Элементы теории трансляции

Транслятор

позволяет преобразовать программу, написанную на ЯП, отличном от машинного языка, к виду, допускающему выполнение на ЭВМ.

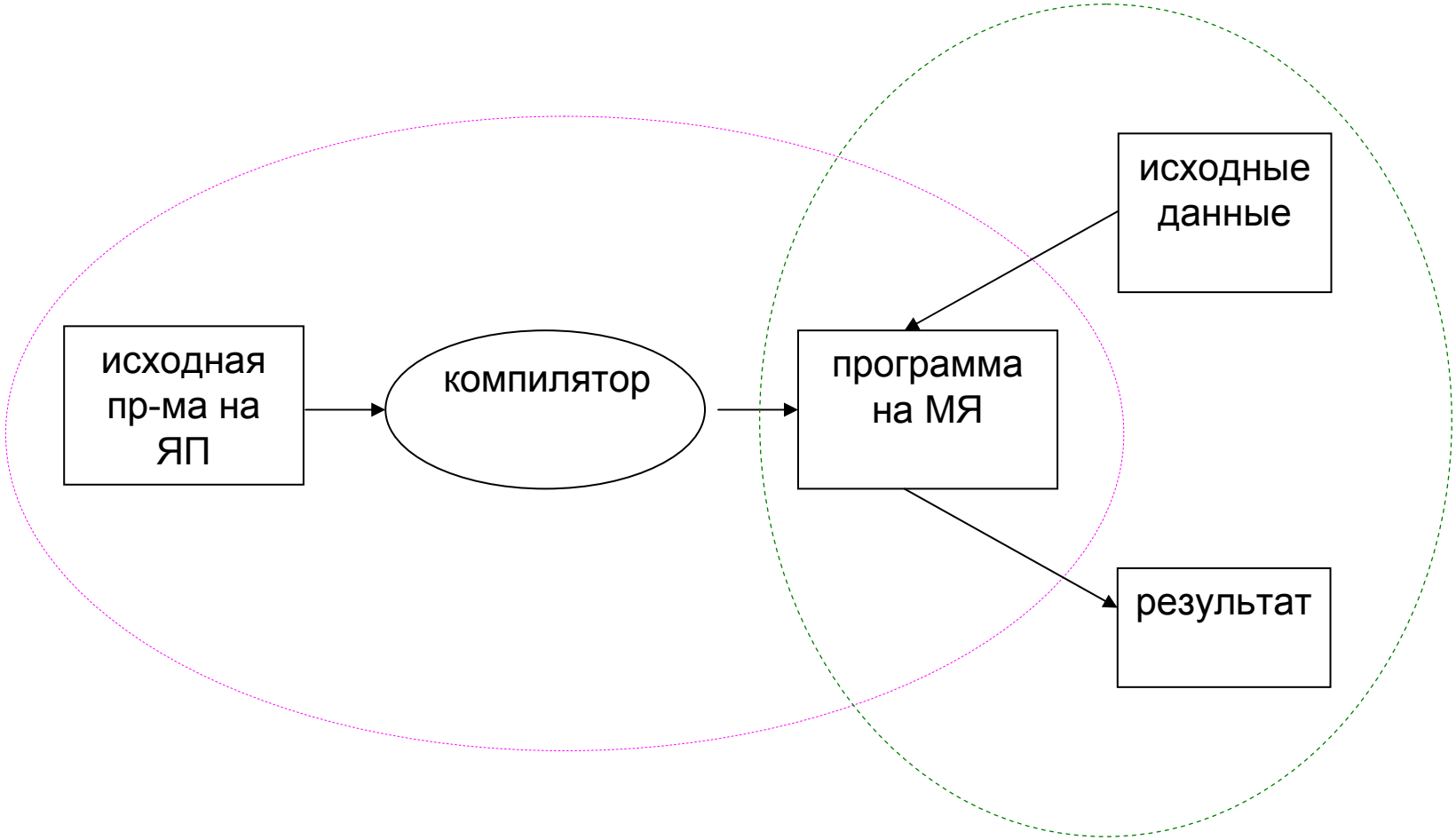
Компилятор

на вход получает программу на некотором ЯП (немашинном), а на выходе выдает объектный модуль (программу на машинном языке).

Интерпретатор

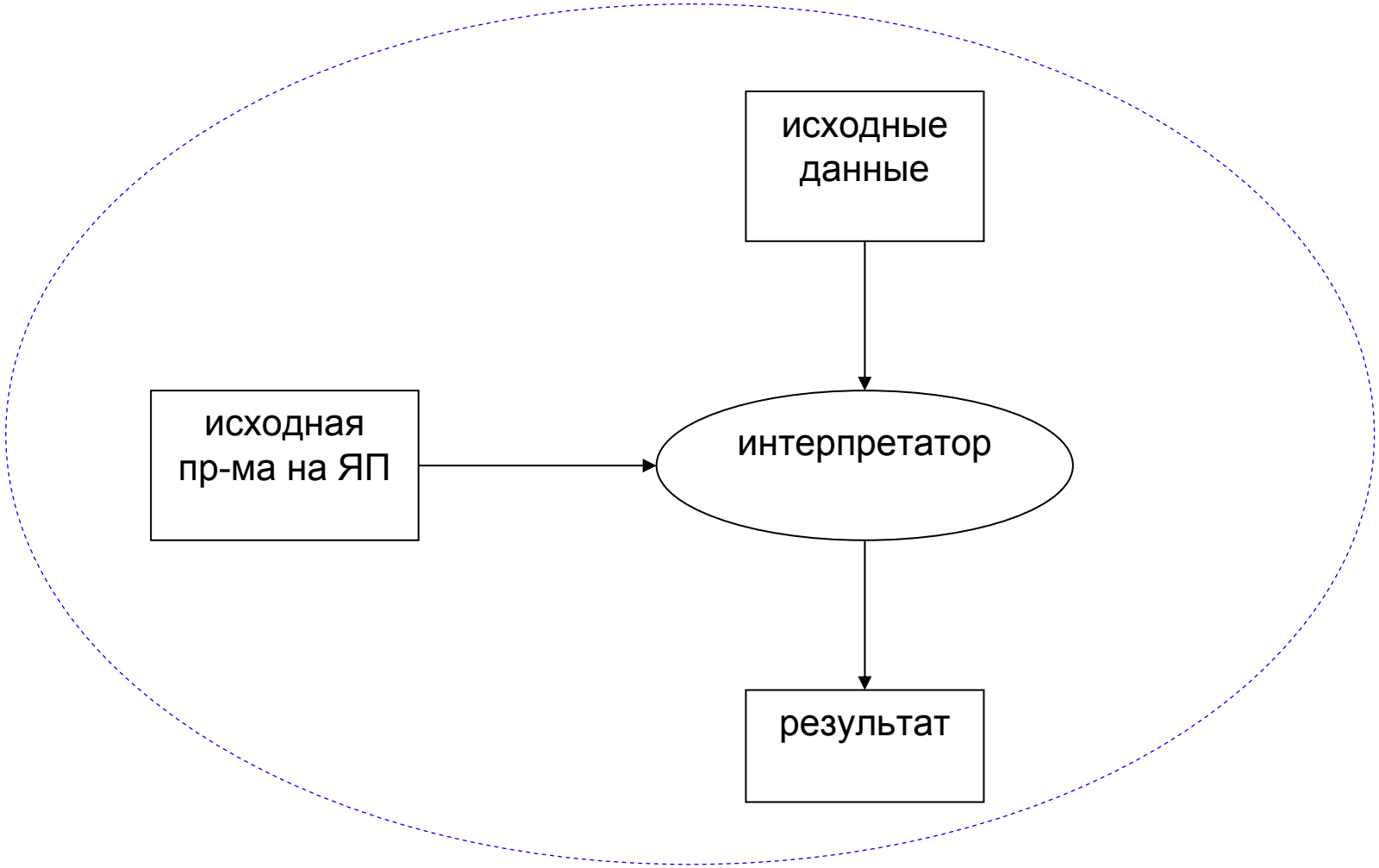
на вход получает программу на некотором ЯП (немашинном) и, считывая предложение за предложением исходной программы, анализирует их и тут же выполняет действия, указанные в этих предложениях.

Система программирования компилирующего типа



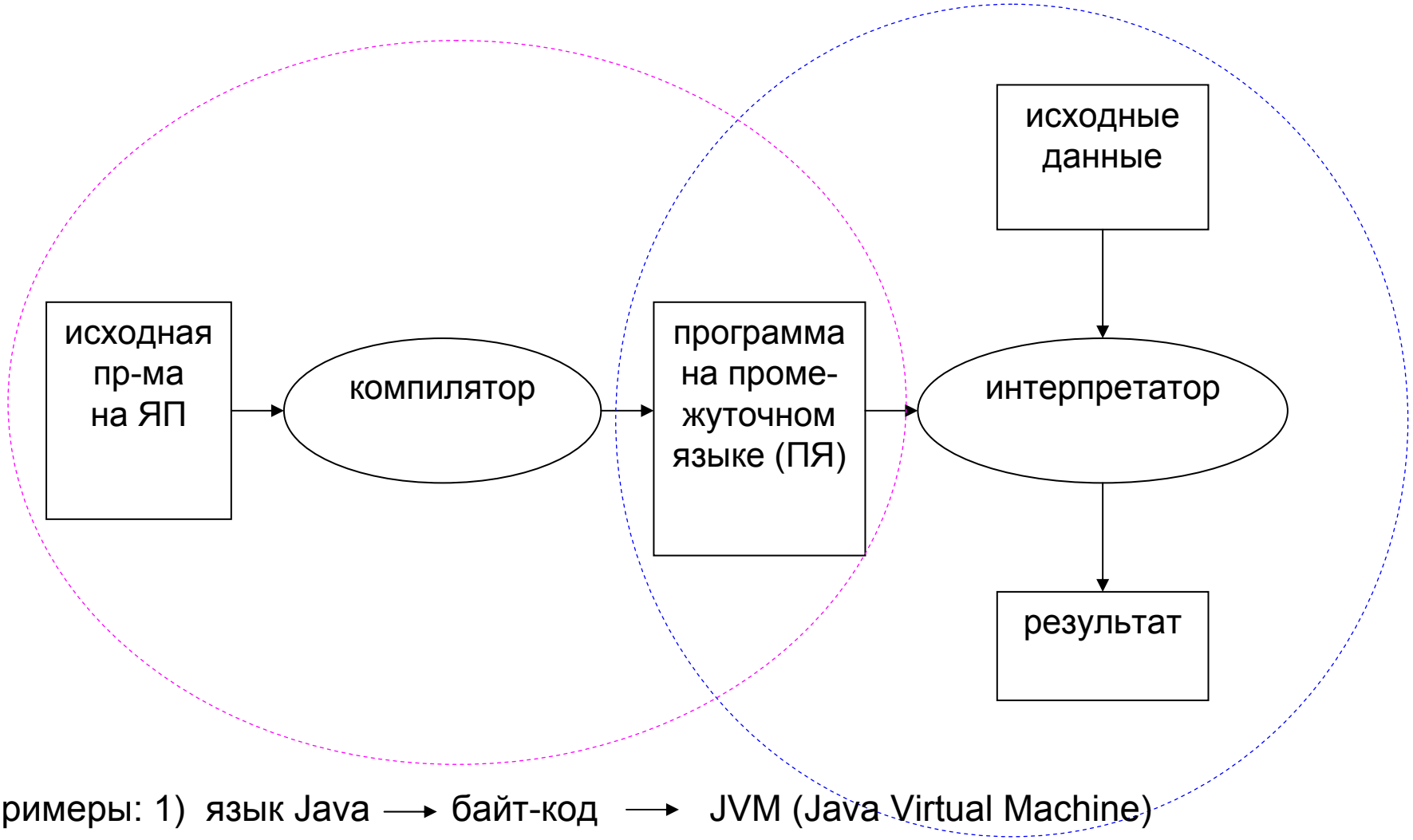
Примеры: MASM, Турбо Паскаль, gcc в UNIX

Система программирования интерпретирующего типа



Примеры: QBasic, командные интерпретаторы в UNIX -- bash, csh и др.

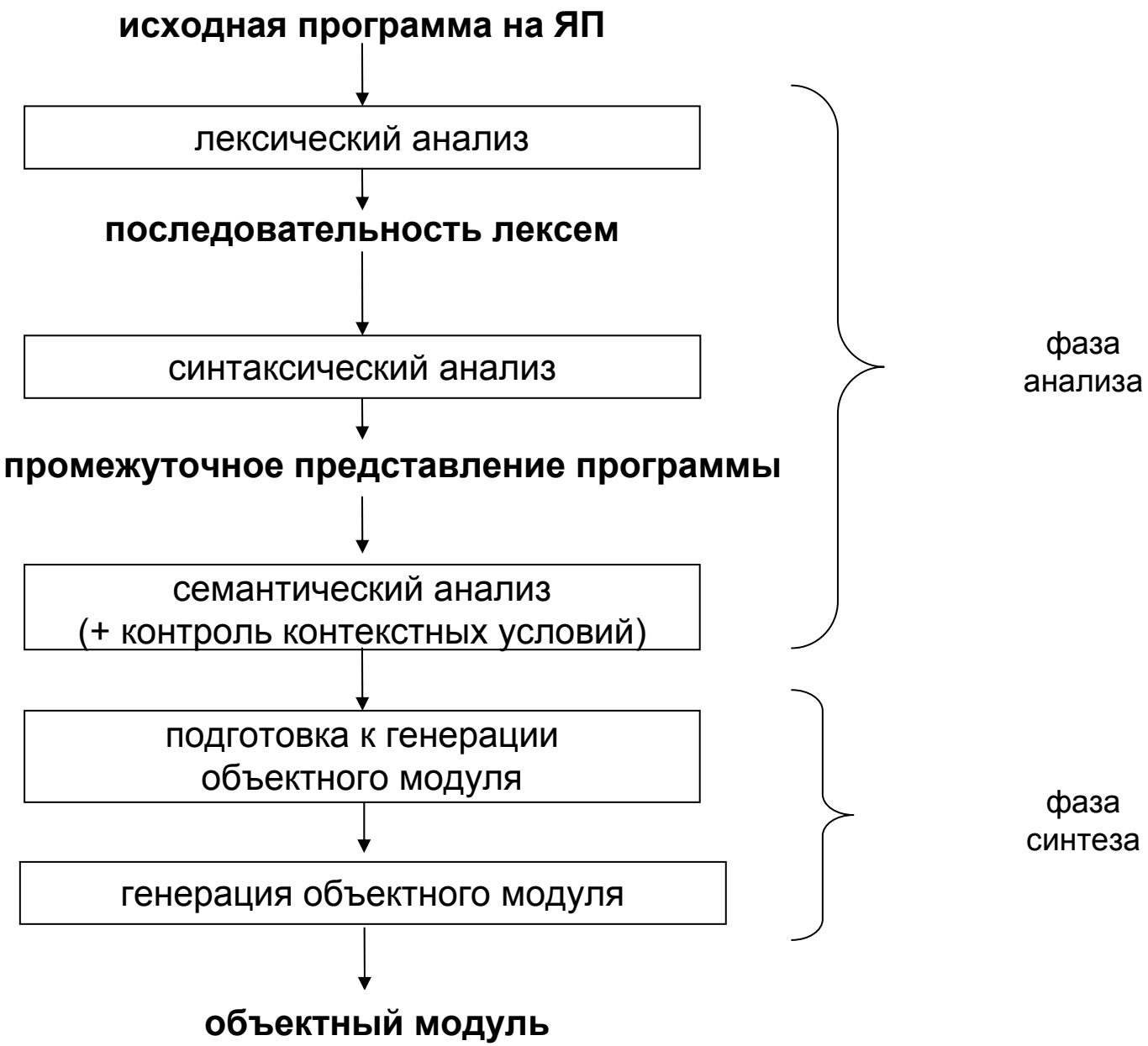
Смешанная стратегия



Примеры: 1) язык Java ⟶ байт-код ⟶ JVM (Java Virtual Machine)

2) Языки на технологии .NET (Basic,C#,C++) ⟶ промежуточный язык: CIL – Common Intermediate Language ⟶ JIT-компилятор (just-in-time)

Схема функционирования компилятора



Основные понятия теории формальных языков

Алфавит:

это конечное множество символов.

Пример: V = {a,b,c}

Цепочка символов в алфавите V :

любая конечная последовательность символов этого алфавита.

Пример: V = {a,b}. Цепочка: abbba

Пустая цепочка :

цепочка, которая не содержит ни одного символа.

Обозначение: ε

(иногда для этой цели используется символ Λ)

Конкатенация (сцепление) цепочек α и β:

цепочка, полученная приписыванием последовательности β справа к α

Обозначение: $\alpha\cdot\beta$ (или $\alpha\beta$)

Пример: если α = ab и β = cd, то

$\alpha\beta$ = abcd.

- Для любой цепочки α верно $\alpha\varepsilon = \varepsilon\alpha = \alpha$.

  (Аналог для чисел: $\forall$ x верно $1\cdot x = x\cdot 1 = x$)

- Операция конкатенации ассоциативна: $(\alpha\cdot\beta)\cdot\gamma = \alpha\cdot(\beta\cdot\gamma)$ для любых α, β, γ.

- Операция конкатенации некоммутативна:

  например, для α = ab и β = cd $\quad \alpha\cdot\beta \neq \beta\cdot\alpha$

Обращение *(*или *реверс)* цепочки α:
цепочка, символы которой записаны в обратном порядке.

Обозначение: α^R

Пример: если α = abcdef, то α^R = fedcba.

Для пустой цепочки: $\varepsilon = \varepsilon^R$.

Рекурсивное определение:

$$\alpha^R = \begin{cases} \varepsilon, & \text{если } \alpha = \varepsilon; \\ \beta^R s, & \text{если } \alpha = s\,\beta^R, \text{ где } s \text{ — символ алфавита} \end{cases}$$

n-я степень цепочки α (обозначается α^n) :

конкатенация n цепочек α

$$\underbrace{\alpha\ \alpha\ \alpha\ \ldots\ \alpha\ \alpha}_{n\ \text{раз}} = \alpha^n$$

Рекурсивное определение:

$$\alpha^n = \begin{cases} \varepsilon, & \text{если } n=0; \\ \alpha^{n-1}\alpha, & \text{если } n>0 \end{cases}$$

Длина *цепочки* :
это число составляющих ее символов

Пример: если α = abcdefg, то длина α равна 7.

Обозначение: $|\alpha|$

$|\varepsilon| = 0$

Рекурсивное определение:

$$|\alpha| = \begin{cases} 0, & \text{если } \alpha=\varepsilon; \\ |\beta| + 1, & \text{если } \alpha = \beta s, \text{ где } s \text{ — символ алфавита} \end{cases}$$

Обозначение $|\alpha|_s$ используется для числа вхождений символа s в цепочку α.

Язык в алфавите V :

подмножество цепочек конечной длины в этом алфавите.

V^* :

множество, содержащее все цепочки конечной длины в алфавите V, включая пустую цепочку ε.

Пример: $V = \{0,1\}$,

$V^* = \{\varepsilon, 0, 1, 00, 11, 01, 10, 000, 001, 011, ...\}$.

V^+ :

множество, содержащее все цепочки конечной длины в алфавите V, исключая пустую цепочку ε.

- $V^* = V^+ \cup \{\varepsilon\}$
- *Для любого языка* $L \subseteq V^*$

способы описания языков

Явное перечисление:

L = {ab, abb,ba, baa}
Язык L — конечный

Формулы:

$L = \{a^n b^n \mid n \geq 0\}$
Цепочки языка L: ε, ab, aabb, aaabbb, …

Порождающие грамматики Хомского

Распознаватели (МТ, ЛОА, конечные автоматы, МП-автоматы)

Декартово произведение множеств A и B :

множество $\{ (a,b) \mid a \in A, b \in B\}$ Обозначение: $A \times B$

Порождающая грамматика G :

это четверка $\langle T, N, P, S \rangle$, где

- T – алфавит *терминальных символов (терминалов)*,
- N – алфавит *нетерминальных символов (нетерминалов)*, не пересекающийся с T,
- P – конечное подмножество множества $(T \cup N)^+ \times (T \cup N)^*$;

элемент (α, β) множества P называется *правилом вывода* и записывается в виде $\alpha \to \beta$,

α содержит хотя бы один нетерминал;

S – *начальный символ (цель)* грамматики, $S \in N$.

правила

$\alpha \to \beta_1 \quad \alpha \to \beta_2 \ \ldots \ \alpha \to \beta_n$

записываются сокращенно

$\alpha \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$

β_i для $i = 1, 2, \ldots, n$, - ***альтернатива*** правила вывода из цепочки α.

Порождающая грамматика G :
это четверка $\langle T, N, P, S \rangle$, где

- T – алфавит *терминальных символов (терминалов)*,
- N – алфавит *нетерминальных символов (нетерминалов)*,
  $T \cap N = \varnothing$,
- P – конечное подмножество множества $(T \cup N)^+ \times (T \cup N)^*$;
  элемент (α, β) множества P называется *правилом вывода* и записывается в виде $\alpha \to \beta$,
- S – *начальный символ (цель)* грамматики, $S \in N$.

Пример грамматики: $G_1 = \langle \{0,1\}, \{A, S\}, P, S \rangle$

P:

$S \to 0A1$

$0A \to 00A1$

$A \to \varepsilon$

Пусть $\beta \in (T \cup N)^*$ $\alpha \in (T \cup N)^+$

β ***непосредственно выводима*** *из* α
в грамматике $G = \langle T, N, P, S \rangle$,
если $\alpha = \xi_1\gamma\xi_2$,
$\beta = \xi_1\delta\xi_2$,
где $\xi_1, \xi_2, \delta \in (T \cup N)^*$, $\gamma \in (T \cup N)^+$ и $\gamma \to \delta \in P$.

Обозначение: $\alpha \to \beta$

Пример: $G_1 = \langle \{0,1\}, \{A, S\}, P, S \rangle$
P:
$S \to 0A1$
$0A \to 00A1$
$A \to \varepsilon$

Цепочка 00A11 непосредственно выводима из 0A1 в грамматике G_1 : $\underline{0A}1 \to \underline{00A11}$ (по правилу $0A \to 00A1$)

Цепочка $\beta \in (T \cup N)^*$ ***выводима*** из цепочки $\alpha \in (T \cup N)^+$
в грамматике $G = \langle T, N, P, S \rangle$,
если существуют цепочки $\gamma_0, \gamma_1, \ldots, \gamma_n$ $(n \geq 0)$, такие, что

$$\alpha = \gamma_0 \to \gamma_1 \to \ldots \to \gamma_n = \beta.$$

Обозначение: $\alpha \Rightarrow \beta$

Вывод длины n :
последовательность $\gamma_0, \gamma_1, \ldots, \gamma_n$

Любая цепочка выводится сама из себя за 0 шагов: $\alpha = \gamma_0 = \alpha$

$G_1 = \langle \{0,1\}, \{A, S\}, P, S \rangle$
P:
$S \to 0A1$
$0A \to 00A1$
$A \to \varepsilon$

$S \Rightarrow 000A111$ в G_1, т. к. $\exists$ вывод $S \to 0A1 \to 00A11 \to 000A111$.

Длина вывода равна 3.

Сентенциальная форма в грамматике $G = \langle T, N, P, S \rangle$:
$\alpha \in (T \cup N)^*$, для которой $S \Rightarrow \alpha$

Язык, порождаемый грамматикой $G = \langle T, N, P, S \rangle$:
множество $L(G) = \{\alpha \in T^* \mid S \Rightarrow \alpha\}$

Грамматика — это не алгоритм, а система правил подстановки, позволяющих строить выводы.

Язык, порождаемый грамматикой $G = \langle T, N, P, S \rangle$:

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

Пример: $G_1 = \langle \{0,1\}, \{A,S\}, P, S \rangle$,

P:

(1) $S \to 0A1$

(2) $0A \to 00A1$

(3) $A \to \varepsilon$

$L(G) = ?$

$S \to_{(1)} 0A1 \to_{(3)} 01$

$S \to_{(1)} 0A1 \to_{(2)} 00A11 \to_{(3)} 0011$

$S \to_{(1)} 0A1 \to_{(2)} 00A11 \to_{(2)} 000A111 \to_{(3)} 000111$

…

Предположительно: $L_1 = \{0^n1^n \mid n>0\}$

Язык, порождаемый грамматикой $G = \langle T, N, P, S \rangle$:

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

Пример: $G_1 = \langle \{0,1\}, \{A,S\}, P, S \rangle$,

P:

(1) $S \to 0A1$

(2) $0A \to 00A1$

(3) $A \to \varepsilon$

Предположительно: $L_1 = \{0^n1^n \mid n>0\}$

Требуется доказать: $L(G_1) = L_1$:

(1) $L_1 \subseteq L(G_1)$, т.е. $\forall\, x \in L_1 \Rightarrow x \in L(G_1)$ (индукция по n)

(2) $L(G_1) \subseteq L_1$, т.е. $\forall\, x \in L(G_1) \Rightarrow x \in L_1$ (индукция по длине вывода)

Эквивалентность грамматик G_1 и G_2:

$L(G_1) = L(G_2)$

Пример:

$G_1 = \langle \{0,1\}, \{A,S\}, P_1, S \rangle$ и $G_2 = \langle \{0,1\}, \{S\}, P2, S \rangle$

P_1: $S \to 0A1$

$0A \to 00A1$

$A \to \varepsilon$

P_2: $S \to 0S1 \mid 01$

$$L(G_1) = L(G_2) = \{0^n1^n \mid n>0\}$$

Грамматики G_1 и G_2 ***почти эквивалентны***,

если $L(G_1) \cup \{\varepsilon\} = L(G_2) \cup \{\varepsilon\}$.

Пример:

$G_1 = \langle \{0,1\}, \{A,S\}, P_1, S \rangle$ и $G_2' = \langle \{0,1\}, \{S\}, P_2', S \rangle$

P_1: $S \to 0A1$

$0A \to 00A1$

$A \to \varepsilon$

P_2' : $S \to 0S1 \mid \varepsilon$

$L(G_1)=\{0^n1^n \mid n>0\}$, а $L(G_2') = \{0^n1^n \mid n\geq 0\}$

т.е. $L(G_2')=L(G_1) \cup \{\varepsilon\}$

$G_1 = (\{0,1\}, \{A,S\}, P_1, S)$

P_1: $S \to 0A1$

$0A \to 00A1$

$A \to \varepsilon$

Рассмотрим

$G_3 = (\{0,1\}, \{S\}, P, S)$, где P:

(1) $S \to 0S$

(2) $S \to 1S$

(3) $S \to \varepsilon$

Любая цепочка вида 0^n1^n порождается следующим способом:

-- n раз применить правило (1), затем

-- n раз применить правило (2)

-- и на последнем шаге применить правило (3).

Однако $L(G_3) \neq L(G_1)$,

т.к. $S \to 1S \to 10S \to 10 \in L(G_3)$,

$10 \notin L(G_1)$

Классификация грамматик и языков по Хомскому

грамматики классифицируются по виду их правил вывода

Четыре типа грамматик:

тип 0, тип 1, тип 2, тип 3

Язык, порождаемый грамматикой типа *k* *(k=0,1,2,3)*, является языком *типа k*.

$$G = \langle T, N, P, S \rangle$$

Тип 0

Любая порождающая грамматика является грамматикой *типа 0*.

На вид правил грамматик этого типа не накладывается никаких дополнительных ограничений.

Класс языков типа 0 совпадает с классом рекурсивно перечислимых языков (распознаваемых МТ).

Грамматики с ограничениями на вид правил вывода

Тип 1

Грамматика $G = \langle T, N, P, S \rangle$ называется *неукорачивающей*, если правая часть каждого правила из P не короче левой части (т. е. для любого правила $\alpha \to \beta \in P$ выполняется неравенство $|\alpha| \le |\beta|$).

В виде исключения в неукорачивающей грамматике допускается наличие правила $S \to \varepsilon$, при условии, что S (начальный символ) не встречается в правых частях правил.

Грамматикой *типа 1* будем называть неукорачивающую грамматику.

Тип 1 в некоторых источниках определяют с помощью так называемых контекстно-зависимых грамматик.

Грамматика $G = \langle T, N, P, S \rangle$ называется *контекстно-зависимой* (*КЗ*), если каждое правило из P имеет вид $\alpha \rightarrow \beta$, где $\alpha = \xi_1 A \xi_2$, $\beta = \xi_1 \gamma \xi_2$, $A \in N$, $\gamma \in (T \cup N)^+$, $\xi_1, \xi_2 \in (T \cup N)^*$.

В виде исключения в КЗ-грамматике допускается наличие правила $S \rightarrow \varepsilon$, при условии, что S (начальный символ) не встречается в правых частях правил.

КЗ-грамматика удовлетворяет определению неукорачивающей.

Неукорачивающие и КЗ-грамматики определяют один и тот же класс языков.

Тип 2

Грамматика $G = \langle T, N, P, S \rangle$ называется *контекстно-свободной* (*КС*), если каждое правило из P имеет вид $A \to \beta$, где $A \in N, \beta \in (T \cup N)^*$.

В КС-грамматиках допускаются правила с пустыми правыми частями.

Язык, порождаемый контекстно-свободной грамматикой, называется *контекстно-свободным* языком.

Грамматикой *типа 2* будем называть контекстно-свободную грамматику.

Любую КС-грамматику можно преобразовать в эквивалентную неукорачивающую КС-грамматику. (т.е. КС, удовлетворяющую также и определению неукорачивающей)

Тип 3

Грамматика $G = \langle T, N, P, S \rangle$ называется *праволинейной*, если каждое правило из P имеет вид $A \to wB$ либо $A \to w$, где $A \in N, B \in N, w \in T^*$.

Грамматика $G = \langle T, N, P, S \rangle$ называется *леволинейной*, если каждое правило из P имеет вид $A \to Bw$ либо $A \to w$, где $A \in N$, $B \in N, w \in T^*$.

Праволинейные и леволинейные грамматики определяют один и тот же класс языков. Такие языки называются *регулярными.* Право- и леволинейные грамматики тоже называют регулярными.

Регулярная грамматика является грамматикой *типа 3*.

Автоматной грамматикой называется праволинейная (леволинейная) грамматика, такая, что каждое правило с непустой правой частью имеет вид: $A \to a$ либо $A \to aB$ (для леволинейной, соответственно, $A \to a$ либо $A \to Ba$), где $A \in N$, $B \in N$, $a \in T$.

Для любой регулярной (автоматной) грамматики G существует неукорачивающая регулярная (автоматная) грамматика G', такая что $L(G) = L(G')$.

Иерархия Хомского

Справедливы следующие соотношения:

1) любая регулярная грамматика является КС-грамматикой;

2) любая неукорачивающая КС-грамматика является КЗ-грамматикой;

3) любая неукорачивающая грамматика является грамматикой типа 0.

Неукорачивающие Регулярные $\subset$ *Неукорачивающие КС* $\subset$ *КЗ* $\subset$ *Тип 0*

(Запись $A \subset B$ означает, что A является собственным подклассом класса B)

Справедливы следующие соотношения для языков:

- каждый регулярный язык является КС-языком, но существуют КС-языки, которые не являются регулярными, например:

$$L = \{a^n b^n \mid n > 0\};$$

- каждый КС-язык является КЗ-языком, но существуют КЗ-языки, которые не являются КС-языками, например:

$$L = \{a^n b^n c^n \mid n > 0\};$$

- каждый КЗ-язык является языком типа 0 (т. е. рекурсивно перечислимым языком), но существуют языки типа 0, которые не являются КЗ-языками, например: язык, состоящий из записей самоприменимых алгоритмов Маркова в некотором алфавите.

Иерархия классов языков

Тип 3 (*Регулярные*) $\subset$ *Тип 2* (*КС*) $\subset$ *Тип 1* (*КЗ*) $\subset$ *Тип 0*

Проблема «Можно ли язык, описанный грамматикой типа k ($k = 0, 1, 2$), описать грамматикой типа $k + 1$?» является алгоритмически неразрешимой.

Язык $L_{a,b} = \{a, b\}$. Какого он типа? Обычно требуется указать максимально возможный тип.

Ответ: типа 3

$S \to a \mid b$ — грамматика типа 3, порождающая данный язык.

($L_{a,b}$ является также языком типа 2, 1, 0 в силу иерархии Хомского)

(1) # *Примеры грамматик и языков*

$$S \rightarrow ABCS \mid ABc$$
$$BA \rightarrow AB$$
$$CA \rightarrow AC$$
$$CB \rightarrow BC$$
$$Cc \rightarrow cc$$
$$Bc \rightarrow bc$$
$$Bb \rightarrow bb$$
$$Ab \rightarrow ab$$
$$Aa \rightarrow aa$$

Тип 1. Неукорачивающая, но не КЗ

Язык: $\{a^n b^n c^n \mid n > 0\}$

Примеры грамматик и языков

(2)

$$S \to aSb \mid ab$$

Язык: $\{a^n b^n \mid n > 0\}$

(3)

$$S \to aS \mid a$$

Язык: $\{a^n \mid n > 0\}$

Иерархия классов Хомского

Задача распознавания

Даны грамматика G и цепочка x

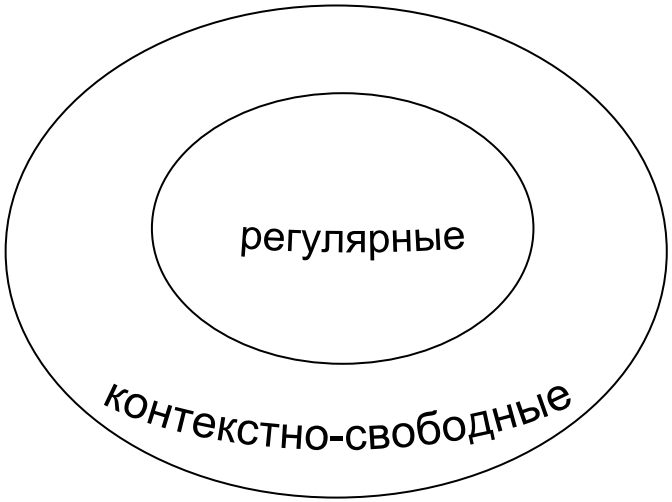
$x \in L(G)$?

Для грамматик типа 1 (а также типов 2 и 3) по классификации Хомского задача распознавания разрешима, т.е. существует общий алгоритм, отвечающий на вопрос: $x \in L(G)$?

Контекстно-свободные грамматики и языки

КС-грамматики позволяют выразить такие свойства языков программирования, как скобочные структуры, последовательность описаний и операторов и др. Но не могут задавать контекстно-зависимые свойства, например, соответствие числа формальных и фактических параметров при вызове функции. Для КС-грамматик существуют эффективные алгоритмы анализа, поэтому они применяются в трансляции, контекстные условия проверяются на

этапе семантического анализа



Левый (левосторонний) вывод цепочки $\beta \in (V_T)^*$ из $S \in V_N$ в КС-грамматике $G = (V_T, V_N, P, S)$:

в этом выводе каждая очередная сентенциальная форма получается из предыдущей заменой самого левого нетерминала.

Правый (правосторонний) вывод цепочки $\beta \in (V_T)^*$ из $S \in V_N$ в КС-грамматике $G = (V_T, V_N, P, S)$:

в этом выводе каждая очередная сентенциальная форма получается из предыдущей заменой самого правого нетерминала.

Рассмотрим пример грамматики:

$G = (\{a,b,+\}, \{S,T\}, \{S \to T \mid T+S;\ T \to a \mid b\}, S)$

можно построить выводы для цепочки a+b+a :

(1) $S \to T+S \to T+T+S \to T+T+T \to a+T+T \to a+b+T \to a+b+a$

(2) $S \to T+S \to a+S \to a+T+S \to a+b+S \to a+b+T \to a+b+a$

(3) $S \to T+S \to T+T+S \to T+T+T \to T+T+a \to T+b+a \to a+b+a$

Здесь (2) - левосторонний вывод, (3) - правосторонний, а (1) не является ни левосторонним, ни правосторонним

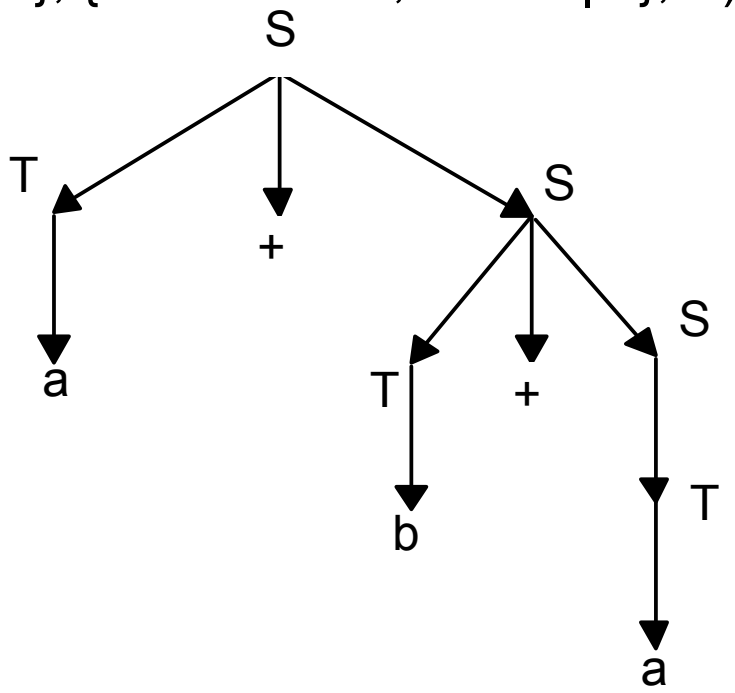
***Определение*: упорядоченное ориентированное дерево называется **деревом вывода** (или **деревом разбора**) в КС-грамматике G = (T, N, P, S), если выполнены следующие условия:**

(1) каждая вершина дерева помечена символом из множества $N \cup T \cup \{\varepsilon\}$, при этом корень дерева помечен символом S; листья - символами из $T \cup \{\varepsilon\}$;

(2) если вершина дерева помечена символом A , а ее непосредственные потомки - символами $a_1, a_2, \ldots, a_n$, где каждое $a_i \in T \cup N$, то $A \to a_1 a_2 \ldots a_n$ - правило вывода в этой грамматике;

(3) если вершина дерева помечена символом A , а ее единственный непосредственный потомок помечен символом ε, то $A \to \varepsilon$ — правило вывода в этой грамматике.

Пример дерева вывода для цепочки a+b+a в грамматике G = ({a,b,+}, {S,T}, {S → T | T+S; T → a | b}, S) :



(1) S→T+S → T+T+S → T+T+T → a+T+T → a+b+T → a+b+a

(2) S → T+S → a+S → a+T+S → a+b+S → a+b+T → a+b+a

(3) S → T+S → T+T+S → T+T+T → T+T+a → T+b+a→ a+b+a

КС-грамматика G называется **неоднозначной**,

если существует хотя бы одна цепочка $\alpha \in L(G)$, для которой может быть построено два или более различных деревьев вывода.

Это утверждение эквивалентно тому, что цепочка α имеет два или более разных левосторонних (или правосторонних) выводов.

В противном случае грамматика называется **однозначной.**

Утв. Проблема определения, является ли заданная КС-грамматика однозначной, является **алгоритмически неразрешимой**.

Язык, порождаемый грамматикой, называется **неоднозначным**, если он не может быть порожден никакой однозначной грамматикой.

Утв. Проблема определения, порождает ли данная КС-грамматика однозначный язык (т.е. существует ли эквивалентная ей однозначная грамматика), является **алгоритмически неразрешимой.**

- Пример неоднозначного языка:

$L = \{a^n b^n c^m \mid n>0, m>0\} \cup \{a^n b^m c^m \mid n>0, m>0\}$

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: $S \to aAB \mid C$

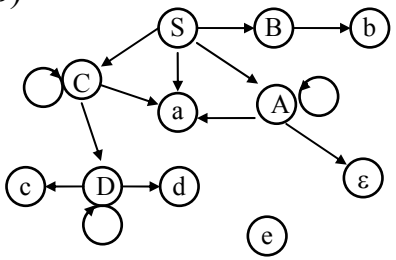
$D \to cDc \mid d$

$C \to aCD$

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

$B \to b$

Граф грамматики G:



Пример. Дана грамматика

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

P: $S \to aAB \mid C$

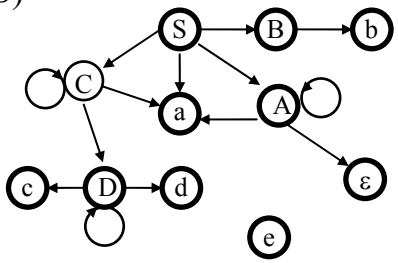
$D \to cDc \mid d$

$C \to aCD$

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

$B \to b$

Граф грамматики G:



Не отмеченные жирным кружком символы бесплодны.

Пример. Дана грамматика

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

P: $S \to aAB \mid \sout{C}$

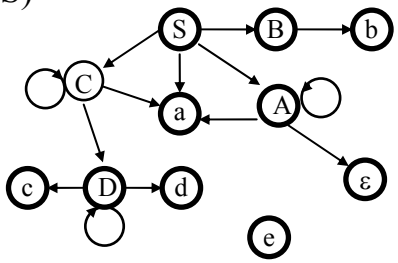
$D \to cDc \mid d$

$\sout{C \to aCD}$

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

$B \to b$

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 : $S \to aAB$

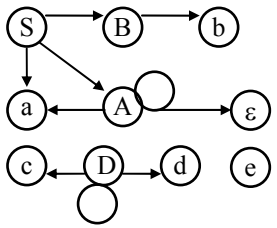
$D \to cDc \mid d$

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

$B \to b$

G_1 не содержит бесплодных символов.

Граф грамматики G_1 :

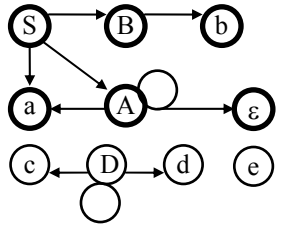


Находим недостижимые символы

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

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.

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

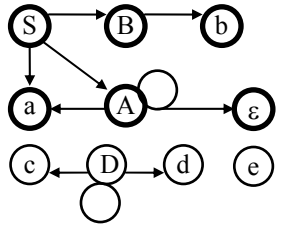
P_1 : $S \to aAB$

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

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

$B \to b$

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 : $S \to aAB$

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

$B \to b$

G_2 – приведенная грамматика

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

<u>Задача.</u> 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 $B \to \alpha_1 A_1 \alpha_2 A_2 ... \alpha_n A_n \alpha_{n+1}$,

где $\alpha_i \in ((N - X) \cup T)^*$

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

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

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

...

$B \to \alpha_1 \alpha_2 A_2 ... \alpha_n A_n \alpha_{n+1}$

$B \to \alpha_1 A_1 \alpha_2 A_2 ... \alpha_n A_n \alpha_{n+1}$

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

5. Удалить бесполезные символы и правила, их содержащие.

Пример.	$S \to BC \mid Ab$		$S \to C \mid b \mid Ab$
исходная	$B \to \varepsilon$	эквивалентная	$C \to c$
грамматика	$C \to c$	грамматика	$A \to Aa \mid a$
	$A \to Aa \mid \varepsilon$		