Введение в теорию алгоритмов (Поляков В.Н., Скорубский В.И. Основы теории алгоритмов)

История

Современное формальное определение **алгоритма** было дано в 30 - 50-х гг. XX века в работах А. Тьюринга[1], Э. Поста[2], А. Чёрча[3], Н. Винера[4], А. А. Маркова[5].

Само слово «алгоритм» происходит от имени учёного Абу Абдуллах Мухаммеда ибн Муса аль-Хорезми. Около 825 г. он написал сочинение, в котором впервые дал описание придуманной в Индии позиционной десятичной системы счисления. К сожалению, арабский оригинал книги не сохранился. Аль-Хорезми сформулировал правила вычислений в новой системе и, вероятно, впервые использовал цифру 0 для обозначения пропущенной позиции в записи числа (её индийское название арабы перевели как as-sifr или просто sifr, отсюда такие слова, как «цифра» и «шифр»). Приблизительно в это же время индийские цифры начали

Определения

Алгоритм (процедура) – решение задач в виде точных последовательно выполняемых предписаний.

Это интуитивное определение сопровождается описанием интуитивных **свойств (признаков)** алгоритмов: эффективность, определенность, конечность [1].

Эффективность – возможность исполнения предписаний за конечное время.

Функция алгоритмически эффективно вычислима, если существует механическая процедура, следуя которой для конкретных значений ее аргументов можно найти значение этой функции [3].

Определенность – возможность точного математического определения или формального описания содержания команд и последовательности их применения в этой процедуре.

Конечность – выполнение алгоритма при конкретных исходных данных за конечное число шагов.

Модели алгоритмов

Для демонстрации алгоритмов в теории используются алгоритмические преобразования слов и предложений формального языка [1, 3].

В **формальных** описаниях алгоритм конструктивно связывают с понятием **машины**, предназначенной для автоматизированных преобразований символьной информации.

Для автоматических вычислений разрабатываются модели алгоритмов распознавания языков и машина, работающая с этими моделями. Таким образом, соединяют математическое, формальное определение алгоритма и конструктивное, позволяющее реализовать модели вычислительными машинами.

Модели алгоритмических преобразований

Общая Теория алгоритмов занимается проблемой эффективной **вычислимости.** Разработано несколько формальных определений алгоритма, в которых эффективность и конечность вычислений может быть определена количественно – числом элементарных шагов и объемом требуемой памяти.

Подобными моделями алгоритмических преобразований символьной информации являются:

- конечные автоматы;
- машина Тьюринга;
- машина Поста;
- ассоциативное исчисление или нормальные алгоритмы Маркова;
- рекурсивные функции.

Некоторые из этих моделей лежат в основе методов программирования и используются в алгоритмических языках.

Формализация

Для того, чтобы представить формальное описание алгоритма необходимо формальное описание решаемой задачи. В большинстве случаев описание задачи неформальное (вербальное) и переход к алгоритму неформальный и требует верификации и тестирования и многократных итераций для приближения к решению.

Верификация (от лат. *verus* – «истинный» и *facere* – «делать») – проверка, способ подтверждения каких-либо теоретических положений, алгоритмов, программ и процедур путем их сопоставления с опытными (эталонными или эмпирическими) данными, алгоритмами и программами [4].

Тестирование применяется для определения соответствия предмета испытания заданным спецификациям [4].

КА как модель алгоритма

К сожалению, теория алгоритмов не дает и не может дать как универсального, формального способа описания задачи, так и ее алгоритмического решения. Однако ценность теории состоит в том, что она дает примеры таких описаний и дает определение алгоритмически неразрешимых задач.

Один из примеров представлен регулярным языком, и задача формулируется как разработка алгоритма для распознавания принадлежности любого предложения к конкретному регулярному языку. Доказывается, что регулярный язык может быть формально преобразован в **модель алгоритма** решения этой задачи за конечное число шагов. Этой моделью является **конечный автомат.**

Регулярные выражения

Алфавит языка обозначается как конечное множество символов. Например: **Σ={a,b,c,d}, Σ={0,1}**.

Символ и цепочка символов образуют слово – a, b, 0, abbcd, 0111000.

Пустое слово (**e**) не содержит символов.

Множество слов **S=**{a, ab, aaa, bc} в алфавите **Σ** называют языком L(**Σ**).

Языки S=L(**Σ**) могут содержать неограниченное число слов, для их определения используют различные формальные правила. В простейшем случае это **алгебраическая формула**, которая содержит операции формирования слов из символов алфавита и ранее полученных слов.

Рассмотрим следующие операции формирования новых множеств из существующих множеств слов:

1) Символы алфавита могут соединяться **конкатенацией** (сцепление, соединение) в цепочки символов-слов, которые соединяются в новые слова.

Конкатенация двух слов **x|y** обозначает, что к слову x справа приписано слово y или $x|y=xy$, причем $xy \neq yx$.

Произведение $S1|S2=S1S2$ множеств слов S1 и S2 - это множество всех различимых слов, построенных конкатенацией соответствующих слов из S1 и S2.

Если S1={a, aa, ba}, S2={e, bb, ab}, то S1S2={a, aa, ba, abb, aabb, baab, …}.

Для конкатенации выполняется ассоциативность, но коммутативность и идемпотентность не выполняются:

$$S1S2 \neq S2S1;$$

$$SS \neq S.$$

Если S1={a, aa, ba}, S2={e, bb, ab}, то S1S2={a, aa, ba, abb, aabb, baab, …}.

Для конкатенации выполняется ассоциативность, но коммутативность и идемпотентность не выполняются:

$$S1S2 \neq S2S1;$$

$$SS \neq S.$$

2) **объединение** $(S1 \cup S2)$ или $(S1+S2)$ множеств

S1={a, aa, ba}, S2={e, bb, ab}, $S1\cup S2$={a, aa, ba, e, bb, ab}.

Для операции объединения выполняются следующие законы:

Коммутативность объединения $S1\cup S2=S2\cup S1$.

Идемпотентность объединения $S \cup S=S$.

Ассоциативность объединения $S1 \cup (S2 \cup S3) =(S1 \cup S2) \cup S3$.

Дистрибутивность конкатенации (умножения) и объединения $S1(S2 \cup S3) = S1S2 \cup S1S3$.

3) **Итерация множества** {S}* состоит из пустого слова и всех слов вида S^0=e, S^1=S, S^2=SS, S^3=SSS.

Формулы, содержащие эти операции с множествами слов, называют **регулярными выражениями**.

Ассоциативность итерации $S1 * (S2 * S3) =(S1 * S2)* S3$.

Дистрибутивность объединения с итерацией

$S1 *(S2 \cup S3) = S1*S2 \cup S1*S3$.

Регулярные языки

Регулярные выражения допускают формальные алгебраические преобразования,

Языки, определяемые регулярными выражениями, называются **регулярными языками**, а множество слов - **регулярными множествами**.

Пример 1.1.

Регулярные выражения регулярного языка в алфавите $\mathbf{\Sigma} = \{0,1\}$

$(0 \cup (1(0)^*)) = 0 \cup 10^*$;

$(0 \cup 1)^* = (0^* \cup 1^*)^*$;

$(0 \cup 1)^*011$ - все слова из 0 и 1, заканчивающиеся на 011;

$(a \cup b)(a \cup +b)^* = (a \cup b)(a^* \cup b^*)^*$ - слова, начинающиеся с a или b;

$(00 \cup 11)^*((01 \cup 10)(00 \cup 11)^*(01 \cup 10)(00 \cup 11)^*)^*$ - все слова, содержащие четное число 0 и 1.

Пример 1.2.

Регулярное выражение, определяющее правильное арифметическое выражение [5].

Входной алфавит **Σ={i, +, -}**, где

i – идентификатор;

(+, -) – знаки арифметических операций.

Примеры правильных арифметических выражений

i , - i , i + i , i - i , - i - i , i + i - i, ...

Обозначим знаки арифметических действий буквами p = (+), m=(-). Тогда соответствующие правильные (регулярные) арифметические выражения имеют вид **i, mi, ipi, imi, mimi, ipimi,** ... и регулярное выражение, определяющее регулярный язык,

L(M)=(mi + i)((p + m)i)*.

Утверждение

Утверждение. Для каждого регулярного множества (языка $L(\Sigma)$) можно построить, по крайней мере, одно регулярное выражение, но для каждого регулярного выражения существует только одно регулярное множество.

Читающие автоматы

Конечные автоматы - модель алгоритма распознавания предложений регулярного языка

Определение. Конечный автомат определяется символами

$$M=(Q, \Sigma, \delta, q0, F), \text{ где}$$

Q={q0, q1, …, qn} – конечное множество состояний;

Σ={a, b, c, ...} – входной алфавит (конечное множество);

δ: Q* Σ → {Pj } – функция переходов, Pj - подмножество Q.

Конечное множество значений для этого функционального отношения может быть определено перечислением в **таблице переходов:**

Q	Σ	Pj
qi	a	qj

q0 - начальное состояние;

F - множество заключительных состояний.

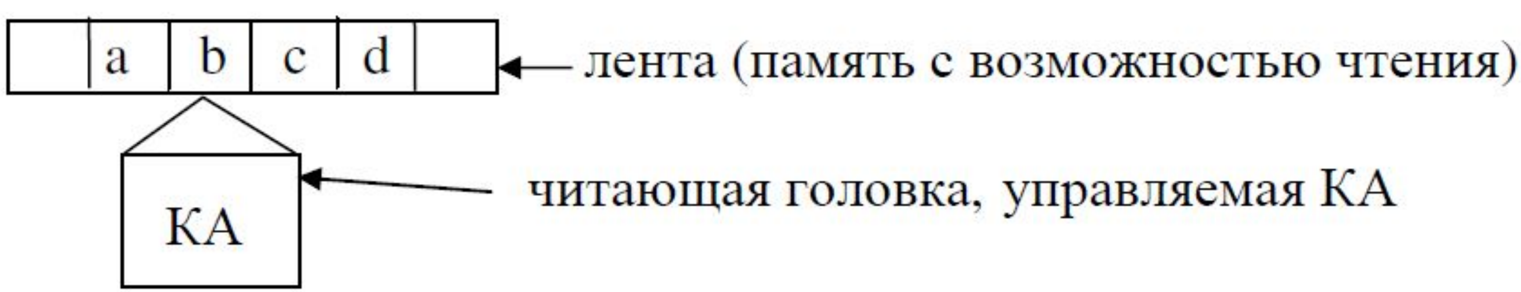
Конечный автомат называется **недетерминированным** (НДКА), если Pj содержит более одного состояния.

КА называется **детерминированным** (ДКА), если Pj содержит не более одного состояния.

КА **полностью определенный**, если Pj в детерминированном автомате не пустое. Если есть пустые элементы множества Pj, то автомат **частично определен.**

Работа КА или выполнение алгоритма распознавания слов регулярного языка могут быть представлены последовательностью шагов, которые определяются текущим состояние Q, входным символом **Σ** и следующим состоянием Pj.

Используется конструктивное описание принципа работы КА как машины М, имеющей следующую организацию:



КА читает входной символ в текущем состоянии $qi \in Q$, переходит в следующее состояние $qj \in Q$ и сдвигает читающую головку к следующему символу.

Автомат допускает входное слово, если приходит в заключительное состояние из F, последовательно считывая символы из памяти и переходя в следующие состояния в соответствии с таблицей переходов. При этом входное слово исчерпано и автомат останавливается.

Конфигурация КА $k=(q, \omega)$, где q-текущее состояние КА, ω - непрочитанная цепочка символов слова на ленте, включая символ под читающей головкой.

$k = (q, \omega)$ текущая конфигурация;

$k_0 = (q_0, \omega 0)$ начальная конфигурация;

$k_f = (q, e)$, где $q \in F$, - заключительная конфигурация и (e) – символ, обозначающий конец строки.

Шаг алгоритма - переход из одной конфигурации КА в другую

$Ki \rightarrow Kj$ или $(qi, \omega i) \rightarrow (qj, \omega j)$.

Функция переходов, заданная в табличной форме, может быть представлена графом переходов $G=(Q, R)$, где Q - вершины графа, R - бинарное отношение между парой вершин, которое представлено множеством дуг (qi, qj).

$(qi, qj) \in Q*Q$, если существует символ $\mathbf{a} \in \mathbf{\Sigma}$ и $\boldsymbol{\delta}(qi, a) = qj$.

На дугах графа (qi, qj) отмечаются соответствующие символы алфавита.

ДКА и НДКА

Различают детерминированные (ДКА) и недетерминированные (НДКА) конечные автоматы.

КА называется *недетерминированным* (НДКА), если в диаграмме его состояний из одной вершины исходит несколько дуг с одинаковыми символами. Если таких вершин нет, то это ДКА.

Пример 2.1.

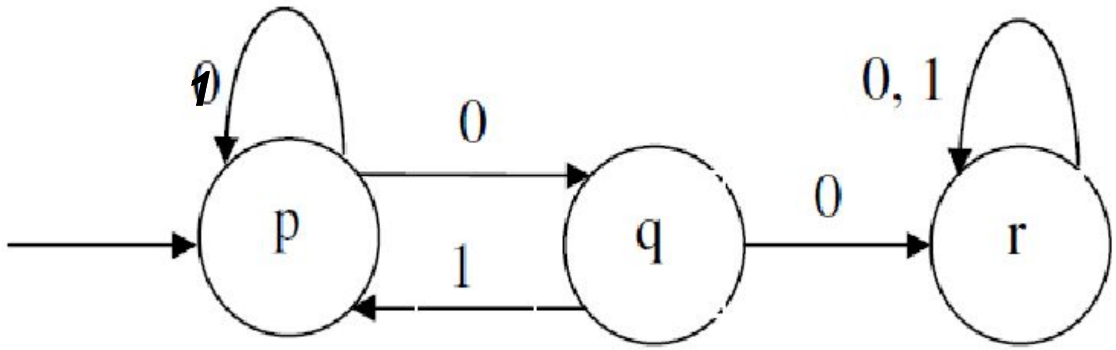


Рис.2.2. Детерминированный читающий КА

Для ДКА, приведенного на рис. 2.2 состояния Q={p, q, r}, входной алфавит **Σ**={0,1}, начальное состояние p, конечное - r.

Таблица переходов ДКА

Q	Σ	
	0	1
p	q	p
q	r	p
r	r	r

Исполнение алгоритма это последовательность шагов, в которых изменяется конфигурация КА:

(p, 01001) → (q, 1001) → (p, 001) → (q, 01) → (r, 1) → (r, e);

(p, 01001)- начальная конфигурация;

(r, e) – конечная конфигурация.

В результате применения слова 01001 в начальном состоянии **p** автомат переходит в следующее состояние **q** и следующее значение цепочки символов на входе 1001.

Автомат M допускает слово ω0 , если существует

(q0, ω0) →*(qf, e),

где →*( ), обозначает **транзитивное замыкание** и существует путь, соединяющий q0 и qf для входного слова ω0.

Язык L(M), определяемый (распознаваемый, допускаемый) автоматом M включает множество всех слов, допускаемых M.

Преобразование регулярного выражения в КА

Утверждение. Язык L является регулярным тогда и только тогда, когда он определяется КА. Для любого регулярного языка, представленного регулярным выражением, можно построить КА - распознаватель слов, допустимых языком.

Метод Ямады преобразования L(M) → M позволяет формально выполнить преобразование [5, 6]:

1) Разметка состояний устанавливает позиции символов в регулярном выражении (п.1, пример 1.2)

$$L(M)=(mi + i)((p + m) i)^*$$

0 12 3 4 5 6

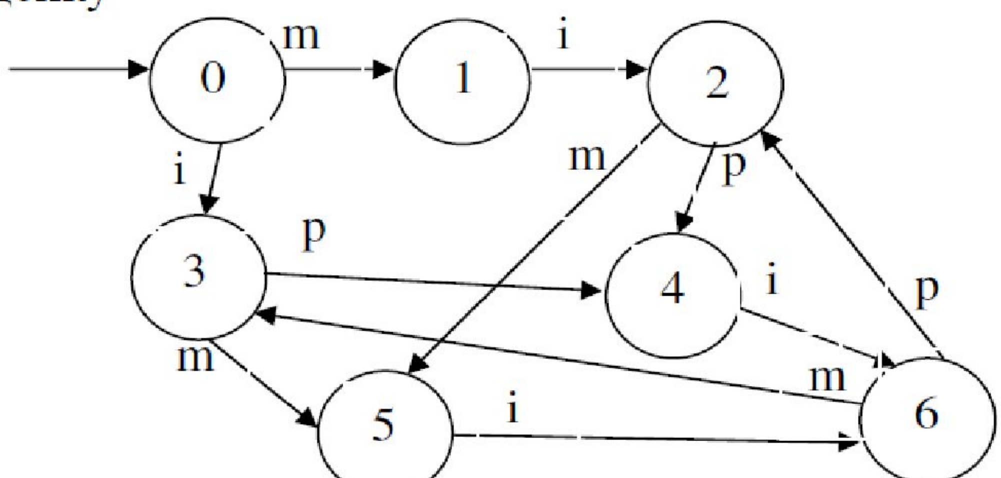
2) Рассматриваются 2^6+1 подмножеств мест предполагаемых состояний и формируются по регулярному выражению возможные переходы между состояниями. Очевидная избыточность состояний устраняется, так как в некоторые состояния переходы отсутствуют.

Общие оценки числа состояний и переходов КА:

- число переходов (ребер графа) КА равно (n+1), где (n) число букв в регулярном выражении (в данном примере n=7 состояний);

- в полностью определенном КА нижнюю границу числа состояний (m) определяем из условия **m*s=n+1**, где s - число символов входного алфавита (в данном случае 3m=7 и m=]7/3[=3);

- если **m*s>n+1**, то автомат частично определенный и верхняя граница **m ≤ n+1** – количество позиций в регулярном выражении (в данном примере m=7). Для рассматриваемого примера можно построить КА, используя верхнюю оценку



Утверждение. Для каждого регулярного множества существует определяющий его КА с минимальным числом состояний.

Слово *различает* состояния qi и qj, если

((qi, ω) →*(qm, e) (qj, ω) →* (qn, e) и **одно** из состояний qm или qn принадлежит F,

Состояния k-*неразличимы,* если они не различимы цепочкой длины k.

Состояния не различимы (*эквивалентны* qi=qj), если они не различимы при k → ∞.

В автомате рис.2.3 каждую пару эквивалентных состояний {2,3}, {4,5} заменяем одним состоянием {2,3}→ 2, {4,5}→3,

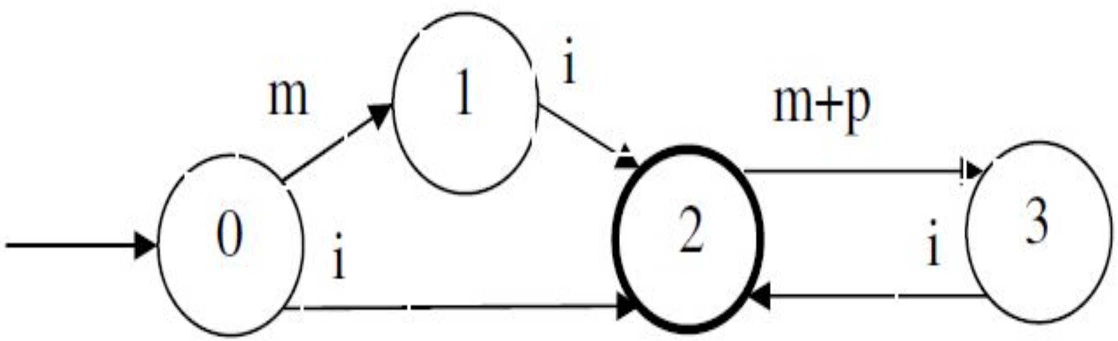


Рис.2.4. ДКА распознавания L(M)

Рассмотренная методика преобразования может быть упрощена с учетом очевидных алгебраических свойств операций регулярного выражения:

- последовательности символов в скобках с операцией (+) имеют общее начальное состояние перед открывающей скобкой и конечное после закрывающей скобки;
- итерация обозначает цикл с произвольным числом повторений, при пустом числе циклов сохраняется состояние, с которым входим в цикл;
- в конкатенации нескольких символов различимые состояния заменяют конкатенацию между парой символов;

$$L(M)=(mi + i)(\ (p + m)i\ )^*$$

0 1 2 3 2

Состояния размешаем в найденные для них позиции, исключая повторения:

$$L(M)=(mi + i)(\ (p + m)i\ )^* = 0(m1i + i)2((p + m)3i2)^*$$

0 1 2 3 2

Заменяем линейную помеченную строку графом КА рис.2.4. Начальное состояние – 0. Конечное состояние – 2.

В общем случае регулярное выражение может быть преобразовано в недетерминированный КА (НДКА).

Пример 2.2.

$$L(M)=aa^*bd^* + ad^* = (aa^*b + a)d^* =(0a1(a1)^*b + 0a)2(d2)^*$$

0 1 2

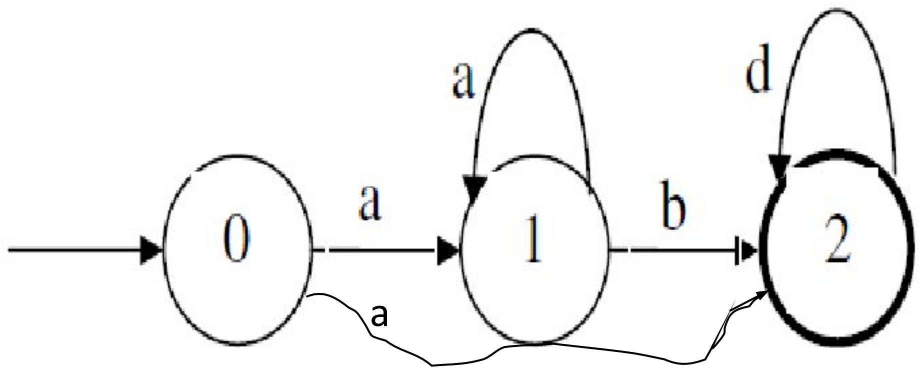


Рис.2.5. НДКА распознавания L(M)

Утверждение. Два автомата M и M' *эквивалентны*, если допускают один и тот же язык L(M) = L(M').

Утверждение. Для НДКА M можно построить эквивалентный ДКА M'.

Преобразование по методике [5] представим следующей процедурой:

- на i-ом шаге множество состояний ДКА обозначим Qi;
- Q_0 обозначает множество состояний в НДКА;
- находим различимые подмножества следующих состояний для всех условий, применяемых к Q_i, включаем их в Q_{i+1}. Итерация повторяется, пока $Q_i \neq Q_{i+1}$.

Лемма. Число состояний в ДКА не превышает $\mathbf{2^n-1}$, где n – число состояний в НДКА.

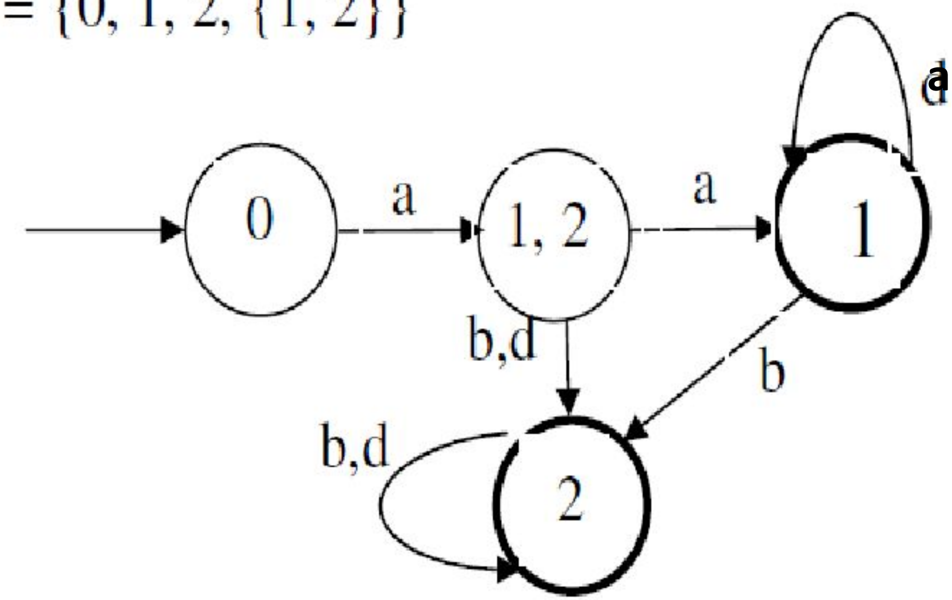
Число $\mathbf{2^n-1}$ – количество собственных подмножеств для множества, состоящего из n различимых элементов.

Следовательно, процедура конечна.

Пример 2.4. для НДКА рис.2.5.

$Q_0 = \{0, 1, 2\}$

$Q_1 = \{0, 1, 2, \{1, 2\}\}$



	a	b	d
0	1,2	-	-
1,2	1	2	2
1	1	2	-
2	-	-	2

Рис.2.6. ДКА получен из НДКА

Эквивалентный детерминированный автомат (рис. 6) содержит 4 состояния и определен тем же регулярным выражением L(M') = L(M).

Преобразование КА в регулярное выражение

Метод исключения состояний [5]:

Для допускающего (финального) состояния исключить промежуточное состояние (кроме начального q0), следующим образом:

пусть qi – предшествующее состояние, qs – промежуточное и qj – следующее, таким образом:

1) над каждой дугой (qi, qs) записать регулярное выражение Qis;

2) на дуге (qs, pj) – выражение Qsj,

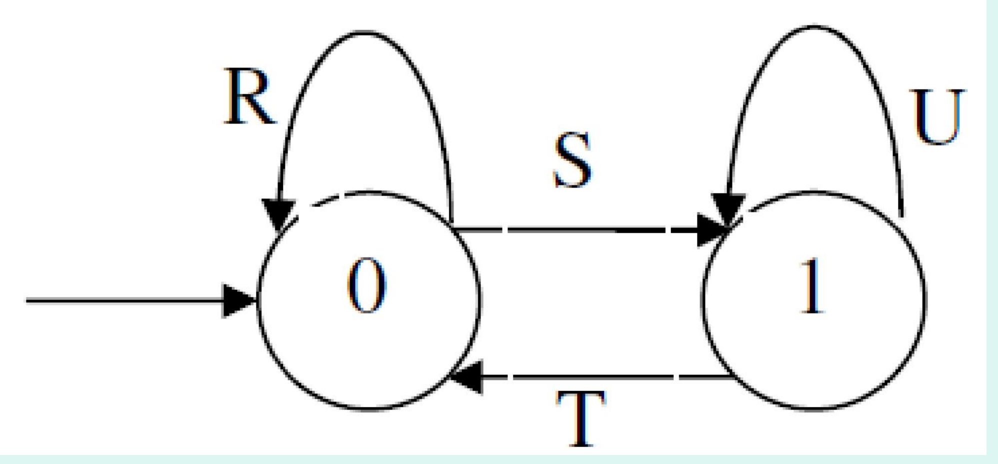
3) петля в s обозначается регулярным выражением S*;

5) дугам (qi, qj) после исключения промежуточного состояния qs приписываются регулярные выражения Qis S*Qsj.

После удаления всех промежуточных вершин остаются начальное состояние и финальные. Финальные состояния объединяются суммированием соответствующих регулярных выражений. В частном случае, если $q0 \in F$, останется одно состояние q0 и регулярное выражение приписано дуге.

Когда останутся два состояния - начальное и финальное, то промежуточное регулярное выражение может быть записано в виде:

(R + SU*T)*SU*.



Пример

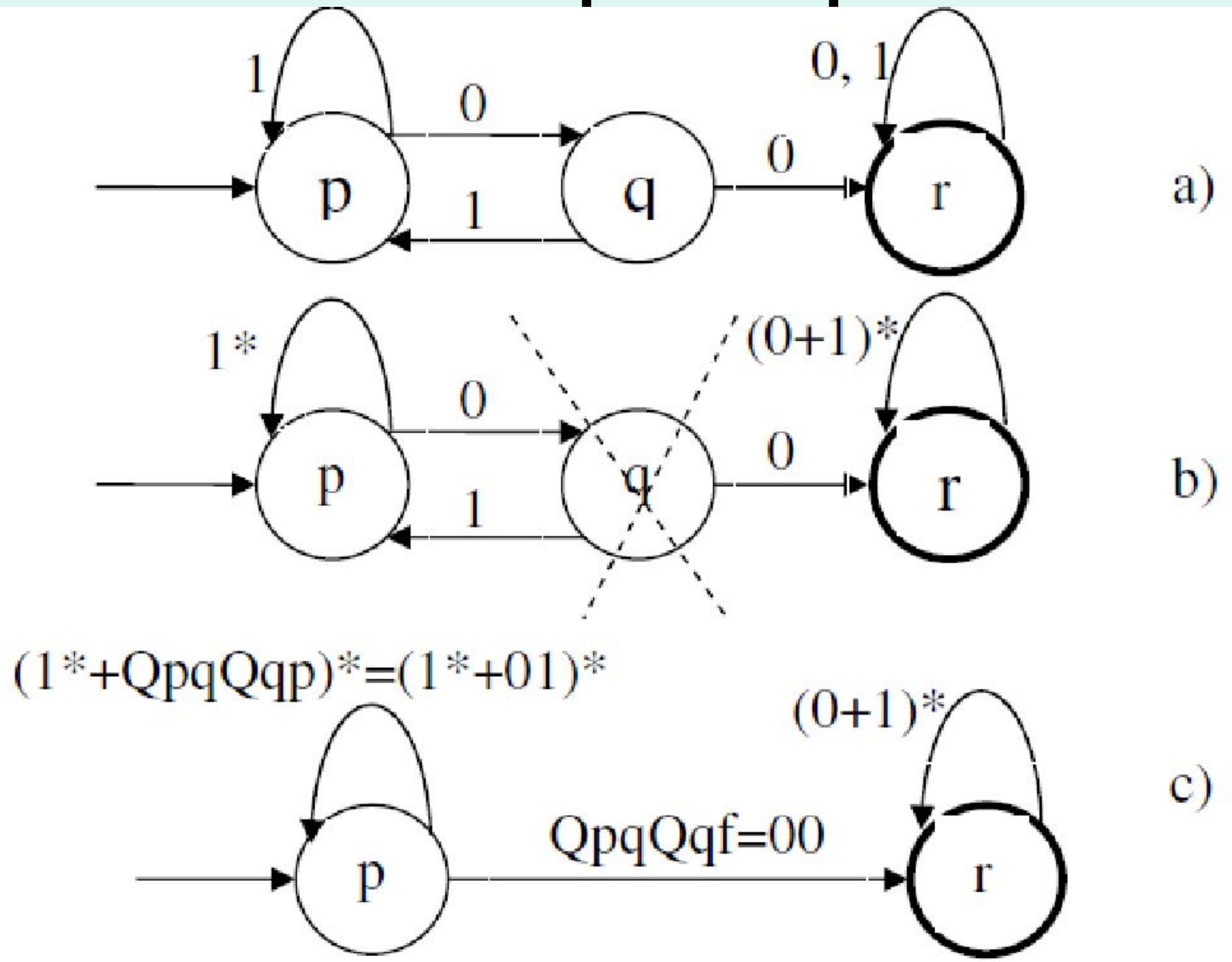


Рис.2.8. Три стадии преобразование КА в регулярное выражение: исходный КА, b) исключение состояния, c) свертка для двух состояний

Завершающий шаг – соединение (конкатенация) выражений для начального и финального состояний:

R=(1* + QpqQqp)*=(1*+01)*

U=(0 + 1)*

S=QpqQqf=00

T отсутствует

(R + SU*T)*SU*= (1*+01)* 00(0 + 1)*

L(M)=(1*+01)*00(0+1)*

0 0 10 23