**Лекция Приведение грамматик**

Разбор предложений грамматики выполняется проще, если грамматика не содержит лишних правил. Можно с достаточным основанием утверждать, что если грамматика языка программирования содержит лишние правила, то она ошибочна. Следовательно, алгоритм, который обнаруживает лишние правила в грамматике, может оказаться полезным при проектировании языка.   
Правило, подобное U ::=U, очевидно, не является необходимым для грамматики; более того, оно приводит к неоднозначности грамматики. Поэтому далее полагается, что грамматика не содержит правил вида

(1) U ::=U

Кроме того предполагаем, что грамматика не должна содержать *цепных* правил, которые приводят к зацикливанию вывода. Например,

A::=B   
B::=C   
C::=A   
A,B,C -нетерминалы.

Грамматики могут также содержать лишние правила, которые невозможно использовать в выводе хотя бы одного предложения. Например, в следующей ниже грамматике G[4] нетерминал <d> не может быть использован в выводе какого – либо предложения, так как он не встречается ни в одной из правых частей правил:

Z ::= <b> e   
<a> ::= <a> e | e   
<b> ::= <c> e | <a> f   
<c> ::= <c> f <d> ::= f

Чтобы появиться в выводе какого–либо предложения, нетерминал U должен удовлетворять двум условиям. Во–первых, символ U должен встречаться в некоторой сентенциальной форме:

(2) Z \*xUy для некоторых цепочек x и y,

где Z - начальный символ грамматики. Во-вторых, из U должна выводиться цепочка t, состоящая из терминальных символов:

(3) U +t для некоторой t принадлежащей VT+ .

Ясно, что если нетерминальный символ U не удовлетворяет этим условиям, то правила, содержащие U в левой части, не могут быть использованы ни в каком выводе. С другой стороны, если все нетерминалы удовлетворяют этим условиям, то грамматика не содержит лишних правил.   
Так, если U ::=u – правило, то, во –первых, Z \*xUy  x u y. Во-вторых, так как мы можем вывести цепочку терминальных символов для каждого нетерминала, содержащего в цепочке x u y, то

x u y  \* t для некоторой t принадлежащей VT+ .

В конечном итоге мы получаем Z  +t, то есть вывод, в котором было использовано правило U ::=u.

Условие (2) проверяет нетерминалы на недостижимость. Нетерминалы, которые не появляются ни в одной цепочке, выводимой из начального символа, называются *недостижимыми нетерминалами*.

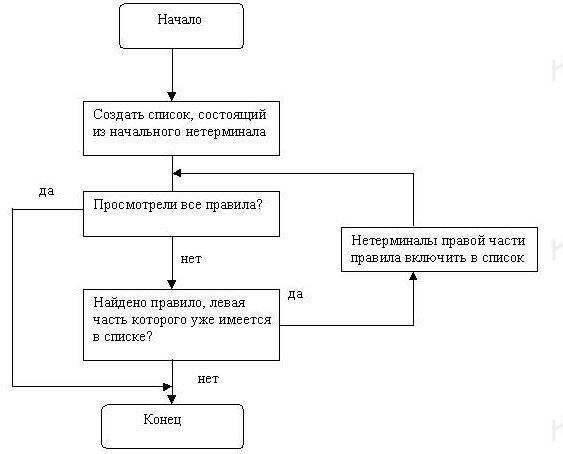
Условие (3) проверяет нетерминалы на бесплодность. Нетерминалы, которые не порождают ни одной терминальной цепочки, называются *бесплодными* или *мертвыми*.

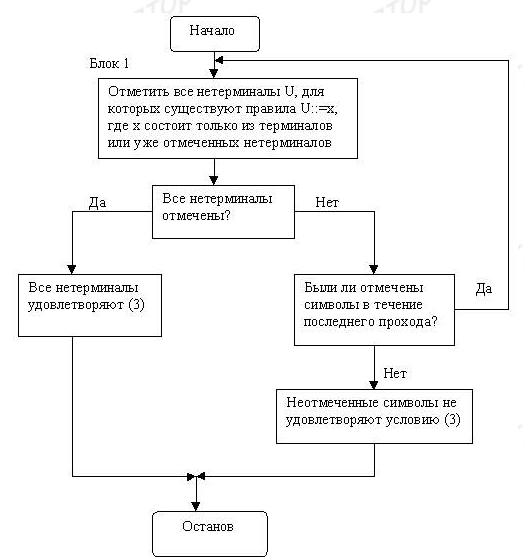
Нетерминалы, которые бесплодны или недостижимы, называются *лишними (бесполезными).*

В описанной выше грамматике G4 нетерминальный символ <c> не удовлетворяет условию (3), так как при непосредственном выводе он должен быть заменен на <c> f. Такая замена не может привести к цепочке из терминальных символов. Если отбросить все правила, которые нельзя использовать при порождении хотя бы одного предложения, то останутся правила Z ::= <b> e, <a> ::= <a> e, <a> ::= e, <b> ::= <a> f .

Определение. Грамматика G называется *приведенной*, если каждый нетерминал U удовлетворяет условиям (2) и (3), то есть не содержит лишних нетерминалов.

Нетерминал U удовлетворяет условию 2, тогда и только тогда, когда Z WITHIN+ U, где WITHIN+ является транзитивным замыканием отношения WITHIN (внутри). U WITHIN S тогда и только тогда, когда существует правило U ::=...S...   
Другими словами, если нетерминал в левой части правила является *достижимым*, то достижимы и все символы правой части этого правила. Это свойство выполняется, так как можно сначала вывести цепочку, содержащую символ, который является левой частью правила, и потом применить к ней это правило. Блок схема алгоритма, проверяющего символы на достижимость, приведена на рис.1. Список, полученный в результате работы алгоритма, содержит только достижимые нетерминалы, а все нетерминалы, не попавшие в этот список являются недостижимыми.

На рис.2 приведена блок схема алгоритма, который проверяет нетерминалы, встречающиеся в наборе правил на продуктивность (проверка условия (3)). Терминальный или нетерминальный символ называется *продуктивным*, если из него выводится какая-нибудь терминальная цепочка (то есть он не является бесплодным терминалом). Работа алгоритма начинается с того, что некоторым образом отмечаются те нетерминалы, для которых существует правило U ::= t , где t принадлежит VT+ (первое выполнение блока 1). Такие нетерминалы, очевидно удовлетворяют условию (3). Далее в алгоритме проверяются все неотмеченные нетерминалы U и для каждого из них делается попытка найти правило U ::= x, где x состоит только из терминальных символов или ранее отмеченных нетерминалов (повторное выполнение блока 1). Символы, для которых такое правило существует, также, очевидно удовлетворяют условию (3). Этот процесс продолжается до тех пор, пока либо все нетерминалы не будут отмечены, либо пока при выполнении блока 1 не будет отмечено ни одного символа. Работа алгоритма завершена. Неотмеченные нетерминалы не удовлетворяют условию (3).   
  
рис.1 Проверка нетерминалов на достижимость.

  
рис.2 Проверка символов на продуктивность.