БУ ВО Ханты-Мансийского автономного округа – Югры
«Сургутский государственный университет»

Теория языков программирования и методы трансляции

Часть 4

- Методы грамматического анализа
- Грамматический анализ регулярных языков
- Построение конечного автомата

Гришмановский Павел Валерьевич
доцент кафедры автоматики и компьютерных систем, к.т.н., доцент

Сургут, 2019

Методы грамматического анализа

Грамматический анализ (разбор) – определение принадлежности некоторой цепочки символов языку, порождаемому грамматикой

Алгоритмы грамматического анализа связаны с классами грамматик:

- Конечные автоматы – регулярные грамматики
  - Автоматные грамматики
- Алгоритмы восходящего анализа (LR) – детерминированные КС-грамматики
  - Алгоритмы без возвратов – грамматики предшествования
  - Алгоритмы с «заглядыванием вперед» (LALR) – подклассы детерминированных и недетерминированных КС-грамматик
- Алгоритмы нисходящего анализа (LL) – подклассы детерминированных КС-грамматик
  - Алгоритм рекурсивного спуска
- Алгоритмы с возвратом – недетерминированные КС-грамматики
  - МП-автоматы
  - Рекурсивные алгоритмы или параллельные вычисления

Методы грамматического анализа

Классификация алгоритмов анализа:

XY(k)

X – направление чтения цепочки символов

- L – слева направо (обычное)
- R – справа налево (не имеет практического смысла)

Y – вид восстанавливаемого вывода

- L – левосторонний (в прямом порядке)
- R – правосторонний (в обратном порядке)

k – количество (максимальное) считанных и одновременно сравниваемых символов

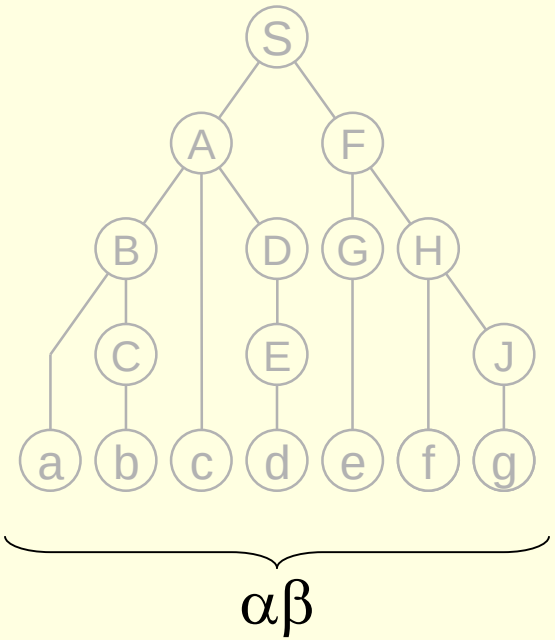
Практический интерес представляют **LR(k)**- и **LL(k)**-алгоритмы анализа, особенно при k = 1

Виды грамматического анализа (пример)

Нисходящий (LL)

Левосторонний вывод,
«сверху вниз»,
прямой порядок

Правило	$\alpha \in V_T^*$,	$\beta \in V^*$
	λ	S
$S \to AF$	λ	AF
$A \to BcD$	λ	BcDF
$B \to aC$	a	CcDF
$C \to b$	abc	DF
$D \to E$	abc	EF
$E \to d$	abcd	F
$F \to GH$	abcd	GH
$G \to e$	abcde	H
$H \to fJ$	abcdef	J
$J \to g$	abcdefg	λ



Восходящий (LR)

Правосторонний вывод,
«снизу вверх»
обратный порядок

$\alpha \in V^*$,	$\beta \in V_T^*$	Правило
S	λ	$S \to AF$
AF	λ	$F \to GH$
AGH	λ	$H \to fJ$
AGfJ	λ	$J \to g$
AG	fg	$G \to e$
A	efg	$A \to BcD$
BcD	efg	$D \to E$
BcE	efg	$E \to d$
B	cdefg	$B \to aC$
aC	cdefg	$C \to b$
λ	abcdefg	

Методы грамматического анализа

Доказано, что:

- Любая LR(k)- и LL(k)-грамматика для k > 0 однозначна
- Для произвольной грамматики G и строго заданного k > 0 можно доказать, является ли она LR(k)- или LL(k)-грамматикой
- Класс LR(k)-грамматик эквивалентен множеству детерминированных КС-языков
- Класс LL(k)-грамматик ýже класса LR(k)-грамматик
- Не существует алгоритма отыскания k для произвольной грамматики G (определения ее принадлежности к классам LR(k)- и LL(k)-грамматик)
- Не существует алгоритма преобразования произвольной грамматики G к грамматике, принадлежащей классу LR(k)- или LL(k)-грамматик
- Любая LR(k)- или LL(k)-грамматика для k > 0 также принадлежит классу LR(n)- или LL(n)-грамматик для n > k

Преобразования грамматик

Цели преобразований:

- **Формальный анализ** – определение или облегчение определения свойств грамматики и порождаемого ею языка
- **Построение транслятора** – обеспечение возможности или упрощение применения определенного метода грамматического анализа

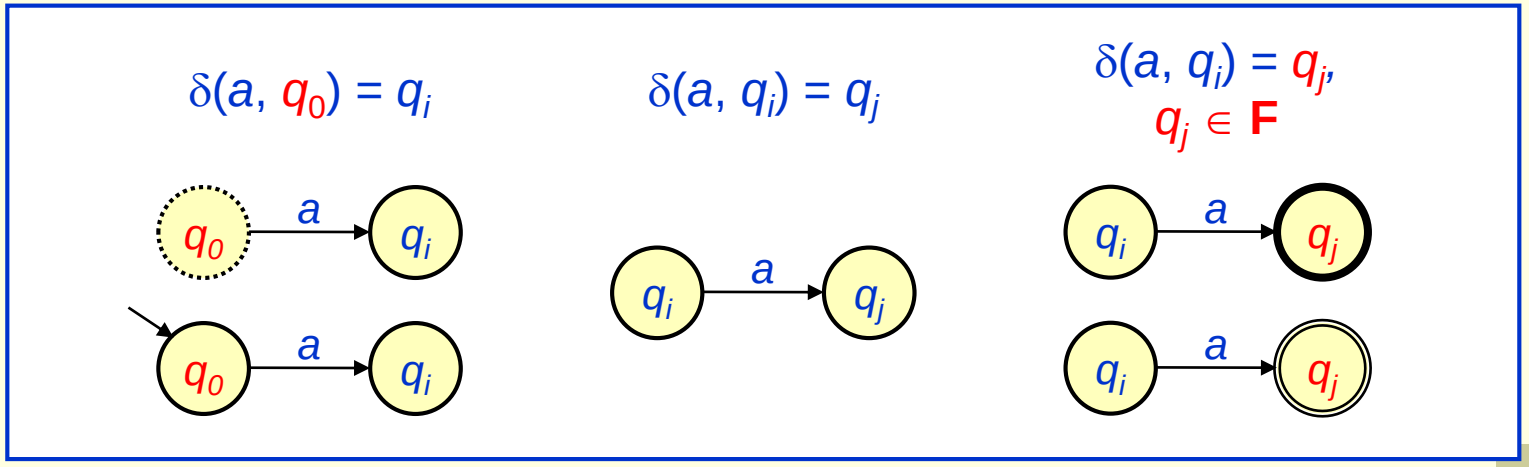
Виды преобразований:

- **Исключение** символов и правил без которых грамматика может существовать и порождать тот же язык – редуцирование, упрощение грамматики с сохранением ее эквивалентности исходной грамматике (в практическом смысле)
- **Изменение** состава множеств символов и правил – построение эквивалентной грамматики путем ее приведения к грамматике требуемого вида (класса), содержащей только правила определенного вида, т.е. удовлетворяющие некоторым ограничениям

Грамматический анализ регулярных языков

Конечный автомат M = (**Q**, **V**, δ, q_0, **F**), где

- **Q** – конечное множество состояний
- **V** – конечное множество символов (алфавит языка)
- δ – функция переходов, $\delta : \mathbf{V} \times \mathbf{Q} \to \mathbf{Q}$
  $\delta(a, q_i) = \mathbf{Q}_i$, $a \in \mathbf{V}$, $q_1 \in \mathbf{Q}$, $\mathbf{Q}_i \subseteq \mathbf{Q}$
  - $\forall i : |\mathbf{Q}_i| \le 1$ – **детерминированный** конечный автомат
  - $\exists i : |\mathbf{Q}_i| > 1$ – **недетерминированный** конечный автомат
- q_0 – начальное состояние, $q_0 \in \mathbf{Q}$
- **F** – множество конечных состояний, $\mathbf{F} \subseteq \mathbf{Q}$



Грамматический анализ регулярных языков

Автоматной грамматикой называют такую грамматику $G = (\mathbf{V}^T, \mathbf{V}^N, \mathbf{P}, S)$, множество правил **P** которой содержит только правила вида

$A \to Ba \mid a$ (леволинейная) или

$A \to aB \mid a$ (праволинейная),

где $A,B \in \mathbf{V}^N$, $a \in \mathbf{V}^T$

Правила вида $A \to B$ и $A \to \lambda$ недопустимы, т.к. при построении конечного автомата приводят к неоднозначности и переходам между состояниями без чтения символа входной цепочки

На практике обычно используют **леволинейные** автоматные грамматики

Построение автоматной грамматики

Вход: леволинейная регулярная грамматика

$G = (\mathbf{V}^T, \mathbf{V}^N, \mathbf{P}, S)$

Выход: леволинейная автоматная грамматика

$G' = (\mathbf{V}^{T'}, \mathbf{V}^{N'}, \mathbf{P}', S')$

Суть алгоритма преобразования заключается в переносе без изменений правил, удовлетворяющих автоматной грамматике, вида

$A \to Ba$ и $A \to a$,

и преобразовании правил, не удовлетворяющих автоматной грамматике, вида

$A \to a_1a_2\ldots a_{n-1}a_n$ и $A \to Ba_1a_2\ldots a_{n-1}a_n$,

$A \to B$ и $A \to \lambda$

в группы допустимых правил, сохраняющих эквивалентность грамматики (обеспечивающих тот же возможный вывод). Множества терминальных и нетерминальных символов сохраняются.

Построение автоматной грамматики

Алгоритм:

1. $\mathbf{V}^{T'} = \mathbf{V}^{T}$, $\mathbf{V}^{N'}_0 = \mathbf{V}^{N}$ — перенос терминальных и нетерминальных символов
2. $\mathbf{P}'_0 = \{ (A \to \gamma) \in \mathbf{P} : (\gamma = Ba \vee \gamma = a), A,B \in \mathbf{V}^{N'}, a \in \mathbf{V}^{T'} \}$ — «правильные» правила
3. $\mathbf{P}'_i = \mathbf{P}'_{i-1} \cup$ — «разделение» правил вида $A \to a_1a_2\ldots a_{n-1}a_n$
   $\cup \{ A \to A_{n-1}a_n, A_{n-1} \to A_{n-2}a_{n-1}, \ldots A_2 \to A_1a_2, A_1 \to a_1 :$
   $\exists (A \to a_1a_2\ldots a_{n-1}a_n) \in \mathbf{P}, A \in \mathbf{V}^{N'}, a_i \in \mathbf{V}^{T'} \}$
   $\mathbf{V}^{N'}_i = \mathbf{V}^{N'}_{i-1} \cup \{ A_{n-1}, A_{n-2}, \ldots A_2, A_1 \}$ — … добавление новых нетерм. символов
4. $\mathbf{P}'_i = \mathbf{P}'_{i-1} \cup$ — «разделение» правил вида $A \to Ba_1a_2\ldots a_{n-1}a_n$
   $\cup \{ A \to A_{n-1}a_n, A_{n-1} \to A_{n-2}a_{n-1}, \ldots A_2 \to A_1a_2, A_1 \to Ba_1 :$
   $\exists (A \to Ba_1a_2\ldots a_{n-1}a_n) \in \mathbf{P}, A,B \in \mathbf{V}^{N'}, a_i \in \mathbf{V}^{T'} \}$
   $\mathbf{V}^{N'}_i = \mathbf{V}^{N'}_{i-1} \cup \{ A_{n-1}, A_{n-2}, \ldots A_2, A_1 \}$ — … добавление новых нетерм. символов
5. $\mathbf{P}'_i = \mathbf{P}'_{i-1} \cup$ — устранение правил вида $A \to B$
   $\cup \{ A \to \gamma : \exists (A \to B) \in \mathbf{P} \wedge \exists (B \to \gamma) \in \mathbf{P}', (\gamma = Ca \vee \gamma = a),$
   $A,B,C \in \mathbf{V}^{N'}, a \in \mathbf{V}^{T'} \}$
   $\mathbf{P}'_i = \mathbf{P}'_{i-1} \cup$ — … и правил вида $A \to \lambda$
   $\cup \{ A \to a : \exists (B \to \lambda) \in \mathbf{P} \wedge \exists (A \to Ba) \in \mathbf{P}', A,B \in \mathbf{V}^{N'}, a \in \mathbf{V}^{T'} \}$
   Если $\mathbf{P}'_i \neq \mathbf{P}'_{i-1}$, то повторить п. 5 — … повторять, пока…
6. $S' = S$, $\mathbf{V}^{N'} = \mathbf{V}^{N'}_i$
7. Если $\lambda \in L(G)$, то $\mathbf{P}' = \mathbf{P}'_i \cup \{ S \to \lambda \}$, иначе $\mathbf{P}' = \mathbf{P}'_i$ — $L(G) = L(G')$

Построение конечного автомата

Вход: леволинейная автоматная грамматика

$G = (\mathbf{V}^T, \mathbf{V}^N, \mathbf{P}, S)$

Выход: конечный автомат

$M = (\mathbf{Q}, \mathbf{V}, \delta, q_0, \mathbf{F})$

Суть алгоритма построения конечного автомата заключается в установлении соответствия между

алфавитом языка **V** и множеством терминальных символов $\mathbf{V}^T$,

множеством состояний **Q** и множеством нетерминальных символов $\mathbf{V}^N$,

функцией переходов δ и множеством правил **P**,

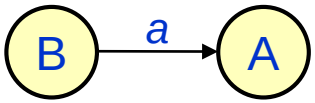
а также в определении

начального состояния $q_0 = H$,

множества конечных состояний **F**

$A \to Ba$

$\delta(a, B) = \{ A \}$



Построение конечного автомата

Алгоритм:

1. $\mathbf{V} = \mathbf{V}^T$

   Алфавит языка автомата эквивалентен множеству нетерминальных символов исходной грамматики

2. $\mathbf{Q} = \mathbf{V}^N \cup \{ H, E \}$

   Множество состояний соответствует множеству нетерминальных символов грамматики, но также включает начальное состояние H и состояние ошибки E (считывание недопустимого символа)

3. Если $(S \rightarrow \lambda) \in \mathbf{P}$, то $\mathbf{F} = \{ H, S, E \}$, иначе $\mathbf{F} = \{ S, E \}$

   Если пустая цепочка принимается (принадлежит языку), то начальное состояние H также является конечным, иначе конечными являются только состояния S (соответствует целевому символу – цепочка принята) и E (ошибка – цепочка не принята)

4. $\delta(a, B) = \{ \forall A \in \mathbf{V}^N : (A \rightarrow Ba) \in \mathbf{P}, B \in \mathbf{Q}, a \in \mathbf{V} \}$,
   $\delta(a, H) = \{ \forall A \in \mathbf{V}^N : (A \rightarrow a) \in \mathbf{P}, H \in \mathbf{Q}, a \in \mathbf{V} \}$

   Переход в состояние A из состояния B или из начального состояния H при считывании входного символа a.

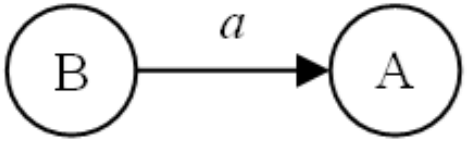
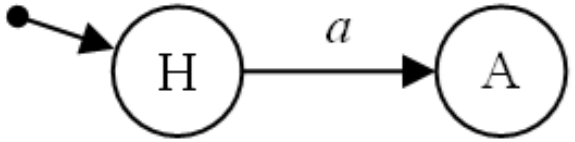
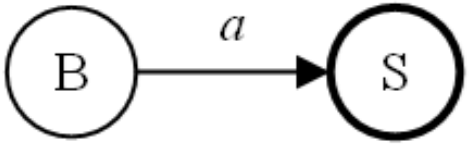
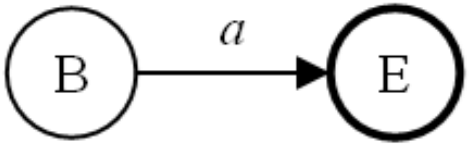
   Если два или более правил имеют одинаковые правые части, то $|\delta(a, X)| > 1$, т.е. автомат является недетерминированным

5. $\delta(a, B) = \{ E \} : (X \rightarrow Ba) \notin \mathbf{P}, \forall X,B \in \mathbf{Q}, \forall a \in \mathbf{V}$

   Доопределение автомата, т.е. все «невозможные» в соответствии с правилами грамматики переходы приводят к переходу в состояние ошибки E, которое является конечным (останов автомата)

Построение конечного автомата

Преобразование автоматной грамматики в конечный автомат

Правило грамматики	Функция перехода	Эквивалентный фрагмент графа конечного автомата
$A \to Ba$	$\delta(a, B) = \{ A \}$	
$A \to a$	$\delta(a, H) = \{ A \}$	
$S \to Ba$	$\delta(a, B) = \{ S \}$	
(несуществующее правило)	$\delta(a, B) = \{ E \}$	

Построение леволинейной грамматики

Вход: праволинейная регулярная (автоматная) грамматика
$G = (\mathbf{V}^T, \mathbf{V}^N, \mathbf{P}, S)$

Выход: леволинейная регулярная (автоматная) грамматика
$G' = (\mathbf{V}^{T'}, \mathbf{V}^{N'}, \mathbf{P}', S')$

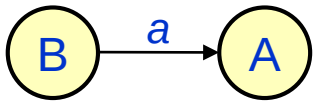
Алгоритм:

1. $\mathbf{V}^{T'} = \mathbf{V}^T$, $\mathbf{V}^{N'} = \mathbf{V}^N$
2. $\mathbf{P}'_0 = \{ A \to B\alpha : \quad (B \to \alpha A) \in \mathbf{P}, A,B \in \mathbf{V}^{N'}, \alpha \in \mathbf{V}^{T'*} \}$
3. $\mathbf{P}'_1 = \mathbf{P}'_0 \cup \{ S \to B\alpha : \quad (B \to \alpha) \in \mathbf{P}, B,S \in \mathbf{V}^{N'}, \alpha \in \mathbf{V}^{T'*} \}$
4. $\mathbf{P}'_2 = \mathbf{P}'_1 \cup \{ A \to \alpha : \quad (S \to \alpha A) \in \mathbf{P}, A,S \in \mathbf{V}^{N'}, \alpha \in \mathbf{V}^{T'*} \}$
5. $S' = S$, $\mathbf{P}' = \mathbf{P}'_2$

Автоматная грамматика

леволинейная	**праволинейная**
$A \to Ba$	$B \to aA$

$\delta(a, B) = \{ A \}$



Для автоматной грамматики:
$|\alpha| = 1$, т.е. $\alpha = a$