МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский Нижегородский государственный
университет им. Н.И. Лобачевского»

Л.П. Жильцова, Т.Г. Смирнова

ОСНОВЫ ТЕОРИИ КОНТЕКСТНО-СВОБОДНЫХ ЯЗЫКОВ В ПРИМЕРАХ И ЗАДАЧАХ

Учебно-методическое пособие

Рекомендовано методической комиссией
института информационных технологий, математики и механики
для студентов ННГУ, обучающихся по направлениям подготовки
020302 «Фундаментальная информатика и информационные технологии»,
090303 «Прикладная информатика (в информационной сфере)»,
090304 «Программная инженерия»

Нижний Новгород
2017

УДК 519.17+519.711.4
ББК В182
Ж-72

Ж−72 Жильцова Л.П., Смирнова Т.Г. Основы теории контекстно-свободных языков в примерах и задачах: учебно-методическое пособие. – [электронный ресурс] – Нижний Новгород: Нижегородский госуниверситет, 2017. – 60 с.

Фонд электронных образовательных ресурсов ННГУ

Рецензент: к. ф.-м. наук, доцент **А.Л.Калашников**

В учебно-методическом пособии рассматриваются основные теоретические понятия и алгоритмы, относящиеся к контекстно-свободным языкам, автоматам с магазинной памятью и синтаксическому анализу. Изучение каждой темы сопровождается необходимым теоретическим материалом и примерами решения типовых задач. Предлагаются также задачи для самостоятельного решения.

Учебно-методическое пособие предназначено для студентов института информационных технологий, математики и механики направлений подготовки «Фундаментальная информатика и информационные технологии», «Прикладная информатика (в информационной сфере)», «Программная инженерия», изучающих курс «Теория автоматов и формальных языков».

УДК 519.17+519.711.4
ББК В182

Глава 1. Контекстно-свободные грамматики и языки

1.1. Основные определения и понятия, связанные с контекстно-свободными грамматиками

Контекстно-свободной грамматикой (КС-грамматикой) называется система $G = \langle V_T, V_N, P, S \rangle$, где V_T, V_N – непересекающиеся конечные множества терминальных и нетерминальных символов (терминалов и нетерминалов) соответственно; $S \in V_N$ – некоторый выделенный символ, называемый аксиомой грамматики, P – конечное множество правил. Терминальные символы называются также основными, нетерминальные – вспомогательными символами.

Каждое правило в КС-грамматике имеет вид $A \to \alpha$, где $A \in V_N$ и $\alpha \in (V_N \cup V_T)^*$.

Для слов α и β из $(V_T \cup V_N)^*$ будем говорить, что β ***непосредственно выводимо*** из α (и записывать $\alpha \Rightarrow \beta$), если α и β можно представить в виде $\alpha = \alpha_1 A \alpha_2$, $\beta = \alpha_1 \gamma \alpha_2$ для некоторых $\alpha_1, \alpha_2 \in (V_T \cup V_N)^*$, и в грамматике G имеется правило $A \to \gamma$. В этом случае говорят, что слово α получено из β применением правила $A \to \gamma$.

Рефлексивное транзитивное замыкание отношения $\Rightarrow$ назовем ***отношением выводимости*** и обозначим через $\Rightarrow^*$. Таким образом, $\alpha \Rightarrow^* \beta$, если слово β может быть выведено из слова α путем применения конечное число раз правил грамматики.

Язык, порождаемый грамматикой G, определим как множество всех слов в терминальном алфавите, выводимых из аксиомы грамматики S: $L(G) = \{ \alpha : S \Rightarrow^* \alpha, \alpha \in V_T^* \}$. Язык L называется ***КС-языком***, если существует КС-грамматика G такая, что $L = L(G)$.

Пример 1.1.1. Пусть задана КС-грамматика $G_1 = \langle \{a,b\}, \{S\}, P, S \rangle$, где P состоит из правил:

$$S \to \overset{(1)}{aSbS} \mid \overset{(2)}{bSaS} \mid \overset{(3)}{\lambda} \quad (\lambda - \text{пустое слово}).$$

Грамматика G_1 порождает множество всех слов в алфавите $\{a,b\}$, содержащих одинаковое число букв a и b.

Рассмотрим слово $\alpha = babbaa$ и процесс его вывода из аксиомы S:

$$S \Rightarrow bSaS \Rightarrow baSbSaS \Rightarrow babSaS \Rightarrow babSa \Rightarrow babbSaSa \Rightarrow babbaa \in L.$$

Последовательность номеров правил грамматики, с помощью которой слово выводится из аксиомы S, называется ***выводом*** слова.

Для $\alpha = babbaa$ рассмотренный вывод имеет вид $w(\alpha) = 2133233.$

Левым (правым) выводом называется вывод, в котором каждое правило применяется к самому левому (самому правому) вхождению нетерминала в слово. $w_l(\alpha) = 2132333$ - левый вывод слова α, $w_r(\alpha) = 2312333$ - правый.

Для КС-грамматик существует удобный графический способ представления вывода – ***дерево вывода***. Дерево строится следующим образом.

(1) Корень дерева помечается аксиомой грамматики.

(2) Если некоторая вершина дерева помечена нетерминалом A, и в процессе вывода к этому нетерминалу применяется правило $A \Rightarrow b_{i_1} b_{i_2} \ldots b_{i_k}$, то из вершины в следующий ярус проводится k ребер, и полученные вершины следующего яруса помечаются слева направо буквами $b_{i_1}, b_{i_2}, \ldots, b_{i_k}$. При применении правил вида $A \Rightarrow \lambda$ из вершины, помеченной нетерминалом A, в следующий ярус проводится одно ребро и новая вершина помечается символом пустого слова λ.

Для слова $\alpha \in L(G_1)$ листья дерева помечены символами терминального алфавита либо символом λ. Само слово α получается обходом кроны дерева слева направо. На рис. 1 для грамматики G_1 из примера 1.1.1 изображено

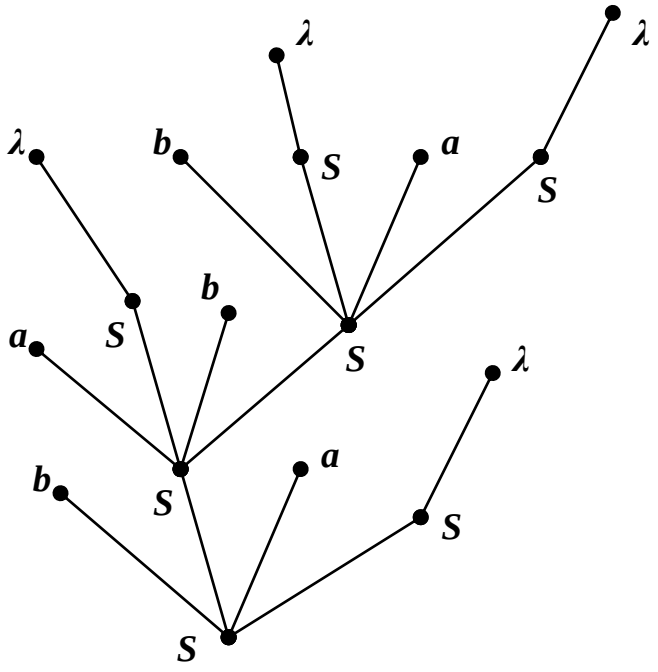

Рис. 1. Дерево вывода для слова α = *babbaa*

дерево вывода слова $\alpha = babbaa$.

КС-грамматика G называется ***грамматикой с однозначным выводом*** (или ***однозначной грамматикой***), если каждое слово из $L(G)$ имеет единственный левый вывод. В противном случае КС-грамматика называется ***грамматикой с неоднозначным выводом***.

Пример 1.1.2. Рассмотрим грамматику $G_2 = \langle \{a, +, *, (,)\}, \{E\}, P, E \rangle$, где P содержит следующие четыре правила:

$$(1) \quad E \to E + E$$
$$(2) \quad E \to E * E$$
$$(3) \quad E \to (E)$$
$$(4) \quad E \to a.$$

Слово $\alpha = a + a * a$ из $L(G_2)$ имеет два различных левых вывода $w_l(\alpha) = 14244$ и $w_l'(\alpha) = 21444$ и соответственно два различных дерева вывода. Поэтому G_2 - грамматика с неоднозначным выводом.

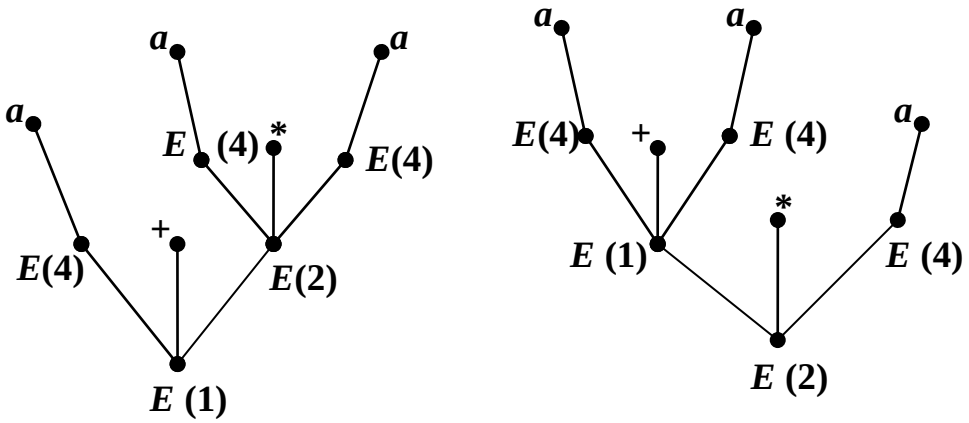

Рис. 2. Деревья вывода для слова $\alpha = a + a * a$

Деревья вывода для слова $\alpha = a + a * a$ приведены на рис. 2.

Пример 1.1.3. Пусть $L = \{a^n b^n c^k \mid n, k \geq 0\}$. Требуется построить КС-грамматику G_3, для которой $L = L(G_3)$.

Введем два нетерминальных символа A и B. Нетерминал A будем использовать для порождения подслова $\alpha_1 = a^n b^n$ с помощью правила $A \to aAb$; последовательно применяя это правило n раз, получим слово $a^n A b^n$. Добавим правило $A \to \lambda$, чтобы иметь возможность удалить нетерминал A из $a^n A b^n$. Нетерминал B будем использовать для порождения

подслова $\alpha_2 = c^k$, последовательно применяя k раз правило $B \to cB$. Для удаления нетерминала B добавим правило $B \to \lambda$. Чтобы получить слово $\alpha = \alpha_1\alpha_2 \in L$, добавим аксиому грамматики S и правило $S \to AB$. Заметим, что пустое слово принадлежит L, его можно получить так: $S \Rightarrow AB \Rightarrow \lambda B \Rightarrow \lambda\lambda = \lambda$.

Таким образом, мы построили грамматику $G_3 = \langle \{a,b,c\}, \{S,A,B\}, P, S \rangle$, где P состоит из следующих правил:

$$S \to AB$$
$$A \to aAb \mid \lambda$$
$$B \to cB \mid \lambda.$$

Задачи и упражнения

В задачах 1.1.1 – 1.1.21 построить контекстно-свободную грамматику, порождающую заданный язык.

1.1.1. $L = \left\{0^i 1^j \mid i \geq j\right\}$.

1.1.2. $L = \left\{0^i 1^j \mid i < j\right\}$.

1.1.3. $L = \left\{0^i 1^j 2^k \mid i, j, k \geq 1,\ i = j\right\}$.

1.1.4. $L = \left\{0^k 1^i 2^k \mid i, k \geq 1\right\}$.

1.1.5. $L = \left\{\alpha\alpha^R \mid \alpha \in \{a,b\}^*\right\}$, где α^R - обращение слова α.

1.1.6. $L = \left\{\alpha c\alpha^R \mid \alpha \in \{a,b\}^*\right\}$.

1.1.7. $L = \left\{\alpha\alpha^R\beta\beta^R \mid \alpha, \beta \in \{a,b\}^*\right\}$.

1.1.8. $L = \left\{\alpha c\alpha^R c\beta c\beta^R \mid \alpha, \beta \in \{a,b\}^*\right\}$.

1.1.9. $L = \left\{\alpha \mid \alpha \in \{a,b\}^*, |\alpha|_a = |\alpha|_b\right\}$.

1.1.10. $L = \left\{\alpha \mid \alpha \in \{a,b\}^*, |\alpha|_a < |\alpha|_b\right\}$.

1.1.11. $L = \left\{\alpha \mid \alpha \in \{a,b\}^*, |\alpha|_a \geq |\alpha|_b\right\}$.

1.1.12. $L = \left\{\alpha \mid \alpha \in \{a,b\}^*, |\alpha|_a \neq |\alpha|_b\right\}$.

1.1.13. $L = \left\{0^i 1^j 2^{2(i+j)} \mid i, j \geq 0\right\}$.

1.1.14. $L = \{0^i 2^{3(i+j)} 1^j \mid i, j \geq 0\}$.

1.1.15. $L = \{2^{2(i+j)} 0^i 1^j \mid i, j \geq 0\}$.

1.1.16. $L = \{0^i 1^j \mid i \leq j \leq 2i\}$.

1.1.17. $L = \{0^i 1^j \mid j \leq i \leq 2j\}$.

1.1.18. $L = \{0^i 1^j 2^k \mid i = 2j$ или $j = 2k\}$.

1.1.19. $L = \{0^i 1^j 2^k \mid i = 2j$ или $k = 2j\}$.

1.1.20. $L = \{0^i 1^j 2^k \mid i = j$ или $j \neq 2k\}$.

1.1.21. $L = \{0^i 1^j 2^k \mid i \geq 2j$ или $j \geq 2k\}$.

1.1.22. Пусть $G = \langle \{a,b\}, \{S,A,B\}, P, S \rangle$ - КС-грамматика, где P состоит из правил:

$$S \to AB$$
$$A \to Aa \mid bB$$
$$B \to a \mid Sb.$$

Для слова α, принадлежащего $L(G)$, построить дерево вывода, левый и правый выводы.

а) $\alpha = baabaab$; б) $\alpha = bbaaaba$; в) $\alpha = babaabb$.

1.1.23. Пусть $G = \langle \{a, +, *, (,)\}, \{E, T, F\}, P, E \rangle$ - КС-грамматика с множеством правил P:

$$E \to E + T \mid T$$
$$T \to T * F \mid F$$
$$F \to (E) \mid a.$$

Грамматика G порождает множество арифметических выражений, содержащих переменную a, знаки арифметических операций $+$, * и правильно расставленные скобки.

Для слова α, принадлежащего $L(G)$, построить дерево вывода, левый и правый выводы.

а) $\alpha = a + a*(a + a + a)$; б) $\alpha = (a*(a + a) + a)*a$;

в) $\alpha = a*a + (a*a + a)*a$; г) $\alpha = a*a*(a + a + a)$;

д) $\alpha = (a + a)*(a + a*a)$; е) $\alpha = (a + a*(a + a))*a$.

1.2. Преобразования контекстно-свободных грамматик

Для одного и того же КС-языка могут существовать различные грамматики, поэтому возникает проблема выбора грамматики, наиболее подходящей по тем или иным свойствам, или проблема эквивалентного преобразования грамматики к нужному виду. Универсальных методов эквивалентных преобразований КС-грамматик не существует из-за неразрешимости алгоритмических проблем распознавания эквивалентности КС-грамматик и существенной неоднозначности КС-языков.

Рассмотрим несколько эквивалентных преобразований, которые с некоторой точки зрения улучшают грамматику.

Символ $X \in V_T \cup V_N$ называется ***полезным*** в КС-грамматике $G = \langle V_T, V_N, P, S \rangle$, если X появляется в процессе вывода из аксиомы S некоторого слова $w \in L(G)$, т.е. $S \Rightarrow^* \alpha X \beta \Rightarrow^* w$, где $w \in V_T^*$. Если символ X не является полезным, то он называется ***бесполезным***. Очевидно, что исключение бесполезных символов из грамматики не изменяет порождаемого языка, поэтому все бесполезные символы можно удалить.

Символ $X \in V_N$ называется ***порождающим***, если $X \Rightarrow^* w$ для некоторой терминальной цепочки w.

Символ $X \in V_T \cup V_N$ называется ***достижимым***, если существует вывод $S \Rightarrow^* \alpha X \beta$ для некоторых α и β.

Очевидно, что полезный нетерминальный символ грамматики является одновременно и порождающим, и достижимым. Можно доказать, что если сначала удалить из грамматики непорождающие символы, а затем недостижимые, то останутся только полезные.

Алгоритм 1.2.1. ***Устранение непорождающих символов.***

Шаг 1. Положить $N_0 = \varnothing$ и $i = 1$.

Шаг 2. $N_i = \left\{ A \middle| A \to \alpha \in P \text{ и } \alpha \in (V_T \cup N_{i-1})^* \right\} \cup N_{i-1}$.

Шаг 3. Если $N_i \neq N_{i-1}$, то положить $i = i + 1$ и перейти к шагу 2. В противном случае положить $N_e = N_i$ - множество порождающих символов.

Шаг 4. Если $N_e \subset V_N$, т.е. имеются непорождающие символы, следует перейти к грамматике $G_1 = \langle V_T, V_N', P', S \rangle$, в которой $V_N' = N_e \cap V_N$, и P' состоит из правил множества P, содержащих нетерминалы только из N_e.

С помощью алгоритма 1.2.1 может быть решена ***проблема пустоты для КС-языков***. Язык $L(G)$ непуст тогда и только тогда, когда S - порождающий символ, т. е. когда имеется вывод $S \Rightarrow^* w$ для некоторого слова $w \in V_T^*$. Другими словами, язык $L(G)$ непуст тогда и только тогда, когда $S \in N_e$.

Пример 1.2.1. Рассмотрим грамматику $G = \langle \{0,1\}, \{S, A, B\}, P, S \rangle$, где P состоит из правил

$$S \to 0 \mid A$$
$$A \to AB$$
$$B \to 1.$$

Применяя алгоритм устранения непорождающих символов к грамматике G, получаем множество $N_e = \{S, B\}$. Так как $S \in N_e$, то язык $L(G)$ непуст. Нетерминальный символ $A \notin N_e$, значит, он является непорождающим. Удалим из исходной грамматики все правила, содержащие символ A, в результате получим грамматику $G_1 = \langle \{0,1\}, \{S, B\}, \{S \to 0,\ B \to 1\}, S \rangle$, которая эквивалентна G и не содержит непорождающих символов.

Алгоритм 1.2.2. ***Устранение недостижимых символов.***

Шаг 1. Положить $V_0 = \{S\}$ и $i = 1$.

Шаг 2. $V_i = \{ X \mid A \to \alpha X \beta \in P \text{ и } A \in V_{i-1} \} \cup V_{i-1}$.

Шаг 3. Если $V_i \neq V_{i-1}$, то положить $i = i + 1$ и перейти к шагу 2. В противном случае положить $V_e = V_i$ - множество достижимых символов.

Шаг 4. Если $V_e \subset (V_T \cup V_N)$, т.е. имеются недостижимые символы, следует перейти к грамматике $G' = \langle V_T', V_N', P', S \rangle$, в которой $V_T' = V_e \cap V_T$, $V_N' = V_e \cap V_N$, и P' состоит из правил множества P, содержащих только символы из V_e.

Пример 1.2.2. Применив алгоритм устранения недостижимых символов к грамматике $G_1 = \langle \{0,1\}, \{S, B\}, \{S \to 0,\ B \to 1\}, S \rangle$, рассмотренной в примере 1.2.1, получим $V_e = V_2 = \{S, 0\}$, а это означает, что терминальный символ 1 и нетерминальный символ B - недостижимы. После удаления этих символов получим грамматику $G' = \langle \{0\}, \{S\}, \{S \to 0\}, S \rangle$, эквивалентную исходной грамматике G и не содержащую бесполезных символов.

Пример 1.2.3. Снова рассмотрим грамматику $G = \langle \{0,1\}, \{S, A, B\}, P, S \rangle$ из примера 1.2.1, где P состоит из правил

$S \to 0 \mid A$

$A \to AB$

$B \to 1.$

Теперь изменим порядок удаления бесполезных символов и начнем с алгоритма устранения недостижимых символов. В этом случае окажется, что все символы достижимы. Затем, применив алгоритм устранения непорождающих символов, получим грамматику $G_1 = \langle \{0,1\}, \{S, B\}, \{S \to 0,\ B \to 1\}, S \rangle$, отличную от $G' = \langle \{0\}, \{S\}, \{S \to 0\}, S \rangle$.

Алгоритм 1.2.3. ***Устранение бесполезных символов.***

Шаг 1. Применив к грамматике $G = \langle V_T, V_N, P, S \rangle$ алгоритм 1.2.1 устранения непорождающих символов, получить $G_1 = \langle V_T, V_N', P', S \rangle$.

Шаг 2. Применив к грамматике $G_1 = \langle V_T, V_N', P', S \rangle$ алгоритм 1.2.2 устранения недостижимых символов, получить $G' = \langle V_T', V_N', P', S \rangle$.

Теорема 1.2.1. *Грамматика G', которую строит алгоритм 1.2.3 по грамматике G, не содержит бесполезных символов, и* $L(G) = L(G')$.

λ-правилом называется правило вида $A \to \lambda$, где $A \in V_N$ и λ - пустое слово.

Контекстно-свободная грамматика $G = \langle V_T, V_N, P, S \rangle$ называется ***грамматикой без λ-правил*** (или ***неукорачивающей***), если либо

(1) P не содержит λ-правил (в случае, когда $\lambda \notin L(G)$), либо

(2) есть точно одно λ-правило $S \to \lambda$, и S не встречается в правых частях остальных правил из P (в случае, когда $\lambda \in L(G)$).

Алгоритм 1.2.4. ***Преобразование в грамматику без λ-правил.***

Пусть $G = \langle V_T, V_N, P, S \rangle$ - КС-грамматика с λ-правилами. Символ $X \in V_N$ называется ***λ-порождающим***, если $X \Rightarrow^* \lambda$.

Шаг 1. Положить $N_0 = \{ A \mid A \to \lambda \in P \}$ и $i = 1$.

Шаг 2. $N_i = \{ B \mid B \to \alpha \in P \text{ и } \alpha \in N_{i-1}^+ \} \cup N_{i-1}$.

Шаг 3. Если $N_i \neq N_{i-1}$, то положить $i = i + 1$ и перейти к шагу 2. В противном случае положить $N_\lambda = N_i$ - множество λ-порождающих символов.

Шаг 4. Построить множество правил P':

(1) Если $A \to \alpha_0 B_1 \alpha_1 B_2 \alpha_2 ... \alpha_{k-1} B_k \alpha_k \in P$, где $k \geq 0$ и $B_i \in N_\lambda$ для $i = \overline{1,k}$, но ни один символ в словах α_j $(j = \overline{0,k})$ не принадлежит N_λ, то включить в P' все правила вида $A \to \alpha_0 X_1 \alpha_1 X_2 \alpha_2 ... \alpha_{k-1} X_k \alpha_k$, где X_i - либо B_i, либо λ, но не включать правило $A \to \lambda$ (это могло бы произойти, если все α_i равны λ).

(2) Если $S \in N_\lambda$, то включить в P' правила $S' \to \lambda \mid S$, где S' - новый символ, и положить $V_N' = \{S'\} \cup V_N$. В противном случае положить $V_N' = V_N$ и $S' = S$.

Шаг 5. Положить $G' = \langle V_T, V_N', P', S' \rangle$; G' - эквивалентная КС-грамматика без λ-правил.

Теорема 1.2.2. *Грамматика G', которую строит алгоритм 1.2.4 по грамматике G, является грамматикой без λ-правил, и $L(G) = L(G')$.*

Пример 1.2.4. Рассмотрим грамматику $G = \langle \{a,b\}, \{S, A, B\}, P, S \rangle$, где P состоит из правил

$$S \to AB$$
$$A \to aAA \mid \lambda$$
$$B \to bBB \mid \lambda.$$

Сначала найдем λ-порождающие символы. $N_0 = \{A, B\}$ и $N_\lambda = \{S, A, B\}$. Построим теперь множество правил P'. Так как $S \to AB \in P$, то в P' следует включить следующие три правила: $S \to AB \mid A \mid B$. Правило $A \to aAA \in P$, значит в P' следует включить следующие правила: $A \to aAA \mid aA \mid a$. Аналогично, правило $B \to bBB \in P$ влечет правила в P': $B \to bBB \mid bB \mid b$. Символ $S \in N_\lambda$, поэтому следует включить в P' правила $S' \to \lambda \mid S$. Таким образом, грамматика $G' = \langle \{a,b\}, \{S', S, A, B\}, P', S' \rangle$, где P' состоит из правил

$$S' \to \lambda \mid S$$
$$S \to AB \mid A \mid B$$
$$A \to aAA \mid aA \mid a$$
$$B \to bBB \mid bB \mid b,$$

является грамматикой без λ-правил, причем $L(G) = L(G')$.

Цепным правилом называется правило вида $A \to B$, где $A, B \in V_N$.

Алгоритм 1.2.5. ***Устранение цепных правил.***

Пусть $G = \langle V_T, V_N, P, S \rangle$ - КС-грамматика без λ-правил.

Шаг 1. Для каждого $A \in V_N$ построить $N_A = \left\{ B \middle| A \Rightarrow^* B \right\}$ следующим образом:

(1) Положить $N_0 = \{ A \}$ и $i = 1$.

(2) Положить $N_i = \left\{ C \middle| B \to C \in P \text{ и } B \in N_{i-1} \right\} \cup N_{i-1}$.

(3) Если $N_i \neq N_{i-1}$, то положить $i = i + 1$ и перейти к шагу 2. В противном случае положить $N_A = N_i$.

Шаг 2. Построить множество правил P': если $B \to \alpha \in P$ и не является цепным правилом, то включить в P' правило $A \to \alpha$ для всех таких A, что $B \in N_A$.

Шаг 3. Положить $G' = \langle V_T, V_N, P', S \rangle$; G' - эквивалентная КС-грамматика без λ-правил и без цепных правил.

Теорема 1.2.3. *Грамматика G', которую строит алгоритм 1.2.5 по грамматике G, не имеет цепных правил, и $L(G) = L(G')$.*

Пример 1.2.5. Рассмотрим грамматику $G = \langle \{a, b\}, \{S, A, B, C\}, P, S \rangle$, где P состоит из правил

$$S \to aB \mid bA \mid B$$
$$A \to aa \mid bBb \mid C$$
$$B \to ab \mid C \mid aAb$$
$$C \to aBa \mid ab.$$

Для нетерминального символа S находим множество $N_S = \{S, B, C\}$, для A - множество $N_A = \{A, C\}$, для B - множество $N_B = \{B, C\}$, для C - множество $N_C = \{C\}$. Построим новое множество правил P':

$$S \to aB \mid bA \mid ab \mid aBa \mid aAb$$
$$A \to aa \mid bBb \mid aBa \mid ab$$
$$B \to ab \mid aBa \mid aAb$$
$$C \to aBa \mid ab.$$

Грамматика $G' = \langle \{a,b\}, \{S, A, B, C\}, P', S \rangle$ не содержит цепных правил и эквивалентна исходной грамматике G.

КС-грамматика $G = \langle V_T, V_N, P, S \rangle$ называется ***приведенной грамматикой*** (или ***грамматикой в приведенной форме***), если множество ее правил вывода P не содержит λ-правил, цепных правил и бесполезных символов.

Рассмотренные выше алгоритмы 1.2.1 – 1.2.5 любую КС-грамматику позволяют преобразовать к эквивалентной приведенной КС-грамматике.

Задачи и упражнения

1.2.1. Найти грамматику, не содержащую бесполезных символов и эквивалентную грамматике с правилами

$$S \to AB \,|\, CA$$
$$A \to a$$
$$B \to BC \,|\, AB$$
$$C \to aB \,|\, b.$$

1.2.2. Найти грамматику, не содержащую бесполезных символов и эквивалентную грамматике с правилами

$$S \to A \,|\, C$$
$$A \to aB \,|\, bS \,|\, b$$
$$B \to AB \,|\, Ba$$
$$C \to AS \,|\, b\ .$$

1.2.3. Найти грамматику, не содержащую бесполезных символов и эквивалентную грамматике с правилами

$$S \to aBCa \,|\, bADb$$
$$A \to aAa \,|\, bEa$$
$$B \to aBa \,|\, bSBb$$
$$C \to cB \,|\, cE$$
$$D \to aEa \,|\, bD \,|\, bBa$$
$$E \to bS \,|\, aba.$$

1.2.4. Найти грамматику, не содержащую бесполезных символов и эквивалентную грамматике с правилами

$$S \to aSb \,|\, bA \,|\, aDEa \,|\, aFE$$
$$A \to aB \,|\, Bba \,|\, aa$$
$$B \to Ca \,|\, bC$$
$$C \to bE \,|\, Bb \,|\, Da$$
$$D \to ab \,|\, ba$$
$$E \to aEA$$
$$F \to ab.$$

1.2.5. Найти приведенную грамматику, эквивалентную грамматике с правилами

$$S \to ASB \,|\, \lambda$$
$$A \to aAS \,|\, a$$
$$B \to SbS \,|\, A \,|\, bb.$$

1.2.6. Найти приведенную грамматику, эквивалентную грамматике с правилами

$$S \to AAA \,|\, B$$
$$A \to aA \,|\, B$$
$$B \to \lambda.$$

1.2.7. Найти приведенную грамматику, эквивалентную грамматике с правилами

$$S \to ABa \,|\, aAb$$
$$A \to aA \,|\, ab \,|\, \lambda$$
$$B \to ba \,|\, Ba \,|\, \lambda.$$

1.2.8. Найти приведенную грамматику, эквивалентную грамматике с правилами

$$S \to aB \,|\, bA \,|\, B$$
$$A \to bBb \,|\, C \,|\, aa$$
$$B \to aAb \,|\, C \,|\, ab$$
$$C \to aBa \,|\, ab.$$

1.2.9. Найти приведенную грамматику, эквивалентную грамматике с правилами

$$S \to aAa \,|\, bBb \,|\, BB$$
$$A \to C$$

$$B \to S \mid A$$
$$C \to S \mid \lambda .$$

1.2.10. Найти приведенную грамматику, эквивалентную грамматике с правилами

$$S \to aAa \mid bBb \mid \lambda$$
$$A \to C \mid a$$
$$B \to C \mid b$$
$$C \to CD \mid \lambda$$
$$D \to A \mid B \mid ab.$$

1.2.11. Найти приведенную грамматику, эквивалентную грамматике с правилами

$$S \to ABC \mid aA \mid bB \mid \lambda$$
$$A \to B \mid b \mid aB \mid \lambda$$
$$B \to C \mid bA \mid a$$
$$C \to B \mid bS \mid \lambda .$$

1.2.12. Найти приведенную грамматику, эквивалентную грамматике с правилами

$$S \to A \mid B$$
$$A \to C \mid D$$
$$B \to D \mid E$$
$$C \to S \mid a \mid \lambda$$
$$D \to S \mid b$$
$$C \to S \mid c \mid \lambda .$$

1.3. Лемма Огдена и лемма о разрастании для КС-языков

Позицией в слове длины k назовем такое целое i, что $1 \le i \le k$. Символ a занимает позицию i в слове w, если $w = w_1 a w_2$ и $|w_1| = i - 1$.

Лемма Огдена. *Пусть* $G = \langle V_T, V_N, P, S \rangle$ *- произвольная КС-грамматика. Существует такая константа* $k \ge 1$, *что если* $z \in L(G)$, $|z| \ge k$ *и в слове* z

выделены k *или более позиций, то* z *можно записать в виде* $z = uvwxy$, *причем*

1) w *содержит хотя бы одну выделенную позицию;*

2) либо слова u *и* v *оба содержат выделенные позиции, либо* x *и* y *оба содержат выделенные позиции;*

3) vwx *содержит не более* k *выделенных позиций;*

4) существует такой нетерминал A, *что* $S \Rightarrow^+ uAy \Rightarrow^+ uvAxy \Rightarrow^+ \ldots$ $\ldots \Rightarrow^+ uv^i Ax^i y \Rightarrow^+ uv^i wx^i y$ *для всех* $i \geq 0$.

Следствием леммы Огдена является

Лемма о разрастании для КС-языков. *Пусть* L *- КС-язык. Тогда существует такая константа* $k \geq 1$, *что если* $z \in L(G)$ *и* $|z| \geq k$, *то* z *можно записать в виде* $z = uvwxy$, *где* $vx \neq \lambda$, $|vwx| \leq k$ *и* $uv^i wx^i y \in L$ *для всех* $i \geq 0$.

Лемма Огдена и лемма о разрастании для КС-языков полезны при доказательстве утверждений о том, что некоторые языки не являются контекстно-свободными.

Пример 1.3.1. Пусть $L = \{a^n b^n c^n \mid n \geq 1\}$. Используя лемму о разрастании, докажем, что L - не КС-язык.

Доказательство проведем методом от противного. Предположим, что L - КС-язык. Рассмотрим слово $z = a^k b^k c^k$, где k - константа из леммы о разрастании. Тогда z можно представить в виде $z = uvwxy$, где $|vwx| \leq k$. В слове vwx не может быть вхождения каждого из символов a, b и c, так как при этом нарушалось бы условие $|vwx| \leq k$. Слово $z' = uwy$ принадлежит L по лемме о разрастании и содержит либо k символов a, либо k символов c. Но оно не может иметь k вхождений каждого из символов a, b, c, так как $|z'| < |z|$ и потому $|uwy| < 3k$. Следовательно, $uwy \notin L$. Полученное противоречие доказывает, что предположение о том, что L - КС-язык, неверно. Следовательно, язык $L = \{a^n b^n c^n \mid n \geq 1\}$ не является КС-языком.

Пример 1.3.2. Пусть $L = \{a^i b^j c^m \mid i, j, m \geq 1,\ i < j, m \neq j\}$. Докажем, что L - не КС-язык, используя лемму Огдена.

Как и в примере 1.3.1, доказательство проведем методом от противного. Предположим, что L - КС-язык. Тогда к нему можно применить лемму Огдена. Рассмотрим слово $z = a^{k!-1} b^{k!} c^{2 \cdot k!} \in L$, где k - константа из леммы. Очевидно, $|z| > k$. Выделим все $k!$ позиций, которые занимают буквы b.

По лемме Огдена слово z можно представить в виде $z = uvwxy$. Покажем, что слово v, так же как и слово x, не может содержать двух различных букв алфавита $\{a,b,c\}$. Действительно, предположим для определенности, что v содержит буквы a и b. Рассмотрим слово $z' = uv^2wx^2y$, которое по лемме Огдена принадлежит языку L. Слово v имеет вид a^lb^n, следовательно, $v^2 = a^lb^na^lb^n$, и в z' будет происходить чередование букв a и b, что невозможно в L.

Рассмотрим теперь случаи, когда каждое из слов v и x состоит из одинаковых букв. По лемме Огдена либо слово v, либо слово x содержат выделенные позиции, поэтому хотя бы одно из них состоит из букв b.

Пусть $v = b^l$, $l \geq 1$. Тогда x не может содержать букву a.

Рассмотрим слово $z' = uwy$, которое по лемме Огдена должно принадлежать L. Но слово v содержит хотя бы одну букву b, поэтому z' содержит хотя бы на одну букву b меньше, чем z, значит, $|z'|_a \geq |z'|_b$ и $z' \notin L$. Получаем противоречие.

Пусть $x = b^l$, $l \geq 1$. Тогда v не может содержать букву c. Обозначим через n общее число вхождений буквы b в слова v и x.

Подберем такое слово $z' = uv^rwx^ry$, для которого $|z'|_b = |z'|_c$. Найдем r из условия $k! + (r-1)\cdot n = 2\cdot k!$, выполнение которого уравнивает число букв b и c в z'. Получаем $r = \dfrac{k!+n}{n}$. По лемме Огдена слово vwx содержит не более k выделенных позиций (на них стоят буквы b), поэтому n не превосходит k. Отсюда следует, что $k!+n$ делится на n без остатка и r - целое число. По лемме $z' = uv^rwx^ry \in L$; с другой стороны, z' содержит одинаковое число букв b и c и потому $z' \notin L$. Получаем противоречие.

Таким образом, для каждого из возможных случаев мы получили противоречие, поэтому предположение о том, что L - КС-язык, неверно. Следовательно, язык $L = \left\{ a^ib^jc^m \mid i, j, m \geq 1, i < j, m \neq j \right\}$ не является КС-языком.

Задачи и упражнения

Используя лемму Огдена или лемму о разрастании, доказать, что следующие языки не являются КС-языками.

1.3.1. $L = \left\{ a^m b^m c^j \mid m, j \geq 1,\ j < m \right\}.$

1.3.2. $L = \left\{ a^i b^j c^k \mid i, j, k \geq 1,\ i < j < k \right\}.$

1.3.3. $L = \left\{ a^i b^j c^k \mid i, j, k \geq 1,\ i \geq j \geq k \right\}.$

1.3.4. $L = \left\{ \alpha \in \{a,b,c\}^* \mid |\alpha|_a = |\alpha|_b = |\alpha|_c \right\}$ (здесь $|\alpha|_x$ - число вхождений буквы x в слово α).

1.3.5. $L = \left\{ 0^m 1^n 0^m 1^n \mid m, n \geq 1 \right\}.$

1.3.6. $L = \left\{ 0^i 1^j 2^k \mid i, j, k \geq 1,\ i \geq j,\ j \neq k \right\}.$

1.3.7. $L = \left\{ a^i b^j c^m \mid i, j, m \geq 1,\ i \leq j, m \neq j \right\}.$

1.3.8. $L = \left\{ a^i b^j a^m \mid i, j, m \geq 1,\ i \leq j, m \neq i \right\}.$

1.3.9. $L = \left\{ a^i b^j c^k \mid i, j, k \geq 1,\ i \neq j, i \neq k,\ j \neq k \right\}.$

1.3.10. $L = \left\{ a^{n^2} \mid n \geq 1 \right\}.$

1.3.11. $L = \left\{ a^n \mid n - простое\ число \right\}.$

1.3.12. $L = \left\{ \alpha\alpha \mid \alpha \in \{0,1\}^* \right\}.$

1.4. Свойства замкнутости контекстно-свободных языков

Рассмотрим некоторые операции над КС-языками, относительно которых класс КС-языков замкнут. Напомним, что множество всех регулярных языков замкнуто относительно основных теоретико-множественных операций (объединения, пересечения, дополнения, разности), конкатенации, возведения в степень и итерации.

Теорема 1.4.1. *Класс контекстно-свободных языков замкнут относительно объединения, конкатенации, возведения в степень и итерации.*

Пусть $G_1 = \left\langle V_T, V_N^1, P_1, S_1 \right\rangle$, $G_2 = \left\langle V_T, V_N^2, P_2, S_2 \right\rangle$ - КС-грамматики, и $V_N^1 \cap V_N^2 = \varnothing.$

1) Построим $G_3 = \left\langle V_T, V_N^1 \cup V_N^2 \cup \{S\}, P_1 \cup P_2 \cup \{S \to S_1 \mid S_2\}, S \right\rangle$, где $S \notin V_N^1 \cup V_N^2$, тогда $L(G_3) = L(G_1) \cup L(G_2)$.

2) Построим $G_4 = \left\langle V_T, V_N^1 \cup V_N^2 \cup \{S\}, P_1 \cup P_2 \cup \{S \to S_1 S_2\}, S \right\rangle$ где $S \notin V_N^1 \cup V_N^2$, тогда $L(G_4) = L(G_1) \cdot L(G_2)$.

3) Построим $G_5 = \left\langle V_T, V_N^1 \cup \{S\}, P_1 \cup \{S \to \underbrace{S_1 S_1 ... S_1}_{n\ раз}\}, S \right\rangle$, где $S \notin V_N^1$, тогда $L(G_5) = (L(G_1))^n$.

4) Построим $G_6 = \left\langle V_T, V_N^1 \cup \{S\}, P_1 \cup \{S \to S_1 S \mid \lambda\}, S \right\rangle$, где $S \notin V_N^1$, тогда $L(G_5) = (L(G_1))^*$.

Пример 1.4.1. Пусть $L_1 = \left\{ a^n b^n c^i \mid n \geq 1, i \geq 1 \right\}$ и $L_2 = \left\{ a^i b^n c^n \mid n \geq 1, i \geq 1 \right\}$. Покажем, что $L_1 \cup L_2$ является КС-языком, построив для него КС-грамматику.

Зададим каждый из этих языков КС-грамматикой. Язык L_1 порождается грамматикой $G_1 = \langle \{a, b, c\}, \{S_1, A\}, P_1, S_1 \rangle$ с правилами P_1:

$$S_1 \to S_1 c \mid A$$
$$A \to aAb \mid ab.$$

Язык L_2 порождается грамматикой $G_2 = \langle \{a, b, c\}, \{S_2, B\}, P_2, S_2 \rangle$ с правилами P_2:

$$S_2 \to aS_2 \mid B$$
$$B \to bBc \mid bc.$$

Рассмотрим $L = L_1 \cup L_2$ и построим для этого языка КС-грамматику $G = \langle \{a, b, c\}, \{S, S_1, S_2, A, B\}, P, S \rangle$ с правилами P:

$$S \to S_1 \mid S_2$$
$$S_1 \to S_1 c \mid A$$
$$S_2 \to aS_2 \mid B$$
$$A \to aAb \mid ab$$
$$B \to bBc \mid bc.$$

В этой грамматике S_1 порождает слова языка L_1, а S_2 - слова L_2.

Теорема 1.4.2. *Класс контекстно-свободных языков не замкнут относительно пересечения*.

Доказательство этой теоремы вытекает из следующего примера.

Пример 1.4.2. Пусть $L_1 = \{a^n b^n c^i \mid n \geq 1, i \geq 1\}$ и $L_2 = \{a^i b^n c^n \mid n \geq 1, i \geq 1\}$. Оба этих языка являются контекстно-свободными (см. пример 1.4.1), но язык $L = L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 1\}$ не является КС-языком (см. пример 1.3.1). Таким образом, пересечение КС-языков не всегда является КС-языком.

Теорема 1.4.3. *Класс контекстно-свободных языков не замкнут относительно дополнения*.

Пусть L_1 и L_2 - произвольные КС-языки. Если бы утверждение теоремы 1.4.3 было неверным, отсюда следовало бы, что $\overline{L}_1$ и $\overline{L}_2$ - КС-языки. Тогда в силу замкнутости класса КС-языков относительно объединения и дополнения получили бы, что $\overline{\overline{L}_1 \cup \overline{L}_2}$ - КС-язык. По закону де Моргана $\overline{\overline{L}_1 \cup \overline{L}_2} = L_1 \cap L_2$, т.е. $L_1 \cap L_2$ - КС-язык для произвольных КС-языков L_1 и L_2, а это противоречит утверждению теоремы 1.4.2.

Теорема 1.4.4. *Класс контекстно-свободных языков не замкнут относительно разности.*

Очевидно, универсальный язык V_T^* является КС-языком для любого алфавита V_T ; нетрудно построить КС-грамматику для этого регулярного языка. Тогда, если бы язык $L_1 - L_2$ был контекстно-свободным для любых КС-языков L_1 и L_2, то и $V_T^* - L$ должен быть КС-языком, если L - КС-язык. Однако по определению $V_T^* - L = \overline{L}$, откуда получаем, что разность двух КС-языков не обязательно является КС-языком.

Задачи и упражнения

1.4.1. Пусть $L_1 = \{a^n b^n c^i \mid n \geq 1, i \geq 1\}$ и $L_2 = \{a^i b^n c^n \mid n \geq 1, i \geq 1\}$. Показать, что $L_1 L_2$ является КС-языком, построив для него КС-грамматику.

1.4.2. Пусть $L = \{a^n b^n c^i \mid n \geq 1, i \geq 1\}$. Показать, что L^2 является КС-языком, построив для него КС-грамматику.

1.4.3. Пусть $L = \{a^n b^n c^i \mid n \geq 1, i \geq 1\}$. Показать, что L^* является КС-языком, построив для него КС-грамматику.

1.4.4. Пусть $L_1 = \{a^n b^{2n} c^m \mid n, m \geq 1\}$ и $L_2 = \{a^n b^m c^{2m} \mid n, m \geq 1\}$. Показать, что каждый из них является КС-языком, построив для них КС-грамматики. Является ли $L_1 \cap L_2$ КС-языком?

1.4.5. Пусть $L_1 = \{a^{2n} b^n c^m \mid n, m \geq 1\}$ и $L_2 = \{a^n b^m c^n \mid n, m \geq 1\}$. Показать, что каждый из них является КС-языком, построив для них КС-грамматики. Является ли $L_1 \cap L_2$ КС-языком?

1.4.6. Пусть $L_1 = \{a^{2n} b^n c^m \mid n, m \geq 1\}$ и $L_2 = \{a^n b^m c^n \mid n, m \geq 1\}$. Показать, что $L_1 \cup L_2$ является КС-языком, построив для него КС-грамматику.

1.4.7. Пусть $L_1 = \{a^{2n} b^n c^m \mid n, m \geq 1\}$ и $L_2 = \{a^n b^m c^n \mid n, m \geq 1\}$. Показать, что $L_1 L_2$ является КС-языком, построив для него КС-грамматику.

1.4.8. Пусть $L_1 = \{0^i 1^j 2^j 3^i \mid i, j \geq 0\}$ и $L_2 = \{0^i 1^i 2^j 3^j \mid i, j \geq 0\}$. Показать, что каждый из них является КС-языком, построив для них КС-грамматики. Является ли $L_1 \cap L_2$ КС-языком?

1.4.9. Пусть $L = \{a^n b^m c^{2(n+m)} \mid n, m \geq 1\}$. Показать, что L^* является КС-языком, построив для него КС-грамматику.

1.4.10. Показать, что язык $\{a^i b^j a^k \mid i, j, k \geq 1\} - \{a^n b^n a^n \mid n \geq 1\}$ является КС-языком.

Глава 2. Автоматы с магазинной памятью

2.1. Основное определение автомата с магазинной памятью. Варианты автоматов с магазинной памятью

Рассмотрим сначала неформальное описание автомата с магазинной памятью, который можно рассматривать как устройство, представленное на рис. 3.

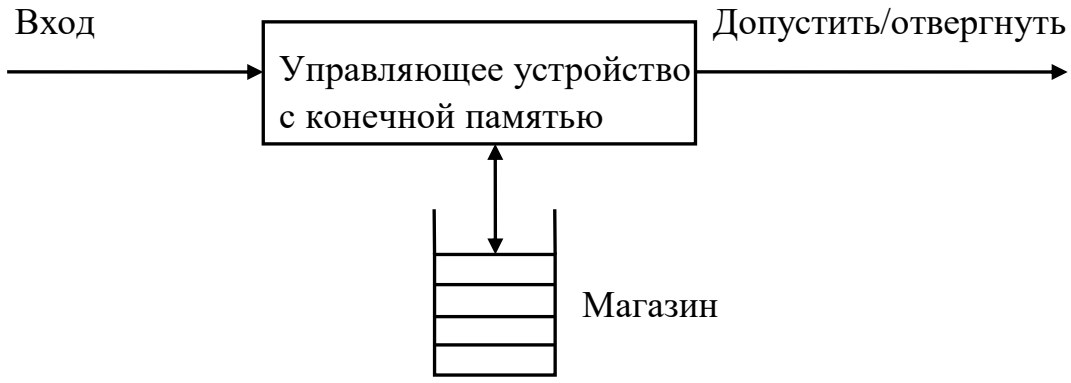

Рис. 3. Автомат с магазинной памятью

Автомат с магазинной памятью – это, по существу, недетерминированный конечный автомат, имеющий дополнительную потенциально неограниченную ленту памяти (магазин). Доступ к информации в магазине возможен только с одного его конца в соответствии с принципом "последним пришел – первым ушел". В начальный момент работы автомата магазин содержит начальный символ Z_0.

Автомат с магазинной памятью будем рассматривать как устройство, которое читает входные символы по одному или может также в качестве входного символа использовать λ (т.е. может не читать входной символ в некоторые моменты времени, совершая внутреннюю работу), обозревать символ на вершине магазина и совершать переход на основе текущего состояния, входного символа и символа на вершине магазина.

За один такт автомат совершает следующие действия.

1. Прочитывает входной символ, перемещаясь на позицию вправо вдоль входной ленты; если в качестве входного символа используется λ, перемещения вправо не происходит.

2. Переходит в новое состояние, которое может и не отличаться от предыдущего.

3. Заменяет символ на вершине магазина некоторой последовательностью магазинных символов, возможно и пустой, что соответствует выталкиванию вершины магазина.

Более формально определение выглядит так.

Автомат с магазинной памятью (сокращенно ***МП-автомат***) − это система $M = \langle Q, A, \Gamma, \delta, q_0, Z_0, F \rangle$, где

Q − конечное ***множество состояний***,

A − конечный ***входной алфавит***,

Γ − конечный ***алфавит магазинных символов***,

$\delta : Q \times (A \cup \{\lambda\}) \times \Gamma \to 2^{Q \times \Gamma^*}$ − ***функция переходов***,

$q_0 \in Q$ − ***начальное состояние***,

Z_0 − ***начальный магазинный символ*** ("маркер дна"),

$F \subseteq Q$ − множество ***заключительных состояний***.

Конфигурация МП-автомата M − это тройка $(q, w, \alpha) \in Q \times A^* \times \Gamma^*$, где

q − текущее состояние;

w − оставшаяся часть входного слова; если $w = \lambda$, то считается, что все входное слово прочитано;

α − содержимое магазина, причем вершина магазина изображается слева, а дно – справа; если $\alpha = \lambda$, то магазин считается пустым.

Начальная конфигурация – конфигурация вида (q_0, w, Z_0), где $w \in A^*$.

З***аключительная конфигурация*** – конфигурация вида (q, λ, α), где $q \in F$ и $\alpha \in \Gamma^*$.

Определим на множестве всех конфигураций МП-автомата M бинарное отношение $\vdash$ (***такт работы*** МП-автомата M) следующим образом.

Будем писать $(q, aw, Z\alpha) \vdash (q', w, \gamma\alpha)$, если множество $\delta(q, a, Z)$ содержит (q', γ), где $q, q' \in Q$, $a \in A \cup \{\lambda\}$, $w \in A^*$, $Z \in \Gamma$ и $\alpha, \gamma \in \Gamma^*$.

Следующий такт работы МП-автомата невозможен, если магазин пуст.

Запись $C \vdash^0 C'$ означает, что $C = C'$, а $C_0 \vdash^k C_k$ (для $k \geq 1$) − что существуют такие конфигурации $C_1, ..., C_{k-1}$, что $C_i \vdash C_{i+1}$ для всех $0 \leq i < k$. Бинарное отношение $\vdash^*$ определяется как ***рефлексивное и транзитивное замыкание*** отношения $\vdash$, т.е. $C \vdash^* C'$ означает, что $C \vdash^k C'$ для некоторого $k \geq 0$. Бинарное отношение $\vdash^+$ определяется как

транзитивное замыкание отношения $\vdash$, т.е. $C \vdash^{+} C'$ означает, что $C \vdash^{k} C'$ для некоторого $k \geq 1$.

Будем говорить, что МП-автомат M ***допускает*** (или ***распознает***) слово w, если $(q_0, w, Z_0) \vdash^{*} (q, \lambda, \alpha)$ для некоторых $q \in F$ и $\alpha \in \Gamma^{*}$.

Язык, допускаемый (или ***распознаваемый***, или ***определяемый***) автоматом с магазинной памятью M (обозначается $L(M)$), – это множество всех входных цепочек, допускаемых МП-автоматом M, т.е.

$$L(M) = \left\{ w \,\middle|\, w \in A^{*} \text{ и } (q_0, w, Z_0) \vdash^{*} (q, \lambda, \alpha), \text{ где } q \in F,\ \alpha \in \Gamma^{*} \right\}.$$

$L(M)$ будем также называть также ***языком, допускаемым M по заключительному состоянию***.

Удобно задавать функцию переходов $\delta(q, a, Z)$ с помощью таблицы переходов.

Таблица переходов автомата с магазинной памятью представляет собой табличное задание функции переходов δ, которая трем своим аргументам $q \in Q$, $a \in A \cup \{\lambda\}$ и $Z \in \Gamma$ ставит в соответствие значение $\delta(q, a, Z)$. Первые три столбца таблицы соответствуют аргументам функции δ, а последний столбец содержит подмножество значений функции $\delta(q, a, Z)$, заключенных в фигурные скобки. Договоримся фигурные скобки опускать, если каждое из подмножеств содержит не более одного элемента.

Пример 2.1.1. Построим автомат с магазинной памятью, допускающий язык $L = \left\{ a^n b^n \mid n \geq 1 \right\}$.

Работа МП-автомата M состоит в том, что он, находясь в состоянии q_0 и считывая из входного слова букву a, дублирует этот символ в магазине и переходит в состояние q_1.

Переход в новое состояние гарантирует, что буква b не сможет появиться перед буквой a. Затем МП-автомат продолжает после каждого считывания из входного потока буквы a заносить в магазин символ a.

Наконец, при считывании из входного слова буквы b автомат стирает из магазина один символ a и переходит в состояние q_2. Переход в новое состояние гарантирует, что буква a не сможет появиться после буквы b. Далее МП-автомат продолжает после каждого считывания из входного потока буквы b стирать из магазина символ a.

Когда в магазине останется только символ Z_0, и автомат будет находиться в состоянии q_2 (это означает, что обнаружено входное слово вида $a^n b^n$ $(n \geq 1)$), МП-автомат M перейдет в заключительное состояние q_3.

Таким образом, МП-автомат M может быть формально задан как $M = \langle \{q_0, q_1, q_2, q_3\}, \{a, b\}, \{Z_0, 0\}, \delta, q_0, Z_0, \{q_3\} \rangle$, где функция переходов δ определяется табл. 1.

Таблица 1

Таблица значений функции переходов для примера 2.1.1

Q	$A \cup \{\lambda\}$	Γ	δ
q_0	a	Z_0	$(q_1, 0Z_0)$
q_1	a	0	$(q_1, 00)$
q_1	b	0	(q_2, λ)
q_2	b	0	(q_2, λ)
q_2	λ	Z_0	(q_3, Z_0)

Построим для входного слова $aabb$ последовательность конфигураций МП-автомата M: $(q_0, aabb, Z_0) \vdash (q_1, abb, 0Z_0) \vdash (q_1, bb, 00Z_0) \vdash \vdash (q_2, b, 0Z_0) \vdash (q_2, \lambda, Z_0) \vdash (q_3, \lambda, Z_0)$.

Так как $(q_0, aabb, Z_0) \vdash^* (q_3, \lambda, Z_0)$ и $q_3 \in F$, слово $aabb \in L(M)$.

Пример 2.1.2. Построим автомат с магазинной памятью, допускающий язык $L = \left\{ ww^R \mid w \in \{a,b\}^+ \right\}$, который состоит из палиндромов четной длины над алфавитом $\{a,b\}$.

Работа автомата начинается в состоянии q_0, подразумевающем, что не достигнута середина входного слова, т.е. конец слова w, за которым должно следовать его обращение. В состоянии q_0 читаются символы входного слова и их копии записываются в магазин.

Кроме того, в любой момент времени при условии, что входной символ совпадает с вершиной магазина, автомат может перейти в состояние q_1 и начать сравнивать содержимое в магазине с оставшейся частью входного слова. Если M обнаруживает несовпадение очередных символов, то предположение о достигнутой середине слова неверно. Так как автомат M - недетерминированный, то эта ветвь вычислений отбрасывается, хотя другие могут продолжаться и вести к тому, что слово допускается.

Если обнаружен маркер дна магазина Z_0 и автомат находится в состоянии q_1, то автомат переходит в заключительное состояние q_2. Таким образом, M допускает слово тогда и только тогда, когда все сравнения обнаружили совпадение символов.

МП-автомат M может быть формально задан как $\langle\{q_0, q_1, q_2\}, \{a, b\}, \{Z_0, a, b\}, \delta, q_0, Z_0, \{q_2\}\rangle$, где функция переходов δ определяется табл. 2.

Таблица 2

Таблица значений функции переходов из примера 2.1.2

Q	$A \cup \{\lambda\}$	Γ	δ
q_0	a	Z_0	$\{(q_0, aZ_0)\}$
q_0	b	Z_0	$\{(q_0, bZ_0)\}$
q_0	a	a	$\{(q_0, aa), (q_1, \lambda)\}$
q_0	a	b	$\{(q_0, ab)\}$
q_0	b	a	$\{(q_0, ba)\}$
q_0	b	b	$\{(q_0, bb), (q_1, \lambda)\}$
q_1	a	a	$\{(q_1, \lambda)\}$
q_1	b	b	$\{(q_1, \lambda)\}$
q_1	λ	Z_0	$\{(q_2, \lambda)\}$

Рассмотрим слово $baab \in L$. Для входного слова $baab$ работе автомата может соответствовать следующая последовательность конфигураций:

$(q_0, baab, Z_0) \vdash (q_0, aab, bZ_0) \vdash (q_0, ab, abZ_0) \vdash (q_0, b, aabZ_0) \vdash$
$\vdash (q_0, \lambda, baabZ_0)$.

Так как эта последовательность не заканчивается заключительным состоянием, то она является ошибочной. Рассмотрим другую возможную последовательность конфигураций для входного слова $baab$:

$(q_0, baab, Z_0) \vdash (q_0, aab, bZ_0) \vdash (q_0, ab, abZ_0) \vdash (q_1, b, bZ_0) \vdash$
$\vdash (q_1, \lambda, Z_0) \vdash (q_2, \lambda, \lambda)$.

Здесь приходим в заключительную конфигурацию и делаем вывод о том, что МП-автомат M допускает слово $baab$.

Рассмотрим теперь определение расширенного МП-автомата, который за один такт может заменять цепочку символов ограниченной длины, расположенную в верхней части магазина, другой цепочкой конечной длины.

Расширенным автоматом с магазинной памятью называется система $M = \langle Q, A, \Gamma, \delta, q_0, Z_0, F \rangle$, где ***функция переходов*** определяется как $\delta : Q \times (A \cup \{\lambda\}) \times \Gamma^* \to 2^{Q \times \Gamma^*}$, а все остальные символы имеют тот же смысл, что и раньше.

Конфигурация расширенного МП-автомата определяется так же, как и прежде. Будем писать $(q, aw, \alpha\gamma) \vdash (q', w, \beta\gamma)$, если $\delta(q, a, \alpha)$ содержит (q', β), где $q, q' \in Q$, $a \in A \cup \{\lambda\}$, $w \in A^*$ и $\alpha, \beta, \gamma \in \Gamma^*$. В этом такте цепочка α, расположенная в верхней части магазина, заменяется цепочкой β. Заметим, что в отличие от обычного МП-автомата расширенный МП-автомат обладает способностью продолжать работу и тогда, когда магазин пуст.

Теорема 2.1.1. *Язык L допускается МП-автоматом тогда и только тогда, когда он допускается расширенным МП-автоматом.*

Пример 2.1.3. Построим расширенный МП-автомат, допускающий язык $L = \left\{ ww^R \mid w \in \{a, b\}^* \right\}$.

Пусть $M = \langle \{q_0, q_1\}, \{a, b\}, \{Z_0, a, b, S\}, \delta, q_0, Z_0, \{q_1\} \rangle$, где функция переходов δ определяется таблицей 3.

Таблица 3

Таблица значений функции переходов из примера 2.1.3

Q	$A \cup \{\lambda\}$	Γ	δ
q_0	a	λ	(q_0, a)
q_0	b	λ	(q_0, b)
q_0	λ	λ	(q_0, S)
q_0	λ	aSa	(q_0, S)
q_0	λ	bSb	(q_0, S)
q_0	λ	SZ_0	(q_1, λ)

Работа расширенного МП-автомата M состоит в том, что вначале он запасает в магазине некоторый префикс входного слова. Затем верхним символом магазина делается S, который указывает предполагаемую середину входного слова. Далее автомат помещает в магазин очередной входной символ

и заменяет в магазине aSa или bSb на S. Автомат M работает до тех пор, пока не будет прочитано все слово. Если после этого в магазине останется SZ_0, тогда M сотрет SZ_0 и попадет в заключительное состояние.

Например, для входного слова $baab$ автомат может выполнить следующую последовательность конфигураций:
$(q_0, baab, Z_0) \vdash (q_0, aab, bZ_0) \vdash (q_0, aab, SbZ_0) \vdash (q_0, ab, aSbZ_0)$. Она является ошибочной. Другая последовательность конфигураций для этого же слова $baab$ приводит МП-автомат в заключительное состояние, т.е. M допускает входную цепочку $baab$:
$(q_0, baab, Z_0) \vdash (q_0, aab, bZ_0) \vdash (q_0, ab, abZ_0) \vdash (q_0, ab, SabZ_0) \vdash$
$\vdash (q_0, b, aSabZ_0) \vdash (q_0, b, SbZ_0) \vdash (q_0, \lambda, bSbZ_0) \vdash (q_0, \lambda, SZ_0) \vdash$
$\vdash (q_1, \lambda, \lambda)$.

Рассмотрим еще одно определение языков, допускаемых МП-автоматами.

Пусть $M = \langle Q, A, \Gamma, \delta, q_0, Z_0, F \rangle$ - МП-автомат.

$L_\lambda(M) = \{ w \mid w \in A^* \quad \text{и} \quad (q_0, w, Z_0) \vdash^* (q, \lambda, \lambda), \quad \text{где} \quad q \in Q \}$, будем называть ***языком, допускаемым M опустошением магазина***.

Лемма 2.1.1. *Пусть язык $L = L(M)$ допускается некоторым МП-автоматом $M = \langle Q, A, \Gamma, \delta, q_0, Z_0, F \rangle$ по заключительному состоянию. Тогда можно построить такой МП-автомат M', что $L_\lambda(M') = L$.*

Лемма 2.1.2. *Пусть язык $L = L_\lambda(M)$ допускается некоторым МП-автоматом $M = \langle Q, A, \Gamma, \delta, q_0, Z_0, \varnothing \rangle$ опустошением магазина. Тогда можно построить такой МП-автомат M', что $L(M') = L$.*

Для МП-автомата, допускающего язык опустошением магазина, множество заключительных состояний обычно берется пустым.

Теорема 2.1.2. *Классы языков, допускаемых МП-автоматами по заключительному состоянию и опустошением магазина, совпадают.*

Пример 2.1.4. Построим МП-автомат, допускающий язык $L = \{ a^n b^n \mid n \geq 1 \}$ опустошением магазина.

В примере 2.1.1 для этого языка мы построили МП-автомат M, допускающий язык по заключительному состоянию, но он никогда не опустошает свой магазин, поэтому $L = L_\lambda(M) = \varnothing$. Однако небольшое

изменение позволит автомату M допускать язык $L = \{ a^n b^n \mid n \geq 1 \}$ опустошением магазина. Вместо перехода $\delta(q_2, \lambda, Z_0) = (q_3, Z_0)$ будем рассматривать переход $\delta(q_2, \lambda, Z_0) = (q_2, \lambda)$.

Теперь M выталкивает последний символ Z_0 из магазина, когда допускает, поэтому $L_\lambda(M) = L$.

Формально МП-автомат можно задать как $M = \langle \{q_0, q_1, q_2\}, \{a, b\}, \{Z_0, 0\}, \delta, q_0, Z_0, \varnothing \rangle$, где функция переходов δ определяется таблицей 4.

Таблица 4

Таблица значений функции переходов для примера 2.1.4

Q	$A \cup \{\lambda\}$	Γ	δ
q_0	a	Z_0	$(q_1, 0Z_0)$
q_1	a	0	$(q_1, 00)$
q_1	b	0	(q_2, λ)
q_2	b	0	(q_2, λ)
q_2	λ	Z_0	(q_2, λ)

Пример 2.1.5. Построим МП-автомат, допускающий язык $L = \{ w \mid w \in \{a, b\}^*, |w|_a = |w|_b \}$ как по заключительному состоянию, так и опустошением магазина.

Получив на вход слово $w \in \{a, b\}^*$, автомат будет выявлять в нем фрагменты ab и ba. Для этого автомат будет стирать верхний символ в магазине, если он не совпадает с очередным входным символом, и записывать входной символ в магазин в противном случае.

После прочтения входного слова w в магазине останется единственный символ Z_0 тогда и только тогда, когда $w \in L$. Автомат удалит Z_0 и опустошит таким образом магазин.

Рассмотрим $M_1 = \langle \{q_0\}, \{a, b\}, \{Z_0, a, b\}, \delta, q_0, Z_0, \varnothing \rangle$, где функция переходов δ определяется таблицей 5.

Проанализируем последовательность тактов работы МП-автомата M_1 для входного слова $baaabb$:

$(q_0, baaabb, Z_0) \vdash (q_0, aaabb, bZ_0) \vdash (q_0, aabb, Z_0) \vdash (q_0, abb, aZ_0) \vdash$
$\vdash (q_0, bb, aaZ_0) \vdash (q_0, b, aZ_0) \vdash (q_0, \lambda, Z_0) \vdash (q_0, \lambda, \lambda)$.

Таблица 5

Таблица значений функции переходов для M_1 из примера 2.1.5

Q	$A \cup \{\lambda\}$	Γ	δ
q_0	a	Z_0	(q_0, aZ_0)
q_0	b	Z_0	(q_0, bZ_0)
q_0	λ	Z_0	(q_0, λ)
q_0	a	a	(q_0, aa)
q_0	a	b	(q_0, λ)
q_0	b	a	(q_0, λ)
q_0	b	b	(q_0, bb)

Так как $(q_0, baaabb, Z_0) \vdash^* (q_0, \lambda, \lambda)$, слово $baaabb \in L_\lambda(M_1)$.

Заметим, что $(q_0, \alpha\beta, Z_0) \vdash^* (q_0, \beta, a^n Z_0)$, если в α символов a на n больше, чем b, $(q_0, \alpha\beta, Z_0) \vdash^* (q_0, \beta, b^n Z_0)$, если в α символов b на n больше, чем a, и $(q_0, \alpha\beta, Z_0) \vdash^* (q_0, \beta, Z_0)$, если в α символов a и b одинаковое количество. Таким образом, $L_\lambda(M_1) = L$, в то время как $L(M_1) = \{a, b\}^*$.

Таблица 6

Таблица значений функции переходов для M_2 из примера 2.1.5

Q	$A \cup \{\lambda\}$	Γ	δ
q_0	a	Z_0	(q_1, aZ_0)
q_0	b	Z_0	(q_1, bZ_0)
q_1	a	a	(q_1, aa)
q_1	a	b	(q_1, λ)
q_1	b	a	(q_1, λ)
q_1	b	b	(q_1, bb)
q_1	λ	Z_0	(q_0, Z_0)

Рассмотрим теперь МП-автомат M_2, допускающий язык L по заключительному состоянию.

Пусть $M_2 = \langle \{q_0, q_1\}, \{a,b\}, \{Z_0, a, b\}, \delta, q_0, Z_0, \{q_0\} \rangle$, где функция переходов δ определяется таблицей 6.

Проанализируем последовательность тактов работы МП-автомата M_2 для входного слова $baaabb$:
$(q_0, baaabb, Z_0) \vdash (q_1, aaabb, bZ_0) \vdash (q_1, aabb, Z_0) \vdash (q_1, abb, aZ_0) \vdash$
$\vdash (q_1, bb, aaZ_0) \vdash (q_1, b, aZ_0) \vdash (q_1, \lambda, Z_0) \vdash (q_0, \lambda, Z_0)$. Так как $(q_0, baaabb, Z_0) \vdash^* (q_0, \lambda, Z_0)$, где $q_0 \in F$, слово $baaabb \in L(M_2)$. Здесь имеем, $L(M_2) = L$ и $L_\lambda(M_2) = \varnothing$.

Если функцию переходов δ автомата M_2 дополнить значением $\delta(q_0, \lambda, Z_0) = (q_0, \lambda)$, то получим автомат M_3, для которого $L(M_3) = L_\lambda(M_3) = L$.

Задачи и упражнения

В задачах 2.1.1 – 2.1.7 построить МП-автомат, допускающий заданный язык как по заключительному состоянию, так и опустошением магазина.

2.1.1. $L = \{ a^n b^{2n} \mid n \geq 1 \}$.

2.1.2. $L = \{ a^{2n} b^n \mid n \geq 1 \}$.

2.1.3. $L = \{ w \mid w \in \{a,b\}^*, |w|_a > |w|_b \}$.

2.1.4. $L = \{ w \mid w \in \{a,b\}^*, |w|_a = 2|w|_b \}$.

2.1.5. $L = \{ 0^i 1^j 0^i \mid i, j \geq 1 \}$.

2.1.6. $L = \{ 0^i 1^j 2^k \mid i = 2j \text{ или } j = 2k \}$.

2.1.7. $L = \{ 0^i 1^j 2^k \mid i \neq j \text{ или } j \neq k \}$.

2.1.8. МП-автомат $M = \langle \{q_0, q_1, q_2, q_3, f\}, \{a, b\}, \{Z_0, 0, 1\}, \delta, q_0, Z_0, \{f\} \rangle$ имеет функцию переходов δ, определенную таблицей 7.

Привести последовательность конфигураций МП-автомата при обработке входного слова α.

а) $\alpha = bab$; б) $\alpha = abb$; в) $\alpha = baaba$.

Указать содержимое магазина после того, как МП-автомат прочитал на входе слово b^7a^4.

Таблица 7

Таблица значений функции переходов для задачи 2.1.8

Q	$A \cup \{\lambda\}$	Γ	δ
q_0	a	Z_0	$(q_1, 00Z_0)$
q_0	b	Z_0	$(q_2, 1Z_0)$
q_0	λ	Z_0	(f, λ)
q_1	a	0	$(q_1, 000)$
q_1	b	0	(q_1, λ)
q_1	λ	Z_0	(q_0, Z_0)
q_2	a	1	(q_3, λ)
q_2	b	1	$(q_2, 11)$
q_2	λ	Z_0	(q_0, Z_0)
q_3	λ	1	(q_2, λ)
q_3	λ	Z_0	$(q_1, 0Z_0)$

2.2. Эквивалентность МП-автоматов и КС-грамматик

Алгоритм 2.2.1. ***Построение МП-автомата по КС-грамматике***.

Пусть $G = \langle V_T, V_N, P, S \rangle$ - КС-грамматика. Построим МП-автомат $M_G = \langle \{q\}, V_T, V_T \cup V_N, \delta, q, S, \varnothing \rangle$, который допускает язык опустошением магазина.

Функция переходов δ для M_G определяется следующим образом:

1) $\delta(q, \lambda, A) = \{(q, \alpha) \mid A \to \alpha \in P\}$ для всех $A \in V_N$.

2) $\delta(q, a, a) = \{(q, \lambda)\}$ для всех $a \in V_T$.

Лемма 2.2.1. *Алгоритм 2.2.1 по КС-грамматике* $G = \langle V_T, V_N, P, S \rangle$ *строит МП-автомат* M_G *такой, что* $L_\lambda(M_G) = L(G)$.

Пример 2.2.1. Пусть $G = \langle \{a,+,*,(,)\}, \{E,T,F\}, P, E \rangle$ - КС-грамматика, для которой множество правил P имеет следующий вид:

$E \to E+T$
$E \to T$
$T \to T*F$
$T \to F$
$F \to (E)$
$F \to a$.

Преобразуем грамматику G в МП-автомат M_G, для которого $L_\lambda(M_G) = L(G)$. Пусть $M_G = \langle \{q\}, A, \Gamma, \delta, q, E, \varnothing \rangle$, где $A = \{a,+,*,(,)\}$ и δ определяется таблицей 8.

Таблица 8

Таблица значений функции переходов для примера 2.2.1

Q	$A \cup \{\lambda\}$	Γ	δ
q	λ	E	$\{(q, E+T), (q, T)\}$
q	λ	T	$\{(q, T*F), (q, F)\}$
q	λ	F	$\{(q, (E)), (q, a)\}$
q	a	a	$\{(q, \lambda)\}$
q	+	+	$\{(q, \lambda)\}$
q	*	*	$\{(q, \lambda)\}$
q	(	(	$\{(q, \lambda)\}$
q	)	)	$\{(q, \lambda)\}$

При входе $a*(a+a)$ для M_G возможна следующая последовательность конфигураций:

$(q, a*(a+a), E) \vdash (q, a*(a+a), T) \vdash (q, a*(a+a), T*F) \vdash$
$\vdash (q, a*(a+a), F*F) \vdash (q, a*(a+a), a*F) \vdash (q, *(a+a), *F) \vdash$
$\vdash (q, (a+a), F) \vdash (q, (a+a), (E)) \vdash (q, a+a), E+T)) \vdash$
$\vdash (q, a+a), T+T)) \vdash (q, a+a), F+T)) \vdash (q, a+a), a+T)) \vdash$
$\vdash (q, +a), +T)) \vdash (q, a), T)) \vdash (q, a), F)) \vdash (q, a), a)) \vdash$
$\vdash (q,),)) \vdash (q, \lambda, \lambda)$.

Нетрудно заметить, что рассмотренная последовательность конфигураций соответствует левому выводу слова $a*(a+a)$ в КС-грамматике G.

$$E \Rightarrow T \Rightarrow T*F \Rightarrow F*F \Rightarrow a*F \Rightarrow a*(E) \Rightarrow a*(E+T) \Rightarrow a*(T+T) \Rightarrow$$
$$\Rightarrow a*(F+T) \Rightarrow a*(a+T) \Rightarrow a*(a+F) \Rightarrow a*(a+a) \in L(G).$$

Алгоритм 2.2.2. ***Построение КС-грамматики по МП-автомату***.

Пусть $M = \langle Q, A, \Gamma, \delta, q_0, Z_0, F \rangle$ - МП-автомат, построим по нему КС-грамматику $G = \langle V_T, V_N, P, S \rangle$, язык которой совпадает с языком, допускаемым M опустошением магазина.

Положим $V_T = A$, $V_N = \{ [qXr] \mid q, r \in Q, X \in \Gamma \} \cup \{ S \}$.

Правила множества P строятся следующим образом:

(1) $S \to [q_0 Z_0 q]$ для каждого $q \in Q$;

(2) если $\delta(q, a, X)$ содержит пару $(r, Y_1 Y_2 ... Y_k)$ $(k \geq 1)$, добавим к P все правила вида $[q\, X\, s_k] \to a\, [r\, Y_1\, s_1][s_1\, Y_2\, s_2]...[s_{k-1}\, Y_k\, s_k]$ для каждой последовательности $s_1, s_2, ..., s_k$ состояний из Q;

(3) если $\delta(q, a, X)$ содержит пару (r, λ), добавим к P правило вида $[q\, X\, r] \to a$.

Индукцией по m и n можно показать, что для любых $q, r \in Q$ и $X \in \Gamma$ $[q\, X\, r] \Rightarrow^m w$ тогда и только тогда, когда $(q, w, X) \vdash^n (r, \lambda, \lambda)$.

Лемма 2.2.2. *Алгоритм 2.2.2 по МП-автомату* $M = \langle Q, A, \Gamma, \delta, q_0, Z_0, F \rangle$ *строит КС-грамматику* G *такую, что* $L(G) = L_\lambda(M)$.

Пример 2.2.2. Построим КС-грамматику по МП-автомату M_3 из примера 2.1.5. Напомним, что он допускает язык $L = \{ w \mid w \in \{a, b\}^*, |w|_a = |w|_b \}$.

$M_3 = \langle \{q_0, q_1\}, \{a, b\}, \{Z_0, a, b\}, \delta, q_0, Z_0, \{q_0\} \rangle$. Заметим, что M_3 имеет два состояния и три магазинных символа, в этом случае процедура построения грамматики достаточно громоздка.

По определению $V_N = \{ [qXr] \mid q, r \in Q, X \in \Gamma \} \cup \{ S \}$, значит $|V_N| = 13$.

Грамматика G имеет следующие правила.

$S \to [q_0 Z_0 q_0] \mid [q_0 Z_0 q_1]$.

Из того, что $\delta(q_0, a, Z_0) = (q_1, aZ_0)$, получаем правила

$$[q_0 Z_0 q_0] \to a[q_1 a q_0][q_0 Z_0 q_0] \mid a[q_1 a q_1][q_1 Z_0 q_0]$$

$$[q_0 Z_0 q_1] \to a[q_1 a q_0][q_0 Z_0 q_1] \mid a[q_1 a q_1][q_1 Z_0 q_1].$$

Аналогично из того, что $\delta(q_0, b, Z_0) = (q_1, bZ_0)$, получаем правила

$$[q_0 Z_0 q_0] \to b[q_1 b q_0][q_0 Z_0 q_0] \mid b[q_1 b q_1][q_1 Z_0 q_0]$$

$$[q_0 Z_0 q_1] \to b[q_1 b q_0][q_0 Z_0 q_1] \mid b[q_1 b q_1][q_1 Z_0 q_1].$$

Из того, что $\delta(q_1, a, a) = (q_1, aa)$, получаем правила

$$[q_1 a\, q_0] \to a[q_1 a\, q_0][q_0 a\, q_0] \mid a[q_1 a\, q_1][q_1 a\, q_0]$$
$$[q_1 a\, q_1] \to a[q_1 a\, q_0][q_0 a\, q_1] \mid a[q_1 a\, q_1][q_1 a\, q_1],$$

и аналогично из того, что $\delta(q_1, b, b) = (q_1, bb)$, имеем

$$[q_1 b\, q_0] \to b[q_1 b\, q_0][q_0 b\, q_0] \mid b[q_1 b\, q_1][q_1 b\, q_0]$$
$$[q_1 b\, q_1] \to b[q_1 b\, q_0][q_0 b\, q_1] \mid b[q_1 b\, q_1][q_1 b\, q_1].$$

Из того, что $\delta(q_1, \lambda, Z_0) = (q_0, Z_0)$, получаем правила $[q_1 Z_0 q_0] \to [q_0 Z_0 q_0]$ и $[q_1 Z_0 q_1] \to [q_0 Z_0 q_1]$.

Из того, что $\delta(q_1, a, b) = (q_1, \lambda)$, $\delta(q_1, b, a) = (q_1, \lambda)$, $\delta(q_0, \lambda, Z_0) = (q_0, \lambda)$ получаем соответственно $[q_1 b\, q_1] \to a$, $[q_1 a\, q_1] \to b$, $[q_0 Z_0\, q_0] \to \lambda$.

Построенная по МП-автомату M_3 грамматика содержит бесполезные символы. С помощью алгоритма 1.2.1 определяем, что нетерминальные символы $[q_1 a\, q_0]$, $[q_1 b\, q_0]$,$[q_0 Z_0 q_1]$, $[q_1 Z_0 q_1]$ - непорождающие, поэтому удалим из грамматики все правила, их содержащие. Получим грамматику с правилами:

$$S \to [q_0 Z_0 q_0]$$

$$[q_0 Z_0 q_0] \to a[q_1 a q_1][q_1 Z_0 q_0] \mid b[q_1 b q_1][q_1 Z_0 q_0] \mid \lambda$$

$$[q_1 Z_0 q_0] \to [q_0 Z_0 q_0]$$

$$[q_1 a\, q_1] \to a[q_1 a\, q_1][q_1 a\, q_1] \mid b$$
$$[q_1 b\, q_1] \to b[q_1 b\, q_1][q_1 b\, q_1] \mid a.$$

Далее, удаляя λ-правила и цепные правила грамматики, преобразуем ее к грамматике в приведенной форме:

$S' \to [q_0 Z_0 q_0] \mid \lambda$

$[q_0 Z_0 q_0] \to a[q_1 a q_1][q_0 Z_0 q_0] \mid b[q_1 b q_1][q_0 Z_0 q_0] \mid a[q_1 a q_1] \mid b[q_1 b q_1]$

$[q_1 a\, q_1] \to a\,[q_1 a\, q_1][q_1 a\, q_1] \mid b$

$[q_1 b\, q_1] \to b\,[q_1 b\, q_1][q_1 b\, q_1] \mid a.$

Заменим нетерминальные символы $[q_0 Z_0 q_0]$, $[q_1 a\, q_1]$ и $[q_1 b\, q_1]$ на простые символы S, A и B соответственно.

Наконец, получаем грамматику $G = \langle \{a, b\}, \{A, B, S, S'\}, P, S' \rangle$, правила P которой имеют вид:

$$S' \to S \mid \lambda$$

$$S \to aAS \mid bBS \mid aA \mid bB$$

$$A \to aAA \mid b$$

$$B \to bBB \mid a.$$

Задачи и упражнения

2.2.1. Построить по грамматике

$$S \to aSb \mid A \mid \lambda$$

$$A \to S \mid bAa \mid ab$$

МП-автомат, допускающий тот же язык опустошением магазина.

2.2.2. Построить по грамматике

$$S \to aAA$$

$$A \to aS \mid bS \mid a$$

МП-автомат, допускающий тот же язык опустошением магазина.

2.2.3. Пусть $L = \left\{0^i 1^j 2^{2(i+j)} \mid i, j \geq 0\right\}$. Построить МП-автомат, допускающий этот язык опустошением магазина. При желании можно сначала построить КС-грамматику, а затем преобразовать ее в МП-автомат.

2.2.4. Пусть $L = \left\{0^i 1^j 2^j 3^i \mid i, j \geq 0\right\}$. Построить МП-автомат, допускающий этот язык опустошением магазина. При желании можно сначала построить КС-грамматику, а затем преобразовать ее в МП-автомат.

2.2.5. Пусть $L = \{0^i 1^i 2^j 3^j \mid i, j \geq 0\}$. Построить МП-автомат, допускающий этот язык опустошением магазина. При желании можно сначала построить КС-грамматику, а затем преобразовать ее в МП-автомат.

2.2.6. Пусть $L = \{0^n 1^m \mid n \leq m \leq 2n\}$. Построить МП-автомат, допускающий этот язык опустошением магазина. При желании можно сначала построить КС-грамматику, а затем преобразовать ее в МП-автомат.

2.2.7. Пусть $M = \langle \{q_0, q_1, q_2\}, \{a, b\}, \{X, Z_0\}, \delta, q_0, Z_0, \{q_2\} \rangle$, где δ определяется таблицей 9. Построить по МП-автомату M КС-грамматику G, для которой $L(G) = L_\lambda(M)$.

Таблица 9

Таблица значений функции переходов для задачи 2.2.3

Q	$A \cup \{\lambda\}$	Γ	δ
q_0	a	Z_0	(q_1, XZ_0)
q_0	a	X	(q_1, XX)
q_1	a	X	(q_0, XX)
q_1	λ	X	(q_2, X)
q_2	b	X	(q_2, λ)

2.3. Детерминированные МП-автоматы

Известно, что для каждой КС-грамматики G можно построить недетерминированный МП-автомат, допускающий язык $L(G)$.

Интуитивно МП-автомат является детерминированным, если в каждой конфигурации у него нет возможности сделать более одного очередного такта. Неоднозначность выбора очередного такта имеет место в следующих случаях. Если $\delta(q, a, Z)$ содержит более одной пары, то МП-автомат безусловно не является детерминированным, поскольку можно выбирать из этих нескольких пар. Однако, если $\delta(q, a, Z)$ всегда одноэлементно, все равно остается возможность выбора между чтением входного символа и совершением λ-переходов.

МП-автомат $M = \langle Q, A, \Gamma, \delta, q_0, Z_0, F \rangle$ называется ***детерминированным*** (сокращенно ***ДМП-автоматом***), если для каждых $q \in Q$ и $Z \in \Gamma$

(1) либо $\delta(q, a, Z)$ содержит не более одного элемента для каждого $a \in A$ и $\delta(q, \lambda, Z) = \varnothing$,

(2) либо $\delta(q, a, Z) = \varnothing$ для всех $a \in A$ и $\delta(q, \lambda, Z)$ содержит не более одного элемента.

Язык L называется ***детерминированным КС-языком***, если он допускается некоторым ДМП-автоматом.

Пример 2.3.1. Построим ДМП-автомат, допускающий язык $L = \left\{ wcw^R \mid w \in \{a, b\}^* \right\}$.

Пусть $M = \langle \{q_0, q_1, q_2\}, \{a, b, c\}, \{Z_0, a, b\}, \delta, q_0, Z_0, \{q_2\} \rangle$, где функция переходов δ определяется таблицей 10.

До тех пор пока M не прочитает символ c, отмечающий середину, он запасает в магазине символы входной цепочки. Когда M достигнет середины слова и прочитает c, он переходит в состояние q_1 и далее сравнивает оставшуюся часть входной цепочки с содержимым магазина.

Построенный МП-автомат является детерминированным, так как в каждой конфигурации у него нет возможности сделать более одного очередного такта.

Таблица 10

Таблица значений функции переходов для примера 2.3.1

Q	$A \cup \{\lambda\}$	Γ	δ
q_0	a	Z_0	(q_0, aZ_0)
q_0	b	Z_0	(q_0, bZ_0)
q_0	a	a	(q_0, aa)
q_0	a	b	(q_0, ab)
q_0	b	a	(q_0, ba)
q_0	b	b	(q_0, bb)
q_0	c	a	(q_1, a)
q_0	c	b	(q_1, b)
q_1	a	a	(q_1, λ)
q_1	b	b	(q_1, λ)
q_1	λ	Z_0	(q_2, λ)

Определение детерминированного МП-автомата можно естественным образом расширить, чтобы включить в него те расширенные МП-автоматы, которые можно считать детерминированными.

Расширенный МП-автомат $M = \langle Q, A, \Gamma, \delta, q_0, Z_0, F \rangle$ называется ***детерминированным***, если выполнены следующие условия:

(1) $\delta(q, a, \gamma)$ содержит не более одного элемента для всех $q \in Q$, $a \in A \cup \{\lambda\}$ и $\gamma \in \Gamma^*$;

(2) если $\delta(q, a, \alpha) \neq \varnothing$, $\delta(q, a, \beta) \neq \varnothing$ и $\alpha \neq \beta$, то ни одна из цепочек α и β не является суффиксом другой;

(3) если $\delta(q, a, \alpha) \neq \varnothing$ и $\delta(q, \lambda, \beta) \neq \varnothing$, то ни одна из цепочек α и β не является суффиксом другой.

Задачи и упражнения

В задачах 2.3.1 – 2.3.4 показать, что L является детерминированным КС-языком, построив допускающий его ДМП-автомат.

2.3.1. $L = \{0^n 1^m \mid m \leq n\}$.

2.3.2. $L = \{0^n 1^m \mid m \geq n\}$

2.3.3. $L = \{a^n b^m a^m \mid m, n \geq 1\}$.

2.3.4. $L = \{a^n b^m a^n \mid m, n \geq 1\}$.

2.3.5. $L = \{ w \mid w \in \{a, b\}^+, |w|_a = |w|_b \}$.

Глава 3. Синтаксический анализ

3.1. МП-преобразователи. Левые и правые разборы и анализаторы

Важной фазой процесса компиляции является фаза синтаксического анализа, или разбора.

Для некоторой КС-грамматики G слово разобрано, или проанализировано, если известно одно (или, может быть, все) из его деревьев вывода.

Для построения разборов используются преобразователи с магазинной памятью, или МП-преобразователи. Эти преобразователи получаются из автоматов с магазинной памятью, если добавить выходную ленту и разрешить на каждом такте выдавать выходное слово конечной длины. Определим МП-преобразователь более формально.

МП-преобразователем называется система $P = \langle Q, A, \Gamma, \Delta, \delta, q_0, z_0, F \rangle$, где все символы имеют тот же смысл, что в определении МП-автомата, за исключением того, что Δ - конечный выходной алфавит, а δ - отображение множества $Q \times (A \cup \{\lambda\}) \times \Gamma$ в множество конечных подмножеств множества $Q \times \Gamma^* \times \Delta^*$.

Определим ***конфигурацию преобразователя*** P как четверку (q, x, α, y), где q, x и α те же, что у МП-автомата, а y - выходное слово, выданное вплоть до настоящего момента.

Если $\delta(q, a, z)$ содержит (r, α, β), то будем писать $(q, ax, z\gamma, y) \vdash (r, x, \alpha\gamma, y\beta)$ для любых $x \in A^*$, $\gamma \in \Gamma^*$, $y \in \Delta^*$.

Слово y назовем ***выходом*** для x, если $(q_0, x, z_0, \lambda) \vdash^* (q, \lambda, \alpha, y)$ для некоторых $q \in F$ и $\alpha \in \Gamma^*$. Аналогично можно определить выход опустошением магазина.

МП-преобразователь $P = \langle Q, A, \Gamma, \Delta, \delta, q_0, z_0, F \rangle$ называется ***детерминированным*** (***ДМП-преобразователем***), если

(1) для всех $q \in Q$, $a \in A \cup \{\lambda\}$ и $z \in \Gamma$ множество $\delta(q, a, z)$ содержит не более одного элемента,

(2) если $\delta(q, \lambda, z) \neq \varnothing$, то $\delta(q, a, z) = \varnothing$ для всех $a \in A$.

Расширенным МП-преобразователем называется система $P=\langle Q, A, \Gamma, \Delta, \delta, q_0, z_0, F\rangle$, где δ теперь обозначает отображение множества $Q\times(A\cup\{\lambda\})\times\Gamma^*$ в множество конечных подмножеств множества $Q\times\Gamma^*\times\Delta^*$.

Конфигурации определяются так же, как прежде, но обычно подразумевается, что верх магазина расположен справа, и запись $(q, aw, \beta\alpha, x)\vdash(p, w, \beta\gamma, xy)$ означает, что $\delta(q,a,\alpha)$ содержит (p,γ,y).

Расширенный МП-преобразователь называется ***детерминированным***, если

(1) $|\delta(q,a,\alpha)|\le 1$ для любых $q\in Q$, $a\in A$ и $\alpha\in\Gamma^*$,

(2) слова α и β не являются суффиксами одно другого, если $\delta(q,a,\alpha)\neq\varnothing$ и $\delta(q,b,\beta)\neq\varnothing$ для $b=a$ или $b=\lambda$.

Пусть $G=\langle V_T, V_N, P, S\rangle$ - КС-грамматика, правила которой занумерованы: $1,2,\ldots,p$. Пусть $\alpha\in(V_T\cup V_N)^*$. Тогда

1) ***левым разбором*** слова α называется последовательность правил, примененных при левом выводе слова α из S;
2) ***правым разбором*** слова α называется обращение последовательности правил, примененных при правом выводе слова α из S.

Эти разборы можно представить в виде последовательности номеров правил из множества $\{1,2,\ldots,p\}$.

Пример 3.1.1. Рассмотрим грамматику $G=\langle\{a,+,*,(,)\},\{E,T,F\},P,E\rangle$ со следующей перенумерованной последовательностью правил P:

$$(1)\quad E\to E+T,$$
$$(2)\quad E\to T,$$
$$(3)\quad T\to T*F,$$
$$(4)\quad T\to F$$
$$(5)\quad F\to(E)$$
$$(6)\quad F\to a.$$

На рис. 4 приведено дерево вывода для слова $\alpha=a*(a+a)$.

Слово $\alpha=a*(a+a)$ выводится из аксиомы E в соответствии с левым выводом следующим образом:

$$E\Rightarrow T\Rightarrow T*F\Rightarrow F*F\Rightarrow a*F\Rightarrow a*(E)\Rightarrow a*(E+T)\Rightarrow a*(T+T)\Rightarrow$$
$$\Rightarrow a*(F+T)\Rightarrow a*(a+T)\Rightarrow a*(a+F)\Rightarrow a*(a+a).$$

Последовательность номеров применяемых правил при левом выводе образует левый разбор: (2 3 4 6 5 1 2 4 6 4 6).

Рассмотрим процесс правого вывода слова $\alpha = a*(a+a)$:

$$E \Rightarrow T \Rightarrow T*F \Rightarrow T*(E) \Rightarrow T*(E+T) \Rightarrow T*(E+F) \Rightarrow T*(E+a) \Rightarrow T*(T+a) \Rightarrow$$
$$\Rightarrow T*(F+a) \Rightarrow T*(a+a) \Rightarrow F*(a+a) \Rightarrow a*(a+a).$$

Правому выводу для α соответствует последовательность номеров правил

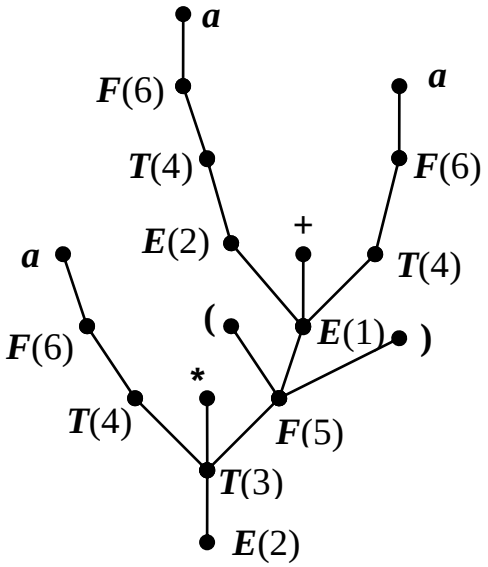

Рис. 4. Дерево вывода для слова $\alpha = a*(a+a)$

(2 3 5 1 4 6 2 4 6 4 6), ее обращение (6 4 6 4 2 6 4 1 5 3 2) – правый разбор.

Среди стратегий разбора наиболее часто используются стратегии нисходящего и восходящего разбора. При использовании стратегии нисходящего разбора строится левый разбор, при стратегии восходящего разбора – правый разбор.

Пусть $\pi_l(w) = i_1 i_2 \ldots i_n$ - левый разбор слова $w \in L$. Левому разбору соответствует дерево вывода. Можно считать, что известны крона и корень дерева вывода и остается восполнить недостающие вершины.

Стратегия левого нисходящего разбора предлагает пытаться заполнять дерево вывода, начиная с корня, и двигаться слева направо, направляясь к кроне.

Пусть $G=\langle V_T, V_N, P, S\rangle$ - КС-грамматика, в которой правила занумерованы от 1 до p. Пусть M_l - недетерминированный МП-преобразователь $\langle\{q\}, A, V_T \cup V_N, \{1,\ldots,p\}, \delta, q, S, \varnothing\rangle$, где δ определяется так:

(1) $\delta(q,\lambda,B)$ содержит (q,α,i), если $B\to\alpha$ - правило из P с номером i;

(2) $\delta(q,a,a)=\{(q,\lambda,\lambda)\}$ для всех $a\in A$.

Назовем M_l ***левым анализатором*** для грамматики G.

Для входа w анализатор M_l моделирует левый вывод в грамматике G слова w из аксиомы S. По правилам (1) M_l каждый раз «развертывает» нетерминал, расположенный наверху магазина, в соответствии с некоторым правилом из P и одновременно выдает номер этого правила. Если наверху магазина находится терминальный символ, то M_l по правилу (2) проверяет, совпадает ли он с текущим входным символом.

Теорема 3.1.1. *Пусть* $G=\langle V_T, V_N, P, S\rangle$. *Тогда* M_l *задает отображение* $w\to\pi_l(w)$ *для любого* $w\in L(G)$.

Пример 3.1.2. Проиллюстрируем построение и работу левого анализатора для грамматики G и слова $\alpha=a*(a+a)$ из примера 3.1.1.

Построим сначала левый анализатор.

$M_l=\langle\{q\},\{a,+,*,(,)\},\{a,+,*,(,),E,T,F\},\{1,2,3,4,5,6\},\delta,q,E,\varnothing\rangle$, где функция переходов δ определяется следующими правилами:

(1) $\delta(q,\lambda,E)=\{(q,E+T,1),(q,T,2)\}$,
$\delta(q,\lambda,T)=\{(q,T*F,3),(q,F,4)\}$,
$\delta(q,\lambda,F)=\{(q,(E),5),(q,a,6)\}$;

(2) $\delta(q,a,a)=\{(q,\lambda,\lambda)\}$,
$\delta(q,+,+)=\{(q,\lambda,\lambda)\}$,
$\delta(q,*,*)=\{(q,\lambda,\lambda)\}$,
$\delta(q,(,()=\{(q,\lambda,\lambda)\}$,
$\delta(q,),))=\{(q,\lambda,\lambda)\}$.

Приведем последовательность конфигураций при работе левого анализатора над входным словом $\alpha=a*(a+a)$:

$$(q,a*(a+a),E,\lambda)\vdash(q,a*(a+a),T,2)\vdash(q,a*(a+a),T*F,23)\vdash$$
$$\vdash(q,a*(a+a),F*F,234)\vdash(q,a*(a+a),a*F,2346)\vdash$$

$$\vdash(q, a*(a+a), a*F, 2346) \vdash(q, *(a+a), *F, 2346) \vdash$$
$$\vdash(q, (a+a), F, 2346) \vdash(q, (a+a), (E), 23465) \vdash(q, a+a), E), 23465) \vdash$$
$$\vdash(q, a+a), E+T), 234651) \vdash(q, a+a), T+T), 2346512) \vdash$$
$$\vdash(q, a+a), F+T), 23465124) \vdash(q, a+a), a+T), 234651246) \vdash$$
$$\vdash(q, +a), +T), 234651246) \vdash(q, a), T), 234651246) \vdash$$
$$\vdash(q, a), F), 2346512464) \vdash(q, a), a), 23465124646) \vdash$$
$$\vdash(q,),), 23465124646) \vdash(q, \lambda, \lambda, 23465124646).$$

Заключительная конфигурация $(q, \lambda, \lambda, 23465124646)$ достигнута опустошением магазина. На выходе левого анализатора получен левый разбор (23465124646).

Левый анализатор представляет собой недетерминированное устройство. Алгоритм, основанный на моделировании работы левого анализатора, для поиска нужной последовательности конфигураций должен перебирать все возможные последовательности, поэтому в общем случае имеет экспоненциальную трудоемкость.

Существует естественный класс грамматик, они называются $LL-$***грамматиками***, для которых левый разбор можно сделать детерминированным с помощью простого приема, состоящего в том, что анализатор заглядывает на входе на несколько символов вперед и делает очередной шаг на основе того, что он при этом видит.

Правый разбор слова – это последовательность правил, с помощью которых можно свернуть слово $w \in L(G)$ к аксиоме S грамматики. Это равносильно процессу «заполнения» дерева вывода, начинающемуся с одной кроны и идущему от листьев к корню. При этом реализуется стратегия восходящего разбора.

Пусть $G = \langle V_T, V_N, P, S \rangle$ - КС-грамматика, в которой правила занумерованы от 1 до p. Обозначим через M_r расширенный МП-преобразователь $\langle \{q\}, A, V_T \cup V_N \cup \{\$\}, \{1, \ldots, p\}, \delta, q, \$, \varnothing \rangle$, причем верх магазина расположен справа, и функция переходов δ определяется так:

(1) $\delta(q, \lambda, \alpha)$ содержит (q, B, i), если $B \to \alpha$ - правило из P с номером i;

(2) $\delta(q, a, \lambda) = \{(q, a, \lambda)\}$ для всех $a \in A$;

(3) $\delta(q, \lambda, \$S) = \{(q, \lambda, \lambda)\}$.

МП-преобразователь M_r назовем ***правым анализатором***.

На такте, соответствующем правилу (2), M_r переносит входной символ в магазин. Всякий раз, когда наверху магазина появляется основа (правая часть правила грамматики), M_r может свернуть ее по правилу (1) и выдать номер правила. M_r работает до тех пор, пока в магазине не останется только начальный символ S с маркером $ на дне магазина. По правилу (3) M_r может перейти тогда в конфигурацию с пустым магазином.

Обозначим через $\pi_r(w) = i_1 i_2 \ldots i_n$ правый разбор слова $w \in L(G)$.

Теорема 3.1.2. *Пусть* $G = \langle V_T, V_N, P, S \rangle$ - *КС-грамматика. Тогда* M_r *задает отображение* $w \to \pi_r(w)$ *для любого* $w \in L(G)$.

Пример 3.1.3. Проиллюстрируем построение и работу правого анализатора для грамматики G и слова $\alpha = a*(a+a)$ из примера 3.1.1.

$M_r = \langle \{q\}, \{a,+,*,(,)\}, \{a,+,*,(,),E,T,F,\$\}, \{1,2,3,4,5,6\}, \delta, q, \$, \varnothing \rangle$, где функция переходов δ определяется следующими правилами:

(1) $\delta(q, \lambda, E+T) = \{(q, E, 1)\}$,
$\delta(q, \lambda, T) = \{(q, E, 2)\}$,
$\delta(q, \lambda, T*F) = \{(q, T, 3)\}$,
$\delta(q, \lambda, F) = \{(q, T, 4)\}$,
$\delta(q, \lambda, (E)) = \{(q, F, 5)\}$,
$\delta(q, \lambda, a) = \{(q, F, 6)\}$;

(2) $\delta(q, a, \lambda) = \{(q, a, \lambda)\}$,
$\delta(q, +, \lambda) = \{(q, +, \lambda)\}$,
$\delta(q, *, \lambda) = \{(q, *, \lambda)\}$,
$\delta(q, (, \lambda) = \{(q, (, \lambda)\}$,
$\delta(q,), \lambda) = \{(q,), \lambda)\}$;

(3) $\delta(q, \lambda, \$E) = \{(q, \lambda, \lambda)\}$.

Приведем последовательность конфигураций при работе правого анализатора над входным словом $\alpha = a*(a+a)$:

$$(q, a*(a+a), \$, \lambda) \vdash (q, *(a+a), \$a, \lambda) \vdash (q, *(a+a), \$F, 6) \vdash$$
$$\vdash (q, *(a+a), \$T, 64) \vdash (q, (a+a), \$T*, 64) \vdash (q, a+a), \$T*(, 64) \vdash$$
$$\vdash (q, +a), \$T*(a, 64) \vdash (q, +a), \$T*(F, 646) \vdash (q, +a), \$T*(T, 6464) \vdash$$

$$\vdash (q,+a),\$T*(E,64642)\vdash (q,a),\$T*(E+,64642)\vdash (q,),\$T*(E+a,64642)\vdash$$
$$\vdash (q,),\$T*(E+F,646426)\vdash (q,),\$T*(E+T,6464264)\vdash$$
$$\vdash (q,),\$T*(E,64642641)\vdash (q,\lambda,\$T*(E),64642641)\vdash$$
$$\vdash (q,\lambda,\$T*F,646426415)\vdash (q,\lambda,\$T,6464264153)\vdash$$
$$\vdash (q,\lambda,\$E,64642641532)\vdash (q,\lambda,\lambda,64642641532).$$

Заключительная конфигурация $(q,\lambda,\lambda,64642641532)$ достигнута опустошением магазина. На выходе правого анализатора получен правый разбор $(6464264153\ 2)$.

Описанный правый анализатор является недетерминированным устройством. Существует важный подкласс КС-грамматик, называемых $LR-$***грамматиками***, для которых соответствующий МП-преобразователь можно сделать детерминированным, позволив ему заглядывать на входе на несколько символов вперед. Таким образом, $LR-$грамматики анализируются снизу вверх (по дереву вывода от кроны к листу) детерминированно.

Задачи и упражнения

3.1.1. Пусть КС-грамматика G с аксиомой S определяется перенумерованными правилами

$(1)\quad S\to AB$

$(2)\quad A\to Aa$

$(3)\quad A\to bB$

$(4)\quad B\to a$

$(5)\quad B\to Sb.$

Построить дерево вывода, левый и правый разборы следующих слов:

а) $baabaab$; б) $baabaa$; в) $baaaaaa$;

г) $bbbbaabbbb$; д) $bbaabaaabab$; е) $bbaabaaaa$.

3.1.2. Для КС-грамматики $G=\langle\{a,+,*,(,)\},\{E,T,F\},P,E\rangle$ с перенумерованной последовательностью правил P:

$(1)\quad E\to E+T$

$(2)\quad E\to T$

$(3)\quad T\to T*F$

$(4)\quad T\to F$

$(5)\quad F \to (E)$

$(6)\quad F \to a$

построить дерево вывода, левый и правый разборы следующих слов:

а) $((a))$; б) $a+(a+a)$; в) $a*(a*a)$;

г) $a+(a*a+a)*a$; д) $a*(a+a*a)+a$; е) $(a+a)*(a*a+a)$.

3.1.3. Пусть КС-грамматика G с аксиомой S определяется перенумерованными правилами

$(1)\quad S \to aBcSdS$

$(2)\quad S \to s$

$(3)\quad B \to BhB$

$(4)\quad B \to BgB$

$(5)\quad B \to b.$

а) Построить МП-преобразователи, реализующие нисходящий и восходящий разборы.

б) Построить последовательность конфигураций левого анализатора для получения левого разбора следующих слов:

$\alpha = abhbcsds$; $\beta = abgbcabhbcsdsds$; $\gamma = abgbcsdabhbcsds$.

в) Построить последовательность конфигураций правого анализатора для получения правого разбора слов α, β, γ.

3.1.4. Пусть КС-грамматика G с аксиомой S определяется перенумерованными правилами

$(1)\quad S \to if\ B\ then\ S\ else\ S$

$(2)\quad S \to s$

$(3)\quad S \to q$

$(4)\quad S \to r$

$(5)\quad B \to B \wedge B$

$(6)\quad B \to B \vee B$

$(7)\quad B \to a$

$(8)\quad B \to b$

$(9)\quad B \to c.$

а) Построить левый и правый анализаторы для грамматики G.

б) Построить последовательность конфигураций левого анализатора для получения левого разбора следующих слов:

$\alpha = if\ \ a \wedge b\ then\ \ s\ else\ q$

$\beta = if\ \ a \vee c\ then\ if\ \ b \wedge c\ \ then\ s\ else\ q\ else\ r$

$\gamma = if\ \ a \vee c\ then\ s\ else\ if\ \ b \wedge c\ \ then\ r\ else\ q$

в) Построить последовательность конфигураций правого анализатора для получения правого разбора слов α, β, γ.

3.1.5. Построить детерминированный левый анализатор для КС-грамматики, определяемой множеством перенумерованных правил

$(1)\quad S \to 0S$

$(2)\quad S \to 1S$

$(3)\quad S \to \lambda.$

Построить последовательность конфигураций левого анализатора для входного слова α:

а) $\alpha = 0010111001$; б) $\alpha = 11100111100010$;

в) $\alpha = 11011011010100$; г) $\alpha = 00110110010011$.

3.1.6. Построить детерминированный левый анализатор для КС-грамматики, определяемой множеством перенумерованных правил

$(1)\quad S \to 0S1$

$(2)\quad S \to A$

$(3)\quad A \to A1$

$(4)\quad A \to \lambda.$

Построить последовательность конфигураций левого анализатора для входного слова α:

а) $\alpha = 00001111$; б) $\alpha = 000011111111$;

в) $\alpha = 11111111111$; г) $\alpha = 011111111$.

3.1.7. Построить детерминированный правый анализатор для КС-грамматики, определяемой множеством перенумерованных правил

$(1)\quad S \to S0$

$(2)\quad S \to S1$

$(3)\quad S \to \lambda.$

Построить последовательность конфигураций правого анализатора для входного слова α:

а) $\alpha = 1001110100011$; б) $\alpha = 11100110010111$;

в) $\alpha = 01101101011100$; г) $\alpha = 000110110011$.

3.1.8. Построить детерминированный правый анализатор для КС-грамматики, определяемой множеством перенумерованных правил

$(1)\quad S \to AB$

$(2)\quad A \to 0A1$

$(3)\quad A \to \lambda$

$(4)\quad B \to B1$

$(5)\quad B \to \lambda.$

Построить последовательность конфигураций правого анализатора для входного слова α:

а) $\alpha = 00001111$; б) $\alpha = 000011111111$;

в) $\alpha = 11111111111$; г) $\alpha = 011111111$.

3.2. Алгоритм Кока-Янгера-Касами

Алгоритм Кока-Янгера-Касами относится к табличным методам синтаксического анализа и применим к КС-грамматике, представленной в нормальной форме Хомского.

КС-грамматика $G = \langle V_T, V_N, P, S \rangle$ называется ***грамматикой в нормальной форме Хомского***, если каждое правило из P имеет один из следующих видов:

1) $A \to BC$, где $A, B, C \in V_N$,

2) $A \to a$, где $a \in V_T$,

3) $S \to \lambda$, если $\lambda \in L(G)$, причем S не встречается в правых частях правил.

Для каждого КС-языка существует КС-грамматика в нормальной форме Хомского. Приведем алгоритм, который по приведенной КС-грамматике строит эквивалентную грамматику в нормальной форме Хомского.

Грамматика G' строится по грамматике $G = \langle V_T, V_N, P, S \rangle$ следующим образом:

1) В P' включается каждое правило из P вида $A \to a$.

2) В P' включается каждое правило из P вида $A \to BC$.

3) В P' включается правило $S \to \lambda$, если оно было в P.

4) Для каждого правила из P вида $A \to X_1 \ldots X_k$, где $k > 2$, в P' включаются правила

$$A \to X_1' \langle X_2 \ldots X_k \rangle,$$

$$\langle X_2 \dots X_k \rangle \to X_2' \langle X_3 \dots X_k \rangle,$$
$$\vdots$$
$$\langle X_{k-2} X_{k-1} X_k \rangle \to X_{k-2}' \langle X_{k-1} X_k \rangle,$$
$$\langle X_{k-1} X_k \rangle \to X_{k-1}' X_k',$$

где $X_i' = X_i$, если $X_i \in V_N$; X_i' - новый нетерминал, если $X_i \in V_T$; $\langle X_i \dots X_k \rangle$ - новый нетерминал.

5) Для каждого правила из P вида $A \to X_1 X_2$, где хотя бы один из символов X_1 и X_2 принадлежит V_T, в P' включается правило $A \to X_1' X_2'$.

6) Для каждого нетерминала вида a', введенного на шагах 4) и 5), в P' включается правило $a' \to a$.

Положим V_N' равным объединению V_N со всеми новыми нетерминалами, введенными при построении P'. Искомой грамматикой будет $G' = \langle V_T, V_N', P', S \rangle$.

Пример 3.2.1. Проиллюстрируем алгоритм построения нормальной формы Хомского на примере приведенной грамматики $G = \langle \{0,1\}, \{S, A, B\}, P, S \rangle$ с множеством правил P:

$$S \to AB \mid \lambda$$
$$A \to 0A1 \mid 01$$
$$B \to B1 \mid 1.$$

1) Включим в P' правило $B \to 1$.

2) Включим в P' правило $S \to AB$.

3) Включим в P' правило $S \to \lambda$.

4) Для правила $A \to 0A1$, содержащего более двух символов в правой части, построим следующие новые правила:

$$A \to 0' \langle A1 \rangle$$
$$\langle A1 \rangle \to A1'.$$

5) а) Для правила $A \to 01$ построим правило $A \to 0'1'$.

б) Для правила $B \to B1$ построим правило $B \to B1'$.

6) Для новых нетерминалов $0'$ и $1'$, соответствующих терминальным символам 0 и 1, построим правила $0' \to 0$ и $1' \to 1$.

Таким образом, новая грамматика $G' = \langle \{0,1\}, V'_N, P', S \rangle$ имеет нетерминальный алфавит $V'_N = \{S, A, B, \langle A1 \rangle, 0', 1'\}$ и множество правил P':

$$S \to AB \mid \lambda$$
$$A \to 0' \langle A1 \rangle \mid 0'1'$$
$$\langle A1 \rangle \to A1'$$
$$B \to B1' \mid 1$$
$$0' \to 0$$
$$1' \to 1.$$

Алгоритм Кока-Янгера-Касами состоит из двух этапов: на первом этапе по КС-грамматике в нормальной форме Хомского и слову в терминальном алфавите строится таблица разбора. По таблице разбора определяется, принадлежит ли анализируемое слово КС-языку, порождаемому заданной КС-грамматикой.

На втором этапе в случае, если слово порождается рассматриваемой КС-грамматикой, строится его левый разбор по таблице разбора.

Алгоритм 3.2.1. ***Построение таблицы разбора***

Пусть $G = \langle V_T, V_N, P, S \rangle$ - КС-грамматика и $w = a_1 a_2 \ldots a_n$ - входное слово в алфавите V_T, которое нужно разобрать согласно грамматике G. Элементы таблицы разбора T будем обозначать t_{ij}, где $1 \le i \le n$ и $1 \le j \le n - i + 1$. Значениями элементов t_{ij} являются подмножества множества V_N.

Нетерминальный символ A принадлежит t_{ij} тогда и только тогда, когда $A \Rightarrow^+ a_i a_{i+1} \ldots a_{i+j-1}$, т.е. когда из A выводятся j входных символов, начиная с позиции i. В частности, входное слово $w \in L(G)$ тогда и только тогда, когда $S \in t_{1n}$.

Таким образом, чтобы выяснить, принадлежит ли w языку $L(G)$, вычислим для w таблицу разбора T и посмотрим, принадлежит ли S ее элементу t_{1n}.

В таблице T индекс i соответствует номеру столбца, а индекс j - номеру строки. Таблица строится путем последовательного заполнения строк, начиная с первой.

(1) На первом шаге для каждого i положим $t_{i1} = \{A \mid A \to a_i \in P\}$. После этого шага из $A \in t_{i1}$ следует, что $A \Rightarrow^+ a_i$.

(2) Пусть уже вычислены $t_{ij'}$ для всех $1 \le i \le n$ и всех $1 \le j' < j$. Положим $t_{ij} = \{A \mid A \to BC \in P,$ где $B \in t_{ik}$ и $C \in t_{i+k,j-k}$ для некоторого $1 \le k < j\}$.

Так как $1 \le k < j$, то k и $j-k$ меньше j. Таким образом, t_{ik} и $t_{i+k,j-k}$ вычисляются раньше, чем t_{ij}. После этого шага из $A \in t_{ij}$ следует

$$A \Rightarrow BC \Rightarrow^+ a_i \ldots a_{i+k-1}C \Rightarrow^+ a_i \ldots a_{i+k-1}a_{i+k} \ldots a_{i+j-1}.$$

(3) Шаг (2) повторяется до тех пор, пока не станут известны t_{ij} для всех $1 \le i \le n$ и $1 \le j \le n-i+1$.

Теорема 3.2.1. *Если алгоритм 3.2.1 применяется к грамматике G в нормальной форме Хомского и входному слову $w = a_1 a_2 \ldots a_n$, то по окончании его работы $A \in t_{ij}$ тогда и только тогда, когда $A \Rightarrow^+ a_i \ldots a_{i+j-1}$.*

Пример 3.2.1. Рассмотрим грамматику $G = \langle \{a,b\}, \{S, A\}, P, S \rangle$ в нормальной форме Хомского с правилами

(1) $S \to AA$,
(2) $S \to AS$,
(3) $S \to b$,
(4) $A \to SA$,
(5) $A \to AS$,
(6) $A \to a$.

Пусть $abaab$ - входное слово. Таблица разбора T, получающаяся в результате работы алгоритма, показана в таблице 11.

Таблица 11

Таблица разбора для слова $abaab$

5	***A, S***			
4	***A, S***	***A,S***		
3	***A,S***	***S***	***A,S***	

	2	***A,S***	***A***	***S***	***A,S***	
$j\uparrow$	1	***A***	***S***	***A***	***A***	***S***
	$i\rightarrow$	1	2	3	4	5

После шага (1) $t_{11}=\{A\}$, так как $A\rightarrow a$ - правило грамматики, и $a_1=a$.

На шаге (2) в t_{32} добавляем S, так как $S\rightarrow AA\in P$ и A принадлежит клеткам t_{31} и t_{41}.

Заметим, что, как видно из рисунка, t_{ij} для $i>1$ можно вычислить, обследовав нетерминалы в следующих парах клеток таблицы разбора:

$(t_{i1},t_{i+1,j-1}),\ (t_{i2},t_{i+2,j-2}),\ \ldots,\ (t_{i,j-1},t_{i+j-1,1})$.

Тогда, если $B\in t_{ik}$ и $C\in t_{i+k,j-k}$ для некоторого $1\le k<j$ и $A\rightarrow BC\in P$, добавляем A к t_{ij}. Это значит, что мы одновременно движемся вверх по i-му столбцу и вниз по диагонали, спускающейся вправо от клетки t_{ij}, обозревая нетерминалы, расположенные в проходимых таким образом парах ячеек.

Так как $S\in t_{15}$, то входное слово $w=abaab$ принадлежит $L(G)$.

Трудоемкость алгоритма построения таблицы разбора

Покажем, что алгоритм 3.2.1, который строит таблицу разбора, можно выполнить за n^3 подходящим образом определенных элементарных операций. ***Элементарной операцией*** будем считать каждую из следующих:

(1) Присваивание переменной значения другой переменной или константы, а также суммы или разности значений двух переменных или констант;

(2) Проверка равенства значений двух переменных;

(3) Обследование и/или изменение значения переменной t_{ij}, если i и j - текущие значения двух целых переменных или констант;

(4) Обследование i-го входного символа a_i, если i - значение некоторой переменной.

Заметим, что операция (3) имеет ограниченный объем, если грамматика заранее известна.

Оценим трудоемкость алгоритма в соответствии с выполняемыми шагами.

(1) ***Вычисление первой строки таблицы*** T**.**

Чтобы вычислить t_{i1} для всех i, нужно для каждого символа a_i входного слова $w = a_1 a_2 \ldots a_n$ найти все правила грамматики с правой частью a_i и записать нетерминалы из левых частей этих правил в t_{i1}.

Так как грамматика G задана, и число правил в ней является константой, для заполнения одной клетки t_{i1} требуется выполнить последовательность элементарных операций, длина которой ограничена некоторой константой. Поэтому заполнение первой строки таблицы разбора T требует $O(n)$ операций.

(2) ***Вычисление*** t_{ij} ***для*** $j > 1$**.**

К моменту вычисления t_{ij} уже вычислены $t_{ij'}$ для всех $1 \le i \le n$ и всех $1 \le j' < j$. Чтобы определить t_{ij}, мы одновременно движемся вверх по *i*-му столбцу и вниз по диагонали, спускающейся вправо от клетки t_{ij}, обозревая нетерминалы, расположенные в соответствующих парах клеток.

Число просматриваемых пар клеток не превосходит $n-1$. Для каждой пары клеток t_{ik} и $t_{i+k,\,j-k}$ рассматриваются всевозможные пары нетерминалов (B,C), где $B \in t_{ik}$ и $C \in t_{i+k,\,j-k}$. Если в грамматике есть правило вида $A \to BC$, нетерминал A заносится в t_{ij}.

Число пар нетерминалов и число правил заданной грамматики G ограничены некоторыми константами, поэтому для выявления по паре клеток t_{ik} и $t_{i+k,\,j-k}$ (при фиксированном k) нетерминалов, входящих в t_{ij}, требуется $O(1)$ операций. Учитывая, что число просматриваемых пар клеток равно $O(n)$, находим, что для вычисления t_{ij} при фиксированных i и j требуется $O(n)$ операций.

(3) ***Вычисление всей таблицы разбора*** T**.**

Число клеток t_{ij} при $j > 1$ в таблице разбора равно $1 + 2 + \ldots + (n-1) = \dfrac{n \cdot (n-1)}{2} = O(n^2)$, поэтому трудоемкость $T_1(n)$ построения всей таблицы разбора равна $O(n) + O(n) \cdot O(n^2) = O(n^3)$.

Алгоритм 3.2.2. ***Нахождение левого разбора по таблице разбора***

Опишем рекурсивную процедуру $analysis(i, j, A)$, порождающую левый разбор для левого вывода $A \Rightarrow^{+} a_i a_{i+1} \ldots a_{i+j-1}$ (очевидно, в этом случае $A \in t_{ij}$).

(1) Если $j = 1$ и $A \to a_i$ - правило из P с номером m, подадим на выход номер m.

(2) Пусть $j > 1$. Просматривая пары клеток $(t_{i1}, t_{i+1,\, j-1})$, $(i_{i2}, t_{i+2,\, j-2})$, …, $(t_{i,\, j-1}, t_{j,1})$, найдем первое k такое, что для некоторых нетерминалов $B \in t_{ik}$ и $C \in t_{i+k,\, j-k}$ в грамматике имеется правило $S \to BC$, к примеру, с номером m. Если найдется несколько правил, удовлетворяющих заданным условиям, возьмем любое из них, например, с наименьшим номером m. Подадим на выход номер m.

(3) Далее выполним $analysis(i, k, B)$, а затем $analysis(i + k, j - k, C)$.

Алгоритм нахождения левого разбора слова $w = a_1 a_2 \ldots a_n$ заключается в выполнении процедуры $analysis(1, n, S)$ при условии, что $S \in t_{1n}$. Если $S \notin t_{1n}$, алгоритм выдает сообщение $w \notin L(G)$.

Трудоемкость алгоритма построения левого разбора по таблице разбора

Покажем, что алгоритм 3.2.2, который строит левый разбор по таблице разбора, можно выполнить за n^2 элементарных операций.

Отметим, что число вызовов процедуры $analysis$ совпадает с числом вершин в дереве вывода входного слова, помеченных нетерминальными символами. Оценим это число.

Пусть x - число всех вершин дерева вывода.

Число листьев в дереве вывода для $w = a_1 a_2 \ldots a_n$ равно n, степень вершины-листа равна 1.

Так как грамматика представлена в нормальной форме Хомского, число вершин, к которым применено правило вида $A \to a$, также равно n; такие вершины имеют степень 2.

Степень корня дерева равна 2.

Число вершин, не являющихся корнем, к которым применено правило вида $A \to AB$, равно $x - 2n - 1$; такие вершины имеют степень 3.

Для любого связного графа (в том числе и для дерева) сумма степеней всех вершин равна удвоенному числу ребер, а число ребер в дереве на 1 меньше числа вершин. Поэтому выполняется следующее равенство:

$$2\cdot(x-1)=n+2n+2+3\cdot(x-2n-1).$$

Из него находим, что $x=3n-1$ и, следовательно, число вершин в дереве вывода, к которым применяются правила грамматики, равно $x-n=2n-1.$

Для получения номера правила на шаге (1) требуется $O(1)$ элементарных операций. Шаг (1) в процессе работы алгоритма выполняется n раз, по одному разу для каждого правила вида $A\to a$ в левом выводе. Следовательно, вклад шага (1) в трудоемкость алгоритма разбора составляет $O(n).$

Рассмотрим трудоемкость шага (2). Для нахождения числа k на этом шаге требуется просмотр не более чем $(n-1)$ пар клеток $(t_{i1},t_{i+1,j-1})$, $(i_{i2},t_{i+2,j-2})$, …, $(t_{i,j-1},t_{j,1})$. Для каждой пары клеток производится поиск правила грамматики вида $A\to BC$, где нетерминал A известен, а нетерминалы B и C находятся из условий $B\in t_{ik}$ и $C\in t_{i+k,j-k}$. Так как число правил заданной грамматики G ограничено константой, поиск правила по паре клеток t_{ik} и $t_{i+k,j-k}$ требует $O(1)$ операций. Учитывая $O(n)$ просматриваемых пар клеток, находим, что для нахождения одного номера правила на шаге (2) достаточно $O(n)$ операций.

Ранее установлено, что число правил вида $A\to BC$ в левом выводе слова длины n равно $2n-1=O(n)$, поэтому вклад шага (2) в трудоемкость алгоритма разбора составляет $O(n)\cdot O(n)=O(n^2)$ операций.

Складывая суммарные трудоемкости шагов (1) и (2), устанавливаем, что трудоемкость $T_2(n)$ построения левого разбора по таблице разбора равна $O(n)+O(n^2)=O(n^2).$

Таким образом, общая трудоемкость алгоритма Кока-Янгера-Касами равна $T(n)=T_1(n)+T_2(n)=O(n^3)$, и справедлива

Теорема 3.2.1. *Пусть G - КС-грамматика в нормальной форме Хомского и $w\in L(G)$. Тогда алгоритм Кока-Янгера-Касами окончит работу, выдав некоторый левый разбор слова w. При этом число элементарных операций, затрачиваемых алгоритмом, равно $T(n)=O(n^3)$, где n - длина слова w.*

Пример 3.2.2. Пусть G - грамматика из примера 3.2.1, и пусть $w=abaab$ - входное слово. Таблица разбора для w приведена в табл. 11.

Так как $S \in t_{15}$, то $w \in L(G)$. Чтобы найти левый разбор слова $w = abaab$, вызовем процедуру $analysis(1,5,S)$. При этом обнаружится, что в паре клеток (t_{11}, t_{24}) содержится нетерминал A и в грамматике G имеется правило $S \to AA$. Будет выдано число 1 (номер этого правила), а затем вызваны $analysis(1,1,A)$ и $analysis(2,4,A)$. Вызов $analysis(1,1,A)$ даст правило номер 6. Так как для $S \in t_{21}$ и $A \in t_{33}$ в грамматике есть правило $A \to SA$, процедура $analysis(2,4,A)$ выдаст номер 4 этого правила и вызовет $analysis(2,1,S)$, а затем $analysis(3,3,A)$. Продолжая этот процесс, получим левый разбор $(1\,6\,4\,3\,5\,6\,2\,6\,3)$.

Задачи и упражнения

3.2.1. Грамматика G определяется следующими правилами:

$S \to AS \mid b$

$A \to SA \mid a$.

а) Построить таблицы разбора следующих слов:

$\alpha = bbaab$; $\beta = ababab$; $\gamma = aabba$.

б) Используя алгоритм Кока-Янгера-Касами, построить левый разбор для тех слов, которые принадлежат языку $L(G)$.

3.2.2. Грамматика G определяется следующими правилами:

$S \to AB \mid AA \mid SB$

$A \to AE \mid CB \mid a$

$B \to AB \mid BA \mid SC \mid b$

$C \to CA \mid c$

$E \to EA \mid a$.

Используя алгоритм построения таблицы разбора, для следующих слов определить, выводимо ли слово в грамматике G:

а) $abcba$; б) $acbba$; в) $caaba$;

г) $accbb$; д) $abcbc$; е) $acbca$.

3.2.3. Грамматика G определяется следующими правилами:

$S \to AB$

$A \to 0A1 \mid \lambda$

$B \to B1 \mid \lambda$.

а) Привести грамматику к нормальной форме Хомского.

б) Используя алгоритм Кока-Янгера-Касами, построить левый разбор для следующих слов:

$\alpha = 000111;$ $\beta = 0001111;$

$\gamma = 111111;$ $\delta = 011111.$

3.2.4. Грамматика $G = \langle \{a,+,*,(,)\}, \{E,T,F\}, P, E \rangle$ определяется следующими правилами:

$E \to E + T \mid T,$

$T \to T * F \mid F,$

$F \to (E) \mid a.$

а) Привести грамматику к нормальной форме Хомского.

б) Используя алгоритм Кока-Янгера-Касами, построить левый разбор для следующих арифметических выражений:

$\alpha = a * a + a;$ $\beta = a * (a + a);$

$\gamma = a * a + a * a;$ $\delta = a * (a + a * a).$

Литература

1. Ахо У., Ульман Дж. ***Теория синтаксического анализа, перевода и компиляции.*** Т.1: Синтаксический анализ. М.: Мир, 1978. − 612 с.
2. Гладкий А.В***. Формальные грамматики и языки.*** М.: Наука, 1973. − 368 с.
3. Пентус А.Е., Пентус М.Р. ***Теория формальных языков***: Учебное пособие. − М.: Изд-во ЦПИ при механико-математическом факультее МГУ, 2004. − 80 с.
4. Хопкрофт Дж.Э., Мотвани Р., Ульман Дж.Д. ***Введение в теорию автоматов, языков и вычислений.*** 2-е изд. М.: Вильямс, 2002. − 528 с.

Оглавление