Symbols

$\cap$	intersection	Σ_ε	$\Sigma \cup \{\varepsilon\}$
$\cup$	union	␣	blank symbol (also □)
$\subseteq$	subset	Γ	stack or tape alphabet
$\subset$	proper subset	Γ_ε	$\Gamma \cup \{\varepsilon\}$
$\not\subset$	not a subset	$\mathcal{P}(Q)$	power set of Q
$\in$	element of	$\mathcal{R}$	all regular expressions over Σ
$\notin$	not an element of	$\times$	Cartesian (cross) product
$\Leftrightarrow$	if and only if	$\neg$	not (negation)
$\rightarrow$	implication	$\wedge$	and (conjunction)
$\forall$	for all	$\vee$	or (disjunction)
$\exists$	there exists	$\#_a(w)$	the number of times symbol a appears in string w
$*$	Kleene star	(a|b)	a or b *(regular expression)*
•	concatenation	$\|w\|$	the number of symbols in string w
δ	transition function	$\{w \mid y\}$	the set of all w such that y is true
ε	empty string (also λ)	w^R	reverse of w
$\varnothing$	empty set	$\langle X \rangle$	encoding of X
Σ	alphabet	$A \leq_P B$	language A is *polynomial reducible* to language B

Definitions

Finite Automaton DFA = $(Q, \Sigma, \delta, q_0, F)$

1. Q is a finite set of *states*
2. Σ is a finite *alphabet*
3. $\delta: Q \times \Sigma \rightarrow Q$ is the *transition function*
4. $q_0 \in Q$ is the *start state*
5. $F \subseteq Q$ is the set of *accept states*

Nondeterministic Finite Automaton NFA = $(Q, \Sigma, \delta, q_0, F)$

1. Q is a finite set of *states*
2. Σ is a finite *alphabet*
3. $\delta: Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q)$ is the *transition function*
4. $q_0 \in Q$ is the *start state*
5. $F \subseteq Q$ is the set of *accept states*

Generalized Nondeterministic Finite Automaton GNFA = $(Q, \Sigma, \delta, q_{\text{start}}, q_{\text{accept}})$

1. Q is a finite set of *states*
2. Σ is a finite *alphabet*
3. $\delta: (Q - \{ q_{\text{accept}} \}) \times (Q - \{ q_{\text{start}} \}) \rightarrow \mathcal{R}$ is the *transition function*
4. q_{start} is the *start state*
5. q_{accept} is the *accept state*

Regular Expression R is a regular expression if R is

1. a for some a in the alphabet Σ
2. ε
3. $\varnothing$
4. $(R_1 \cup R_2)$ where R_1 and R_2 are regular expressions (union)
5. (R_1R_2) where R_1 and R_2 are regular expressions (concatenation), *or*
6. $(R_1{}^*)$ where R_1 is a regular expression (Kleene star).

Grammar G = (V, Σ, R, S)

1. V is a finite set of *variables (non-terminals)*
2. Σ is a finite set of *terminals*, disjoint from V
3. R is a finite set of *rules*
4. $S \in V$ is the *start variable*

Right-Linear Grammar Grammar *G* is *Right-Linear* if

3. All rules in R are of the form $A \rightarrow xB$ or $A \rightarrow x$ where $A, B \in V$, $x \in \Sigma^*$

Context-Free Grammar CFG Grammar *G* is *Context-Free* if

3. All rules in R are of the form $A \rightarrow w$ where $A \in V$, $w \in (V \cup \Sigma)^*$

Greibach Normal Form

A grammar is in Greibach normal form if every rule is of the form $S \rightarrow \varepsilon$ or $A \rightarrow aX$ where S is the start variable, A is any nonterminal, a is any terminal, and X is a (possibly empty) sequence of nonterminals not including S.

Chomsky Normal Form (CNF)

A grammar is in Chomsky normal form if every rule is of one of the following forms

$$\boldsymbol{S \rightarrow \varepsilon}$$
$$\boldsymbol{A \rightarrow BC}$$
$$\boldsymbol{A \rightarrow a}$$

where S is the start variable, a is any terminal, and $A, B,$ and C are any variables—except that B and C may not be the start variable S.

Pushdown Automaton PDA = $(Q, \Sigma, \Gamma, \delta, q_0, F)$

1. Q is a finite set of *states*
2. Σ is a finite *input alphabet*
3. Γ is a finite *stack alphabet*
4. δ: $Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the *transition function*
5. $q_0 \in Q$ is the *start state*
6. $F \subseteq Q$ is the set of *accept states*

Turing Machine TM = $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$

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

Theorems

Pumping Lemma for Regular Languages

If A is a regular language, then there is a number p *(the pumping length)* where, if s is any string in A of length at least p, $|s| \geq p$, then s may be divided into three pieces, $s=xyz$, satisfying all of the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$
2. $|y| > 0$
3. $|xy| \leq p$

Pumping Lemma for Context-Free Languages

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, $|s| \geq p$, then s may be divided into five pieces, $s=uvxyz$, satisfying all of the following conditions:

1. for each $i \geq 0$, $uv^ixy^iz \in A$
2. $|vy| > 0$
3. $|vxy| \leq p$

Language Terminology Equivalences

Recusively Enumerable ≡ Turing-recognizable ≡ Recognizable ≡ Semi-Decidable ≡ Partially Decidable

Recursive ≡ Turing-decidable ≡ Decidable

Decidable Languages

$\mathbf{A_{DFA}} = \{ <B,w> \mid B$ is a DFA that accepts input string $w \}$

$\mathbf{A_{NFA}} = \{ <B,w> \mid B$ is an NFA that accepts input string $w \}$

$\mathbf{A_{REX}} = \{ <R,w> \mid R$ is a regular expression that generates string $w \}$

$\mathbf{E_{DFA}} = \{ <A> \mid A$ is a DFA and $L(A) = \varnothing \}$

$\mathbf{EQ_{DFA}} = \{ <A,B> \mid A$ and B are DFAs and $L(A) = L(B) \}$

$\mathbf{A_{CFG}} = \{ <G,w> \mid G$ is a CFG that generates string $w \}$

$\mathbf{E_{CFG}} = \{ <G> \mid G$ is a CFG and $L(G) = \varnothing \}$

Undecidable Languages

$\mathbf{EQ_{CFG}} = \{ <G,H> \mid G$ and H are CFGs and $L(G) = L(H) \}$

$\mathbf{A_{TM}} = \{ <M,w> \mid M$ is a Turing Machine that accepts $w) \}$

$\mathbf{S_{TM}} = \{ <M> \mid M$ is a Turing Machine that does not accept $M) \}$