Введение в теорию формальных языков

Лекция 10

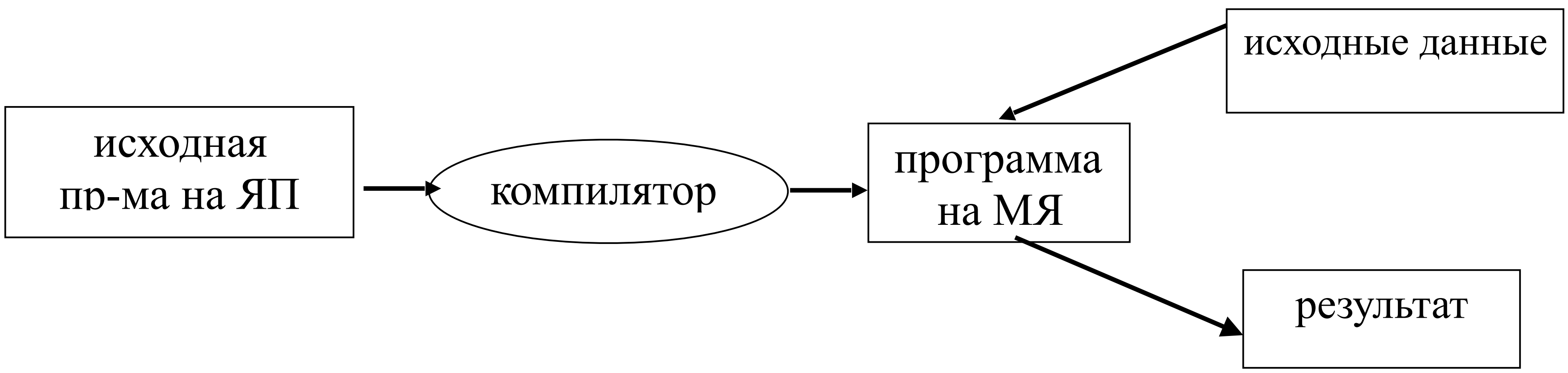
Трансляторы

Главным компонентом систем программирования является транслятор.

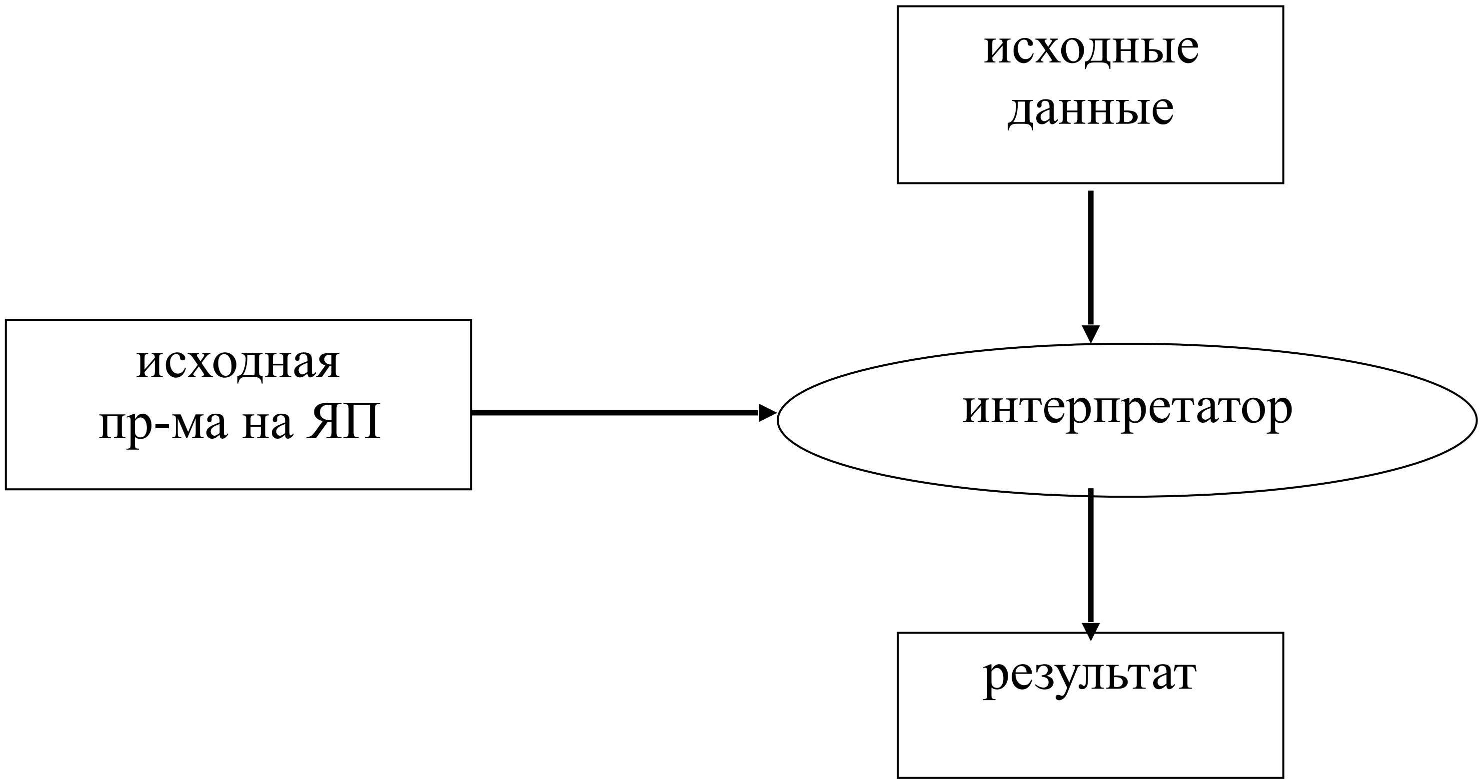
Все трансляторы подразделяются на два основных класса:

- **компиляторы**,
- **интерпретаторы**.

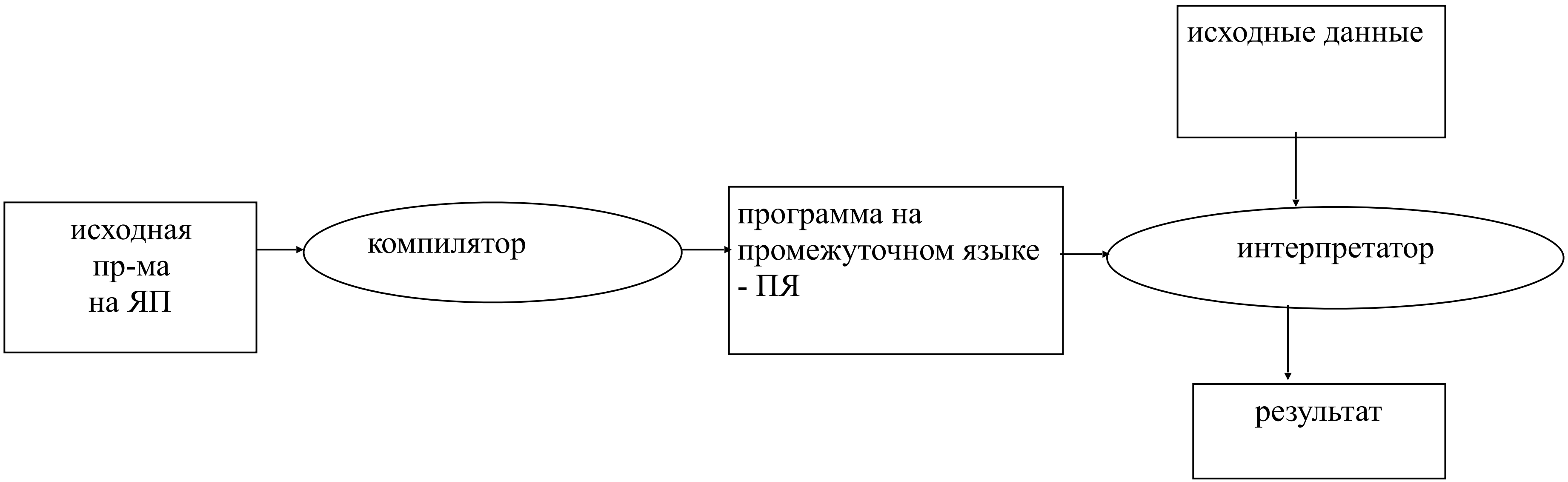
Компилятор


исходная
пр-ма на ЯП
компилятор
программа
на МЯ
исходные данные
результат

Интерпретатор



Смешанная стратегия трансляции

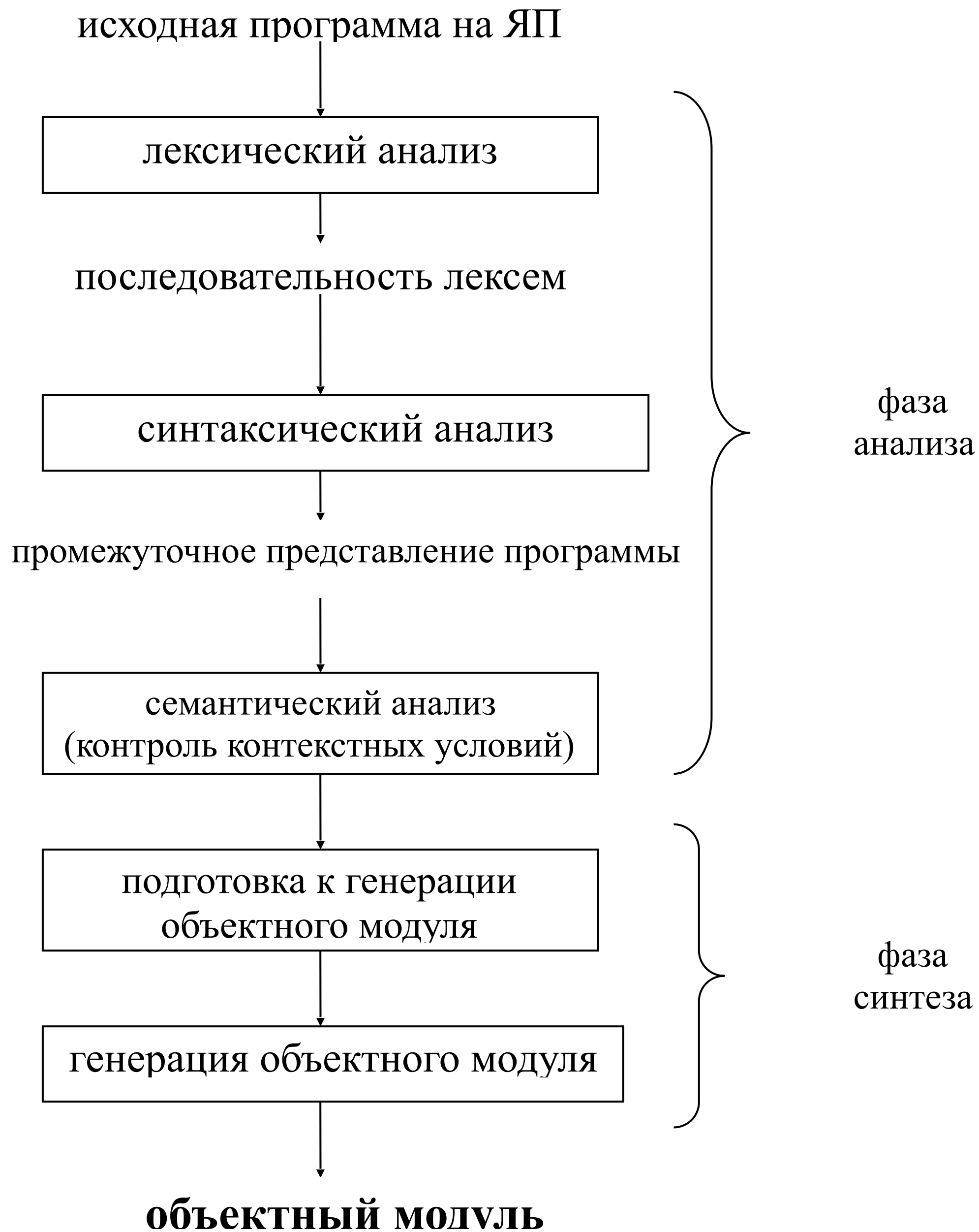


Проход компилятора – процесс последовательного чтения компилятором данных из внешней памяти, их обработка и помещение результата во внешнюю память, в частности, ОП.

Один проход включает в себя выполнение одного или нескольких этапов компиляции.

Результат промежуточных проходов – внутреннее представление исходной программы, результат последнего прохода – объектная программа**.**

Схема функционирования компилятора



Описание формального языка

- **Алфавит** задается перечислением конечного непустого множества символов, которые могут быть использованы для записи текстов на каком-либо языке.

- **Синтаксис** определяется набором правил, устанавливающих, какие комбинации символов алфавита являются правильными текстами на определяемом языке и позволяющих связать с каждым правильным текстом на этом языке некоторую синтаксическую структуру.

- **Семантика** определяет смысл синтаксически правильных конструкций языка, то, что означает конструкция.

  Семантика обычно описывается словами. Четкое и точное описание семантики очень важно для транслятора, т.к. его цель - получить *эквивалентную* программу на МЯ (для компилятора), либо точно выполнить указанные действия (для интерпретатора).

- **Прагматика** формального языка сводится к аргументации того, зачем та или иная конструкция вошла в состав языка.

Основные компоненты компилятора

- Информационные таблицы:
  - служебных идентификаторов
  - констант
  - имен
  - процедур
  - блоков
  - циклов
  - …

Фазы компиляции

- Фаза анализа программ (заполнение таблиц):
  - лексический анализатор (сканер);
  - синтаксический и семантический анализаторы (парсер).
- Построение внутреннего представления программы.
- Фаза оптимизации.
- Фаза синтеза (асс. или маш. код).
- Распределение памяти.
- Генерация команд и маш.-зав. оптимизация.

Теория формальных языков и грамматик. Определения 1.

***Цепочка символов в алфавите V* -** любая конечная последовательность символов этого алфавита.

***Пустая цепочка (ε)* -** цепочка, которая не содержит ни одного символа.

Если α и β - цепочки, то цепочка $\alpha\beta$ **-** ***конкатенация* цепочек α и β.**

Например, если α **= ab и** β **= cd, то** $\alpha\beta$ **= abcd,**
$\alpha\varepsilon = \varepsilon\alpha = \alpha$.

Обращение (или реверс) цепочки α - цепочка, символы которой записаны в обратном порядке, обозначается как α^R.

Например, если α **= abcdef, то** α^R **= fedcba,** $\varepsilon = \varepsilon^R$.

Теория формальных языков и грамматик. Определения 1.

n-ая степенью цепочки α (α^n) – конкатенация n цепочек α;

$$\alpha^0 = \varepsilon; \quad \alpha^n = \alpha\, \alpha^{n-1} = \alpha^{n-1}\, \alpha.$$

Длина цепочки **-** количество составляющих ее символов.

Например, если α = abcdefg, то длина α равна 7. Длину цепочки α обозначается $|\,\alpha\,|$. $|\,\varepsilon\,| = 0$

Определения 2.

Язык в алфавите V - это подмножество цепочек конечной длины в этом алфавите.

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

Например, если V = { 0, 1 }, то

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

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

$V^* = V^+ \cup \{ \varepsilon \}$.

Порождающая грамматика

Порождающая грамматика G - это четверка

$$G = (T, N, P, S) \quad , \quad \text{где}$$

T – непустое множество ***терминальных*** *символов* *(алфавит терминалов)*,

N – непустое множество ***нетерминальных*** *символов* *(алфавит нетерминалов)*, не пересекающийся с Т,

P - конечное подмножество множества $(T \cup N)^+ \times (T \cup N)^*$.

Элемент (α, β) множества Р называется ***правилом вывода*** и записывается в виде

$$\alpha \to \beta,$$

причем α содержит **хотя бы один нетерминальный символ**.

S - ***начальный символ (цель)*** грамматики, $\mathbf{S} \in \mathbf{N}$.

Декартовым произведением $A \times B$ множеств А и В называется множество $\{ (a,b) \mid a \in A, b \in B\}$.

Соглашения

1) Большие латинские буквы будут обозначать нетерминальные символы.

2) **S** - будет обозначать начальный символ (цель) грамматики.

3) Маленькие греческие буквы будут обозначать цепочки символов.

4) Все остальные символы (маленькие латинские буквы, знаки операций и пр.) будем считать терминальными символами.

Соглашения

5) для записи правил вывода с одинаковыми левыми частями

$$\alpha \to \beta 1 \qquad \alpha \to \beta 2 \qquad \ldots \qquad \alpha \to \beta n$$

будем пользоваться сокращенной записью

$$\alpha \to \beta\mathbf{1} \mid \beta\mathbf{2} \mid \ldots \mid \beta\mathbf{n}.$$

Каждое βi , $i = 1, 2, \ldots, n$, будем называть ***альтернативой*** правила вывода из цепочки α**.**

Пример грамматики:

G1 = ({0,1}, {A,S}, P, S), где P состоит из правил:

$$S \to 0A1$$

$$0A \to 00A1$$

$$A \to \varepsilon$$

Определения 3.

Цепочка $\beta \in (T \cup N)^*$ ***непосредственно выводима*** из цепочки $\alpha \in (T \cup N)^+$ в грамматике $G = (T, N, P, S)$, если $\alpha = \xi_1 \gamma \xi_2$, $\beta = \xi_1 \delta \xi_2$, где ξ_1, ξ_2, $\delta \in (T \cup N)^*$, $\gamma \in (T \cup N)^+$ и правило вывода $\gamma \to \delta$ содержится в P.

обозначается: $\alpha \to \beta$

Цепочка $\beta \in (T \cup N)^*$ ***выводима*** из цепочки $\alpha \in (T \cup N)^+$ в грамматике $G = (T, N, P, S)$, если существуют цепочки $\gamma 0$, $\gamma 1$, ... , γn $(n >= 0)$, такие, что

$$\alpha = \gamma 0 \to \gamma 1 \to \ldots \to \gamma n = \beta.$$

обозначается $\alpha \Rightarrow \beta$

Последовательность $\gamma 0$, $\gamma 1$,..., γn называется ***выводом длины n.***

Определения 3.

Язык, порождаемый грамматикой **$G = (T, N, P, S)$:**

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

Сентенциальная форма в грамматике $G = (T, N, P, S)$ - цепочка $\alpha \in (T \cup N)^*$, для которой $S \Rightarrow \alpha$.

Язык, порождаемый грамматикой - множество терминальных сентенциальных форм.

Определения 4.

Грамматики G1 и G2 называются ***эквивалентными***, если

$L(G1) = L(G2)$.

Например, $G1 = (\{0,1\}, \{A,S\}, P1, S)$ и $G2 = (\{0,1\}, \{S\}, P2, S)$

P1: $S \to 0A1$

$0A \to 00A1$

$A \to \varepsilon$

P2: $S \to 0S1 \mid 01$

эквивалентны, т.к. обе порождают язык $L(G1) = L(G2) = \{ 0^n 1^n \mid n > 0\}$.

Грамматики G1 и G2 ***почти эквивалентны***, если

$L(G1) \cup \{\varepsilon\} = L(G2) \cup \{\varepsilon\}$.

Например, $G1 = (\{0,1\}, \{A,S\}, P1, S)$ и $G2 = (\{0,1\}, \{S\}, P2, S)$

P1: $S \to 0A1$

$0A \to 00A1$

$A \to \varepsilon$

P2: $S \to 0S1 \mid \varepsilon$

почти эквивалентны, так как

$L(G1) = \{ 0^n 1^n \mid n > 0 \}$, а $L(G2) = \{ 0^n 1^n \mid n >= 0 \}$.

Классификация грамматик и языков по Хомскому

ТИП 0:

Грамматика G = (T, N, P, S) - *грамматика типа 0*, если на ее правила вывода не накладывается никаких ограничений.

ТИП 1:

Грамматика G = (T, N, P, S) - ***неукорачивающая*** *грамматикой*, если каждое правило из P имеет вид

$$\alpha \rightarrow \beta, \text{ где } \alpha \in (T \cup N)^{+}, \beta \in (T \cup N)^{+} \text{ и } |\alpha| <= |\beta|.$$

Исключение - в неукорачивающей грамматике допускается ***наличие правила $S \rightarrow \varepsilon$, при условии, что S (начальный символ) не встречается в правых частях правил.***

Грамматика G = (T, N, P, S) - ***контекстно-зависимая*** *(КЗ)*, если каждое правило из P имеет вид

$$\alpha \rightarrow \beta, \text{ где } \alpha = \xi1\, A\, \xi2;\ \ \beta = \xi1\, \gamma\, \xi2;\ \ A \in N;\ \ \gamma \in (T \cup N)+;\ \ \xi1, \xi2 \in (T \cup N)^{*}.$$

В КЗ-грамматике допускается ***Исключение.***

Грамматику типа 1 можно определить как неукорачивающую либо как контекстно-зависимую.

Классификация грамматик и языков по Хомскому

ТИП 2:

Грамматика $G = (T, N, P, S)$ - ***контекстно-свободная*** *(КС)*, если каждое правило из P имеет вид

$$A \to \beta, \text{ где } A \in N,\ \beta \in (T \cup N)^*.$$

Грамматика $G = (T, N, P, S)$ - ***неукорачивающая контекстно-свободная*** *(НКС)*, если каждое правило из P имеет вид

$$A \to \beta, \text{ где } A \in N,\ \beta \in (T \cup N)^+.$$

В неукорачивающей КС-грамматике допускается ***Исключение.***

Грамматику типа 2 можно определить как контекстно-свободную либо как неукорачивающую контекстно-свободную.

Классификация грамматик и языков по Хомскому

ТИП 3:

Грамматика $G = (T, N, P, S)$ - ***праволинейная***, если каждое правило из P имеет вид имеет вид:

$A \to wB$ либо $A \to w$, где $A, B \in N$, $w \in T^*$.

Грамматика $G = (T, N, P, S)$ - ***леволинейная*****,** если каждое правило из P имеет вид: $A \to Bw$ либо $A \to w$, где $A, B \in N$, $w \in T^*$.

*Грамматику типа 3 (****регулярную****, Р-грамматику)* можно определить как праволинейную либо как леволинейную.

Автоматная грамматика - праволинейная (леволинейная) грамматика, такая, что каждое правило с непустой правой частью имеет вид: $A \to a$ либо $A \to aB$

(для леволинейной, $A \to a$ либо $A \to Ba$), где $A, B \in N$, $a \in T$.

Соотношения между типами грамматик

неук. Р ⊂ неук. КС ⊂ КЗ ⊂ Тип 0

(1) Любая регулярная грамматика является КС-грамматикой.

(2) Любая неукорачивающая КС-грамматика является КЗ-грамматикой.

$\to$,

(3) Любая неукорачивающая грамматика является грамматикой типа 0.

Язык L(G) является ***языком типа k*** по Хомскому, если его можно описать грамматикой типа k, где k - максимально возможный номер типа грамматики по Хомскому.

Соотношения между типами языков

Р $\subset$ КС $\subset$ КЗ $\subset$ Тип 0

(1) Каждый регулярный язык является КС-языком, но существуют КС-языки, которые не являются регулярными
(например, $L = \{ a^n b^n \mid n > 0 \}$).

(2) Каждый КС-язык является КЗ-языком, но существуют КЗ-языки, которые не являются КС-языками
(например, $L = \{ a^n b^n c^n \mid n > 0 \}$).

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

(4) Кроме того, существуют языки, которые вообще нельзя описать с помощью порождающих грамматик. Такие языки не являются рекурсивно перечислимым множеством.

Проблема, можно ли язык, описанный грамматикой типа **k**, описать грамматикой типа **k + 1** (k = 0, 1, 2), является **алгоритмически неразрешимой**.

КС-грамматики

Разбор цепочки - процесс построения вывода цепочки α из цели S грамматики G = (T, N, P, S).

Вывод цепочки $\beta \in T^*$ из $S \in N$ в КС-грамматике G = (T, N, P, S), называется:

- *левосторонним*, если в нем каждая очередная сентенциальная форма получается из предыдущей заменой самого левого нетерминала.

- *правосторонним*, если в нем каждая очередная сентенциальная форма получается из предыдущей заменой самого правого нетерминала.

Например, для цепочки a+b+a в грамматике

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

можно построить выводы:

(1) $S \rightarrow T+S \rightarrow T+T+S \rightarrow T+T+T \rightarrow a+T+T \rightarrow a+b+T \rightarrow a+b+a$ - произвольный

(2) $S \rightarrow T+S \rightarrow a+S \rightarrow a+T+S \rightarrow a+b+S \rightarrow a+b+T \rightarrow a+b+a$ - левый

(3) $S \rightarrow T+S \rightarrow T+T+S \rightarrow T+T+T \rightarrow T+T+a \rightarrow T+b+a \rightarrow a+b+a$ - правый

Выводы (1) – (3) являются **эквивалентными** в том смысле, что в них в одних и тех же местах применяются одни и те же правила вывода, но в различном порядке.

Дерево вывода

Дерево вывода (или ***дерево разбора***) в КС-грамматике G = (T, N, P, S) – дерево, для которого выполнены следующие условия:

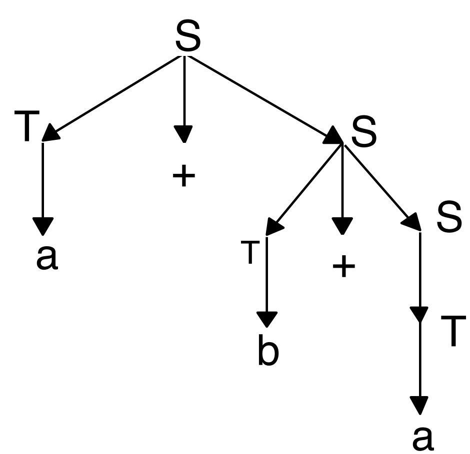
(1) дерево ориентировано и упорядочено;

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

(3) если вершина дерева помечена символом $A \in N$, а ее непосредственные потомки - символами a1, a2, ... , an , где каждое $ai \in T \cup N$, то $A \rightarrow$ a1 a2 ... an - правило вывода в этой грамматике;

(4) если вершина дерева помечена символом $A \in N$, а ее единственный непосредственный потомок помечен символом ε, то $A \rightarrow \varepsilon$ - правило вывода в этой грамматике.

Пример дерева вывода для цепочки a + b + a в грамматике G:



Дерево вывода можно строить ***нисходящим*** либо ***восходящим*** способом.

Неоднозначность грамматик

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

В противном случае грамматика является ***однозначной***.

Если грамматика однозначная, то при любом способе построения, нисходящем или восходящем, будет получено одно и то же дерево разбора.

Пример неоднозначной грамматики:

G_if = ({ if, then, else, a, b }, { S }, P, S),

где P = { $S \rightarrow$ if b then S else S | if b then S | a }.

В этой грамматике для цепочки

if b then if b then a else a

можно построить два различных дерева вывода.

Неоднозначность грамматик

Неоднозначность - это свойство грамматики, а не языка.

Если грамматика используется для определения языка программирования, то она должна быть однозначной.

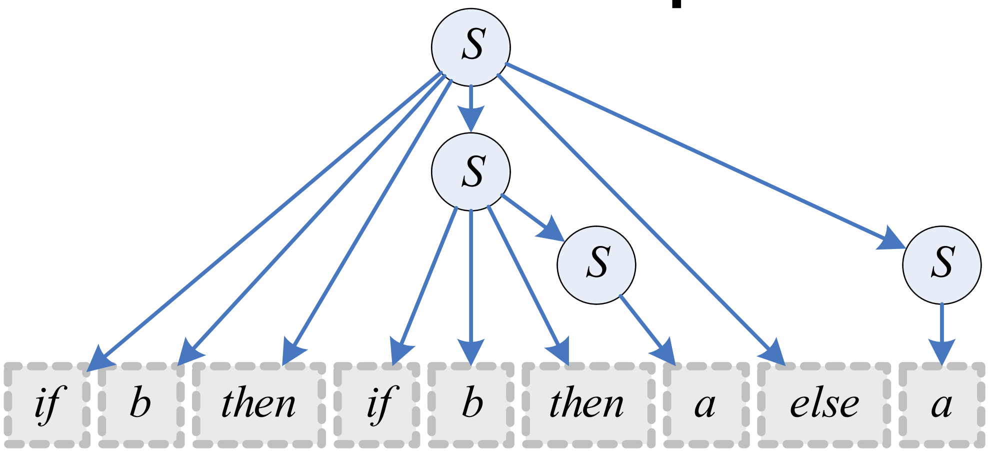
Можно преобразовать грамматику G_if, устранив неоднозначность:

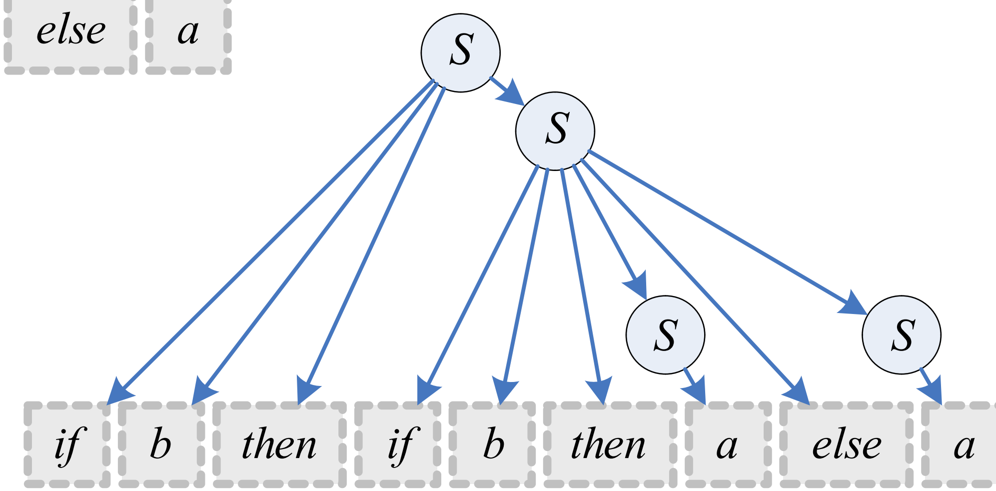
$$S \rightarrow \text{ if b then } S \mid T$$

$$T \rightarrow \text{ if b then } T \text{ else } S \mid a$$

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

Деревья вывода для цепочки *if b then if b then a else a* в **грамматике G_if**





Грамматика **G_if** неоднозначна**,** однако, это **не** означает, что язык **L(G_if)** неоднозначный.

Преобразование неоднозначных грамматик

Некоторые виды правил вывода, которые приводят к неоднозначности и некоторые способы эквивалентных преобразований неоднозначных грамматик к однозначным:

1. $A \longrightarrow AA \mid \alpha \qquad \Rightarrow \qquad A \longrightarrow \alpha A \mid \alpha$

(док-во для $\alpha\alpha\alpha$) – порождаются подцепочки α^n (n >= 1);

2. $A \longrightarrow A\alpha A \mid \beta \qquad \Rightarrow \qquad A \longrightarrow \beta\alpha A \mid \beta$

(док-во для $\beta\alpha\beta\alpha\beta$) – порождаются подцепочки $\beta\ (\alpha\ \beta)^n$ (n >= 0);

3. $A \longrightarrow \alpha A \mid A\beta \mid \gamma \qquad \Rightarrow \qquad A \longrightarrow \alpha A \mid B;$

$B \longrightarrow B\beta \mid \gamma, \quad B \notin N$

(док-во для $\alpha\gamma\beta$) – порождаются подцепочки $\alpha^n \gamma\ \beta^m$ (n, m >= 0);

4. $A \longrightarrow \alpha A \mid \alpha A\beta A \mid \gamma \Rightarrow \qquad A \longrightarrow \alpha A \mid B;$

$B \longrightarrow \alpha B\beta A \mid \gamma,\ B \notin N$

(док-во для $\alpha\alpha\gamma\beta\gamma$) – порождаются подцепочки $\delta = \alpha^n\alpha^m\ \gamma(\beta\delta)^m$ (n,m >=0)

Таким приемом преобразована грамматика G_if :

($\alpha \equiv$ if_b_then, $\beta \equiv$ else, a $\equiv \gamma$, A $\equiv$ S, B $\equiv$ T).

Неоднозначные языки

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

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

Пример неоднозначного КС-языка:

$L = \{a^i b^j c^k \mid i = j \text{ или } j = k\}$.

Одна из грамматик, порождающих L, такова:

$S \rightarrow AB \mid DC$ (док-во для цепочки abc)

$A \rightarrow aA \mid \varepsilon$

$B \rightarrow bBc \mid \varepsilon$

$C \rightarrow cC \mid \varepsilon$

$D \rightarrow aDb \mid \varepsilon$

Бесплодные символы грамматики.

Нетерминал (символ) $A \in N$ является ***бесплодным*** в грамматике $G = (T, N, P, S)$, если множество $\{ \alpha \in T^* \mid A \Rightarrow \alpha \}$ пусто.

Алгоритм удаления бесплодных символов:
Вход: КС-грамматика $G = (T, N, P, S)$,
Выход: КС-грамматика $G' = (T, N', P', S)$, не содержащая бесплодных символов, для которой $L(G) = L(G')$.

Метод:
Строим множества $N_0, N_1, \ldots$
1. $N_0 := \varnothing$; $i := 1$.

2. $N_i := N_{i-1} \cup \{ A \mid A \rightarrow \alpha \in P$ и $\alpha \in (T \cup N_{i-1})^* \}$.

Если $N_i \neq N_{i-1}$, то $i := i + 1$ и переходим к шагу 2,
иначе $N' := N_i$; P' состоит из правил множества P, содержащих только символы из $N_i \cup T$; $G' := (T, N', P', S)$.

Удаление бесплодных символов грамматики. Пример.

$S \longrightarrow AC \mid Bb \mid \varepsilon$

$A \longrightarrow aCb$

$B \longrightarrow bB$

$C \longrightarrow cCc \mid c$

$D \longrightarrow Aa \mid Bb \mid d$

Шаг 0: $N_0 := \varnothing$; $i := 1$.

Шаг 1: $N_1 := \{S, C, D\}$; $i := 2$.

Шаг 2: $N_2 := \{S, C, D, A\}$; $i := 3$.

Шаг 3: $N_3 := \{S, C, D, A\} = N_2$, т.е. искомое множество построено.

Удаляем все правила, содержащие нетерминал B, не вошедший в построенное множество:

$S \longrightarrow AC \mid \varepsilon$

$A \longrightarrow aCb$

$C \longrightarrow cCc \mid c$

$D \longrightarrow Aa \mid d$

Недостижимые символы грамматики

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

Алгоритм удаления недостижимых символов:

Вход: КС-грамматика $G = (T, N, P, S)$,

Выход: КС-грамматика $G' = (T', N', P', S)$, не содержащая недостижимых символов, для которой $L(G) = L(G')$.

Метод:

Строим множества $V_0, V_1, \ldots$

1. $V_0 := \{S\}$; $i := 1$.

2. $V_i := V_{i-1} \cup \{ x \mid x \in T \cup N, \quad A \to \alpha x \beta \in P, \; A \in V_{i-1}, \; \alpha, \beta \in (T \cup N)^* \}$.

<u>Если</u> $V_i \neq V_{i-1}$, <u>то</u> $i := i + 1$ и переходим к шагу 2,

<u>иначе</u> $N' := V_i \cap N$; $T' := V_i \cap T$; P' состоит из правил множества P, содержащих только символы из V_i ; $G' := (T', N', P', S)$.

Удаление недостижимых символов грамматики. Пример.

$S \longrightarrow AC \mid \varepsilon$

$A \longrightarrow aCb$

$C \longrightarrow cCc \mid c$

$D \longrightarrow Aa \mid d$

Шаг 0: $V_0 := \{S\};\ i := 1.$

Шаг 1: $V_1 := \{S, A, C, \varepsilon\};\ i := 2.$

Шаг 2: $V_2 := \{S, A, C, \varepsilon, a, b, c\};\ i := 3.$

Шаг 3: $V_3 := \{S, A, C, \varepsilon, a, b, c\} = V_2,$ т.е. искомое множество построено

Удаляем все правила, содержащие символы D и d, не вошедшие в построенное множество:

$S \longrightarrow AC \mid \varepsilon$

$A \longrightarrow aCb$

$C \longrightarrow cCc \mid c$

Приведенные грамматики

Недостижимые и бесплодные символы в грамматике $G = (T, N, P, S)$ называются ***бесполезными*** символами в этой грамматике.

КС-грамматика G называется ***приведенной,*** если в ней нет бесполезных символов.

Алгоритм приведения грамматики:

1) обнаруживаются и удаляются все **бесплодные** нетерминалы.
2) обнаруживаются и удаляются все **недостижимые** символы.

Удаление символов сопровождается удалением правил вывода, содержащих эти символы.

Если в алгоритме переставить шаги 1) и 2), то не всегда результатом будет приведенная грамматика. Например, при такой перестановке шагов грамматика

$S \to AB \mid a$

$A \to b$

$B \to BA$

останется неприведенной.

Алгоритм устранения правил с пустой правой частью

Вход: КС-грамматика $G = (T, N, P, S)$.

Выход: КС-грамматика $G' = (T, N', P', S')$ - неукорачивающая, $L(G') = L(G)$.

Метод:

1. Построить множество $X = \{A \in N \mid A \Rightarrow \varepsilon\}$; $N' := N$.
2. Построить P', удалив из множества правил P все правила с пустой правой частью.
3. Если $S \in X$, то ввести новый начальный символ S', добавив его в N', и в множество правил P' добавить правило $S' \to S \mid \varepsilon$. Иначе просто переименовать S в S'.
4. Изменить P' следующим образом. Каждое правило вида $B \to \alpha_1 A_1 \alpha_2 A_2 \ldots \alpha_n A_n \alpha_{n+1}$, где $A_i \in X$ для $i = 1,\ldots, n$, $\alpha_i \in ((N' - X) \cup T)^*$ для $i = 1,\ldots, n + 1$ (т. е. α_i — цепочка, не содержащая символов из X), заменить 2^n правилами, соответствующими всем возможным комбинациям вхождений A_i между α_i:

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

Если $\alpha_i = \varepsilon$ для всех $i = 1, \ldots, n + 1$, то получившееся на данном шаге правило $B \to \varepsilon$ не включать в множество P'.

5. Удалить бесполезные символы и правила, их содержащие.

Устранение правил с пустой правой частью.

Пример.

$S \rightarrow BC \mid Ab \mid AB$

$A \rightarrow Aa \mid \varepsilon$

$B \rightarrow \varepsilon$

$C \rightarrow c$

Шаг 1: $X := \{ A, B \};$

Шаг 2, 3: $S1 \rightarrow S \mid \varepsilon$

$S \rightarrow BC \mid C \mid Ab \mid b \mid AB \mid A \mid B$

$A \rightarrow Aa \mid a$

$C \rightarrow c$

Приводим грамматику:

$S1 \rightarrow S \mid \varepsilon$

$S \rightarrow C \mid Ab \mid b \mid A$

$A \rightarrow Aa \mid a$

$C \rightarrow c$