

Specification of Languages

- Let A be the set of all strings of properly balanced left and right parentheses, such as $(())$ or $()()$.
- There are various formal definitions of A . We next explore the use of “substring replacement rules” for specifying formal languages.

- **Example**

- Take the three rules,

$$\begin{aligned} S &\rightarrow \epsilon \\ S &\rightarrow SS \\ S &\rightarrow (S) \end{aligned}$$

- Begin with the symbol S and derive new strings by repeatedly replacing an occurrence of a left-hand side of a rule (i.e., S) by the corresponding right-hand side (i.e., ϵ or SS or (S)).
- For instance, from S we obtain (S) (by the third rule) and then $((S))$ (by the third rule again) and finally $(())$ (by the first rule).

Context-Free Grammars

- Grammars are language generators specified by substring replacement rules.

- **Definition**

A *context-free grammar* is a quadruple $G = (V, \Sigma, S, R)$, where

V is a finite set, the elements of which are called *variables* or *nonterminals*,

Σ is a finite set disjoint from V , the elements of which are called *terminals*,

S is an element of V , called the *start variable*, and

R is a finite subset of $V \times (V \cup \Sigma)^*$, the elements of which are called *productions* or *rules*.

- The rules of a grammar G are usually written as $A \rightarrow u$, provided G is clear from the context.

Examples

- Formally, the context-free grammar in the example above corresponds to a quadruple $G' = (V, \Sigma, S, R)$ with

$$\begin{aligned} V &= \{S\}, \\ \Sigma &= \{(,)\}, \\ R &= \{S \rightarrow \epsilon, \\ &\quad S \rightarrow SS, \\ &\quad S \rightarrow (S)\} \end{aligned}$$

- Another example of a context-free grammar is $G'' = (V, \Sigma, S, R)$ with

$$\begin{aligned} V &= \{S\}, \\ \Sigma &= \{(,)\}, \\ R &= \{S \rightarrow \epsilon, \\ &\quad S \rightarrow (S)S\} \end{aligned}$$

- In both examples, the start symbol S is the only nonterminal.

More Examples

- Let Σ be the alphabet $\{a, b\}$. Give grammars for the following languages.
 - The set Σ^* of all strings.
 - The empty set.
 - The set of all strings of even length.
 - The set $\{a^n b^n : n \geq 0\}$.
 - The set of all palindromes.
 - The set of all strings with an even number of a 's.
 - The set of all strings with an equal number of a 's and b 's.

Derivations

- We say that a string v can be obtained from u by *replacement* with G , written $u \Rightarrow v$, if and only if there are strings x and y and a rule $A \rightarrow w$ in G such that $u = xAy$ and $v = xwy$.

- For example, we have

$$((S)S) \Rightarrow (()S)$$

and also

$$(()S) \Rightarrow ().$$

- A sequence of replacement steps

$$u_0 \Rightarrow u_1 \Rightarrow \cdots \Rightarrow u_n$$

is called a *derivation* (of u_n from u_0).

- We also say that u_n can be derived from u_0 , and write $u_0 \Rightarrow^* u_n$.

- **Examples**

$$S \Rightarrow SS \Rightarrow S(S) \Rightarrow S((S)) \Rightarrow S(()) \Rightarrow (S)(()) \Rightarrow ()(())$$

$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S) \Rightarrow ()((S)) \Rightarrow ()(())$$

Language Generation

- The *language* $L(G)$ generated by G is the set

$$\{w \in \Sigma^* : S \Rightarrow^* w\},$$

where S is the start symbol of G .

- In other words, $L(G)$ is the set of all strings of terminals that can be derived from the start symbol.
- For example, we gave two grammars that generated the set of all strings of properly balanced left and right parentheses.
- A language is said to be *context-free* if it is generated by a context-free grammar.
- Here is a grammar for the set of all strings in $\{a, b\}^*$ with an equal number of a 's and b 's:

$$\begin{aligned} V &= \{S\}, \\ \Sigma &= \{a, b\}, \\ R &= \{S \rightarrow \epsilon, \\ &\quad S \rightarrow aSbS, \\ &\quad S \rightarrow bSaS\} \end{aligned}$$

Similarity of Derivations

- The same string can often be generated by different derivations.
- For example, with the grammar G' we can generate the string $()()$ by

$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S) \Rightarrow ()()$$

or

$$S \Rightarrow SS \Rightarrow S(S) \Rightarrow S() \Rightarrow (S)() \Rightarrow ()()$$

- The two derivations differ only in the order in which replacement steps are applied.
- The following two derivations of use different replacement steps:

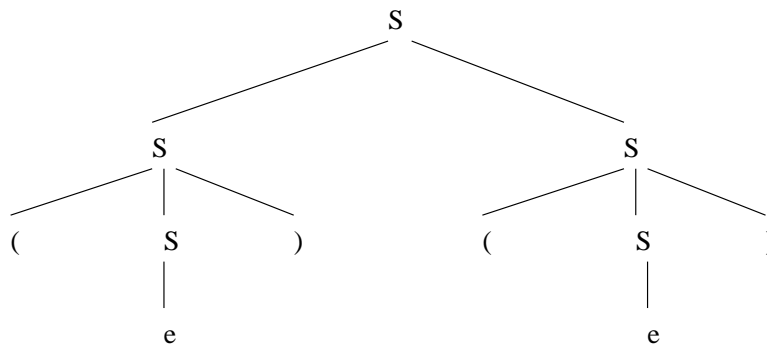
$$\begin{array}{l} S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()SS \\ \Rightarrow ()(S)S \Rightarrow ()()S \Rightarrow ()()(S) \Rightarrow ()()() \end{array}$$

and

$$\begin{array}{l} S \Rightarrow SS \Rightarrow SSS \Rightarrow (S)SS \Rightarrow ()SS \\ \Rightarrow ()(S)S \Rightarrow ()()S \Rightarrow ()()(S) \Rightarrow ()()() \end{array}$$

Parse Trees

- A *parse tree* for a grammar G is a tree T , where
 1. each leaf is labeled either by a terminal or by the empty string,
 2. each non-leaf node is labeled by a nonterminal, and
 3. for each node i labeled by a nonterminal A there is a rule $A \rightarrow x_1x_2\dots x_n$, such that the children of i are labeled by x_1, x_2, \dots, x_n (in this order).
- For example,



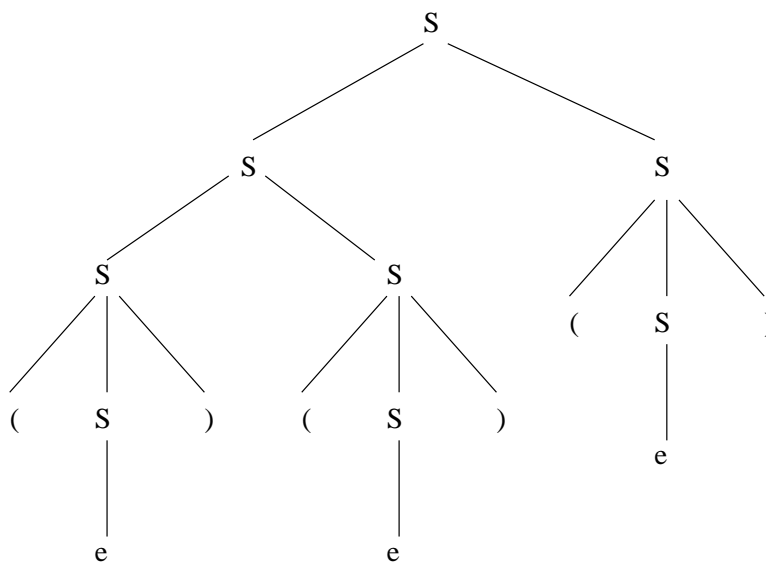
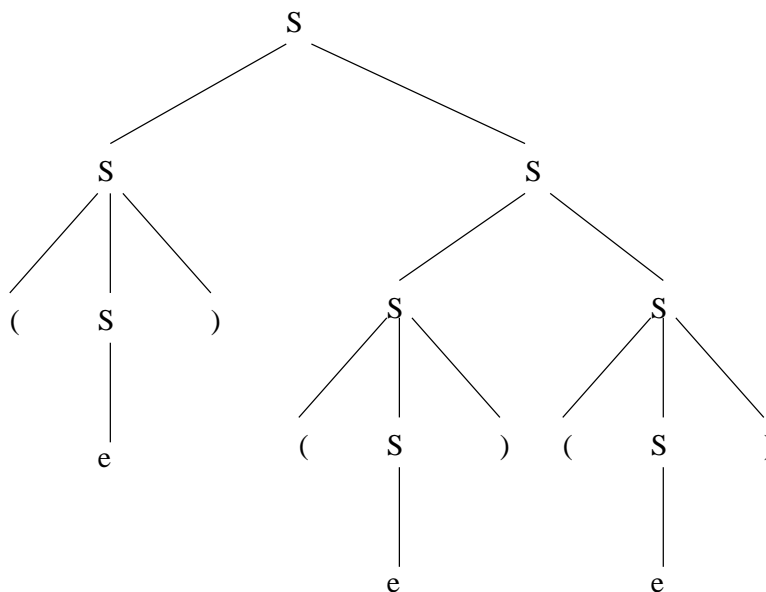
is a parse tree for the grammar G' .

Parse Trees (cont.)

- The string obtained by concatenating the labels of leaves from left to right is called the *yield* of the tree.
- The yield of the parse tree in the above example is the string $()()$.
- To construct a parse tree, start with a single node, usually labeled by the start symbol of the grammar, and then expand nodes labeled with a variable according to a suitable grammar rule.
- The tree above represents the derivations of $()()$ shown earlier: both derivations employ the same replacement steps, only in a different order.
- Two derivations are *similar* if their parse trees are the same.

Ambiguity

- Different parse trees with the same yield represent an ambiguity:



Ambiguous Grammars

- A grammar G is said to be *ambiguous* if there is a string with two or more distinct parse trees.
- Note that a grammar is ambiguous if some string has two different *leftmost* derivations.
- The grammar G' and G'' generate the same language, but one of them is ambiguous.
- The structure of the string $()()()$ is not uniquely determined by the rules of G' .
- Some languages are *inherently ambiguous* in that all context-free grammars generating the language are ambiguous.
- An example of such a language is the set

$$L = \{a^n b^n c^m d^m : n \geq 1, m \geq 1\} \\ \cup \{a^n b^m c^m d^n : n \geq 1, m \geq 1\}.$$

Closure Properties

- **Theorem**

The class of context-free languages is closed under union, concatenation, and Kleene star.

- *Proof sketch.*

- Let $G_1 = (V_1, \Sigma_1, S_1, R_1)$ and $G_2 = (V_2, \Sigma_2, S_2, R_2)$ be context-free grammars with disjoint sets of variables.

- *Union.* Let S be a new symbol and R be the set of rules

$$R_1 \cup R_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}.$$

Then $G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, S, R)$ generates the language $L(G_1) \cup L(G_2)$.

- *Concatenation.* Let S be a new symbol and R be the set of rules

$$R_1 \cup R_2 \cup \{S \rightarrow S_1 S_2\}.$$

Then $G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, S, R)$ generates the language $L(G_1)L(G_2)$.

- *Kleene star.* Let S be a new symbol and R be the set of rules

$$R_1 \cup \{S \rightarrow \epsilon, S \rightarrow SS_1\}.$$

Then $G = (V_1 \cup \{S\}, \Sigma_1, S, R)$ generates the language $L(G_1)^*$.

- Finite sets of strings are evidently context-free, so that we obtain the following result.

- **Corollary**

Every regular language is context-free.

- We have already seen examples of languages that are context-free, but not regular. Next we characterize regular languages more directly by grammars.

Regular Grammars

- A context-free grammar $G = (V, \Sigma, S, R)$ is said to be *regular* if the right-hand side of each rule either contains no variable or else contains a single variable as its last symbol.
- Finite automata can be transformed to regular grammars by taking the states of the automaton as nonterminals and encoding transitions by grammar rules.
- Let $M = (Q, \Sigma, q_0, T, \delta)$ be a deterministic finite automaton and $G_M = (Q, \Sigma, q_0, R)$ be a context-free grammar where

$$R = \{q \rightarrow ap : \delta(q, a) = p\} \\ \cup \{q \rightarrow \epsilon : q \in T\}.$$

Then $L(M) = L(G_M)$.

- Conversely, it is also possible to construct a finite automaton M_G for a given regular grammar G , such that $L(G) = L(M_G)$.

Linear Rules

- A rule $A \rightarrow w$ is said to be *linear* if w contains at most one variable.
- A (linear) rule is called *right-linear* (resp., *left-linear*) if its right-hand side either contains no variable or else contains a single variable as the last (resp., first) symbol.
- A context-free grammar is called (*left, right*) *linear* if each of its rules is (*left, right*) *linear*.
- **Theorem**
 1. A language is regular if, and only if, it is generated by a right-linear grammar.
 2. A language is regular if, and only if, it is generated by a left-linear grammar.
- A language generated by a grammar with both left- and right-linear rules need not be regular, though.