Приведенные КС-грамматики

Символ $x \in (T \cup N)$ называется *недостижимым* в грамматике $G=(T, N, P, S)$, если он не появляется ни в одной сентенциальной форме этой грамматики.

Символ $A \in N$ называется *бесплодным* в грамматике $G=(T, N, P, S)$, если множество выводимых из этого символа терминальных цепочек пусто.

КС-грамматика называется приведенной, если в ней нет недостижимых и бесплодных символов.

Приведенные КС-грамматики

Алгоритм приведения грамматики:

1. Найти и удалить все бесплодные символы и правила, их содержащие.
2. Найти и удалить все недостижимые символы и правила, их содержащие.

Примечание. Если начальный символ грамматики окажется бесплодным, то следует удалить содержащие его правила, а сам символ оставить в алфавите нетерминалов N, так как, по определению грамматики, N обязан содержать начальный символ.

Для нахождения бесплодных и недостижимых символов полезен граф КС-грамматики:

- каждому символу из $T \cup N$ соответствует единственная вершина, помеченная этим символом; если в P есть правило с пустой правой частью ε, то граф имеет вершину, помеченную ε ;
- вершина X соединяется с вершиной Y стрелкой (дугой), если в грамматике есть правило $X \to \alpha Y \beta$, $\alpha, \beta \in (T \cup N)^*$;
- X соединяется с вершиной ε, если в грамматике есть правило $X \to \varepsilon$.

Алгоритм удаления бесплодных символов:

1. Отметить терминальные вершины (вершины, помеченные терминальными символами), а также вершину ε, если таковая имеется.

2. Если в P есть правило $A \to \alpha$, где α состоит из уже отмеченных в графе символов, а вершина A не отмечена, то отметить эту вершину. Повторять шаг 2 пока возможно.

3. Из грамматики удалить неотмеченные символы и правила, их содержащие.

Алгоритм удаления бесплодных символов:

1. Отметить терминальные вершины (вершины, помеченные терминальными символами), а также вершину ε, если такая имеется.
2. Если в P есть правило $A \to \alpha$, где α состоит из уже отмеченных в графе символов, а вершина A не отмечена, то отметить эту вершину. Повторять шаг 2 пока возможно.
3. Из грамматики удалить неотмеченные символы и правила, их содержащие.

Алгоритм удаления недостижимых символов:

1. Отметить вершины, в которые есть путь из вершины S (достижимы из вершины S).
2. Удалить из грамматики неотмеченные символы и правила, их содержащие.

Пример. Дана грамматика

G=({a, b, c, d, e}, {S, A, B, C, D}, P, S)

P: $S \to aAB \mid C$

$D \to cDc \mid d$

$C \to aCD$

$A \to aA \mid a \mid \varepsilon$

$B \to b$

Пример. Дана грамматика

G=({a, b, c, d, e}, {S, A, B, C, D}, P, S)

P: $S \to aAB \mid C$

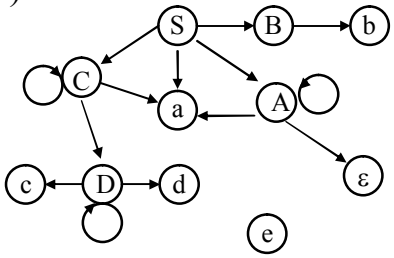
$D \to cDc \mid d$

$C \to aCD$

$A \to aA \mid a \mid \varepsilon$

$B \to b$

Граф грамматики G:



Пример. Дана грамматика

G=({a, b, c, d, e}, {S, A, B, ~~C~~, D}, P, S)

P: $S \to aAB \mid ~~C~~$

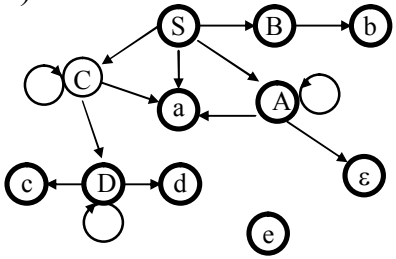
$D \to cDc \mid d$

~~$C \to aCD$~~

$A \to aA \mid a \mid \varepsilon$

$B \to b$

Граф грамматики G:



Не отмеченные жирным кружком символы бесплодны.

Удалив из G бесплодные символы, получим эквивалентную грамматику

G_1 =({a, b, c, d, e}, {S, A, B, D}, P_1 , S)

P_1 : $S \to aAB$

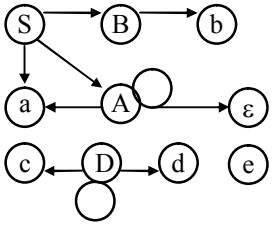
$D \to cDc \mid d$

$A \to aA \mid a \mid \varepsilon$

$B \to b$

G_1 не содержит бесплодных символов.

Граф грамматики G_1 :



Находим недостижимые символы

$G_1 = (\{a, b, c, d, e\}, \{S, A, B, D\}, P_1, S)$

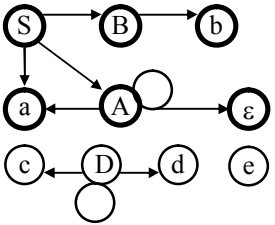
P_1 : $S \to aAB$

$D \to cDc \mid d$

$A \to aA \mid a \mid \varepsilon$

$B \to b$

Граф грамматики G_1 :



Здесь неотмеченные символы являются недостижимыми.

$G_1 = (\{a, b, \sout{c}, \sout{d}, \sout{e}\}, \{S, A, B, \sout{D}\}, P_1, S)$

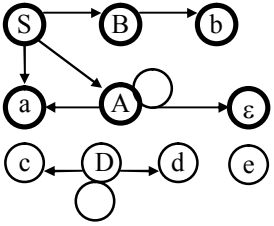
P_1 : $S \to aAB$

~~$D \to cDc \mid d$~~

$A \to aA \mid a \mid \varepsilon$

$B \to b$

Граф грамматики G_1 :



Здесь неотмеченные символы являются недостижимыми.

Удалив из G_1 недостижимые символы, получим эквивалентную грамматику:

$G_2 = (\{a, b\}, \{A, B\}, P_2, S)$

P_2 : $S \to aAB$ $\qquad$ G_2 – приведенная грамматика

$A \to aA \mid a \mid \varepsilon$

$B \to b$ $\qquad$ $L(G)=L(G_1)=L(G_2)=\{ a^n b \mid n \geq 1 \}$

Задача. Убедиться, что если в рассмотренном выше примере поменять местами шаги (1) и (2) алгоритма приведения грамматики, то результатом будет неприведенная грамматика.

Устранение правил с пустой правой частью из КС-грамматики

1. Построить множество $X=\{A\in N \mid A\Rightarrow\varepsilon\}$.
2. Удалить правила с пустой правой частью.
3. Если $S\in X$, то S' – новый начальный символ, $S'\to S \mid \varepsilon \in P$.
4. $\forall\ A\in X$ правило вида $B\to\alpha_1A_1\alpha_2A_2...\alpha_nA_n\alpha_{n+1}$,

   где $\alpha_i\in((N-X)\cup T)^*$

заменить 2^n правилами, соответствующими всем возможным комбинациям вхождений A между α_i:

$B\to\alpha_1\alpha_2...\alpha_n\alpha_{n+1}$

$B\to\alpha_1\alpha_2...\alpha_nA_n\alpha_{n+1}$

...

$B\to\alpha_1\alpha_2A_2...\alpha_nA_n\alpha_{n+1}$

$B\to\alpha_1A_1\alpha_2A_2...\alpha_nA_n\alpha_{n+1}$

Замечание: если все $\alpha_i=\varepsilon\ \ \forall\ i=1,...,n+1$, то правило $B\to\varepsilon$ не включать в новую грамматику.

5. Удалить бесполезные символы и правила, их содержащие.

Пример.

	S→BC | Ab		S→C | b | Ab
исходная	B→ε	эквивалентная	C→c
грамматика	C→c	грамматика	A→Aa | a
с ε-правилами	A→Aa | ε	после применения алгоритма	

Мы рассмотрели основные понятия теории формальных языков.

ТФЯ является одной из старейших фундаментальных областей информатики, ее результаты используются при программировании, например, в теории трансляции языков программирования, в некоторых областях математики, лингвистики, биологии.

Основы трансляции

- задача разбора
- лексический анализ
- синтаксический анализ
- семантические действия
- формальный перевод
- генерация кода
- интерпретация

***Задача разбора**:*

Даны КС-грамматика G и цепочка x.

$x \in L(G)$?

Если да, то построить дерево вывода для x

(или левый вывод для x, или правый вывод для x).

***Задача распознавания**: $x \in L(G)$? Дерево или вывод не требуются в качестве ответа, только ответ - «да» или «нет».*

Задача распознавания алгоритмически неразрешима в классе языков типа 0;

разрешима в классе языков типа 1.

Для КС-языков и регулярных языков существуют эффективные алгоритмы разбора.

Регулярные и КС-языки используются при описании синтаксиса языков программирования

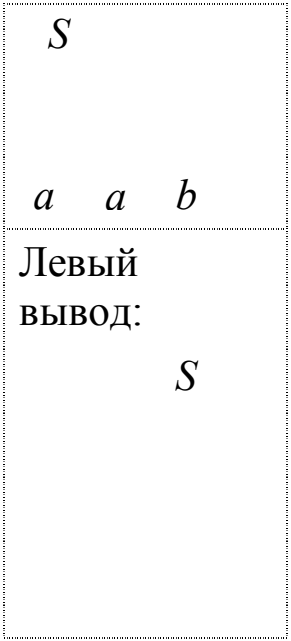
Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

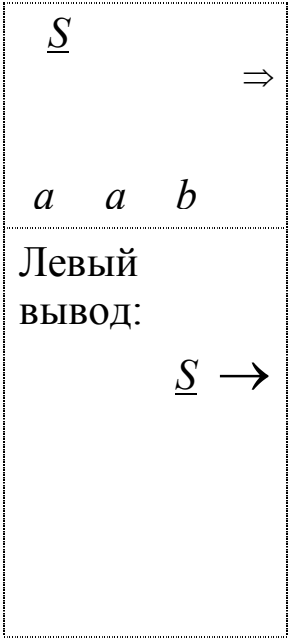
Сверху вниз:



Построение дерева вывода

G: $S \to a\,S \mid b$ Цепочка: aab

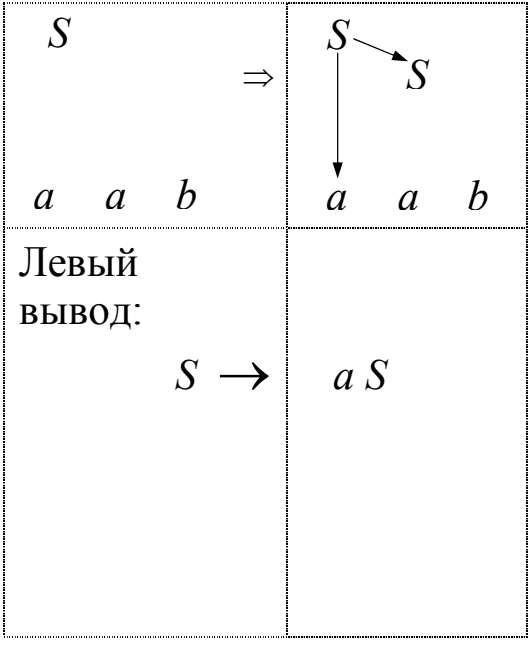
Сверху вниз:



Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

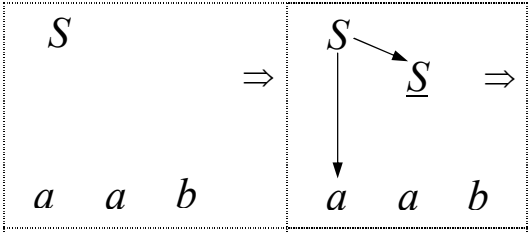
Сверху вниз:



Построение дерева вывода

G: $S \to a\,S \mid b$ Цепочка: aab

Сверху вниз:



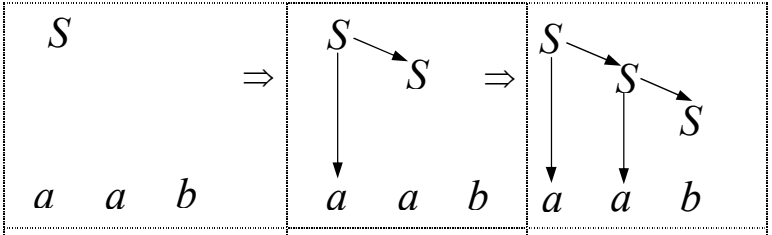
Левый вывод:

$S \to$ $a\,\underline{S} \to$

Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

Сверху вниз:



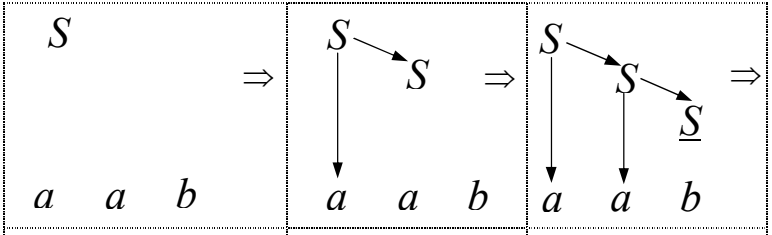
Левый вывод:

$S \to$ $a\ S \to$ $a\ a\ S$

Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

Сверху вниз:



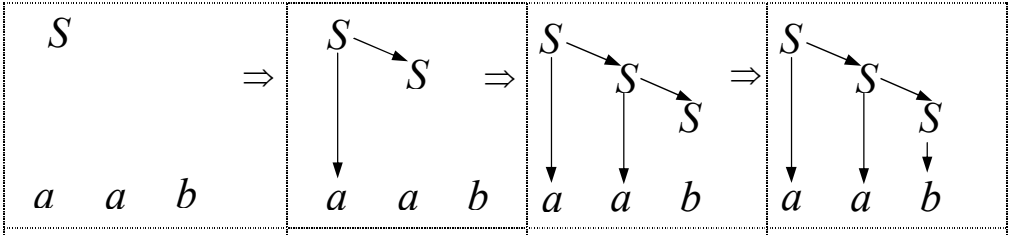
Левый вывод:

$S \to$ $a\ S \to$ $a\ a\ \underline{S} \to$

Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

Сверху вниз:



Левый вывод:

$S \to$ $a\ S \to$ $a\ a\ S \to$ $a\ a\ b$

Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

Свертка – это применение правила вывода «в обратную сторону», замена правой части на нетерминал из левой части:

Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

Свертка – это применение правила вывода «в обратную сторону», замена правой части на нетерминал из левой части:

$aa\underline{b} \leftarrow aaS$ — *свертка* по правилу $S \to b$. Обозначаем свертку с помощью обратной стрелки $\leftarrow$.

Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

Свертка – это применение правила вывода «в обратную сторону», замена правой части на нетерминал из левой части:

$aa\underline{b} \leftarrow aaS$ — *свертка* по правилу $S \to b$. Обозначаем свертку с помощью обратной стрелки $\leftarrow$.

С помощью сверток можно построить вывод «задом наперед» (обращение вывода): от цепочки к цели грамматики S. Например, сентенциальную форму aaS можно свернуть к aS, а затем к S :

Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

Свертка – это применение правила вывода «в обратную сторону», замена правой части на нетерминал из левой части:

$aa\underline{b} \leftarrow aaS$ — *свертка* по правилу $S \to b$. Обозначаем свертку с помощью обратной стрелки $\leftarrow$.

С помощью сверток можно построить вывод «задом наперед» (обращение вывода): от цепочки к цели грамматики S. Например, сентенциальную форму aaS можно свернуть к aS, а затем к S :

Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

Свертка – это применение правила вывода «в обратную сторону», замена правой части на нетерминал из левой части:

$aa\underline{b} \leftarrow aaS$ — *свертка* по правилу $S \to b$. Обозначаем свертку с помощью обратной стрелки $\leftarrow$.

С помощью сверток можно построить вывод «задом наперед» (обращение вывода): от цепочки к цели грамматики S. Например, сентенциальную форму aaS можно свернуть к aS, а затем к S :

aab

Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

Свертка – это применение правила вывода «в обратную сторону», замена правой части на нетерминал из левой части:

$aa\underline{b} \leftarrow aaS$ — *свертка* по правилу $S \to b$. Обозначаем свертку с помощью обратной стрелки $\leftarrow$.

С помощью сверток можно построить вывод «задом наперед» (обращение вывода): от цепочки к цели грамматики S. Например, сентенциальную форму aaS можно свернуть к aS, а затем к S :

$aab \leftarrow aaS$

Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

Свертка – это применение правила вывода «в обратную сторону», замена правой части на нетерминал из левой части:

$aa\underline{b} \leftarrow aaS$ — *свертка* по правилу $S \to b$. Обозначаем свертку с помощью обратной стрелки $\leftarrow$.

С помощью сверток можно построить вывод «задом наперед» (обращение вывода): от цепочки к цели грамматики S. Например, сентенциальную форму aaS можно свернуть к aS, а затем к S :

$aab \leftarrow aaS \leftarrow aS$

Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

Свертка – это применение правила вывода «в обратную сторону», замена правой части на нетерминал из левой части:

$aa\underline{b} \leftarrow aaS$ — *свертка* по правилу $S \to b$. Обозначаем свертку с помощью обратной стрелки $\leftarrow$.

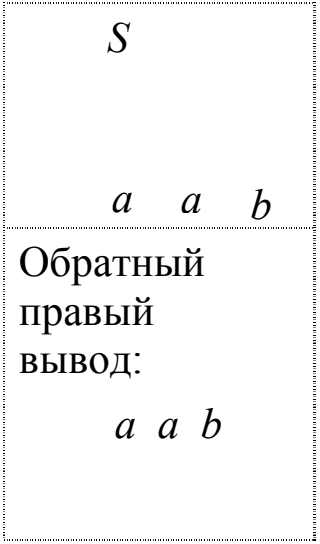
С помощью сверток можно построить вывод «задом наперед» (обращение вывода): от цепочки к цели грамматики S. Например, сентенциальную форму aaS можно свернуть к aS, а затем к S :

$aab \leftarrow aaS \leftarrow aS \leftarrow S$

Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

Снизу вверх:



Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

Снизу вверх:

S $\Rightarrow$

$a \quad a \quad \underline{b}$

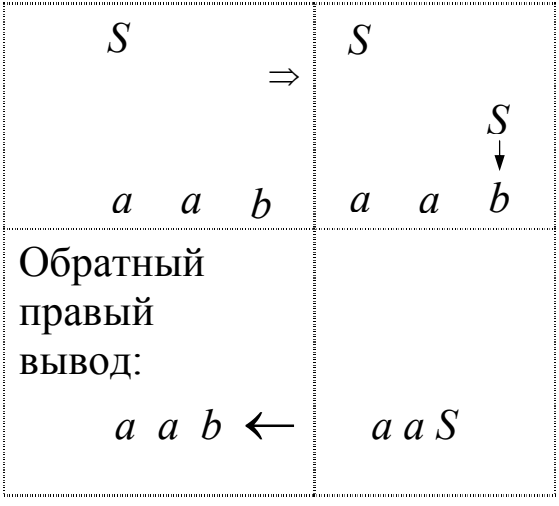
Обратный правый вывод:

$a \quad a \quad \underline{b} \leftarrow$

Построение дерева вывода

G: $S \to a\,S \mid b$ Цепочка: aab

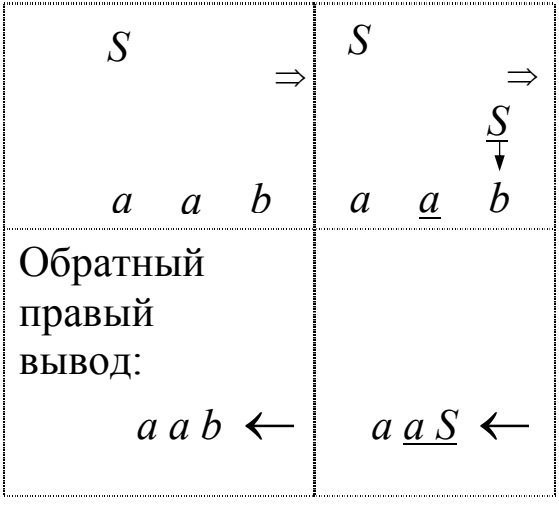
Снизу вверх:



Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

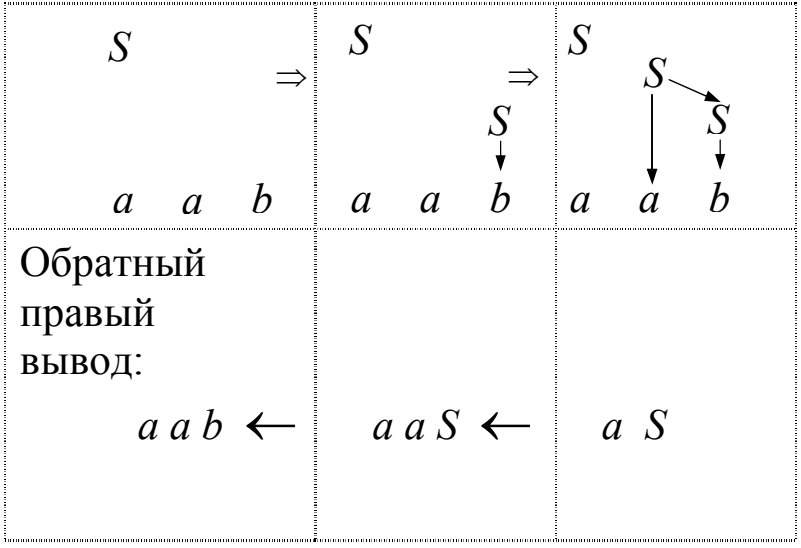
Снизу вверх:



Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

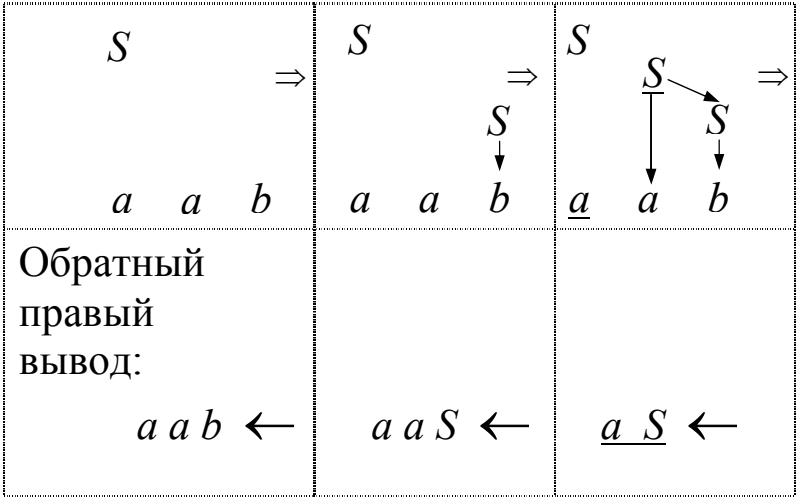
Снизу вверх:



Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

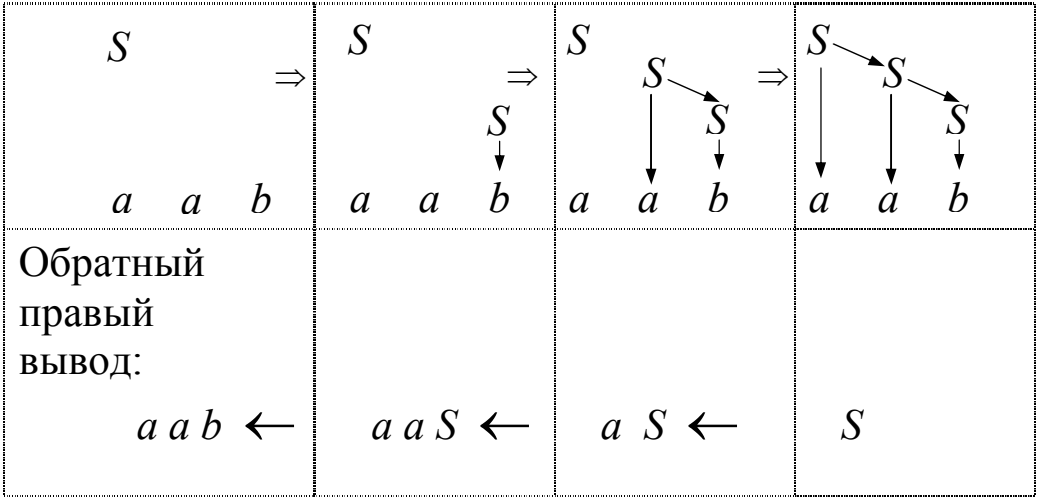
Снизу вверх:



Построение дерева вывода

G: $S \to a\ S \mid b$ Цепочка: aab

Снизу вверх:



РЕГУЛЯРНЫЕ ЯЗЫКИ

способы описания:

-- регулярные грамматики

(леволинейные либо праволинейные)

-- конечные автоматы

(недетерминированные или детерминированные)

-- регулярные выражения

Недетерминированный конечный автомат (НКА) – это пятерка $\mathcal{A} = (K, \Sigma, \delta, I, F)$, где:

K — конечное множество состояний, или вершин;
Σ — входной алфавит (также конечный);
$\delta \subseteq K \times \Sigma \times K$ — множество команд, или дуг;
$I \subseteq K$ — множество начальных состояний;
$F \subseteq K$ — множество заключительных состояний.

Множество δ можно также интерпретировать как отображение $K \times \Sigma$ в множество подмножеств K.

$\mathcal{A} = (K, \Sigma, \delta, I, F)$ — НКА.

Каждая дуга НКА $\mathcal{A}$ имеет пометку из Σ.

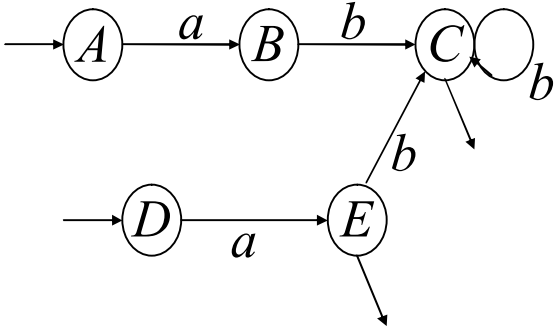
Путь в ориентированном графе может быть представлен последовательностью дуг. *Пустой* путь можно представить одной вершиной, которая считается одновременно началом и концом пути.

Пометка пути — это сцепление (конкатенация) пометок его дуг. Пустой путь имеет пустую пометку. Путь из начальной вершины в заключительную называется *успешным*.

Язык, допускаемый автоматом $\mathcal{A}$ (обозначается $L(\mathcal{A})$), – это множество пометок всех успешных путей автомата.

Пример 1. $\mathcal{A}_1 = (\{A, B, C, D, E\}, \{a, b\}, \delta, \{A, D\}, \{C, E\})$, где $\delta = \{(A, a, B), (D, a, E), (B, b, C), (E, b, C), (C, b, C)\}$.

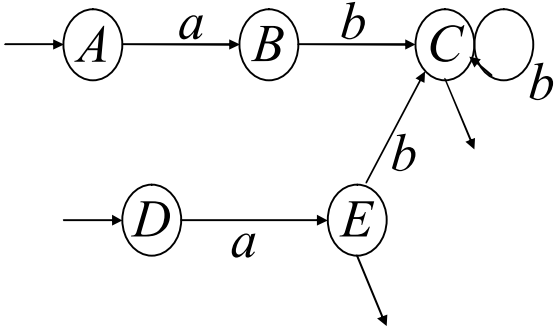
Автомат удобно представлять в виде ориентированного размеченного графа:



Входящими непомеченными стрелками отмечены начальные вершины A и D, исходящими – заключительные вершины E и C. $L(\mathcal{A}_1) =$?

Пример 1. $\mathcal{A}_1 = (\{A, B, C, D, E\}, \{a, b\}, \delta, \{A, D\}, \{C, E\})$, где $\delta = \{(A, a, B), (D, a, E), (B, b, C), (E, b, C), (C, b, C)\}$.

Автомат удобно представлять в виде ориентированного размеченного графа:



Входящими непомеченными стрелками отмечены начальные вершины A и D, исходящими – заключительные вершины E и C. $L(\mathcal{A}_1) = \{ab^n \mid n \geq 0\}$

Вопросы и задачи

1. Показать, что что если в рассмотренном выше примере поменять местами шаги (1) и (2) алгоритма приведения грамматики (грамматика на слайде 51), то результатом будет неприведенная грамматика.
2. Построить приведенную КС- грамматику, эквивалентную данной:

   $S \to aA \mid C$

   $D \to aD \mid DD \mid C$

   $C \to D \mid cC$

   $B \to S$

   $A \to aA \mid a$
3. Построить неукорачивающую КС-грамматику, эквивалентную данной:

   $S \to aS \mid CD \mid Sc$

   $D \to aD \mid D \mid \varepsilon$

   $C \to Cc \mid \varepsilon$
4. Построить конечный автомат, допускающий язык всех цепочек в алфавите {a,b}, в которых количество символов a четно, а количество символов b – нечетно.