Лекция 3

Распознавание формальных языков

Метасинтаксис: EBNF

Extended Backus-Naur Form -
“синтаксис для записи синтаксиса” (КС-грамматики)

левая часть = правая часть или
левая часть ::= правая часть

Принятые обозначения

Нетерминалы:	*Имена*	или	<Имена>
Терминалы:	строки	или	строки в кавычках

Варианты записи рег. выражений (в правой части)

[ ] - необязательные составляющие
{ } - замыкание Клини
| - альтернативы

Метасинтаксис: EBNF

примеры

integer = *digit* {*digit*} | 0x*hexDigit*{*hexDigit*}

hexDigit = *digit*|A|B|C|D|E|F|a|b|c|d|e|f

digit = 0|1|2|3|4|5|6|7|8|9

<QualIdent> ::= [<Classname> “.”] <identifier>

ArrayCreationExpr =

new *Typename* “[“*integer* “]“ { “[“ *integer* “]“ }

Метасинтаксис: EBNF

сокращение письма

EBNF:

```
integer = digit {digit} | 0xhexDigit{hexDigit}
```

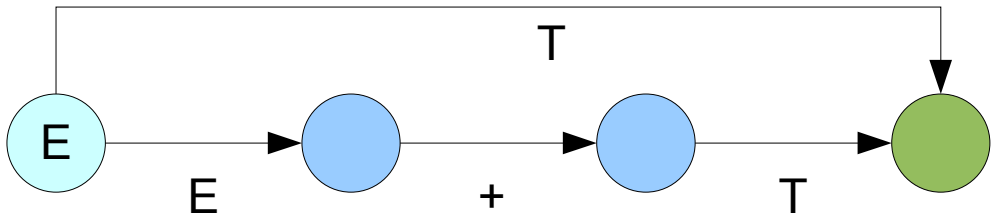
Математическая нотация:

```
Integer   → Digits    | 0xHexdigits
Digits    → Digit     | Digit Digits
HexDigits → HexDigit  | HexDigit HexDigits
```

Метасинтаксис:
диаграммные грамматики

G = < Σ, N, S, P>, где P = {<D, E>} – множество аттрибутированных оргграфов (*диаграммы Вирта*)

вершины D (пометки “стартовый нетерминал”, ”конец”)
ребра E (пометки из Σ ∪ N)



$$E \rightarrow E+T \mid T$$

- вершины показывают состояние разбора
- диагр. грамматики полезны при проектировании лексических анализаторов

Метасинтаксис: разное

Нотация, использованная в Java Language Specification:

ClassDeclaration:
  ClassModifiers$_{opt}$ `class` *Identifier Super*$_{opt}$ *Interfaces*$_{opt}$ *ClassBody*

ClassModifiers:
  ClassModifier
  ClassModifiers ClassModifier

ClassModifier: *one of*
  `public protected private`
  `abstract static final strictfp`

Вывод разверткой

G = < Σ, N, S, P> - грамматика

Сентенциальная форма α: $S \Rightarrow^* \alpha$ $\qquad\qquad$ $\alpha \in (\Sigma \cup N)^*$

(в общем случае не является словом из L(G), т.к. может содержать нетерминалы - “незаконченный вывод”)

Левосторонний вывод (разверткой) $\Rightarrow_l$

(в сентенциальной форме заменяется самый левый нетерминал)

Правосторонний вывод (разверткой) $\Rightarrow_r$

(в сентенциальной форме заменяется самый правый нетерминал)

Вывод разверткой
примеры

G = < { **a**, **+**, *, **(**, **)** }, {E}, E, { E→ E+E | E*E | (E) | a } >

Выводы разверткой для цепочки a+a*a

левосторонний:

$$E \Rightarrow_l E+E \Rightarrow_l a+E \Rightarrow_l a+E*E \Rightarrow_l a+a*E \Rightarrow_l a+a*a$$

правосторонний:

$$E \Rightarrow_r E+E \Rightarrow_r E+E*E \Rightarrow_r E+E*a \Rightarrow_r E+a*a \Rightarrow_r a+a*a$$

произвольный:

$$E \Rightarrow E*E \Rightarrow E*a \Rightarrow E+E*a \Rightarrow a+E*a \Rightarrow a+a*a$$

Дерево разбора

$G = < \Sigma, N, S, P >$ - грамматика

Дерево синтаксического разбора для G – корневое ориентированное упорядоченное помеченное дерево:

- корень помечен S
- нелистовые вершины помечены нетерминалами из N
- листья помечены ε или терминалами из Σ

таким образом что

потомки вершины, помеченной нетерминалом A, образуют упорядоченное множество $B_1, \ldots, B_n$, и $A \to B_1 \ldots B_n$ - правило из G.

Дерево разбора
пример

Правила грамматики

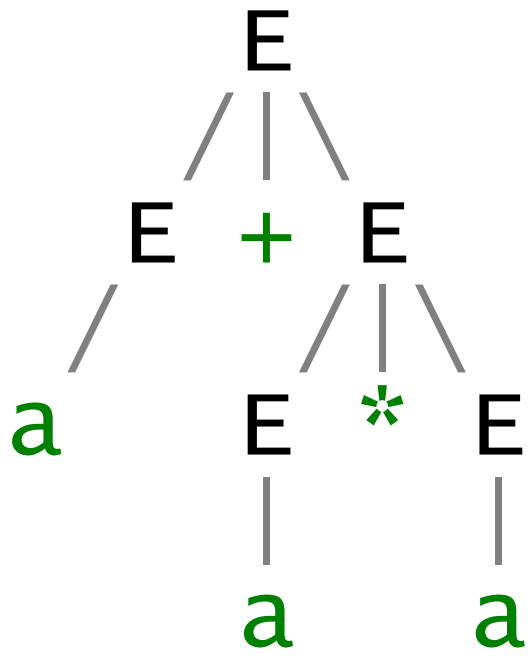
$E \rightarrow E+E$

$E \rightarrow E*E$

$E \rightarrow (E)$

$E \rightarrow a$

Дерево разбора



Дерево разбора
сечение и крона

Сечение дерева – цепочка пометок вершин, удовлетворяющая условиям:

1. каждый путь из корня к некоторому листу содержит один и только один узел, пометка которого входит в сечение

2. пометки выписаны в порядке прямого левостороннего обхода дерева в глубину

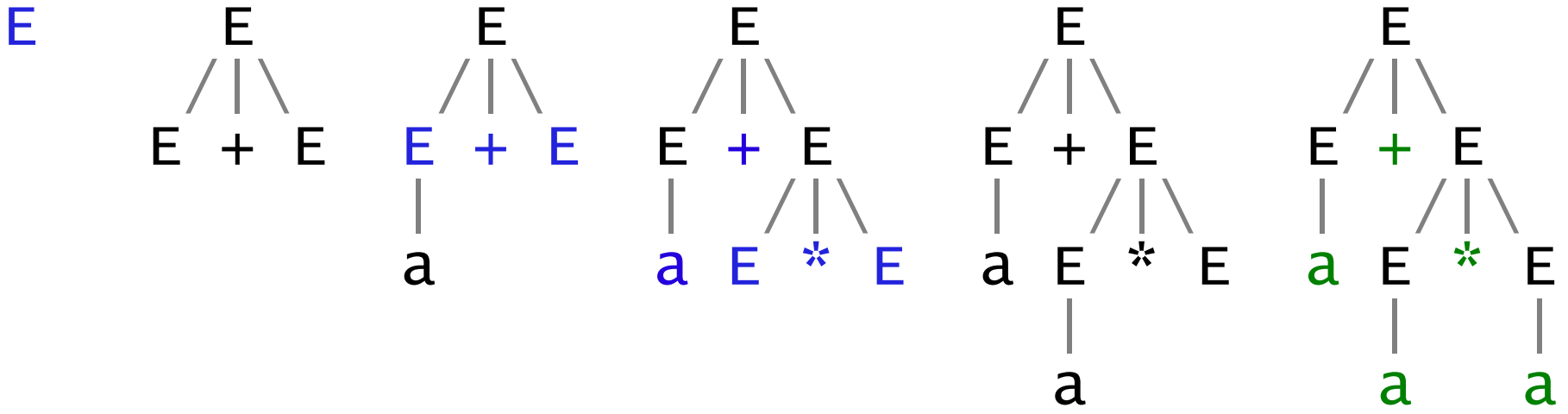
Крона дерева – сечение, содержащее только пометки листьев

Возможно ли реализовать обход дерева без рекурсии?

Дерево разбора
вывод

Цепочка ω *выводима деревом в* G, если существует дерево разбора с корнем *S* и кроной ω

Вывод деревом для a+a*a:

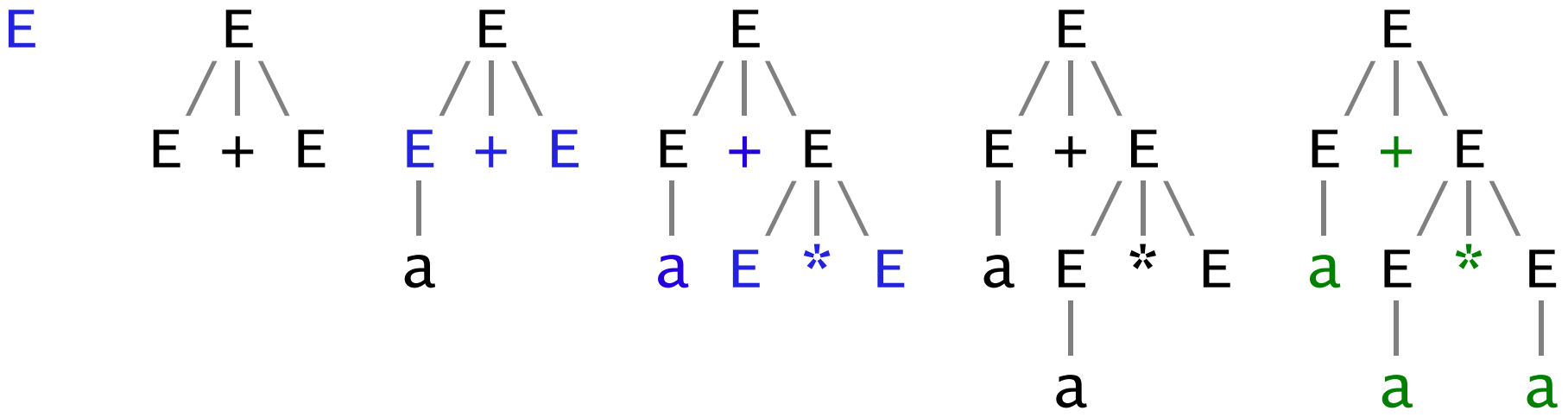


сечение
крона

Дерево разбора
эквивалент выводимости

Выводимость деревом	≈	Выводимость разверткой в G
Сечение дерева	≈	Сентенциальная форма
Крона	≈	Слово из L(G)

Вывод деревом для a+a*a:



сечение
крона

Неоднозначные грамматики

Грамматика *неоднозначна*, если для некоторой выводимой цепочки существуют два различных дерева разбора.

 Неоднозн. грамматики создают трудности для реализации:

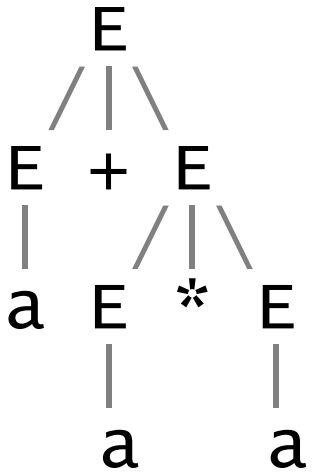
> ***Указ***
>
> *Казнить нельзя помиловать.*

***Теорема**. Множество однозначных грамматик неразрешимо (доказывается сведением к проблеме Поста)*

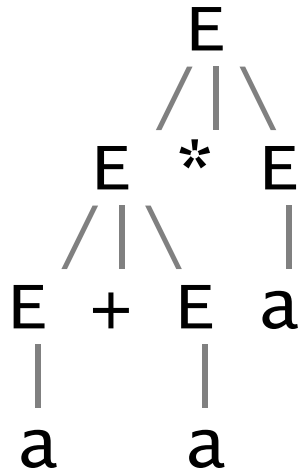
Неоднозначные грамматики
примеры

G из предыдущего примера неоднозначна:

(1)



(2)



Задают выражения:

```
a+(a*a)    (1)
(a+a)*a    (2)
```

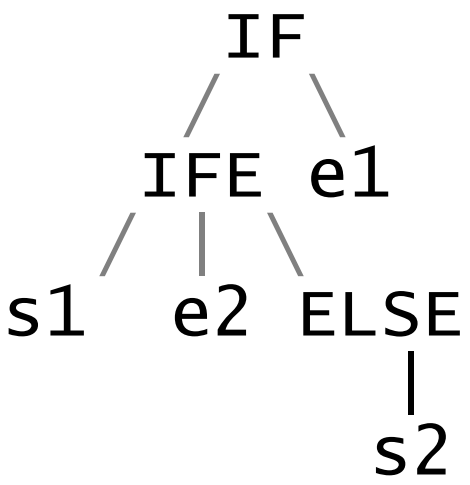
! дерево (2) нарушает принятый приоритет операций

Неоднозначные грамматики
примеры

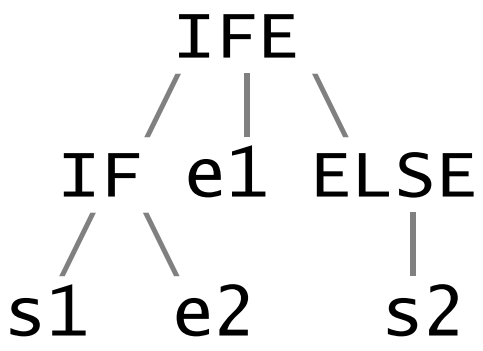
```
if (e1)
  if (e2)
    ++i; //s1
  else
    --i; //s2
```

или `if(e1)if(e2)++i;else--i;`

(1)



(2)



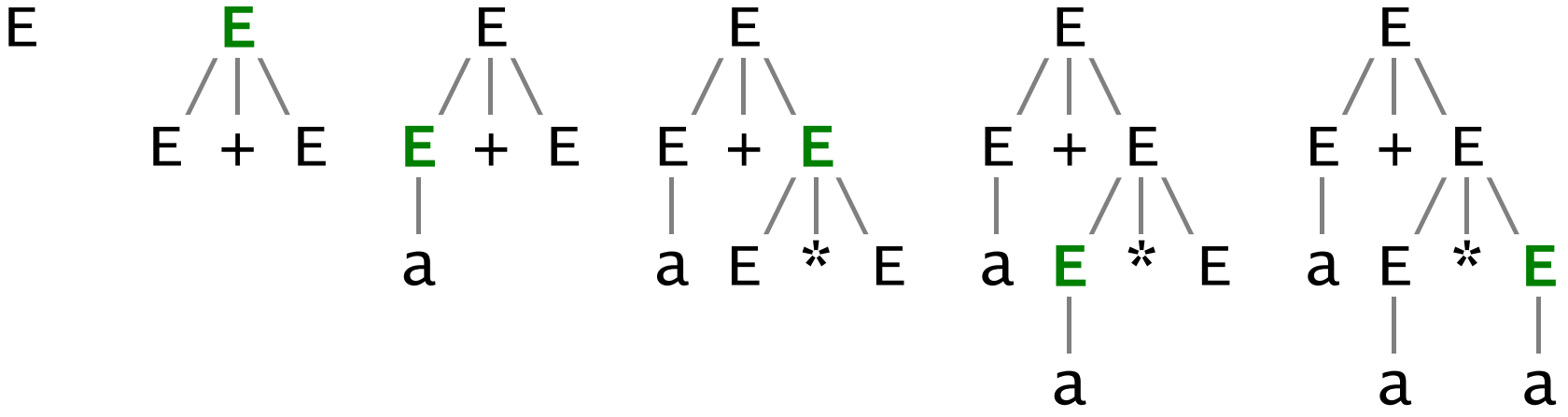
Какое из деревьев “правильное”?

Нисходящий разбор

Нисходящий разбор цепочки ω – построения дерева разбора для ω:

1. начинаем с корня (начальный нетерминал грамматики)
2. на каждом шаге добавляются все потомки одной вершины, помеченной нетерминалом

По завершении крона построенного дерева совпадает с ω



Метод рекурсивного спуска

(recursive-descent parsing) для регулярных грамматик

Для каждого нетерминала, выписываем правило вида:

$$A \rightarrow a_1B \mid a_2C \mid \ldots \mid a_kM \mid \mathbf{b_1} \mid \mathbf{b_2} \mid \ldots \mid \mathbf{b_n}$$

все a_i, b_i - различны - можно привести (почему?)

```
procedure ParseA;      -- для каждого нетерминала A из G
begin
  select Scanner() of -- чтение символа из потока
    {a1}:  ParseB;
    {a2}:  ParseC;
     ...
    {b1,b2,...,bn}: ;  -- нет действий
  else
    Error ("Ошибка разбора A");
  end select;
end;
```

Метод рекурсивного спуска

для регулярных грамматик

Работа рекурсивного анализатора:

1. вызываем `ParseS` для начального нетерминала

2. каждая `Parse`-процедура:

   - либо выбирает правило (и, возможно, вызывает процедуру для нетерминала из правой части)

   - либо выдает ошибку

Дерево вызовов изоморфно дереву разбора (без кроны)

- на каждом шаге считывается 1 символ
- возвратов при разборе входного потока нет
- “заглядывания” вперед нет

☺ Временная сложность $O(|\omega|)$

Метод рекурсивного спуска?

для КС-грамматик

***Теорема**. Регулярные грамматики однозначны.*

***Замечание**. КС-грамматики в общем случае неоднозначны.*

***Наблюдение.** С разбором КС-грамматик есть проблемы*

1. неоднозначный вывод
2. нетерминалы в правой части: $A \to B\beta$ (особенно $A \to A\beta$)
3. недетерминированность: $A \to B\beta \mid B\gamma$

? Что делать?

Разбор КС-грамматик

Универсальные алгоритмы:

Алгоритм (Кока-Янгера-Кесами). Реализует разбор для произвольных КС-грамматик

 временная сложность $O(|\omega|^3)$

Алгоритм (Эрли). Делает то же самое.

 временная сложность $O(|\omega|^2)$

Рекурсивный разбор с возвратами

 есть риск “комбинаторного взрыва”

Рекурсивный разбор без возвратов

? для подкласса КС-грамматик

k-префиксы

(helping hand for context-free grammars)

Множество k-префиксов всех терминальных цепочек, выводимых **из** $\alpha \in (\Sigma \cup N)^*$:

$$First_k(\alpha) =_{def} \{ x \in \Sigma^* \mid (\alpha \Rightarrow^* x\beta, |x| = k) \vee (\alpha \Rightarrow^* x, |x| < k)\}$$

Множество k-префиксов всех терминальных цепочек, выводимых **после** $A \in N$:

$$Follow_k(A) =_{def} \{ x \in \Sigma^* \mid S \Rightarrow^+ \upsilon A\beta \wedge x \in First_k(\beta)\}$$

Множество *k-префиксов* выводимых **из** A с **помощью правила** $A \rightarrow \beta$:

$$First_k(A \rightarrow \beta) =_{def} First_k(\beta Follow_k(A))$$