- $\varepsilon^\star = \varepsilon$
- $\emptyset^\star = \emptyset$
- $\emptyset \cup R \circ R^\star = R \circ R^\star \cup \varepsilon = R^\star$
- $(a|b)^\star = (a^\star | b^\star)^\star = (a^\star b^\star)^\star = (a^\star | b)^\star = (a | b^\star)^\star = a^\star (b a^\star)^\star = b^\star (a b^\star)^\star$

Theorem 3. *Languages accepted by DFAs = languages accepted by NFAs = regular languages*

Theorem 4. *If L is a finite language, L is regular.*

If a computation path of any finite automaton is longer than the number of states it has, there must be a cycle in that computation path.

Lemma 1 (Pumping Lemma)**.** *Every regular language satisfies the pumping condition.*

Pumping condition: There exists an integer p such that for every string $w \in L$, with $|w| \geq p$, there exist strings $x, y, z \in \Sigma^\star$ with $w = xyz, y \neq \varepsilon, |xy| \leq p$ such that for all $i \geq 0$, $xy^i z \in L$.

Negation of pumping condition: For all integers p, there exists a string $w \in L$, with $|w| \geq p$, for all $x, y, z \in \Sigma^\star$ with $w = xyz, y \neq \varepsilon, |xy| \leq p$, there exists $i \geq 0, i \neq 1$ such that $xy^i z \notin L$.

Limitations of finite automata:

- Only read input once, left to right.
- Only finite memory.

3 Context-Free Languages

A pushdown automaton is a 6-tuple $M = (Q,\Sigma,\Gamma,\delta,q_0,F)$, where

- Q is a finite set of states,
- Σ is its input alphabet,
- Γ is its stack alphabet,
- $\delta : Q\times(\Sigma\cup\{\varepsilon\})\times(\Gamma\cup\{\varepsilon\}) \to 2^{Q\times(\Gamma\cup\varepsilon)}$ is its transition function,
- $q_0 \in Q$ is its starting state, and
- $F \subseteq Q$ is a finite set of accepting states.

Labels: $a,b \to c$: if input symbol is a, and top of stack is b, pop it and push c. In other words, input symbol read, stack symbol popped $\to$ stack symbol pushed, e.g. $0,\varepsilon \to \$$.

Suppose u,v,w are strings of variables and terminals, and there is a rule $A \to w$. From the string uAv, we can obtain uwv. We write $uAv \to uwv$, and say uAv yields uwv.

If $u_1 \to u_2 \to \cdots \to u_k$, then $u_1 \to^\star u_k$, or u_1 derives u_k. There must be a finite number of arrows between u_1 and u_k.

Given a grammar G, the language derived by the grammar is $L(G) = \{w \in \Sigma^\star : S \to^\star w$ and S is the start variable$\}$

Context-free grammar: the lhs of rules is a single variable, rhs is any string of variables and terminals. A *context-free language* is one that can be derived from a context-free grammar. An example context-free grammar is $G = (V,\Sigma,R,\langle\texttt{EXPR}\rangle)$, where $V = \{\langle\texttt{EXPR}\rangle,\langle\texttt{TERM}\rangle,\langle\texttt{FACTOR}\rangle\}$, $\Sigma = \{a,+,\times,(,)\}$, and $R = \{\langle\texttt{EXPR}\rangle \to \langle\texttt{EXPR}\rangle + \langle\texttt{TERM}\rangle|\langle\texttt{TERM}\rangle, \langle\texttt{TERM}\rangle \to \langle\texttt{TERM}\rangle \times \langle\texttt{FACTOR}\rangle|\langle\texttt{FACTOR}\rangle, \langle\texttt{FACTOR}\rangle \to (\langle\texttt{EXPR}\rangle)\}$.

A *left-most derivation* is a sequence $S \to u_1 \to u_2 \to \cdots \to u_k \to w$ where each step applies a rule to the left-most variable. A grammar is *ambiguous* when it has multiple left-most derivations for the same string.

Theorem 5. *A language L is recognized by a pushdown automaton iff L is described by a context-free grammar.*

Theorem 6. *Context-free languages are closed under union, concatenation, star.*

4 Recognizable Languages

Differences from previous models

- The input is written on tape.
- It can write to the tape.
- It can move left and right on tape.
- It halts immediately when it reaches an accepting or rejecting state. The rejecting state must exist but may not be shown.

A deterministic Turing machine is a 7-tuple $M = (Q,\Sigma,\Gamma,\delta,q_0,q_{accept},q_{reject})$, where

- Q is its finite non-empty set of states,
- Σ is its input alphabet,
- Γ is its tape alphabet ($\Sigma \subset \Gamma$ and $\llcorner\!\lrcorner \in \Gamma\setminus\Sigma$),
- $\delta : Q\times\Gamma \to Q\times\Gamma\times\{L,R\}$ is its transition function,
- $q_0 \in Q$ is its starting state,
- $q_{accept} \in Q$ is its accepting state, and
- $q_{reject} \in Q$ is its rejecting state ($q_{reject} \neq q_{accept}$).

Labels: $a \to b,R$: if tape symbol is a, write b and move head right. $a \to R$: if tape symbol is a, move head right. $a,b,c \to R$: if tape symbol is a,b, or c, move head right.

On input x, a Turing machine can (1) accept, (2) reject, or (3) run in an infinite loop.

The language *recognized* by a Turing machine M is $L(M) = \{x :$ on input x, M halts in $q_{accept}\}$. A language is *recognizable* if there exists a Turing machine which recognizes it.

Regular languages $\subseteq$ context-free languages $\subseteq$ decidable languages $\subseteq$ recognizable languages

A *configuration* is a way to describe the entire state of the Turing machine. It is a string aqb where $a \in \Gamma^\star, q \in Q, b \in \Gamma^\star$, which indicates that q is the current state of the Turing machine, the tape content currently is ab and its head is currently pointing at the first symbol of b. Any Turing machine halts if its configuration is of the form $aq_{accept}b$, or $aq_{reject}b$ for any ab. Config(i) uniquely determines Config($i+1$).

Theorem 7. *Every k-tape Turing machine has an equivalent single tape Turing machine.*

If the alphabet of the multitape Turing machine is Γ, we can make the single tape Turing machine's alphabet $(\Gamma\cup\{\#\})\times\{\texttt{normal},\texttt{bold}\}$.

A non-deterministic Turing machine is a 7-tuple $M = (Q,\Sigma,\Gamma,\delta,q_0,q_{accept},q_{reject})$, where the only difference from a deterministic Turing machine is the transition function $delta : Q\times\Gamma \to 2^{Q\times\Gamma\times\{L,R\}}$.

A non-deterministic Turing machine accepts its input iff some node in the configuration tree has q_{accept}. It does not accept its input iff the configuration tree grows forever (infinite loop) or no node in the tree has q_{accept}.

Acceptance of a non-deterministic Turing machine: input w is accepted if there exist configurations $c_0,c_1,\ldots,c_k$ where

- $c_0 = q_{start}w$, and
- $c_i \Rightarrow c_{i+1}$ (c_{i+1} is a possible configuration from c_i, following the transition function δ).

The outcomes could be

- w is accepted, i.e. there exists a node in the tree which is an accepting configuration,
- w is explicitly rejected, i.e. the tree is finite but no node is an accepting configuration (all leaves are rejecting configurations), or
- the non-deterministic Turing machine runs forever on w, i.e. the tree is infinite but no node is an accepting configuration (there might be finite branches terminating in a rejecting configuration in the tree).

A Turing machien is a *decider* if it halts on all inputs, i.e. it either rejects or accepts all inputs.

Theorem 8. *Every non-deterministic Turing machine has an equivalent deterministic Turing machine. If that non-deterministic Turing machine is a decider, there is an equivalent deterministic Turing machine decider.*

Theorem 9. *Recognizable languages are closed under union, intersection, concatenation, star.*

Implementation level description of a multitape Turing machine for $L = \{x\#x : x \in \{0,1\}^\star\}$:

- Scan the first head to the right until it reads a #. Move right. The second head is still at the start of the second tape.
- Repeatedly read symbol from the first tape (reject if the symbol is not 0 or 1), write it to the second tape, and move both heads right, until seeing a blank on the first tape.
- Move the first head left until a # is under it. Replace the symbol with a blank (␣).
- Move both heads left until they reach the start of their respective tapes (using the $ sign hack to mark the start of the tape).
- Repeat until seeing a blank on both tapes.
  - If the symbols on the two tapes differ, reject.
  - Otherwise, move both head right.

$\langle O \rangle$ is a string encoding for the object O.

Cardinality of Sets: two sets A and B have the same *cardinality* if there exists a bijection $f : A \to B$.

$\mathbb{N} = \{1,2,3,\dots\}$ is the set of all natural numbers. A set is *finite* if it has a bijection to $\{1..n\}$ for some natural number n. A set is *countably infinite* if it has the same cardinality as $\mathbb{N}$. A set is *countable* or *at most countable* if it is finite or countably infinite.

Lemma 2. *Any language L is countable.*

Lemma 3. *The set of all Turing machines is countable.*

Lemma 4. *The set $\mathscr{B}$ of all infinite bit-sequences is not countable.*

Lemma 5. $2^{\Sigma^\star}$ *is uncountable.*

$A_{TM} = \{\langle M, w\rangle : M \text{ accepts } w\}$ is recognizable but not decidable.