Context-free languages

Stefan D. Bruda

CS 310, Winter 2025

Context-Free Grammars

- Recall that languages defined using only union, concatenation, and recursion from symbols from Σ, ε, and $\emptyset$ are called context-free languages
  - A language description as above is called a context-free grammar
    - Examples: $B = \varepsilon + 0B1$, $L = \varepsilon + 1 + 0 + 0L0 + 1L1$
  - The customary way of writing down a context-free grammar is using a definition by cases rather than the recursive equation (the Backus-Naur form or BNF)

$$
\begin{array}{lcl}
B & \to & \varepsilon \\
B & \to & 0B1
\end{array}
\qquad
\begin{array}{lcl}
\langle \text{balanced} \rangle & ::= & \\
\langle \text{balanced} \rangle & ::= & 0\ \langle \text{balanced} \rangle\ 1
\end{array}
$$

$$
\begin{array}{lcl}
L & \to & \varepsilon \\
L & \to & 0 \\
L & \to & 1 \\
L & \to & 0L0 \\
L & \to & 1L1
\end{array}
\qquad
\begin{array}{lcl}
\langle \text{palindrome} \rangle & ::= & \\
\langle \text{palindrome} \rangle & ::= & 0 \\
\langle \text{palindrome} \rangle & ::= & 1 \\
\langle \text{palindrome} \rangle & ::= & 0\ \langle \text{palindrome} \rangle\ 0 \\
\langle \text{palindrome} \rangle & ::= & 1\ \langle \text{palindrome} \rangle\ 1
\end{array}
$$

$$\langle \text{palindrome} \rangle ::= \varepsilon \mid 0 \mid 1 \mid 0\ \langle \text{palindrome} \rangle\ 0 \mid 1\ \langle \text{palindrome} \rangle\ 1$$

Context-Free Grammars (cont'd)

- Formally a context-free grammar is a tuple $G = (N, \Sigma, R, S)$, where
  - Σ is an alphabet of terminals
  - N alphabet of symbols called by contrast nonterminals
    - Traditionally nonterminals are capitalized or surrounded by $\langle$ and $\rangle$, everything else being a terminal
  - $S \in N$ is the axiom (or the start symbol)
  - $R \subseteq N \times (N + \Sigma)^*$ is the set of (rewriting) rules or productions
    - Common ways of expressing $(\alpha, \beta) \in R$: $\alpha \to \beta$ or $\alpha \ ::= \ \beta$
- Further examples:

$$
\begin{array}{lcl}
\langle \text{exp} \rangle & ::= & \langle \text{const} \rangle \\
& | & \langle \text{var} \rangle \\
& | & \langle \text{exp} \rangle \ \langle \text{op} \rangle \ \langle \text{exp} \rangle \\
& | & (\ \langle \text{exp} \rangle \ ) \\
\langle \text{op} \rangle & ::= & + \mid - \mid * \mid /
\end{array}
$$

$$
\begin{array}{lcl}
\langle \text{stmt} \rangle & ::= & ; \\
& | & \langle \text{var} \rangle = \langle \text{exp} \rangle \ ; \\
& | & \texttt{if} \ (\ \langle \text{exp} \rangle \ ) \ \langle \text{stmt} \rangle \ \texttt{else} \ \langle \text{stmt} \rangle \\
& | & \texttt{while} \ (\ \langle \text{exp} \rangle \ ) \ \langle \text{stmt} \rangle \\
& | & \{ \ \langle \text{seq} \rangle \ \} \\
\langle \text{seq} \rangle & ::= & \varepsilon \mid \langle \text{stmt} \rangle \ \langle \text{seq} \rangle
\end{array}
$$

Derivations

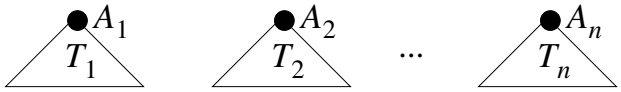
- $G = (N, \Sigma, R, S)$
- A rewriting rule $A \to v' \in R$ is used to rewrite its left-hand side (A) into its right-hand side (v'):
  - $u \Rightarrow v$ iff $\exists x, y \in (N + \Sigma)^* : \exists A \in N : u = xAy, v = xv'y, A \to v' \in R$
- Rewriting can be chained ($\Rightarrow^*$, the reflexive and transitive closure of $\Rightarrow$ = derivation)
  - $s \Rightarrow^* s'$ iff $s = s'$, $s \Rightarrow s'$, or there exist strings $s_1, s_2, \ldots, s_n$ such that $s \Rightarrow s_1 \Rightarrow s_2 \Rightarrow \cdots \Rightarrow s_n \Rightarrow s'$
  - $\langle \text{pal} \rangle \Rightarrow 0\langle \text{pal} \rangle 0 \Rightarrow 01\langle \text{pal} \rangle 10 \Rightarrow 010\langle \text{pal} \rangle 010 \Rightarrow 0101010$

$$\langle \text{pal} \rangle \ ::= \ \varepsilon \mid 0 \mid 1 \mid 0 \ \langle \text{pal} \rangle \ 0 \mid 1 \ \langle \text{pal} \rangle \ 1$$

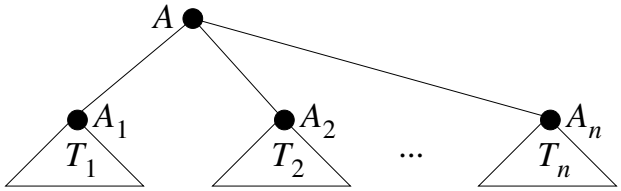
- The language generated by grammar G: exactly all the terminal strings generated from S: $\mathcal{L}(G) = \{w \in \Sigma^* : S \Rightarrow^* w\}$
  - Same language as defined by the respective recursive equation (and the approximation scheme shown earlier)

Parse Trees

- Definition:
  1. For every $a \in N + \Sigma$ the following is a parse tree (with yield a): 
  2. For every $A \to \varepsilon \in R$ the following is a parse tree (with yield ε): 
  3. If the following are parse trees (with yields y_1, y_2, ..., y_n, respectively):

     and $A \to A_1 A_2 \ldots A_n \in R$, then the following is a parse tree (with yield $y_1 y_2 \ldots y_n$):

     
- Yield: concatenation of leaves in inorder

Derivations and Parse Trees

- Every derivation starting from some nonterminal has an associated parse tree (rooted at the starting nonterminal)
- Two derivations are similar iff only the order of rule application varies = can obtain one derivation from the other by repeatedly flipping consecutive rule applications
  - Two similar derivations have identical parse trees
  - Can use a "standard" derivation: leftmost ($A \overset{L}{\Rightarrow}^* w$) or rightmost ($A \overset{R}{\Rightarrow}^* w$)

Theorem

The following statements are equivalent:

- *there exists a parse tree with root A and yield w*
- $A \Rightarrow^* w$
- $A \overset{L}{\Rightarrow}^* w$
- $A \overset{R}{\Rightarrow}^* w$

- Ambiguity of a grammar: there exists a string that has two derivations that are not similar (i.e., two derivations with diferent parse trees)
  - Can be inherent or not — impossible to determine algorithmically

Context-Free and Regular Languages

- Regular grammar: $G = (N, \Sigma, R, S)$ with N, Σ, S as before, and $R \subseteq N \times (\varepsilon + \Sigma + \Sigma N)$
  - Special form of context-free grammar (only rules of form $A \to \varepsilon$, $A \to a$, and $A \to aB$ allowed)

Theorem

Exactly all the regular languages are generated by regular grammars

- Let $M = (K, \Sigma, \Delta, s, F)$ be some finite automaton
- We construct the grammar $G = (K, \Sigma, s, R)$ with

$$R = \{q \to ap : (q, a, p) \in \Delta\} + \{q \to \varepsilon : q \in F\}$$

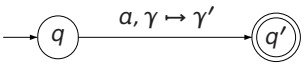
Corollary

All regular languages are context-free

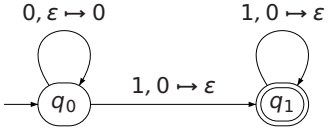
- However, there are more context-free than regular languages

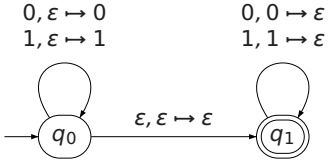
$$S \to aSb \quad S \to \varepsilon$$

Push-Down Automata

- Push-down automaton: finite automaton + "push-down store" (or stack)
- $M = (K, \Sigma, \Gamma, \Delta, s, F)$
  - K, Σ, s, F as before (for finite automata)
  - Γ is the stack alphabet
  - $\Delta \subseteq \{(K \times (\Sigma + \{\varepsilon\}) \times \Gamma^*) \times (K \times \Gamma^*)\}$
  - Transition: $((q, a, \gamma), (q', \gamma'))$ with a the current input symbol (or ε), γ the old stack top, and γ' the replacement top
    - Graphical representation: 

- Acceptance: there exists a path (or run) that spells the input and ends in a final state and the stack is empty at the beginning as well as at the end of the path



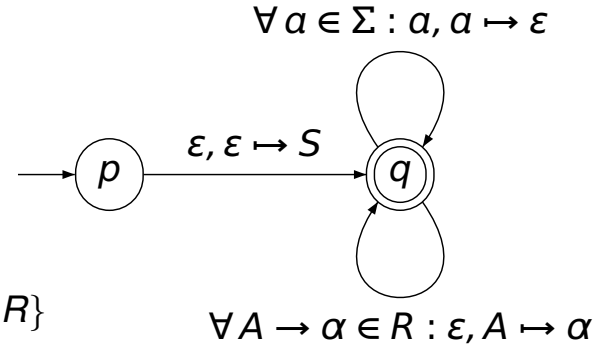


Theorem

Theorem: Push-down automata accept exactly all the context-free languages

- $\supseteq$: Given the grammar $G = (N, \Sigma, S, R)$ construct the push-down automaton $M = (K, \Sigma, \Gamma, \Delta, s, F)$ such that:

$$
\begin{aligned}
\Gamma &= N + \Sigma \\
K &= \{p, q\} \\
s &= p \\
F &= \{q\} \\
\Delta &= \{((p, \varepsilon, \varepsilon), (q, S))\} \\
&+ \{((q, \varepsilon, A), (q, \alpha)) : A \to \alpha \in R\} \\
&+ \{((q, a, a), (q, \varepsilon)) : a \in \Sigma\}
\end{aligned}
$$



- $\subseteq$: Proof left for a full course in formal languages

CLOSURE PROPERTIES

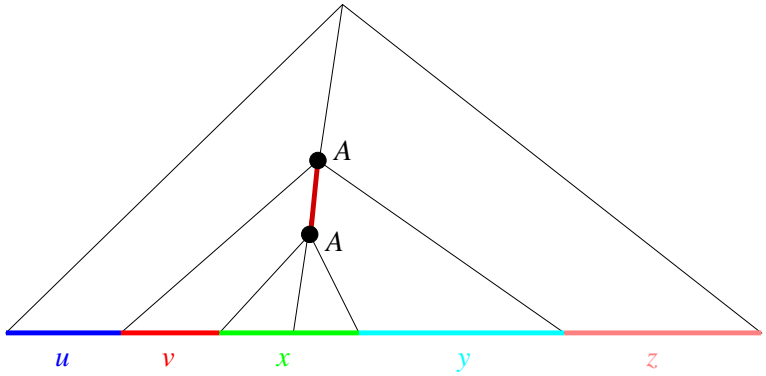
- Consider two grammars with axioms S_1 and S_2; construct a grammar with axiom S
- Context-free languages are closed under
  - Union: Add rules $S \to S_1$ and $S \to S_2$
  - Concatenation: Add rule $S \to S_1 S_2$
  - Kleene star: Add rules $S \to \varepsilon$ and $S \to SS_1$
  - Intersection and complement:
    - If context-free languages are not closed under one of these then they are not closed under the other either

Pumping Context-Free Languages

- Let $\Phi(G)$ be the maximum fanout (branching factor) of any node in any parse tree constructed based on grammar G
- A parse tree of height h has a yield of size no more than $\Phi(G)^h$

Theorem (Pumping context-free languages)

Let $n = \Phi(G)^{|N|}$. For any $w \in \mathcal{L}(G)$ such that $|w| \geq n$ we can write w as $uvxyz$ such that $vy \neq \varepsilon$, $|vxy| \leq n$ and $uv^i xy^i z \in \mathcal{L}(G)$ for any $i \geq 0$



Pumping Context-Free Languages (cont'd)

- Some interesting non-context-free languages:
  - $\{a^n b^n c^n : n \geq 0\}$
  - $\{a^n : n \text{ is prime}\}$
  - $\{w \in \{a, b, c\}^* : |w|_a = |w|_b = |w|_c\}$

Corollary

Context-free languages are not closed under intersection and complement

  - Indeed, $\{a^n b^n c^n : n \geq 0\} = \{a^n b^n c^m : n, m \geq 0\} \cap \{a^m b^n c^n : n, m \geq 0\}$
  - That $\{a^n b^n c^m : n, m \geq 0\}$ is context free can be shown by constructing a grammar/automaton or by using closure properties
- Tricky language: $\{w \in \{a, b, c\}^* : |w|_a = |w|_b = |w|_c\}$ satisfies the pumping lemma yet is not context-free