

Models of Computation

6: Probabilistic automata, Pushdown automata, Context-free languages

Probabilistic automaton

- Let s_1, \dots, s_n be the states of the **probabilistic automaton** PA . Reading an input symbol x in state s the automaton PA goes to state s_i with probability $p_i(s, x)$, where for every s and for x :

$$\sum_{i=1}^n p_i(s, x) = 1, \quad \text{and} \quad p_i(s, x) \geq 0, i = 1, \dots, n.$$

- Instead of the initial state, there is a **distribution of initial states**, i.e. every state is an initial state with a fixed probability.
- The accepted language $L(PA, S_1, \eta)$ depends on the **final states** S_1 and the so-called **cutting point** η , $0 \leq \eta < 1$.
- The **accepted language** $L(PA, S_1, \eta)$ is the set of words, for which PA reaches a state in S_1 with a probability greater than η .

Probabilistic automaton

- An **n -dimensional stochastic matrix** $(p_{ij})_{1 \leq i, j \leq n}$ is a square matrix, for which
 - 1.) $p_{ij} \geq 0 \quad (1 \leq i, j \leq n),$
 - 2.) $\sum_{j=1}^n p_{ij} = 1 \quad (1 \leq i \leq n).$
- An **n -dimensional stochastic row vector (column vector)** an n -dimensional row vector (column vector) whose components are non-negative and the sum of the components is 1.
- If only one component of the stochastic row vector is 1, then it is called a **coordinate vector**.
- The n -dimensional unit matrix E_n is a stochastic matrix.

Probabilistic automaton

- A **finite probabilistic automaton** over an alphabet V is a triple $PA = (S, s_0, M)$, where
 - $S = \{s_1, \dots, s_n\}$ is a finite, nonempty set of states,
 - s_0 is a n -dimensional stochastic row vector, the distribution of the initial states
 - M is a mapping that maps V to the set of n -dimensional stochastic matrices.
- For $x \in V$, the (i,j) -th element of the matrix $M(x)$ is $p_j(s_i, x)$, it is the probability that reading x in state s_i , PA goes to state state s_j .

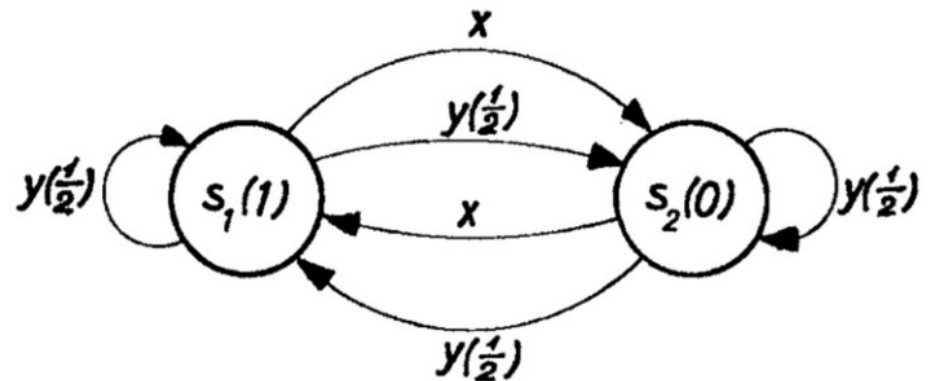
Probabilistic automaton

- Example: Consider the following probabilistic automaton: $PA = (\{s_1, s_2\}, (1,0), M)$ over the alphabet $\{x,y\}$, where

$$M(x) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, M(y) = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$

- The initial distribution shows that the initial state is s_1 .

- The state transition digram:



Probabilistic automaton

- Let $PA = (S, s_0, M)$ be a finite probabilistic automaton over alphabet V . The function M on V can be extended to V^* as follows:
- $\hat{M}(\varepsilon) := E_n$
- $\hat{M}(x_1 \dots x_n) := M(x_1)M(x_2)\dots M(x_n)$, where $k \geq 2$, $x_i \in V$.
- Instead of \hat{M} , we write M hereafter.
- For some word $w \in V^*$, the (i,j) -th element of $M(w)$ is $p_j(s_i, w)$, which is the probability that processing w in state s_i the automaton goes to state s_j .

Probabilistic automaton

- Let $PA = (S, s_0, M)$ be a finite probabilistic automaton over alphabet V , and $w \in V^*$. The stochastic row vector $s_0M(w)$, denoted by $PA(w)$, is the **state distribution resulting from w** .
- Note: $PA(\varepsilon) = s_0$.

Probabilistic automaton

- Let $PA = (S, s_0, M)$ be a finite probabilistic automaton over alphabet V , $0 \leq \eta < 1$, and \bar{s}_1 is an n -dimensional column vector, all elements of which are either 0 or 1. \bar{s}_1 can be understood as a **membership function**.
- The language $L(PA, \bar{s}_1, \eta)$ **accepted with \bar{s}_1 with cut point η** is: $L(PA, \bar{s}_1, \eta) = \{ w \in V^* \mid s_0 M(w) \bar{s}_1 > \eta \}$.
- A language L is called **η -stochastic** if \exists probabilistic finite automaton $PA = (S, s_0, M)$ and column vector \bar{s}_1 , s.t. $L = L(PA, \bar{s}_1, \eta)$ holds.
- A language L is called stochastic if it is η -stochastic for a $0 \leq \eta < 1$.

Probabilistic automaton

- Example: Consider the automaton in the previous example. It holds that
 - $PA(x^n) = (1, 0)M(x^n) = (1, 0)$ if n is even,
 - $PA(x^n) = (0, 1)$, if n is odd, and
 - $PA(w) = (1/2, 1/2)$ if w contains at least one y .

- Thus, for $s_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

$$L(PA, s_1, \eta) = \begin{cases} V^* - (xx)^* & \text{if } 0 \leq \eta < 1/2, \\ x(xx)^* & \text{if } 1/2 \leq \eta < 1. \end{cases}$$

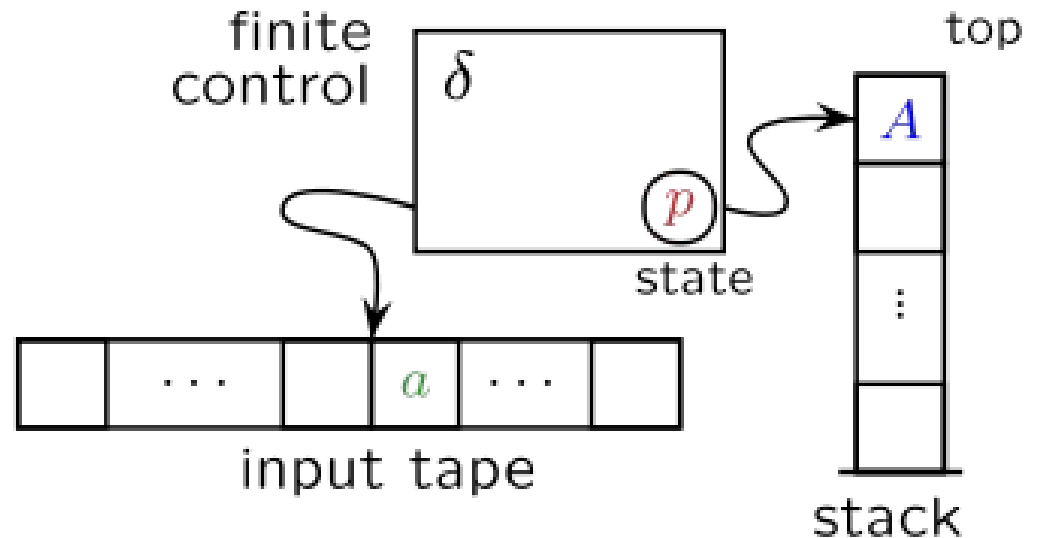
- Thus, $V^* - (xx)^*$ is, e.g., a 1/3-stochastic language, while $x(xx)^*$ is, e.g., a 2/3-stochastic language. Therefore, both are stochastic languages.

Regular and (η -)stochastic languages

- **Theorem** [Rabin 1963]: All regular languages are stochastic, but not all stochastic language is regular.
- **Theorem** [Rabin 1963]: All 0-stochastic languages are regular.

Pushdown automaton (PDA)

- A pushdown automaton (PDA) is a generalization of a finite automaton with (potentially) infinite stack and finite control.
- The new data is added to the top of the stack, and removed in reverse order.
- The stack is a last in, first out (LIFO) data structure.



Pushdown automata

- A **pushdown automaton (PDA)** is a 7-tuple $A = (Z, Q, T, \delta, z_0, q_0, F)$, where
 - Z is a finite set of stack symbols (stack alphabet),
 - Q is a finite set of states,
 - T is the finite set of input symbols (input alphabet),
 - $\delta : Z \times Q \times (T \cup \{\varepsilon\}) \rightarrow P(Z^* \times Q)$ is the transition function,
 - where $P(X)$ is set of finite subsets of X ,
(example: $\delta(z, q, a) = \{(z', q'), (z'', q'')\}$,
note: non-deterministic by default).
 - $z_0 \in Z$ is the initial stack symbol,
 - $q_0 \in Q$ is the initial state,
 - $F \subseteq Q$ is the set of accepting states or final states.

PDA

- The symbol at the top of the stack, the current state, and the input symbol together determine the transition.
- At each step, the automaton takes one element from the top of the stack (**pop**) and writes several instead (0, 1, 2, . . . symbols) (**push**).
- If $\delta(z, q, \varepsilon)$ is not empty, then so-called **ε -transition** (ε -step, ε -movement) can be performed, which allows to change the state and modify the top of the stack without reading a symbol from the input tape.
- ε -transition is possible even before reading the first input symbol or even after reading the last input symbol.

PDA

- The **configuration of the PDA** is a word of a form of zqw , where
 - $z \in Z^*$ is the current content of the stack,
 - $q \in Q$ is the current state, and
 - $w \in T^*$ is the unprocessed part of the input.
- z has its first letter at the bottom of the stack, and its last letter at the top of the stack.
- The reading head of the input is on the first letter w .
- The symbol on the left of q is the symbol on the top of the stack and the symbol on the right of q is the next letter of the input to be processed.
- The initial configuration of the PDA $A=(Z,Q,T,\delta,z_0,q_0,F)$ for input $w \in T^*$ is z_0q_0w .

PDA – operations

- Let $t \in T \cup \{\varepsilon\}$, $q, r \in Q$ and $z \in Z$
 - $(\varepsilon, r) \in \delta(z, q, t)$: element z can be removed from the stack (**POP** operation)
 - $(z, r) \in \delta(z, q, t)$: the contents of the stack may remain unchanged
 - $(z', r) \in \delta(z, q, t)$: z can be replaced with z' at the top of the stack
 - $(zz', r) \in \delta(z, q, t)$: we can put z' on top of the stack (**PUSH** operation)
 - Other possibilities, for example $(zz'z'', r) \in \delta(z, q, t)$: we can put $z'z''$ on top of the stack, z'' will be on top ($z'', z' \in Z$).
 - In general $(w, r) \in \delta(z, q, t)$, where $w \in Z^*$. The symbol z is replaced by the word w . The last letter of w is on the top of the stack.

PDA – reduction

- The PDA A **reduces** the configuration $\alpha \in Z^*QT^*$ to a configuration $\beta \in Z^*QT^*$ **in one step**, denoted by $\alpha \Rightarrow_A \beta$, if $\exists z \in Z, q, p \in Q, a \in T \cup \{\varepsilon\}, x, y \in Z^*$, and $w \in T^*$, s.t. $(y, p) \in \delta(z, q, a)$ and $\alpha = xzqaw$ and $\beta = xypw$.
- Examples:
 - if $\delta(c, q_1, a) = \{(dd, q_2), (\varepsilon, q_4)\}$ and $z_0cddc q_1$ is a configuration, then
 - $z_0cddc q_1 ababba \Rightarrow_A z_0cddd d q_2 babba$ and
 - $z_0cddc q_1 ababba \Rightarrow_A z_0cdd q_4 babba$ also holds.
 - if $\delta(c, q_3, \varepsilon) = \{(dd, q_2)\}$ and $z_0cddc q_3 ababba$ is a configuration, then
 - $z_0cddc q_3 ababba \Rightarrow_A z_0cddd d q_2 ababba$
 - if $\delta(c, q_5, \varepsilon) = \emptyset$ and $\delta(c, q_5, a) = \emptyset$, then \nexists configuration C s.t. $z_0cc q_5 aab \Rightarrow_A C$.

PDA – reduction

- The PDA A **reduces** the configuration $\alpha \in Z^*QT^*$ to a configuration $\beta \in Z^*QT^*$, denoted by $\alpha \Rightarrow^*_A \beta$, if
 - either $\alpha = \beta$,
 - or $\exists \alpha_1, \dots, \alpha_n$ a finite sequence of words, s.t.
 $\alpha = \alpha_1$, $\beta = \alpha_n$ and $\alpha_i \Rightarrow_A \alpha_{i+1}$, $1 \leq i \leq n - 1$.
- The relation $\Rightarrow^*_A \subseteq Z^*QT^* \times Z^*QT^*$ is the reflexive and transitive closure of relation \Rightarrow_A .
- Example:
 - If $\delta(d, q_6, b) = \{(\varepsilon, q_5)\}$ and $\delta(d, q_5, \varepsilon) = \{(dd, q_2), (\varepsilon, q_4)\}$ then
 - $\#cddq_6bab \Rightarrow_A \#cdq_5ab \Rightarrow_A \#cddq_2ab$ and
 - $\#cddq_6bab \Rightarrow_A \#cdq_5ab \Rightarrow_A \#cq_4ab$.
 - So, $\#cddq_6bab \Rightarrow^*_A \#cddq_2ab$ and $\#cddq_6bab \Rightarrow^*_A \#cq_4ab$.

PDA – reduction

- The **accepted language with accepting status** (end status) by a PDA A is:

$$L(A) = \{w \in T^* \mid z_0q_0w \Rightarrow^*_A xp, \text{ where } x \in Z^*, p \in F\}.$$

Deterministic PDA

- The PDA $A = (Z, Q, T, \delta, z_0, q_0, F)$ is **deterministic** if for all $(z, q, a) \in Z \times Q \times T$ it holds that $|\delta(z, q, a)| + |\delta(z, q, \varepsilon)| = 1$.
- So, for all $q \in Q$ and $z \in Z$
 - either $\delta(z, q, a)$ contains exactly one element for each input symbol $a \in T$ and $\delta(z, q, \varepsilon) = \emptyset$,
 - or $\delta(z, q, \varepsilon)$ contains exactly one element and $\delta(z, q, a) = \emptyset$ for all input symbols $a \in T$.
- Remark: If for all $(z, q, a) \in Z \times Q \times T$ it holds that $|\delta(z, q, a)| + |\delta(z, q, \varepsilon)| \leq 1$ then the PDA can be easily extended to a deterministic one accepting the same language. Thus, PDAs fulfilling this condition can be considered as deterministic in a broader sense.

Deterministic PDA

- The **acceptance (recognition) power** of deterministic PDAs is less than of non-deterministic PDAs.
- Example: Let
 - $L_1 = \{wcw^{-1} \mid w \in \{a, b\}^*\}$,
 - $L_2 = \{ww^{-1} \mid w \in \{a, b\}^*\}$.
 - L_1 can be accepted by a deterministic PDA, but L_2 not.
 - Both L_1 and L_2 can be accepted by a non-deterministic PDA.

Deterministic PDA

- Example: Accepting $L_2 = \{ww^{-1} \mid w \in \{a, b\}^*\}$ non-deterministically.
 - Idea:
 - 1. read and push input symbols non-deterministically either repeat 1. or go to 2.
 - 2. read input symbols and pop stack symbols, compare if not equal reject.
 - 3. enter accept state if stack is empty.
 - Non-deterministic PDA:
 $A = (\{q_0, q_1, q_2\}, \{a, b\}, \{\#, a, b\}, \delta, q_0, \#, \{q_2\})$, where:
 - $(zt, q_0) \in \delta(z, q_0, t), \forall t \in \{a, b\}, z \in \{\#, a, b\}$
 - $(z, q_1) \in \delta(z, q_0, \varepsilon), \forall z \in \{\#, a, b\}$
 - $(\varepsilon, q_1) \in \delta(t, q_1, t), \forall t \in \{a, b\}$
 - $(\#, q_2) \in \delta(\#, q_1, \varepsilon)$

PDA

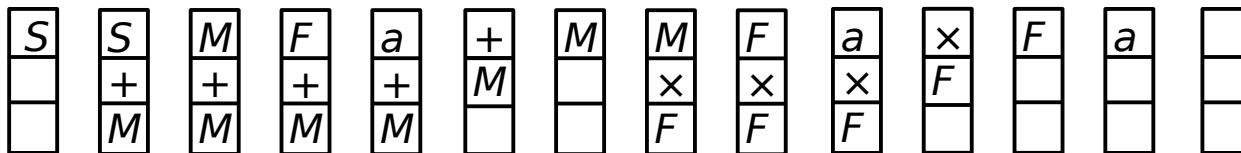
- The language **accepted** by the PDA A **with an empty stack** is
 - $N(A) = \{w \in T^* \mid z_0q_0w \Rightarrow^*_A p, \text{ where } p \in Q\}$.
- Example: Let $A = (\{\$, a\}, \{q_0, q_1\}, \{a, b\}, \delta, \$, q_0, \{\})$, where δ is:
 - $\$q_0a \rightarrow \aq_0
 - $aq_0a \rightarrow aaq_0$
 - $aq_0b \rightarrow q_1$
 - $aq_1b \rightarrow q_1$
 - $\$q_1 \rightarrow q_1$.Then $N(A) = \{a^n b^n \mid n \geq 1\}$.

Computing power of PDAs

- **Theorem:** For every PDA A , a PDA A' can be constructed, s.t. $N(A') = L(A)$ is fulfilled.
- **Theorem:** For every context-free grammar G , a PDA A can be constructed, s.t. $L(A) = L(G)$.
- **Theorem:** For every PDA A , a context-free grammar G can be given, s.t. $L(G) = N(A)$
- Therefore, the computing power of PDAs (either we consider acceptance with accepting end state or acceptance with an empty stack) equal to the computing power of context-free (type 2) grammars.

Converting CFGs to PDAs

- **Theorem:** For every context-free grammar (CFG) G , a PDA A can be constructed, s.t. $L(A) = L(G)$.
- **Proof construction:** Convert the CFG G to the following PDA.
 - Push the start symbol on the stack.
 - If the top of stack is
 - Non-terminal: replace with right hand side of rule (non-deterministic choice).
 - Terminal: pop it and match with next input symbol.
 - If the stack is empty, accept.
- Example: Let $G=(N,T,P,S)$ be the CFG with $T = \{a, +, \times, (,)\}$, $N = \{S, M, F\}$, and $P = \{S \rightarrow S+M \mid M, M \rightarrow M \times F \mid F, F \rightarrow (S) \mid a\}$.
Input: $a+a \times a$.

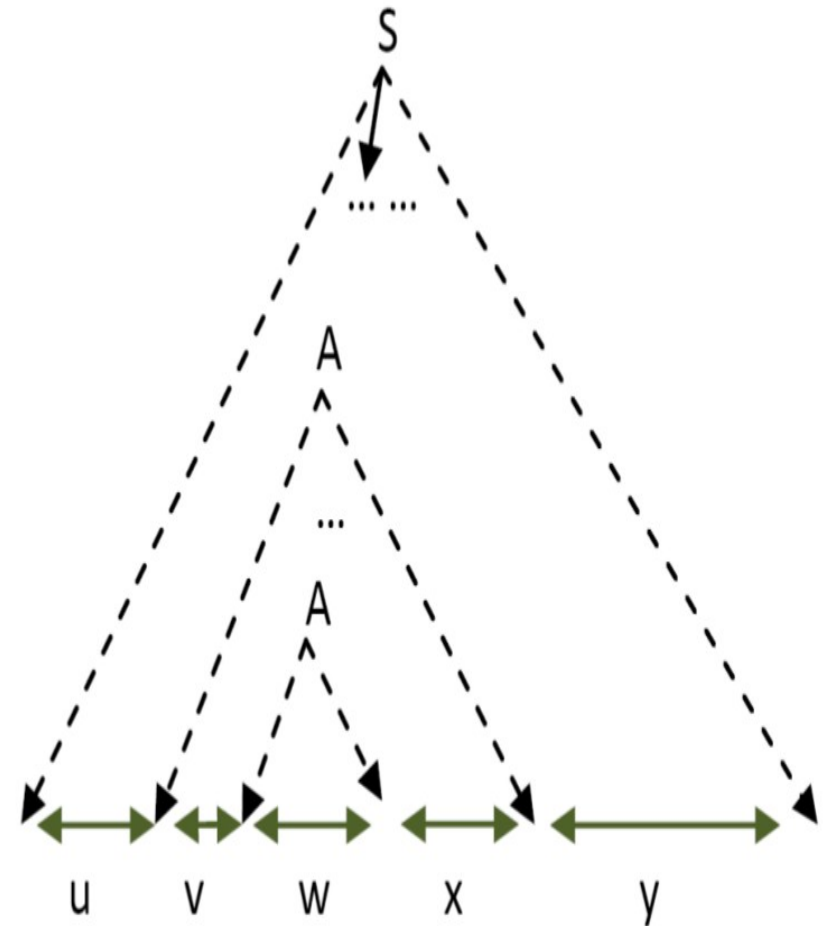


Bar-Hillel Lemma

- A necessary condition that a language is context-free (thus, it can be recognized by a PDA).
- **Theorem** (Bar-Hillel lemma, or pumping lemma for context-free languages):
For all context-free languages L , there exists a natural number n , s.t. that for every word $z \in L$ with $|z| > n$ holds that z can be written as $z = uvwxy$ ($u, v, w, x, y \in T^*$), where $|vwx| \leq n$, $vx \neq \varepsilon$, and $uv^iwx^iy \in L$, for all $i \geq 0$.

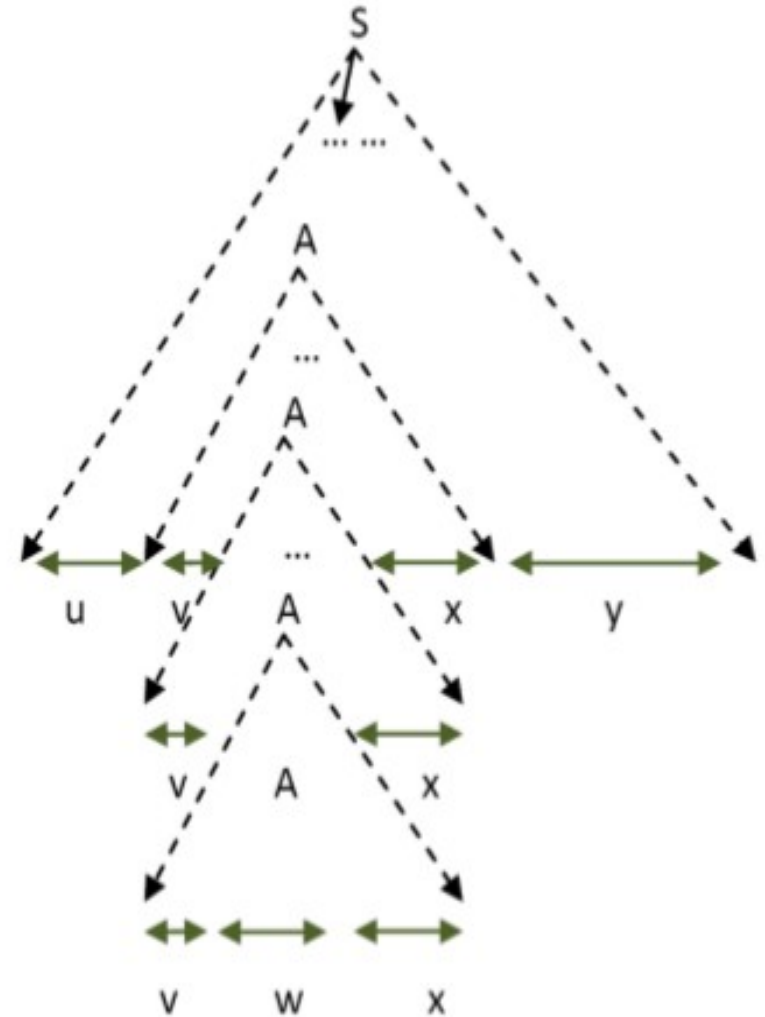
Bar-Hillel Lemma

- **Proof:** Assume, that the grammar is ε -free and it is given in Chomsky normal form (i.e. all production rules are of the form: $A \rightarrow BC$, or $A \rightarrow a$, or $S \rightarrow \varepsilon$). The derivation of a word $z \in L(G)$ can be represented by a tree T_S . If the depth of T_S (length of the longest path from S to a leaf) is k , then $|z| \leq 2^k$, due to the Chomsky normal form. Let N be the set of non-terminals in G and $j = |N|$. Let $n = 2^{j+1}$. If $z \in L$ and $|z| > n$, then the longest path in the derivation tree of $S \Rightarrow^* z$ must be longer than j . Consider the last section of this path of length $j+1$. There must be a non-terminal $A \in N$ that occurs at least twice in this section.



Bar-Hillel Lemma

- Proof** (cont.): Consider two such occurrences of A on this path. Let r be the word corresponding to the subtree of the first one (closer to S), and let w be the word corresponding to the other one. Then, $A \Rightarrow^* r$ and $A \Rightarrow^* w$, and w is a subword of r , so $r=vwx$ for some $v,x \in T^*$. Furthermore, $z=ury$, for some $u,y \in T^*$. Due to the choice of the occurrences of A , $|r| \leq 2^{j+1}$. On the other hand, $S \Rightarrow^* uAy$ and $A \Rightarrow^* vAx$. Therefore, $S \Rightarrow^* uv^iwx^i y$, for any $i \geq 0$. Here, $A \Rightarrow^* vAx$ contains at least one step, and the first step must be the application of a rule of the form $A \rightarrow BC$. Therefore $|vx| \geq 1$, since G is ϵ -free. ■



Application of the Bar-Hillel Lemma

- **Claim:** The language $L = \{a^j b^j c^j : j \geq 1\}$ is not context-free.
- **Proof:** Assume for contradiction, that G is a context-free grammar generating L .
Then, by the lemma, $\exists n \geq 0$ that for \forall word $z \in L$, $|z| > n$ can be written in the form $z = uvwxy$, s.t. $|vwx| \leq n$, $vx \neq \varepsilon$, and for all $i \geq 0$, $uv^i wx^i y \in L$.
Consider a word $a^m b^m c^m$ with $m > n$. Since $|vwx| \leq n$, vwx can not contain all three symbols of a, b, c .
Assume, w.l.o.g., it contains at least one a and does not contain any c . Then by pumping, for $i \geq 2$, $uv^i wx^i y$ contains more a 's than c 's. Consequently, $uv^i wx^i y \notin L$. ■

References

- Michael O. Rabin: Probabilistic Automata. *Information and Control*. 6 (3): 230–245, 1963. doi:10.1016/s0019-9958(63)90290-0
- Michael Sipser: *Introduction to the Theory of Computation*. 3rd edition, 2012.