

2. Context-Free Languages

- The language of balanced pairs of parentheses is not a regular one
- On the other hand, it can be described using the following *substitution rules*
 1. $S \rightarrow \varepsilon$ and
 2. $S \rightarrow (S)$
- These *productions* generate the strings of the language L_{parenth} starting from the start variable S

$$S \xrightarrow{2} (S) \xrightarrow{2} ((S)) \xrightarrow{2} (((S))) \xrightarrow{1} (((\varepsilon))) = ((()))$$



- The string being described is generated by substituting *variables* one by one according to the given rules
- The string surrounding a variable does not determine the chosen production \Leftrightarrow *context-free grammar*
- One often abbreviates

$$A \rightarrow w_1 \mid \dots \mid w_k$$
 to describe the alternative productions associated with the variable A

$$A \rightarrow w_1, \dots, A \rightarrow w_k$$
- $S \rightarrow \varepsilon \mid (S)$



Simple arithmetic expressions
 (E = expression, T = term and F = factor)

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T \times F \mid F$$

$$F \rightarrow (E) \mid a$$

Generation the expression $(a + (a)) \times a$

$$\begin{aligned} E &\Rightarrow T \Rightarrow T \times F \Rightarrow F \times F \Rightarrow (E) \times F \Rightarrow (E + T) \times F \Rightarrow \\ &(T + T) \times F \Rightarrow (F + T) \times F \Rightarrow (a + T) \times F \Rightarrow (a + F) \times F \Rightarrow \\ &(a + (E)) \times F \Rightarrow (a + (T)) \times F \Rightarrow (a + (F)) \times F \Rightarrow \\ &(a + (a)) \times F \Rightarrow (a + (a)) \times a \end{aligned}$$



Definition 2.2 A context-free grammar is a 4-tuple

$G = (V, \Sigma, R, S)$, where

- V is a finite set called the *variables*,
- Σ is a finite set, disjoint from V , called the *terminals*
- $V \cup \Sigma$ is the *alphabet* of G ,
- $R \subseteq V \times (V \cup \Sigma)^*$ is a finite set of *rules*, and
- $S \in V$ is the *start variable*

$(A, w) \in R$ is usually denoted as $A \rightarrow w$



- Let $G = (V, \Sigma, R, S)$, strings $u, v, w \in (V \cup \Sigma)^*$, and $A \rightarrow w$ a production in R

- uAv yields string uwv in grammar G , written

$$uAv \Rightarrow_G uwv$$

- String u derives string v in grammar G , written

$$u \Rightarrow_G v,$$

if a sequence $u_1, u_2, \dots, u_k \in (V \cup \Sigma)^*$ ($k \geq 0$) exists s.t.

$$u \Rightarrow_G u_1 \Rightarrow_G u_2 \Rightarrow_G \dots \Rightarrow_G u_k \Rightarrow_G v$$

- $k = 0$: $u \Rightarrow_G u$ for any $u \in (V \cup \Sigma)^*$



- $u \in (V \cup \Sigma)^*$ is a **sentential form** of G if

$$S \Rightarrow_G u$$

- A sentential form consisting of only terminals $w \in \Sigma^*$ is a **sentence** of G

- The **language of the grammar** G consists of sentences

$$L(G) = \{ w \in \Sigma^* \mid S \Rightarrow_G w \}$$

- A formal language $L \subseteq \Sigma^*$ is **context-free**, if it can be generated using a context-free grammar



A context-free grammar is *right-linear* if all its productions are of type $A \rightarrow \varepsilon$ or $A \rightarrow aB$

Theorem Any regular language can be generated using a right-linear context-free grammar

Theorem Any right-linear context-free language is regular

- Hence, right-linear grammars generate exactly regular languages
- However, there are context-free languages which are not regular; e.g., the language of balanced pairs of parentheses L_{parenth}
- Therefore, *context-free languages are a proper superset of regular languages*



Ambiguity

- The sequence of one-step derivations leading from the start variable S to string w

$$S \Rightarrow w_1 \Rightarrow \dots \Rightarrow w_k \Rightarrow w$$

is called the *derivation* of w

In the grammar for arithmetic expressions the sentence $a+a$ can be derived in many different ways:

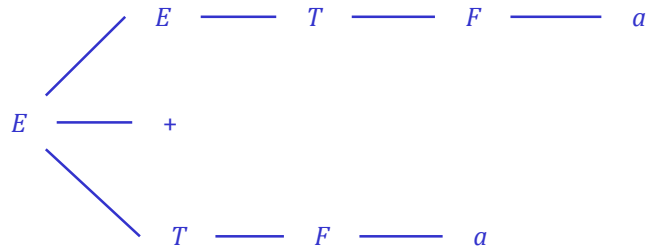
$$1. E \Rightarrow E + T \Rightarrow T + T \Rightarrow F + T \Rightarrow a + T \Rightarrow a + F \Rightarrow a + a$$

$$2. E \Rightarrow E + T \Rightarrow E + F \Rightarrow T + F \Rightarrow F + F \Rightarrow F + a \Rightarrow a + a$$

$$3. E \Rightarrow E + T \Rightarrow E + F \Rightarrow E + a \Rightarrow T + a \Rightarrow F + a \Rightarrow a + a$$

- The differences caused by varying substitution order of variables can be abstracted away by examining *parse trees*





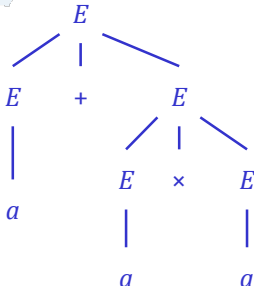
- Context-free grammar G is **ambiguous** if some sentence of G has two (or more) distinct parse trees
- Otherwise the grammar is **unambiguous**
- Language that has no unambiguous context-free grammar is **inherently ambiguous**
- E.g. language $\{a^i b^j c^k \mid i = j \vee j = k\}$ is inherently ambiguous
- An alternative grammar for the simple arithmetic expressions:

$$E \rightarrow E + E \mid E \times E \mid (E) \mid a$$



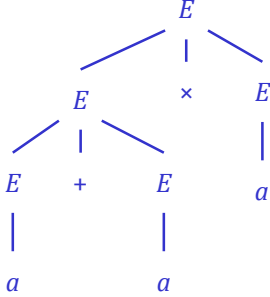
82

$a + a \times a$



```

graph TD
    E1[E] --- E2[E]
    E1 --- P1[+]
    E1 --- E3[E]
    E2 --- A1[a]
    E3 --- E4[E]
    E3 --- M1[×]
    E3 --- E5[E]
    E4 --- A2[a]
    E5 --- A3[a]
            
```




```

graph TD
    E1[E] --- E2[E]
    E1 --- M1[×]
    E1 --- E3[E]
    E2 --- E4[E]
    E2 --- P1[+]
    E2 --- E5[E]
    E4 --- A1[a]
    E5 --- A2[a]
    E3 --- A3[a]
            
```

TAMPERE UNIVERSITY OF TECHNOLOGY
Department of Software Systems
OHJ-2306 Introduction to Theoretical Computer Science, Fall 2009
1.10.2009

83

Chomsky Normal Form



A context-free grammar is in Chomsky normal form (CNF), if

- At most the start variable S derives the empty string,
- Every rule is of the form $A \rightarrow BC$ or $A \rightarrow a$ (except maybe $S \rightarrow \epsilon$),
- The start variable S does not appear in the right-hand side of any rule.

Theorem 2.9 *Any context-free language is generated by a context-free grammar in CNF.*

Proof We convert any grammar into CNF. The conversion has three stages. First, we add a new start variable. Then, we eliminate all ϵ rules and *unit rules*.

TAMPERE UNIVERSITY OF TECHNOLOGY
Department of Software Systems
OHJ-2306 Introduction to Theoretical Computer Science, Fall 2009
1.10.2009

Eliminating ϵ rules

Lemma Any context-free language can be converted into an equivalent grammar in which at most the start variable derives the empty string.

Proof

Let $G = (V, \Sigma, R, S)$.

Computing the variables of G that derive the empty string:

- $NULL = \{A \in V \mid A \rightarrow \epsilon \in R\}$
- Repeat until set $NULL$ does not change any more:

$$NULL += \{A \in V \mid A \rightarrow B_1 \dots B_k \in R, B_i \in NULL \forall i = 1, \dots, k\}$$



Each rule $A \rightarrow X_1 \dots X_k$ in G is replaced by the set of *all* such rules that are of form $A \rightarrow \alpha_1 \dots \alpha_k$, where

$$\alpha_i = \begin{cases} X_i & \text{if } X_i \notin NULL \\ X_i | \epsilon & \text{if } X_i \in NULL \end{cases}$$

In the end we remove all rules that have the form $A \rightarrow \epsilon$.

If $S \rightarrow \epsilon$ belongs to the removed rules, we take a new start variable S' for the grammar and give it rules $S' \rightarrow S \mid \epsilon$.

□





$$\begin{aligned} S &\rightarrow A \mid B \\ A &\rightarrow aBa \mid \varepsilon \\ B &\rightarrow bAb \mid \varepsilon \end{aligned}$$

$$\text{NULL} = \{A, B, S\}$$

$$\begin{aligned} S &\rightarrow A \mid B \mid \varepsilon \\ A &\rightarrow aBa \mid aa \mid \varepsilon \\ B &\rightarrow bAb \mid bb \mid \varepsilon \end{aligned}$$

$$\begin{aligned} S' &\rightarrow S \mid \varepsilon \\ S &\rightarrow A \mid B \\ A &\rightarrow aBa \mid aa \\ B &\rightarrow bAb \mid bb \end{aligned}$$


Eliminating unit rules

A *unit rule* has the form $A \rightarrow B$, where A and B are variables.

Lemma Any context-free language can be converted into an equivalent grammar which has no unit rules.

Proof Let $G = (V, \Sigma, R, S)$.

Computing the unit followers for each variable in G :


1. $F(A) = \{B \in V \mid A \rightarrow B \in R\}$

2. Until the F -sets do not change anymore

$$F(A) += \{F(B) \mid A \rightarrow B \in R\}$$

In the end we remove all unit rules in G and replace them by all rules of the form $A \rightarrow \Omega$, where $B \in F(A)$ and $B \rightarrow \Omega$. \square






$$\begin{aligned}
 S' &\rightarrow S \mid \varepsilon \\
 S &\rightarrow A \mid B \\
 A &\rightarrow aBa \mid aa \\
 B &\rightarrow bAb \mid bb
 \end{aligned}$$

$$\begin{aligned}
 F(S') &= \{ S, A, B \} \\
 F(S) &= \{ A, B \} \\
 F(A) &= \emptyset \\
 F(B) &= \emptyset
 \end{aligned}$$

$$\begin{aligned}
 S' &\rightarrow aBa \mid aa \mid bAb \mid bb \mid \varepsilon \\
 S &\rightarrow aBa \mid aa \mid bAb \mid bb \\
 A &\rightarrow aBa \mid aa \\
 B &\rightarrow bAb \mid bb
 \end{aligned}$$



Once all ε rules and unit rules have been eliminated, all rules have form $A \rightarrow a$, $A \rightarrow X_1 \dots X_k$, $k \geq 2$, or $S \rightarrow \varepsilon$.


For every $a \in \Sigma$ we add to the grammar the variable C_a and rule $C_a \rightarrow a$.

A rule $A \rightarrow X_1 \dots X_k$, $k \geq 2$, is replaced by a set of rules

$$\begin{aligned}
 A &\rightarrow X'_1 A_1 \\
 A_1 &\rightarrow X'_2 A_2 \\
 &\dots \\
 A_{k-2} &\rightarrow X'_{k-2} A_{k-1} \\
 A_{k-1} &\rightarrow X'_{k-1} X'_{k'}
 \end{aligned}$$

where

$$X'_i = \begin{cases} X_i & \text{if } X_i \in V \\ C_a & \text{if } X_i = a \in \Sigma \end{cases}$$

$$\begin{aligned}
 S' &\rightarrow aBa \mid aa \mid bAb \mid bb \mid \varepsilon \\
 S &\rightarrow aBa \mid aa \mid bAb \mid bb \\
 A &\rightarrow aBa \mid aa \\
 B &\rightarrow bAb \mid bb
 \end{aligned}$$

$$\begin{aligned}
 S' &\rightarrow C_a S'_1 \\
 S'_1 &\rightarrow BC_a \\
 S' &\rightarrow C_a C_a \\
 S' &\rightarrow C_b S'_2 \\
 S'_2 &\rightarrow AC_b \\
 S' &\rightarrow C_b C_b \\
 S' &\rightarrow \varepsilon \\
 S &\rightarrow C_a S_1 \\
 S_1 &\rightarrow BC_a
 \end{aligned}$$

$$\begin{aligned}
 S &\rightarrow C_a C_a \\
 S &\rightarrow C_b S_2 \\
 S_2 &\rightarrow AC_b \\
 S &\rightarrow C_b C_b \\
 A &\rightarrow C_a A_1 \\
 A_1 &\rightarrow BC_a \\
 A &\rightarrow C_a C_a \\
 B &\rightarrow C_b B_1 \\
 B_1 &\rightarrow AC_b \\
 B &\rightarrow C_b C_b
 \end{aligned}$$

$$\begin{aligned}
 C_a &\rightarrow a \\
 C_b &\rightarrow b
 \end{aligned}$$



Algorithm CYK

- The strings of a context-free grammar that has been converted into CNF can be *parsed* in $\theta(n^3)$ time using the Cocke-Younger-Kasami algorithm
- In other words, context-free languages can be efficiently recognized
- The operating principle of algorithm CYK is dynamic programming
- For each substring we tabulate those variables from which the substring can be derived from
- If in the end the start variable of the grammar belongs to the set of variables that derive the whole string, the string at hand belongs to the language

