Дискретная математика. Теория конечных языков и автоматов. Курс лекций

П.Н. Иваньшин

Оглавление

Глава 1

Регулярные языки

Определение 1. Алфавит Σ — *набор символов.* **Строка** *или* **слово** — *последовательность* $a_1a_2a_3a_4\dots a_n$, *где* $a_i \in \Sigma$.

Определение 2. *Обозначим через* Σ^* *множество всех слов алфавита* Σ, *включая пустое слово. Определим бинарную операцию* **конкатенация** $\circ$ *на* Σ^* *следующим образом:*

Пусть $a_1a_2a_3a_4\dots a_n$ *и* $b_1b_2b_3b_4\dots b_m \in \Sigma^*$, *тогда*

$$a_1a_2a_3a_4\dots a_n \circ b_1b_2b_3b_4\dots b_m = a_1a_2a_3a_4\dots a_nb_1b_2b_3b_4\dots b_m.$$

Пусть $S, T \subseteq \Sigma^*$, *тогда* $S \circ T = \{s \circ t | s \in S, t \in T\}$. *Множество* $S \circ T$ *будем также обозначать через* ST.

Определение 3. *Пусть* $B \subseteq \Sigma^*$, *тогда* **замыкание Клини** B^* — *множество всех возможных конкатенаций слов из* B *вместе с пустым словом, т.е.* $B^* = \{w_1w_2\dots w_n | w_i \in B\} \bigcup \{\lambda\}$. *Условимся, что* $\emptyset^* = \{\lambda\}$. *Символ* $*$ *называется* **звезда (звездочка) Клини**.

Определение 4. *Пусть* Σ — *алфавит. Тогда множество* $L \subseteq \Sigma^*$ *называется* **языком**.

Определение 5. *Пусть* Σ — *алфавит. Класс* **регулярных выражений** R *над* Σ *определен по следующим правилам:*

(i) Символ $\emptyset$ — *регулярное выражение и* $\forall a \in \Sigma$, *символ* a — *тоже регулярное выражение.*

(ii) Пусть w_1 *и* w_2 — *регулярные выражения, тогда* w_1w_2, $w_1 \vee w_2$, w_1^* *и* (w_1) — *регулярные выражения.*

(iii) Не существует никаких регулярных выражений, кроме тех, что получены по правилам (i) и (ii).

Определение 6. *Класс* R **регулярных языков** *над алфавитом* Σ *имеет следующие свойства:*

(i) $\emptyset \in \mathrm{R}$, *если* $a \in \Sigma$, *то* $\{a\} \in \mathrm{R}$.

(ii) Если $s_1, s_2 \in \mathrm{R}$, *то* $s_1 \bigcup s_2, s_1 \circ s_2, s_1^* \in \mathrm{R}$.

(iii) Только множества, построенные по правилам (i) и (ii) принадлежат R.

Определение 7. Код C *— подмножество* Σ^*. *Если* C *— подмножество* Σ^* *и каждое слово множества* $S \subset \Sigma^*$ *может быть получено конкатенацией элементов* C, *то говорят, что* C *— код* S. *Код* C **однозначен**, *если каждая строка* S *однозначно представима в виде конкатенации элементов* C.

Определение 8. *Пусть* Σ *— алфавит. Непустой код* $C \subseteq \Sigma$ *—* **префиксный код**, *если для всех слов* $u, v \in C$, *представление* $u = vw$, *где* $w \in \Sigma^*$ *влечет* $u = v$ *и* $w = \lambda$.

Непустой код $C \subseteq \Sigma$ *—* **суффиксный код**, *если для всех слов* $u, v \in C$, *представление* $u = wv$, *где* $w \in \Sigma^*$ *влечет* $u = v$ *и* $w = \lambda$.

Непустой код $C \subseteq \Sigma$ *—* **бипрефиксный код**, *если* C *одновременно и префиксный и суффиксный код.*

Непустой код $C \subseteq \Sigma$ *—* **инфиксный код**, *если* $u, wuv \in C$ *влечет* $w = v = \lambda$.

Код C *—* **блочный код** *если все строки* C *имеют одну и ту же длину.*

Теорема 1. *Если код суффиксный, префиксный, бипрефиксный, инфиксный или блочный, то он однозначен.*

Теорема 2. *Блочный код — и суффиксный, и префиксный, и бипрефиксный, и инфиксный.*

Теорема 3. *Инфиксный код всегда бипрефиксный.*

Задачи

1. Пусть $w = 10110$, найти такие слова v_1, v_2, v_3, v_4, v_5, $v_i \neq v_j$, $i \neq j$, что $v_i w = w v_i$, $i = 1, \ldots, 5$.

2. Найти множества регулярных слов, соответствующих выражениям:

a) $a(b \vee c \vee d)a$;

b) $a^* b^* c$;

c) $(a \vee b)(c \vee d)$;

d) $(ab^*\lambda) \vee (cd)^*$;

e) $a(bc)^*d$.

f) $bc(bc)^*$;

g) $(a \vee b^* \vee \lambda)(c \vee d^*)$.

3. Найти регулярные выражения для множеств слов:

a) $\{ab, ac, ad\}$;

b) $\{ab, ac, bb, bc\}$;

c) $\{a, ab, abb, abbb, abbbb, \ldots\}$;

d) $\{ab, abab, ababab, abababab, ababababab, \ldots\}$;

e) $\{ab, abb, aab, aabb\}$.

4. Пусть $\Sigma = \{a, b, c\}$. Найти регулярные выражения множеств

a) Слов, содержащих ровно две буквы b.

b) Слов, содержащих ровно две буквы b и ровно две буквы c.

c) Слов, содержащих не меньше двух букв b.

d) Слов, начинающихся и заканчивающихся на a и содержащих не менее, чем по одной букве b и c.

e) Слов, содержащих ровно две буквы b и ровно две буквы a.

f) Слов, содержащих четное число букв b.

5. Которые из следующих кодов однозначны?

a) $\{ab, ba, a, b\}$;

b) $\{ab, acb, accb, acccb, \ldots\}$;

c) $\{a, b, c, bd\}$;

d) $\{ab, ba, a\}$;

e) $\{a, ab, ac, ad\}$;.

f) ab^*;

g) $ab^* \vee baaa$;

h) $ab^*c \vee baaac$;

i) $(a \vee b)(b \vee a)$;

j) $(a \vee b \vee \lambda)(b \vee a \vee \lambda)$.

6. Являются ли следующие коды однозначными? Суффиксными?

a) $\{ab, ba\}$;

b) $\{ab, abc, bc\}$;

c) $\{a, b, c, bd\}$;

d) $\{aba, ba, c\}$;

e) $\{ab, acb, accb, acccb\}$.

7. Которые из следующих кодов суффиксные (префиксные)?

a) ab^*;

b) ab^*c;

c) a^*bc^*;

d) $(a \vee b)(b \vee a)$;

e) a^*b.

8. Доказать последние три теоремы первой главы.

Глава 2

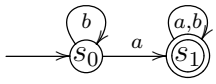
Автоматы

2.1 Детерминированные и недетерминированные автоматы

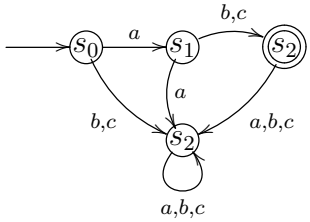
Определение 9. Детерминированный (детерминистский) автомат $(\Sigma, Q, s_0, \Upsilon, F)$, *состоит из конечного алфавита* Σ, *конечного набора* **состояний** Q, *и отображения* $\Upsilon : Q \times \Sigma \to Q$, *называемого* **функцией переходов**, *и множества* F **заключительных (конечных, допускающих) состояний**. *Множество* Q *содержит как элемент* s_0, *так и подмножество* F.

Примеры:

1.

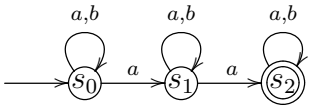


2.



Определение 10. Недетерминированный автомат *есть набор* $(\Sigma, Q, s_0, \Upsilon, F)$ *и состоит из конечного алфавита* Σ, *конечного набора* **состояний** Q, *и отображения* $\Upsilon : Q \times \Sigma \to 2^Q$, *называемого* **функцией переходов**, *и множества* F **заключительных (конечных, допускающих) состояний**. *Множество* Q *содержит как элемент* s_0, *так и подмножество* F. *Здесь множество* 2^Q *— множество всех подмножеств* Q.

Пример:



Определение 11. *Язык* $M(L)$ **принимается автоматом** M *если* $\forall w = w_1 \dots w_n \in M(L)$, $w_1(w_2(\dots(w_n(s_0)))) \in F$, $w_i \in \Sigma$ $(i = 1, \dots, n)$, *и наоборот* $w(s_0) \in F$ *влечет* $w \in M(L)$.

Теорема 4. *Для каждого недетерминированного автомата существует детерминированный автомат, принимающий тот же язык.*

Доказательство. Доказательство конструктивно.

(1) Начинаем с состояния $\{s_0\}$.

(2) $\forall a_i \in \Sigma$ построим a_i-ребро из $\{s_0\}$ во множество всех образов s_0 под действием a_i.

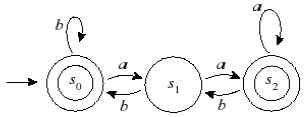
(3) Для каждого построенного множества состояний $\{s_{i_1}, \dots, s_{i_k}\}$ и каждого $a_i \in \Sigma$ снова строим a_i-ребро из первого множества в множество всех состояний, достижимых хотя бы из одного состояния s_{i_j}, $(j = 1, \dots, k)$.

(4) Продолжаем применять шаг (3) до тех пор пока не закончим строить новые состояния—наборы.

(5) Каждое новое состояние, содержащее элемент F — приемное состояние нового автомата. □

Задачи

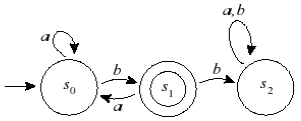
1. Которые из слов принимаются автоматом?



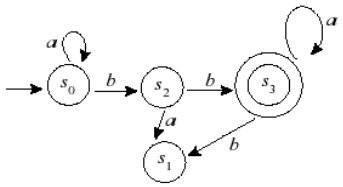
a) abba.
b) aabbb.
c) babab.
d) aaabbb.
e) bbaab.

2. Найти язык, принимаемый автоматом.

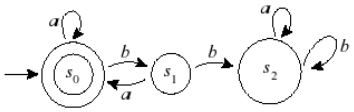
a)



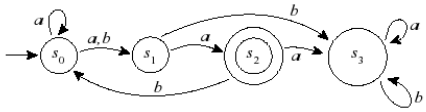
b)



c)



d)



3. Найти детерминированный автомат, принимающий язык

a) $aa^*bb^*cc^*$;

b) $(a^*ba^*ba^*b)^*$;

c) $(a^*(ba)^*bb^*a)^*$;

d) $(a^*b) \vee (b^*a)^*$.

4. Найти недетерминированный автомат, принимающий язык

a) $aa^*bb^*cc^*$;

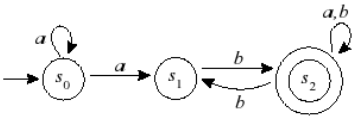
b) $(a^*b) \vee (c^*b) \vee (ac)^*$;

c) $(a \vee b)^*(aa \vee bb)(a \vee b)^*$;

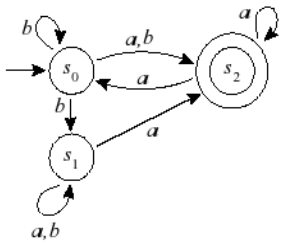
d) $((aa^*b) \vee bb^*a)ac^*$.

5. Построить детерминированный автомат, принимающий тот же язык, что и недетерминированный.

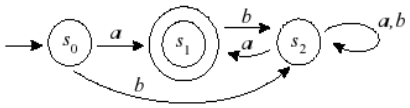
a)



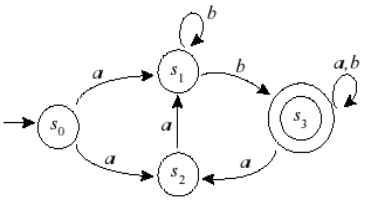
b)



c)



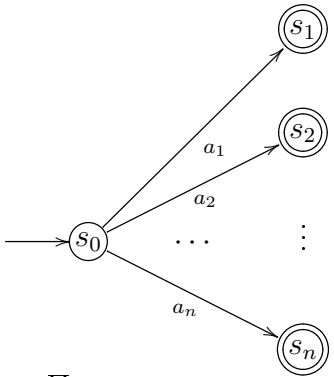
d)



2.2 Теорема Клини

Теорема 5. *Язык L регулярен $\Leftrightarrow$ L — язык, принимаемый автоматом.*

Доказательство. ($\Rightarrow$) Докажем сначала, что для любого конечного подмножества $U = \{a_1, \ldots, a_n\} \subseteq \Sigma$ определен автомат, принимающий U.



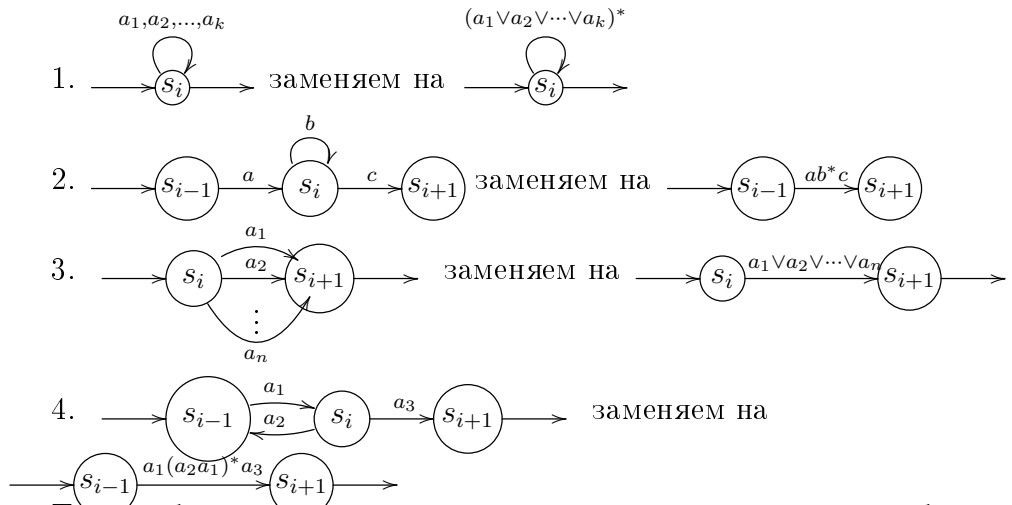
Построим теперь автомат, соответствующий конкатенации языков L_1L_2. Пусть $M_1 = (\Sigma, Q_1, s_0, \Upsilon_1, F_1)$, $M_2 = (\Sigma, Q_2, s_0', \Upsilon_2, F_2)$. Построим $M = (\Sigma, Q, s_0'', \Upsilon, F)$. Положим $Q = Q_1 \bigcup Q_2$, $s_0'' = s_0$, $F = F_2$. Если $a(s_i) = s_j \in \Upsilon_1$ и $s_j \notin F_1$, то $a(s_i) = s_j \in \Upsilon$. Если $a(s_i) = s_j \in \Upsilon_1$ и $s_j \in F_1$, полагаем

$a(s_i) = s_j \in \Upsilon$ и $a(s_i) = s'_0 \in \Upsilon$. Кроме того, $\Upsilon_2 \subset \Upsilon$. Далее если $s_0 \in F_1$, отождествим s'_0 с s_0.

Рассмотрим замыкание Клини L^* языка L. Пусть $M = (\Sigma, Q, s_0, \Upsilon, F)$. Построим $M' = (\Sigma, Q', s'_0, \Upsilon', F')$ для L^*. Положим $Q' = Q \bigcup \{s'_0\}$. Если $a(s_0) = s_j \in \Upsilon$ то $a(s'_0) = s_j \in \Upsilon'$. Также $\Upsilon \subset \Upsilon'$. Если $a(s_0) = s_j \in \Upsilon$, $s_j \in F$, то $a(s_0) = s'_0 \in \Upsilon'$.

Построим еще автомат $L'' = L \bigcup L'$. Пусть $M(L) = (\Sigma, Q, s_0, \Upsilon, F)$, $M(L') = (\Sigma, Q', s'_0, \Upsilon', F')$. Положим $Q'' = Q \bigcup Q' \bigcup \{s''_0\}$, $F'' = F \bigcup F'$. Далее $\Upsilon \bigcup \Upsilon' \subset \Upsilon''$. Кроме того, добавим еще ребра вида: если $a(s_0) = s_j \in \Upsilon$, то $a(s''_0) = s_j \in \Upsilon''$. Аналогично, если $a(s'_0) = s_j \in \Upsilon'$, то $a(s''_0) = s_j \in \Upsilon''$.

($\Leftarrow$) Удаление промежуточных состояний и лишних ребер по правилам:



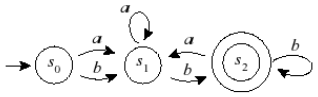
Таким образом, язык, определяемый данным автоматом — объединение регулярных языков, следовательно, регулярен. □

Задачи

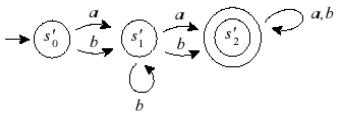
1. Построить автомат для языка $\Sigma^* \setminus M(L)$, если известен автомат M.

2. Построить автомат для языка $L_1 \bigcap L_2$, если известны автоматы M_1 и M_2.

3. Пусть язык L_1 задан автоматом

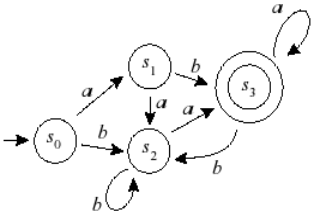


Язык L_2 — автоматом

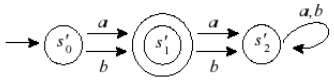


Построить автоматы для языков
a) $L_1 \bigcup L_2$;
b) $L_1 L_2$;
c) L_1^*, L_2^*.

4. Пусть язык L_1 задан автоматом

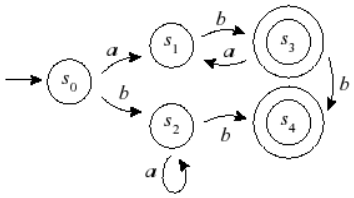


Язык L_2 — автоматом

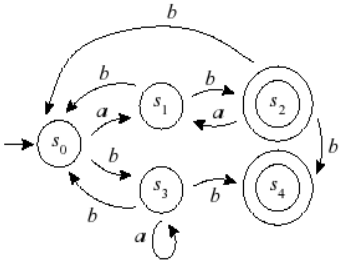


Построить автоматы для языков
a) $L_1 \bigcup L_2$;
b) $L_1 L_2$;
c) L_1^*, L_2^*.

5. Пусть язык L_1 задан автоматом



Язык L_2 — автоматом



Построить автоматы для языков
a) $L_1 \bigcup L_2$;

b) L_1L_2;
c) L_1^*, L_2^*.

2.3 Минимальные детерминированные автоматы и синтаксические моноиды

Определение 12. *Детерминированный автомат M* **минимален** *если число состояний M меньше или равно числу состояний любого другого детерминированного автомата, принимающего тот же язык, что и M.*

Приведем сначала конструкцию, основанную на языке данного автомата.

Определение 13. *Пусть Σ — алфавит, Σ^* — множество всех слов над Σ, $L \subseteq \Sigma^*$. Тогда для языка L определен* **внутренний (минимальный) автомат** *M_i: Состояния M_i — классы эквивалентности $\forall x \in \Sigma^*$ $[x] = \{y \in \Sigma^* | R(y) = R(x)\}$, где $R(x)$ — множество "правых контекстов", принимаемых x под действием L, а именно, $R(x) = \{v \in \Sigma^* | xv \in L\}$. Символ a действует на состояние $[x]$ по правилу $[x]a = [xa]$, где $xa = \Upsilon(x, a)$. При этом стартовое состояние M_i — $[\lambda]$, а приемные состояния $F = \{[x] | x \in L\}$.*

Определение 14. Синтаксический моноид *S языка L состоит из классов эквивалентности, определенных $\forall x \in \Sigma^*$ $[[x]] = \{y \in \Sigma^* | LR(y) = LR(x)\}$, где $LR(x)$ — множество двусторонних контекстов, принимаемых x под действием L. То есть, $LR(x) = \{(u, v) \in \Sigma^* \times \Sigma^* | uxv \in L\}$. Кроме того, на S корректно определена бинарная операция $[[x]][[y]] = [[xy]]$.*

S — моноид, то есть полугруппа с единицей $[[1]] = [[\lambda]]$.

Теперь рассмотрим процедуру получения минимального автомата из данного посредством "схлопывания состояний".

Шаг 1. Для каждой пары состояний $\{p, q\}$ определим, существует ли строка, под действием которой только одно состояние из пары переходит в конечное. Если существует, пометим эту пару как неотождествимую.

Шаг 2. Для каждой пары $\{p, q\}$ из оставшихся непомеченными, и каждого символа $b \in \Sigma$ рассмотрим $\{\Upsilon(p, b), \Upsilon(q, b)\}$. Если $\Upsilon(p, b) \neq \Upsilon(q, b)$ и $\{\Upsilon(p, b), \Upsilon(q, b)\}$ помечена на предыдущем шаге, то и $\{p, q\}$ — тоже помеченная пара.

Повторяем второй шаг до тех пор, пока не переберем все помечиваемые пары.

Получим отношение эквивалентности $p \sim q$, если $\{p, q\}$ — непомеченная пара. Рассмотрим $\Sigma' = \Sigma/\sim$, $F' = F/\sim$, $\Upsilon'([p], a) = [\Upsilon(p, a)]$.

Теорема 6. *Для регулярного языка L, внутренний и минимальный автоматы изоморфны.*

Доказательство. Пусть $M = (\Sigma, Q, s_0, \Upsilon, F)$ — минимальный автомат, полученный методом пометок, а $M_i = (\Sigma, Q_i, [1], \Upsilon_i, F_i)$ — внутренний автомат. Определим $f : Q \to Q_i$ по правилу

$$f([x]) = \{w \in \Sigma^* | \Upsilon(s_0, w) \in [x]\}.$$

Тогда

$$f([x]) = \{w \in \Sigma^* | \Upsilon(s_0, w) = y, y \in [x]\}.$$

Пусть $[x] = [y]$, тогда $\Upsilon(x, u) \in F \Leftrightarrow \Upsilon(y, u) \in F$ для $u, v \in \Sigma^*$. Пусть $f([x]) = [w]$, а $f([y]) = ([w'])$.

Тогда $wu \in L \Leftrightarrow w'u \in L\ (= F_i)$. Следовательно, $[w] = [w']$, и f корректно определена. Пусть наоборот, $f([x]) = f([y])$, тогда $wu \in L \Leftrightarrow w'u \in L\ (= F_i)$, где $\Upsilon(s_0, w) = x$ и $\Upsilon(s_0, w') = y$. То есть, $\Upsilon(x, u) \in F \Leftrightarrow \Upsilon(y, u) \in F$ и $[x] = [y]$. Итак, f корректно определена и взаимнооднозначна.

Покажем наконец, что $f(\Upsilon'([x], a)) = \Upsilon_i(f([x]), a)$,

$$\Upsilon'([x], a) = [\Upsilon(x, a)],$$

и

$$\Upsilon_i(f([x]), a) = f([x])a.$$

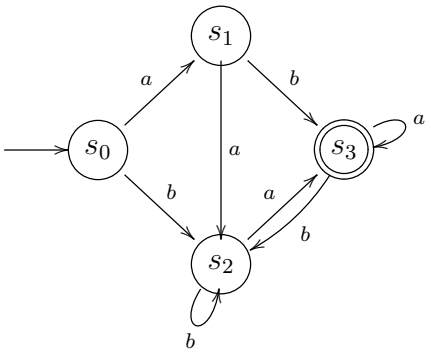
Пусть $w \in f([x])$, тогда $\Upsilon(s_0, w) = x$ для $x \in [x]$. Пусть

$$\Upsilon(x, a) = y \in [\Upsilon(x, a)] = \Upsilon'([x], a)$$

и $[y] = \Upsilon'([x], a)$. Так как $\Upsilon(s_0, wa) = y$, $f([y])_i = [wa]_i = [w]_i a = f([x])a = [\Upsilon_i(f([x]), a)]$ и, следовательно, $f(\Upsilon'([x], a)) = \Upsilon_i(f([x]), a)$. □

Определение 15. Моноид перестановок *детерминированного автомата $M(L) = (\Sigma, Q, s_0, \Upsilon, F)$ — образ гомоморфизма $\phi : \Sigma^* \to T_M$, где T_M — подмоноид всех отображений $Q \to Q$. Положим $\forall a \in \Sigma$, $\phi(a) = \overline{a}$, $\overline{a}(s_i) = s_j$, если $\Upsilon(a, s_i) = s_j$. Далее, $\forall a, b \in \Sigma$, $\overline{ab} = \overline{a}\overline{b}$.*

Пример:



Тогда $\overline{a} = \begin{pmatrix} s_0 & s_1 & s_2 & s_3 \\ s_1 & s_2 & s_3 & s_3 \end{pmatrix}$, $\overline{b} = \begin{pmatrix} s_0 & s_1 & s_2 & s_3 \\ s_2 & s_3 & s_2 & s_3 \end{pmatrix}$.

Теорема 7. *Пусть $M(\Sigma, Q, s_0, \Upsilon, F)$ — минимальный детерминированный автомат, а T_M — моноид перестановок M, тогда T_M конечен.*

Доказательство. Каждый представитель T_M — отображение из Q в Q. Если Q состоит из n элементов, то существует не более n^n различных функций из Q в Q. Следовательно, порядок M не больше n^n. □

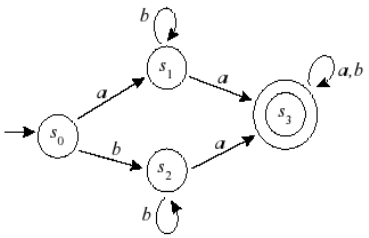
Теорема 8. *Синтаксический моноид регулярного языка L изоморфен моноиду перестановок минимального детерминированного автомата M, принимающего L.*

Доказательство. По определению синтаксического моноида, он представляет собой моноид перестановок внутреннего минимального детерминированного автомата. Так как все минимальные автоматы, принимающие один и тот же язык, изоморфны, синтаксический моноид изоморфен моноиду перестановок минимального автомата. □

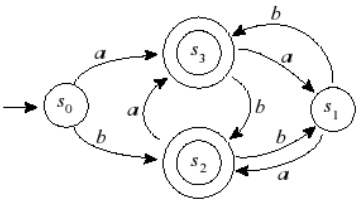
Задачи

1. Найти минимальный автомат, принимающий тот же язык, что и данный.

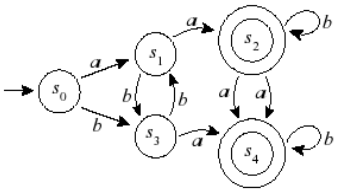
а)



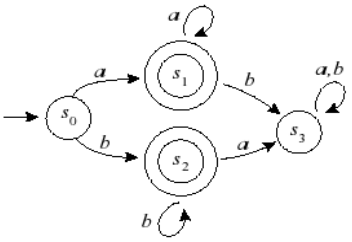
b)



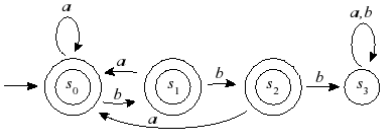
c)



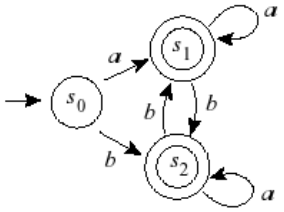
d)



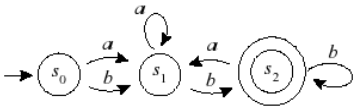
2. Построить минимальный автомат, принимающий язык
a) $aa^*(b \vee c)$;
b) $a(b \vee c)^*bb^*$;
c) $(abc)^*(b \vee c)$;
d) $(a \vee bc)c(ab)^*$.
3. Найти синтаксический моноид языка, принимаемого автоматом
a)



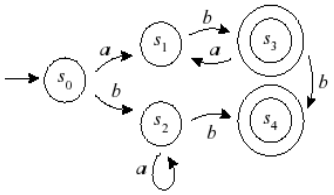
b)



c)



d)



2.4 Лемма о накачке

Лемма 1. *(Лемма о накачке) Пусть L — бесконечный регулярный язык. Тогда существует такая константа n, что для $z \in L$, $|z| > n$ найдутся такие $u, v, w \in \Sigma^*$, $v \neq \lambda$, что $z = uvw$ и $uv^k w \in L$ для всех $k \geq 0$. При этом длина строки uw не больше n. Далее, если M — автомат, принимающий язык L и M имеет q состояний, то $n < q$. Кроме того, можно утверждать, что $z = uvw$, где длина uv не больше q.*

Доказательство. Пусть L принимаем автоматом $M = (\Sigma, Q, s_0, \Upsilon, F)$. Пусть $\Upsilon(s_i, a_i) = s_{i+1}$ для $i = 1, \ldots, t$; обозначим это через

$$(s_1, a_1 a_2 a_3 \cdots a_t) \vdash (s_{t+1}, \lambda).$$

L содержит слово длины m, где $m > q$, пусть $w = a_1 a_2 a_3 \cdots a_m$. Заметим, что если $(s_1, a_1 a_2 a_3 \cdots a_m) \vdash (s_m, \lambda)$, то s_m — принимающее состояние. Так как $m > q$, в процессе чтения w, M дважды проходит одно и то же состояние. Следовательно, $(s_1, a_1 a_2 a_3 \cdots a_{j-1}) \vdash (s_k, \lambda)$ и $(s_1, a_1 a_2 a_3 \cdots a_{k-1}) \vdash (s_k, \lambda)$ для некоторого $j < k$ и одновременно

$$(s_j, a_j a_{j+1} \cdots a_m) \vdash (s_m, \lambda) \text{и} (s_j, a_k a_{k+1} \cdots a_m) \vdash (s_m, \lambda).$$

Следовательно

$$(s_1, a_1 a_2 a_3 \cdots a_m) \vdash (s_m, \lambda) \text{и} (s_1, a_1 a_2 \cdots a_{j-1} a_k a_{k+1} \cdots a_m) \vdash (s_m, \lambda).$$

В то же время, $(s_j, a_j a_{j+1} \cdots a_{k-1}) \vdash s_j$, то есть существует $a_j a_{j+1} \cdots a_{k-1}$-петля в M и, следовательно,

$$(s_1, a_1 a_2 \cdots a_{j-1} (a_j a_{j+1} \cdots a_{k-1})^n a_k a_{k+1} \cdots a_m) \vdash (s_m, \lambda).$$

Положим $u = a_1 a_2 \cdots a_{j-1}$, $v = a_j a_{j+1} \cdots a_{k-1}$, и $w = a_k a_{k+1} \cdots a_m$. Тогда $uv^n w \in L$ для всех $n \in \mathbb{N}$.

Так как $|uw| < |uvw| = m$, если $|uw| > q$, мы можем повторять этот процесс для uv пока $u(v)^n w \in L$ для всех $n \in \mathbb{N}$, где $|uw| < q$. Пусть v — первый цикл в слове z. Тогда длина uv не больше q. □

Теорема 9. *Язык $L = \{a^n b^n | N \geq 1\}$ нерегулярен.*

Доказательство. Пусть $L = \{a^n b^n | N \geq 1\}$ регулярен. Так как L бесконечен, существуют такие строки $u, v, w \in \Sigma^*$, $v \neq \lambda$, что $u(v)^* w \subseteq L$. Рассмотрим все возможные частные случаи.

1) Пусть $u = a^{m-k}$, $v = a^k$, $w = b^m$. Тогда $a^{m-k} a^{2k} b^m = a^{m+k} b^m \in L$, что противоречит определению L.

2) Пусть $u = a^m$, $v = b^k$, $w = b^{m-k}$. Аналогично 1) получим противоречие определению L.

3) Пусть $u = a^{m-k}$, $v = a^k b^r$, $w = b^{m-r}$. Тогда $a^{m-k} a^k b^r a^k b^r b^{m-r} \in L$, опять противоречие определению L.

Следовательно, L нерегулярен. □

Задачи

1. Выяснить, которые из приведенных множеств регулярны.
a) $\{a^n b^{2n} a^n | n \geq 1\}$;
b) $\{(ab)^n | n \geq 1\}$;
c) $\{a^n b^n a^n | n \geq 1\}$;
d) $\{a^n b^m | m, n \geq 1\}$;
e) $\{ww | w \in \Sigma^*$ и $|\Sigma| = 2\}$;
f) $\{a^{2n} | n \geq 1\}$;
g) $\{w \in \{a, b\}^* | w$ содержит одно и то же количество a и b $\}$;
h) $\{w \in \{a, b\}^* | w$ содержит ровно четыре b $\}$;
i) $\{ww^R | w \in \{a, b\}^*$ длина w не больше трех $\}$;
j) $\{ww^R | w \in \{a, b\}^*\}$.

2.5 Алгоритмы сравнения языков и автоматов

Теорема 10. *Существует алгоритм, выявляющий пустоту языка $M(L)$, принимаемого данным автоматом M.*

Доказательство. Пусть M имеет n состояний. Тогда $M(L)$ пуст $\Leftrightarrow$ s_0 не конечное состояние и не существует строки длины не больше n, принимаемой M, поскольку кратчайшая из возможных строк не проходит дважды через одно и то же состояние. Так как таких строк конечное число, их все можно перебрать. □

Теорема 11. *Существует алгоритм, отвечающий на вопрос о совпадении языков для двух различных автоматов.*

Доказательство. Пусть автоматы M_1 и M_2 принимают языки $M_1(L)$ и $M_2(L)$, соответственно. Тогда определены автоматы, принимающие языки $M_1(L) \bigcap M_2(L)$, и $M_1(L) \bigcup M_2(L)$. Следовательно, можно построить автомат для языка $(M_1(L) \bigcap M_2(L)) \bigcup (M_2(L) \bigcap M_1(L))$, — симметрическую разность языков $M_1(L)$ и $M_2(L)$. Это множество пусто $\Leftrightarrow$ $M_1(L) = M_2(L)$. Следовательно, можно использовать предыдущую теорему для ответа на поставленный вопрос. □

Теорема 12. *Существует алгоритм сравнения двух регулярных языков.*

Доказательство. Пусть даны языки L_1 и L_2. Построим такие автоматы M_1 и M_2, что $L_1 = M_1(L)$ и $L_2 = M_2(L)$. Далее применим предыдущую теорему. □

Лемма 2. *Пусть M содержит n состояний. Язык L, соответствующий M бесконечен $\Leftrightarrow$ существует такое слово в L, что его длина больше n но меньше $2n$.*

Доказательство. $\Rightarrow$ Пусть L бесконечен. По лемме о накачке, существует набор $u, v, w \in \Sigma^*$, $uv^m w \in L$ для всех $m \in \mathbb{N}$, кроме того $|uw| \leq n$. Также можно считать, что $|v| < n$. Так как $v \neq \lambda$, существует $m_0 \in \mathbb{N}$, $n < |uv^{m_0}w| = |uw| + m|v| < 2n$. □

Теорема 13. *Существует алгоритм, определяющий конечность языка $M(L)$.*

Доказательство. Пусть M имеет n состояний. Тогда $M(L)$ бесконечен $\Leftrightarrow$ M принимает строку s $n \leq |s| \leq 2n$. Так как таких слов конечное число, простым перебором можно ответить на заданный в формулировке утверждения вопрос. □

Теорема 14. *Существует алгоритм, определяющий конечность языка $M(L)$.*

Доказательство. Как и в предыдущей теореме рассмотрим все слова s длины $n \leq |s| \leq 2n$. Если ни одно такое слово не допустимо, автомат конечен. □

Теорема 15. *Существует алгоритм, позволяющий определить, верно ли что* $L_1 \subseteq L_2$.

Доказательство. Сначала построим автомат для $L_1 \bigcap \Sigma^* \setminus L_2$. Язык $L_1 \bigcap \Sigma^* \setminus L_2$ пуст $\Leftrightarrow L_1 \subseteq L_2$. □

Задачи

1. Доказать, что существует алгоритм проверки тождества $M(L) = \Sigma^*$.
2. Доказать, что существует алгоритм проверки того, содержит ли $M(L)$ слова, включающие данную букву алфавита Σ.
3. Доказать, что существует алгоритм проверки того, содержит ли $M(L)$ слова, включающие все буквы алфавита Σ.
4. Доказать, что существует алгоритм проверки того, содержит ли $M(L)$ слова, начинающиеся с данной буквы алфавита Σ.
5. Доказать, что существует алгоритм проверки того, содержит ли $M(L)$ слова четной длины.
6. Доказать, что существует алгоритм проверки того, содержит ли $M(L)$ слова длины mk, для фиксированного $m \in \mathbb{N}$.

2.6 Автоматы Мили и Мура

Определение 16. Автомат Мура — *набор* $(\Sigma, A, S, s_0, \Upsilon, \phi)$.

S — *конечное множество состояний, содержащее начальное состояние* s_0.

Σ — *алфавит ввода.*

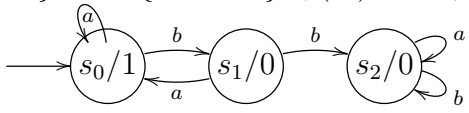
A — *алфавит вывода.*

$\Upsilon : S \times \Sigma \to S$ — *функция перехода.*

$\phi : S \to A$ — *функция вывода.*

При обработке слова $w \in \Sigma^*$ *сначала действует* ϕ, *потом* Υ.

Пример:

$\Sigma = \{a, b\}$, $A = \{0, 1\}$, $S = \{s_0, s_1, s_2\}$, $\phi(s_0) = 1$, $\phi(s_1) = \phi(s_2) = 0$. Υ задана по автомату 

.

Введя слово *aba* на выходе получим 1101.

Определение 17. Автомат Мили — *набор* $(\Sigma, A, S, s_0, \Upsilon, \delta)$.

S — *конечное множество состояний, содержащее начальное состояние* s_0.

Σ — *алфавит ввода.*

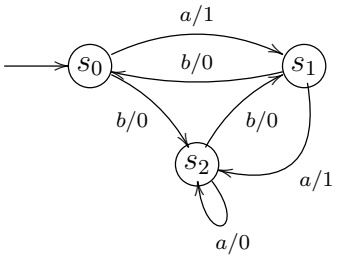
A — *алфавит вывода.*

$\Upsilon : S \times \Sigma \to S$ — *функция перехода.*

$\delta : S \times \Sigma \to A$ — *функция вывода.*

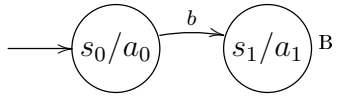
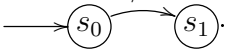
Пример:

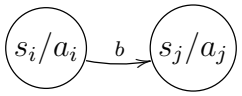
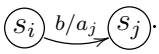
Пусть $S = \{s_0, s_1, s_2\}$, $A = \{0, 1\}$, $\Sigma = \{a, b\}$, δ и Υ определены по автомату



Ввод *aaabb* дает вывод 11000.

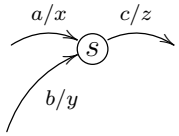
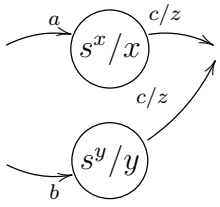
Определение 18. *Строка вывода автомата Мили* **эквивалентна** *строке вывода автомата Мура если она равна подстроке автомата Мура без первого символа. То есть, при выводе автоматом Мура* 010010101*, его эквивалент в автомате Мили —* 10010101.

Построим автомат Мили с выводом, эквивалентным данному автомату Мура. Для этого преобразуем  в  .

Аналогично  в  .

Заметим, что количество состояний при этом не меняется.

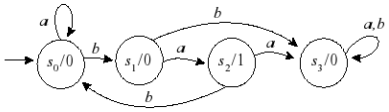
Построение автомата Мура, эквивалентного данному автомату Мили подразумевает, вообще говоря, введение новых состояний.

Например,  перейдет в  .

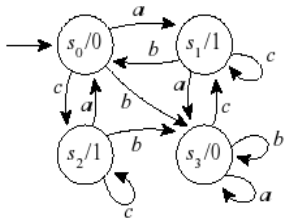
Задачи

1. Для автомата Мура найти эквивалентный автомат Мили.

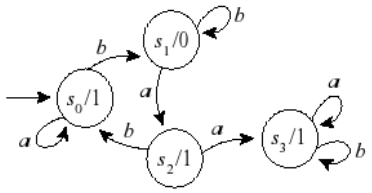
a)



b)

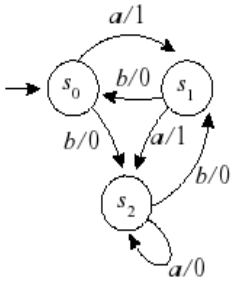


c)

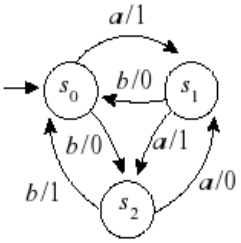


2. Для автомата Мили построить эквивалентный автомат Мура.

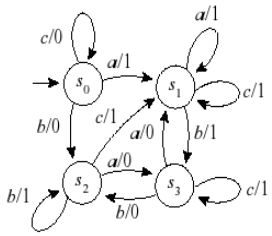
a)



b)



c)



Глава 3

Грамматики

3.1 Формальные грамматики

Определение 19. Формальная грамматика Γ — *набор* (N, Σ, S, P).

* Σ — *набор (алфавит) терминальных символов*

* N — *набор (алфавит) нетерминальных символов*

* P — *набор правил вида: «левая часть»* $\to$ *«правая часть», где:*

∘ *«левая часть» — непустая последовательность терминалов и нетерминалов, содержащая хотя бы один нетерминал*

∘ *«правая часть» — любая последовательность терминалов и нетерминалов*

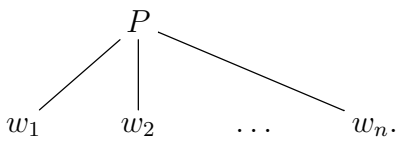
* S — *стартовый (начальный) символ из набора нетерминалов.*

Определение 20. *Пусть* W *и* W' — *элементы* $(N \bigcup \Sigma)^*$ *и* $v \to v'$ — *правило* P, *тогда* $W = uvw$, $W' = uv'w$, *обозначается через* $W \Rightarrow W$. *Пусть*

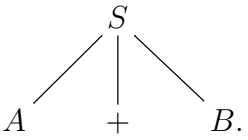
$$W_1 \Rightarrow W_2 \Rightarrow W_3 \Rightarrow \cdots \Rightarrow W_n,$$

для $n \in \mathbb{N}$, *тогда говорят, что* W_n *выводится из* W_1. *Обозначим это через* $W_1 \Rightarrow_n W_n$ *и назовем* **выводом**. *Набор всех строк из элементов* Σ, *порожденных правилами* P, *называется* **языком, порожденным грамматикой** Γ, *и обозначается* $\Gamma(L)$.

Определение 21. *Каждому правилу* $P \to w_1 w_2 \dots w_n$ *соответствует дерево*



Пример 1. $S \to A + B$



Определение 22. *Если деревья, соответствующие правилам вывода некоторой строки, связны, то они образуют дерево с корнем S, называемое* **деревом разбора** *(parse tree, derivation tree). Если в выводе встречается правило $A \to B$ то в дереве разбора есть ребро, соединяющее A с B. Символы A и B называются вершинами, при этом B — потомок A. Терминалы не имеют потомков, следовательно, являются листьями дерева.*

Определение 23. Контекстно-свободная грамматика — *грамматика, в которой все правила P имеют вид $A \to W$, где $A \in N$.*

Контекстно-зависимая грамматика — *грамматика, в которой встречаются правила вида $aAb \to W$, где $A \in N$, $ab \neq \lambda$.*

Определение 24. Регулярная грамматика — *такая контекстно-свободная грамматика $\Gamma = (N, \Sigma, S, P)$, что $\forall p \in P$ $p = n \to w$, где w — либо пустое слово λ, либо строка w содержит не более одного нетерминала, являющегося при этом последним символом w.*

Определение 25. Линейная регулярная грамматика — *такая контекстно-свободная грамматика $\Gamma = (N, \Sigma, S, P)$, что $\forall p \in P$ $p = n \to w$, где w есть либо xY, либо x, либо λ, где $x \in \Sigma$, $Y \in N$.*

Теорема 16. *Язык порожден линейной регулярной грамматикой тогда и только тогда, когда он порожден регулярной грамматикой.*

Доказательство. ($\Rightarrow$) — Очевидно.

($\Leftarrow$): 1. Рассмотрим новые правила вывода для $p = A \to a_1 a_2 \cdots a_n B$, $p_1 = A \to a_1 A_1$, …, $p_n = A \to a_n A_n$. Получим новый язык L'.

2. Ясно, что $L' \subseteq L$. $L \subseteq L'$ по построению.

□

Теорема 17. *Язык регулярен тогда и только тогда, когда он порожден регулярной грамматикой.*

Доказательство. Построим автомат, соответствующий языку, порожденному регулярной грамматикой и наоборот, по автомату построим регулярную грамматику.

Вторая задача решается конструктивно. Пусть допустимое слово — $aabc$, тогда рассмотрим $\Sigma = s_0, s_1, s_2, s_3, s_4$, $N = a, b, c$, $S = s_0$ и правила $s_0 \to as_1$, $s_1 \to as_2$, $s_2 \to bs_3$, $s_3 \to cs_4$, $s_4 \to \lambda$. Тогда правило $s_4 \to \lambda$ применяется только для терминала s_4.

Определение 26. $\Gamma_M = (N, T, S, P)$ — **грамматика, ассоциированная с автоматом** $M = (\Sigma, Q, s_0, T, F)$. *Для нее* $N = Q$, $s_0 = S$. *Правило* $s_i \to as_j$ *принадлежит* P *тогда и только тогда, когда* $F(a, s_i) = s_j$, *а* $s_j \to \lambda$ — *тогда и только тогда, когда* s_j — *конечное состояние.*

Лемма 3. *Язык* L_1, *принимаемый* M, *совпадает с языком* L_2, *порожденным* Γ_M.

Обратная процедура. Пусть дана регулярная грамматика $\Gamma = (N, T, S, P)$ в линейной форме. Определим недетерминированный автомат M_Γ, принимающий эту линейную грамматику. Итак, $M_\Gamma = (\Sigma, Q, s_0, \Upsilon, F)$, где $\Sigma = T\ N$ — множество нетерминалов вместе с дополнительным нетерминалом t, $s_0 = S$. Множество Υ определено следующим образом: $B \in \Upsilon(a, A)$ если $A \to aB \in P$; $t \in \Upsilon(a, A)$, если $A \to a \in P$. Состояние $B \in T$ если $B \to \lambda \in P$ или $B = t$.

Лемма 4. *Язык* L_1, *принимаемый* M_Γ, *совпадает с языком* L_2, *порожденным* Γ.

□

Задачи

1. Найти язык, порожденный грамматикой $\Gamma = (N, T, S, P)$, где
a) $N = \{S, A, B\}$, $T = \{a, b\}$, а множество правил P состоит из

$$S \to AB,\ A \to aA,\ A \to \lambda,\ B \to Bb,\ B \to \lambda.$$

b) $N = \{S, A, B\}$, $T = \{a, b\}$, множество правил P состоит из

$$S \to aB,\ B \to bA,\ A \to aB,\ B \to b.$$

c) $N = \{S, A, B\}$, $T = \{a, b\}$, множество правил P состоит из

$$S \to aA,\ B \to aA,\ S \to bB,\ A \to aB,$$

$$B \to bB,\ A \to bA,\ B \to b,\ A \to a.$$

d)$N = \{S, A, B, C\}$, $T = \{a, b\}$, множество правил P состоит из

$$S \to C,\ A \to aB,\ C \to bC,\ B \to bB,$$

$$C \to aA,\ B \to aA,\ A \to bA,\ B \to \lambda.$$

2. Найти грамматику, порождающую язык

a) ww^r, где w — строка из a и b, а w^r — строка, записанная в обратном порядке. Например, $abba$, $abaaba$, $abbbba \in ww^r$.

b) wcw^r, где $w \in a, b^*$, а w^r — строка, записанная в обратном порядке.

c) $L = \{w | w \in \{a, b\}^*, w = w^r\}$.

d) aa^*bb^*.

e) $(abc)^*$.

f) $(ab)^* \vee (ac)^*$.

g) $ac(bc)^*d$.

h) $(a^*ba^*ba^*b)^*$.

i) $(a^*b) \vee (b^*a)^*$.

j) $(a^*b) \vee (c^*b) \vee (ac)^*$.

3. Построить автомат, принимающий язык, порожденный грамматикой $\Gamma = (N, T, S, P)$.

a) $N = \{S, A, B\}$, $T = \{a, b\}$, P состоит из

$$S \to aB,\ B \to bA,\ A \to aB,\ B \to b.$$

b) $N = \{S, A, B\}$, $T = \{a, b\}$, P состоит из

$$S \to aA,\ B \to aA,\ S \to bB,\ A \to aB,$$

$$B \to bB,\ A \to bA,\ B \to b,\ A \to a.$$

c) $N = \{S, A, B, C\}$, $T = \{a, b\}$, P состоит из

$$S \to C,\ A \to aB,\ C \to bC,\ B \to bB,$$

$$C \to aA,\ B \to aA,\ A \to bA,\ B \to \lambda.$$

d) $N = \{S, A, B, C\}$, $T = \{a, b\}$, P состоит из

$$S \to C,\ C \to b,\ C \to aA,\ A \to aA,$$

$$C \to aC,\ A \to a,\ C \to a,\ A \to \lambda.$$

e) $N = \{S, A, B, C\}$, $T = \{a, b\}$, P состоит из

$$S \to C,\ C \to aaC,\ C \to abC,$$

$$C \to baC,\ C \to bbC,\ C \to \lambda.$$

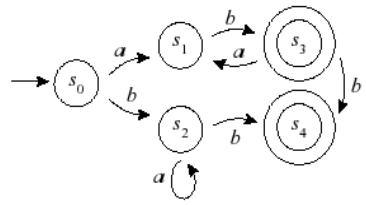
f) $N = \{S, A, B, C\}$, $T = \{a, b\}$, P состоит из

$$S \to C,\ B \to aB,\ C \to bC,\ B \to bB,\ C \to aA,$$

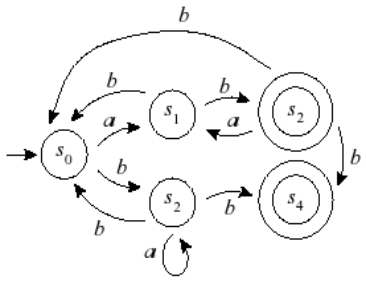
$$B \to a,\ A \to bC,\ B \to b,\ A \to aB.$$

4. Найти грамматику, порождающую язык, принимаемый автоматом

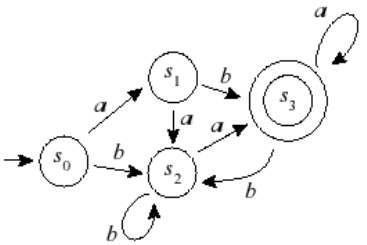
a)



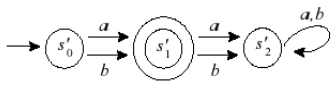
b)



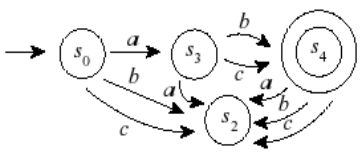
c)



d)



e)



3.2 Нормальные формы Хомского и Грейбаха

Определение 27. *Контекстно-свободная грамматика представлена в* **нормальной форме Хомского** *(Chomsky normal form), если* $\forall p \in P$ *либо* $p = A \to BC$ *либо* $p = A \to a$, *где* $A, B, C \in N$, $a \in T$.

Определение 28. *Контекстно-свободная грамматика представлена в* **нормальной форме Грейбаха** *(Greibach normal form) если* $\forall p \in P\ p = A \to aW$, *где* $a \in T$, W *— строка (может пустая) из нетерминалов.*

Лемма 5. *Пусть* $\Gamma = (N, T, S, P)$ *— грамматика, и* $UV \Rightarrow^* W$ $(U, V, W \in (N \bigcup T)^*)$ *— вывод длины* n. *Тогда строка* W *может быть представлена в виде* W_1W_2, *где* $U \Rightarrow^* W_1$, $V \Rightarrow^* W_2$ *— выводы в* Γ, *не длиннее* n *шагов.*

Доказательство. Доказательство по индукции. Индукция по количеству применений правил P. □

Определение 29. Левый вывод *слова* w *в языке, порожденном* Γ *— вывод, порожденный заменой крайнего левого нетерминала правилом из* P *на каждом шаге вывода*

Лемма 6. *Пусть* $w \in L(\Gamma)$, *где* $L(\Gamma)$ *— язык, порожденный грамматикой* $\Gamma = (N, T, S, P)$. *Тогда существует левый вывод* w.

Доказательство. Индукция по числу применений правил выводы из P. $n = 1$, правило имеет вид $S \Rightarrow w$, и, очевидно, является левым. Пусть $n = k > 1$, и $S \Rightarrow w$, пусть $S \Rightarrow UV$, где $U, V \in (N \bigcup T)^*$. По лемме 5 существуют выводы $U \Rightarrow^* w_1$, $V \Rightarrow^* w_2$, где $w = w_1w_2$. Так как оба этих вывода длины не больше k, существуют левые выводы $U \Rightarrow^* w_1$, $V \Rightarrow^* w_2$. Следовательно, $S \Rightarrow UV \Rightarrow^* w_1V \Rightarrow^* w_1w_2$ — левый вывод w. □

Лемма 7. *Пусть* Γ *такая грамматика, что* $L(\Gamma)$ *не содержит пустого слова. Построим грамматику* Γ' *начиная с правил из* Γ *и*

(i) удалив λ*-правила;*

(ii) для каждого правила $p = A \to w \in P$, *где* $w = w_1X_1w_2X_2 \dots w_nX_n$ *и нильпотентов* $X_1, X_2, \dots, X_n$ *в* w, *положим* $P = 2^{\{1,\dots,n\}}$. *Теперь для* $p \in P$ *определим правило* $A \to w^p$, *где* w^p *— строка* w *без элементов* X_i, $i \in p$. X_i *порождают* λ*-правила.*

Тогда $L(\Gamma') = L(\Gamma)$.

Доказательство. Пусть L – язык порожденный $\Gamma = (N, T, S, P)$, и L' — $\Gamma' = (N, T, S, P')$. Язык $L' \subseteq L$ по построению Γ'.

Докажем, что $L \subseteq L'$, пусть $w \in L$. По индукции по количеству шагов в выводе покажем, что если $S \Rightarrow^* w$, с использованием правил P, то $S \Rightarrow^* w$ по правилам P'. Если $n = 1$, то $S \Rightarrow w$ — правило из P', так как $w = \lambda$. Пусть $n = k$ и $S \Rightarrow^* w$ — вывод из k шагов. Пусть $S \Rightarrow A_1A_2A_3 \dots A_m$ — первый вывод, в котором $A_i \in N \bigcup T$. Тогда $S \Rightarrow A_1A_2A_3 \dots A_m \Rightarrow^* w$ — вывод $S \Rightarrow^* w$.

По лемме 5 существуют выводы $A_i \Rightarrow^* w_i$ в Γ, для $i = 1, \dots, m$, где $w = w_1w_2 \dots w_m$, причем в каждом выводе меньше k шагов. По индукции, если $w_i \neq \lambda$, существуют выводы $A_i \Rightarrow^* w_i$ в Γ' для $i = 1, \dots, m$ (заметим, что если A_i — терминал, то $A_i = w_i$ и $A_i \Rightarrow^* w_i$ содержит 0 шагов). Положим $A'_i = A_i$ если $w_i \neq \lambda$, и $A'_i = \lambda$ если $w_i = \lambda$. Тогда $S \Rightarrow A'_1A'_2A'_3 \dots A_m$ — вывод в Γ' и $S \Rightarrow A'_1A'_2A'_3 \dots A_m \Rightarrow^* w_1A'_2A'_3 \dots A'_m \Rightarrow^* w_1w_2A'_3 \dots A'_m \Rightarrow^* \dots \Rightarrow^* w_1w_2 \dots w_m$ — вывод в Γ'. □

Определение 30. Тривиальное правило – *правило вида* $A \to B$, $A, B \in N$.

Лемма 8. *Если* $L(\Gamma)$ *– язык, порожденный грамматикой* $\Gamma = (N, T, S, P)$, *не содержит* λ, *то существует такая грамматика* Γ' *без* λ*-правил и тривиальных правил, что* $L(\Gamma') = L(\Gamma)$.

Доказательство. Предположим сразу, что Γ не содержит λ-правил по Лемме 7. Построим Γ' удалив все тривиальные правила и для $A_1 \to A_2 \to A_3 \to \dots \to A_m \Rightarrow^* B$, где $A_i \to A_{i+1}$ — тривиальные правила и $B \to w$, включим $A_1 \to B$.

По построению $L(\Gamma') \subseteq L(\Gamma)$.

Докажем обратное включение. Пусть $S \Rightarrow^* w$ — вывод в Γ. По индукции по количеству применения тривиальных правил в выводе покажем, что существует вывод $S \Rightarrow^* w$ в Γ'. Очевидно, что при отсутствии тривиальных правил этот вывод может быть осуществлен и в Γ'. Пусть в выводе использовано k тривиальных правил. Допустим, что вывод w — левый. Пусть $S \Rightarrow^* w$ имеет вид

$$\begin{aligned} S \Rightarrow^* V_1A_1V_2 \to w_1A_1V_2 \to w_1A_2V_2 \to w_1A_3V_2 \to \dots \to w_1A_mV_2 \\ \Rightarrow^* w_1w'V_2 \\ \Rightarrow w_1w'w_2, \end{aligned}$$

где $A_1 \to A_2 \to A_3 \to \dots \to A_m$ — последняя последовательность тривиальных правил в выводе и $A_m \to w'$. Тогда существуют выводы $V1 \Rightarrow^* w_1$, $V_2 \Rightarrow^* w_2$ в Γ и вывод

$$S \Rightarrow^* V_1A_1V_2 \Rightarrow^* w_1A_1V_2 \Rightarrow w_1w'V_2 \Rightarrow^* w_1w'w_2$$

содержит меньше тривиальных правил, причем все правила — правила $P \bigcup P'$. Следовательно, по гипотезе индукции, определен вывод $S \Rightarrow^* w$ в Γ'. □

Лемма 9. *Если язык $L(\Gamma)$, порожденный грамматикой $\Gamma = (N, T, S, P)$, не содержит λ, то существует такая грамматика $\Gamma' = (N, T, S, P')$, в которой каждое правило имеет вид либо $A \to A_1A_2A_3 \dots A_m$ для $n \geq 2$, где $A, A_1, A_2, A_3 \dots, A_m \in N$, либо $A \to a$, где $A \in N$, $a \in T$, что $L(\Gamma) = L(\Gamma')$.*

Доказательство. Пусть мы уже удалили все λ-правила и тривиальные правила. Следовательно, оставшиеся элементы P имеют вид $A \to A_1A_2A_3 \dots A_m$, где $m \geq 2$ и $A_i \in N \bigcup T$ или $A \to a$, где $A \in N, a \in T$. Если $A_1, A_2, A_3, \dots, A_m \in N$, то правило имеет нужный нам вид. Предположим обратное, тогда для каждого символа $A_i = a_i$, $a_i \in T$, введем новый нетерминал X_{a_i}. Заменим правило $A \to A_1A_2A_3 \dots A_m$ на $A \to A'_1A'_2A'_3 \dots A'_m$, где $A'_i = A_i$, если $A_i \in N$, и $A_i = X_{a_i}$, если $A_i \in T$. Следовательно, строку $V_1a_1V_2a_2V_3a_3 \dots V_na_nV_{n+1}$, где $V_i \in N^*$, и $a_i \in T$, заменяем строкой $V_1X_{a_1}V_2X_{a_2}V_3X_{a_3} \dots V_n$ и добавляем правила $X_{a_i} \to a_i$ для $1 \leq i \leq n$. Пусть $\Gamma' = (N, T, S, P')$ — новая грамматика. Докажем, что $L(\Gamma) = L(\Gamma')$. $L(\Gamma) \subseteq L(\Gamma')$ поскольку

$$A \Rightarrow V_1a_1V_2a_2V_3a_3 \dots V_na_nV_{n+1}$$

в Γ можно заменить на

$$A \Rightarrow V_1X_{a_1}V_2X_{a_2}V_3X_{a_3} \dots V_nX_{a_n}V_{n+1}$$

$$\Rightarrow V_1a_1V_2X_{a_2}V_3X_{a_3} \dots V_nX_{a_n}V_{n+1}$$

$$\Rightarrow V_1a_1V_2a_2V_3X_{a_3} \dots V_nX_{a_n}V_{n+1}$$

$$\dots$$

$$\Rightarrow V_1a_1V_2a_2V_3a_3 \dots V_na_nV_{n+1}$$

в Γ'.

Пусть теперь в выводе $S \Rightarrow^* w_1ww_2$, где $U \Rightarrow^* w_1$, $A \Rightarrow^* w$ и $V_i \Rightarrow v_i$ — правила Γ

$$S \Rightarrow^* UAU \Rightarrow UV_1X_{a_1}V_2X_{a_2} \dots V_nX_{a_n}V_{n+1}V$$

$$\Rightarrow^* w_1v_1a_1v_2a_2v_3a_3 \dots v_na_nv_{n+1}w_2 = w_1ww_2,$$

где

$$A \to V_1X_{a_1}V_2X_{a_2} \dots V_nX_{a_n}V_{n+1}$$

правило Γ', но не Γ, и вывод выглядит следующим образом:

$$S \Rightarrow UAV$$
$$\Rightarrow^* w_1AV$$
$$\Rightarrow w_1V_1X_{a_1}V_2X_{a_2}\dots V_nX_{a_n}V_{n+1}V$$
$$\Rightarrow^* w_1v_1X_{a_1}V_2X_{a_2}\dots V_nX_{a_n}V_{n+1}V$$
$$\Rightarrow w_1v_1a_1V_2X_{a_2}\dots V_nX_{a_n}V_{n+1}V$$
$$\Rightarrow^* w_1v_1a_1v_2X_{a_2}\dots V_nX_{a_n}V_{n+1}V$$
$$\Rightarrow w_1v_1a_1v_2a_2V_3X_{a_3}\dots V_nX_{a_n}V_{n+1}V$$
$$\dots$$
$$\Rightarrow w_1v_1a_1v_2a_2v_3a_3\dots v_na_nV_{n+1}V$$
$$\Rightarrow^* w_1v_1a_1v_2a_2v_3a_3\dots v_na_nv_{n+1}V$$
$$\Rightarrow^* w_1ww_2.$$

Заменим его на

$$S \Rightarrow^* UV_1a_1V_2a_2\dots V_na_nV_{n+1}V$$
$$\Rightarrow^* w_1V_1a_1V_2a_2\dots V_na_nV_{n+1}V$$
$$\Rightarrow^* w_1v_1a_1V_2a_2\dots V_na_nV_{n+1}V$$
$$\dots$$
$$\Rightarrow^* w_1v_1a_1v_2a_2\dots a_nV_{n+1}V$$
$$\Rightarrow^* w_1v_1a_1v_2a_2\dots a_nv_{n+1}V$$
$$\Rightarrow^* w_1v_1a_1v_2a_2\dots a_nv_{n+1}w_2$$
$$\Rightarrow^* w_1ww_2.$$

Таким образом, мы получили вывод $S \Rightarrow^* w_1ww_2$ в Γ. Следовательно, $L(\Gamma') \subseteq L(\Gamma)$. □

Лемма 10. *(Нормальная форма Хомского) Если язык $L(\Gamma)$, порожденный грамматикой $\Gamma = (N, T, S, P)$, не содержит λ, то существует грамматика Γ', в которой каждое правило имеет вид либо*

$$A \to BC,$$

либо

$$A \to a,$$

где $A, B, C \in N$, $a \in T$, и $L(\Gamma) = L(\Gamma')$.

Доказательство. По лемме 9, можем считать, что каждое правило Γ имеет вид либо $A \to A_1A_2A_3 \dots A_m$, $A, A_1, A_2, A_3, \dots, A_m \in N$ либо $A \to a$, $A \in N$, $a \in T$. Построим Γ', заменив правила вида $A \to A_1A_2A_3 \dots A_m$ на множество правил $A \to A_1X_1$, $X_1 \to A_2X_2$, …, $X_{m-2} \to A_{m-1}A_m$, где каждая замена правила использует новое множество символов. Вывод

$$A \Rightarrow A_1X_2 \Rightarrow A_1A_2X_3 \Rightarrow^* A_1A_2A_3 \dots A_m$$

в Γ' влечет $L(\Gamma) \subseteq L(\Gamma')$.

Если вывод $S \Rightarrow^* w$ в Γ' использует правила только из Γ, то $w \in L(\Gamma')$. Пусть используются еще и правила из Γ', и W_m — последняя строка в выводе, содержащая символ не из Γ, то есть $W_m \Rightarrow W_{m+1} \Rightarrow^* w$, и $W_m \Rightarrow W_{m+1}$ имеет вид $UX_{m-2}V \Rightarrow UA_{m-1}A_mV$. Следовательно, вывод использует множество правил $A \to A_1X_1$, $X_1 \to A_2X_2$, …, $X_{m-2} \to A_{m-1}A_m$ и представляется в виде

$$S \Rightarrow^* UAV \Rightarrow^* UA_1X_1V \Rightarrow^*$$

$$\Rightarrow^* UA'_1X_1V \Rightarrow^* UA'_1A'_2X_2V \Rightarrow^*$$

$$\Rightarrow^* UA'_1A'_2X_2V \Rightarrow^* UA'_1A'_2A_3X_3V \Rightarrow^*$$

$$\Rightarrow^* UA'_1A'_2A'_3X_3V \Rightarrow^*$$

$$\dots$$

$$\Rightarrow^* UA'_1 \dots A_{m-2}X_{m-2}V \Rightarrow^* UA'_1 \cdots A'_{m-2}X_{m-2}V \Rightarrow^* w \Rightarrow W_{m+1},$$

где $U' = UA'_1A'_2A'_3 \cdots A'_{m-2}$ и $A_i \Rightarrow^* A'_i$ — вывод Γ. Если вывод $S \Rightarrow^* w$ все еще не вывод Γ, снова рассмотрим последнюю строку вывода, содержащую символ из $\Gamma' \backslash \Gamma$, и продолжим этот процесс до тех пор пока не исключим все подобные подстроки. Таким образом, $w \in \Gamma$ и $L(\Gamma') \subseteq L(\Gamma)$. □

Определение 31. *Пусть теперь $L(\Gamma)$ содержит λ. Добавим к Γ' два нетерминала — S' и ψ, где S' — новый стартовый символ и правила $S' \to S\psi$ и $\psi \to \lambda$. Полученная грамматика называется грамматикой в* **λ-дополненной форме Хомского**.

Теорема 18. *Пусть дана контекстно-свободная грамматика Γ, содержащая λ, тогда существует такая контекстно-свободная грамматика в λ-дополненной форме Хомского Γ', что $L(\Gamma) = L(\Gamma')$.*

Определение 32. Удаление левой рекурсии — *удаление правил вида $A \to Aa$.*

Пусть в грамматике Γ без λ-правил

$$A \to AV_1, A \to AV_2, \dots, A \to AV_n$$

есть множество всех правил, правая часть которых начинается с A.

Пусть

$$A \to U_1, A \to U_2, \ldots, A \to U_m -$$

остальные правила.

Построим грамматику Γ' *добавив новый нетерминал* A' *и осуществив следующее:*

1) Удалим все правила вида $A \to AV_i$ *для* $1 \le i \le n$.

2) Введем новые правила $A \to U_i A'$ *для* $1 \le i \le n$.

3) Введем новые правила $A' \to V_i A'$ *и* $A' \to V_i$.

Лемма 11. $L(\Gamma') = L(\Gamma)$.

Доказательство. Пусть правило, начинающееся с A, имеет вид $A \to U_i A$ для $1 \le i \le n$, и $A \Rightarrow AV_{(1)} \Rightarrow AV_{(2)}V_{(1)} \Rightarrow^* AV_{(k)} \ldots V_{(2)}V_{(1)} \Rightarrow U_{(i)}V_{(k)} \ldots V_{(2)}V_{(1)}$, где $V_{(j)} \in \{V_1, V_2, \ldots, V_n\}$ для все $1 \le j \le k$ и $U_{(i)} \in \{U_1, U_2, \ldots, U_m\}$. следовательно, используя левый вывод, каждый вывод, содержащий A имеет вид

$$wAW \Rightarrow wAV_{(1)}W \Rightarrow wAV_{(2)}V_{(1)}W \Rightarrow^*$$

$$\Rightarrow^* wAV_{(k)} \ldots V_{(2)}V_{(1)}W \Rightarrow wU_{(i)}V_{(k)} \ldots V_{(2)}V_{(1)}W.$$

Но

$$A \Rightarrow AV_{(1)} \Rightarrow AV_{(2)}V_{(1)} \Rightarrow^*$$

$$\Rightarrow^* AV_{(k)} \ldots V_{(2)}V_{(1)} \Rightarrow U_{(i)}V_{(k)} \ldots V_{(2)}V_{(1)}$$

можно заменить на

$$A \Rightarrow U_{(i)}A' \Rightarrow U_{(i)}V_{(k)}A' \Rightarrow U_{(i)}V_{(k)}V(k-1)A' \Rightarrow^*$$

$$\Rightarrow^* U_{(i)}V_{(k)} \ldots V_{(2)}A' \Rightarrow U_{(i)}V_{(k)} \ldots V_{(2)}V_{(1)}.$$

Приписав w слева, а W — справа от каждой строки в выводе, получим $wAW \Rightarrow^* wU_{(i)}V_{(k)} \ldots V_{(2)}V_{(1)}W$ в Γ'. Следовательно, $L(\Gamma) \subseteq L(\Gamma')$.

Доказательство $L(\Gamma') \subseteq L(\Gamma)$ аналогично. □

Лемма 12. *Пусть* $A \to UBV$ *— правило грамматики* Γ *и* $B \to W_1$, $B \to W_2$, ..., $B \to W_m$ *— множество всех правил с* B *слева. Пусть* Γ' *— грамматика без правила* $A \to UBV$, *но с добавленными правилами* $A \to UW_iV$ *для* $1 \le i \le m$, *тогда* $L(\Gamma) = L(\Gamma')$.

Доказательство. Правило $A \to UW_iV$ всегда можно заменить на правило $A \to UBV$, которое предшествует правилу $B \to W_i$. Следовательно, $L(\Gamma') \subseteq L(\Gamma)$.

Обратное включение доказывается аналогично. □

Лемма 13. *(Нормальная форма Грейбаха) Любая контекстно-свободная грамматика, непорождающая λ может быть выражена так, что каждое правило вывода имеет вид*

$$A \to aW,$$

где $a \in T$, а W — строка, которая либо пуста, либо состоит из терминалов и/или нетерминалов.

Доказательство. Пронумеруем все нетерминалы, начиная со стартового символа S. Обозначим все нетерминалы через $A_1, A_2, A_3, \ldots, A_m$. Наша первая цель — изменить каждое правило так, чтобы оно приняло вид

$$1) A \to aW,$$

где $a \in T$, а W — строка, которая либо пуста, либо состоит из терминалов и/или нетерминалов, или

$$2) A_i \to A_j Y,$$

где $i < j$, а Y — строка, которая состоит из терминалов и/или нетерминалов. Напомним, что процедуры удаления левой рекурсии и нетерминалов из лемм 11, 12 не меняют языка грамматики.

По индукции, для $i = 1$, поскольку $S = A_1$ автоматически имеет меньший порядковый номер, чем любой другой нетерминал, нужно рассмотреть только $S \to SY$. Тогда S справа от $\to$ можно удалить по правилу удаления левой рекурсии. Пусть утверждение верно для A_i, $i < k$. Докажем его для $i = k$. В каждом случае $A_k \to A_j Y$ — правило для $k > j$, используем процедуру из леммы 12 для удаления A_j. Если $A_k \to A_k Y$ — правило, воспользуемся правилом из леммы 11 для удаления A_k с правой стороны.

Итак, можем считать, что каждое правило представлено в одном из следующих двух видов:

$$1) A \to aW,$$

где $a \in T$, а W — строка, которая либо пуста, либо состоит из терминалов и/или нетерминалов, или

$$2) A_i \to A_j Y,$$

где $i < j$, а Y — строка, которая состоит из терминалов и/или нетерминалов.

Каждое правило с A_m слева должно иметь вид $A_m \to aW$, поскольку нет нетерминалов с номером, большим m. Если нет правил вида $A_{m-1} \to A_m W'$, воспользуемся процедурой Леммы 12 для удаления A_m. В результате получим правило в виде $A_m \to bW''$. Пусть k — наибольшее

из таких чисел, что $A_k \to A_jY$ — правило, для которого $k < j$. Снова применив Лемму 12 для удаления A_j, получим правило типа $A_k \to aW$. После завершения этого процесса, получим

$$A_i \to aW,$$

где $a \in T$, а W — строка, которая либо пуста, либо состоит из терминалов и/или нетерминалов для каждого i. Рассмотрим теперь нетерминалы B_i, полученные при применении Леммы 11. По построению B_i, не существует правил вида $B_i \to B_jW$. Следовательно, правила с B_i слева от $\to$ имеют тип либо $B_i \to aW$, либо $B_i \to A_jW$. Повторяя уже приведенный выше процесс, получим правила вида $B_i \to aW$. □

Теорема 19. *Каждая контекстно-свободная грамматика* Γ*, язык которой не содержит* λ *может быть представлена в нормальной форме Грейбаха.*

Доказательство. Каждое правило может быть представлено в виде

$$A \to aW,$$

где $a \in T$, а W — строка, которая либо пуста, либо состоит из терминалов и/или нетерминалов. Далее каждый терминал b в W заменим на нетерминал A_b и добавим правила $A_b \to b$. □

Определение 33. *Для каждой контекстно-свободной грамматики, содержащей* λ*, построим грамматику в* **расширенной нормальной форме Грейбаха**, *принимающей пустое слово, просто добавив правило* $S \to \lambda$ *после применения процедур из предыдущей леммы.*

Задачи

1. Пусть $\Gamma = (N, T, S, P)$ — грамматика, для которой $N = \{S, A, B\}$, $T = \{a, b\}$, а P состоит из правил

$$S \to ABABABA,\ A \to Aa,\ A \to \lambda,\ B \to b.$$

a) Найти нормальную форму Хомского Γ.

b) Найти нормальную форму Грейбаха Γ.

2. Пусть $\Gamma = (N, T, S, P)$ — грамматика, для которой $N = \{S, A, B\}$, $T = \{a, b\}$, а P состоит из правил

$$S \to SS,\ B \to aa,\ S \to BS,\ B \to bb,\ S \to SB,$$

$$A \to ab,\ S \to \lambda,\ A \to ba,\ S \to ASA.$$

a) Найти нормальную форму Хомского Г.

b) Найти нормальную форму Грейбаха Г.

3. Пусть $\Gamma = (N, T, S, P)$ — грамматика, для которой $N = \{S, A, B\}$, $T = \{a, b\}$, а P состоит из правил

$$S \to AbaB,\ A \to bAa,\ A \to \lambda,\ B \to AAb,\ B \to aabA.$$

a) Найти нормальную форму Хомского Г.

b) Найти нормальную форму Грейбаха Г.

3.3 Автоматы с магазинной памятью

Определение 34. Автомат с магазинной памятью (PDA-автомат) *— набор*

$$M = (\Sigma, Q, s, I, \Upsilon, F),$$

где

Σ *— конечный алфавит,*

Q *— конечное множество состояний,*

s *— начальное или стартовое состояние,*

I *— алфавит памяти (магазина),*

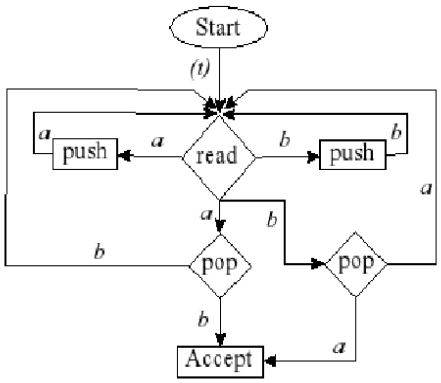
Υ *— функция перехода* $\Upsilon \subseteq ((\Sigma \bigcup \{\lambda\} \times Q \times I \bigcup \{\lambda\}) \times (Q \times I \bigcup \{\lambda\}))$,

F *— множество приемных состояний.*

То есть Υ прочитывает букву $\Sigma \bigcup \{\lambda\}$, определяет состояние, прочитывает букву $I \bigcup \{\lambda\}$. Затем меняет или нет состояние, и выводит букву из $I \bigcup \{\lambda\}$. Аналогично случаю обычного автомата, буква слова при прочтении удаляется из слова. Последняя буква стека также удаляется при прочтении. Говорят, что она выводится из стека. Буква I, полученная с помощью Υ записывается в стек. Слово принимается АМП, если и только если после прочтения этого слово начиная со стартового состояния и пустого стека, автомат переходит в конечное состояние и стек снова пуст. Язык, принимаемый M, обозначим $L(M)$.

Определение 35. M **— детерминированный автомат с магазинной памятью** *если* $\Upsilon \subseteq ((\Sigma \bigcup \{\lambda\} \times Q \times I \bigcup \{\lambda\}) \times (Q \times I \bigcup \{\lambda\}))$ *имеет следующее свойство: если* $((s, a, c), (s', c'))$ *и* $((s, a, c), (s'', c'')) \in F$, *то* $s' = s''$ *и* $c' = c''$.

Пример:



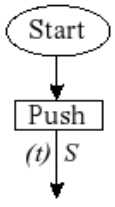
Работа этого автомата при обработке строки *abba*

Действие	Стек	Лента
Start (начало)	λ	*abba*
read (ввод)	λ	*bba*
push (запись в стек) a	a	*bba*
resd (ввод)	a	*ba*
pop (вывод)	λ	*ba*
read (ввод)	λ	a
push (запись в стек) b	b	a
read (ввод)	b	λ
pop (вывод)	λ	λ
accept (прием)	λ	λ

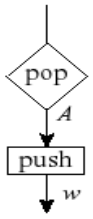
Теорема 20. *Язык контекстно-свободной грамматики* Γ *— язык некоторого автомата с магазинной памятью.*

Доказательство. Доказательство конструктивно. Построим автомат по данной грамматике.

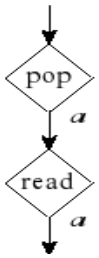
1. Начинаем с автомата записи в стек S.



2. Если нетерминал A выводится из стека, то для некоторого правила $A \to w$ в Γ, w записывается в стек, то есть строим автомат



3. Если терминал a выводится из стека, то этот же символ должен быть прочитан, то есть строим автомат



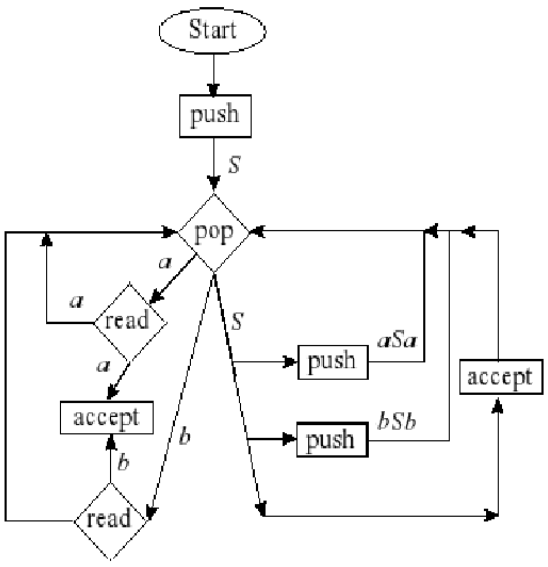
□

Пример:

Пусть $\Gamma = (N, T, S, P)$ — грамматика $N = \{S\}$, $T = \{a, b\}$, и P состоит из правил

$$S \to aSa,\ S \to bSb,\ S \to \lambda.$$

Соответствующий этой грамматике автомат:



Задачи

1. Построить автомат с магазинной памятью, принимающий язык: порожденный грамматикой $\Gamma = (N, T, S, P)$, где $N = \{S, A, B\}$, $T = \{a, b, c\}$, и P состоит из правил

a)
$$S \to aA,\ A \to aAB,\ A \to a,\ B \to bB \to \lambda.$$

b)
$$S \to AB,\ A \to abaA,\ A \to \lambda,\ B \to Bcacc,\ B \to \lambda.$$

c)
$$S \to AcB,\ A \to abaA,\ A \to \lambda,\ B \to Bcacb,\ B \to \lambda.$$

d)
$$S \to AB,\ A \to acA,\ B \to bcB,\ B \to bB,$$
$$A \to aAa,\ B \to \lambda,\ A \to \lambda.$$

e)
$$S \to AB,\ A \to aAc,\ B \to bBc,\ B \to bB,$$
$$A \to AaA,\ B \to \lambda,\ A \to \lambda.$$

3.4 Контекстно-свободные языки и лемма о накачке

Определение 36. *Пусть* Γ *— контекстно-свободная грамматика, назовем язык* $L(\Gamma)$ **контекстно-свободным**.

Предположим, что грамматика Γ представлена в нормальной форме Хомского.

Лемма 14. *Пусть* $A \Rightarrow^* w$*, где* $A \in N$*, и высота соответствующего дерева разбора с корнем* A *равна* n*. Тогда длина* w *не больше* 2^{n-1}*.*

Доказательство. Индукция по высоте дерева разбора. Если $n = 1$, то вывод имеет вид $A \to a$, и длина $w = a$ равна $1 = 2^0$. Пусть утверждение верно для $n = k$. Рассмотрим вывод $A \Rightarrow^* w$ с деревом разбора высоты $k + 1$. Тогда $A \Rightarrow BC \Rightarrow^* uv = w$, где $B \Rightarrow^* u$, $C \Rightarrow^* v$ и дерево разбора для двух последних выводов имеет высоту k. То есть, по предположению индукции, строки u и v имеют длину не больше 2^{k-1}. Значит w — длины не больше $2 \cdot 2^{k-1} = 2^k = 2^{(k+1)-1}$. □

Теорема 21. *(Лемма о накачке) Пусть L — контекстно-свободный язык. Тогда существует такое число $M \in \mathbb{N}$, что любое слово длиннее M в L представимо в форме $xuvwy$, где $uw \neq \lambda$, $|uvw| \leq M$, и $xu^n wv^n y \in L$ для всех $n \geq 0$.*

Доказательство. Пусть $L \setminus \{\lambda\}$ — непустой язык, порожденный грамматикой $\Gamma = (N, T, S, P)$ в нормальной форме Хомского, причем $|P| = p$. Положим $M = 2^p$. Пусть существует слово $w \in L$ длины не меньше M. Тогда по предыдущему утверждению, соответствующее дерево разбора выше p. Следовательно, в дереве разбора существует путь $S \to \cdots \to a$, где a — буква, длины больше p, и $a \in w$. Поскольку Γ содержит только p различных правил, некоторые нетерминалы встречаются в правилах вывода w слева чаще одного раза. Пусть C — первый такой нетерминал. Тогда определен вывод

$$S \Rightarrow^* \alpha C \beta \Rightarrow^* xuCvy \Rightarrow^* xuwvy,$$

где $\alpha \Rightarrow^* x$, $\beta \Rightarrow^* y$, $C \Rightarrow^* uCv$ и $C \Rightarrow w$. Однако, используя эти выводы мы можем получить вывод

$$S \Rightarrow xuCvy \Rightarrow^* xuuCvvy \Rightarrow^* xu^n wv^n y$$

для каждого $n \in \mathbb{N} \setminus \{0\}$.

Так как первое правило вывода имеет вид $C \Rightarrow AB \Rightarrow^* uCv$, и нет пустых слов, либо u либо v непусто. Зафиксируем букву a в слове uwv; можно пройти от нее обратно до S по дереву разбора использую по одному разу выводы $C \Rightarrow^* uCv$ и $C \Rightarrow w$. Следовательно, длина этого пути не больше p, значит $|uwv| \leq M$. □

Следствие 1. *Язык $L = \{a^m b^m a^m | m \geq 1\}$ — не контекстно-свободный язык.*

Доказательство. Пусть m настолько велико, что длина $a^m b^m a^m$ больше M. Тогда $a^m b^m a^m = puqvr$, где $pu^n qv^n r \in L$, для всех $n \geq 1$. Пусть либо u, либо v, либо оба u и v содержат и a и b, то $(a^i b^j)^n$ должно быть подсловом $pu^n qv^n r$, что невозможно. Следовательно, u и v состоят целиком из a или b. Обе эти строки на могут быть одинаковыми, иначе при возведении их в степень количество одних букв будет расти, а других — не изменится. Следовательно, u состоит из a, v — из b. С другой стороны, u состоит из a в начале каждого слова $a^m b^m a^m$, а v из a в конце каждого такого же слова, полученное противоречие доказывает утверждение. □

Теорема 22. *Множество контекстно-свободных языков замкнуто относительно операций конкатенации, объединения и замыкания Клини.*

Доказательство. Пусть L_1 и L_2 порождены грамматиками $\Gamma_1 = (N_1, T_1, S_1, P_1)$ и $\Gamma_2 = (N_2, T_2, S_2, P_2)$, соответственно. Пусть N_1 и N_2 различны. Этого всегда можно добиться переименованием букв.

Язык L_1L_2 можно представить как язык, порожденный грамматикой $\Gamma = (N, T, S, P)$, где $N = N_1 \bigcup N_2 \bigcup \{S\}$, $T = T_1 \bigcup T_2$, а $P = P_1 \bigcup P_2 \bigcup \{S \to S_1S_2\}$. Если $u \in L_1$, $v \in L_2$, то $S_1 \Rightarrow^* u$ в Γ_1, $S_2 \Rightarrow^* v$ в Γ_2, и применяя левый вывод, получим $S \Rightarrow S_1S_2 \Rightarrow^* uS_2 \Rightarrow^* uv$ в Γ.

Язык $L_1 \bigcup L_2$ порожден грамматикой $\Gamma = (N, T, S, P)$, где $N = N_1 \bigcup N_2 \bigcup \{S\}$, $T = T_1 \bigcup T_2$, и $P = P_1 \bigcup P_2 \bigcup \{S \to S_1, S \to S_2\}$. Пусть $w \in L_1 \bigcup L_2$. Тогда $w \in L_1$ или $w \in L_2$. Если $w \in L_1$, то $S_1 \Rightarrow^* w$ в Γ_1, и $S \Rightarrow S_1 \Rightarrow^* w$ в Γ. Если $w \in L_2$, то $S_2 \Rightarrow^* w$ в Γ_2, и $S \Rightarrow S_2 \Rightarrow^* w$ в Γ.

Язык L_1^* порожден грамматикой $\Gamma = (N, T, S, P)$, где $N = N_1 \bigcup \{S\}$, $T = T_1$, и $P = P_1 \bigcup P_2 \bigcup \{S \to S_1S, S \to \lambda\}$. Пусть $w_1, w_2, w_3, \ldots, w_n \in L_1$. Теперь используя правила $S \to S_1S$ и $S \to \lambda$, мы получаем вывод $S \Rightarrow^* S_1^n = S_1S_1S_1 \cdots S_1$. Дальше левыми выводами получаем $S \Rightarrow^* S_1S_1S_1 \cdots S_1 \Rightarrow^* w_1S_1S_1 \cdots S_1 \Rightarrow^* w_1w_2S_1 \cdots S_1 \Rightarrow^* w_1w_2 \cdots w_n$ в Γ. Следовательно, $L_1^* = L$. □

Теорема 23. *Множество контекстно-свободных языков незамкнуто относительно операций пересечения и дополнения.*

Доказательство. Достаточно рассмотреть языки $\{a^nb^na^m | m, n \geq 0\}$ и $\{a^nb^ma^m | m, n \geq 0\}$. Первый порожден грамматикой с правилами $P = \{S \to BC, B \to aBb, B \to \lambda, C \to aC, C \to \lambda\}$, второй — $P = \{S \to AB, A \to aA, A \to \lambda, B \to bBa, B \to \lambda\}$. □

Теорема 24. *Пересечение регулярного и контекстно-свободного языков— контекстно-свободный язык.*

Доказательство. Пусть автомат с магазинной памятью $M = (\Sigma, Q, s, I, \Upsilon, F)$, где Σ — алфавит, Q — множество состояний, s — начальное состояние, I — множество стековых символов, F — множество конечных состояний, Υ — функция перехода. Рассмотрим детерминированный автомат $M_1 = (\Sigma_1, Q_1, q_0, \Upsilon_1, F_1)$, где Σ_1 — алфавит, Q_1 — множество состояний, q_0 — начальное состояние, F_1 — множество приемных состояний, и Υ_1 — функция перехода. Определим автомат с магазинной памятью следующим образом: $M_2 = (\Sigma_2, Q_2, s_2, I_2, \Upsilon_2, F_2)$, где $s_2 = (s, q_0)$, $\Sigma_2 = \Sigma_1 \bigcup \Sigma$, $I_2 = I$, $Q_2 = Q \times Q_1$, и $F2 = F \times F_1$. Определим Υ_2 как $(((s_i, q_j), a, X), ((s_m, q_n), b)) \in \Upsilon_2 \Leftrightarrow ((s_i, a, X), (s_m, b)) \in \Upsilon$ и $\Upsilon_1(q_j, u) = q_n$ в M_1. Строка w принимается $M_2 \Leftrightarrow ((s, q_0), w, \lambda) \vdash_2^* ((s_a, q_b), \lambda) \in M_2$, где s_a и q_b — приемные состояния M и M_1, соответственно. Таким образом, w принимается как автоматом с магазинной памятью M, так и регулярным автоматом M_1. □

Определение 37. *Нетерминал контекстно-свободной грамматики* **бесполезен**, *если он не фигурирует ни в одном выводе* $S \Rightarrow^* w$, $w \in \Sigma^*$. *Иначе, нетерминал* **полезен**.

Теорема 25. *Для данной контекстно-свободной грамматики можно найти и удалить все бесполезные нетерминалы.*

Доказательство. Сначала удалим любой такой нетерминал U, что нет ни одного вывода $\Rightarrow^* w$, $w \in \Sigma^*$. Для этого, определим X по правилам:

1) Для каждого нетерминала V, такого что $V \to w$ — правило для $w \in \Sigma^*$, положим $V \in X$.

2) Если $V \to V_1 V_2 \cdots V_n$, где $V_i \in X$ или Σ^* для $1 \leq i \leq n$, опять $V \in X$.

Продолжим применять 2) до тех пор пока не прекратим добавлять новые нетерминалы к X. Каждый нетерминал U, не принадлежащий X не имеет ни одного вывода вида $U \Rightarrow^* w$. Если $S \notin X$, то язык, порожденный данной контекстно-свободной грамматикой, пуст и доказательство закончено. Нет ни одного полезного терминала. Пусть $S \in X$. Все правила, содержащие нетерминала не из X удалим из множества P.

Удалим теперь каждый нетерминал U, недостижимый из S, т.е. нет ни одного вывода $S \Rightarrow^* W$, в котором U — элемент строки W. Для каждого нетерминала U построим множество Y_U:

1) Если $V \to W$ и U — элемент строки W, то $V \in Y_U$.

2) Если $R \Rightarrow T$, где элемент Y_U присутствует в строке T, то $R \in Y_U$.

Продолжим применять 2) до тех пор пока не закончим добавлять нетерминалы к Y_U. Если $S \in Y_U$, то U достижим из S. Иначе, U недостижим из S. Удалим все правила, содержащие нетерминалы, недостижимые из S. □

Теорема 26. *Существует алгоритм определения того, является ли данный контекстно-свободный язык L пустым*

Доказательство. Контекстно-свободный язык L пуст $\Leftrightarrow$ L не содержит полезных терминалов. □

Теорема 27. *Пусть даны $w \in \Sigma^*$, и контекстно-свободная грамматика G. Тогда существует алгоритм определения принадлежит ли w $L(G)$.*

Доказательство. Пусть G представлена в нормальной форме Хомского. Пусть длина w равна $n \geq 1$. Вывод $S \Rightarrow^* w$ имеет длину $k \leq 2^{n-1}$, поскольку G в нормальной форме Хомского. Следовательно, достаточно проверить все выводы длины $k \leq 2^{n-1}$. □

Теорема 28. *Пусть G — контекстно-свободная грамматика в нормальной форме Хомского с p правилами. Язык $L(G)$ бесконечен $\Leftrightarrow$ существует такая строка $\omega \in L(G)$, что $2^p < |\omega| < 2^{p+1}$.*

Доказательство. Пусть существует слово длины больше 2^p, тогда по Лемме о накачке, $L(G)$ бесконечен. Иначе, пусть ω — кратчайшее слово длины большей 2^{p+1}. По Лемме о накачке, $\omega = xu^iwv^iy$, где длина $uvw \leq 2^p$ и $\mu = xu^{i-1}wv^{i-1}y \in L(G)$. Но $|\mu| > |\omega| - |uv| \geq 2^p$. Также $|\mu| < |\omega|$ и w — кратчайшее слово длины не меньше 2^{p+1}. Следовательно, $|\mu| < 2^{p+1}$. □

Задачи

1. Пусть грамматика $\Gamma = (N, T, S, P)$ состоит из $N = \{S, A, B\}$, $T = \{a, b, c\}$, и P состоит из правил

$$S \to AB,\ A \to acA,\ B \to bcB,\ B \to bB,$$

$$A \to aBa,\ B \to \lambda,\ A \to a.$$

Построить грамматику, порождающую L^*.

2. Пусть L_1 — язык, порожденный грамматикой $\Gamma_1 = (N, T, S, P_1)$, где $N = \{S, A, B\}$, $T = \{a, b, c\}$, и P_1 состоит из правил

$$S \to aA,\ A \to aAB,\ A \to a,\ B \to b,\ B \to \lambda,$$

а L_2 — язык, порожденный грамматикой $\Gamma_2 = (N, T, S, P_2)$, где $N = \{S, A, B\}$, $T = \{a, b, c\}$, и P_2 состоит из правил

$$S \to AB,\ A \to abaA,\ A \to \lambda,\ B \to Bcacc,\ B \to \lambda.$$

Найти грамматику, порождающую L_1L_2.

3.Пусть L_1 — язык, порожденный грамматикой $\Gamma_1 = (N, T, S, P)$, где $N = \{S, A, B\}$, $T = \{a, b, c\}$, и P_1 состоит из правил

$$S \to AdB,\ A \to abaA,\ A \to \lambda,\ B \to Bcacb,\ B \to \lambda,$$

а L_2 — язык, порожденный грамматикой $\Gamma_2 = (N, T, S, P_2)$, где $N = \{S, A, B\}$, $T = \{a, b, c\}$, и P_2 состоит из правил

$$S \to AB,\ A \to acA,\ B \to bcB,\ B \to bB,$$

$$A \to aAa,\ B \to \lambda,\ A \to \lambda.$$

Найти грамматику, порождающую $L_1 \bigcup L_2$.

4. Является ли язык контекстно-свободным? Если да, то построить грамматику, порождающую его.

a) $L = \{a^m b^n c^{2n} | m, n = 1, 2, \ldots\}$.

b) $L = \{ww^R w | w \in a, b^*\}$.

c) $L = \{w \in \{a, b, c\}^* | w$состоит из одинакового числаaи$b\}$.

d) $L = \{a^n b^{2n} c^n | n = 1, 2, \ldots\}$.

Глава 4

Машина Тьюринга

4.1 Детерминированная машина Тьюринга

Определение 38. Детерминированная машина Тьюринга — *набор* $(Q, \Sigma, \Gamma, \delta, s_0, h)$, *где*

Q — *множество состояний,*

Γ — *конечное множество символов на ленте, включающее в себя алфавит* Σ *и символ пустой ячейки* $\sharp$,

s_0 — *начальное состояние,*

h — *конечное состояние,*

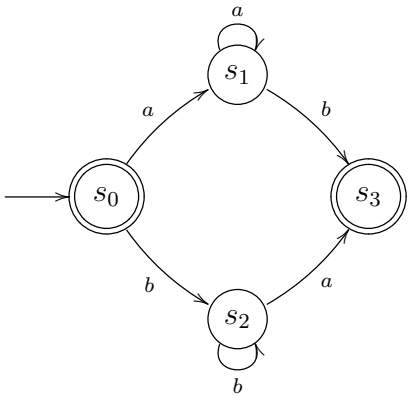
δ — *функция из* $Q \times \Gamma$ *в* $Q \times \Gamma \times N$, *где* N *состоит из* L, *что означает переход к левой ячейке от рассматриваемой,* R — *к правой и* $\sharp$ — *отсутствие перехода.*

Определение 39. *Слово* $w \in \Sigma^*$ **принимается машиной Тьюринга** T *если, начиная со стартового состояния по прочтении* w *машина переходит в конечное. Язык, принимаемый машиной Тьюринга* T — *множество всех таких слов.*

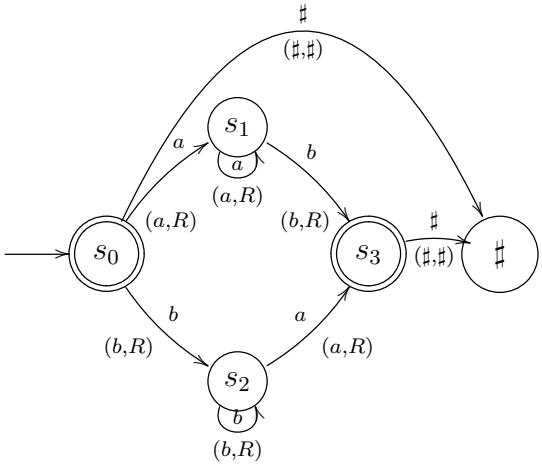
Представим правило (s_i, a, s_j, b, R) как $s_i \overset{a}{\underset{(b,R)}{\rightarrow}} s_j$

Теорема 29. *Любой регулярный язык распознается некоторой машиной Тьюринга.*

Пример:
Автомату



Соответствует машина Тьюринга



Определение 40. *Языки, принимаемые машинами Тьюринга называются* **рекурсивно-перечислимыми**.

Примеры:

1.

Построим машину Тьюринга, описывающую язык $\{a^n b^n | n \in \mathbb{N} \setminus \{0\}\}$. Прочитываем первую букву a, меняем ее на A:

$$(s_0, a, s_1, A, R)$$

Прочитываем все a пока не встретим букву b, которую заменим на B и повернем назад:

$$(s_1, a, s_1, a, R),\ (s_1, B, s_1, B, R),\ (s_1, b, s_2, B, L).$$

Возвращаясь ко второй a, проходим через все B и a, не меняя их:

$$(s_2, B, s_2, B, L),\ (s_2, a, s_2, a, L).$$

После встречи A снова идем по слову вперед

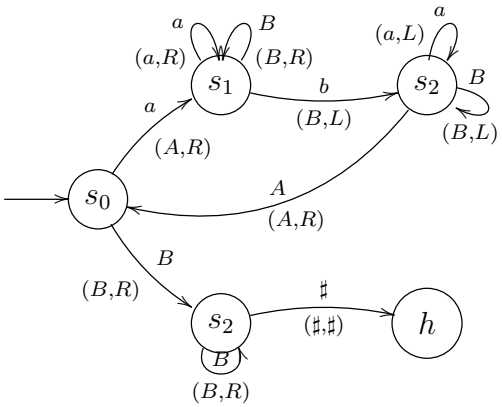
$$(s_2, A, s_0, A, R).$$

Пусть процесс считывания a закончен

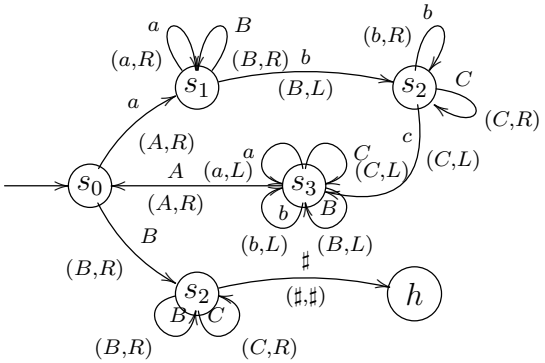
$$(s_0, B, s_3, B, R).$$

В состоянии s_3 не воспринимаем ничего, кроме B и $\sharp$:

$$(s_3, B, s_3, B, R),\ (s_3, \sharp, h, \sharp, \sharp).$$



2. Аналогично строится машина Тьюринга для языка $\{a^n b^n c^n | n \in \mathbb{N} \setminus \{0\}\}$.



Задачи

1. Построить машину Тьюринга, принимающую язык
   a) $ab^*c^*(b \vee ac)$;
   b) $abc(b \vee ac)^*b$;
   c) $(aa^*bb^*)^*$.

2. Построить машину Тьюринга, принимающую строки, состоящие из одинакового количества a и b.

3. Построить машину Тьюринга, которая по введенной строке печатает ее в обратном порядке.

4. Построить машину Тьюринга, принимающую все четные палиндромы.

5. Построить машину Тьюринга, принимающую все нечетные палиндромы.

6. Построить машину Тьюринга, принимающую все строки вида $a^n b^n a^n$, $n \in \mathbb{N} \setminus \{0\}$.

7. Построить машину Тьюринга, принимающую строки вида ww, $w \in \{a, b\}^*$.

4.2 Недетерминированные машины Тьюринга и контекстно-свободные языки

Определение 41. *Машина Тьюринга* **недетерминированная**, *если* δ *— конечное подмножество* $(Q \times \Gamma) \times (Q \times \Gamma \times N)$.

Теорема 30. *Любой контекстно-свободный язык принимается некоторой недетерминированной машиной Тьюринга.*

Доказательство. Для каждого из правил автомата с магазинной памятью построим соответствующую команду машины Тьюринга.

1. $((a, s, E), (t, D))$ — В состоянии s, прочитываем a, выводим E, переходим в состояние t и перемещаем в стек D.

2. $((a, s, \lambda), (t, D))$ — В состоянии s, прочитываем a, переходим в состояние t и перемещаем в стек D.

3. $((\lambda, s, \lambda), (s, D))$ — В состоянии s перемещаем в стек D.

4. $((a, s, E), (t, \lambda))$ — В состоянии s, прочитываем a, выводим E и переходим в состояние t.

5. $((\lambda, s, E), (s, \lambda))$ — В состоянии s выводим E.

6. $((a, s, \lambda), (t, \lambda))$ — В состоянии s, прочитываем a и переходим в состояние t.

7. $((a, s, \lambda), (s, \lambda))$ — В состоянии s, прочитываем a.

1. Переходим в первую ячейку после $\triangledown$, стираем E и вписываем D. Возвращаемся к a и стираем a. Переходим в состояние t.

2. Переходим в первую ячейку после $\triangledown$, вписываем D. Возвращаемся к a и стираем a. Переходим в состояние t.

3. Переходим в первую ячейку после $\triangledown$, вписываем D. Возвращаемся к первоначальной букве. Переходим в состояние s.

4. Переходим в первую ячейку после $\triangledown$, стираем E. Возвращаемся к a и стираем a. Переходим в состояние t.

5. Переходим в первую ячейку после $\triangledown$, стираем E. Возвращаемся к первоначальной ячейке. Переходим в состояние s.

6. Стираем a. Переходим в состояние t.

7. Стираем a. Переходим в состояние s. □

Теорема 31. *Язык, принимаемый недетерминированной машиной Тьюринга T, принимается и детерминированной машиной Тьюринга T'.*

Доказательство. Поскольку T недетерминированна, и $\theta(s,x) \subset (Q \times \Gamma \times N)$, $\theta(s,x)$ состоит из k элементов, которые обозначим через

$$\theta_0(s,x), \theta_1(s,x), \theta_2(s,x), \ldots, \theta_{k-1}(s,x).$$

Таким образом, каждый элемент $\theta_i(s,x)$ корректно определен. Например, для $\theta_0(s,x) = (s',b,R)$ получим правило (s,x,s',b,R), и для $\theta_1(s,x) = (s'',C,L)$ имеем правило (s,x,s'',C,L).

Пронумеруем элементы каждого $\theta(s_i,a_i)$ для всех состояний s_i и каждого $a_i \in \Sigma$. Следовательно, если мы находимся в состоянии s, получаем ввод x, и номер j, мы можем воспользоваться $\theta_j(s,x)$ для упрощения используемого правила. Пусть нам никогда не потребуется больше $n+1$ числа для нумерации элементов $\theta(s_i,a_i)$, тогда для любой последовательности натуральных чисел $m_1, m_2, \ldots, m_p$, каждое из которых не больше n, мы можем последовательно применить $\theta_{m_1}, \theta_{m_2}, \ldots, \theta_{m_p}$, которые вместе с состояниями и вводом определяют используемые правила. По применению всех возможных последовательностей, мы получим все возможные вычисления. То есть, если слово воспринимается машиной Тьюринга T, оно также воспринимается одной из приведенных последовательностей. □

Задачи

1. Построить (не обязательно детерминированные) машины Тьюринга, соответствующие языкам

a) Язык из строк, содержащих вдвое больше a чем b.

b) Язык из строк, содержащих одно и то же количество a и b.

c) Язык $\{a^n b^n | n = 1,2,\ldots\}$.

d) Язык $\{a^n b^k c^n | k,n = 1,2,\ldots\}$.

e) Язык из палиндромов нечетной длины над алфавитом $\{a,b,c\}$.

f) Язык из палиндромов четной длины над алфавитом $\{a,b,c\}$.

g) Язык из палиндромов над алфавитом $\{a,b,c\}$.

h) Язык $\{a^n b^n a^m b^m | m,n = 1,2,\ldots\}$.

i) Язык $\{a^n b^{2n} a^m b^{2m} | m,n = 1,2,\ldots\}$.

4.3 Проблема остановки для машин Тьюринга

Проблема остановки Существует ли алгоритм, определяющий, для данной машины Тьюринга T и произвольной строки w, достигнет ли T конечного состояния после обработки w?

Определение 42. *Язык L разрешим по Тьюрингу, если существует машина Тьюринга, которая по приведенному слову выдает Y если слово принадлежит L и N в противном случае.*

Теорема 32. *Если язык разрешим по Тьюрингу, то он принимается некоторой машиной Тьюринга.*

Доказательство. Если язык разрешим по Тьюрингу, то существует машина Тьюринга, которая по приведенному слову выдает Y если слово принадлежит L и N в противном случае. Модифицируем эту машину так, чтобы вместо печати N, она зацикливалась, а вместо вывода Y, останавливалась. □

Теорема 33. *Если язык L разрешим по Тьюрингу, то и язык $L' = A^* \setminus L$ разрешим по Тьюрингу.*

Доказательство. Если язык L разрешим по Тьюрингу, то существует машина Тьюринга, которая по приведенному слову выдает Y если слово принадлежит L и N в противном случае. Модифицируем эту машину так, чтобы вместо печати N, она выдавала Y и наоборот. □

Теорема 34. *Язык L разрешим по Тьюрингу $\Leftrightarrow$ оба языка L и L' принимаемы по Тьюрингу.*

Доказательство. ($\Rightarrow$) Если язык L разрешим по Тьюрингу, то по теореме 33 его дополнение L' тоже разрешимо по Тьюрингу. Следовательно, по теореме 32 оба языка L и L' принимаемы по Тьюрингу.

($\Leftarrow$) Пусть существую машины Тьюринга M и M', принимающие L и L', соответственно. Если строка ввода принимаема M, введем Y. Если она принимается M', выведем N. Так как этот процесс алгоритмичен, по гипотезе Черча его можно реализовать в виде машины Тьюринга M''. Следовательно L разрешим по Тьюрингу и по теореме 33, L' тоже разрешим по Тьюрингу. □

Теорема 35. *Любая машина Тьюринга над алфавитом $\{a, b\}$ однозначно реализуется в виде строки из a и b.*

Доказательство. Пусть множество состояний машины — $S = \{s_1, s_2, s_3, \ldots, s_n\}$. Каждое правило имеет вид

$$(s_i, a_1, s_j, a_2, N),$$

где $a_1, a_2 \in \{a, b\} \bigcup \{\sharp, \triangledown\}$, $N \in \{L, R, \sharp\}$.

Заменим s_i на строку из i букв a. После этой строки пишем b — разделитель. Заменим символы $a, b, \sharp, \triangledown$ на aa, bb, ab, ba, соответственно. Последний символ L заменяем на a, а R на b. Например, строка для $(s_4, b, s_2, \sharp, R)$ — $aaaabbbaababb$, где $aaaa$ представляет s_4, b — разделитель, bb — b, aa — s_2, b — разделитель, ab — $\sharp$, b — R.

□

Обозначим строку, представляющую машину Тьюринга M, через $c(M)$.

Теорема 36. *Существует язык, принимаемый некоторой машиной Тьюринга, но не разрешимый по Тьюрингу.*

Доказательство. Рассмотрим язык L_0, состоящий из строк $c(M)bw$, представляющих машину Тьюринга M и строк w, принимаемых M. Легко построить машину Тьюринга IMM, принимающую L_0. Для данной строки t, MM сначала расшифровывает t и если строка представляет строку машину Тьюринга M и вводную строку, MM поставляет последнюю подстроку t в M, и MM принимает $t = c(M)bw \Leftrightarrow M$ принимает w. Следовательно, L_0 принимаем по Тьюрингу.

Если, кроме того, L_0 разрешим по Тьюрингу, то каждый принимаемый по Тьюрингу язык разрешим по Тьюрингу.

Докажем, что L_0 не разрешим по Тьюрингу. Докажем сначала, что если L_0 разрешим по Тьюрингу, то и язык $L_1 = \{c(M) | M \text{принимает} c(M)\}$ тоже разрешим по Тьюрингу. Построим M_1 по следующим правилам: Для строки s, рассмотрим строку sbs и используем ее как ввод для MM_2 (машина, принимающая L_0). Если MM_2 выводит Y, то $s = c(M)$ для некоторой машины M, принимающей $c(M)$. Следовательно, $s \in L_1$ и M_1 тоже выводит Y. Если MM_2 выдает N, то $s = c(M)$ для каждой машины M, которая принимает $c(M)$, то есть $s \notin L_1$ и M_1 выводит N. Следовательно. L_1 разрешим по Тьюрингу.

Покажем, что L_1 не разрешим по Тьюрингу. По теореме 33 L_1 разрешим по Тьюрингу $\Leftrightarrow L_1'$ разрешим. То есть, достаточно доказать, что L_1 не разрешим по Тьюрингу. Язык $L_1 = \{\{w | w \in \{a, b\}^*\}$ либо $w = c(M)$для каждой машины M, либо $w = c(M)$ для некоторой машины M, но Mне принимает $w\}$. Вспомним парадокс Расселла. Пусть M_1' — машина, принимающая L_1', тогда верно ли, что $c(M_1') \in L_1$? Если так, то M_1' не

принимает $c(M_1')$ по определению L_1'. Но M_1' принимает $c(M_1')$ поскольку она принимает любой элемент L_1', и мы пришли к противоречию. Предположим теперь, что $c(M_1') \notin L_1'$. Тогда $c(M_1') \in L_1$. Следовательно, M_1' принимает $c(M_1')$ по определению L_1. Но M_1' принимает только элементы L_1', то есть $c(M_1') \in L_1'$, снова противоречие. □

Задачи

1. Доказать, что конечное множество всегда разрешимо по Тьюрингу.

2. Построить строку, соответствующую правилу $(s_5, \triangledown, s_2, a, R)$.

3. Найти $c(M)$, где M — машина, определяемая правилами

$$(s_1, a, s_2, a, R),\ (s_1, b, s_2, b, R),\ (s_2, a, s_2, a, R),$$

$$(s_2, b, s_2, b, R),\ (s_1, \sharp, s_3, \sharp, R).$$

4. Найти $c(M)$, где M — машина, определяемая правилами

$$(s_1, a, s_2, \sharp, R),\ (s_1, b, s_2, a, R),\ (s_2, a, s_2, \sharp, R),$$

$$(s_2, b, s_2, a, R),\ (s_1, \sharp, s_3, \sharp, R).$$

5. Найти правило, соответствующее строке

a) *aaababbbaa*.

b) *aabbbaaabbab*.

6. Которые из строк представляют правила?

a) *baaabbaabb*;

b) *aabbbaaabbbb*;

c) *aababaabbaa*;

d) *aabaabaabaab*;

e) *aabaaabbabb*.

7. Восстановить машину Тьюринга по строке

a) *abaaabbbbaabbbabaababababbaababb*;

b) *abaaaaababbabbbaababbaababaaaabab*.

8. Восстановить машину Тьюринга и строку ввода по строке

a) *abaaaabbbbabbbaabbbbaabbbabbbaababaaaababbbaaabbb*;

b) *abaaaabbbbabbbaabaabaabbbabbbaaabaaabbbaaababaaaababababaabb*.

9. Построить метод кодирования, который допускает использование символов A и B наравне с a, b посредством применения слов длины 3 для обозначения вводимых символов.

10. Используйте код из предыдущей задачи для нахождения строки, представляющей

$$(s_1, a, s_3, A, R).$$

11. Найти строку, реализующую машину

$$(s_1, a, s_2, b, R),\ (s_1, b, s_2, b, R),\ (s_2, a, s_2, \sharp, R),$$

$$(s_2, b, s_2, b, R),\ (s_1, \sharp, s_3, \sharp, R)$$

вместе с вводимым словом *ababaab*.

12. Найти строку, реализующую машину

$$(s_1, a, s_2, b, R),\ (s_1, b, s_2, a, R),\ (s_2, a, s_2, \sharp, R),$$

$$(s_2, b, s_2, \sharp, R),\ (s_1, \sharp, s_3, \sharp, R)$$

вместе с вводимым словом *babbab*.

13. Пусть L — язык. Доказать, что верно только одно из перечисленного:

a) Ни L ни L' не принимаются машинами Тьюринга.

b) И L и L' разрешимы по Тьюрингу.

c) Либо L либо L' принимается некоторой машиной Тьюринга, но не разрешим по Тьюрингу.

4.4 Проблемы неразрешимости для контекстно-свободных языков

Определение 43. *Пусть Σ — алфавит. Обозначим через P множество упорядоченных пар непустых строк $(u_1, v_1), (u_2, v_2), \ldots, (u_m, v_m)$ из символов Σ. То есть P — конечное подмножество $\Sigma^* \times \Sigma^*$.* **Решение** *P — строка w, для которой существует такая последовательность пар (u_{i_1}, v_{i_1}), (u_{i_2}, v_{i_2}), …, (u_{i_m}, v_{i_m}), что $w = u_{i_1} u_{i_2} \cdots u_{i_m} = v_{i_1} v_{i_2} \cdots v_{i_m}$.*

Проблема соответствий Поста — *определить, существует ли решение.*

Определение 44. *Пусть Σ — алфавит. Обозначим через P множество упорядоченных пар непустых строк $(u_1, v_1), (u_2, v_2), \ldots, (u_m, v_m)$ из символов Σ вместе со специальной парой. (u_0, v_0). В* **модифицированной системе соответствий**, **решение** *P — такая строка w, что существует последовательность пар (u_0, v_0), (u_{i_1}, v_{i_1}), (u_{i_2}, v_{i_2}), …, (u_{i_m}, v_{i_m}), $w = u_0 u_{i_1} u_{i_2} \cdots u_{i_m} = v_0 v_{i_1} v_{i_2} \cdots v_{i_m}$.* **Модифицированная проблема соответствий Поста** — *определить, существует ли решение для модифицированной системы соответствий.*

Лемма 15. *Если разрешима проблема соответствий Поста, то разрешима и модифицированная проблема Поста.*

Доказательство. Пусть P_1 — модифицированная система соответствий (u_0, v_0), (u_1, v_1), (u_2, v_2), ..., (u_m, v_m). Пусть символы \$ и $\star$ не принадлежат Σ. Для строки $w = a_1a_2a_3\cdots a_k$, определим $L(w) = \star a_1\star a_2\star a3\star\cdots\star a_k$ и $R(w) = a_1 \star a_2 \star a_3 \star \cdots \star a_k\star$. Пусть P_2 содержит пару $(L(u_0), L(v_0)\star)$, и для остальных пар $(u, v) \in P_1$, пусть $(L(u), R(v)) \in P_2$. Кроме того, включим в P_2 пару $(\star\$, \$)$. Очевидно, что только $(L(u_0), L(u_0)\star)$ может начинать решение P_2, так как это единственная пара, в которой одно слово не начинается с $\star$, а другое — начинается. Также очевидно, что единственная пара, заканчивающая решение P_2 — $(\star\$, \$)$, поскольку это единственная пара, в которой последние символы совпадают.

Пусть существует такое семейство пар (u_0, v_0), (u_{i_1}, v_{i_1}), (u_{i_2}, v_{i_2}), ..., (u_{i_m}, v_{i_m}), что $w = u_0u_{i_1}u_{i_2}\cdots u_{i_m} = v_0v_{i_1}v_{i_2}\cdots v_{i_m}$. Тогда последовательность $(L(u_0), L(v_0))$, $(L(u_{i_1}), R(v_{i_1}))$, $(L(u_{i_2}), R(v_{i_2}))$, ..., $(L(u_{i_m}), R(v_{i_m}))$, $(\star\$, \$)$ порождает решение $w' = L(u_0)L(u_{i_1})L(u_{i_2})\cdots L(u_{i_m}) \star \$ = L(v_0) \star R(v_{i_1})R(v_{i_2})\cdots\$$ в P_2. Строки $L(u_0)L(u_{i_1})L(u_{i_2})\cdots L(u_{i_m}) \star \$$ и $L(v_0) \star R(v_{i_1})R(v_{i_2})\cdots\$$ в P_2 отличаются от строк $u_0u_{i_1}u_{i_2}\cdots u_{i_m}$ и $v_0v_{i_1}v_{i_2}\cdots v_{i_m}$ в P_1, соответственно, наличием $\star$ между букв и \$ в конце слова.

Следовательно, поскольку решение модифицированной системы соответствий Поста влечет решение системы соответствий Поста, разрешимость проблемы соответствий Поста влечет разрешимость модифицированной проблемы соответствий Поста. □

Теорема 37. *Проблема соответствий Поста неразрешима.*

Доказательство. Докажем, что разрешимость модифицированной проблемы Поста влечет существование машины Тьюринга, принимающей L_0 из предыдущего параграфа. По строке, соответствующей данной машине Тьюринга M и слову w, построим систему соответствий Поста, которая имеет решение $\Leftrightarrow$ M принимает w.

Начинаем с пары $(\sharp, \sharp s_0w)$, $\dfrac{\sharp}{\sharp s_0w}$.

Для каждого символа $X \in \Gamma$ положим $\dfrac{X}{X}$.

Для каждого состояния s, которое не является конечным, каждого состояния s', и символов X, Y, $Z \in \Gamma$, введем

$\dfrac{sX}{Ys'}$, если $\delta(s, X) = (s', Y, R)$.

$\dfrac{XsY}{s'XZ}$, если $\delta(s, Y) = (s', Z, L)$.

$\dfrac{s\sharp}{Xs'\sharp}$, если $\delta(s, \sharp) = (s', X, R)$.

$\boxed{\begin{array}{|l}\hline Xs\sharp \\ \hline s'XY\sharp \\ \hline\end{array}}$, если $\delta(s,\sharp)=(s',Y,L)$.

Назовем эти строки строками, порожденными δ.

Кроме того, добавим правила $(0s_m0,s_m)$, $(1,1)$, $(\sharp,\sharp)$, $(0s_m1,s_m)$, $(1s_m0,s_m)$, $(1s_m1,s_m)$, $(0s_m,s_m)$, (s_m0,s_m), $(1s_m,s_m)$, (s_m1,s_m), $(s_m\sharp\sharp,\sharp)$.

Пусть существует семейство последовательностей, описывающих принятие строки s машиной M, пользуясь индукцией по числу вычислений, покажем существование частичного решения

$$\frac{\sharp s_0w\sharp\alpha_1s_1\beta_1\sharp\alpha_2s_2\beta_2\sharp\ldots\sharp\alpha_{n-1}s_{n-1}\beta_{n-1}\sharp}{\sharp s_0w\sharp\alpha_1s_1\beta_1\sharp\alpha_2s_2\beta_2\sharp\ldots\sharp\alpha_{n-1}s_{n-1}\beta_{n-1}\sharp\alpha_ns_n\beta_n\sharp}.$$

Для $n=0$ имеем $\boxed{\begin{array}{|l}\hline \sharp \\ \hline \sharp s_0w \\ \hline\end{array}}$. Пусть дано

$$\frac{\sharp s_0w\sharp\alpha_1s_1\beta_1\sharp\alpha_2s_2\beta_2\sharp\ldots\sharp\alpha_{k-1}s_{k-1}\beta_{k-1}\sharp}{\sharp s_0w\sharp\alpha_1s_1\beta_1\sharp\alpha_2s_0\beta_2\sharp\ldots\sharp\alpha_{k-1}s_{k-1}\beta_{k-1}\sharp\alpha_ks_k\beta_k\sharp}.$$

Используем одно из δ-правил, чтобы продлить верхнюю строку на $\alpha_ks_k\beta_k\sharp$. Если мы не достигаем конечного состояния, мы не имеем решения. Если мы достигнем конечного состояния, то мы получим и решение. Следовательно, если проблема соответствий Поста разрешима, то и L_0 разрешим. Так как последнее неверно, то и проблема Поста неразрешима. □

Теорема 38. *Проблема проверки $L(G_1)\bigcap L(G_2)=\emptyset$ неразрешима для двух контекстно-свободных грамматик G_1 и G_2.*

Доказательство. Пусть $P\subset\Sigma^*\times\Sigma^*$ — произвольное семейство состояний (u_0,v_0), (u_1,v_1), (u_2,v_2), …, (u_m,v_m). Обозначим далее через w^{-1} слово w в обратном порядке. Пусть G_1 порождена правилами

$$S\to u_iCv_i^{-1},\ i=1,\ldots n.$$

$$C\to u_iCv_i^{-1},\ i=1,\ldots n.$$

$$C\to c.$$

Следовательно, каждое слово $L(G_1)$ имеет вид $u_{i_0}u_{i_1}u_{i_2}\cdots u_{i_m}cv_{i_m}^{-1}\cdots v_{i_2}^{-1}v_{i_1}^{-1}v_{i_1}^{-1}$.

Пусть $L(G_2)=\{wcw^{-1}|w\in\Sigma^*\}$. Тогда $w\in L(G_1)\bigcap L(G_2)\Leftrightarrow w=u_{i_0}u_{i_1}u_{i_2}\cdots u_{i_m}=v_{i_0}v_{i_1}v_{i_2}\cdots v_{i_m}$, что есть решение проблемы соответствий Поста. Следовательно, задача проверки $L(G_1)\bigcap L(G_2)=\emptyset$ неразрешима для произвольной пары контекстно-свободных грамматик. □

Определение 45. *Контекстно-свободная грамматика* **переопределена** *если существует два независимых левых вывода одного и того же слова.*

Теорема 39. *Проблема выяснения переопределенности грамматики неразрешима.*

Доказательство. Пусть $P \subset \Sigma^* \times \Sigma^*$ — произвольное семейство состояний (u_0, v_0), (u_1, v_1), (u_2, v_2), ..., (u_n, v_n). Пусть $\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_n$ — символы не из Σ. Построим две грамматики G_1 и G_2 по правилам:

$$G_1 = (N_1, \Sigma_\alpha, S_1, P_1),$$

где $N_1 = \{S_1\}$, $\Sigma_\alpha = \Sigma \bigcup \{\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_n\}$, и $P_1 = \{S_1 \to \alpha_i S_1 u_i | i = 0, 1, \ldots, n\} \bigcup \{S_1 \to \lambda\}$.

$$G_2 = (N_2, \Sigma_\alpha, S_2, P2),$$

где $N_2 = \{S_2\}$, $\Sigma_\alpha = \Sigma \bigcup \{\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_n\}$, и $P_2 = \{S_2 \to \alpha_i S_2 v_i | i = 0, 1, \ldots, n\} \bigcup \{S_2 \to \lambda\}$.

Очевидно G_1 и G_2 не переопределены.

Пусть $G = (N, \Sigma_\alpha, S, P)$, где $N = \{S, S_1, S_2\}$ и $P = P_1 \bigcup P_2 \bigcup \{S \to S_1, S \to S_2\}$. Если существует решение, одно правило начинается с $S \to S_1$, а другое — с $S \to S_2$, то есть G переопределена. Пусть наоборот, G переопределена, тогда $u_{i_0} u_{i_1} u_{i_2} \cdots u_{i_m} = v_{i_0} v_{i_1} v_{i_2} \cdots v_{i_m}$ и существует решение. Следовательно, решение проблемы соответствий Поста существует $\Leftrightarrow$ контекстно-свободная грамматика переопределена. □

Задачи

1. Доказать, что класс языков, принимаемых машинами Тьюринга замкнут относительно операции объединения.

2. Доказать, что класс языков, принимаемых машинами Тьюринга замкнут относительно операции пересечения.

3. Доказать, что класс языков, разрешимых по Тьюрингу, замкнут относительно операции пересечения.

4. Доказать, что класс языков, разрешимых по Тьюрингу, замкнут относительно операции объединения.

5. Доказать, что класс языков, разрешимых по Тьюрингу, замкнут относительно операции конкатенации.

6. Доказать, что класс языков, принимаемых машинами Тьюринга замкнут относительно операции замыкания Клини.

7. Доказать, что неразрешима проблема выяснения для данных машины Тьюринга M и строки w, принимает ли M все свои состояния в процессе обработки w.

8. Доказать, что проблема проверки тождества $L(\Gamma) = \Sigma^*$ неразрешима для любой контекстно-свободной грамматики Γ.

9. Доказать, что проблема проверки тождества $L(\Gamma) = L(\Gamma')$ неразрешима для произвольной пары контекстно-свободных грамматик Γ, Γ'.

10. Доказать, что не существует алгоритма, позволяющего определить, бесконечно ли пересечение языков двух контекстно-свободных грамматик.

11. Доказать, что не существует алгоритма, позволяющего определить, бесконечно ли дополнение языка контекстно-свободной грамматики.

Литература

[1] J. A. Anderson, *Automata theory with modern applications*, Cambridge University Press, 2006.

[2] Ю. Г. Карпов, *Теория автоматов*, Питер, 2002.

[3] Д. Хопкрофт, Р. Мотвани, Д. Ульман, *Введение в теорию автоматов, языков и вычислений. Второе издание*, Вильямс, 2010.