А. А. Вылиток

О регулярных языках

Регулярные языки играют важную роль в математических теориях и в приложениях. К наиболее известным формализмам, описывающим регулярные языки, относятся: регулярные выражения, конечные автоматы, регулярные грамматики.

Определение. Пусть Σ — алфавит, не содержащий символов *, +, ε, ∅, (,). Определим рекурсивно регулярное выражение γ над алфавитом Σ и регулярный язык $L(\gamma)$, задаваемый этим выражением:

1) $a \in \Sigma \cup \{\varepsilon, \varnothing\}$ есть регулярное выражение; $L(a) = \{a\}$ для $a \in \Sigma \cup \{\varepsilon\}$; $L(\varnothing) = \varnothing$;
2) если α и β — регулярные выражения, то
   а) $(\alpha) + (\beta)$ — регулярное выражение; $L((\alpha) + (\beta)) = L(\alpha) \cup L(\beta)$;
   б) $(\alpha)(\beta)$ — регулярное выражение; $L((\alpha)(\beta)) = L(\alpha)L(\beta)$;
   в) $(\beta)^*$ — регулярное выражение; $L((\beta)^*) = (L(\beta))^*$;
3) все регулярные выражения конструируются только с помощью пунктов 1 и 2.

Если считать, что операция «+» имеет самый низкий приоритет, а операция «*» — наивысший, то в регулярных выражениях можно опускать лишние скобки.

Примеры регулярных выражений над алфавитом $\{a, b\}$: $a + b$, $(a + b)^*$, $((a)^*b)^*$. Соответствующие языки: $L(a + b) = \{a\} \cup \{b\} = \{a, b\}$, $L((a + b)^*) = \{a,b\}^*$, $L(((a)^*b)^*) = \{ x_1bx_2b...x_kb \mid k \geq 1, x_i \in \{a\}^*$ для $i = 1, ..., k\} \cup \{\varepsilon\}$.

Определение. Недетерминированный конечный автомат (НКА) — это пятерка $\mathcal{A} = (K, \Sigma, \delta, I, F)$, где:

K — конечное множество состояний, или вершин;

Σ — входной алфавит (также конечный);

$\delta \subseteq K \times \Sigma \times K$ — множество команд, или дуг;

$I \subseteq K$ — множество начальных состояний;

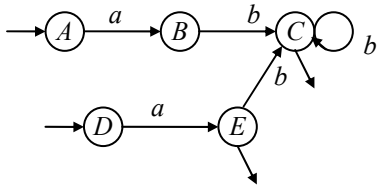
$F \subseteq K$ — множество заключительных состояний.

Множество δ можно также интерпретировать как отображение $K \times \Sigma$ в множество подмножеств K.

Существуют алгоритмы, позволяющие по регулярному выражению построить эквивалентный (т. е. определяющий тот же язык) конечный автомат, и наоборот, по автомату построить эквивалентное ему выражение.

Пример 1. $\mathcal{A}_1 = (\{A, B, C, D, E\}, \{a, b\}, \delta, \{A, D\}, \{C, E\})$, где $\delta = \{(A, a, B), (D, a, E), (B, b, C), (E, b, C), (C, b, C)\}$.

Автомат удобно представлять в виде ориентированного размеченного графа:



Входящими непомеченными стрелками отмечены начальные вершины A и D, исходящими — заключительные вершины E и C.

Каждая дуга НКА $\mathcal{A}$ имеет пометку из Σ. *Путь* в ориентированном графе может быть представлен последовательностью дуг. *Пустой* путь можно представить одной

вершиной, которая считается одновременно началом и концом пути. *Пометка* пути — это сцепление (конкатенация) пометок его дуг. Пустой путь имеет пустую пометку. Путь из начальной вершины в заключительную называется *успешным*. Язык, допускаемый автоматом $\mathcal{A}$ (обозначается $L(\mathcal{A})$), — это множество пометок всех успешных путей автомата.

Приведем примеры путей автомата $\mathcal{A}_1$ (см. пример 1):

(а) путь $E \xrightarrow{b} C \xrightarrow{b} C \xrightarrow{b} C$ имеет пометку *bbb*;

(б) пустой путь A имеет пометку ε;

(в) путь $D \xrightarrow{a} E$ является успешным и имеет пометку *a*;

(г) путь $A \xrightarrow{a} B \xrightarrow{b} C \xrightarrow{b} C \xrightarrow{b} C$ является успешным и имеет пометку *abbb*.

Язык, допускаемый автоматом $\mathcal{A}_1$: $L(\mathcal{A}_1) = \{ab^n \mid n \geq 0\} = L(ab^*)$

Заметим, что $\varepsilon \in L(\mathcal{A})$ если и только если $\mathcal{A}$ имеет вершину, являющуюся одновременно начальной и заключительной.

Еще один способ задать регулярный язык — описать его с помощью регулярной грамматики (грамматики типа 3 по классификации Хомского). Грамматика называется *регулярной*, если она либо праволинейна, либо леволинейна.

Грамматика *праволинейна*, если каждое ее правило относится к одному из трех видов правил: $A \to aB$, $A \to a$, $A \to \varepsilon$, где A, B — означают нетерминалы, a означает терминальный символ, ε — пустая цепочка. Грамматика *леволинейна*, если ее правила имеют вид: $A \to Ba$, $A \to a$, $A \to \varepsilon$.

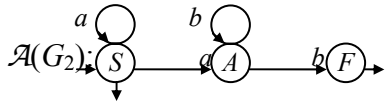
Заметим, что иногда в литературе, описывающей практическое применение регулярных грамматик, в этих грамматиках не допускаются правила вида $A \to \varepsilon$. Это ограничение вполне оправдано. Например, синтаксис лексем языка программирования всегда можно описать грамматикой без ε-правил, так как не существует пустых лексем.

Существуют способы преобразования регулярной грамматики в эквивалентный конечный автомат и наоборот — конечного автомата в регулярную грамматику.

Алгоритм построения НКА по праволинейной грамматике

1. Множество вершин НКА состоит из нетерминалов грамматики и еще одной новой вершины *F*, которая объявляется заключительной.
2. Каждому правилу вида $A \to aB$ в автомате соответствует дуга из вершины A в вершину B, помеченная символом a: $A \xrightarrow{a} B$. Каждому правилу вида $A \to a$ соответствует дуга $A \xrightarrow{a} F$. Других дуг в автомате нет.
3. Начальной вершиной автомата является вершина, соответствующая начальному символу грамматики. Вершина B является заключительной, если в грамматике есть правило $B \to \varepsilon$. Кроме того, заключительной является F, построенная на шаге 1.

Пример 2. G_2: $S \to aS \mid aA \mid \varepsilon$
$A \to b \mid bA$



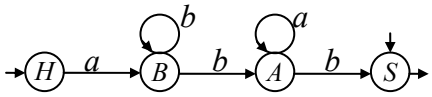
$L(G)=\{a^n b^m \mid n \geq 1, m \geq 1\} \cup \{a^k \mid k \geq 0\} = L\left(a^*(\varepsilon + ab^*b)\right)$

Алгоритм построения НКА по леволинейной грамматике

1. Множество вершин НКА состоит из нетерминалов грамматики и еще одной новой вершины H, которая объявляется начальной.
2. Каждому правилу вида $A \to Ba$ в автомате соответствует дуга из вершины B в вершину A, помеченная символом a: $B \xrightarrow{a} A$. Каждому правилу вида $A \to a$ соответствует дуга $H \xrightarrow{a} A$. Других дуг нет.
3. Заключительной вершиной автомата является вершина, соответствующая начальному символу грамматики. Вершина B является начальной, если в грамматике есть правило $B \to \varepsilon$. Кроме того, начальной является вершина H, построенная на шаге 1.

Пример 3. G_3: $S \to Ab \mid \varepsilon$
$A \to Aa \mid Bb$
$B \to Bb \mid a$

$\mathcal{A}(G_3)$:



$L(G_3) = \{ab^n a^m b \mid n \geq 1, m \geq 0\} \cup \{\varepsilon\} = L(ab^* ba^* b + \varepsilon)$

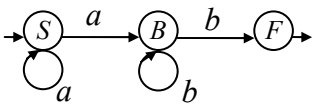
Для любого НКА, имеющего несколько начальных вершин, можно построить эквивалентный ему НКА с единственной начальной вершиной. По такому автомату легко построить эквивалентную праволинейную грамматику.

Алгоритм построения праволинейной грамматики по НКА с единственной начальной вершиной

1. Нетерминалами грамматики будут вершины автомата, терминалами — пометки дуг.
2. Для каждой дуги вида $A \xrightarrow{a} B$ в грамматику добавляется правило $A \to aB$. Для каждой заключительной вершины B в грамматику добавляется правило $B \to \varepsilon$.
3. Начальным символом является нетерминал, соответствующий начальной вершине.
4. К построенной по пунктам 1—3 грамматике применить алгоритм устранения ε-правил, затем — алгоритм удаления бесполезных символов (приведения грамматики).

Пример 4.

$\mathcal{A}_4$:



После шага 3: $S \to aS \mid aB$
$B \to bB \mid aF$
$F \to \varepsilon$

$G(\mathcal{A}_4)$: $S \to aS \mid aB$
$B \to bB \mid a$

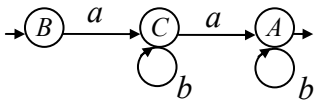
Для построения леволинейной грамматики по НКА, НКА следует сначала преобразовать в эквивалентный автомат с единственной заключительной вершиной (такое преобразование всегда возможно).

Алгоритм построения леволинейной грамматики по НКА с единственным заключительным состоянием

1. Нетерминалами грамматики будут вершины автомата, терминалами — пометки дуг.
2. Для каждой дуги $A \xrightarrow{a} B$ в грамматику добавляется правило $B \to Aa$. Для каждой начальной вершины B в грамматику добавляется правило $B \to \varepsilon$.
3. Начальным символом будет нетерминал, соответствующий заключительной вершине.

4. К построенной по пунктам 1—3 грамматике применить алгоритм устранения ε-правил, затем — алгоритм приведения грамматики.

Пример 5.

$\mathcal{A}_5$: 

После шага 3: $A \to Ca \mid Ab$
$C \to Cb \mid Ba$
$B \to \varepsilon$

$G(\mathcal{A}_5)$: $A \to Ca \mid Ab$
$C \to Cb \mid a$

Определение. Конечный автомат называется детерминированным конечным автоматом (ДКА), если он имеет единственное начальное состояние, и любые две дуги, исходящие из одной и той же вершины имеют различные пометки.

Все автоматы в примерах 1—4 не являются детерминированными.

Для каждого НКА можно построить эквивалентный ему детерминированный конечный автомат.

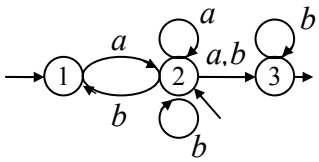
Алгоритм построения ДКА по НКА

Вход: $\mathcal{A}' = (K', \Sigma, \delta', I, F)$ — НКА.

Выход: $\mathcal{A} = (K, \Sigma, \delta, InitState, FinalStates)$ — ДКА.

Метод: Вершинами (состояниями) автомата $\mathcal{A}$ будут подмножества множества K' автомата $\mathcal{A}'$. *CurState* и *NewState* — вспомогательные переменные для хранения таких подмножеств. Сам алгоритм запишем в паскалеподобном стиле. Фигурные скобки означают конструкторы множеств.

begin $InitState := \{s \mid s \in I\}$; $K := \{InitState\}$; $\delta := \varnothing$;
  while (*в K есть нерассмотренный элемент*)
  begin
    CurState := *нерассмотренный элемент из K*;
    for (*каждого* $a \in \Sigma$)
    begin
      $NewState := \{q \mid (p \xrightarrow{a} q) \in \delta, p \in CurState\}$;
      $K := K \cup \{NewState\}$;
      $\delta := \delta \cup \{(CurState \xrightarrow{a} NewState)\}$;
    end
  end;
  FinalStates := $\{P \in K \mid$ *существует* $q \in P$: $q \in F\}$
end.

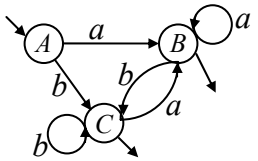
Пример 6. $\mathcal{A}_6'$: 

Процесс построения удобно изобразить в виде таблицы, начав с состояния {1, 2}

символ / состояние	*a*	*b*
{1, 2}	{2, 3}	{1, 2, 3}
{2, 3}	{2, 3}	{1, 2, 3}
{1, 2, 3}	{2, 3}	{1, 2, 3}

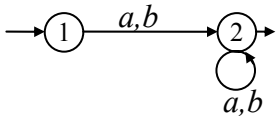
Обозначим состояние {1, 2} через *A*, {2, 3} — *B*, {1, 2, 3} — *C*.

С учетом переобозначений построим по таблице ДКА $\mathcal{A}_6$:



$L(\mathcal{A}_6)=\{a, b\}^+$

Можно заметить, что язык $L = \{a, b\}^+$, допускаемый автоматом $\mathcal{A}_6$, допускается также ДКА $\mathcal{A}_6''$, имеющим только два состояния.

$\mathcal{A}_6''$: 

Существует алгоритм, позволяющий по любому ДКА построить эквивалентный ДКА с минимальным числом состояний.

Работу детерминированного конечного автомата можно промоделировать с помощью компьютерной программы. К такого рода программам относятся, например, лексические анализаторы. В основе анализатора лежит автомат, задающий синтаксис лексем некоторого языка. Заметим, что в детерминированном автомате $\mathcal{A} = (K, \Sigma, \delta, I, F)$ δ можно интерпретировать как частичную функцию из $K \times \Sigma$ в K. Доопределим δ на всем множестве $K \times \Sigma$. Для этого добавим в K ещё одно состояние *ERR* (переход автомата в это состояние будет означать, что входная цепочка не допускается (т. е. ошибка в цепочке)) и для каждой пары $(p, a) \in K \times \Sigma$ такой, что $\delta(p, a)$ не определена, положим $\delta(p, a) = ERR$.

Алгоритм моделирования работы ДКА

Вход: ДКА $\mathcal{A} = (K, \Sigma, \delta, I, F)$ и цепочка $x\perp$, где $x \in \Sigma^*$, $\perp \notin \Sigma$ — маркер конца цепочки.

Выход: «Да», если $x \in L(\mathcal{A})$, иначе — «Нет».

Метод: Введем переменные *St* для хранения текущего состояния автомата и *c* для хранения очередного считанного символа входной цепочки *x*.

```
begin
        c := первый символ цепочки x;
        St := I; {начальное состояние}
        while ( St ≠ ERR and c ≠ '⊥' )
        begin
                St := δ (St, c);
                c := очередной символ
        end;
        if St ∈ F then
                write ('Да')
        else
                write ('Нет')
end;
```

Покажем, что класс регулярных языков является собственным подклассом класса контекстно-свободных языков (т.е. языков, порождаемых грамматиками типа 2). Для этого достаточно найти КС-язык, который не является регулярным.

Утверждение. *Контекстно-свободный язык* $L = \{a^n b^n \mid n \geq 1\}$ *нерегулярен.*

Доказательство (от противного). Предположим, что язык L регулярен. Тогда существует конечный автомат $A = (K, \Sigma, \delta, I, F)$, допускающий язык L: $L(A) = L$. (Напомним, что любой регулярный язык может быть задан конечным автоматом). Пусть число состояний автомата A равно k ($k > 0$). Рассмотрим цепочку $a^k b^k \in L$. Так как $L = L(A)$, $a^k b^k \in L(A)$. Это означает, что в автомате A есть успешный путь T с пометкой $a^k b^k$:

$$p_1 \xrightarrow{a} p_2 \xrightarrow{a} \cdots \xrightarrow{a} p_k \xrightarrow{a} p_{k+1} \xrightarrow{b} p_{k+2} \xrightarrow{b} \cdots \xrightarrow{b} p_{2k} \xrightarrow{b} p_{2k+1},$$

где $p_i \in K$ *для* $i = 1, \ldots, 2k+1$. Так как в автомате A всего k состояний, среди p_1, p_2, …, p_{k+1} найдутся хотя бы два одинаковых. Иными словами, существуют i, j, $1 \leq i < j \leq k$, такие что $p_i = p_j$. Таким образом, участок $p_i \xrightarrow{a} \cdots \xrightarrow{a} p_j$ пути T является циклом. Пусть длина этого цикла равна l ($l > 0$, так как цикл — это непустой путь). Рассмотрим успешный путь T', который отличается от T тем, что циклический участок $p_i \xrightarrow{a} \cdots \xrightarrow{a} p_j$ присутствует в нем дважды:

$$p_1 \xrightarrow{a} \cdots p_i \xrightarrow{a} \cdots \xrightarrow{a} (p_i = p_j) \xrightarrow{a} \cdots \xrightarrow{a} p_j \xrightarrow{a} \cdots p_{k+1} \xrightarrow{b} \cdots p_{2k+1}$$

Пометкой пути T' является цепочка $a^{k+l} b^k$. Поскольку T' — успешный путь, $a^{k+l} b^k \in L(A)$. Так как $a^{k+l} b^k \notin L$, получаем, что $L \neq L(A)$. Это противоречит равенству $L = L(A)$. Следовательно, предположение о том, что L регулярен, неверно □