Words and Languages

alphabet	—	non-empty finite set
word (over V)	—	(finite) sequence of letters (of V)
λ	—	empty word
V^* (resp. V^+)	—	set of all (non-empty) words over V
$\#_a(w)$	—	number of occurrences of letter a in word w
$\lvert w\rvert = \sum_{a\in V} \#_a(w)$	—	length of word $w \in V^*$
language (over V)	—	subset of V^*

We say that two languages L_1 and L_2 are equal iff $L_1 \setminus \{\lambda\} = L_2 \setminus \{\lambda\}$.

Parikh Vectors and Semi-linear Sets

For $V = \{a_1, a_2, \ldots, a_n\}$,
Parikh vector of $w \in V^*$ — $\Psi(w) = (\#_{a_1}(w), \#_{a_2}(w), \ldots, \#_{a_n}(w))$,
Parikh set of $L \subset V^*$ — $\Psi(L) = \{Psi(w) \mid w \in L\}$

A set $M \subset \mathbb{N}^n$ is called semi-linear if and only if there are natural numbers $m \geq 1$ and $r_i \geq 1$, $1 \leq i \leq m$, and vectors $\underline{a_{ij}} \in \mathbb{N}^m$, $1 \leq i \leq m,\ 0 \leq j \leq r_i$, such that

$$M = \bigcup_{i=1}^{m} \{\underline{a_{i0}} + \sum_{j=1}^{r_i} \alpha_{ij}\underline{a_{ij}} \mid \alpha_{ij} \in \mathbb{N} \text{ for } 1 \leq j \leq r_i\}\,.$$

A language L is called semi-linear if its Parikh set $\Psi(L)$ is semi-linear.

Theorem: The intersection of two semi-linear sets is semi-linear, too.

Grammars and Languages

Definition:
i) A phrase structure grammar is a quadruple $G = (N, T, P, S)$, where
— N and T are alphabets (sets of nonterminals and terminals, resp.),
— $V_G = N \cup T$, $N \cap T = \emptyset$,
— P is a finite subset of $(V^* \setminus T^*) \times V^*)$ (set of rules/productions),
(instead of (α, β) we write $\alpha \to \beta$),
— $S \in N$ (axiom/start symbol).

ii) We say that x directly derives (generates) y (written as $x \Longrightarrow_G y$) iff

$$x = x_1 \alpha x_2, \quad y = x_1 \beta x_2, \quad \alpha \to \beta \in P.$$

iii) The language generated by G is defined as

$$L(G) = \{z \mid z \in T^* \text{ and } S \Longrightarrow_G^* z\}$$

where $\Longrightarrow_G^*$ is the reflexive and transitive closure of $\Longrightarrow_G$.

Examples

$G_1 = (\{S\},\ \{(,),[,]\},\ P_1, S)$

$P_1 = \{S \to SS,\ S \to (S),\ S \to [S],\ S \to (\ ),\ S \to [\ ]\}$

$L(G_1)$ is the set of all correctly bracketed expressions over the pairs $(,)$ and $[,]$ of brackets

$G_2 = (\{S, \#, \S, A, B, C, \},\ \{a, b\},\ P_2,\ S)$

$$\begin{aligned} P_2 = \{ & S \to bbabb,\ S \to \#Aa\S, \\ & \#Aa \to \#aaA,\ aAa \to aaaA,\ aA\S \to aB\S, \\ & aB \to Ba, \#B \to \#A, \#B \to \#C, \\ & \#Ca \to bbaC, aCa \to aaC, aC\S \to abb\} \end{aligned}$$

$L(G_2) = \{bba^{2^n}bb \mid n \geq 0\}$

Types of Grammars and Languages

Definition:

i) G is called monotone, if $|\alpha| \leq |\beta|$ holds for all rules $\alpha \rightarrow \beta$ of P.

ii) G is called context-free, if all rules of P are of the form $A \rightarrow w$ with $A \in N$ and $w \in V^*$.

iii) G is called regular, if all rules of P are of the form $A \rightarrow wB$ or $A \rightarrow w$ with $A, B \in N$ and $w \in T^*$.

iv) A language L is called monotone or context-free or regular, iff $L = L(G)$ for some monotone or context-free or regular grammar G, respectively.

We denote the families of all regular, context-free and monotone languages by $\mathcal{L}(REG)$, $\mathcal{L}(CF)$ and $\mathcal{L}(CS)$, respectively.
$\mathcal{L}(RE)$ denotes the family of all languages which can be generated by phrase-structure grammars.

Normal Forms

Theorem:

i) For any language $L \in \mathcal{L}(RE)$, there is a phrase-structure grammar $G = (N, T, P, S)$ such that $L = L(G)$ and P has only rules of the forms $A \to B$, $A \to BC$, $AB \to CD$, $A \to a$ and $A \to \lambda$, where $A, B, C, D \in N$ and $a \in T$.

ii) For any language $L \in \mathcal{L}(CS)$, there is a monotone grammar $G = (N, T, P, S)$ such that $L = L(G)$ and P has only rules of the forms $A \to B$, $A \to BC$, $AB \to CD$ and $A \to a$, where $A, B, C, D \in N$ and $a \in T$.

iii) For any language $L \in \mathcal{L}(CF)$, there is a context-free grammar $G = (N, T, P, S)$ such that $L = L(G)$ and P has only rules of the forms $A \to BC$ and $A \to a$, where $A, B, C \in N$ and $a \in T$.

iv) For any language $L \in \mathcal{L}(REG)$, there is a regular grammar $G = (N, T, P, S)$ such that $L = L(G)$ and P has only rules of the forms $A \to aB$ and $A \to a$, where $A, B \in N$ and $a \in T$.

Pumping Lemmas and Parikh's Theorem

Theorem: a) Let L be a regular language. Then there is a constant k (which depends on L) such that, for any word $z \in L$ with $|z| \geq k$, there are words u, v, w which satisfy the following properties:

i) $z = uvw$,
ii) $|uv| \leq k$, $|v| > 0$, and
iii) $uv^iw \in L$ for all $i \geq 0$.

b) Let L be context-free language. Then there is a constant k (which depends on L) such that, for any word $z \in L$ with $|z| \geq k$, there are words u, v, w, x, y which satisfy the following properties:

i) $z = uvwxy$,
ii) $|vwx| \leq k$, $|vx| > 0$, and
iii) $uv^iwx^iy \in L$ for all $i \geq 0$.

Theorem: For any context-free language L, $\Psi(L)$ is semi-linear.

Lindenmayer Systems

Definition: i) An extended tabled Lindenmayer system (abbr. by ET0L) with n tables is an $(n+3)$-tuple $G=(V,T,P_1,P_2,\ldots,P_n,w)$, where

– V is a finite alphabet, T is a non-empty subset of V,

– for $1\leq i\leq n$, P_i is a finite subset of $V\times V^*$ such that, for any $a\in V$, there is a pair (a,w_a) in P_i,

– $w\in V^+$.

ii) We say that x directly derives (generates) y (written as $x\Longrightarrow_G y$) iff there is an i, $1\leq i\leq n$ such that

$$x=x_1x_2\ldots x_m,\ x_j\in V \text{ for } 1\leq j\leq m,\ \ y=y_1y_2\ldots y_m,$$
$$x_j\rightarrow y_j\in P_i \text{ for } 1\leq j\leq m$$

iii) The language generated by G is defined as

$$L(G)=\{z\mid z\in T^* \text{ and } w\Longrightarrow_G^* z\}$$

where $\Longrightarrow_G^*$ is the reflexive and transitive closure of $\Longrightarrow_G$.

Examples

$H_1 = (\{a,b\},\ \{a,b\},\ \{a \to aa, b \to b\},\ bbabb)$
$L(H_1) = \{bba^{2^n}bb \mid n \geq 0\}$

$H_2 = (\{a,b\},\ \{a\},\ \{a \to a,\ a \to aa,\ b \to b,\ b \to \lambda\},\ ab)$
$L(H_2) = \{a^n \mid n \geq 1\}$

$H_3 = (\{a,b,c\},\ \{a,b\},\ P_1,\ P_2,\ ca)$
$P_1 = \{a \to aa, b \to b, c \to ca\}$ and $P_2 = \{a \to b, b \to bbb, c \to a\}$

$$L(H_3) = \{a^{2^m}b^{2^n-1} \mid m \geq 1, n \geq 1\} \cup \{b^{3^k(2^m+3\cdot 2^{n-1})} \mid k \geq 0, m \geq 0, n \geq 1\}$$

Types of Lindenmayer Systems

$\mathcal{L}(ET0L)$ — family of all languages generated by ET0L systems

We omit the letter E if the generating system satisfies $V = T$.

We omit the letter T if the generating system satisfies $n = 1$ (non-tabled case).

We add the letter D if the generating system is deterministic, i.e., for all $1 \leq i \leq n$ and all $a \in V$, there is exactly one rule with left side a in P_i.

Generative Power I

$$
\begin{tikzcd}
 & & & \mathcal{L}(RE) \\
 & & & \mathcal{L}(CS) \arrow[u] \\
 & \mathcal{L}(ET0L) \arrow[urr] & & \\
\mathcal{L}(EDT0L) \arrow[ur] & \mathcal{L}(T0L) \arrow[u] & \mathcal{L}(E0L) \arrow[ul] & \\
\mathcal{L}(DT0L) \arrow[u] \arrow[ur] & \mathcal{L}(ED0L) \arrow[ul] \arrow[ur] & \mathcal{L}(0L) \arrow[ul] \arrow[u] & \mathcal{L}(CF) \arrow[ul] \arrow[uuu] \\
 & \mathcal{L}(D0L) \arrow[ul] \arrow[u] \arrow[ur] & & \mathcal{L}(REG) \arrow[u]
\end{tikzcd}
$$

Operations I

X and Y — alphabets
L, L_1 and L_2 — languages over X, K — language over Y

$L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1,\ w_2 \in L_2\}$ (product, concatenation)

$L^0 = \{\lambda\}$ and $L^{i+1} = L^i \cdot L$ for $i \geq 0$ (power)

$L^+ = \bigcup_{i\geq 1} L^i$ and $L^* = \bigcup_{i\geq 0} L^i$ (Kleene-closure)

A mapping $h : X^* \to Y^*$ is a (homo)morphism if $h(w_1w_2) = h(w_1)h(w_2)$ for all $w_1, w_2 \in X^*$

$h(L) = \{h(w) \mid w \in L\}$ and $h^{-1}(K) = \{w \mid h(w) \in K\}$

Operations II

A substitution $\sigma : X^* \rightarrow Y^*$ is defined inductively as follows:
– $\sigma(\lambda) = \{\lambda\}$,
– $\sigma(a)$ is a finite subset of Y^* for any $a \in X$,
– $\sigma(wa) = \sigma(w)\sigma(a)$ for $w \in X^*$ and $a \in X$.

For a language $L \subseteq X^*$, $\sigma(L) = \bigcup_{w \in L} \sigma(w)$.

A substitution σ (or homomorphism h) is called λ-free iff $\lambda \notin \sigma(a)$ (or $h(a) \neq \lambda$) for all $a \in X$.

Closure Properties I

Let $\tau : (2^X)^n \to 2^X$ be an n-ary operation on language. A family $\mathcal{L}$ is closed under τ, if $\tau(L_1, L_2, \ldots, L_n) \in \mathcal{L}$ holds for all $L_1, L_2, \ldots, L_n \in \mathcal{L}$.

	union	product	Kleene-closure	morph.	inverse morph.	intersect. with reg. sets
$\mathcal{L}(RE)$	+	+	+	+	+	+
$\mathcal{L}(CS)$	+	+	+	−	+	+
$\mathcal{L}(CF)$	+	+	+	+	+	+
$\mathcal{L}(REG)$	+	+	+	+	+	+
$\mathcal{L}(ET0L)$	+	+	+	+	+	+
$\mathcal{L}(EDT0L)$	+	+	+	+	−	+
$\mathcal{L}(E0L)$	+	+	+	+	−	+
$\mathcal{L}(T0L)$	−	−	−	−	−	−
$\mathcal{L}(DT0L)$	−	−	−	−	−	−
$\mathcal{L}(0L)$	−	−	−	−	−	−
$\mathcal{L}(D0L)$	−	−	−	−	−	−

Closure Properties II

Theorem:
The families $\mathcal{L}(REG)$ and $\mathcal{L}(CS)$ are closed under complement, but $\mathcal{L}(CF)$ and $\mathcal{L}(RE)$ are not closed under complement.

Theorem: A language L over X is regular if and only if L can be generated by a finite number of iterated applications of the operations union, product and Kleene-closure $*$ starting with to the sets $\emptyset$, $\{\lambda\}$ and $\{x\}$, $x \in X$.

Closure Properties III

For a language $L \subset V^*$,

$sub(w) = \{w' \mid w = w_1 w' w_2 \text{ for some } w \in L, w_1, w', w_2 \in V^*\}$,
$pref(w) = \{w' \mid w = w' w_2 \text{ for some } w \in L, w', w_2 \in V^*\}$,
$suff(w) = \{w' \mid w = w_1 w' \text{ for some } w \in L, w_1, w' \in V^*\}$,

Theorem:
i) For any regular language L, the sets $sub(L)$, $pref(L)$ and $suff(L)$ are regular, too.
ii) For any context-free language L, the sets $sub(L)$, $pref(L)$ and $suff(L)$ are context-free, too.

TURING Machines

Definition: A (non-deterministic) TURING machine is a seven-tuple $\mathcal{M} = (\Gamma, X, *, Z, z_0, Q, F, \delta)$, where

— Γ is an alphabet (of tape symbols), $X \subseteq \Gamma$ is an alphabet (of input symbols), and $*$ is a special symbol,

— Z is a finite set (of states), $z_0 \in Z$ is the initial state, $Q \subseteq Z$ is the set of halt states, $F \subseteq Q$ is the set of accepting states, and

— $\delta : (Z \setminus Q) \times (\Gamma \cup \{*\}) \to 2^{Z \times ((\Gamma \cup \{*\}) \times \{R,L,N\}}$ is a (total) function.

Intuitively, a TURING machine consists of a unit (storing the state), an infinite tape which cells a filled with letters from $\Gamma \cup \{*\}$ and a read/write head. If a machine is in a state z and reads the symbol a in some cell and $(z', a', r) \in \delta(z, x)$, then it changes the state from z to z', replaces a by a' and moves the head one cell to the right, if $r = R$, or one cell to the left, if $r = L$, or does not move the head, if $r = N$. $\mathcal{M}$ halts if it reaches a state of Q.

TURING **Machines and Languages**

A TURING machine $\mathcal{M}$ is called deterministic if f maps $(Z \setminus Q) \times (\Gamma \cup \{*\})$ into $Z \times ((\Gamma \cup \{*\}) \times \{R, L, N\}$.

Definition: The set $T(\mathcal{M})$ of words accepted by a TURING machine $\mathcal{M}$ consists of all words z such that $\mathcal{M}$ reaches a state in F if it starts in state z_0 with z written on the tape (i.e., the letters of z are written in consecutive cells and all other cells are filled with $*$) and the head positioned on the first letter of z.

Definition: We say that a language L is decidable if there exists a deterministic TURING machine $\mathcal{M}$ such that
— $L = T(\mathcal{M})$ and
— $\mathcal{M}$ halts on any input.

Turing Machines – Example

$$\mathcal{M} = (\{a,b\},\ \{a\},\ *,\ Z,\ \{z_5,z_6\},\ \{z_5\},\ \delta)$$

$$Z = \{z_0,\ z_1,\ z_2,\ z_3,\ z_4,\ z_5,\ z_6\}$$

δ	z_0	z_1	z_2	z_3	z_4
$*$	$(z_0,*,N)$	$(z_5,*,N)$	$(z_4,*,L)$	$(z_6,*,N)$	$(z_0,*,R)$
a	(z_1,a,R)	(z_2,b,R)	(z_3,a,R)	(z_2,b,R)	(z_4,a,L)
b	$(z_0,*,N)$	(z_1,b,R)	(z_2,b,R)	(z_3,b,R)	(z_4,b,L)

$$T(\mathcal{M}) = \{a^{2^n} \mid n \geq 0\}$$

Generative Power II

Theorem:
A language L is in $\mathcal{L}(RE)$ if and only if there is a (deterministic or non-deterministic) $\textsc{Turing}$ machine $\mathcal{M}$ such that $\mathcal{M}$ accepts the language L (i.e., $T(\mathcal{M}) = L$).

A non-deterministic $\textsc{Turing}$ machine is called a linearly bounded automaton if, for any w, the head position while working on the input w is restricted to the cells in which the letters of w are written, the cell before w and the cell after w.

Theorem:
A language L is in $\mathcal{L}(CS)$ if and only if there is a linearly bounded automata $\mathcal{M}$ such that $\mathcal{M}$ accepts the language L (i.e., $T(\mathcal{M}) = L$).

Decision Problems

Membership Problem:	Given grammar/system G and word w Decide whether or not $w \in L(G)$.
Emptiness Problem:	Given grammar/system G Decide whether or not $L(G) = \emptyset$.
Finiteness Problem:	Given grammar/system G Decide whether or not $L(G)$ is a finite language.
Equivalence Problem:	Given grammars/systems G_1 and G_2 Decide whether or not $L(G_1) = L(G_2)$.

Decidability Properties I

	membership problem	emptiness problem	finiteness problem	equivalence problem
$\mathcal{L}(RE)$	$-$	$-$	$-$	$-$
$\mathcal{L}(CS)$	$+$	$-$	$-$	$-$
$\mathcal{L}(CF)$	$+$	$+$	$+$	$-$
$\mathcal{L}(REG)$	$+$	$+$	$+$	$+$
$\mathcal{L}(ET0L)$	$+$	$+$	$+$	$-$
$\mathcal{L}(EDT0L)$	$+$	$+$	$+$	$-$
$\mathcal{L}(E0L)$	$+$	$+$	$+$	$-$
$\mathcal{L}(T0L)$	$+$	t	$+$	$-$
$\mathcal{L}(DT0L)$	$+$	t	$+$	$-$
$\mathcal{L}(0L)$	$+$	t	$+$	$-$
$\mathcal{L}(D0L)$	$+$	t	$+$	$+$

Decidability Properties II

Theorem:
i) The membership problem for semi-linear sets is decidable.
ii) For two semi-linear sets M_1 and M_2 (given by their sets of vectors), it is decidable whether or not $M_1 \subseteq M_2$ holds.

Theorem:
i) Let G be a regular grammar. Then there exists an algorithm which decides whether or not $w \in L(G)$ with a time bound $O(|w|)$, i.e., there is a constant c such that the algorithm stops after at most $c|w|$ steps).
ii) Let G be a context-free grammar. Then there exists an algorithm which decides whether or not $w \in L(G)$ with a time bound $O(|w|^3)$. $\square$

P versus NP I

Definition:

The set **P** is defined as the set of all languages which are decidable in polynomial time (by deterministic TURING machines).

The set **NP** is defined as the set of all languages which can be accepted in polynomial time (by non-deterministic TURING machines).

Definition:
A language L is called **NP**-complete if the following two conditions are satisfied:
— $L \in \mathbf{NP}$ and
— any language $L' \in \mathbf{NP}$ can be polynomially transformed to L (i.e., there is a mapping h such that $h(w)$ can be computed with a polynomial time bound and $h(w) \in L$ holds if and only if $w \in L'$).

P versus NP II

Theorem: The problem 3-SAT defined as

> Given a finite set of disjunctions of three literals, decide
> whether there is an assignment such that any disjunction gets true.

is **NP**-complete.

Theorem: The following assertions are equivalent:
i) **P**=**NP**.
ii) All **NP**-complete language are in **P**.
iii) There is an **NP**-complete language which is in **P**.

Theorem: If a language L' is a **NP**-complete and L' can be polynomially transformed to $L \in \mathbf{NP}$, then L is **NP**-complete, too.