Глава 2 Регулярные языки

2.1 Регулярные множества и регулярные выражения

2.1.1 Определение и свойства регулярных выражений

Проблема генерирования бесконечных языков посредством конечных описаний решается различными способами. Одним из них является использование регулярных выражений для порождения бесконечных цепочек языка.

Регулярное множество и *регулярное выражение* для некоторого алфавита V определяется рекурсивно следующим образом:

- 0 – регулярное выражение, обозначает $\varnothing$ регулярное множество;
- λ – регулярное выражение, обозначает регулярное множество $\{\lambda\}$;
- $\forall a \in V$ a – регулярное выражение, обозначает регулярное множество {a};
- если p и q – произвольные регулярные выражения, обозначающие регулярные множества P и Q, то p+q, pq, p* – регулярные выражения, обозначающие соответственно регулярные множества $P \cup Q$, PQ, P*;
- ничто другое регулярным выражением и регулярным множеством не является.

Иными словами, *регулярные множества* – это цепочки символов над заданным алфавитом, построенные с использованием операций объединения, конкатенации и замыкания.

Все регулярные языки представляют собой регулярные множества.

Два регулярных выражения α и β *эквивалентны*: $\alpha = \beta$, если они обозначают одно и то же множество.

Каждое регулярное выражение обозначает одно и только одно регулярное множество, но для одного регулярного множества может существовать сколько угодно задающих его регулярных выражений.

При записи регулярных выражений используются круглые скобки, как для обычных арифметических выражений. При отсутствии скобок операции выполняются слева направо с учетом приоритета. Наивысшим приоритетом обладает операция итерации, затем конкатенации, потом – объединение множеств.

Например, язык, представляющий собой множество всех цепочек из нулей и единиц произвольной длины, может быть описан эквивалентными регулярными выражениями α_1 и α_2: $\alpha_1 = (0+1)^*$, $\alpha_2 = (0^*1^*)^*$. $\alpha_1 = \alpha_2$.

Пусть α, β, γ – регулярные выражения. Тогда свойства регулярных выражений можно записать в виде следующих формул:

1. $\alpha+\beta=\beta+\alpha$
6. $\alpha+\alpha=\alpha$
11. $0^*=\lambda$

2. $\alpha+(\beta+\gamma)=(\alpha+\beta)+\gamma$
3. $\alpha(\beta\gamma)=(\alpha\beta)\gamma$
4. $\alpha(\beta+\gamma)=\alpha\beta+\alpha\gamma$
5. $(\beta+\gamma)\alpha=\beta\alpha+\gamma\alpha$
7. $\alpha+\alpha^*=\alpha^*$
8. $\lambda+\alpha^*=\alpha^*+\lambda=\alpha^*$
9. $\lambda+\alpha\alpha^*=\lambda+\alpha^*\alpha=\alpha^*$
10. $(\alpha^*)^*=\alpha^*$
12. $0\alpha=\alpha0=0$
13. $0+\alpha=\alpha+0=\alpha$
14. $\lambda\alpha=\alpha\lambda=\alpha$

Все эти свойства доказываются с применением аппарата теории множеств, т.к. регулярные выражения – это обозначения для соответствующих множеств.

2.1.2 Уравнения с регулярными коэффициентами

Простейшие уравнения с регулярными коэффициентами будут выглядеть следующим образом: $X=\alpha X+\beta$; $X=X\alpha+\beta$, где $\alpha,\beta\in V^*$ – регулярные выражения над алфавитом V, а $X\notin V$. Два вида записи уравнений (правосторонняя и левосторонняя запись) связаны с тем, что для регулярных выражений операция конкатенации не обладает свойством коммутативности. Обе записи равноправны.

Решениями таких уравнений будут регулярные множества. Это значит, что, если взять РМ, являющееся решением уравнения, обозначить его соответствующим РВ и подставить в уравнение, то получится тождественное равенство.

Решением первого уравнения является регулярное множество, обозначенное $\alpha^*\beta$. Если подставить его вместо переменной X в уравнение, получим $\alpha X+\beta=\alpha(\alpha^*\beta)+\beta=^{6}(\alpha\alpha^*)\beta+\beta=^{13}(\alpha\alpha^*)\beta+\lambda\beta=^{5}(\alpha\alpha^*+\lambda)\beta=^{1}\alpha^*\beta=X$.

Аналогично, для второго уравнения решение будет иметь вид $\beta\alpha^*$. Подставив его в уравнение, также получим тождество.

Указанные решения уравнений не всегда являются единственными. Например, если регулярное выражение α в первом уравнении обозначает множество, которое содержит пустую цепочку, то решением уравнения может быть любое множество, обозначенное выражением $X=\alpha^*(\beta+\gamma)$, где γ может обозначать произвольное множество (в том числе не регулярное). Однако указанные решения – наименьшие возможные для данных уравнений. Они называются *наименьшей неподвижной точкой*.

Из рассмотренных уравнений можно формировать систему уравнений с регулярными коэффициентами. Например, в правосторонней записи такая система будет иметь вид:

$$\begin{cases} X_1=\alpha_{10}+\alpha_{11}X_1+\alpha_{12}X_2+\ldots+\alpha_{1n}X_n; \\ X_2=\alpha_{20}+\alpha_{21}X_1+\alpha_{22}X_2+\ldots+\alpha_{2n}X_n; \\ \ldots \\ X_i=\alpha_{i0}+\alpha_{i1}X_1+\alpha_{i2}X_2+\ldots+\alpha_{in}X_n; \\ \ldots \\ X_n=\alpha_{n0}+\alpha_{n1}X_1+\alpha_{n2}X_2+\ldots+\alpha_{nn}X_n; \end{cases}$$

В данной системе все коэффициенты α_{ij} – регулярные выражения над алфавитом V, а переменные не входят в этот алфавит: $X_i\notin V\ \forall\ i$.

Системы уравнений с регулярными коэффициентами решаются

методом последовательных подстановок. Рассмотрим метод решения для правосторонней записи. Алгоритм работает с переменной номера шага i.

Алгоритм решения:

1. Положить i:=1.
2. Если i=n, то перейти к шагу 4, иначе записать i-е уравнение в виде: $X_i=\alpha_i X_i+\beta_i$, где $\alpha_i=\alpha_{ii}$, $\beta_i=\beta_{i0}+\beta_{i\,i+1}\,X_{i+1}+\ldots+\beta_{i\,n}\,X_n$, решить уравнение и получить $X_i=\alpha_i^*\beta_i$. Затем подставить это выражение во все уравнения для переменных $X_{i+1},\ldots, X_n$.
3. Увеличить i на единицу и вернуться к шагу 2.
4. После всех подстановок уравнение для X_n будет иметь вид: $X_n=\alpha_n X_n+\beta$, где $\alpha_n=\alpha_{nn}$, а β – регулярное выражение над алфавитом V^*, т.е. не содержит переменных X_i. Тогда можно найти решение $X_n=\alpha_n^*\beta$.
5. Уменьшить i на единицу. Если i=0, то алгоритм завершен.
6. Взять найденное решение для $X_i=\alpha_i X_i+\beta_i$, где $\alpha_i=\alpha_{ii}$, $\beta_i=\beta_{i0}+\beta_{i\,i+1}\,X_{i+1}+\ldots+\beta_{i\,n}\,X_n$ и подставить в него найденные окончательные решения для переменных $X_{i+1},\ldots, X_n$. Получим окончательное решение для X_i. Вернуться к шагу 5.

Система уравнений с регулярными коэффициентами всегда имеет решение, но оно не всегда единственно. Рассмотренный алгоритм всегда находит решение, которое является наименьшей неподвижной точкой системы уравнений.

2.1.3 Контрольные вопросы

1. Что такое регулярное множество? Какие операции допустимо использовать в регулярном выражении?
2. Какого типа язык задаётся регулярным выражением?
3. Может ли одно и то же регулярное множество задаваться разными регулярными выражениями?
4. Может ли одно и то же регулярное выражение соответствовать разным регулярным множествам?
5. Будут ли среди трёх приведённых регулярных выражений какие-то эквивалентные друг другу? $\alpha_1 = (1+0)^*$, $\alpha_2 = (0^*+1)^*$, $\alpha_3 = (0^*1^*)^*$.
6. Построить регулярные выражения для генерации следующих языков:
   1) множество всех цепочек из {0,1}*, содержащих подцепочку ’101’;
   2) множество всех цепочек из {0,1,a,b}*, начинающихся с 0 или 1;
   3) множество всех цепочек из {0,1}*, длины которых кратны трём;
   4) множество всех цепочек из {0,1,a,b}*, начинающихся с цепочки ’01’ и заканчивающихся цепочкой ’01a’; 5) множество всех цепочек из {0,1}*, содержащих четное число нулей и четное число единиц.

2.2 Конечные автоматы и грамматики

2.2.1 Автоматные грамматики

Среди регулярных грамматик можно выделить отдельный класс – *автоматные* грамматики. Они могут быть леволинейными и праволинейными.

Леволинейные автоматные грамматики G(VT, VN, P, S) могут иметь правила двух видов: $A \to Bt$ или $A \to t$, где $A, B \in VN, t \in VT$.

Праволинейные автоматные грамматики могут тоже иметь правила двух видов: $A \to tB$ или $A \to t$, где $A,B \in VN, t \in VT$.

Классы обычных регулярных грамматик и автоматных почти эквивалентны (с точностью до правила $S \to \lambda$). В общем случае в автоматных грамматиках разрешается правило вида $S \to \lambda$, где S – целевой символ грамматики, который не должен встречаться в правых частях других правил грамматики. Тогда язык, задаваемый автоматной грамматикой G, будет включать в себя пустую цепочку: $\lambda \in L(G)$. В таком случае классы регулярных и автоматных грамматик становятся эквивалентными.

Как следует из определения, в правилах автоматных грамматик в отличие от регулярных вместо цепочки терминальных символов присутствует только один терминальный символ. Любая автоматная грамматика является регулярной, обратное же справедливо не всегда.

Существует алгоритм, позволяющий преобразовать регулярную грамматику к автоматному виду. Он очень полезен, т.к. упрощает задачу построения распознавателей для регулярных языков. Кратко его идея состоит в том, что при наличии в правой части правила регулярной грамматики цепочки терминальных символов длины n необходимо ввести n–1 дополнительный нетерминальный символ и записать для каждого из них соответствующие правила, т.е. разбить стоящую справа терминальную цепочку на символы и разнести их по разным правилам.

Пример. Рассмотрим регулярную грамматику и приведём её к автоматному виду. G({0,1},{S,A},P,S), $S \to 001A$, $A \to 0A \mid 1A \mid 11$. Эта грамматика задаёт язык с алфавитом {0,1}, все цепочки которого начинаются с цепочки ‘001’, а заканчиваются цепочкой‘11’.

В первом правиле присутствует цепочка из трёх терминальных символов, значит, потребуется добавить два нетерминала. Пусть это будут B и C. Первое правило примет вид: $S \to 0B$, $B \to 0C$, $C \to 1A$. 2 правила для A останутся без изменения: $A \to 0A \mid 1A$, а третье правило $A \to 11$ тоже придётся разбить на два путём добавления ещё одного нетерминала, например, D: $A \to 1D$, $D \to 1$. Итак, теперь грамматика будет содержать 5 нетерминалов и 7 правил: G({0,1},{S,A,B,C,D},P,S), $S \to 0B$, $B \to 0C$, $C \to 1A$, $A \to 0A \mid 1A \mid 1D$, $D \to 1$.

2.2.2 Определение конечного автомата

С автоматной грамматикой тесно связано понятие конечного автомата. Автоматная грамматика позволяет сгенерировать все цепочки некоторого регулярного языка, а конечный автомат – распознать этот язык.

Конечным автоматом (КА) называют пятёрку вида: $M(Q,V,\delta,q_0,F)$, где

- Q – конечное *множество состояний УУ* автомата;
- V – *алфавит* автомата (конечное множество допустимых символов);
- δ – *функция переходов*, отображающая $Q\times V$ в множество подмножеств множества Q: $\forall a\in V,\ q\in Q\ \ \delta(q,a)=\{R, \mid R\subseteq Q\}$;
- q_0 – *начальное состояние* автомата: $q_0\in Q$;
- F – множество *конечных* (*заключительных*) состояний: $F\subseteq Q,\ F\neq\varnothing$.

КА называется *полностью определённым*, если в каждом его состоянии существует функция переходов для всех возможных входных символов: $\forall a\in V,\ \forall q\in Q\ \ \exists\delta(q,a)=\{R,\ R\subseteq Q,\ R\neq\varnothing\}$.

Работа автомата представляет собой последовательность *тактов* (или *шагов*). На каждом шаге работы автомат может остаться в том же состоянии или перейти в другое. Поведение автомата на каждом такте определяется функцией переходов, которая зависит от текущего состояния и входного символа. Если функция переходов допускает несколько переходов в следующее состояние, то КА может перейти в любое из них, и такой КА является *недетерминированным* (НКА). Конечный автомат является *детерминированным* (ДКА) если множество $\delta(q,a)$ содержит не более одного состояния $\forall a\in V,\ \forall q\in Q$. Если $\delta(q,a)$ всегда содержит ровно одно состояние, то М является *полностью определённым детерминированным* (ПО ДКА).

В начале работы автомат находится в начальном состоянии q_0. Работа автомата продолжается до тех пор, пока на его вход поступают символы входной цепочки, или автомат не придёт в конфигурацию, переход из которой не определён.

Мгновенным описанием (МО), или *конфигурацией* автомата М называется пара (q, w), где $q\in Q$ – состояние УУ, $w\in V^*$ – неиспользованная часть входной цепочки (символ, обозреваемый считывающей головкой и все символы справа от него). Тогда пара (q_0, w) *называется начальной конфигурацией* для цепочки w, а конфигурация (q, λ) является *заключительной*, или *допускающей*, если $q\in F$ (q – одно из заключительных состояний автомата).

Представим такт автомата М бинарным отношением $\vdash_M$ на множестве конфигураций. Тогда: если $\delta(q,a)$ содержит p: $p\in\delta(q,a)$, то для обозначения такта записывают $(q,aw)\vdash_M(p,w)$ для всех цепочек $w\in V^*$. Здесь $q,p\in Q$, $a\in V$, т.е. a является символом входного алфавита.

Рефлексивное и транзитивное замыкание отношения $\vdash$ обозначается

через $\vdash^*$.

Цепочка w допускается автоматом M, если $(q_0,w)\vdash^*(q,\lambda)$ для некоторого $q\in F$, т.е. получив на вход эту цепочку, автомат из начальной конфигурации может перейти в заключительную.

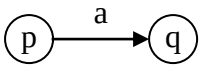
Язык, допускаемый (определяемый) автоматом M, представляет собой множество $L(M)=\{w \mid w\in V^* \text{ и } \exists\, q\in F \mid (q_0,w)\vdash^*(q,\lambda)\}$. Два автомата эквивалентны, если они задают один и тот же язык.

КА является распознавателем для регулярных языков.

Функция переходов КА может быть задана таблицей или графом переходов (диаграммой).

В *таблице* принято в качестве заголовков строк брать состояния автомата, а в качестве заголовков столбцов – символы алфавита. Если в ячейке пересечения строки ***p*** и столбца ***a*** стоит состояние ***q***, то это означает, что при прочтении символа ***a*** автомат переходит из состояния ***p*** в ***q***. Тогда в случае детерминированного полностью определённого автомата в каждой ячейке таблицы будет находиться ровно одно состояние.

Граф переходов КА – это направленный помеченный граф, вершины которого обозначены состояниями автомата, а дуги – символами входного алфавита. При этом дуга $(p,q) \mid p,q\in Q$ обозначена символом $a\in V$, если в КА определена $\delta(p,a)$ и $q\in\delta(p,a)$.



Начальное и конечное состояния на графе помечаются специальным образом, как правило, начальное – дополнительной пунктирной линией, а конечное – двойным кружком.

Для моделирования работы КА удобно, чтобы в нём были заданы все возможные переходы для всех символов входного алфавита. Если это не так, то автомат можно привести к полностью определённому виду путем ввода дополнительного состояния «ошибка», на которое следует замкнуть все неопределённые переходы, а все переходы из этого нового состояния замкнуть на него же.

Рассмотрим два примера конечных автоматов – ДКА и НКА.

1. Пусть $M=(\{p,q,r\},\{0,1\},\delta,p,\{r\})$ – ДКА, где функция переходов задана таблицей:

	вход	
состояние	0	1
p	{q}	{p}
q	{r}	{p}
r	{r}	{r}

Автомат M допускает все цепочки из $\{0,1\}^*$, содержащие два стоящих рядом нуля. Попав в состояние r, автомат из него уже не выходит.

Пусть w=’01001’.

Рассмотрим последовательность тактов для данной цепочки w.

$(p,01001)\vdash(q,1001)\vdash(p,001)\vdash(q,01)\vdash(r,1)\vdash(r,\lambda)$.

Поскольку $r\in F$, это означает, что ‘01001’$\in L(M)$.

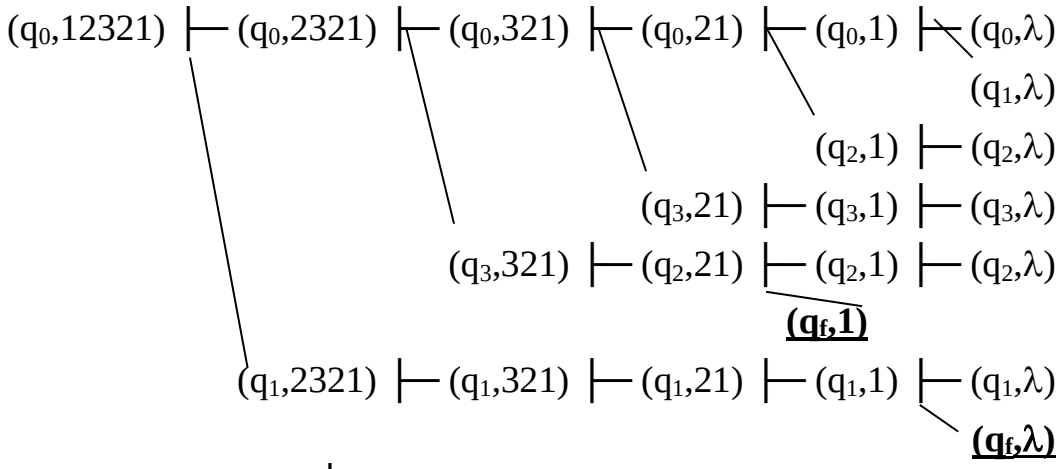
2. Построим НКА, допускающий цепочки в алфавите {1,2,3}, у которых последний символ цепочки уже появлялся в ней ранее (т.е., например, цепочка ‘1232’ должна быть допущена, а ‘122113’ – нет). Введем состояния: q_0 – нейтральное, q_i делает предположение, что последний символ цепочки совпадает с индексом состояния, q_f – заключительное состояние. Из состояния q_0 автомат может перейти в q_a, если наблюдает символ ”a”, или остаться в q_0. Находясь в q_a и наблюдая символ ”a”, автомат может перейти в заключительное состояние q_f или остаться в q_a. Из q_f автомат никуда не переходит.

Формально такой автомат определим следующим образом:

$M=(\{q_0,q_1,q_2,q_3,q_f\},\{1,2,3\},\delta,q_0,\{q_f\})$, функция переходов задана таблицей:

	вход		
состояние	1	2	3
q_0	$\{q_0,q_1\}$	$\{q_0,q_2\}$	$\{q_0,q_3\}$
q_1	$\{q_1,q_f\}$	$\{q_1\}$	$\{q_1\}$
q_2	$\{q_2\}$	$\{q_2,q_f\}$	$\{q_2\}$
q_3	$\{q_3\}$	$\{q_3\}$	$\{q_3,q_f\}$
q_f	$\varnothing$	$\varnothing$	$\varnothing$

Рассмотрим для примера цепочку w=’12321’. Процесс порождения конфигураций будет выглядеть следующим образом:



Поскольку $(q_0,12321)\vdash^*(q_f,\lambda)$, то ‘12321’$\in$L(M).

Построение ДКА, эквивалентного заданному НКА

Моделировать работу ДКА значительно проще, чем НКА. Доказано, что для любого НКА можно построить эквивалентный ему ДКА. Для этого существует специальный алгоритм, идея которого состоит в следующем:

- В качестве состояний нового ДКА рассматриваются все состояния исходного автомата, а также всевозможные сочетания этих состояний по два, по три, …, по n, если в исходном автомате было n состояний.
- Для всех новых состояний строится функция переходов, в которой для каждого состояния q_iq_j появляется переход в новое состояние, представляющий собой объединение всех прежних переходов из q_i и q_j.

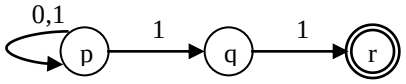
▪ Начальное состояние нового автомата остается тем же, а множество конечных состояний нового автомата будет состоять из всех сочетаний, в которых присутствовало q_f – конечное состояние исходного автомата.

После построения ДКА из него удаляют все недостижимые состояния (состояния, переход в которые невозможен при любой входной цепочке).

В итоге построен ДКА, эквивалентный заданному НКА. При этом если исходный автомат имел n состояний, то в наихудшем случае построенный ДКА будет иметь $(2^n–1)$ состояний.

Рассмотрим на примере построение ДКА, эквивалентного заданному НКА.

1. Пусть M=({p,q,r},{0,1},δ,p,{r}) – НКА, где функция переходов δ задана графом:



Недетерминированный автомат M допускает все цепочки из $\{0,1\}^*$, заканчивающиеся двумя единицами: $(0+1)^*11$.

Представим функцию переходов в табличном виде (таблица справа).

состояние	***вход***	
	0	1
p	{p}	{p,q}
q	–	{r}
r	–	–

состояние	***вход***	
	0	1
p	{p}	{pq}
q	–	{r}
r	–	–
pq	{p}	{pqr}
pr	{p}	{pq}
qr	–	{r}
pqr	{p}	{pqr}

Далее создадим все возможные новые состояния, которые могут получиться в результате сочетаний исходных состояний, и занесём их в таблицу. Новые состояния будем записывать в виде {pq}. Тогда из состояния p по символу 1 переход будет происходить в такое состояние pq. В таблице присутствуют недостижимые состояния q, r, pr и qr, которые можно удалить без изменения результата.

Можно было упростить процесс и сразу заносить в новую таблицу только те состояния, которые действительно могут возникнуть. Тогда состояния pr и qr в ней бы не появились. Заключительным состоянием исходного автомата было состояние r, следовательно, в новом автомате это будет состояние pqr. Если бы, к примеру, состояние pr было достижимым, то оно тоже являлось бы заключительным, так как автомат может иметь несколько заключительных состояний. После удаления недостижимых состояний таблица примет вид:

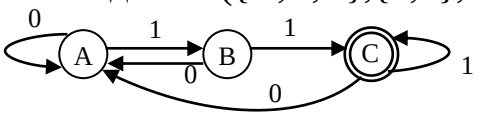
состояние	***вход***	
	0	1
p	{p}	{pq}
pq	{p}	{pqr}
pqr	{p}	{pqr}

Для удобства переобозначим состояния: заменим p на A, pq на B, pqr – на C. Тогда начальным состоянием будет A, конечным C.

состояние	***вход***	
	0	1
A	{A}	{B}
B	{A}	{C}
C	{A}	{C}

В итоге полученный детерминированный автомат, эквивалентный

исходному НКА, будет иметь вид: M=({A,B,C},{0,1},δ,A,{C}), граф функции переходов δ:



Минимизация КА

Многие КА возможно минимизировать, т.е. построить эквивалентный КА с минимально возможным числом состояний. Для этого существует алгоритм минимизации автомата. Дадим необходимые определения.

Пусть q_1 и q_2 – состояния автомата M, на вход подается цепочка символов w длины $k \geq 0$: $w \in V^*$, $|w| \leq k$, $(q_1,w) \vdash^* (q_3,\lambda)$, $(q_2,w) \vdash^* (q_4,\lambda)$. Если одно из состояний (q_3 или q_4) входит в F, а другое – нет, то говорят, что цепочка w *различает состояния q_1 и q_2*. Если же для любой входной цепочки w длины k она не различает состояния q_1 и q_2, то говорят, что они являются *k-эквивалентными* или *k-неразличимыми*. Множество *K-эквивалентных* состояний составляет класс эквивалентности R(k).

Очевидно, что множества F и Q\F являются 0-эквивалентными, записывается этот факт следующим образом: R(0)={F,Q\F}.

Для построения минимального автомата строятся классы эквивалентности R(n).

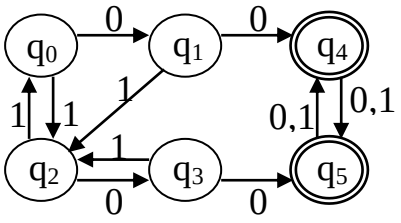
Алгоритм построения классов эквивалентности:

1. Полагаем n:=0, строится R(0)= {F,Q\F}.
2. n:=n+1. Строится R(n) на основании R(n–1): в класс эквивалентности R(n) входят те состояния, которые по одинаковым символам переходят в n–1-эквивалентные состояния: $R(n):=\{r_{ij}(n): \{q_{ij} \in Q: \forall a \in V\ \delta(q_{ij},a) \subseteq r_j(n-1)\}\ \forall i,j \in N\}$.
3. Если R(n)= R(n–1), то работа заканчивается. Иначе переход на шаг 2.

Алгоритм минимизации автомата:

1. Исключить все недостижимые состояния.
2. Построить классы эквивалентности.
3. Каждый из этих классов становится состоянием нового автомата.
4. Функция переходов строится на основании функции переходов исходного автомата и новых состояний.

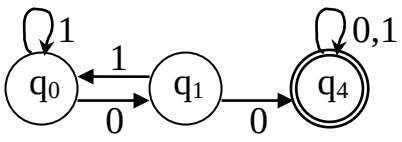
Рассмотрим пример. Пусть ДКА имеет вид: $M = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{0,1\}, \delta, q_0, \{q_4, q_5\})$, а функция переходов задана графом:



Требуется минимизировать данный автомат.

Недостижимых состояний нет. Построим классы эквивалентности: $R(0)=\{\{q_0,q_1,q_2,q_3\}, \{q_4,q_5\}\}$. Из состояния q_2 по 0 происходит переход в q_3, а из q_1 – в q_4. Состояния q_3 и q_4 находятся в разных классах эквивалентности $R(0) \Rightarrow q_1$ и q_2 будут входить в разные классы R(1).

Аналогично рассуждая про остальные состояния, получим: R(1)={{q_4,q_5},{q_0,q_2},{q_1,q_3}}. Аналогично: R(2)={{q_4,q_5}, {q_0,q_2}, {q_1,q_3}}. ⇒ В новом автомате 3 состояния. Обозначим их по номеру одного из состояний каждого класса. M' = {q_0,q_1,q_4}, {0,1}, δ', q_0, {q_4}).



2.2.3 Контрольные вопросы

1. В чём отличие автоматной грамматики от регулярной?
2. Как связаны понятия автоматной грамматики и конечного автомата?
3. Может ли конечный автомат иметь несколько заключительных состояний?
4. В чём состоит работа конечного автомата? В каком случае конечный автомат заканчивает работу?
5. Какие составные элементы входят в понятие конфигурации КА?
6. Как определить, являются ли два КА эквивалентными?
7. Чем различаются детерминированный и недетерминированный конечные автоматы? Есть ли какие-то различия в их работе или же отличаются только описания автоматов?
8. Какой КА является полностью определённым?
9. Какие существуют способы для задания функции переходов конечного автомата?
10. Пусть M=({p,q,r},{0,1},δ,p,{r}) – ДКА, где функция переходов задана таблицей:

	вход	
состояние	0	1
p	{q}	–
q	{r}	{r}
r	{q}	{q}

Выясните, какой язык задаётся данным автоматом. Доопределите автомат до полностью определённого вида.

11. С какой целью преобразуют НКА в ДКА? Всегда ли возможно это сделать?
12. Для чего нужно строить классы эквивалентности?

2.3 Особенности регулярных языков

2.3.1 Способы задания регулярных языков

Регулярные грамматики, конечные автоматы и регулярные множества (и обозначающие их регулярные выражения) представляют собой три различных способа задания регулярных языков.

Утверждение

1. Язык является РМ тогда и только тогда, когда он задан леволинейной

(праволинейной) грамматикой. Язык может быть задан леволинейной (праволинейной) грамматикой тогда и только тогда, когда он является регулярным множеством.

2. Язык является РМ тогда и только тогда, когда он задан с помощью конечного автомата. Язык распознается с помощью конечного автомата тогда и только тогда, когда он является РМ.

Все эти три способа эквивалентны. Существуют алгоритмы, позволяющие для языка, заданного одним из способов, построить другой способ, определяющий этот же язык (см. список литературы).

Например, для нахождения регулярного выражения для языка, заданного праволинейной грамматикой, необходимо построить и решить систему уравнений с регулярными коэффициентами.

В теории языков программирования наиболее важную роль играет эквивалентность КА и регулярных грамматик, поскольку такие грамматики используются для определения лексических конструкций языков программирования. Создав на основе известной грамматики автомат, получаем распознаватель для данного языка. Таким образом удается решить задачу разбора для лексических конструкций языка.

Для построения КА на основе известной регулярной грамматики её необходимо привести к автоматному виду. Множество состояний автомата будет соответствовать множеству нетерминальных символов грамматики.

2.3.2 Свойства регулярных языков

Регулярные множества замкнуты относительно операций пересечения, объединения, дополнения, итерации, конкатенации, изменения имен символов и подстановки цепочек вместо символов.

Для регулярных языков разрешимы многие проблемы, неразрешимые для других типов языков. Например, следующие проблемы являются разрешимыми независимо от того, каким из способов задан язык:

Проблема эквивалентности: Даны два регулярных языка $L_1(V)$ и $L_2(V)$. Необходимо установить, являются ли они эквивалентными.

Проблема принадлежности цепочки языку. Дан регулярный язык L(V), цепочка символов $\alpha \in V^*$. Требуется проверить, принадлежит ли эта цепочка языку.

Проблема пустоты языка. Дан регулярный язык L(V). Необходимо проверить, является ли этот язык пустым, т.е. найти хотя бы одну цепочку $\alpha \neq \lambda$, $\alpha \in L(V)$.

Иногда бывает необходимо доказать, является ли некоторый язык регулярным. Если возможно задать этот язык одним из рассмотренных способов, то он является регулярным. Но если такой способ найти не удается, неизвестно, является язык регулярным или нет.

Для регулярных языков выполняется т.н. лемма о разрастании

регулярных языков. Она часто используется для доказательства (от противного) того факта, что некоторый язык не является регулярным. Для такого доказательства предполагают, что язык является регулярным, в таком случае лемма должна выполняться. Если в результате получается противоречие, это доказывает, что предположение было неверным, следовательно, язык регулярным не является.

Лемма о разрастании регулярных языков формулируется следующим образом. Если дан регулярный язык и достаточно длинная цепочка символов, принадлежащая этому языку, то в ней можно найти непустую подцепочку, которую можно повторить сколь угодно много раз, и все полученные таким образом цепочки также будут принадлежать рассматриваемому регулярному языку.

Формально лемма записывается так. Если дан регулярный язык L, то $\exists$ константа $p>0$, такая, что если $\alpha \in L$ и $|\alpha| \geq p$, то цепочку α можно записать в виде $\alpha=\delta\beta\gamma$, где $0<|\beta| \leq p$, и тогда $\alpha'=\delta\beta^i\gamma$, $\alpha' \in L$ $\forall i \geq 0$.

Пример. Рассмотрим язык $L=\{a^n b^n \mid n>0\}$. Докажем, что он не является регулярным, используя для этого лемму о разрастании языков.

Пусть этот язык регулярный, тогда для него должна выполняться лемма о разрастании. Возьмем некоторую цепочку этого языка $\alpha= a^n b^n$ и запишем ее в виде $\alpha=\delta\beta\gamma$. Если $\beta \in a^+$ или $\beta \in b^+$, то тогда цепочка $\delta\beta^i\gamma$ не принадлежит языку для любого i, что противоречит условиям леммы. Если же $\beta \in a^+b^+$, то цепочка $\delta\beta^2\gamma$ также не принадлежит языку L. Получили противоречие, следовательно, язык не является регулярным.

2.3.3 Контрольные вопросы

1. В каком случае язык является регулярным? Какие существуют эквивалентные способы его задания?
2. Что означает замкнутость множества относительно операции?
3. Каковы способы проверки языка на регулярность? Является ли возможность построения регулярного выражения, задающего язык, доказательством его регулярности?
4. Любую ли цепочку языка можно брать согласно лемме о разрастании, или достаточно найти одну подходящую условию?