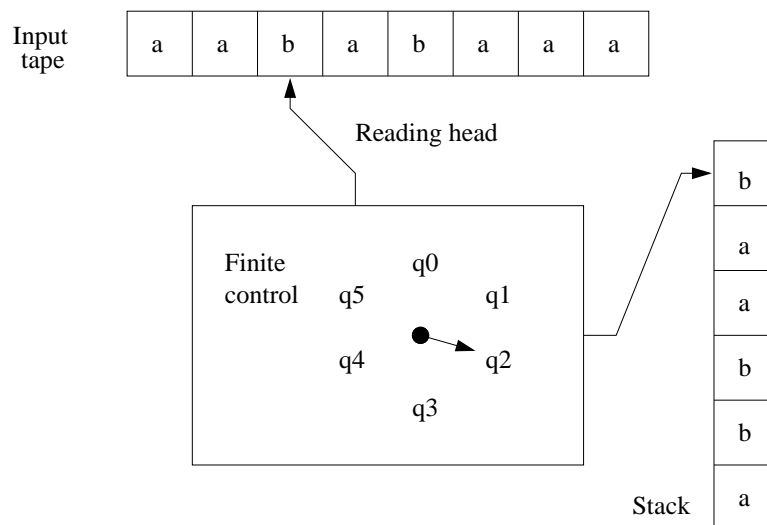


Pushdown Automata

- We have seen examples of context-free languages that are not regular, and hence can not be recognized by finite automata.
- Next we consider a more powerful computation model, called a “pushdown automaton,” by adding the storage device of a “stack.”
- The stack allows read and write access, but only to the top, or last, symbol. It is of finite but unbounded size.



Pushdown Automata (cont.)

- Each computation step by a pushdown automaton depends on (i) the current state, (ii) the next input symbol, and (iii) the top symbol(s) on the stack.
- Let Σ_ϵ denote the set $\Sigma \cup \{\epsilon\}$.

- **Definition**

A *pushdown automaton* is a tuple

$$M = (Q, \Sigma, \Gamma, q_0, T, \Delta),$$

where

Q is finite set of *states*,
 Σ is an alphabet (the *input symbols*),
 Γ is an alphabet (the *stack symbols*),
 $q_0 \in Q$ is the *start state*,
 $T \subseteq Q$ is the set of *final* or *accept states*,
and
 Δ is a finite subset of $(Q \times \Sigma_\epsilon \times \Gamma^*) \times (Q \times \Gamma^*)$,
called the *transition relation*.

- Pushdown automata are *nondeterministic*.
- The transition relation can also be defined as a subset of $(Q \times \Sigma_\epsilon \times \Gamma_\epsilon) \times (Q \times \Gamma_\epsilon)$. This slightly more restrictive definition does not limit the expressive power of the computation model.

Transitions

- A transition $((p, a, \beta), (q, \gamma)) \in \Delta$ indicates that the pushdown automaton may change from state p to state q , provided the next input symbol is a and β is on top of the stack. In addition, β is replaced by γ at the top of the stack.
- A transition $((p, u, \epsilon), (q, a))$ *pushes* a single symbol a onto the stack.
- A transition $((p, u, a), (q, \epsilon))$ *pops* a symbol a off the stack.
- More general transitions can always be represented by a sequence of pop and push operations.
- A *configuration* for a pushdown automaton is a triple of type $Q \times \Sigma^* \times \Gamma^*$, where the first component represent the current state, the second component the unread portion of the input tape, and the third component the contents of the stack (read top-down).
- A configuration (p, x, α) *yields* another configuration (q, y, β) , written

$$(p, x, \alpha) \vdash (q, y, \beta),$$

if there is a transition $((p, a, \gamma), (q, \eta))$ such that $x = ay$, γ is a prefix of α , and β is obtained from α by replacing the prefix β by η .

Computations

- A *computation* by a pushdown automaton is a sequence of configurations

$$C = C_0 \vdash C_1 \vdash \dots \vdash C_n = C'.$$

- We also say that the configuration C *yields* C' , and write $C \vdash^* C'$.
- A string $w \in \Sigma^*$ is said to be *accepted* by a pushdown automaton M if, and only if, there is a computation from a start configuration (s, w, ϵ) to a configuration (p, ϵ, ϵ) , where p is an accept state of M .
- The *language accepted* by M is the set $L(M)$ of all strings accepted by M .

Example

- Let M be a pushdown automaton $(Q, \Sigma, \Gamma, q_0, T, \Delta)$, where

$$\begin{aligned}Q &= \{s, f\}, \\ \Sigma &= \{a, b, c\}, \\ \Gamma &= \{a, b\}, \\ T &= \{f\},\end{aligned}$$

and Δ contains the following transitions:

$$\begin{aligned}&((s, a, \epsilon), (s, a)), \\ &((s, b, \epsilon), (s, b)), \\ &((s, c, \epsilon), (f, \epsilon)), \\ &((f, a, a), (f, \epsilon)), \text{ and} \\ &((f, b, b), (f, \epsilon)).\end{aligned}$$

- This pushdown automaton accepts the language

$$L = \{w c w^R : w \in \{a, b\}^*\}.$$

Another Example

- The following pushdown automaton accepts the language

$\{w \in \{a, b\}^* : w \text{ has the same number of } a\text{'s and } b\text{'s}\}.$

- We have $M = ((Q, \Sigma, \Gamma, q_0, T, \Delta)$, where

$$K = \{s, q, f\},$$

$$\Sigma = \{a, b\},$$

$$\Gamma = \{a, b, c\},$$

$$T = \{f\},$$

and Δ contains the following transitions:

$$((s, \epsilon, \epsilon), (q, c)),$$

$$((q, a, c), (q, ac)),$$

$$((q, a, a), (q, aa)),$$

$$((q, a, b), (q, \epsilon)),$$

$$((q, b, c), (q, bc)),$$

$$((q, b, b), (q, bb)),$$

$$((q, b, a), (q, \epsilon)), \text{ and}$$

$$((q, \epsilon, c), (f, \epsilon)).$$

Context-Free Languages

- **Theorem**

A language is accepted by a pushdown automaton if, and only if, it is context-free.

- *Proof sketch.*

- We first show that each context-free language is accepted by some pushdown automaton.
- Let $G = (V, \Sigma, R, S)$ be a context-free grammar. We define a corresponding pushdown automaton

$$M = (\{p, q\}, \Sigma, V \cup \Sigma, p, \{q\}, \Delta),$$

where Δ contains the following transitions:

$$\begin{aligned} &((p, \epsilon, \epsilon), (q, S)), \\ &((q, \epsilon, A), (q, x)), \text{ for each rule } A \rightarrow x \text{ in } R, \\ &\text{and} \\ &((q, a, a), (q, \epsilon)), \text{ for each symbol } a \in \Sigma. \end{aligned}$$

- This automaton mimics a leftmost derivation of a string in $L(G)$, and $L(M) = L(G)$.

- Next we show that each language accepted by a pushdown automaton is context-free .
- Let $M = (Q, \Sigma, \Gamma, q_0, T, \Delta)$ be a pushdown automaton. We assume that T contains only one accept state, say f , and that each transition either pushes a single symbol onto the stack or pops one off the stack.
- We construct a context-free grammar G for M . The variables of G are written as A_{pq} , where p and q are states of M .
- The rules of G are designed so that from A_{pq} one can derive exactly those strings that drive the automaton M from state p with empty stack to state q with empty stack:
 - $A_{pq} \rightarrow aA_{rs}b$, where Δ contains transitions $((p, a, \epsilon), (r, c))$ and $((s, b, c), (q, \epsilon))$, for some $c \in \Gamma$;
 - $A_{pq} \rightarrow A_{pr}A_{rq}$, where $p, q, r \in K$; and
 - $A_{pp} \rightarrow \epsilon$, where $p \in K$.
- The start variable of G is $S = A_{sf}$, and it can be shown that $L(G) = L(M)$.

Closure Properties

- **Theorem**

The intersection of a context-free language and a regular language is context-free.

- *Proof sketch.*

- Let L be a context-free language accepted by a PDA

$$M_1 = (Q_1, \Sigma, \Gamma_1, s_1, F_1, \Delta_1,)$$

and R be a regular language accepted by a DFA

$$M_2 = (Q_2, \Sigma, s_2, F_2, \delta).$$

- We design a PDA $M = (Q, \Sigma, \Gamma, s, F, \Delta)$ that simulates M_1 and M_2 in parallel:

$$Q = Q_1 \times Q_2,$$

$$\Gamma = \Gamma_1,$$

$$s = (s_1, s_2),$$

$$F = F_1 \times F_2, \text{ and}$$

$$\Delta = \{(((q_1, q_2), a, \beta), ((p_1, \delta(q_2, a)), \gamma)) : \\ ((q_1, a, \beta), (p_1, \gamma)) \in \Delta_1, q_2 \in K_2\} \\ \cup \{(((q_1, q_2), \epsilon, \beta), ((p_1, q_2), \gamma)) : \\ ((q_1, \epsilon, \beta), (p_1, \gamma)) \in \Delta_1, q_2 \in K_2\}.$$

- It can be shown that $L(M) = L(M_1) \cap L(M_2)$.

Pumping Lemma

- Infinite context-free languages display more subtle periodicity patterns than regular languages.

- **Theorem**

If L is a context-free language then there is an integer $n_L \geq 1$ (the pumping length) such that any string $w \in L$ with $|w| \geq n_L$ can be written as $w = uvxyz$, where

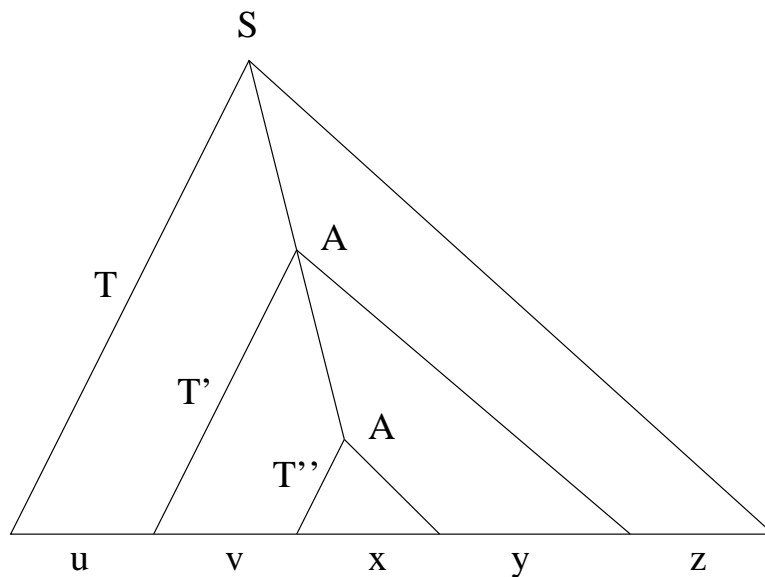
- (i) $|vy| > 0$,
- (ii) $|vxy| \leq n_L$, and
- (iii) for each $i \geq 0$, $uv^i xy^i z \in L$.

- *Proof sketch.*

- Let L be a context-free language and G be a context-free grammar (V, Σ, S, R) such that $L(G) = L$.
- Let l be the maximum number of symbols on the right-hand side of a rule in R . We may assume that $l \geq 2$. No node in a parse tree for G can have more than l children. Thus, if a parse tree has height h , its yield is a string of length no more than l^h . We choose n_L to be the number $l^{|V-\Sigma|+1}$.
- Let w be a string in L with $|w| \geq n_L$. We have to show that w can be divided into five parts,

$w = uvxyz$, such that conditions (i), (ii), and (iii) are satisfied.

- Let T be a parse tree for w with the smallest number of nodes.
- Since T must be of height at least $|V - \Sigma| + 1$, we may infer that any longest (root-to-leaf) path in T contains at least $|V - \Sigma| + 1$ nodes labeled by nonterminals. Some nonterminal must occur more than once on this path. Let A be a non-terminal that occurs repeatedly among the last $|V - \Sigma| + 1$ nodes on a selected longest path.
- We have the following situation:



- That is, from S we can derive the string uAz , whereas from A we can derive both x and vAy .

Consequently, from S we can obtain all strings of the form $uv^i xy^i z$ in Σ^* .

- The two strings v and y can not both be empty, as T was assumed to be a smallest parse tree with yield w .
- Moreover, we picked repeated occurrences of A within the bottom $|V - \Sigma| + 1$ nodes, and hence $|vxy| \leq n_L$.

Non-Context-Free Languages

- The Pumping Lemma can be used to show that certain languages, including
 - the set $\{a^n b^n c^n : n \geq 0\}$ and
 - the set $\{ww : w \in \{a, b\}^*\}$,are not context-free.
- This also implies that context-free languages do not satisfy certain closure properties.

- **Theorem**

The class of context-free languages is not closed under intersection or complement.

Proof.

- The languages $\{a^n b^n c^m : m, n \geq 0\}$ and $\{a^m b^n c^n : m, n \geq 0\}$ are context-free, but their intersection, the set $\{a^n b^n c^n : n \geq 0\}$, is not.
- Thus, context-free languages are not closed under intersection.
- Since intersection of sets can be expressed in terms of complement and union, we may infer that context-free languages are not closed under complement either.