Теория алгоритмических языков и трансляторов

(09.03.04 – Программная инженерия)

Лектор:
Крючкова Елена Николаевна
Кафедра прикладной математики АлтГТУ

Что будем рассматривать?

Что такое ЯЗЫК?
Какие существуют ФОРМАЛЬНЫЕ СПОСОБЫ описания языков?
Что такое ЯЗЫК ПРОГРАММИРОВАНИЯ с точки зрения разработчика компилятора?
Как строится компилятор?
Чем программа компилятора отличается от программы интерпретатора?
Как реализовать интерпретатор или компилятор?

Структура курса (учебный план 09.03.04)

Семестр 16 недель
Лекции: 32 часа (1 пара в неделю)
Лабораторные работы: 32 часа (1 пара в неделю)
Экзамен

Литература
Крючкова Е.Н. Методы анализа в теории формальных языков. / Учебное пособие – 276 с.

Лабораторные работы

1 (2 часа) Синтез КС-грамматик
2 (2 часа) КС-грамматика языка программирования
3 (2 часа) Лексика (таблица лексем, конечный автомат лексики, конечный автомат игнорируемых символов)
4 (2 часа) Программа лексического анализатора (сканер)
5 (2 часа) Построение синтаксических диаграмм и их преобразование
6 (2 часа) Синтаксический анализатор (метод рекурсивного спуска)
7 (2 часа) Анализ контекстных условий языка программирования.
8 (2 часа) Реализация семантического дерева
9 (2 часа) Полный семантический анализ
10 (2 часа) Данные интерпретатора
11 (2 часа) Запись и выборка данных при интерпретации
12 (2 часа) Интерпретация выражений
13(2 часа) Работа с флагом интерпретации - проектирование
14(2 часа) Работа с флагом интерпретации - реализация
15 (2 часа) Промежуточный код - проект конструкций

Отчеты по работам

Отчет по каждой работе в электронном виде должен быть оформлен и сдан:

1. Отчет по каждой работе оформляется в виде одного файла *.txt или *.doc или *.docx.
2. Название любого такого файла стандартное:
   <номер работы>_<номер группы>_<Фамилия студента>
3. Примеры: 06_92_Иванов.zip, 03_92_Ivanov.doc
4. Отчеты отсылаются на адрес kruchkova_elena@mail.ru

Задания №1 и №2

Задание №1 (работа №1) – в учебнике.
Залание №2 (работы 2-15) - в файле в ЛК.

1. Лабораторная работа 1 – синтез КС-грамматик
2. Лабораторные работы №2-15 - методы трансляции языков программирования

Задание по трансляторам

В каждом задании описывается некоторый очень усеченный вариант известных языков программирования Java и C++. В задании указывается:

- структура программы,
- типы данных, которые могут использоваться,
- допустимые операции над этими данными,
- операторы,
- операции и операнды, из которых строятся выражения,
- все виды констант, которые могут использоваться в выражениях.

Пример задания по трансляторам

Программа: главная программа языка C++.
Допускается описание функций без параметров, функции возвращают значение.
Типы данных: int (в том числе short , long long, long) .
Операции: арифметические, сравнения.
Операторы: присваивания и do{}while().
Операнды: простые переменные и константы.
Константы: целые в 10 с/с и 16 с/с .

Обратите внимание на полноту реализации задания

1. во всех заданиях предполагается использование ***составного и пустого оператора***,
2. всегда разрешается описание ***глобальных данных***,
3. все ***перечисленные элементы языка должны использоваться в программе*** (например, если разрешается описание функций, то, безусловно, в перечень операторов Вам необходимо включить вызовы функций).

Тема 1

Формальные грамматики и языки

Базовые понятия

Алфавит - непустое конечное множество символов.

Цепочка над алфавитом Σ - конечная последовательность символов алфавита. Длина цепочки x - число ее символов, обозначается $|x|$. Цепочка нулевой длины называется пустой цепочкой и обозначается ε.

Множество всех цепочек (включая ε) над алфавитом Σ обозначается Σ^*

Язык L над алфавитом Σ - подмножество множества Σ^*, то есть $L \subseteq \Sigma^*$

Примеры алфавита

Алфавит $\Sigma = \{0,1\}$

$\Sigma^* = \{\varepsilon , 0, 1 ,00, 01, 10, 11, 000,\ldots\}$

Алфавит $\Sigma = \{0,1,\ldots,9,x,X,a,\ldots f,A,\ldots F\}$

$0Xff8a4 \in \Sigma^*$ $\quad |0Xff8a4| = 7$

$aa4Xff8x \in \Sigma^*$

Множество часел в 16с/с – подмножество Σ^* , но не равно Σ^*

Алфавит языка программирования C++ - все символы клавиатуры

Грамматика

Порождающая грамматика - упорядоченная четверка

$G = (V_T , V_N , P , S)$, где

V_T - алфавит терминальных символов;

V_N – алфавит нетерминальных символов;

P --- конечное множество правил вывода,

$u \to v$, где $u , v \in (V_T \cup V_N)^*$;

S- начальный нетерминальный символ - аксиома грамматики.

Вывод цепочек языка из S по правилам грамматики

Типы $G = (V_T, V_N, P, S)$

Тип грамматики определяется сложностью правил $P = \{u \to v \mid u, v \in (V_T \cup V_N)^*$

Тип 0 – нет ограничений

Тип 1 - НС-грамматики

$\alpha A \beta \to \alpha \phi \beta$, $A \in V_N$, $\alpha, \phi, \beta \in (V_T \cup V_N)^*$

Тип 2 – КС-грамматики

$A \to \phi$, $A \in V_N$, $\phi \in (V_T \cup V_N)^*$

Тип 3 – Автоматные грамматики

- праволинейные

$A \to xB$ или $A \to x$, $A, B \in V_N$, $x \in V_T$

- леволинейные

Пример грамматики

$G = (V_T, V_N, P, S)$, где

V_T = {0,1,…,9,x,a,…f}

V_N = {S, H, C}

P = {S□0xH, H□HC, H□C, C□0, C□1, … , C□9, C□a, …, C□f}

Стандартная запись грамматики

G: S □ 0xH

H □ HC | C

C □ 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f

Пример вывода

S □ 0xH □ 0xHC □ 0xHCC □ 0xCCC -> 0xaCC □ 0xa8C □ 0xa8f

Обозначение нетерминалов

Вариант 1 - большие латинские буквы
(при условии их отсутствия в V_T)

G: S → 0xH
H → HC | C
C → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a |b | c | d | e | f

Вариант 2 – текст в угловых скобках

G: <число 16> → 0x <цифры 16>
<цифры 16> → <цифры 16> <одна цифра 16>
| <одна цифра 16>
<одна цифра 16> → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
| a |b | c | d | e | f

Правила КС-грамматики

$G = (V_T, V_N, P, S)$

конструкция правила

<нетерминал> → <цепочка из V_T и V_N >

Пример

S → SaB | c

B → xBy | ε ε - пустая цепочка, | ε | = 0

Пример вывода

S → SaB → SaBaB → caBaB → caaB → caaxBy → caaxxByy → caaxxxByyy → caaxxxyyy

Вывели $ca^2x^3y^3$

КС-грамматика и дерево вывода

Правила вывода в КС-грамматике $A \to \phi$

$A \in V_N$, $\phi \in (V_T \cup V_N)^*$

G: S → aSAb | c

A → cA | d

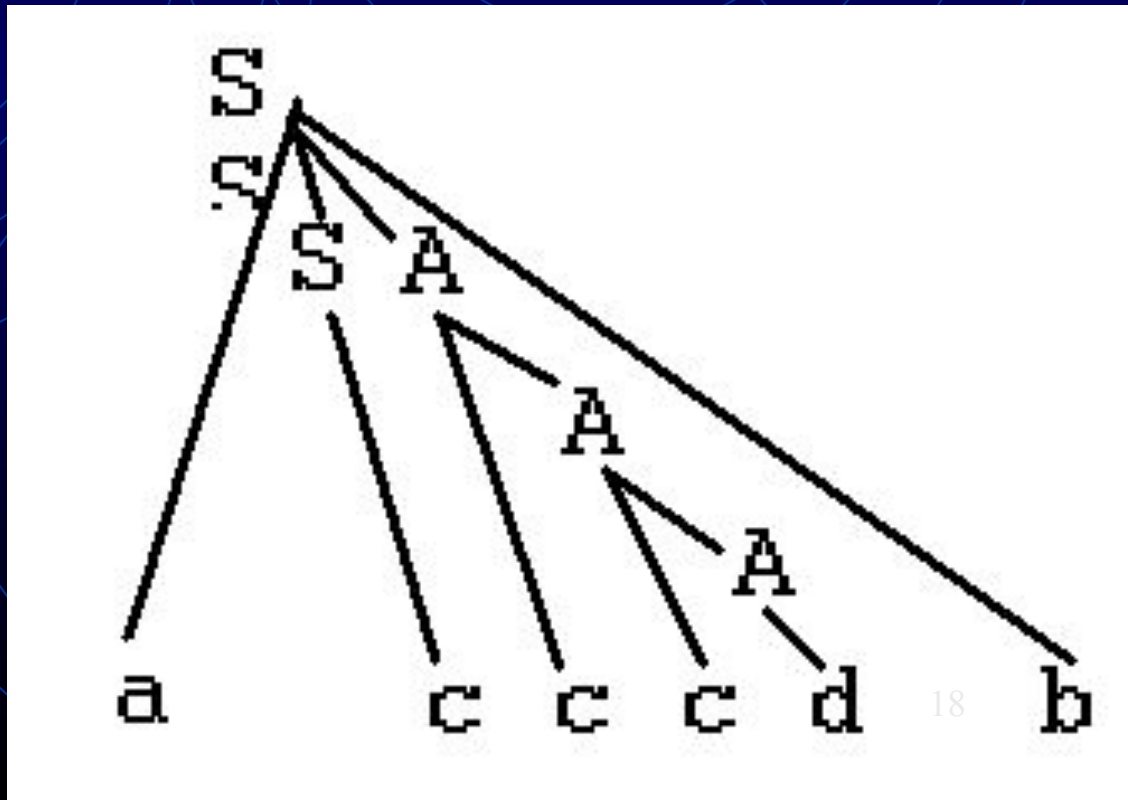
Вывод

S →aSAb → acAb

→accAb

→ acccAb→ acccdb

Дерево вывода



Операции над языками

1) Обычные теоретико-множественные:

Объединение $L_1 \cup L_2$

Пересечение $L_1 \cap L_2$

Разность $L_1 \setminus L_2$

2) Специальные:

Произведение или конкатенация $L_1 L_2$

Возведение в степень L^n,

Итерация L^*,

Усеченная итерация L^+

Операции – инструмент синтеза КСГ

Теорема

Семейство КС-языков замкнуто относительно операций объединения, произведения, итерации и усеченной итерации.

Метод синтеза КСГ

1) Есть КСГ G_1 и G_2 простых языков L_1 и L_2
2) Строим нужную нам КСГ как вариант из $L_1 \cup L_2$, $L_1 L_2$, L_2^*, L_2^+

Доказательство для L_1 L_2

Конкатенация (умножение)

Дано:

$G_1 = (V_{T1}, V_{N1}, P_1, S_1)$, $G_2 = (V_{T2}, V_{N2}, P_2, S_2)$

Нужна грамматика языка $L(G) = L(G_1)\, L(G_2)$

Решение:

$V_{N1} \cap V_{N2} = \varnothing$

Новая грамматика с новым нетерминалом S

$P = P_1 \cup P_2 \cup \{S \to S_1 S_2\}$

Вывод в G

$S \to S_1 S_2 \to \alpha S_2 \to \alpha \beta$

α выводится в G_1, β выводится в G_2

Доказательство для $L_1 \cup L_2$

Объединение

Дано:

$G_1 = (V_{T1}, V_{N1}, P_1, S_1)$, $G_2 = (V_{T2}, V_{N2}, P_2, S_2)$

Нужна грамматика языка $L(G) = L(G_1) \cup L(G_2)$

Решение:

$V_{N1} \cap V_{N2} = \varnothing$

Новая грамматика с новым нетерминалом S

$P = P_1 \cup P_2 \cup \{S \to S_1 \mid S_2\}$

Выводы в G (два варианта, так как два привила)

$S \to S_1 \to \alpha$ или $S \to S_2 \to \beta$

α выводится в G_1, β выводится в G_2

Доказательство для L^*

Итерация

Дано: $G_1 = (V_{T1}, V_{N1}, P_1, S_1)$

Нужна грамматика языка $L(G) = L(G_1)^*$

Решение:

Новая грамматика с новым нетерминалом S

$P = P_1 \cup \{S \to S\,S_1,\ S \to \varepsilon\}$

Выводы в G

$S \to \varepsilon$

$S \to SS_1 \to S_1 \to \beta$

$S \to SS_1 \to SS_1S_1 \to S_1S_1 \to \ldots$

$S \to SS_1 \to SS_1S_1 \to SS_1S_1S_1 \to S_1S_1S_1 \to \ldots$

Правая рекурсия для L*

Итерация

Дано: $G_1 = (V_{T1}, V_{N1}, P_1, S_1)$

Нужна грамматика языка $L(G) = L(G_1)^*$

Решение:

Новая грамматика с новым нетерминалом S

$P = P_1 \cup \{S \to S_1 S, \; S \to \varepsilon\}$

Выводы в G

$S \to \varepsilon$

$S \to S_1 S \to S_1 \to \beta$

$S \to S_1 S \to S_1 S_1 S \to S_1 S_1 \to \ldots$

$S \to S_1 S \to S_1 S_1 S \to S_1 S_1 S_1 S \to S_1 S_1 S_1 \to \ldots$

Доказательство для L^+

Усеченная итерация

Дано: $G_1 = (V_{T1}, V_{N1}, P_1, S_1)$

Нужна грамматика языка $L(G) = L(G_1)^+$

Решение:

Новая грамматика с новым нетерминалом S

$P = P_1 \cup \{S \to S S_1, \; S \to S_1\}$

Выводы в G

$S \to S_1 \to \beta$

$S \to SS_1 \to S_1 S_1 \to \ldots$

$S \to SS_1 \to SS_1S_1 \to S_1S_1S_1 \to \ldots$

…

Правая рекурсия для L^+

Усеченная итерация

Дано: $G_1 = (V_{T1}, V_{N1}, P_1, S_1)$

Нужна грамматика языка $L(G) = L(G_1)^+$

Решение:

Новая грамматика с новым нетерминалом S

$P = P_1 \cup \{S \to S_1 S,\ S \to S_1\}$

Выводы в G

$S \to S_1 \to \beta$

$S \to S_1 S \to S_1 S_1 \to \ldots$

$S \to S_1 S \to S_1 S_1 S \to S_1 S_1 S_1 \to \ldots$

…

Пример: язык (ab)*c ∪ b+

G: S → A | B // объединение (ab)*c и b+

A → TF // произведение (ab)* и c

F → c

T → Tab | ε // итерация ab

B → Bb | b // усеченная итерация b

Можно проще:

G: S → A | B

A → Tc

T → Tab | ε

B → Bb | b

Или еще проще*

G: S → Tc | B

T → Tab | ε

B → Bb | b

Пример: идентификатор C++

Идентификатор – это последовательность букв и цифр, начинающаяся с буквы. Знак подчеркивания рассматривается как буква.

G: <идент> → <буква> <окончание>

<буква> → a |b | … | z | A | B | … | Z | _

<окончание> → <окончание> <буква>

| <окончание> <цифра>

<цифра> → 0 | 1 | 2 | … | 9

Симметрия в языке

$L = \alpha^n \gamma \beta^n \;,\quad n >= 0$

$S \rightarrow \alpha S \beta \mid \gamma$

Вывод

$S \Rightarrow \alpha S \beta \Rightarrow \alpha^2 S \beta^2 \Rightarrow \alpha^3 S \beta^3 \Rightarrow \ldots \Rightarrow \alpha^n S \beta^n \Rightarrow \alpha^n \gamma \beta^n$

Не путать с $L = \alpha^* \gamma \beta^*$

$S \rightarrow A \gamma B$

$A \rightarrow A\,\alpha \mid \varepsilon$

$B \rightarrow B \beta \mid \varepsilon$

Простой пример симметрии

Симметричные конструкции

$S \rightarrow ASB \mid X$

$L(S) = L(A)^n \, L(X) \, L(B)^n \qquad n >= 0$

Пример

$(ab)^n \, c \, (xyz)^n \qquad n>=0$

$S \rightarrow abSxyz \mid c$

Пример симметрии

$L = a^{2n} d (a^{\pm} b^* c)^{n+1}, \quad n >= 0$

1. Выделим симметрию

$L = a^{2n} d (a^{+} b^* c)^{n+1}$

$= (aa)^n \; d \; a^{+} b^* c \; (a^{\pm} b^* c)^n$

2. Строим КС-грамматику

G: S → aaSB | dB

B → THc

T → Ta | a

H → Hb | ε

Преобразования КСГ

1. Подстановки
2. Новые нетерминалы для фрагментов
3. Неукорачивание
4. Левая рекурсия правая
5. Правая рекурсия левая
6. Удаление одинаковых правых частей правил
7. Удаление правил «нетерминал нетерминал»
8. Удаление лишних нетерминалов
9. …

Левая рекурсия правая

Были правила

S SA | B

Вывод:

S SA SAA SA* BA*

Новые правила

S BH

H AH | ε

Новый вывод

S BH BAH BAAH … BA*

Левая рекурсия правая (Общий случай)

Были правила

$S \to S\beta_1 \mid S\beta_2 \mid \ldots \mid S\beta_k \mid \phi_1 \mid \phi_2 \mid \ldots \mid \phi_m$

Преобразуем :

$S \to SX \mid Y$ (будет $S \to Y H$ $H \to X H \mid \varepsilon$)

$X \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_k$

$Y \to \phi_1 \mid \phi_2 \mid \ldots \mid \phi_m$

Новые правила строим так же, как и ранее

$S \to \phi_1 H \mid \phi_2 H \mid \ldots \mid \phi_m H$

$H \to \beta_1 H \mid \beta_2 H \mid \ldots \mid \beta_k H \mid \varepsilon$

Правая рекурсия левая (Общий случай)

Были правила

$S \to \beta_1 S \mid \beta_2 S \mid \ldots \mid \beta_k S \mid \phi_1 \mid \phi_2 \mid \ldots \mid \phi_m$

Преобразуем :

S -> XS | Y (будет $S \to HY$ $H \to HX \mid \varepsilon$)

$X \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_k$

$Y \to \phi_1 \mid \phi_2 \mid \ldots \mid \phi_m$

Новые правила строим так же, как и ранее

$S \to H\phi_1 \mid H\phi_2 \mid \ldots \mid H\phi_m$

$H \to H\beta_1 \mid H\beta_2 \mid \ldots \mid H\beta_k \mid \varepsilon$

Рекурсия

Грамматика конечна по опреледению.
Язык, как правило, бесконечен.

Рекурсия – единственный способ представить бесконечный язык конечными средствами.

Типы рекурсии:

- Левая A □ Ab,
- Правая A -> bA,
- Центральная A □ aAb,
- Неявная A □ xBy B □ uDv D □ aAb .

Рекурсия конструкций ЯП

<оператор> → <простой оператор>
| <составной оператор>
| <пустой оператор>

<пустой оператор> → ;

<составной оператор> → {<опер и описания>}

<простой оператор> → <if> |<while> | <for> |…

<if> → if (<выражение>) <оператор>
|if (<выражение>) <оператор> else <оператор>

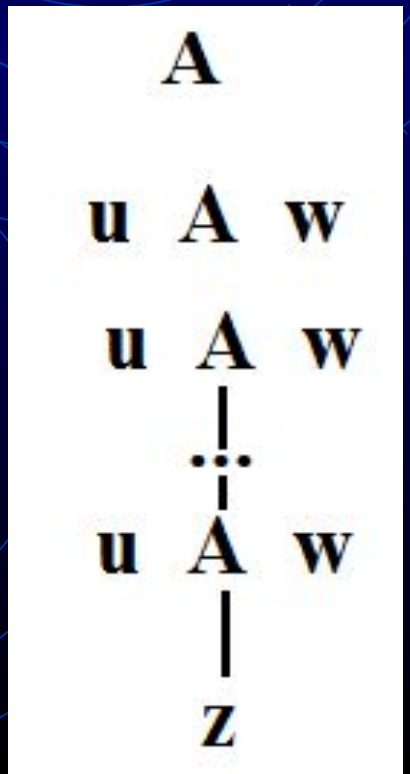
<while> → while (<выражение>) <оператор>

<опер и описания> → <опер и описания> < оператор >
| <опер и описания> < описание >
| ε

Синтаксический анализ

Задача любого транслятора – построить дерево грамматического разбора.

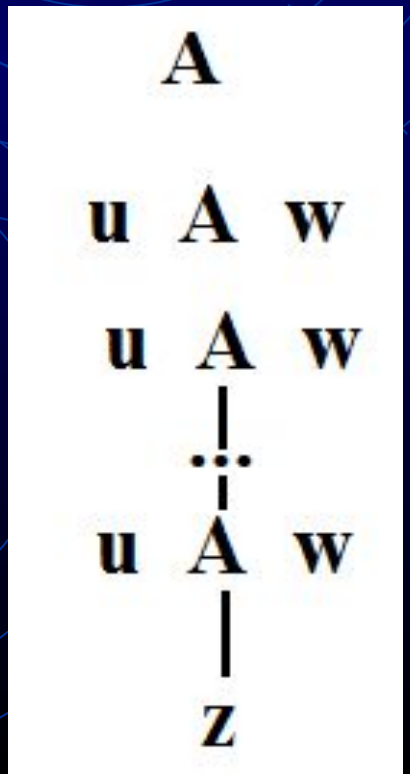
Если язык бесконечен, а КС-грамматика конечна, то в дереве СА достаточно длинной цепочки обязательно на некотором пути из корня в лист повторится некоторый нетерминал $A \Rightarrow u^nAw^n \Rightarrow u^nzw^n$



Лемма о разрастании

Для любой КС - грамматики, порождающей бесконечный язык, существуют такие натуральные числа p и q, что каждая цепочка $\gamma \in L(G)$, $|\gamma| > p$ может быть представлена в виде $\gamma = xuzwy$, где $|uw| > q$ и для любого $n > 0$ цепочка $x u^n z w^n y \in L(G)$.

Мы получили на предыдущем слайде нетерминал $A \Rightarrow u^n A w^n \Rightarrow u^n z w^n$



Следствие – теорема о языке $a^n b^n c^n$

Теорема

Язык $a^n b^n c^n$ не является КС-языком.

Вывод: КС-грамматика не может синхронизировать более двух фрагментов.

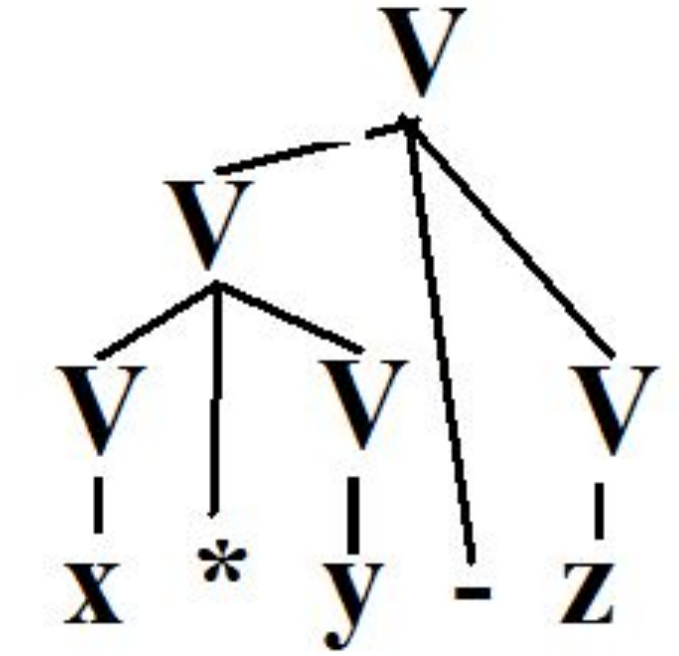
Пример

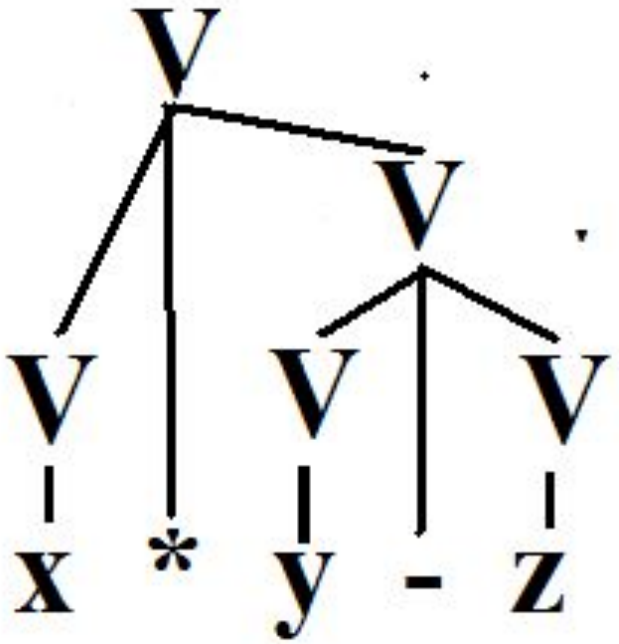
```
int aaaa;
        // идентификаторы состоят только из букв a
int main() {
        cin >> aaaa;
        cout << aaaa * 2;
}
```

Недетерминированность КСГ недопустима !!!

Пусть V == <выражение> a == <идент>

G: V → V+V | V-V | V*V | V/V | (V) | a





Выражения

Нет понятия логического, арифметического и т.п. выражения:

Работает семантика контекстных условий для контроля корректности выражений:

```
int x, z[3][100],  T[100];
bool y;
x[i] = z * y(6.7) – 5;   // семантические ошибки
T[i] = x * sin(6.7) – 5;  // верная конструкция
double  a = x – 7.78;     // приведение типов
```

Синтаксис выражений

1 Упорядочить группы операций по приоритетам, начиная с низкоприоритетных .
2 Каждой группе поставить в соответствие нетерминал A1, A2, A3, … ,An
3 Для каждого нетерминала Ai записать правила для каждой операции группы(см. следующий слайд)
4 Добавить правила, соответствующие неприменению операции
Ai Ai+1
5 Определить правила для элементарного выражения An+1

Правила для группы многократных операций

Операции многократные, т.е. разрешается запись a*a*a*a

Операция бинарная

выполнение слева направо Ai Ai * Ai+1

справа налево Ai Ai+1 * Ai

Операция префиксная унарная Ai * Ai

постфиксная унарная Ai Ai *

Правила для группы однократных операций

Операции однократные, т.е. НЕ разрешается запись a*a*a*a

Операция бинарная

Ai Ai +1* Ai+1

Операция префиксная унарная Ai * Ai+1

постфиксная унарная Ai Ai +1*

Пример выражения с бинарными операциями

A1 > < ==
A2 + -
A3 * /

A1 → A1 > A2 | A1 < A2 | A1 == A2 | A2
A2 → A2 + A3 | A2 – A3 | A3
A3 → A3 * A4 | A3 / A4 | A4
A4 → <идентификатор> | <константа> | (A1)

Лабораторная работа №1

1. Построить КС-грамматику по заданию в учебнике (№ задания == номер в списке группы)
2 . Построить деревья разбора для двух разных цепочек

Учебное пособие (содержит теоретический материал и пример выполнения задания):

- глава 1 - материал по грамматикам

Лабораторная работа № 2

1. Построить таблицу приоритетов операций языка программирования индивидуального задания
2. Построить КС-грамматику языка программирования индивидуального задания

Учебное пособие (содержит теоретический материал и пример выполнения задания):

- глава 3

Языки, грамматика, автоматы

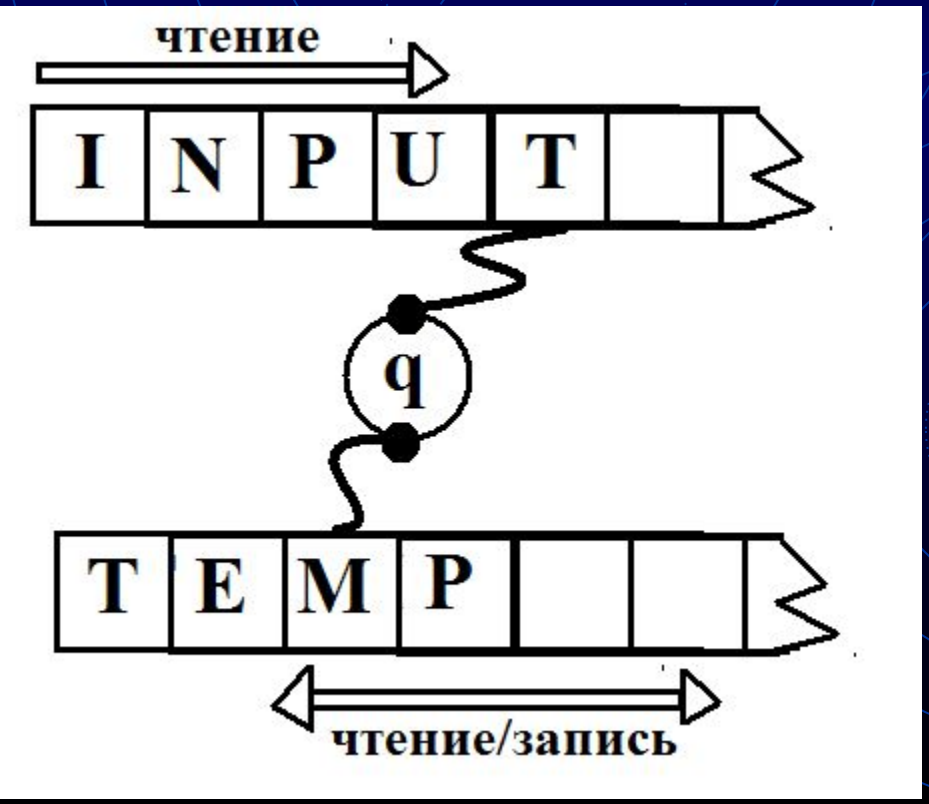
1. Грамматика порождает язык
2. Автомат распознает язык
3. Программист использует (1) при создании программы
4. Транслятор использует (2) в процессе работы
5. Вывод: (1) и (2) должны быть эквивалентны

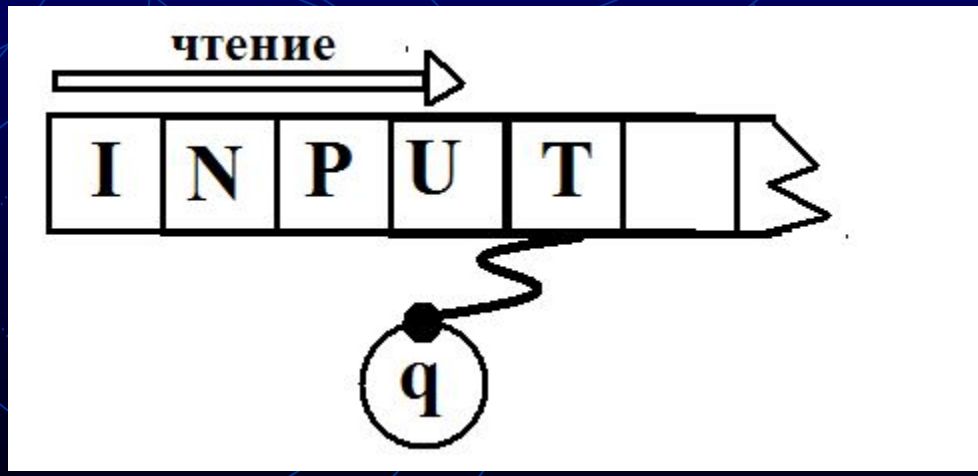
Разработчик транслятора создает грамматику, выбирает метод распознавания, реализует метод распознавания в соответствии с грамматикой

Типы автоматов при трансляции

Машина Тьюринга == любой алгоритм.
Требование трансляции - эффективность

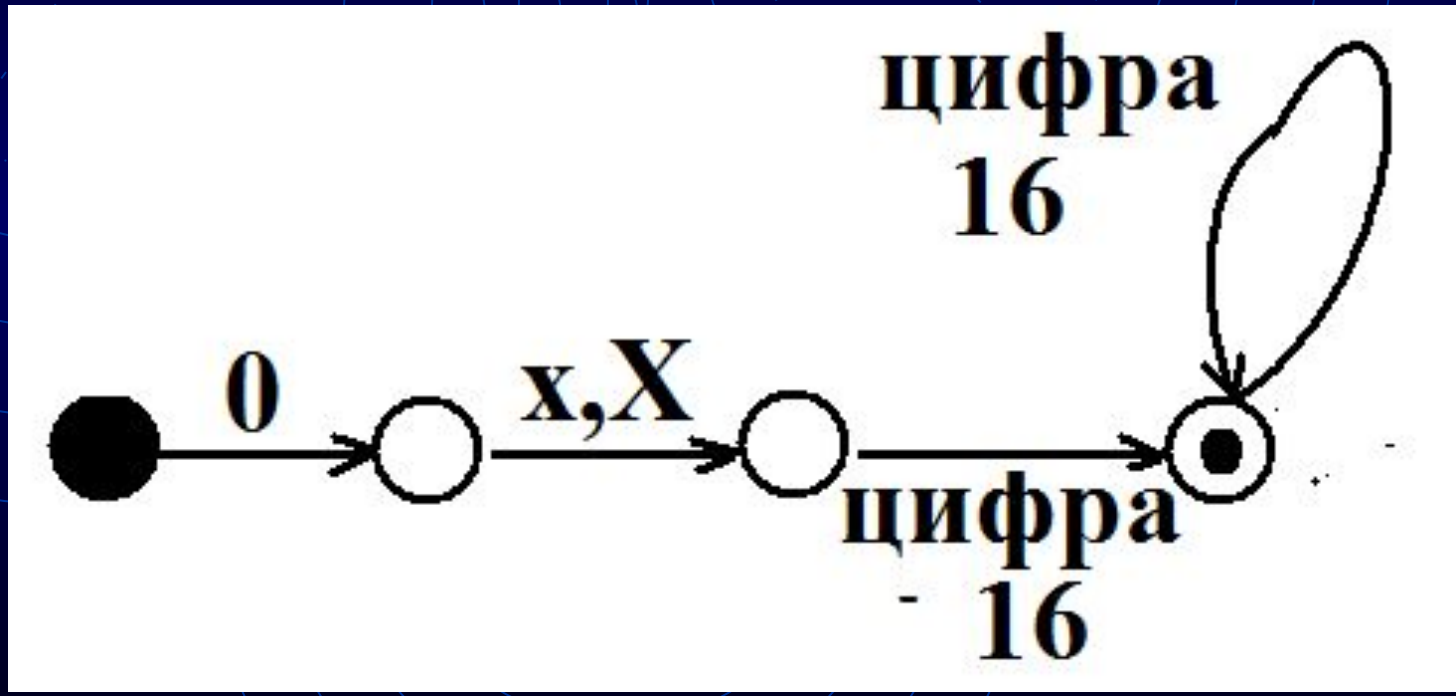
$$T(v) = O(n)$$





Автоматы при трансляции

Конечный автомат лексика языка
(переходит из состояния в состояние, читая символы)

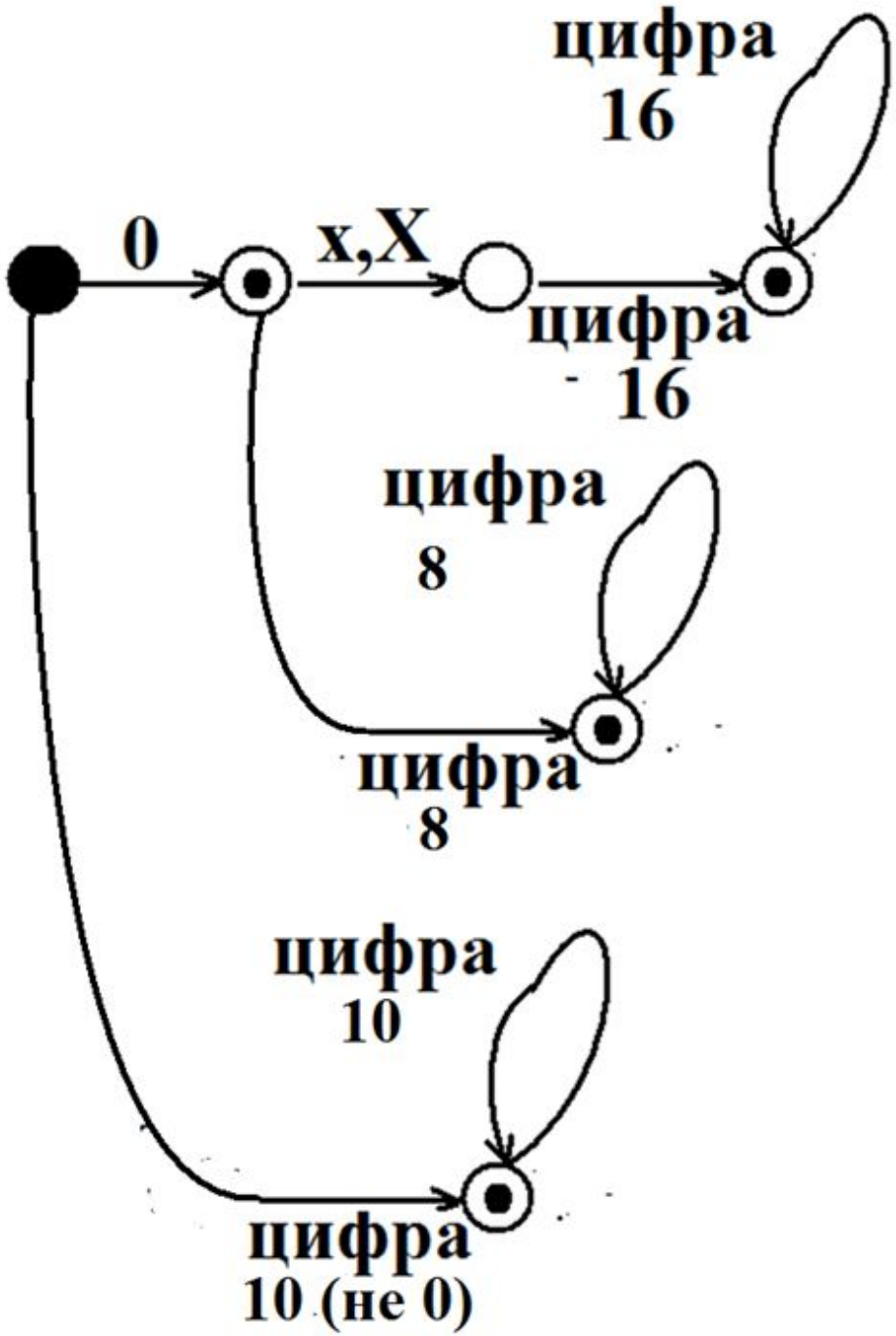


МП-автомат (автомат с магазинной памятью синтаксис языка)

Синтез конечных автоматов

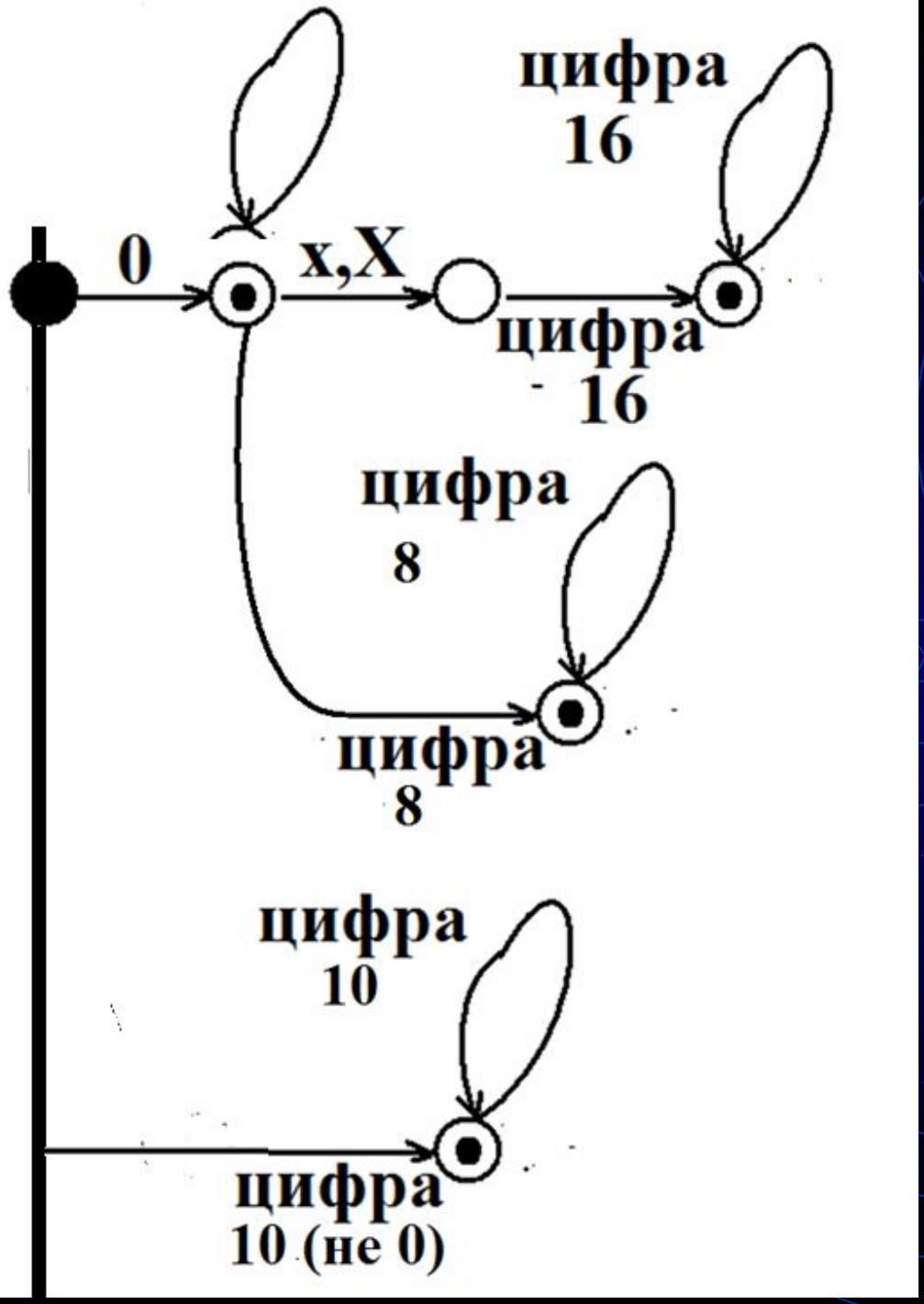
Учебное пособие (содержит теоретический материал и примеры):

- глава 2



Целые константы:
1) 8с/с – начинается с 0,
2) 16с/с – начинаются 0x или 0X,
3) 10 с.с – начинаются с цифры, не равной нулю

Целые константы



The end

Лабораторные работы далее – лексический анализ