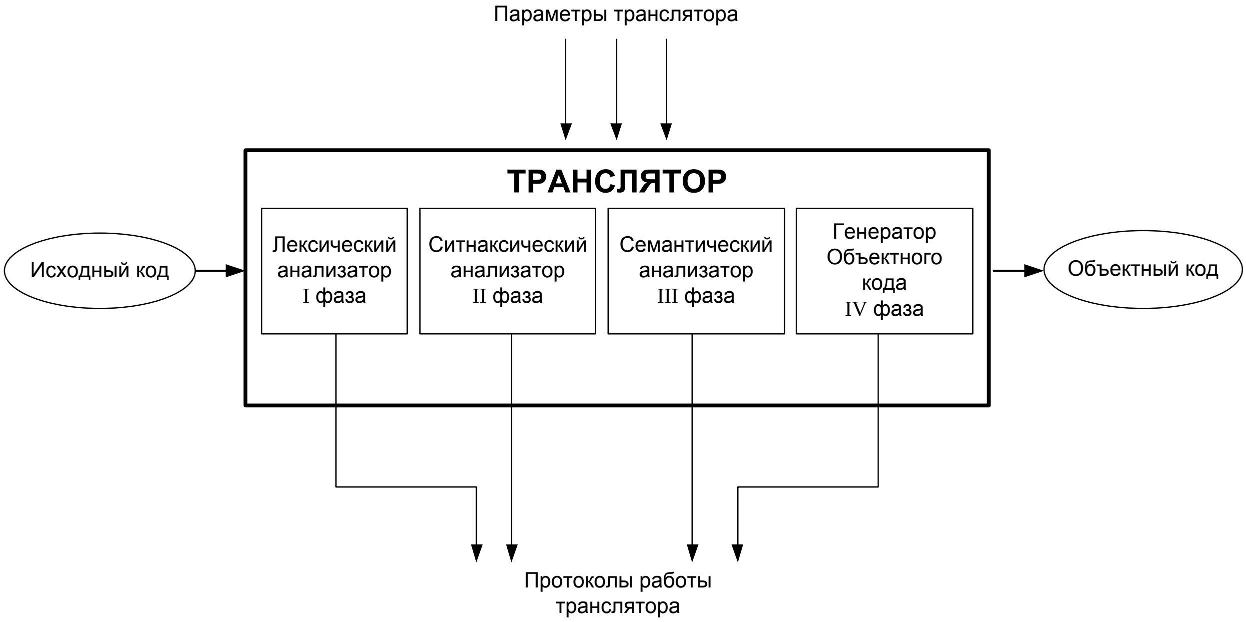
Лекция 13
БГТУ, ФИТ, ПОИТ, 2 семестр
Конструирование программного обеспечения

Синтаксический анализ

1. Синтаксический анализ: вторая фаза трансляции.



Синтаксический анализ выполняется после фазы лексического анализа и предназначен для распознавания синтаксических конструкций и формирования промежуточного кода.

2. Синтаксический анализ: основная фаза трансляции.

Без нее процесс трансляции не имеет смысла.

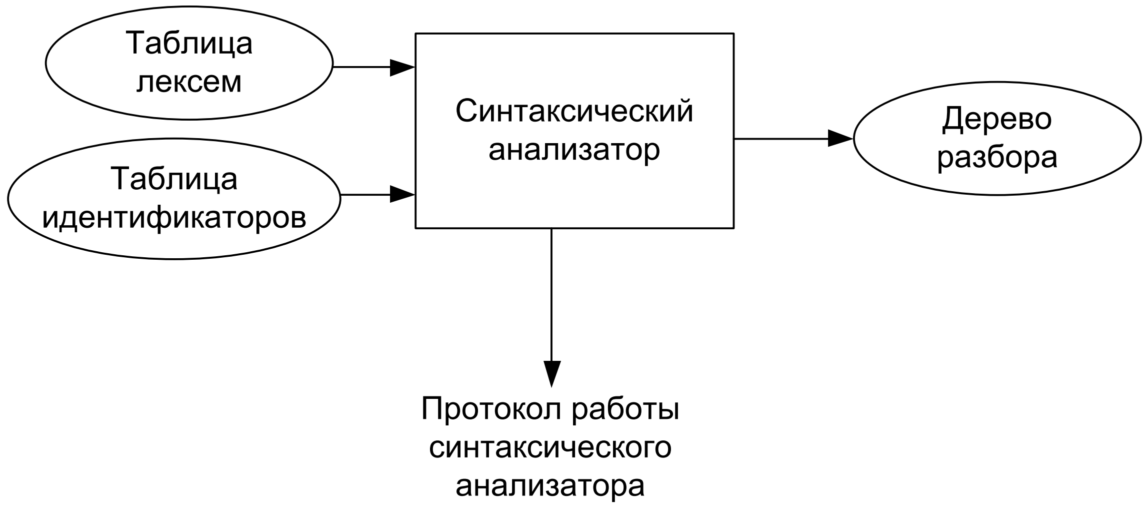
Все задачи лексического анализа могут быть решены в рамках синтаксического анализа. Т.е. можно создать транслятор без лексического анализатора. Лексический анализ необходим для освобождения алгоритма синтаксического разбора от рутинных алгоритмов.

Программа, выполняющая синтаксический анализ называется **синтаксическим анализатором**.

Вход синтаксического анализатора:
- таблица лексем (ТЛ);
- таблица идентификаторов (ТИ).

Выход синтаксического анализатора:
- дерево разбора.

Входная и ***выходная*** информация синтаксического анализатора:



> Нет четкой границы между лексическим и синтаксическим анализатором. Алгоритм разбора распределяется между лексическим и синтаксическим анализатором.
> Конструкции языка, которые разбираются лексическим или синтаксическим анализатором, определяет разработчик транслятора.

3. Задачи, выполняемые синтаксическим анализатором.

1) поиск и выделение синтаксических конструкций в исходном тексте (разбор);
2) распознавание (проверка правильности) синтаксических конструкций;
3) выявление ошибок и продолжение процесса распознавания после обработки ошибок;
4) формирование дерева разбора в случае, если нет ошибок.

4. Исходный текст программы для синтаксического анализатора представляется в виде таблицы лексем.

5. Для описания языка, разбираемого синтаксическим анализатором, применяют грамматики **типа 2 иерархии Хомского – контекстно-свободные грамматики** (см. лекция 9).

6. Грамматики типа 2 иерархии Хомского (КС-грамматики):

$G_{II} = \langle T, N, P, S \rangle$ – контекстно-свободные грамматики. Правила (продукции) грамматики P имеют вид: $A \to \alpha$, где $A \in N$, $\alpha \in V^*$, $V = N \cup T$ – словарь грамматики G_{II}. S – стартовый символ грамматики.

8. Имеется три основных типа синтаксических анализаторов КС-грамматик:

- нисходящие;
- восходящие;
- универсальные.

9. Преобразование грамматики

Формально преобразование грамматики определяется следующим образом:

$$G_1 = \langle T_1, N_1, P_1, S \rangle \to G_2 = \langle T_2, N_2, P_2, S \rangle \Leftrightarrow L(G_2) = L(G_1)$$

Основные цели преобразований КС-грамматик: упрощение правил грамматики и облегчение создания распознавателя языка.

10. Грамматика G является ***приведенной грамматикой***, если в грамматике нет:

- бесплодных символов;
- недостижимых символов;
- λ-правил;
- цепных правил.

ВНИМАНИЕ! Шаги преобразования грамматики должны выполняться строго в указанном порядке.

11. Приведение грамматики G – это отыскание эквивалентной приведенной грамматики G'.

Процесс приведения – упрощение грамматики.

12. ***Определение бесплодного символа.***

Нетерминальный символ $A \in N$ в грамматике $G = \langle T, N, P, S \rangle$ называется **бесплодным**, если множество $\{\alpha \mid \alpha \in T^*, A \Rightarrow *\alpha\} = \varnothing$

т.е. в грамматике G из нетерминала A нельзя вывести хотя бы одну цепочку, состоящую из терминальных символов или пустого символа λ.

Другими словами, нетерминальный символ называется бесплодным, если из него нельзя вывести ни одной цепочки.

13. Алгоритм удаления бесплодных символов

Рекурсивно строим множества $N_0, N_1, N_2, ...$

1) $N_0 = \varnothing$
2) $N_1 = \{A \mid (A \to \alpha) \in P \wedge \alpha \in (N_0 \cup T)^*\} \cup N_0$
3) если $N_1 \neq N_0$, то переход на шаг 4

   иначе $G' = (T, N_1, P', S)$,

   где P' – правила из P, содержащие только символы $V' = N_1 \cup T$
4) $N_2 = \{A \mid (A \to \alpha) \in P \wedge \alpha \in (N_1 \cup T)^*\} \cup N_1$
5) если $N_2 \neq N_1$, то переход на шаг 6

   иначе $G' = (T, N_2, P', S)$, где P' – правила из P, содержащие только символы $V' = N_2 \cup T$
6) $N_3 = \{A \mid (A \to \alpha) \in P \wedge \alpha \in (N_2 \cup T)^*\} \cup N_2$
7) если $N_3 \neq N_2$, то переход на шаг 8

   иначе $G' = (T, N_3, P', S)$, где P' – правила из P, содержащие только символы $V' = N_3 \cup T$
8) ...

14. В общем виде алгоритм удаления бесплодных символов можно записать следующим образом:

1) $N_0 = \varnothing$, i=0
2) $i = i + 1$, $N_i = \{A \mid (A \to \alpha) \in P \wedge \alpha \in (N_{i-1} \cup T)^*\} \cup N_{i-1}$
3) если $N_i \neq N_{i-1}$, то переход на шаг 2
   иначе $G' = (T, N_i, P', S)$,
   где P' – правила из P, содержащие только символы $V' = N_i \cup T$

15. ***Определение недостижимого символа.***

Символ $X \in (N \cup T)$ в грамматике $G = \langle T, N, P, S \rangle$ называется ***недостижимым***, если он не встречается ни в одной сентенциальной форме грамматики.

Другими словами: недостижимым символом называется символ, который не может быть выведен из стартового символа грамматики S.
Очевидно, что недостижимый символ грамматике не нужен.

Напоминание из лекции 9:
если $S \Rightarrow^* \beta$ и $\beta \in (T \cup N)^*$,
то β называется **сентенциальной** формой грамматики $G = \langle T, N, P, S \rangle$).

16. Алгоритм удаления недостижимых символов

> ***ВНИМАНИЕ!*** Перед удалением недостижимых символов, должны быть удалены бесплодные символы

Строим множества $V_0, V_1, V_2, \ldots$:

Описание алгоритма. Строим множество достижимых символов.

Первоначально в это множество входит только стартовый (целевой) символ S грамматики, затем множество пополняем на основе правил грамматики.
Все символы, которые не войдут в это множество, являются недостижимыми и могут быть исключены в новой грамматике из словаря и из правил.

1) $V_0 = \{S_0\}$, $\boldsymbol{i} = 1$
2) $V_i = \{x \mid x \in (N \cup T) \;\; и \;\; (A \to \alpha x \beta) \in P, A \in V_{i-1}\} \cup V_{i-1}$,

   где $\alpha, \beta \in (N \cup T)^*$
3) если $V_i \neq V_{i-1}$, то i=i+1 и переход на шаг 2,

   иначе $G' = (T', N', P', S)$,

   где $N' = N \cap V_i, T' = T \cap V_i$ P' — правила из P, содержащие только символы V_i.

17. Пример удаления *бесплодных* и *недостижимых* символов:

$G = \langle \{a,b,c,d\}, \{A,B,C,S\}, P, S \rangle$, где

$P = \{S \to aSa \mid bAd \mid c, \;\; A \to cBd \mid aAd, \;\; C \to d\}$

Здесь ***бесплодные*** символы: A, B (нельзя вывести ни одной цепочки);
Недостижимые символы: C (не выводится из стартового символа).

$$N_i = \{A \mid (A \to \alpha) \in P \;\; и \;\; \alpha \in (N_{i-1} \cup T)^*\} \cup N_{i-1}$$

1) $N_0 = \varnothing$
2) $N_1 = \{S, C\}$ // есть правило вывода из нетерминалов S, C терминальных символов
3) $N_1 \neq N_0$, переход на следующий шаг
4) $N_2 = \{S, C\}$
5) $N_2 = N_1$ $\;G' = (\{a,b,c,d\}, \{S,C\}, \{S \to aSa, S \to c, C \to d\}, S)$

$G = \langle \{a,b,c,d\}, \{A,B,C,S\}, P, S \rangle$, где

$P = \{S \to aSa \,|\, bAd \,|\, c, \quad A \to cBd \,|\, aAd, \quad C \to d\}$

$$V_i = \{x \,|\, x \in (N \cup T) \wedge (A \to \alpha x \beta) \in P, A \in V_{i-1}\} \cup V_{i-1}$$

1) $V_0 = \{S\}$
2) $V_1 = \{S, a, c\} \cup V_0$ // добавляем символы, которые можно вывести из S
3) $V_2 = \{S, a, c\}$
4) $G' = (\{a,c\}, \{S\}, \{S \to aSa, S \to c\}, S)$

Алгоритмы удаления бесплодных и недостижимых символов упрощают грамматики, сокращают количество символов алфавита и правил грамматики.

18. ***Определение.*** Правило вида $A \to B$, где $A, B \in N$ называется ***цепным***.

Теорема: для грамматики G, содержащей цепные правила, всегда можно найти эквивалентную ей грамматику G', не содержащую ***цепных*** правил.

Правила вида $A \to B$, $B \to C$, $C \to aX$
могут быть заменены одним правилом $A \to aX$,
т.к. вывод $A \Rightarrow B \Rightarrow C \Rightarrow aX$ цепочки aX в грамматике G может быть заменен выводом $A \Rightarrow aX$ с помощью правила $A \to aX$.

19. Алгоритм исключения цепных правил

P – множество правил грамматики G.

Разобьем множество P на два подмножества P_1 и P_2 по следующему принципу: P_1 содержит правила вида $A_i \to B_i$, и P_2 – все остальные.

На множестве P_1 построим множество правил $P(A_i)$,

для которых $B_i \to \alpha$ и $\alpha \in (N \cup T)^*$ это правила из P_2.

Замещаем цепные правила $A_i \to B_i$ в $P(A_i)$ на правила $A_i \to \alpha$.

Тогда правила грамматики $P' = P(A_1) \cup P(A_2) \cup \ldots \cup P_2$ являются правилами грамматики G', не содержащие цепных правил.

20. ***Пример***. Исключение ***цепных*** правил:

$G = \langle \{+,*,(,),a\},\{E,T,F\},P,E \rangle$

где $P = \{E \to E+T\,|\,T,\;\; T \to T*F\,|\,F,\;\; F \to (E)\,|\,a\}$

Необходимо исключить правила $E \to T,\quad T \to F$.

Пусть следующие нетерминалы определяют понятия

E – выражения, состоящие из слагаемых, разделенных знаками +,

T – выражения, состоящие из сомножителей, разделенных знаками *,

F – выражения, которые могут быть выражением, либо терминалом a.

$P_1 = \{E \to T,\;\; T \to F\}$

$P_2 = \{E \to E+T,\;\; T \to \underline{T*F},\;\; F \to \underline{(E)\,|\,a}\}$

// строим множества, заменяя цепные правила $E \to T,\quad T \to F$:

$P_1(E) \Rightarrow P_1(E) = \{E \to T+T,\;\; \underline{E \to T*F},\;\; \underline{E \to (E)\,|\,a}\}$

$P_1(T) \Rightarrow P_1(T) = \{\underline{T \to (E)\,|\,a}\}$

$$P' = P_1(E) \cup P_1(T) \cup P_2:$$

$P' = \{E \to T*F\,|\,(E)\,|\,a\,|\,T+T,\;\; T \to T*F\,|\,(E)\,|\,a,\;\; F \to (E)\,|\,a\}$

21. ***Определение.*** Правило вида $A \to \lambda$ называется λ-правилом или ***аннулирующим*** правилом.

Теорема: для грамматики G, содержащей λ-правила, всегда можно найти эквивалентную грамматику G' не содержащую λ-правил.

22. Алгоритм исключения λ-правил.
Выполнить все возможные подстановки пустой цепочки вместо аннулирующего нетерминала во всех правилах грамматики.

23. ***Пример.***

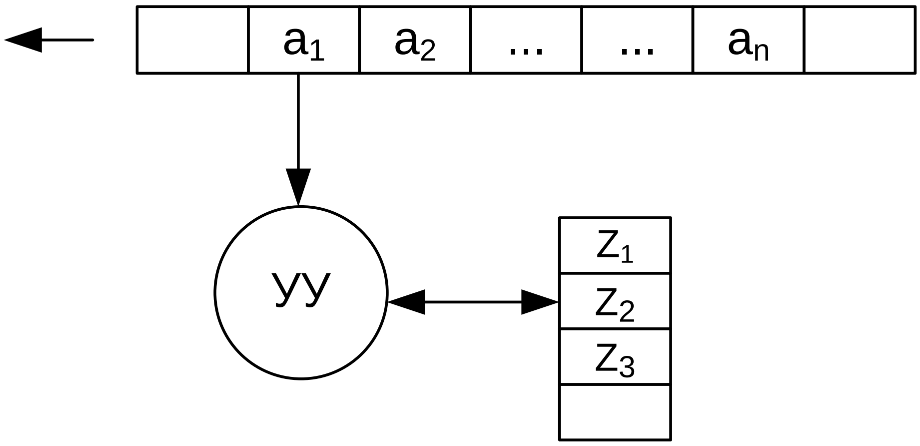
$$G = \langle \{a,b\}, \{I, A, B, C\}, P, I \rangle,$$

где $P = \{I \to ABC,\ \ A \to BB \mid \lambda,\ \ B \to CC \mid a,\ \ C \to AA \mid b\}$,

Заменим аннулирующий нетерминал A пустой цепочкой λ в тех правилах грамматики, где он встречается в правой части, остальные правила оставим без изменения.

$$I \to ABC \mid BC,\ \ C \to AA \mid A \mid b$$
$$P' = \{I \to ABC \mid BC,\ \ A \to BB,\ \ B \to CC \mid a,\ \ C \to AA \mid A \mid b\}$$

24. **Автоматы с магазинной памятью (МП-автоматы) –** распознаватели контекстно-свободных языков, которые можно представить в виде следующей схемы:



25. Формальное описание МП-автомата:

$M = \langle Q, V, Z, \delta, q_0, z_0, F \rangle$

Q – множество состояний управляющего устройства;

V – алфавит входных символов;

Z – специальный алфавит магазинных символов;

δ – функция переходов автомата $Q \times (V \cup \{\lambda\}) \times Z \to P(Q \times Z^*)$,

где $P(Q \times Z^*)$ - множество подмножеств $Q \times Z^*$;

$q_0 \in Q$ – начальное состояние автомата;

$z_0 \in Z$ – начальное состояние магазина (маркер дна);

$F \subseteq Q$ – множество конечных состояний.

Функция переходов δ отображает тройки (q, a, z) в пары (q', γ) для детерминированного автомата или во множество таких пар для недерминорованного автомата, где $q' \in Q^*$, γ – символ в вершине магазина. Эта функция описывает состояние магазинного автомата, при чтении символа с входной ленты и перемещении головки.

26. **Конфигурация МП-автомата**

Конфигурация автомата (текущее состояние) описывается тройкой: (q, α, ω), где

q – текущее состояние автомата;

α – остаток цепочки. Первый символ этой цепочки просматривается входной головкой автомата. Если $\alpha = \{\lambda\}$, то входной символ прочитан;

ω – цепочка-содержимое магазина (стека). Если $\omega = \{\lambda\}$, то магазин пустой.

27. Один такт работы автомата:

$(q, a\alpha, z\omega) \succ (q', \alpha, \gamma\omega)$ (читается как «переходит в конфигурацию»),

если $(q', \gamma) \in \delta(q, a, z)$.

При выполнении такта из магазина (стека) удаляется верхний символ, соответствующий условию перехода, и добавляется цепочка, соответствующая правилу перехода.

Первый символ цепочки становится вершиной стека.

Допускаются переходы, при которых входной символ игнорируется. Эти переходы (такты) называются λ-переходами.

28. Начальным состоянием МП-автомата называется состояние (q_0, α, z_0), где q_0 – начальное состояние автомата, α – входная цепочка, z_0 – маркер дна магазина.

29. ***Определение.*** Цепочка α является **допустимой** (распознается) автоматом, если из начальной конфигурации за конечное число тактов работы автомат перейдет в заключительное состояние:

$M = \langle Q, V, Z, \delta, q_0, z_0, F \rangle$, если $(q_0, \alpha, z_0) \succ^* (q', \lambda, \lambda)$ и $q' \in F$.

30. Работа автомата $M = \langle Q, V, Z, \delta, q_0, z_0, F \rangle$

1) состояние автомата $(q, a\alpha, z\beta)$
2) читает символ a находящийся под головкой (сдвигает ленту);
3) не читает ничего (читает λ, не сдвигает ленту);
4) из δ определяет новое состояние q', если $(q', \gamma) \in \delta(q, a, z)$ или $(q', \gamma) \in \delta(q, \lambda, z)$.
5) читает верхний (в стеке) символ z и записывает цепочку γ т.к. $(q', \gamma) \in \delta(q, a, z)$, при этом, если $\gamma = \lambda$, то верхний символ магазина просто удаляется.
6) работа автомата заканчивается (q, λ, λ)

31. На каждом шаге автомата возможны три случая:

1) функция $\delta(q, a, z)$ определена – осуществляется переход в новое состояние;
2) функция $\delta(q, a, z)$ не определена, но определена $\delta(q, \lambda, z)$ – осуществляется переход в новое состояние (лента не продвигается);
3) функции $\delta(q, a, z)$ и $\delta(q, \lambda, z)$ не определены – дальнейшая работа автомата не возможна (цепочка не разобрана).

32. Язык $L(M)=\{\alpha \mid (q_0,\alpha,z_0)\succ^* (q',\lambda,\lambda), q'\in F\}$ – язык, допускаемый автоматом M.

Пример:

$$M=\langle\{q_0,q_1,q_2\},\{a,b\},\{z_0,a\},\delta,q_0,z_0,\{q_0\}\rangle$$

$\underbrace{\delta(q_0,a,z_0)}_{\text{состояние, символ, стек}}=\underbrace{(q_1,z_0a)}_{\text{новое сост., стек}}$ (1) // из нач. сост. q_0 автомат сдвигает ленту и переходит в q_1 при этом символ a записывается в стек

$\delta(q_1,a,a)=(q_1,aa)$ (2) // в q_1 видит символ a и в вершине стека символ a, сдвигает ленту, переходит в q_1, помещает символ a в стек

$\delta(q_1,b,a)=(q_2,\lambda)$ (3) // в сост. q_1 видит символ b, а в стеке a, сдвигает ленту, удаляет символ с вершины стека и переходит в q_2

$\delta(q_2,b,a)=(q_2,\lambda)$ (4) // в сост. q_2 автомат, обозревая символ b, а в вершине стека символ a, сдвигает ленту, удаляет из стека a переходит в q_2

$\delta(q_2,\lambda,z_0)=(q_0,\lambda)$ (5) // в q_2 видит символ λ (цепочка пуста) и в стеке символ дна стека z_0 переходит в заключительное состояние q_0

Этот автомат является детерминированным.

Работf автомата виде последовательности конфигураций:

$(q_0,aabb,z_0)$ // начальная конф. МП-автомата (сост. q_0, цепочка $aabb$, в стеке z_0)

(q_1,abb,z_0a) // по (1)

(q_1,bb,z_0aa) // по (2)

(q_2,b,z_0a) // по (3)

(q_2,λ,z_0) // по (4)

(q_0,λ,λ) // по (5)

Цепочка разобрана

$$\delta(q_0,a,z_0)=(q_1,z_0a)$$
$$\delta(q_1,a,a)=(q_1,aa)$$
$$\delta(q_1,b,a)=(q_2,\lambda)$$
$$\delta(q_2,b,a)=(q_2,\lambda)$$
$$\delta(q_2,\lambda,z_0)=(q_0,\lambda)$$

Шаг	Состояние	Входная цепочка	Магазин	Функция перехода
1	q_0	$aabb$	z_0	1
2	q_1	abb	az_0	2
3	q_1	bb	aaz_0	3
4	q_2	b	az_0	4
5	q_2	λ	z_0	5
6	q_0	λ	λ	

Пример:

$$M = \langle \{q_0, q_1, q_2\}, \{a, b\}, \{z_0, a\}, \delta, q_0, z_0, \{q_2\} \rangle$$

Правила:

$\delta(q_0, a, z_0) = (q_0, z_0 a)$ (1)

$\delta(q_0, b, z_0) = (q_0, z_0 b)$ (2)

$\delta(q_0, a, a) = \{(q_0, aa), (q_1, \lambda)\}$ (3)

$\delta(q_0, b, a) = (q_0, ab)$ (4)

$\delta(q_0, a, b) = (q_0, ba)$ (5)

$\delta(q_0, b, b) = \{(q_0, bb), (q_1, \lambda)\}$ (6.1 и 6.2)

$\delta(q_1, a, a) = (q_1, \lambda)$ (7)

$\delta(q_1, b, b) = (q_1, \lambda)$ (8)

$\delta(q_1, \lambda, z_0) = (q_2, \lambda)$ (9)

$(q_0, abba, z_0)$

$(q_0, bba, z_0 a)$

$(q_0, ba, z_0 ab)$ // первое правило 6.1 (из двух альтернатив)

$(q_0, a, z_0 abb)$ // по 5

$(q_0, \lambda, z_0 abba)$ // лента пустая, в стеке есть символы (не z_0)

Цепочка не разобрана, т.к. входная цепочка прочитана и переход для конфигурации $(q_0, \lambda, z_0 abba)$ не определен.

$(q_0, abba, z_0)$

$(q_0, bba, z_0 a)$

$(q_0, ba, z_0 ab)$ // по 6.2

$(q_1, a, z_0 a)$ // по 7

(q_1, λ, z_0)

(q_2, λ, λ)

Цепочка разобрана