Классификация грамматик. Вывод в грамматиках

Классификация грамматик

Аврам Ноам Хомский (Avram Noam Chomsky , 1928) – американский лингвист, политический публицист, философ и теоретик, профессор лингвистики Массачусетского технологического института, автор классификации формальных языков, называемой иерархией Хомского.

Включающая иерархия языков Хомского :

- *0-грамматики – Грамматики без ограничений (грамматики с фразовой структурой)*

  $\alpha ::= \beta,$

  где $\alpha \in \mathbf{V}^{+}$, $\beta \in \mathbf{V}^{*}$

Это самый общий тип грамматик. В него попадают все без исключения формальные грамматики, но часть из них может быть также отнесена и к другим классификационным типам. Грамматики, которые относятся только к типу 0 и не могут быть отнесены к другим типам, являются самыми сложными по структуре. Практического применения грамматики, относящиеся только к типу 0, не имеют.

Классификация грамматик

- *1-грамматики – Контекстно-зависимые* (неукорачивающие) грамматики
  - *Контекстно-зависимые грамматики*:

    $\alpha_1 A \alpha_2 ::= \alpha_1 \beta \alpha_2$,

    где $\alpha_1, \alpha_2 \in \mathbf{V}^*$, $A \in \mathbf{V_N}$, $\beta \in \mathbf{V}^+$.
  - *Неукорачивающие грамматики :*

    $\alpha ::= \beta$,

    где $\alpha, \beta \in \mathbf{V}^+$, $|\beta| \geq |\alpha|$.

Эти два класса грамматик эквивалентны. Это значит, что для любого языка, заданного КЗ-грамматикой, можно построить неукорачивающую грамматику, которая будет задавать эквивалентный язык, и, наоборот, для любого языка, заданного неукорачивающей грамматикой, можно построить КЗ-грамматику, которая будет задавать эквивалентный язык.

Классификация грамматик

- *2-грамматики – Контекстно-свободные* грамматики
  - *Неукорачивающие контекстно-свободные* (НКС) грамматики

    $A ::= \beta$,

    где $A \in \mathbf{V_N}$, $\beta \in \mathbf{V^+}$
  - *Укорачивающие контекстно-свободные* (УКС) грамматики

    $A ::= \beta$,

    где $A \in \mathbf{V_N}$, $\beta \in \mathbf{V^*}$

Эти два класса, составляющие тип контекстно-свободных грамматик, почти эквивалентны. Разница между ними заключается лишь в том, что в УКС-грамматиках в правой части правил может присутствовать пустая цепочка, а в НКС-грамматиках – нет. Отсюда ясно, что язык, заданный НКС-грамматикой, не может содержать пустой цепочки.

Классификация грамматик

- *3-грамматики – Регулярные грамматики*
  - *Леволинейные грамматики*

    $A ::= B\gamma$ или $A ::= \gamma$,

    где $A, B \in \mathbf{V_N}$, $\gamma \in \mathbf{V_T}^*$.
  - *Праволинейные грамматики*

    $A ::= \gamma B$ или $A ::= \gamma$,

    где $A, B \in \mathbf{V_N}$, $\gamma \in \mathbf{V_T}^*$.

Частным случаем регулярных грамматик являются автоматные грамматики, в которых $\gamma \in \mathbf{V_T}$, т.е. правила имеют вид:

$A ::= Bx$ или $A ::= x$ для левосторонних грамматик,

$A ::= xB$ или $A ::= x$ для правосторонних грамматик,

где $A, B \in \mathbf{V_N}$, $x \in \mathbf{V_T}$.

Классификация языков

Языки классифицируются в соответствии с типами грамматик, с помощью которых они заданы.

Один и тот же язык в общем случае может быть задан сколь угодно большим количеством грамматик, которые могут относиться к различным классификационным типам.

Для классификации самого языка среди всех его грамматик выбирается грамматика с максимально возможным классификационным типом.

*Например, если язык **L** может быть задан грамматиками G_1 и G_2, относящимися к типу 1 (КЗ), грамматикой G_3, относящейся к типу 2 (КС), и грамматикой G_4, относящейся к типу 3 (регулярные), сам язык должен быть отнесён к типу 3 и является регулярным языком.*

Пример языка 1

$L(G) = \{a^2 b^{n2-1} \mid n >= 1\}$

G: S ::= aaCFD

F ::= AFB | AB

AB ::= bBA

Ab ::= bA

AD ::= D

Cb ::= bC

CB ::= C

bCD ::= λ

- *Регулярная грамматика?*

Нет

AB ::= b**BA**

- *КС-грамматика?*

Нет

AB ::= bBA

- *КЗ/Неукорчивающая грамматика?*

Нет

bCD ::= **λ**

Вывод – тип 0, грамматика без ограничений

Пример языка 2

L(G) = { a^n b^n c^n, n >= 1}

G: S ::= aSBC | abC

CB ::= BC

bB ::= bb

bC ::= bc

cC ::= cc

- *Регулярная грамматика?*

  Нет

  bB ::= bb

- *КС-грамматика?*

  Нет

  bB ::= bb

- *КЗ/Неукорчивающая грамматика?*

  Да

Вывод – тип 1, неукорачивающая грамматика

Пример языка 3

$L(G) = \{(ac)^n (cb)^n \mid n > 0\}$

G: S ::= aQb | accb

Q ::= cSc

- *Регулярная грамматика?*

Нет

Q ::= cSc

- *КС-грамматика?*

Да

Вывод – тип 2, КС-грамматика

$L(G) = \{\beta \perp \mid \beta \in \{a,b\}^+$, где нет двух рядом стоящих a$\}$

G: S ::= A⊥ | B⊥

A ::= a | Ba

B ::= b | Bb | Ab

- *Регулярная грамматика?*

Да

- *Автоматная грамматика?*

Да

- *Лево- или правосторонняя грамматика?*

Вывод – тип 3, левосторонняя автоматная грамматика

Вывод в грамматике

Выводом называется процесс порождения предложения языка на основе правил определяющей язык грамматики.

Цепочка $\beta = \delta_1\gamma\delta_2$ **непосредственно выводима из цепочки** α, $\alpha = \delta_1\omega\delta_2$ в грамматике **G** ($\mathbf{V_T}$, $\mathbf{V_N}$, **P**, *S*) ($\alpha \Rightarrow \beta$), если в грамматике существует правило $\omega ::= \gamma$.

Цепочка β называется **выводимой** из цепочки α ($\alpha \Rightarrow^* \beta$) в случае, если выполняется одно из двух условий:

- β непосредственно выводима из α ($\alpha \Rightarrow \beta$);
- существует $\gamma_1, \ldots, \gamma_n$ такие, что

$$\alpha \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \ldots \Rightarrow \gamma_n \Rightarrow \beta.$$

Отношение вывода

С математической точки зрения вывод можно рассматривать как отношение на множестве V^*.

Оно обладает свойствами

- *рефлексивности*

$$\alpha \Rightarrow^* \alpha$$

- *транзитивности*

$$\text{если } \alpha \Rightarrow^* \gamma,\ \gamma \Rightarrow^* \beta, \text{ то } \alpha \Rightarrow^* \beta$$

Примеры вывода

Грамматика для языка целых десятичных чисел со знаком:

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

P: *S* ::= *T* | *+T* | *–T*

T ::= *F* | *TF*

F ::= 0 | I | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

1) $\boldsymbol{S} \Rightarrow \text{-}\boldsymbol{T} \Rightarrow \text{-}\boldsymbol{T}F \Rightarrow \text{-}\boldsymbol{T}FF \Rightarrow \text{-}\boldsymbol{F}FF \Rightarrow \text{-}4\boldsymbol{F}F \Rightarrow \text{-}47\boldsymbol{F} \Rightarrow \text{-}479$

2) $\boldsymbol{S} \Rightarrow \boldsymbol{T} \Rightarrow T\boldsymbol{F} \Rightarrow \boldsymbol{T}8 \Rightarrow \boldsymbol{F}8 \Rightarrow 18$

3) $\boldsymbol{T} \Rightarrow T\boldsymbol{F} \Rightarrow \boldsymbol{T}0 \Rightarrow T\boldsymbol{F}0 \Rightarrow \boldsymbol{T}50 \Rightarrow \boldsymbol{F}50 \Rightarrow 350$

4) $\boldsymbol{F} \Rightarrow 5$

Получаем, что: *S* ⇒* -479, *S* ⇒* 18, *T* ⇒* 350, *F* ⇒* 5.

Вывод и сентенциальная форма

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

Цепочка символов $\alpha \in \mathbf{V}^*$ называется **сентенциальной формой** грамматики **G** ($\mathbf{V_T}$, $\mathbf{V_N}$, **P**, *S*), если она выводима из целевого символа грамматики *S*:

$$S \Rightarrow^* \alpha.$$

Если цепочка $\alpha \in \mathbf{V_T}^*$ получена в результате законченного вывода, то она называется **конечной сентенциальной формой** или **предложением**.

Язык L, заданный грамматикой G — это множество всех конечных сентенциальных форм грамматики G.

Алфавит языка L(G) — это множество терминальных символов грамматики V_T, поскольку все конечные сентенциальные формы грамматики — это цепочки над алфавитом V_T.

Примеры сентенциальных форм

В грамматике для языка целых десятичных чисел со знаком:

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

P: *S* ::= *T* | *+T* | *–T*

T ::= *F* | *TF*

F ::= 0 | l | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Является сентенциальной формой?

- S
- +TF4
- ST6
- -3F6
- 89T
- 345

Левосторонний и правосторонний выводы

Вывод левосторонний, если на каждом шаге вывода правило грамматики применяется всегда к *крайнему левому* нетерминальному символу в цепочке.

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

В грамматике для одной и той же цепочки может быть несколько выводов, **эквивалентных** в том смысле, что в них в одних и тех же местах применяются одни и те же правила вывода, но в различном порядке.

Эквивалентные выводы

Для цепочки *a+b+a* в грамматике

G ({*a, b*, +}, {*S, T*}, **P**, *S*)

P: *S* ::= *T* | *T+S;*

T ::= *a* | *b*

можно построить выводы:

1) $\boldsymbol{S} \Rightarrow T+\underline{\boldsymbol{S}} \Rightarrow T+T+\underline{\boldsymbol{S}} \Rightarrow \underline{\boldsymbol{T}}+T+T \Rightarrow a+\underline{\boldsymbol{T}}+T \Rightarrow a+b+\underline{\boldsymbol{T}} \Rightarrow a+b+a$

2) $\boldsymbol{S} \Rightarrow \underline{\boldsymbol{T}}+S \Rightarrow a+\underline{\boldsymbol{S}} \Rightarrow a+\underline{\boldsymbol{T}}+S \Rightarrow a+b+\underline{\boldsymbol{S}} \Rightarrow a+b+\underline{\boldsymbol{T}} \Rightarrow a+b+a$

3) $\boldsymbol{S} \Rightarrow T+\underline{\boldsymbol{S}} \Rightarrow T+T+\underline{\boldsymbol{S}} \Rightarrow T+T+\underline{\boldsymbol{T}} \Rightarrow T+\underline{\boldsymbol{T}}+a \Rightarrow \underline{\boldsymbol{T}}+b+a \Rightarrow a+b+a$

Дерево вывода

Деревом вывода (синтаксическим деревом) грамматики G (V_T, V_N, P, *S*), называется дерево (граф), которое соответствует некоторой цепочке вывода и удовлетворяет следующим условиям:

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

Построение синтаксического дерева сверху вниз

Для построения дерева вывода, достаточно иметь только цепочку вывода.

Для строго формализованного построения дерева вывода всегда удобнее пользоваться строго определенным выводом: либо левосторонним, либо правосторонним.

1. Целевой символ грамматики помещается в корень дерева.
2. В грамматике выбирается необходимое правило, и на первом шаге вывода корневой символ раскрывается на несколько символов первого уровня.
3. Среди всех концевых вершин дерева выбирается крайняя (крайняя левая — для левостороннего вывода, крайняя правая — для правостороннего) вершина, обозначенная нетерминальным символом, для этой вершины выбирается нужное правило грамматики, и она раскрывается на несколько вершин следующего уровня.
4. Построение дерева заканчивается, когда все концевые вершины обозначены терминальными символами, в противном случае надо вернуться ко третьему шагу и продолжить построение.

Построение синтаксического дерева снизу вверх

Построение дерева вывода снизу вверх начинается с листьев дерева.

1. В качестве листьев выбираются терминальные символы конечной цепочки вывода, которые на первом шаге построения образуют последний уровень (слой) дерева.
2. Построение дерева идет по слоям. В грамматике выбирается правило, правая часть которого соответствует крайним символам в слое дерева (крайним правым символам при правостороннем выводе и крайним левым — при левостороннем).
3. Выбранные вершины слоя соединяются с новой вершиной, которая выбирается из левой части правила. Новая вершина попадает в слой дерева вместо выбранных вершин.
4. Построение дерева закончено, если достигнута корневая вершина (обозначенная целевым символом), а иначе надо вернуться ко второму шагу и повторить его над полученным слоем дерева.

Пример дерева вывода

Грамматика для языка целых десятичных чисел со знаком:

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

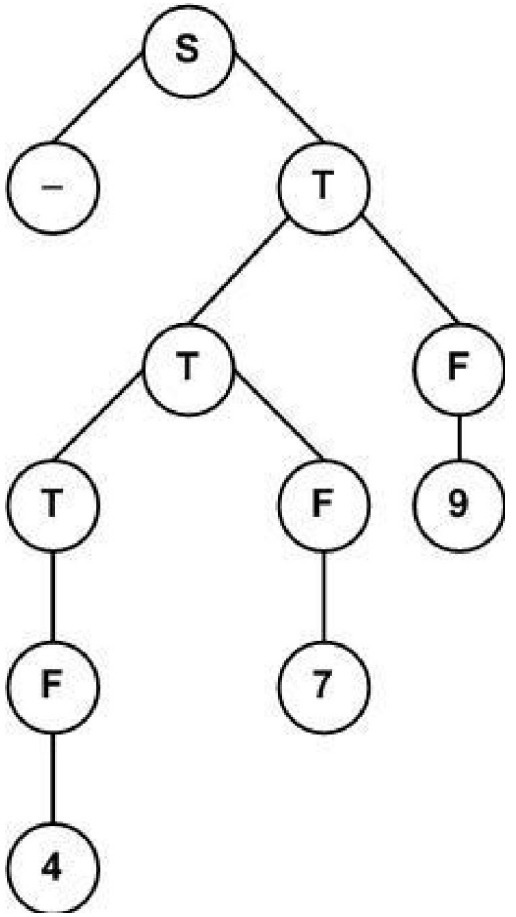
P: *S* ::= *T* | *+T* | *–T*

T ::= *F* | *TF*

F ::= 0 | I | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

1) $\boldsymbol{S} \Rightarrow -\boldsymbol{T} \Rightarrow -\boldsymbol{T}F \Rightarrow -\boldsymbol{T}FF \Rightarrow -\boldsymbol{F}FF \Rightarrow -4\boldsymbol{F}F \Rightarrow -47\boldsymbol{F} \Rightarrow -479$

2) $\boldsymbol{S} \Rightarrow -\boldsymbol{T} \Rightarrow -T\boldsymbol{F} \Rightarrow -\boldsymbol{T}9 \Rightarrow -T\boldsymbol{F}9 \Rightarrow -\boldsymbol{T}79 \Rightarrow -\boldsymbol{F}79 \Rightarrow -479$



Вывод в компиляторах

Поскольку все известные языки программирования имеют нотацию записи «слева — направо», компилятор также всегда читает входную программу слева направо (и сверху вниз, если программа разбита на несколько строк).

Поэтому для построения дерева вывода методом «сверху вниз», как правило, используется левосторонний вывод, а для построения «снизу вверх» — правосторонний вывод.

Нотация чтения программ «слева направо» влияет не только на порядок разбора программы компилятором (для пользователя это, как правило, не имеет значения), но и на порядок выполнения операций — при отсутствии скобок большинство равноправных операций выполняется в порядке слева направо, а это уже имеет существенное значение.

Однозначные и неоднозначные грамматики

Рассмотрим грамматику G({+,—,*,/,(,),a,b},{S},**P**,S):

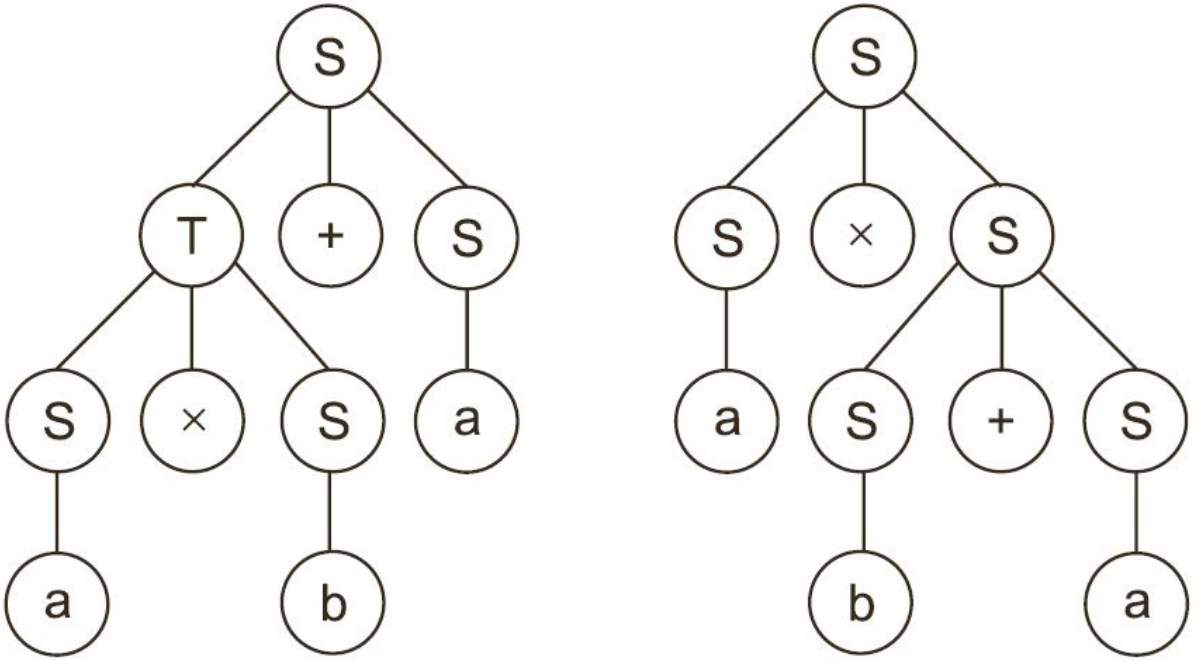
P: S ::= S+S | S—S | S*S | S/S | (S) | a | b

Построим левосторонний вывод для цепочки ***a*b+a.***

S ⇒ **S**+S ⇒ **S***S+S ⇒ a***S**+S ⇒ a*b+**S** ⇒ a*b+a,

S ⇒ **S***S ⇒ a***S** ⇒ a***S**+S ⇒ a*b+**S** ⇒ a*b+a.

Каждому из этих вариантов будет соответствовать свое дерево в



Однозначные и неоднозначные грамматики

Структура предложения и его значение (смысл) взаимосвязаны.
Дерево вывода (или цепочка вывода) является формой представления структуры предложения языка.
Для языков программирования, которые несут смысловую нагрузку, имеет принципиальное значение то, какая цепочка вывода будет построена для предложения языка.
Например, в примере языка арифметических выражений с точки зрения семантики порядок построения дерева вывода соответствует порядку выполнения арифметических действий.
При отсутствии скобок умножение всегда выполняется раньше сложения (умножение имеет более высокий приоритет), но в рассмотренной выше грамматике это ниоткуда не следует — в ней все операции равноправны.
С точки зрения арифметических операций приведенная грамматика имеет неверную семантику — в ней нет приоритета операций, а для равноправных операций не определен порядок выполнения (в арифметике принят порядок выполнения действий «слева направо»), хотя синтаксическая структура построенных с ее помощью выражений будет правильной.

Однозначные и неоднозначные грамматики

Грамматика называется ***однозначной***, если для каждой цепочки символов языка, заданного этой грамматикой, можно построить ***единственный*** левосторонний (и единственный правосторонний) вывод.

Или, что то же самое, грамматика называется однозначной, если для каждой цепочки символов языка, заданного этой грамматикой, существует ***единственное дерево вывода***.

В противном случае грамматика называется ***неоднозначной.***

Рассмотренная в примере грамматика арифметических выражений, очевидно, является неоднозначной.

Пример перехода от неоднозначной грамматики к однозначной

Если грамматика является неоднозначной, то необходимо попытаться преобразовать ее в однозначный вид. Для рассмотренной неоднозначной грамматики арифметических выражений над операндами a и b существует эквивалентная ей однозначная грамматика вида

G'({+,—,*,/,(,),a,b},{S,T,E},P',S):

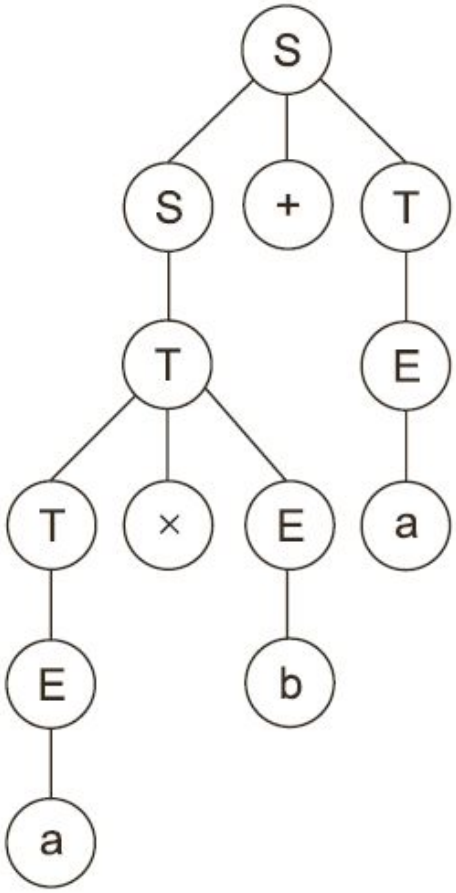
P': S ::= S+T | S - T | T

T ::= T*E | T/E | E

E ::= (S) | a | b

В этой грамматике для той же цепочки символов языка a*b+a возможен только один левосторонний вывод :

S ⇒ **S**+T ⇒ **T**+T ⇒ **T***E+T ⇒ **E***E+T ⇒ a***E**+T ⇒ a*b+**T** ⇒ a*b+**E** ⇒ a*b+a



Как проверить, является ли грамматика однозначной?

Для доказательства неоднозначности грамматики достаточно найти в заданном ею языке хотя бы одну цепочку, которая допускала бы более чем один левосторонний или правосторонний вывод. Но это не всегда легко.

Если такая цепочка не найдена, нельзя утверждать, что данная грамматика является однозначной, поскольку перебрать все цепочки бесконечного языка невозможно. Следовательно, нужны другие способы проверки однозначности грамматики.

Доказана **алгоритмическая неразрешимость проблемы однозначности** в общем виде. То есть доказано, что не существует (и никогда не будет существовать) алгоритм, который бы позволял проверить, является ли произвольно заданная грамматика G однозначной или нет. Аналогично, не существует алгоритма, который бы позволял преобразовать заведомо неоднозначную грамматику G в эквивалентную ей однозначную грамматику G'.

Как убедиться, что две грамматики эквивалентны?

Проблема эквивалентности грамматик в общем виде формулируется следующим образом: имеется две грамматики, G и G', необходимо построить алгоритм, который бы позволял проверить, являются ли эти две грамматики эквивалентными. То есть надо проверить утверждение L(G) = L(G').

Доказано, что ***проблема эквивалентности грамматик*** в общем случае алгоритмически неразрешима. Это значит, что не только до сих пор не существует алгоритма, который бы позволял проверить, являются ли две заданные грамматики эквивалентными, но и доказано, что такой алгоритм в принципе не существует, а значит, он никогда не будет создан.

Неразрешимость проблем эквивалентности и однозначности грамматик в общем случае совсем не означает, что они не разрешимы вообще. Для многих частных случаев — например, для определенных типов и классов грамматик (в частности для регулярных грамматик) — эти проблемы решены.

Правила, задающие неоднозначность в грамматиках

Для КС-грамматик существуют виды правил, по наличию которых во всем множестве правил грамматики G можно утверждать, что она является неоднозначной:

1. $A ::= AA \mid \alpha$.
2. $A ::= A\alpha A \mid \beta$.
3. $A ::= \alpha A \mid A\beta \mid \gamma$.
4. $A ::= \alpha A \mid \alpha A\beta A \mid \gamma$,

здесь $A \in \mathbf{V_N}$; $\alpha,\beta,\gamma \in (\mathbf{V_N} \cup \mathbf{V_T})^*$.

- Если в заданной грамматике встречается хотя бы одно правило подобного вида (любого из приведенных вариантов), то доказано, что такая грамматика точно будет **неоднозначной**.
- Отсутствие правил указанного вида (всех вариантов) — это необходимое, но **не достаточное** условие однозначности грамматики. Если подобных правил во всем множестве правил грамматики нет, такая грамматика может быть однозначной, а может и не быть.