



## GRAMMARS

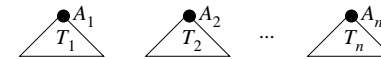
- A **grammar** is a tuple  $G = (V, \Sigma, R, S)$ , where
  - $V$  is an alphabet;  $\Sigma \subseteq V$  is the alphabet of **terminals**
    - \*  $V \setminus \Sigma$  called by contrast **nonterminals**
  - $S \in V \setminus \Sigma$  is the **axiom** (or the **start symbol**)
  - $R \subseteq V^*(V \setminus \Sigma)V^* \times V^*$  is the set of (**rewriting**) **rules** or **productions**
    - \* Common ways of expressing  $(\alpha, \beta) \in R$  for a grammar  $G$ :  
 $\alpha \rightarrow_G \beta$  (or just  $\alpha \rightarrow \beta$ ),  $\alpha \rightarrow \beta \in R$
- **Context-free grammar**: a grammar with  $R \subseteq (V \setminus \Sigma) \times V^*$ 
  - (**left**) **regular grammar**:  $R \subseteq (V \setminus \Sigma) \times (\Sigma V \cup \{\varepsilon\})$
- $u \Rightarrow_G v$  iff  $\exists x, y \in V^* : \exists A \in V \setminus \Sigma : u = xAy, v = xv'y, A \rightarrow_G v'$
- $\Rightarrow_G^*$  is the reflexive and transitive closure of  $\Rightarrow_G$  (**derivation**)
- The language generated by grammar  $G$ :  $\mathcal{L}(G) = \{w \in \Sigma^* : S \Rightarrow_G^* w\}$

## DERIVATIONS AND PARSE TREES

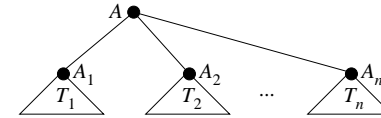
- Every derivation starting from some nonterminal has an associated parse tree (rooted at the starting nonterminal)
- Two derivations are **similar** iff only the order of rule application varies = we can obtain one derivation from the other by repeatedly flipping **consecutive** rule applications
  - Two similar derivations have identical parse trees
  - We could choose some “standard” derivations: leftmost ( $A \xrightarrow{L} w$ ) and rightmost ( $A \xrightarrow{R} w$ )
  - **Theorem**: The following statements are equivalent:
    - \* there exists a parse tree with root  $A$  and yield  $w$
    - \*  $A \Rightarrow w$
    - \*  $A \xrightarrow{L} w$
    - \*  $A \xrightarrow{R} w$
- **Ambiguity** of a grammar: there exists a string that has two derivations that are not similar (i.e., two derivations with different parse trees)
  - Can be **inherent** or not

## PARSE TREES

- Tree with labelled nodes
- Yield: concatenation of leaves in inorder
- Definition:
  1. For every  $a \in \Sigma$  the following is a parse tree (with yield  $a$ ): 
  2. For every  $A \rightarrow \varepsilon$  the following is a parse tree (with yield  $\varepsilon$ ): 
  3. If the following are parse trees (with yields  $y_1, y_2, \dots, y_n$ , respectively):



and  $A \rightarrow A_1 A_2 \dots A_n$ , then the following is a parse tree (with yield  $y_1 y_2 \dots y_n$ ):



## CONTEXT-FREE AND REGULAR LANGUAGES

- Languages generated by context-free grammars are called **context-free**
- **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  $G = (K \cup \Sigma, \Sigma, s, R)$  with
- **Corollary**: All regular languages are context-free
- However, there are more context-free than regular languages

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

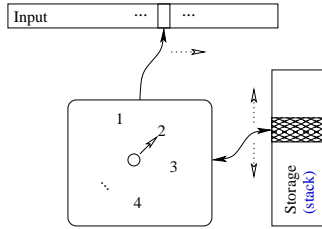
$$S \rightarrow aSb \quad S \rightarrow \varepsilon$$

## PUSHDOWN AUTOMATA

- **Pushdown automaton:**  $M = (K, \Sigma, \Gamma, \Delta, s, F)$ 
  - $K, \Sigma, s, F$  as before (for finite automata)
  - $\Gamma$  is the stack alphabet
  - $\Delta \subseteq \{(K \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^*) \times (K \times \Gamma^*)\}$
  - Transition:

$$((q, a, \gamma), (q', \gamma'))$$

with  $a$  the current input symbol (or  $\varepsilon$ ),  $\gamma$  the old stack head, and  $\gamma'$  the replacement head



- Configuration: a member of  $K \times \Sigma^* \times \Gamma^*$
- $(q, w, u) \vdash_M (q', w', u')$  iff  $\exists ((q, a, \gamma), (q', \gamma')) \in \Delta : w = aw', u = \gamma x, u' = \gamma'x$  for some  $x \in \Gamma^*$
- $M$  accepts  $w$  iff  $(s, w, \varepsilon) \vdash_M^* (f, \varepsilon, \varepsilon)$  for some  $f \in F$
- The language accepted by  $M$  is

$$\mathcal{L}(M) = \{w \in \Sigma^* : (s, w, \varepsilon) \vdash_M^* (f, \varepsilon, \varepsilon) \text{ for some } f \in F\}$$

## PUSHDOWN AUTOMATA AND CONTEXT-FREE LANGUAGES

- **Theorem:** Pushdown automata accept exactly all the context-free languages
- Construct a finite automaton  $M = (K, \Sigma, \Gamma, \Delta, s, F)$  out of a grammar  $G = (V, \Sigma, S, R)$  and the other way around
- $\supseteq$ 
  - $\Gamma = V, K = \{p, q\}, s = p, F = \{q\}$
  - $\Delta = \{((p, \varepsilon, \varepsilon), (q, S))\} \cup \{((q, \varepsilon, A), (q, \alpha)) : A \rightarrow \alpha \in R\} \cup \{((q, a, a), (q, \varepsilon)) : a \in \Sigma\}$
  - Complete proof on page 138
- $\subseteq$ 
  - We work with **simplified automata**:  $((q, a, \gamma), (q', \gamma')) \in \Delta \Rightarrow \gamma \in \Gamma \wedge |\gamma'| \leq 2$  for any  $q \neq s$
  - Given a normal automaton it is easy to construct the simplified automaton  $M' = (K', \Sigma, \Gamma \cup \{Z\}, \Delta', s', \{f'\})$ , with  $K' = K \uplus \{s', f'\}, Z \notin \Gamma$  and  $\Delta'$  contains for a starter the transitions  $((s', \varepsilon, \varepsilon), (s, Z))$  and  $((f, \varepsilon, Z), (f', \varepsilon))$  for any  $f \in F$

## PUSHDOWN AUTOMATA AND CONTEXT-FREE LANGUAGES (CONT'D)

- $\subseteq$  (cont'd)
  - We add then to  $\Delta'$  all the transitions in  $\Delta$  that are already in the desired form
  - For any  $((q, a, \gamma), (q', \gamma')) \in \Delta$  such that  $\gamma = \gamma_1 \gamma_2 \dots \gamma_n$  for some  $n > 1$  we add in  $\Delta'$ :
  - For any  $((q, a, \gamma), (q', \gamma')) \in \Delta \cup \Delta'$  such that  $\gamma' = \gamma_1 \gamma_2 \dots \gamma_m$  for some  $m > 2$  we add/replace in  $\Delta'$ :
  - For any  $((q, a, \varepsilon), (q', \gamma)) \in \Delta \cup \Delta'$  we add/replace in  $\Delta'$ :

## PUSHDOWN AUTOMATA AND CONTEXT-FREE LANGUAGES (CONT'D)

- $\subseteq$  (cont'd)
  - We add then to  $\Delta'$  all the transitions in  $\Delta$  that are already in the desired form
  - For any  $((q, a, \gamma), (q', \gamma')) \in \Delta$  such that  $\gamma = \gamma_1 \gamma_2 \dots \gamma_n$  for some  $n > 1$  we add in  $\Delta'$ :
$$((q, \varepsilon, \gamma_1), (q_1, \varepsilon)) \quad ((q_1, \varepsilon, \gamma_2), (q_2, \varepsilon)) \quad \dots$$

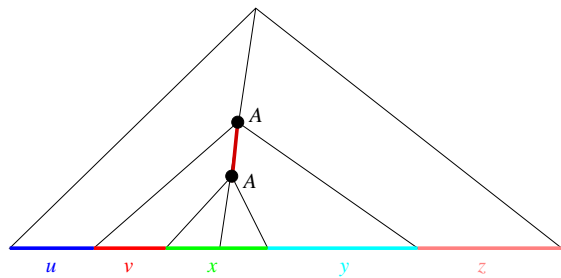
$$((q_{n-2}, \varepsilon, \gamma_{n-1}), (q_{n-1}, \varepsilon)) \quad ((q_{n-1}, a, \gamma_n), (q', \gamma'))$$
  - For any  $((q, a, \gamma), (q', \gamma')) \in \Delta \cup \Delta'$  such that  $\gamma' = \gamma_1 \gamma_2 \dots \gamma_m$  for some  $m > 2$  we add/replace in  $\Delta'$ :
$$((q, a, \gamma), (q_1, \gamma_m)) \quad ((q_1, \varepsilon, \varepsilon), (q_2, \gamma_{m-1})) \quad \dots$$

$$((q_{m-2}, \varepsilon, \varepsilon), (q_{m-1}, \gamma_2)) \quad ((q_{m-1}, \varepsilon, \varepsilon), (q', \gamma_1))$$
  - For any  $((q, a, \varepsilon), (q', \gamma)) \in \Delta \cup \Delta'$  we add/replace in  $\Delta'$ :
$$((q, a, A), (q', \gamma A)) \text{ for all } A \in \Gamma \cup \{Z\}$$

- $\subseteq$  (cont'd)
  - Now we take  $V = \{S\} \cup \{\langle q, A, p \rangle : q, p \in K', A \in \Gamma \cup \{\varepsilon, Z\}\}$
  - Every nonterminal  $\langle q, A, p \rangle$  corresponds to the input consumed by the automaton starting from state  $q$  with  $A$  at the top of the stack and ending in state  $p$
  - Then  $R$  is constructed as follows:
    - \*  $S \rightarrow \langle s, Z, f' \rangle$
    - \* For each  $((q, a, B), (r, C)) \in \Delta, B, C \in \Gamma$  we add  $\langle q, B, p \rangle \rightarrow a \langle r, C, p \rangle$  for each  $p \in K'$
    - \* For each  $((q, a, B), (r, CC')) \in \Delta, B, C, C' \in \Gamma$  we add  $\langle q, B, p' \rangle \rightarrow a \langle r, C, p \rangle \langle p, C', p' \rangle$  for each combination  $p, p' \in K'$
    - \* For each  $q \in K'$  we add  $\langle q, \varepsilon, q \rangle \rightarrow \varepsilon$

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  $\Phi(G)^h$
- **Pumping theorem:** For any  $w \in \mathcal{L}(G)$  such that  $|w| \geq \Phi(G)^{|V-\Sigma|}$  we can write  $w$  as  $uvxyz$  such that  $vy \neq \varepsilon$  and  $uv^nxy^n z \in \mathcal{L}(G)$  for any  $n \geq 0$



- 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  $S \rightarrow S_1$  and  $S \rightarrow S_2$
  - Concatenation: Add rule  $S \rightarrow S_1 S_2$
  - Kleene star: Add rules  $S \rightarrow \varepsilon$  and  $S \rightarrow S S_1$
- Context-free languages are closed under intersection with regular languages
  - $M_1 = (K_1, \Sigma, \Gamma_1, \Delta_1, s_1, F_1) \quad \mathcal{L}(M_1) = L_1$
  - $M_2 = (K_2, \Sigma, \delta_2, s_2, F_2) \quad \mathcal{L}(M_2) = L_2$
  - Construct  $M = (K, \Sigma, \Gamma, \Delta, s, F) \quad \mathcal{L}(M) = L_1 \cap L_2$
  - $K = K_1 \times K_2, \Gamma = \Gamma_1, s = (s_1, s_2), F = F_1 \times F_2$
$$((q_1, a, \gamma), (p, \gamma')) \in \Delta_1 \Rightarrow (((q_1, q_2), a, \gamma), ((p, \delta_2(q_2, a)), \gamma')) \in \Delta$$

$$((q_1, \varepsilon, \gamma), (p, \gamma')) \in \Delta_1 \Rightarrow (((q_1, q_2), \varepsilon, \gamma), ((p, q_2), \gamma')) \in \Delta$$

PUMPING CONTEXT-FREE LANGUAGES (CONT'D)

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

- Some interesting non-context-free languages:

$$\{a^n b^n c^n : n \geq 0\}$$

$$\{w \in \{a, b, c\}^* : |w|_a = |w|_b = |w|_c\}$$

$$\{a^n : n \text{ is prime}\}$$

- **Corollary:** Context-free languages are **not** closed under intersection and complementation
  - 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
- Then  $\{w \in \{a, b, c\}^* : |w|_a = |w|_b = |w|_c\}$  can be shown non-context-free using closure properties