

## Основы теории формальных языков

### Понятие языка, синтаксис, семантика. Формальные и естественные языки. Языки программирования

#### 1. ОСНОВНЫЕ ПОНЯТИЯ

**Естественный язык** (в лингвистике) – язык, используемый для общения людей и не созданный искусственно.

**Формальный язык** – это множество конечных слов над конечным алфавитом, например, язык программирования.

**Лексика** языка программирования – правила составления слов программы из символов языка (идентификаторы, константы, служебные слова, комментарии).

**Синтаксис языка** – система правил, определяющих допустимые конструкции языка программирования из слов языка (построение, порядок, составление).

**Семантика** (смысловое значение) – смысл, который закладывается в каждую конструкцию языка.

**Алфавит** – конечное непустое множество допустимых символов языка (букв языка).

**Пример:**  $V = \{a, b\}$  – алфавит  $V$ , состоящий из двух символов  $a$  и  $b$ .

**Цепочка** – конечная последовательность символов языка (слово в языке).

**Пример:**  $abc$  – цепочка из трех символов.

## 2. ВВЕДЕМ СОГЛАШЕНИЯ

**Алфавит** языка будем обозначать заглавными буквами латинского алфавита:

**$I, V, G, \dots$**

**Символы** алфавита будем обозначать строчными буквами латинского алфавита:

**$a, b, c, \dots$**

**Цепочки** будем обозначать символами греческого алфавита:

**$\alpha, \beta, \gamma, \delta, \lambda, \varepsilon, \omega, \dots$**

*Пример:  $\alpha = b_1 \dots b_n$  – цепочка из  $n$  символов.*

**Пустую цепочку** символов будем обозначать  **$\lambda, \varepsilon$** .

## 3. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

1). **Алфавитом**  $V$  называется конечное множество символов, допустимых в языке.

*Пример:  $V = \{a, b, c\}, I = \{a^n b^n\}$ ,*

где  $n$  — натуральное число.

Алфавит задает язык  $I$ , состоящий из цепочек вида  $ab, aabb, aaabbb$  и т.д.

Язык  $I$  представляет собой бесконечное множество цепочек, но его описание состоит всего из 8 символов, т.е. конечно.

2). **Цепочкой**  $\alpha$  (*альфа*) в алфавите  $I$  называется любая конечная последовательность символов этого алфавита.

Цепочка, которая не содержит ни одного символа, называется пустой цепочкой и обычно обозначается  $\varepsilon$  или  $\lambda$ .

Определение:

**Формальное определение цепочки символов в алфавите  $I$ :**

- a)  $\varepsilon$  – пустая цепочка в алфавите  $I$ ;
- b) если  $\alpha$  – цепочка в алфавите  $I$  и  $a$  – символ этого алфавита, то  $a\alpha$  – цепочка в алфавите  $I$ ;
- c)  $\beta$  – цепочка в алфавите  $I$  тогда и только тогда, когда она является таковой в силу утверждений a) и b).

*Пример:  $\alpha = abc, \beta = aaaa, \gamma = abcaaaa$ .*

3). **Длиной цепочки  $\alpha$**  (обозначается  $|\alpha|$ ) называется число составляющих ее символов.

**Пример:**  $|\alpha| = 3$ ,  $|\beta| = 4$ ,  $|\gamma| = 7$ . Длина пустой цепочки  $|\varepsilon| = 0$ .

4). **Конкатенацией** (сцеплением) цепочек  $\alpha$  и  $\beta$  называется цепочка  $\gamma = \alpha\beta$ , в которой символы данных цепочек записаны друг за другом.

**Пример:**  $\alpha = abc$ ,  $\beta = aaaa$ ,  $\alpha\beta = abcaaaa$ .

5). Для любой цепочки  $\alpha$  справедливо утверждение  $\alpha\varepsilon = \varepsilon\alpha$ .

6).  $\alpha^n$  называется **итерацией цепочки  $\alpha$** .

Справедливы следующие утверждения:

$$\alpha^0 = \varepsilon$$

$$\alpha^n = \alpha^{n-1}\alpha = \alpha\alpha^{n-1} \text{ для } n \geq 1.$$

**Пример:**  $\alpha = ab$ ,  $\alpha^3 = (ab)^3 = ababab$ ,  $\alpha^0 = \varepsilon$

7). Цепочки  $\alpha$  и  $\beta$  **равны** ( $\alpha = \beta$ ), если они имеют один и тот же состав символов, одно и тоже количество символов  $|\alpha| = |\beta|$  и тот же порядок следования символов.

8). **Реверсом** (обращением) цепочки  $\alpha$  называется цепочка  $\alpha^R$ , составленная из символов цепочки  $\alpha$ , записанных в обратном порядке.

**Пример.** Пусть алфавит  $I = \{a, b, c, d\}$ , тогда для цепочек этого алфавита  $\alpha = ab$  и  $\beta = bcd$  будет справедливо  $|\alpha| = 2$ ,  $|\beta| = 3$ ,  $\alpha\beta = abbcd$ ,  $\alpha^2 = abab$ ,  $\beta^R = dc b$ .

9). Пусть  $I$  – алфавит, тогда  $I^+$  – множество всех цепочек, состоящих из символов алфавита  $I$ , исключая пустую цепочку ( $\lambda$ ).

$$\lambda \notin I^+$$

10). Пусть  $I$  – алфавит, тогда  $I^*$  – множество всех цепочек, состоящих из символов алфавита  $I$ , включая пустую цепочку.

$$\lambda \in I^*$$

$$I^* = I^+ \cup \lambda$$

11). Языком  $L(I)$  над алфавитом  $I$  называется произвольное подмножество цепочек из  $I^*$ ,  $L(I) \subseteq I^*$ .

*Пример:*  $I = \{a, b, c\}$ ,

$L_1(I) = \{ab, ac, bc\}$ ,  $L_2(I) = \{a, b, c\}$ ,  $L_3(I) = \{aa, bb, cc\}$ , ...

12). Язык  $L_1(I)$  является подмножеством языка  $L_2(I)$ , если каждая цепочка, входящая в язык  $L_1$ , входит в язык  $L_2$ ; язык  $L_2$  включает язык  $L_1$ .

$$L_1(I) \subseteq L_2(I) \Leftrightarrow (\forall \alpha \in L_1(I) \Rightarrow \alpha \in L_2(I))$$

13). Языки  $L_1(I)$  и  $L_2(I)$  совпадают (эквивалентны), если язык  $L_1(I)$  включает язык  $L_2(I)$  и язык  $L_2(I)$  включает язык  $L_1(I)$ .

$$L_1(I) = L_2(I) \Leftrightarrow (L_1(I) \subseteq L_2(I) \wedge L_2(I) \subseteq L_1(I)).$$

#### 4. ФОРМАЛЬНЫЕ ГРАММАТИКИ. ОСНОВНЫЕ ПОНЯТИЯ.

Способ задания языка называется **грамматикой** этого языка.

Грамматикой мы называем любой способ задания языка.

**Язык  $L$**  можно определить тремя способами:

- а) перечислением всех цепочек языка;
- б) указанием способа (алгоритма) порождения цепочек;
- с) определением метода (алгоритма) распознавания цепочек.

а) язык  $L$  может быть задан с помощью *перечисления цепочек*, если количество цепочек конечно.

б) *способ порождения* (задания) цепочек языка  $L$  называется грамматикой языка  $L$ .

*Например*, грамматика языка

$L = \{a^n b^n\}$ , где  $n$  — натуральное число, задает язык, состоящий из цепочек вида  $ab$ ,  $aabb$ ,  $aaabbb$  и т.д.

с) алгоритм (программа), *распознающий* цепочки языка  $L$ , называется распознавателем.

*Пример:*  $I = \{0,1\}$ ,  $L_1(I) = \{0,1,00,01,10,11\}$ ,  $L_2(I) = \{0^n 1^n, n > 0\}$ , тогда распознаватель  $L_2$  – алгоритм (программа), проверяющий, что цепочка начинается с 0 и содержит одинаковое количество 0 и 1.

---

**Лексика** языка программирования – множество цепочек языка.

**Синтаксис языка** – набор формальных правил, определяющий конструкции (последовательности цепочек) языка.

**Семантика языка** – набор неформальных правил, которые описываются словесно (например, в руководстве программиста). *Пример:* переменную надо объявить до ее применения.

Чтобы создать **язык программирования**, следует определить:

- множество допустимых символов языка (алфавит);
- формально описать множество правильных программ;
- задать семантические правила языка.

Множество допустимых символов может быть проверено.

Определить множество формально правильных программ можно с помощью алгоритма-распознавателя. Распознаватель строится на основе формального описания языка – его формальной грамматики.

Семантические правила могут быть реализованы в виде эвристических алгоритмов (алгоритм, не имеющий строгого обоснования, но дающий приемлемое решение) или в виде словесного (неформального) описания правил языка.

## 5. ОПРЕДЕЛЕНИЕ ПОРОЖДАЮЩЕЙ ГРАММАТИКИ.

$G = \langle T, N, P, S \rangle$  – грамматика языка (порождающая грамматика) – это четверка, где:  
 $T$  – множество терминальных символов,  
 $N$  – множество нетерминальных символов,  
 $P$  – множество правил (или продукций) грамматики,  
 $S$  – начальный символ грамматики.

$T$  – множество терминальных символов (терминалы, алфавит языка) – это символы языка, определяемые грамматикой. Терминалы будем обозначать строчными символами.

$N$  – множество нетерминальных символов (нетерминалы) – символы, применяемые в продукциях  $P$  (символы, определяющие слова, понятия, конструкции языка).  $N \cap T = \emptyset$

Нетерминалы будем обозначать прописными буквами.

$P$  – множество правил вида  $\alpha \rightarrow \beta$  говорят,  $\alpha$  порождает  $\beta$ ,

где  $\alpha \in (N \cup T)^+$ ,  $\beta \in (N \cup T)^*$ .

$S$  – стартовый символ грамматики (аксиома грамматики),  $S \in N$ .

**Пример:**  $G = \langle \{a, b, +\}, \{S, A, B\}, P, S \rangle$ ,  
где правила  $P = \{S \rightarrow A + B, S \rightarrow B + A, A \rightarrow a, B \rightarrow b\}$ .  
Порождаемый язык  $L = \{a + b, b + a\}$

$V = N \cup T$  – **словарь** грамматики  $G = \langle T, N, P, S \rangle$ .

## 6. Нотация Бэкуса-Наура

Для задания *схем грамматик* используются различные формы описания:

- символическая;
- **форма Бэкуса-Наура**;
- итерационная форма;
- синтаксические диаграммы.

Цепочки языка могут содержать метасимволы, имеющие особое назначение. Метаязык, предложенный Бэкусом и Науром (БНФ) использует следующие обозначения:

- символ «**::=**» отделяет левую часть правила от правой (читается: «определяется как»);
- нетерминалы обозначаются произвольной символьной строкой, заключенной в угловые скобки «**<**» и «**>**»;
- терминалы – это символы, используемые в описываемом языке;
- правило может определять порождение нескольких альтернативных цепочек, которые отделяются друг от друга символом вертикальной черты «**|**» (читается: «или»).

**Форма Бэкуса–Наура** (сокр. БНФ, Бэкуса — Наура форма) – формальная система описания синтаксиса, в которой одни синтаксические категории последовательно определяются через другие категории.

**Расширенная БНФ-нотация** включает две конструкции, полезные при спецификации практических языков программирования.

Первая конструкция с помощью фигурных скобок позволяет описать повторение 0 или произвольное число раз некоторой цепочки.

**Например**, синтаксис Турбо-Паскаля (1985 г.) описывает структуру описаний переменных так:

```
<variable-declaration-part> ::=  
    var <variable-declaration> {; <variable-declaration>}
```

Вторая конструкция с помощью квадратных скобок определяет опцию – необязательный элемент.

**Например**, целая константа может быть описана как:

```
<integer-constant> ::= [ + | - ] <digit> {<digit>}
```

## Расширенные Бэкуса-Наура формы (РБНФ)

Для удобства и компактности описаний, в расширенных БНФ (РБНФ) вводятся следующие дополнительные конструкции (*метасимволы*):

- квадратные скобки «[» и «]» означают, что заключенная в них синтаксическая конструкция может отсутствовать;
- фигурные скобки «{» и «}» означают повторение заключенной в них синтаксической конструкции ноль или более раз;
- сочетание фигурных скобок и косой черты «{/» и «/}» используется для обозначения повторения один и более раз;
- круглые скобки «(» и «)» используются для ограничения альтернативных конструкций;
- кавычки используются в тех случаях, когда один из метасимволов нужно включить в цепочку обычным образом.

**Пример1.** Правила, определяющие понятие «идентификатор» некоторого языка программирования:

```
<буква> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t  
          | u | v | w | x | y | z  
<цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9  
<идентификатор> ::= <буква> { (<буква> | <цифра> ) }
```

**Пример2:** форма Бэкуса-Наура, описывающая целое число

```
<целое> := <целое без знака>  
<целое> := +<целое без знака>  
<целое> := -<целое без знака>  
<целое без знака> := <цифра>  
<целое без знака> := <целое без знака><цифра>  
<цифра> := 0  
<цифра> := 1  
<цифра> := 2  
<цифра> := 3  
<цифра> := 4  
<цифра> := 5  
<цифра> := 6  
<цифра> := 7  
<цифра> := 8  
<цифра> := 9
```



**Пример3:** сокращенная форма Бэкуса-Наура, описывающая целое число:

$$\langle \text{целое} \rangle := \langle \text{целое без знака} \rangle | + \langle \text{целое без знака} \rangle | - \langle \text{целое без знака} \rangle$$
$$\langle \text{целое без знака} \rangle := \langle \text{цифра} \rangle | \langle \text{целое без знака} \rangle \langle \text{цифра} \rangle$$
$$\langle \text{цифра} \rangle := 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$$

## 7. ЦЕПОЧКИ ВЫВОДА.

**Выводом** называется процесс порождения предложения языка на основе правил грамматики языка. Записывается  $\alpha \Rightarrow \beta$

Цепочка  $\beta = \delta_1 \gamma \delta_2$  называется непосредственно выводимой из цепочки  $\alpha = \delta_1 \omega \delta_2$  в грамматике  $G = \langle T, N, P, S \rangle$ ,

где  $V = N \cup T$ ,  $\delta_1, \gamma, \delta_2 \in V^*$ ,  $\omega \in V^+$ ,

если в грамматике существует правило:  $\omega \rightarrow \gamma$ .

Цепочка  $\beta$  называется выводимой из цепочки  $\alpha$ , если выполняется одно из двух условий:

- $\beta$  непосредственно выводима из  $\alpha$  ( $\alpha \rightarrow \beta$ );
- существует  $\gamma$  такая, что  $\gamma$  выводима из  $\alpha$ , и  $\beta$  непосредственно выводима из  $\gamma$  ( $\alpha \rightarrow \gamma, \gamma \rightarrow \beta$ ).

Запись ( $\alpha \rightarrow \beta$ ) может читаться одним из следующих способов:

- цепочка  $\alpha$  порождает цепочку  $\beta$ ;
- из цепочки  $\alpha$  выводится цепочка  $\beta$ .

**Пример1:** Грамматика для языка целых десятичных чисел со знаком:

$G (\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, -, +\}, \{S, T, F\}, P, S)$

Правила P:

$S \rightarrow T \mid +T \mid -T$

$T \rightarrow F \mid TF$

$F \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

Построим несколько цепочек вывода в этой грамматике:

1.  $S \rightarrow -T \rightarrow -TF \rightarrow -TFF \rightarrow -FFF \rightarrow -4FF \rightarrow -47F \rightarrow -479$

2.  $S \rightarrow T \rightarrow TF \rightarrow T8 \rightarrow F8 \rightarrow 18$

3.  $T \rightarrow TF \rightarrow T0 \rightarrow TF0 \rightarrow T50 \rightarrow F50 \rightarrow 350$

4.  $F \rightarrow 5$

Получаем, что:  $S \rightarrow -479, S \rightarrow 18, T \rightarrow 350, F \rightarrow 5$ .

**Пример2:**

$G = \langle \{a, b, +\}, \{S, A, B\}, \{S \rightarrow A + B, S \rightarrow B + A, A \rightarrow a, B \rightarrow b\}, S \rangle,$

$S \Rightarrow A + B,$

$A + B \Rightarrow a + B,$

$a + B \Rightarrow a + b,$

$A + B \Rightarrow A + b,$

$A + b \Rightarrow a + b,$

$S \Rightarrow B + A,$

$B + A \Rightarrow b + A,$

$b + A \Rightarrow b + a.$

Порождаемый язык  $L = \{a + b, b + a\}$

## 8. СЕНТЕНЦИАЛЬНАЯ ФОРМА ГРАММАТИКИ.

Вывод называется **законченным** (или **конечным**), если на основе цепочки  $\beta$ , полученной в результате этого вывода (нельзя больше сделать ни одного шага вывода).

$\beta$  называется **сентенциальной формой** грамматики  $G = \langle T, N, P, S \rangle,$

если  $S \Rightarrow^* \beta$  и  $\beta \in (T \cup N)^*$ .

Если  $S \Rightarrow^* \beta$  и  $\beta \in T^*$ , то  $\beta$  называется **терминальной сентенциальной** формой грамматики  $G = \langle T, N, P, S \rangle.$

Тогда цепочки  $-479$  и  $18$  являются конечными сентенциальными формами грамматики целых десятичных чисел со знаком, так как существуют выводы:

$S \Rightarrow^* -479, S \Rightarrow^* 18$

## 9. ЛЕВОСТОРОННИЙ И ПРАВОСТОРОННИЙ ВЫВОДЫ

Вывод называется *левосторонним*, если в нем на каждом шаге вывода правило грамматики применяется к крайнему левому нетерминальному символу в цепочке.

Вывод называется *правосторонним*, если в нем на каждом шаге вывода правило грамматики применяется всегда к крайнему правому нетерминальному символу в цепочке.

*Пример* праволинейной грамматики:

$$G_2 = (\{S\}, \{0, 1\}, P, S),$$

где правила грамматики  $P$  имеют вид:

1.  $S \rightarrow 0S$ ;
2.  $S \rightarrow 1S$ ;
3.  $S \rightarrow \varepsilon$ ,

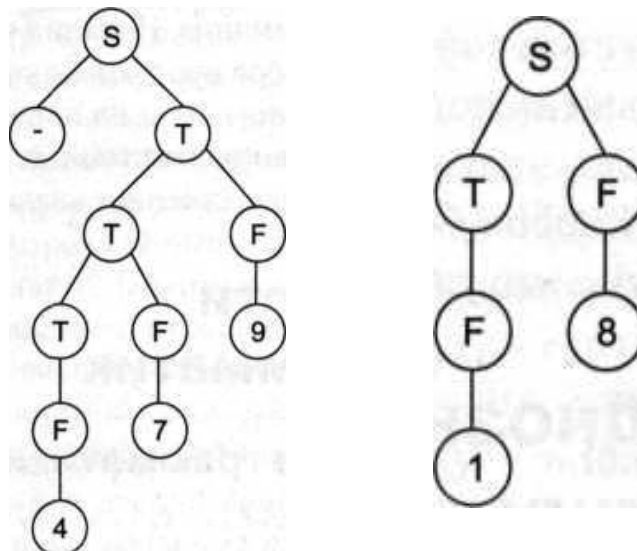
Грамматика определяет язык  $\{0, 1\}^*$ .

## 10. ДЕРЕВО ВЫВООДА

Деревом вывода грамматики  $G(T, N, P, S)$  называется дерево (граф), которое соответствует некоторой цепочке вывода и удовлетворяет следующим условиям:

- каждая вершина дерева обозначается символом грамматики  $A \in (T \cup N \cup \{\varepsilon\})$ ;
- корнем дерева является вершина, обозначенная целевым символом грамматики (или аксиома грамматики) –  $S$ ;
- листьями дерева (концевыми вершинами) являются вершины, обозначенные терминальными символами грамматики или символом пустой цепочки;
- если некоторый узел дерева обозначен нетерминальным символом  $A \in N$ , а связанные с ним узлы — символами  $b_1, b_2 \dots b_n$ ;  $n > 0$ ,  $n \geq i > 0$ :  $b_i \in (T \cup N \cup \{\varepsilon\})$ , то в грамматике  $G(T, N, P, S)$  существует правило  $A \rightarrow b_1 b_2 \dots b_n$ .

Дерево вывода цепочек –479 и 18:



1). Цепочка  $\beta \in (N \cup T)^*$  **выводима** из цепочки  $\alpha \in (N \cup T)^+$  в грамматике  $G = \langle T, N, P, S \rangle$ , если существуют цепочки, такие что  $\alpha \Rightarrow \gamma_0 \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \dots \Rightarrow \gamma_n \Rightarrow \beta$ . Записывается как  $\alpha \Rightarrow^* \beta$ .

Тогда последовательность  $\alpha, \gamma_0, \gamma_1, \gamma_2, \gamma_n, \beta$  называется **выводом** длины  $n$ .

Пример:  $G = \langle \{0,1\}, \{S, A\}, \{S \rightarrow 0A1, 0A \rightarrow 00A1, A \rightarrow \lambda\}, S \rangle$

Вывод  $S \Rightarrow 0A1 \Rightarrow 00A1 \Rightarrow 000A1$

Длина вывода цепочки  $000A1$  из стартового символа  $S$  равна 2.

(Запись второго правила рекурсивна).

2). Запись  $\alpha \Rightarrow^* \beta$  предполагает  $n \geq 0$  шагов вывода  $\beta$  из  $\alpha$ .

В том случае, если  $\alpha \Rightarrow \beta$ , то число шагов вывода  $n = 0$ .

Запись  $\alpha \Rightarrow^+ \beta$  предполагает  $n > 0$  шагов вывода  $\beta$  из  $\alpha$ .

3). Для записи правил вывода с одинаковыми левыми частями

$$\alpha \rightarrow \beta_1 \quad \alpha \rightarrow \beta_2 \quad \dots \quad \alpha \rightarrow \beta_n$$

будем пользоваться сокращенной записью

$$\alpha \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_n.$$

Каждое  $\beta_i$ ,  $i = 1, 2, \dots, n$  – *альтернатива* правила вывода из цепочки  $\alpha$ .

4).  $L(G)$  – язык, *порождаемый* грамматикой  $G$ .

Язык  $L(G)$  содержит все терминальные цепочки, выводимые из  $S$ :

$$L(G) = \{\alpha \in T^* \mid S \Rightarrow^* \alpha\}.$$

Язык, порождаемый грамматикой – это множество всех выводимых из аксиомы грамматики терминальных цепочек

5).  $L(G)$  – множество терминальных сентенциальных форм грамматики  $G$ .

**Пример:**

$$G = \langle \{a, b, +\}, \{S, A, B\}, \{S \rightarrow A + B, S \rightarrow B + A, A \rightarrow a, B \rightarrow b\}, S \rangle,$$

$$L(G) = \{a + b, b + a\}$$

6).  $G_2 = G_1 \Leftrightarrow L(G_2) = L(G_1)$  – грамматики *эквивалентны*, если они порождают один язык.

**Пример:**

Язык  $L$  в алфавите  $V = \{1, 0\}$ , состоящий из пустой строки и строк, каждая из которых содержит строку, состоящую из нулей и такого же количества единиц, можно также описать с помощью формальной системы определения множеств как  $L = \{0^n 1^n \mid n \geq 0\}$ .

## 11. КЛАССИФИКАЦИЯ ЯЗЫКОВ И ГРАММАТИК ПО ХОМСКОМУ

Хомский Ноам: 1928, США, лингвист, профессор Массачусетского технологического института, автор классификации формальных языков (иерархия Хомского), ввел понятие порождающей грамматики (1950).

### 1) Иерархия Хомского:

$$G_0 \supset G_I \supset G_{II} \supset G_{III},$$

где  $G_0, G_I, G_{II}, G_{III}$  множества грамматик типа 0, 1, 2 и 3.

### 2) Грамматики типа 0:

$G_0 = \langle T, N, P, S \rangle$  – **неограниченные** грамматики, у которых нет никаких ограничений для правил.

Правила имеют вид:  $\alpha \rightarrow \beta$ , где  $\alpha \in V^+$ ,  $\beta \in V^*$ .

Для грамматики без ограничений на вид правил такой алгоритм распознавания в общем случае построить нельзя.

### 3) Грамматики типа 1:

$G_I = \langle T, N, P, S \rangle$  – **контекстно-зависимые (КЗ)** грамматики

Правила контекстно-зависимых грамматик имеют вид:

$$\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2, \text{ где } \alpha_1, \alpha_2 \in V^*, A \in VN, \beta \in V^+.$$

Правила неукорачивающих грамматик имеют вид:

$$\alpha \rightarrow \beta, \text{ где } \alpha \in V^+, \beta \in V^+ \text{ и } |\alpha| \leq |\beta|.$$

**Контекстно-зависимая грамматика:** один и тот же нетерминальный символ может быть заменен на ту или иную цепочку символов в зависимости от контекста (цепочки) в которой они встречаются.

В **неукорачивающих грамматиках** любая цепочка символов может быть заменена на цепочку символов не меньшей длины.

### 4) Грамматики типа 2:

$G_{II} = \langle T, N, P, S \rangle$  – **контекстно-свободные (КС)** грамматики.

Правила имеют вид:  $A \rightarrow \alpha$ , где  $A \in N$ ,  $\alpha \in V^*$ , где  $A$  — нетерминал,  $\beta$  — цепочка нетерминалов и терминалов.

**Пример:** в языках программирования символ “присваивание” раскрывается однозначно и не зависит от того, что окружает присваивание.

**5) Грамматики типа 3:**

$G_{III} = \langle T, N, P, S \rangle$  – регулярные грамматики.

Регулярные грамматики бывают праволинейными и леволинейными.

Правила праволинейной грамматики имеют вид:

$$A \rightarrow \alpha \text{ или } A \rightarrow \alpha B, \text{ где } A, B \in N, \alpha \in T^*.$$

Правила леволинейной грамматики имеют вид:

$$A \rightarrow \alpha \text{ или } A \rightarrow B\alpha, \text{ где } A, B \in N, \alpha \in T^*.$$

**6) Соотношения грамматик в иерархии Хомского:**

- a) любая регулярная грамматика является контекстно-свободной грамматикой;
- b) любая контекстно-свободная грамматика является контекстно-зависимой грамматикой;
- c) любая контекстно-зависимая грамматика является грамматикой типа 0.

**7) Формальные языки классифицируются по типу порождающих их грамматик.**

**8) Соотношения между типами формальных языков:**

- a) каждый регулярный язык является контекстно-свободным языком, но существуют контекстно-свободные языки, которые не являются регулярными;
- b) каждый контекстно-свободный язык является контекстно-зависимым, но существуют контекстно-зависимые, которые не являются контекстно-свободными.
- c) каждый контекстно-зависимый язык является языком типа 0.



**Пример1:**  $G_1 = \langle T_1, N_1, P_1, S \rangle$ , где  $T_1 = \{a, 0, 1, \dots, 9\}$ ,  $N_1 = \{S, D\}$ ,

Правила:  $P_1: S \rightarrow a \mid aD$ ,

$D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid \dots 9 \mid 0D \mid 1D \mid 2D \mid 3D \mid \dots 9D$ .

$L(G_1) = \{a, a0, a1, \dots, a9, a01, a02, \dots a09, \dots, a1, \dots\}$  –

регулярный правосторонний язык (тип 3).

**Пример2:**  $G_2 = \langle T_2, N_2, P_2, S \rangle$ , где  $T_2 = \{a, 0, 1, \dots, 9\}$ ,  $N_2 = \{S, D, F\}$ ,

Правила:  $P_2: S \rightarrow a \mid aD$ ,

$F \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid \dots 9 \mid 0$ ,

$D \rightarrow F \mid DF$ .

$L(G_2) = \{a, a0, a1, \dots, a9, a01, a02, \dots a09, \dots, a1, \dots\}$  –

язык, порожденный контекстно-свободной грамматикой (тип 2).

$L(G_2) = L(G_1) \Leftrightarrow G_2 = G_1$ , следовательно  $G_2$  и  $G_1$  эквивалентные грамматики.

**Пример3:**

$G_3 = \langle T_3, N_3, P_3, S \rangle$ ,  $T_3 = \{0, 1\}$ ,  $N_3 = \{S, A\}$ ,

где правила  $P_3: S \rightarrow 0A1, 0A \rightarrow 00A11, A \rightarrow \lambda$ .

$L(G_3) = \{01, 0011, 000111, 00001111, \dots\} = \{0^n 1^n \mid n > 0\}$  -грамматика

язык типа 0.

**Пример4:**

$G_4 = \langle T_4, N_4, P_4, S \rangle$ ,  $T_4 = \{a, c, b\}$ ,  $N_4 = \{S, Q\}$ ,

где правила  $P_4: S \rightarrow aQb \mid acsb, Q \rightarrow cSc$ .

$L(G_4) = \{acsb, acacsbcb, acacacsbcbcb, \dots\} = \{(ac)^n (cb)^n \mid n > 0\}$  -

грамматика и язык типа 2 (контекстно-независимая).

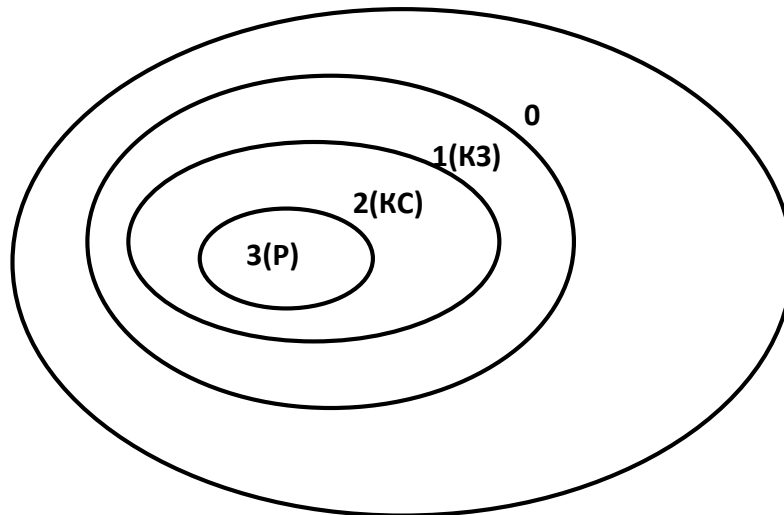
**Пример5:**  $G_5 = \langle T_5, N_5, P_5, S \rangle$ ,  $T_5 = \{+, -, *, /, (, ), a, b\}$ ,  $N_5 = \{S\}$ , где  $P_5$

$: S \rightarrow S + S \mid S - S \mid S * S \mid S / S \mid (S) \mid a \mid b$ .

$L(G_5) = \{a * b, a + b, a - b, b / a, (a + b) * (b - a), \dots\}$  - грамматика и язык

типа 2 (контекстно-независимая).

## Соотношение типов грамматик и языков:



- Тип 3 (Р) – регулярная грамматика;  
Тип 2 (КС) – контекстно-свободная грамматика;  
Тип 1 (КЗ) – контекстно-зависимая грамматика;  
Тип 0 – неограниченная грамматика.

**Примеры** различных типов формальных языков и грамматик по классификации Хомского.

Дана грамматика  $G = \langle T, N, P, S \rangle$ .

а) **Язык типа 0**  $L(G) = \{a^2b^{n^2-1} \mid n \geq 1\}$  определяется грамматикой с правилами вывода:

- |                                  |                                    |
|----------------------------------|------------------------------------|
| 1) $S \rightarrow aaCFD$ ;       | 2) $AD \rightarrow D$ ;            |
| 3) $F \rightarrow AFB \mid AB$ ; | 4) $Cb \rightarrow bC$ ;           |
| 5) $AB \rightarrow bBA$ ;        | 6) $CB \rightarrow C$ ;            |
| 7) $Ab \rightarrow bA$ ;         | 8) $bCD \rightarrow \varepsilon$ . |

б) **Контекстно-зависимый язык**  $L(G) = \{a^n b^n c^n \mid n \geq 1\}$  определяется грамматикой с правилами вывода:

- |                                    |                          |
|------------------------------------|--------------------------|
| 1) $S \rightarrow aSBC \mid abc$ ; | 2) $bC \rightarrow bc$ ; |
| 3) $CB \rightarrow BC$ ;           | 4) $cC \rightarrow cc$ ; |
| 5) $BB \rightarrow bb$ .           |                          |

в) **Контекстно-свободный язык**  $L(G) = \{(ab)^n (cb)^n \mid n > 0\}$  определяется грамматикой с правилами вывода:

- |                                    |
|------------------------------------|
| 1) $S \rightarrow aQb \mid acsb$ ; |
| 2) $Q \rightarrow cSc$ .           |

г) **Регулярный язык**  $L(G)=\{\omega \mid \omega \in \{a, b\}^+\}$ , где нет двух рядом стоящих  $a$  определяется грамматикой с правилами вывода:

- 1)  $S \rightarrow A / B$ ;
- 2)  $A \rightarrow a / Ba$ ;
- 3)  $B \rightarrow b / Bb / Ab$ .

Все реальные языки программирования являются контекстно-зависимыми (тип 1), но большинство может быть разобрано как контекстно-свободные (тип 2) с последующей обработкой в виде проверок типов и т.п.

Фронтенд компилятора делят на три стадии:

- лексический анализ (в рамках регулярной грамматики текст делится на строковые литералы, числа, идентификаторы, операторы, разделители, убираются пробелы и комментарии);
- синтаксический анализ (в рамках контекстно-свободной грамматики);
- семантический анализ (проверка соответствия контекстно-зависимым правилам, таким как “переменная должна быть объявлена заранее”).

## 12. СПОСОБЫ ЗАДАНИЯ СХЕМ ГРАММАТИК.

Схема грамматики содержит правила вывода, определяющие синтаксис языка, другими словами, определяющие структуру цепочек порождаемого языка. Для задания правил используются различные формы описания:

- символическая;
- форма Бэкуса-Наура;
- итерационная форма;
- синтаксические диаграммы.

*Пример* символической формы задания правил грамматики:

$G = \langle \{ \text{if, then, else, a, b} \}, \{ S \}, P, S \rangle$ ,  
где  $P = \{ S \rightarrow \text{if b then S else S} \mid \text{if b then S} \mid \text{a} \}$ .

## 12.1. Формы Бэкуса - Наура

**Нотация Бэкуса-Наура.** Метаязык для формального задания различных грамматик формальных языков.

**Форма Бэкуса — Наура** (сокр. БНФ, Бэкуса — Наура форма) — формальная система описания синтаксиса, в которой одни синтаксические категории последовательно определяются через другие категории.

БНФ использует следующие обозначения:

- символ «**::=**» отделяет левую часть правила от правой (читается: «определяется как»);
- нетерминалы обозначаются произвольной символьной строкой, заключенной в угловые скобки «**<**» и «**>**»;
- терминалы – это символы, используемые в описываемом языке;
- правило может определять порождение нескольких альтернативных цепочек, которые отделяются друг от друга символом вертикальной черты «**|**» (читается: «или»).

### Расширенные Бэкуса-Наура формы (РБНФ)

Для удобства и компактности описаний, в расширенных БНФ (РБНФ) вводятся следующие дополнительные конструкции (**метасимволы**):

- квадратные скобки «**[**» и «**]**» означают, что заключенная в них синтаксическая конструкция может отсутствовать;
- фигурные скобки «**{**» и «**}**» означают повторение заключенной в них синтаксической конструкции ноль или более раз;
- сочетание фигурных скобок и косой черты «**{/}**» и «**/}**» используется для обозначения повторения один и более раз;
- круглые скобки «**(**» и «**)**» используются для ограничения альтернативных конструкций;
- кавычки используются в тех случаях, когда один из метасимволов нужно включить в цепочку обычным образом.

**Пример1.** Цифру можно определить как:

```
<digit> ::= "1" | "2" | ... | "9" | "0"
```

**Пример2.** Общей формы грамматики **РБНФ-описания** в виде РБНФ:

```
Синтаксис      ::= { СинтОператор }.
СинтОператор   ::= идентификатор "=" СинтВыражение ".".
СинтВыражение  ::= СинТерм {"|" СинТерм}.
СинТерм        ::= СинтФактор { СинтФактор }.
СинтФактор     ::= идентификатор | цепочка
                  | "(" СинтВыражение ")" | "[" СинтВыражение "]"
                  | "{" СинтВыражение "}";
```

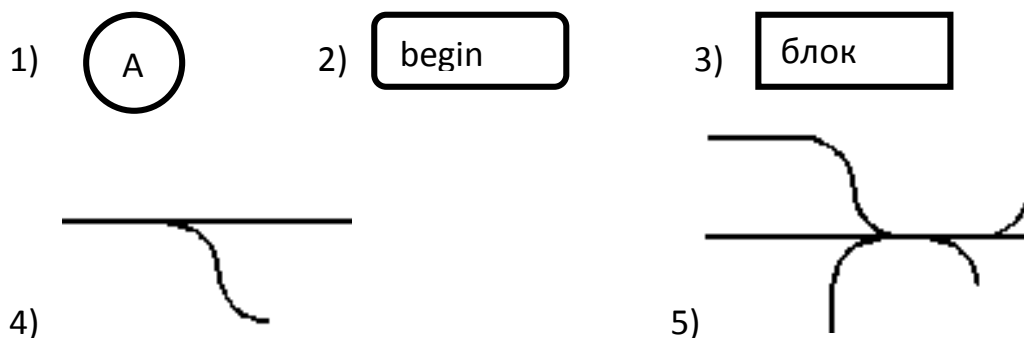
## 12.2. Диаграммы Вирта

Наряду с текстовыми способами описания синтаксиса языков широко используются и графические метаязыки, среди которых наиболее широкую известность получил язык диаграмм Вирта, впервые примененный для описания языка Паскаль.

В метаязыке диаграмм Вирта используются графические примитивы, представленные на рисунке 1.4.1.

При построении диаграмм учитывают следующие правила:

- каждый графический элемент, соответствующий терминалу или нетерминалу, имеет по одному входу и выходу, которые обычно изображаются на противоположных сторонах;
- каждому правилу соответствует своя графическая диаграмма, на которой терминалы и нетерминалы соединяются посредством дуг;
- альтернативы в правилах задаются ветвлением дуг, а итерации - их слиянием;
- должна быть одна входная дуга (располагается обычно слева или сверху), задающая начало правила и помеченная именем определяемого нетерминала, и одна выходная, задающая его конец (обычно располагается справа и снизу);
- стрелки на дугах диаграмм обычно не ставятся, а направления связей отслеживаются движением от начальной дуги в соответствии с плавными изгибами промежуточных дуг и ветвлений.



Графические примитивы диаграмм Вирта

- 1) – терминальный символ, принадлежащий алфавиту языка;
- 2) – постоянная группа терминальных символов, определяющая название лексемы, ключевое слово и т.д.;
- 3) – нетерминальный символ, определяющий название правила;
- 4) – входная дуга с именем правила, определяющая его название;
- 5) – соединительные линии, обеспечивающие связь между терминальными и нетерминальными символами в правилах.

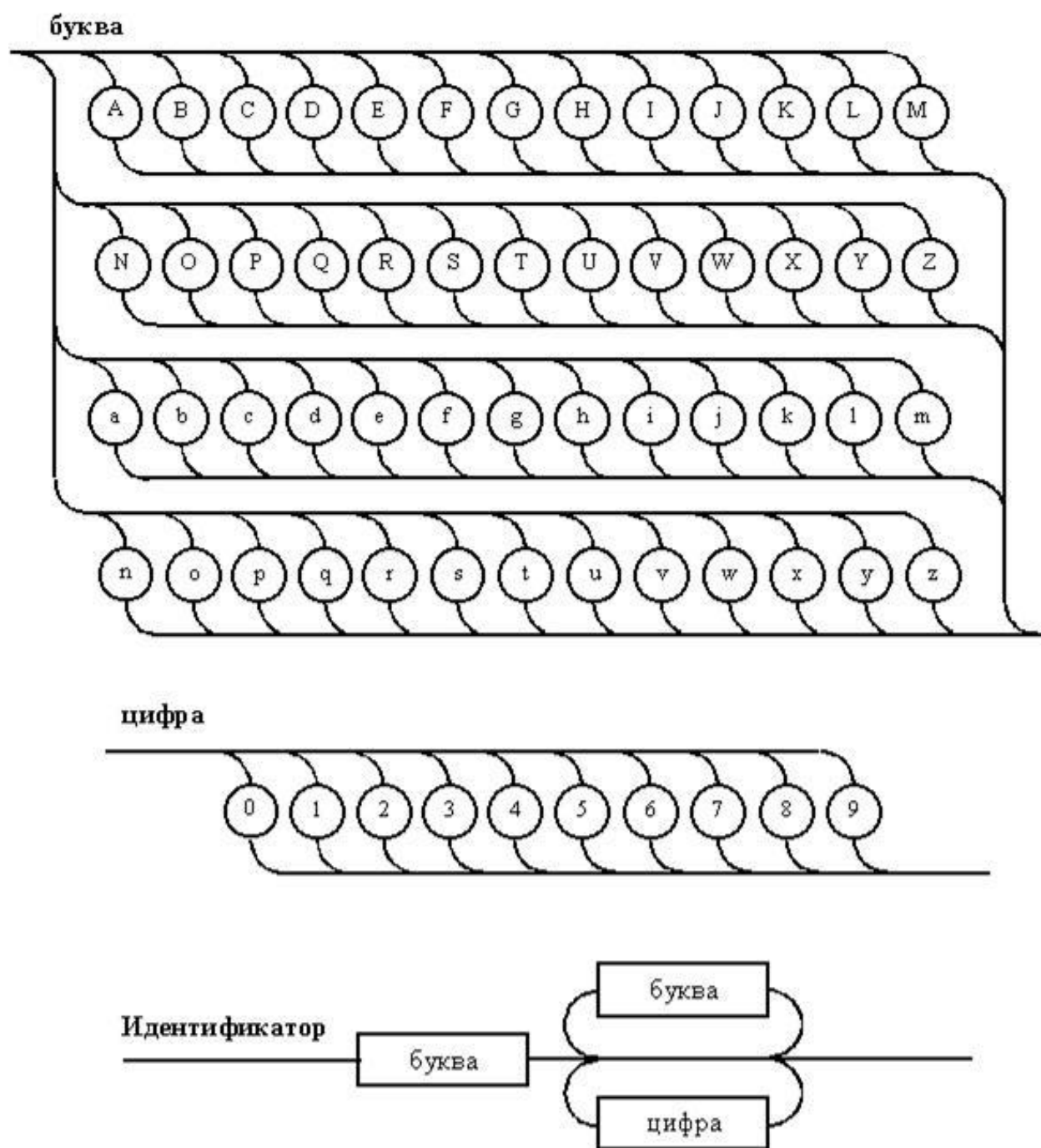
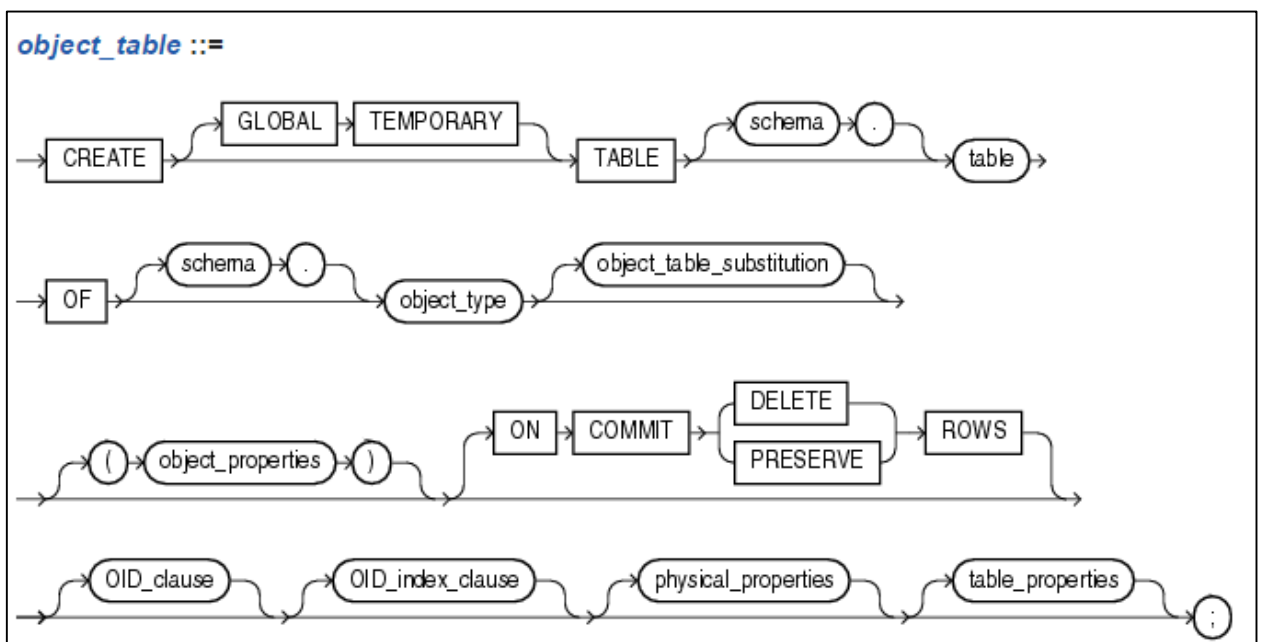
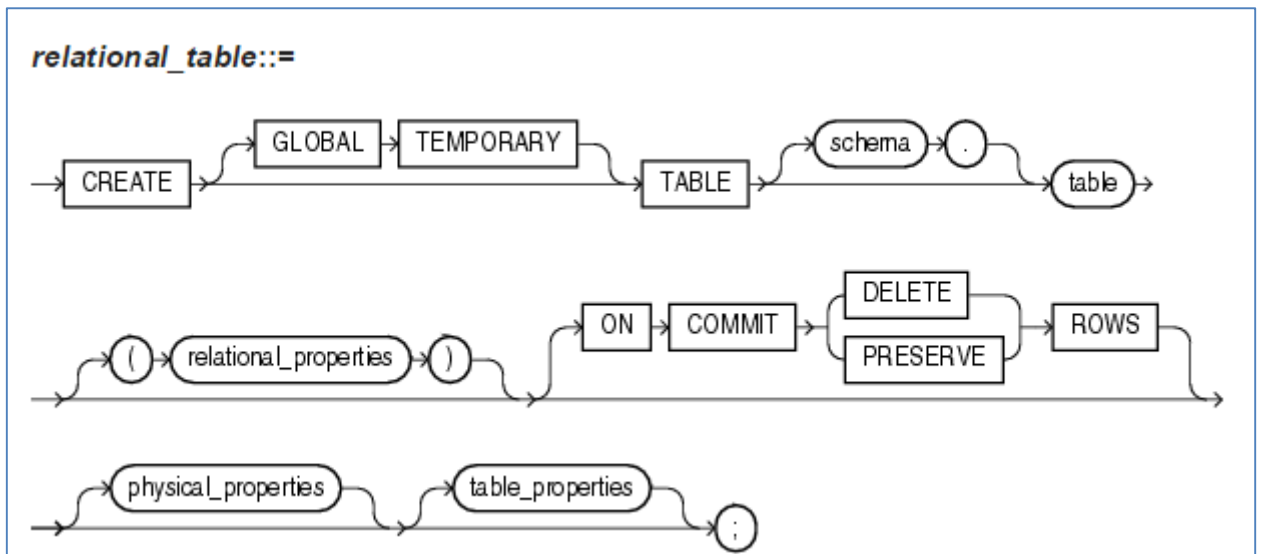
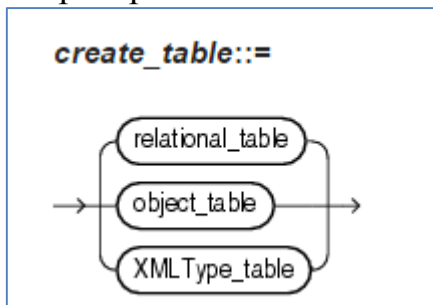


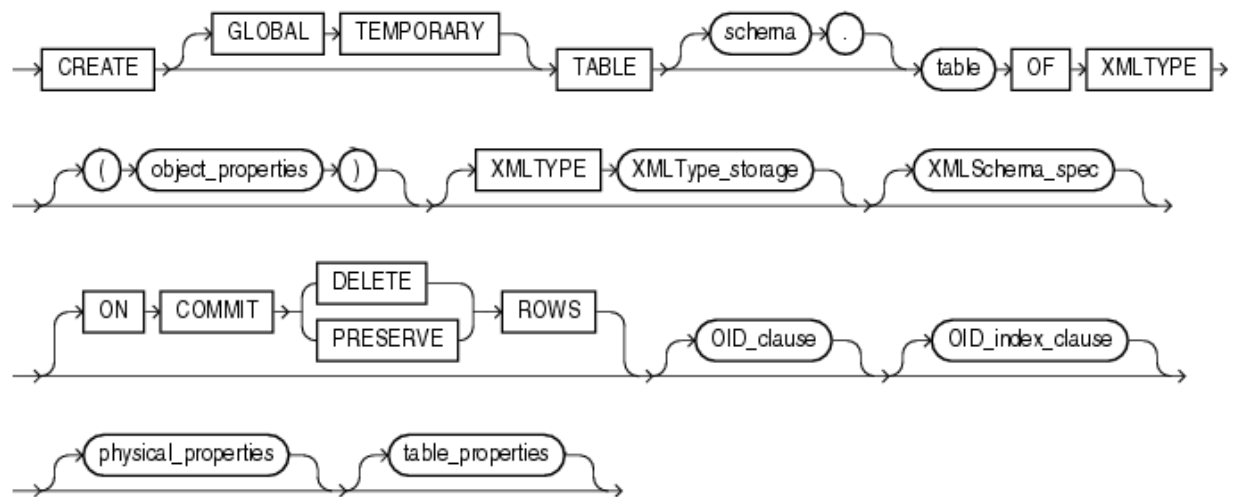
Диаграмма Вирта понятия «идентификатор»

**Пример:** диаграмма задающая язык SQL СУБД Oracle. В примере описывается оператор CREATE TABLE – создание таблицы в базе данных.





***XMLType\_table ::=***



***relational\_properties ::=***

