**Порождающие грамматики (грамматики Н. Хомского)**

Порождающие грамматики — это простой и мощный механизм, позволяющий задавать обширный класс языков, содержащих бесконечное множество цепочек.

С помощью порождающих грамматик мы сможем, в частности, определить языки L3, L4 и L5, для задания которых мы раньше ограничивались словесными формулировками.

Порождающие грамматики используются и при описании синтаксиса языков программирования.

Порождающей грамматикой называется четверка:

**G = (T,N,P,S),**

где **Т** — конечное множество терминальных (основных) символов — основной алфавит. Элементами множества **Т** являются символы, из которых в конечном итоге и состоят цепочки языка, порождаемого данной грамматикой. Терминальный (от лат. terminus — предел, конец) и означает «конечный, концевой».

**Т** — это не что иное, как алфавит языка, порождаемого грамматикой.

В дальнейшем, если не оговорено особо, терминальные символы, или просто терминалы, будут обозначаться малыми буквами латинского алфавита: а,b,с и т. д.;

**N** — конечное множество нетерминальных (вспомогательных) символов — вспомогательный алфавит. Нетерминальные символы, по-другому нетерминалы, — это понятия грамматики (языка), которые используются при его описании.

Нетерминалы будем обозначать заглавными латинскими буквами: А, В, C,D,E и т. д.;

**Р** — конечное множество правил вывода, называемых также продукциями. Каждое правило множества **Р** имеет вид:

**α** → **β**,

где **α** и **β** — цепочки терминальных и нетерминальных символов. Цепочка **α** не должна быть пустой, цепочка **β** может быть пуста:

**α** ∈ (**Т** ∪ **N**)+;

**β** ∈ (**Т** ∪ **N**)\*

Правило **α** → **β** определяет возможность подстановки **β** вместо **α** в процессе вывода (порождения) цепочек языка;

**S** (S ∈ N) — начальный символ грамматики — один из множества нетерминальных символов, начальный (стартовый) нетерминал. Начальный нетерминал — это понятие, соответствующее правильному предложению языка. Например, начальный нетерминал грамматики выражений обозначает «выражение», а начальный нетерминал грамматики языка Паскаль — «программа».

**Примеры грамматик. Порождение предложений языка**

***Пример 1***. Рассмотрим грамматику

G1 = ( {а, b}, {S}, {S → aSb, S → **ε** }, S).

Здесь все элементы четверки записаны явно. Множество терминальных символов Т = {а, b}; множество нетерминалов содержит один элемент: N = {S}, а множество правил — два: Р = {S →aSb, S → **ε** }; роль начального нетерминала исполняет S.

Грамматика может использоваться для порождения (вывода) цепочек — предложений языка. Процесс порождения начинается с начального нетерминала, в нашем примере это S. Если среди правил есть такое, в левой части которого записана цепочка S, то начальный нетерминал может быть заменен правой частью любого из таких правил. Оба правила грамматики G1 содержат в левой части S.

Применим подстановку, заданную первым правилом, заменив S на aSb:

S $→$ aSb.

К получившейся цепочке aSb снова, если удастся, можно применить одно из правил грамматики. Если в цепочке есть подцепочка, совпадающая с левой частью хотя бы одного из правил, то эту подцепочку можно заменить правой частью любого из таких правил. В цепочке aSb есть подцепочка S, совпадающая с левой частью обоих правил грамматики G1. Мы вправе применить любое из этих правил.

Используем снова правило (1) для продолжения вывода:

S$→$aSb$→$aaSbb.

Теперь к получившейся цепочке применим правило (2) {S → **ε**), заменив S пустой цепочкой. Получим такую последовательность подстановок (саму букву **ε** в последней цепочке записывать, конечно, не нужно):

S $→$aSb $→$ aaSbb $→$ aabb.

Очевидно, что к получившейся цепочке ни одно из правил грамматики G1 больше не применимо. Процесс порождения завершен.

 Нетрудно заметить, что с помощью грамматики G1 можно породить любую цепочку языка L1 = {anbn | n ≥ 0}, применив к начальному нетерминалу правило (1) (S → aSb) n раз, а затем один раз правило (2).

В то же время, грамматика G1 не порождает ни одной цепочки терминальных символов, не принадлежащей языку L1. To есть множество терминальных цепочек, порождаемых грамматикой G1 совпадает с языком L1. Другими словами, грамматика G1 порождает язык L1.

L(G1)=L1.

Обычно при записи грамматики не выписывают четверку ее элементов явно. При соблюдении соглашений об обозначениях терминалов и нетерминалов достаточно записать только правила. Правила с одинаковой левой частью можно объединять, отделяя альтернативные правые части вертикальной чертой. Первым записывается правило, содержащее в левой части начальный нетерминал. С учетом этого грамматика G1 может быть записана так:

G1: S →aSb | **ε** .

***Пример 2***. Рассмотрим грамматику G2 (цифры справа — номера правил)

G2: S→aSBc (1)

 S→abc (2)

 сВ → Bc (3)

 bВ → bb (4)

 S→ **ε** (5)

Проведем вывод цепочек из начального нетерминала грамматики G2. Под стрелкой, обозначающей подстановку, будем указывать, как и раньше, номер примененного правила. Итак:

S$→$aSBc$→$aabcBc$→$aabBcc$→$ ааbbсс.

Еще одна серия подстановок:

S$→$aSBc$→$aaSBcBc$→$aaabcBcBc$→$ aaabBccBc$→$aaabBcBcc$→$

aaabBBccc$→$aaabbBccc$→$ аааbbbссс.

Можно убедиться, что грамматика G2 порождает цепочки терминалов вида anbncn

и никакие другие. Количество повторений символов в результирующей цепочке определяется тем, на каком шаге применяется правило (2). Наличие правила (5)

позволяет получить пустую цепочку, если применить это правило первым, в то время как попытка использования этого правила на последующих шагах не позволит вывести цепочку терминалов. Грамматика G2 порождает множество терминальных цепочек, совпадающее с языком L2 = {anbncn | n ≥ 0}:

L(G2) = L2.

***Пример 3***. Грамматика, порождающая язык правильных скобочных выражений (язык Дика).

G3: S→(S) (1)

 S→SS (2)

 S→ **ε** (3)

Нетрудно понять логику построения правил этой грамматики. Смысл первого правила таков: заключив в скобки правильное скобочное выражение S, мы снова получим правильное скобочное выражение. Второе правило означает, что два правильных скобочных выражения, записанные одно за другим, дают новое правильное выражение. Наконец, по правилу (3) пустая цепочка считается правильным выражением. Если бы мы решили, что не следует разрешать пустые выражения, правило (3) можно было бы заменить на S → ().

***Пример 4***. Грамматика простых арифметических выражений. Единственным нетерминалом этой грамматики (он же начальный) будет *Выражение*:

G4:

*Выражение*→*Выражение*+*Выражение* |

*Выражение* - *Выражение* |

*Выражение*\* *Выражение* |

*Выражение* / *Выражение*

| а |b |с | ( Выражение ) .

Такая грамматика порождает цепочки терминалов, являющиеся правильными арифметическими выражениями. Символы а,b и с в таких выражениях обозначают операнды, а «+», «-», «\*», и «/» — знаки операций. Разрешаются круглые скобки (в том числе вложенные). Примеры правильных выражений:

(a+b)/(b\*c),

(а).

Все операции двуместные, унарные плюс и минус не предусмотрены, поэтому, например, цепочка -а не принадлежит языку, порождаемому этой грамматикой.

Выражение — одно из основных понятий языков программирования. В дальнейшем грамматикам выражений будет уделено немалое внимание. Чтобы запись этих грамматик была короче, заменим нетерминал Выражение на Е (от expression - выражение), вернувшись тем самым к принятым раньше соглашениям об именовании нетерминалов. Тогда грамматика, которую обозначим G5, запишется следующим образом:

G5: E→E + E|E-E|E\*E|E/E|a|b|c| (E).

Очевидно, что она порождает тот же язык, что и грамматика G4. А именно:

L(G5) = L5.

**Порождающие грамматики (грамматики Н. Хомского)**

Порождающие грамматики — это простой и мощный механизм, позволяющий задавать обширный класс языков, содержащих бесконечное множество цепочек.

С помощью порождающих грамматик мы сможем, в частности, определить языки L3, L4 и L5, для задания которых мы раньше ограничивались словесными формулировками.

Порождающие грамматики используются и при описании синтаксиса языков программирования.

Порождающей грамматикой называется четверка:

**G = (T,N,P,S),**

где **Т** — конечное множество терминальных (основных) символов — основной алфавит. Элементами множества **Т** являются символы, из которых в конечном итоге и состоят цепочки языка, порождаемого данной грамматикой. Терминальный (от лат. terminus — предел, конец) и означает «конечный, концевой».

**Т** — это не что иное, как алфавит языка, порождаемого грамматикой.

В дальнейшем, если не оговорено особо, терминальные символы, или просто терминалы, будут обозначаться малыми буквами латинского алфавита: а,b,с и т. д.;

**N** — конечное множество нетерминальных (вспомогательных) символов — вспомогательный алфавит. Нетерминальные символы, по-другому нетерминалы, — это понятия грамматики (языка), которые используются при его описании.

Нетерминалы будем обозначать заглавными латинскими буквами: А, В, C,D,E и т. д.;

**Р** — конечное множество правил вывода, называемых также продукциями. Каждое правило множества **Р** имеет вид:

**α** → **β**,

где **α** и **β** — цепочки терминальных и нетерминальных символов. Цепочка **α** не должна быть пустой, цепочка **β** может быть пуста:

**α** ∈ (**Т** ∪ **N**)+;

**β** ∈ (**Т** ∪ **N**)\*

Правило **α** → **β** определяет возможность подстановки **β** вместо **α** в процессе вывода (порождения) цепочек языка;

**S** (S ∈ N) — начальный символ грамматики — один из множества нетерминальных символов, начальный (стартовый) нетерминал. Начальный нетерминал — это понятие, соответствующее правильному предложению языка. Например, начальный нетерминал грамматики выражений обозначает «выражение», а начальный нетерминал грамматики языка Паскаль — «программа».

**Примеры грамматик. Порождение предложений языка**

***Пример 1***. Рассмотрим грамматику

G1 = ( {а, b}, {S}, {S → aSb, S → **ε** }, S).

Здесь все элементы четверки записаны явно. Множество терминальных символов Т = {а, b}; множество нетерминалов содержит один элемент: N = {S}, а множество правил — два: Р = {S →aSb, S → **ε** }; роль начального нетерминала исполняет S.

Грамматика может использоваться для порождения (вывода) цепочек — предложений языка. Процесс порождения начинается с начального нетерминала, в нашем примере это S. Если среди правил есть такое, в левой части которого записана цепочка S, то начальный нетерминал может быть заменен правой частью любого из таких правил. Оба правила грамматики G1 содержат в левой части S.

Применим подстановку, заданную первым правилом, заменив S на aSb:

S $→$ aSb.

К получившейся цепочке aSb снова, если удастся, можно применить одно из правил грамматики. Если в цепочке есть подцепочка, совпадающая с левой частью хотя бы одного из правил, то эту подцепочку можно заменить правой частью любого из таких правил. В цепочке aSb есть подцепочка S, совпадающая с левой частью обоих правил грамматики G1. Мы вправе применить любое из этих правил.

Используем снова правило (1) для продолжения вывода:

S$→$aSb$→$aaSbb.

Теперь к получившейся цепочке применим правило (2) {S → **ε**), заменив S пустой цепочкой. Получим такую последовательность подстановок (саму букву **ε** в последней цепочке записывать, конечно, не нужно):

S $→$aSb $→$ aaSbb $→$ aabb.

Очевидно, что к получившейся цепочке ни одно из правил грамматики G1 больше не применимо. Процесс порождения завершен.

 Нетрудно заметить, что с помощью грамматики G1 можно породить любую цепочку языка L1 = {anbn | n ≥ 0}, применив к начальному нетерминалу правило (1) (S → aSb) n раз, а затем один раз правило (2).

В то же время, грамматика G1 не порождает ни одной цепочки терминальных символов, не принадлежащей языку L1. To есть множество терминальных цепочек, порождаемых грамматикой G1 совпадает с языком L1. Другими словами, грамматика G1 порождает язык L1.

L(G1)=L1.

Обычно при записи грамматики не выписывают четверку ее элементов явно. При соблюдении соглашений об обозначениях терминалов и нетерминалов достаточно записать только правила. Правила с одинаковой левой частью можно объединять, отделяя альтернативные правые части вертикальной чертой. Первым записывается правило, содержащее в левой части начальный нетерминал. С учетом этого грамматика G1 может быть записана так:

G1: S →aSb | **ε** .

***Пример 2***. Рассмотрим грамматику G2 (цифры справа — номера правил)

G2: S→aSBc (1)

 S→abc (2)

 сВ → Bc (3)

 bВ → bb (4)

 S→ **ε** (5)

Проведем вывод цепочек из начального нетерминала грамматики G2. Под стрелкой, обозначающей подстановку, будем указывать, как и раньше, номер примененного правила. Итак:

S$→$aSBc$→$aabcBc$→$aabBcc$→$ ааbbсс.

Еще одна серия подстановок:

S$→$aSBc$→$aaSBcBc$→$aaabcBcBc$→$ aaabBccBc$→$aaabBcBcc$→$

aaabBBccc$→$aaabbBccc$→$ аааbbbссс.

Можно убедиться, что грамматика G2 порождает цепочки терминалов вида anbncn

и никакие другие. Количество повторений символов в результирующей цепочке определяется тем, на каком шаге применяется правило (2). Наличие правила (5)

позволяет получить пустую цепочку, если применить это правило первым, в то время как попытка использования этого правила на последующих шагах не позволит вывести цепочку терминалов. Грамматика G2 порождает множество терминальных цепочек, совпадающее с языком L2 = {anbncn | n ≥ 0}:

L(G2) = L2.

***Пример 3***. Грамматика, порождающая язык правильных скобочных выражений (язык Дика).

G3: S→(S) (1)

 S→SS (2)

 S→ **ε** (3)

Нетрудно понять логику построения правил этой грамматики. Смысл первого правила таков: заключив в скобки правильное скобочное выражение S, мы снова получим правильное скобочное выражение. Второе правило означает, что два правильных скобочных выражения, записанные одно за другим, дают новое правильное выражение. Наконец, по правилу (3) пустая цепочка считается правильным выражением. Если бы мы решили, что не следует разрешать пустые выражения, правило (3) можно было бы заменить на S → ().

***Пример 4***. Грамматика простых арифметических выражений. Единственным нетерминалом этой грамматики (он же начальный) будет *Выражение*:

G4:

*Выражение*→*Выражение*+*Выражение* |

*Выражение* - *Выражение* |

*Выражение*\* *Выражение* |

*Выражение* / *Выражение*

| а |b |с | ( Выражение ) .

Такая грамматика порождает цепочки терминалов, являющиеся правильными арифметическими выражениями. Символы а,b и с в таких выражениях обозначают операнды, а «+», «-», «\*», и «/» — знаки операций. Разрешаются круглые скобки (в том числе вложенные). Примеры правильных выражений:

(a+b)/(b\*c),

(а).

Все операции двуместные, унарные плюс и минус не предусмотрены, поэтому, например, цепочка -а не принадлежит языку, порождаемому этой грамматикой.

Выражение — одно из основных понятий языков программирования. В дальнейшем грамматикам выражений будет уделено немалое внимание. Чтобы запись этих грамматик была короче, заменим нетерминал Выражение на Е (от expression - выражение), вернувшись тем самым к принятым раньше соглашениям об именовании нетерминалов. Тогда грамматика, которую обозначим G5, запишется следующим образом:

G5: E→E + E|E-E|E\*E|E/E|a|b|c| (E).

Очевидно, что она порождает тот же язык, что и грамматика G4. А именно:

L(G5) = L5.