Конечные автоматы

Пример 1. На Рис. 1 показан граф переходов НКА, распознающего язык регулярного выражения $(a+b)^*abb$. Этот абстрактный язык, распознающий строки из a и b, заканчивающиеся подстрокой abb.

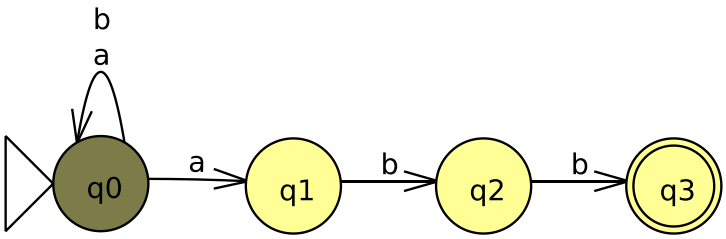


Рис. 1: Недетерминированный конечный автомат

Двойной кружок вокруг состояния 3 указывает, что это — допускающее состояние. Единственный способ попасть из состояния 0 в допускающее — проследовать по некоторому пути, который некоторое время остается в состоянии 0, а затем проходит по состояниям 1, 2 и 3, считывая из входного потока символы abb. Таким образом, в допускающее состояние ведут только те строки, которые заканчиваются суффиксом abb.

Пример 2. На Рис. 2 Рис. 1показан НКА, принимающий $\mathcal{L} = aa^*+bb^*$. Строка aaa распознается этим автоматом в силу существования пути ϵaaa.

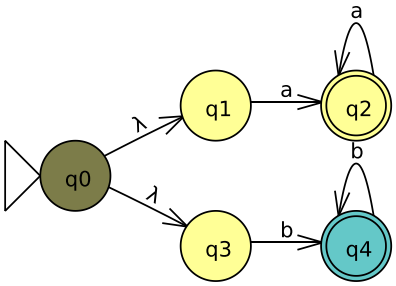


Рис. 2: НКА, распознающий язык aa^*+bb^*

Конечные автоматы представляют собой графы, подобные диаграммам переходов, со следующими отличиями:

1. Конечные автоматы являются распознавателями[1]; они просто говорят «да» или «нет» для каждой возможной входной строки.
2. Конечные автоматы разделяются на два класса:
    a) Недетерминированные конечные автоматы[2] не имеют ограничений на свои дуги. Символ может быть меткой нескольких дуг, исходящих из одного и того же состояния; кроме того, одна из возможных меток — пустая строка $\lambda = \epsilon$.
    b) Детерминированные конечные автоматы[3] для каждого состояния и каждого символа входного алфавита имеют ровно одну дугу с указанным символом, покидающим это состояние.

Как детерминированные, так и недетерминированные конечные автоматы способны распознавать одни и те же языки. По сути, это те же языки, именуемые регулярными языками[4], которые могут быть описаны регулярными выражениями[5].

Недетерминированные конечные автоматы

Недетерминированный конечный автомат (НКА) — это пятерка, которая состоит из

1. множества состояний S;
2. множества входных символов над входным алфавитом Σ, и символ, обозначающий пустую строку $\epsilon \notin \Sigma$;
3. функции переходов, которая для каждого состояния и каждого символа из множества $\Sigma \cup \{\epsilon\}$ дает множество последующих состояний[6];
4. состояния s_0 из множества S, известного как начальное или стартовое состояние,
5. множества состояний $F \subset S$, являющегося подмножеством множества S, известных как допускающие или конечные состояния.

Как недетерминированный, так и детерминированный конечные автоматы представляется в виде графа переходов, узлы которого представляют состояния, а помеченные дуги — функцию переходов. Дуга из состояния s в состояние t, помеченная символом a, существует тогда и только тогда, когда t — одно из последующих состояний для состояния s и входного символа $a \in \Sigma \cup \{\epsilon\}$. Этот граф очень похож на диаграмму переходов, за исключением того, что

a) один и тот же символ может помечать дуги, исходящие из одного состояния в несколько разных других;
b) дуга может быть помечена пустой строкой ϵ вместо символа входного алфавита (или вместе с ним).

---

[1] recognizer
[2] nondeterministic finite automata — NFA
[3] deterministic finite automata — DFA
[4] regular language
[5] Регулярные выражения не могут описывать пустой язык, поскольку этот шаблон не используется на практике. Однако конечный автомат может определять пустой язык. .
[6] next state

Детерминированный конечный автомат

Детерминированный конечный автомат (ДКА) представляет собой частный случай НКА, в котором

a) нет переходов для входа ϵ;
b) для каждого состояния s и входного символа a имеется ровно одна дуга, выходящая из состояния s и помеченная символом алфавита a.

Если воспользоваться для представления ДКА таблицей переходов, то каждая запись в ней будет являться единственным состоянием, — так что его можно указывать без фигурных скобок, означающих множество.

В то время как НКА — это абстрактное представление алгоритма для распознавания строк некоторого языка, ДКА является простым, конкретным алгоритмом распознавания строк. Каждое регулярное выражение и каждый НКА преобразуются в ДКА, распознающий тот же язык.

Алгоритм I. Моделирование ДКА

Вход: входная строка x, завершенная символом конца файла **eof**, и детерминированный конечный автомат D с начальным состоянием s_0, принимающими состояниями F и функцией переходов $\text{move}(s,c)$.

Выход: ответ «да», если автомат D принимает x, и ответ «нет» — в противном случае.

Метод: применение алгоритма, приведенного на Рис. 3, к входной строке x. Функция $\text{move}(s,c)$ дает состояние, в которое из состояния s ведет дуга при входном символе c. Функция nextchar возвращает очередной символ из входной строки x.

```
s = s0;
c = nextchar();
while(c!=eof) {
s = move(s,c);
c = nextchar();
}
if(s∈F) return «да»;
else return «нет»;
```

Рис. 3. Псевдокод модели ДКА

Пример 3. На Рис. 4 показан граф ДКА, принимающий язык $(a+b)^*abb$, тот же, что и в случае НКА на Рис. 1. Для данной входной строки $ababb$ этот ДКА проходит последовательность состояний 0, 1, 2, 1, 2, 3 и возвращает «да».

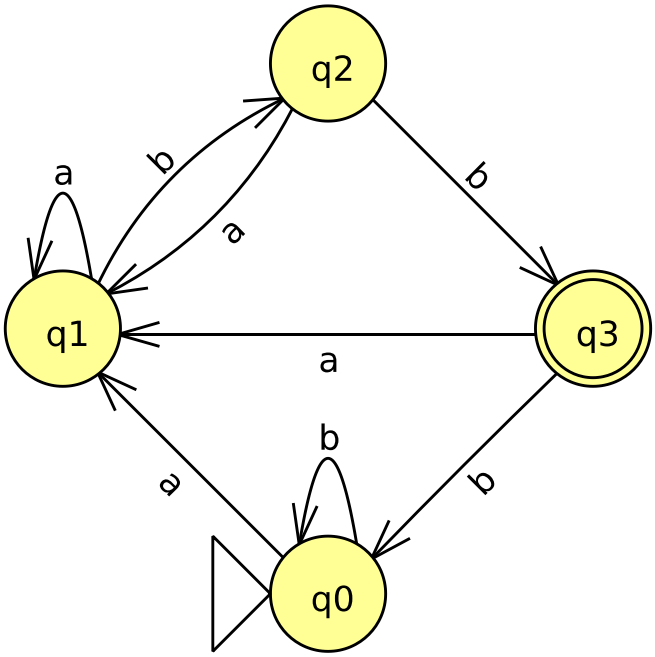


Рис. 4: ДКА, принимающий язык (a+b)*abb

От регулярных выражений к автоматам

Регулярное выражение предоставляет способ описания лексических анализаторов и другого программного обеспечения для работы с шаблонами. Реализация программы требует моделирования ДКА, см. Алгоритм I. Моделирование НКА существенно сложнее, чем моделирование ДКА, поскольку при работе приходится перебирать множественные переходы для входного символа. Например, на Рис. 1 имеется недетерминированное состояние 0 для символа a. На Рис. 2 представлена возможность спонтанного перехода из состояния 0 в состояния 1 или 2 по дугам пустой цепочки $\lambda = \epsilon$.

Для написания квалифицированной программы, не обращающейся без необходимости к переборному алгоритму, оказывается очень важной задача конвертации НКА в ДКА, который распознает тот же язык.

Преобразование НКА в ДКА

Общая идея, лежащая в основе построения подмножеств, заключается в том, что каждое состояние строящегося ДКА соответствует множеству состояний НКА.

После чтения входной строки $a_1 a_2 \ldots a_n$ ДКА находится в состоянии, соответствующем множеству состояний, которых может достичь из своего стартового состояния НКА по пути, помеченному $a_1 a_2 \ldots a_n$.

Возможна ситуация, когда количество состояний ДКА экспоненциально зависит от количества состояний НКА, что может привести к сложностям при реализации такого ДКА. Для реальных

языков НКА и ДКА имеют примерно одинаковое количество состояний, без экспоненциального поведения.

Алгоритм II. Построение ДКА из НКА[7]

Вход: НКА N.

Выход: ДКА D, принимающий тот же язык, что и N.

Метод: алгоритм строит таблицу переходов D_{tran} для автомата D. Каждое состояние D представляет собой множество состояний НКА, и D_{tran} строится таким образом, чтобы «параллельно» моделировать все возможные переходы, которые N может выполнить для данной входной строки. Очень важно понять корректную обработку ϵ-переходов N. На Рис. 5 приведены определения некоторых функций, которые описывают базовые вычисления состояний N, необходимые для данного алгоритма. Здесь s — единственное состояние N, в то время как T — множество состояний N.

Исследуются те множества состояний, в которых N может оказаться после обработки некоторой входной строки. Перед прочтением первого символа N может находиться в любом из состояний $\epsilon\text{-closure}(s_0)$,

Операция	Описание
$\epsilon\text{-closure}(s)$	Множество состояний НКА N, достижимых из состояния s при одном ϵ-переходе
$\epsilon\text{-closure}(T)$	Множество состояний НКА N $\bigcup_{s\in T}\epsilon\text{-closure}(s)$, достижимых из состояния s из множества T при одном ϵ-переходе
$\text{move}(T,a)$	Множество состояний НКА N, в которые имеется переход из некоторого состояния $s\in T$ при входном символе a

Рис. 5: Операции над состояниями НКА, где s_0 — начальное состояние.

Предположим, что после чтения строки x автомат N может находиться во множестве состояний T. Если следующий прочитанный символ — a, то N может перейти в любое из состояний $\text{move}(T,a)$. Однако после чтения a может быть выполнено несколько ϵ-переходов; таким образом, после прочтения ax конечный автомат N может быть в любом состоянии из множества $\epsilon\text{-closure}(\text{move}(T,a))$. Построение множества состояний D_{states} конечного автомата D и его функции переходов D_{tran} в соответствии с этими идеями показано на рис. 3.32.

```
Изначально в Dstates содержится только одно состояние, ε-closure(s0),
и оно не помечено.
while (в Dstates имеется непомеченное состояние T) {
Пометить T;
for(каждый входной символ a ) {
```

[7] subset construction

$U = \epsilon\text{-closure}\left(\text{move}\left(T, a\right)\right)$;
if ($U \notin D_{\text{states}}$)
Добавить U в D_{states} как непомеченное состояние;
$D_{\text{tran}}[T, a] = U$;
}
}

Рис. 6: Построение множества состояний D_{states}

Стартовым состоянием детерминированного автомата D является множество $\epsilon\text{-closure}\left(s_0\right)$, а принимающими состояниями D являются те множества состояний N, которые включают как минимум одно принимающее состояние N. Для завершения описания построения подмножества требуется показать, как вычислить $\epsilon\text{-closure}\left(T\right)$ для произвольного множества состояний T недетерминированного конечного автомата. Этот процесс показан на Рис. 7. В данном случае представьте, что в графе имеются только ребра с метками ϵ.

Поместить все состояния T в стек **stack**;
Инициализировать $\epsilon\text{-closure}\left(T\right)$ множеством T;
while (**stack** не пуст) {
Снять со стека **stack** верхний элемент t;
for(каждое состояние u с дугой из t в u, помеченной ϵ)
if($u \notin \epsilon\text{-closure}\left(T\right)$) {
Добавить u в $\epsilon\text{-closure}\left(T\right)$;
Поместить u в **stack**;
}
}

Рис. 7: Вычисление множества ϵ-closure(T)

Пример 4. На Рис. 8 показан еще один НКА, принимающий (a+b)*abb ; это автомат, который позже будет построен непосредственно из регулярного выражения. Применим Алгоритм II к Рис. 8.

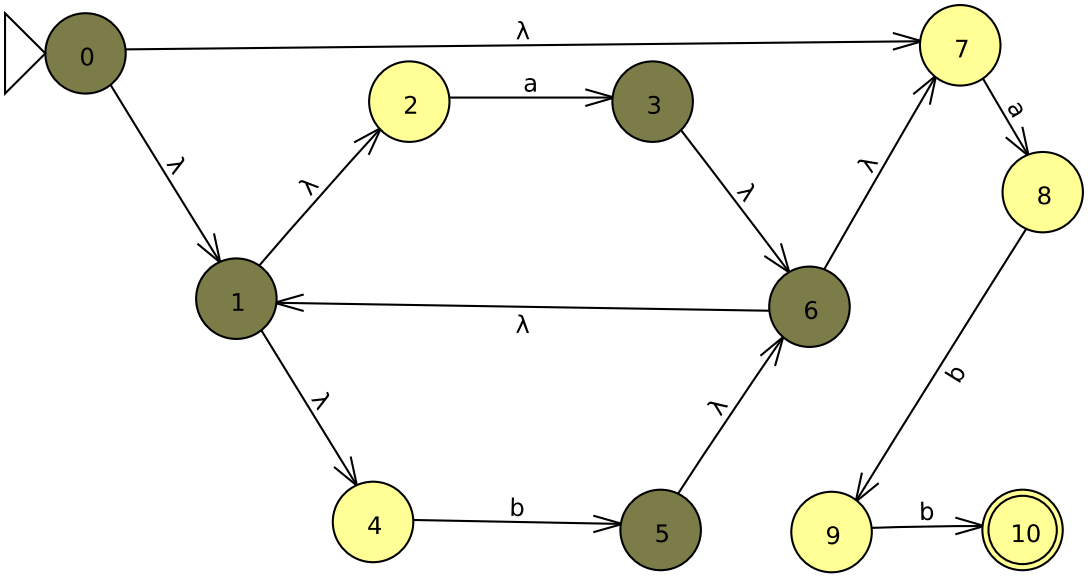


Рис. 8: НКА N для (a+b)*abb

Стартовое состояние $\mathcal{A}$ эквивалентного ДКА — это множество ϵ-closure(0), или $\mathcal{A} = \{0,1,2,4,7\}$. Это именно те состояния, в которые можно перейти из состояния 0 по пути, у которого все ребра имеют метки ϵ. Путь может состоять из нуля дуг, так что состояние 0 достижимо само из себя по ϵ-пути.

Входной алфавит представляет собой $\{a,b\}$. Первый шаг состоит в том, чтобы пометить $\mathcal{A}$ и вычислить $D_{\text{tran}}[\mathcal{A},a] = \epsilon\text{-closure}\left(\text{move}(\mathcal{A},a)\right)$ и $D_{\text{tran}}[\mathcal{A},b] = \epsilon\text{-closure}\left(\text{move}(\mathcal{A},b)\right)$. Среди состояний 0, 1, 2, 4 и 7 только 2 и 7 имеют переходы по символу a в состояния 3 и 8 соответственно. Таким образом, $\text{move}(\mathcal{A},a) = \{3,8\}$. Далее, $\epsilon\text{-closure}(3,8) = \{1,2,3,4,6,7,8\}$, так что: $D_{\text{tran}}[\mathcal{A},a] = \epsilon\text{-closure}\left(\text{move}(\mathcal{A},a)\right) = \epsilon\text{-closure}\left(\{3,8\}\right) = \{1,2,3,4,6,7,8\}$. Назовем это множество именем $\mathcal{B}$, так что $D_{\text{tran}}[\mathcal{A},a] = \mathcal{B}$.

Теперь вычисляется $D_{\text{tran}}[\mathcal{A},b]$. Среди состояний $\mathcal{A}$ только 4 имеет переход по b, и это переход в состояние 5. Таким образом, $D_{\text{tran}}[\mathcal{A},b] = \epsilon\text{-closure}(5) = \{1,2,4,5,6,7\}$

Назовем это множество именем $\mathcal{C}$, так что $D_{\text{tran}}[\mathcal{B},a] = \mathcal{C}$.

Состояния НКА	Состояния ДКА	a	b
{0,1,2,4,7}	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$
{1,2,3,4,6,7,8}	$\mathcal{B}$	$\mathcal{B}$	$\mathcal{D}$
{1,2,4,5,6,7}	$\mathcal{C}$	$\mathcal{B}$	$\mathcal{C}$
{1,2,4,5,6,7,9}	$\mathcal{D}$	$\mathcal{B}$	$\mathcal{E}$
{1,2,4,5,6,7,10}	$\mathcal{E}$	$\mathcal{B}$	$\mathcal{C}$

Рис. 9: Таблица переходов D_{tran} для ДКА D

Если продолжить процесс с непомеченными состояниями $\mathcal{B}$ и $\mathcal{C}$, в конечном счете, мы достигнем точки, в которой все состояния ДКА будут помечены. Это заключение справедливо, потому что имеется конечное число подмножеств множества из 11 состояний НКА. Фактически построено пять различных состояний ДКА, которые соответствуют множеству состояний НКА. Соответствующая таблица переходов показана Рис. 9, а граф переходов D — на Рис. 10. Состояние $\mathcal{A}$ — начальное, а состояние $\mathcal{E}$, которое содержит состояние 10, является единственным принимающим состоянием ДКА D.

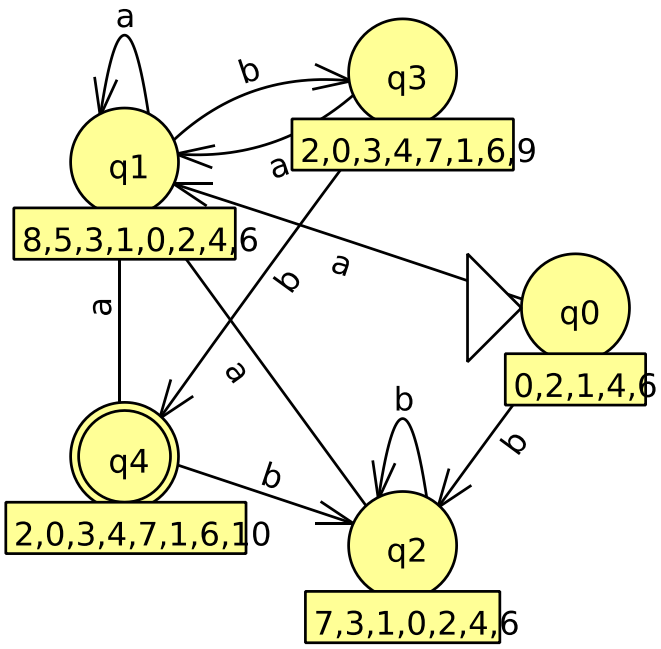


Рис. 10: Результат применения построения подмножеств к НКА на рис. 3.34

Обратите внимание, что ДКА D имеет на одно состояние больше, чем ДКА для того же языка на Рис. 1. Состояния $\mathcal{A}$ и $\mathcal{C}$ имеют одну и ту же функцию переходов, так что их следует объединить.

Минимизация количества состояний

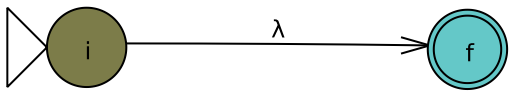
Алгоритм III. Алгоритм Мак-Нотона-Ямады-Томпсона для преобразования регулярного выражения в НКА[8]

Вход: регулярное выражение r над алфавитом Σ.

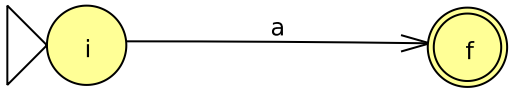
Выход: НКА N, принимающий $L(r)$.

Метод: начнем с разбора r на составляющие подвыражения. Правила построения НКА состоят из базисных правил для обработки подвыражений без операторов и индуктивных правил для построения больших НКА из НКА для непосредственных подвыражений данного выражения.

Базис: для выражения $\lambda = \epsilon$ строим НКА.



Здесь i — новое состояние, представляющее собой начальное состояние НКА, а f — другое новое состояние, являющееся принимающим состоянием НКА.
Для любого литерала a из алфавита Σ строим НКА.



Здесь также i и f — новые состояния, являющиеся соответственно начальным и заключительным. Обратите внимание, что в обоих базовых случаях мы строим отдельные НКА с новыми состояниями для каждого ϵ и некоторого a, являющихся подвыражением r.

[8] McNaughton-Yamada-Thompson

Индукция: предположим, что $N(s)$ и $N(t)$ — недетерминированные конечные автоматы для регулярных выражений s и t соответственно.

а) Пусть $r = s|t$. Тогда $N(r)$, — НКА для r, строится так, как показано на Рис. 11. Здесь r и t — новые состояния, являющиеся соответственно начальным и принимающим состояниями $N(r)$. Имеются также ϵ-переходы от r к стартовым состояниям $N(s)$ и $N(t)$ и ϵ-переходы от каждого из их принимающих состояний в t. Обратите внимание, что принимающие состояния $N(s)$ и $N(t)$ не являются таковыми в $N(r)$. Поскольку любой путь из r в t должен пройти либо исключительно по $N(s)$, либо исключительно по $N(t)$ и поскольку метки пути не изменяются при добавлении к ним е при выходе из r и входе в t, можно заключить, что $N(r)$ принимает язык $L(s) \bigcup L(t)$, который представляет собой не что иное, как язык $L(r)$.
Таким образом, на Рис. 11 представлено корректное построение НКА для оператора объединения.

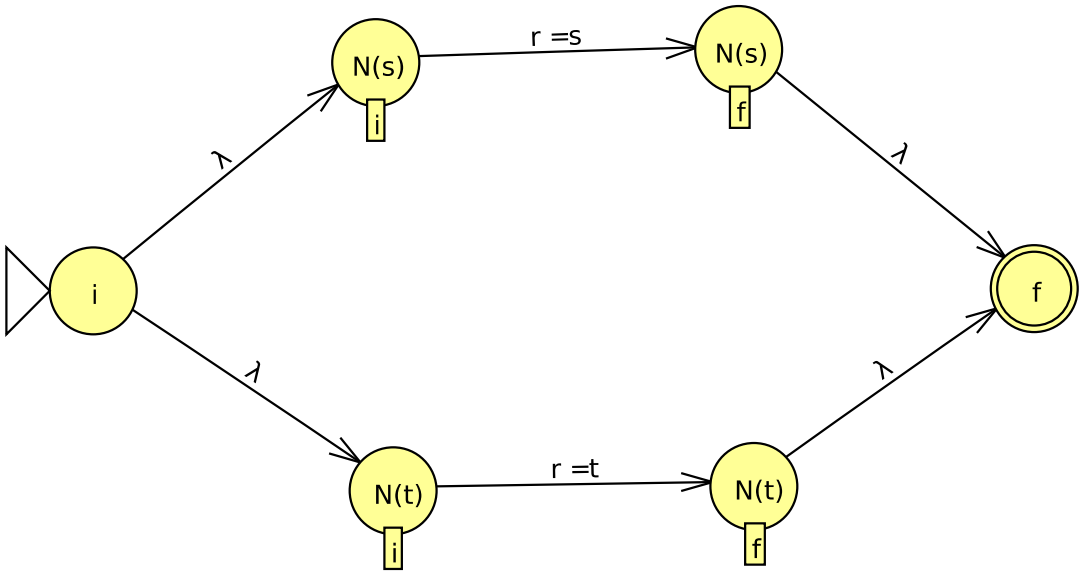


Рис. 11: НКА для объединения двух регулярных выражений

б) Пусть $r = st$. В этом случае построим $N(r)$ так, как показано на Рис. 12.

Начальное состояние $N(s)$ становится начальным состоянием $N(r)$, а принимающее состояние $N(t)$ — единственным принимающим состоянием $N(r)$. Принимающее состояние $N(s)$ и начальное состояние $N(t)$ объединяются в одно состояние со всеми входящими и исходящими переходами обоих состояний. На Рис. 12 показано, что путь от i к f сначала проходит через автомат $N(s)$. Метка этого пути начинается с некоторой строки из множества $L(s)$. Затем путь проходит через автомат $N(t)$. Его метка заканчивается строкой из множества $L(t)$.
Принимающие состояния не имеют исходящих дуг, а начальные состояния — входящих. Как только путь покинул $N(s)$, то обратного пути нет. Таким образом, $N(r)$ принимает

исключительно строки из языка $L(s)L(t)$ и является корректным НКА для регулярного выражения $r = st$.

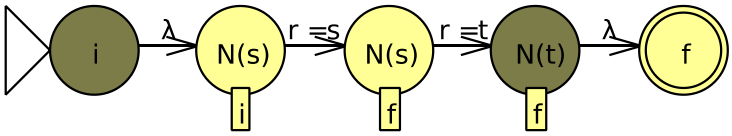


Рис. 12: НКА для конкатенации двух регулярных выражений

в) Пусть $r = s*$. Тогда для НКА $N(r)$ строится так, как показано на Рис. 13.
Здесь i и f — новые начальное состояние и принимающее состояние автомата $N(s)$. Для достижения заключительного состояния f из начального состояния i необходимо либо пройти по пути с меткой ϵ, соответствующему строке $L^0(s) = \epsilon$, либо перейти к начальному состоянию $N(s)$, пройти по пути $N(s)$ и вернуться из его принимающего состояния в начальное состояние нуль или несколько раз. Эта возможность позволяет $N(r)$ принимать все строки из множеств $L^1(s) = L(s)$, $L^2(s) = L(s)L(s)$ и так далее, так что полное множество строк, принимаемое $N(r)$, представляет собой $L(s*)$.

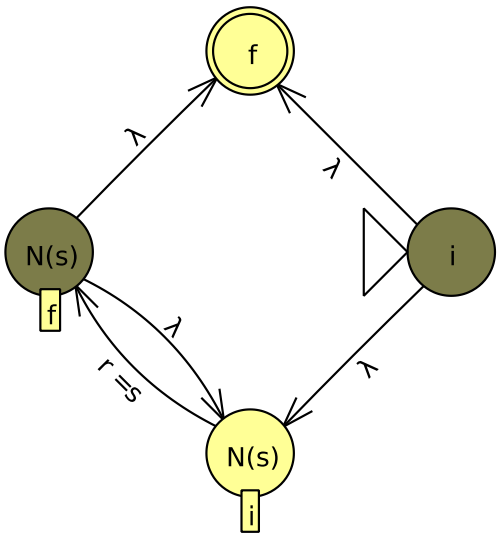


Рис. 13: НКА для замыкания регулярного выражения

г) Наконец, пусть $r = (s)$. Тогда $L(r) = L(s)$ и можно использовать НКА $N(s)$ как $N(r)$.

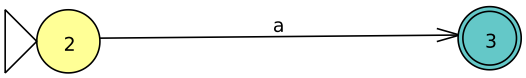
Описание Алгоритм III содержит указания о том, почему корректны индуктивные построения. Мы не даем формального доказательства их корректности, но перечислим некоторые свойства построенных НКА в дополнение к тому главному факту, что $N(r)$ принимает язык $L(r)$. Эти свойства интересны как сами по себе, так и с точки зрения использования в формальном доказательстве.

1. Количество состояний в $N(r)$ не более чем в два раза превышает количество операторов и операндов в r. Это следует из того факта, что каждый шаг алгоритма создает не более двух новых узлов.

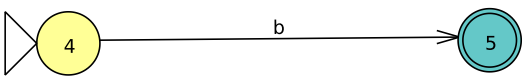
2. $N(r)$ имеет одно начальное и одно принимающее состояние. Принимающее состояние не имеет исходящих переходов, а начальное — входящих.

3. Каждое состояние $N(r)$, отличное от принимающего, имеет либо один исходящий переход по символу из алфавита Σ, либо два исходящих перехода, оба по ϵ.

Пример 5. Воспользуемся Алгоритм III для построения НКА для регулярного выражения $r = (a+b)^*abb$. На Рис. 14 показано синтаксическое дерево разбора r, подобное деревьям разбора для арифметических выражений. Для подвыражения $r_1 = a$ граф выглядит так



Номера состояний выбираются для согласования с дальнейшими стадиями построения НКА. Для подвыражения $r_2 = b$ граф выглядит аналогично.



Теперь можно объединить $N(r_1)$ и $N(r_2)$ с использованием построения на рис. 3.40 и получить НКА для регулярного выражения $r_3 = r_1 \mid r_2$. Этот НКА показан на Рис. 15.

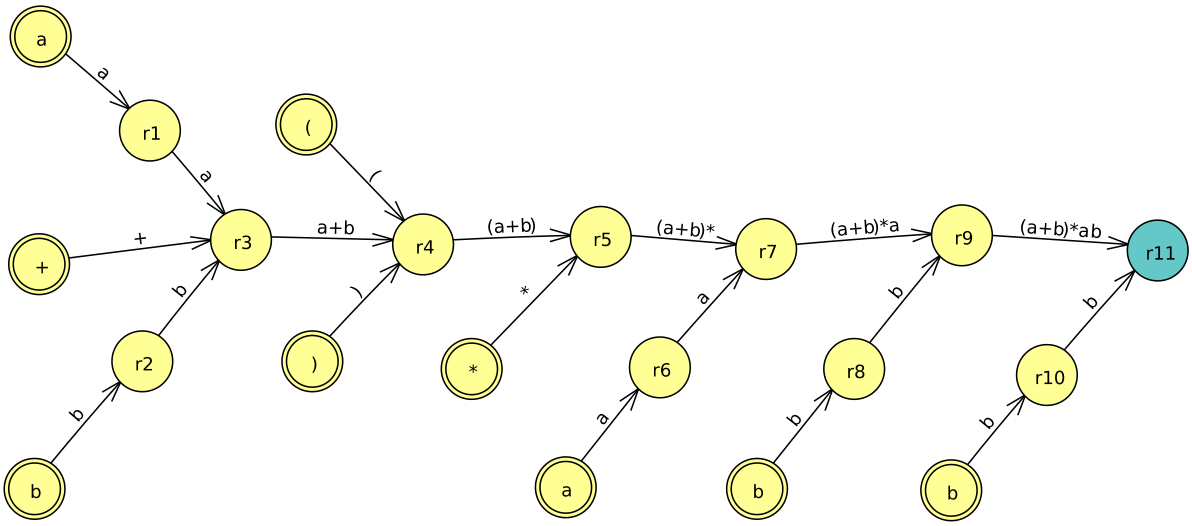


Рис. 14: Дерево разбора для регулярного выражения (a+b)*abb

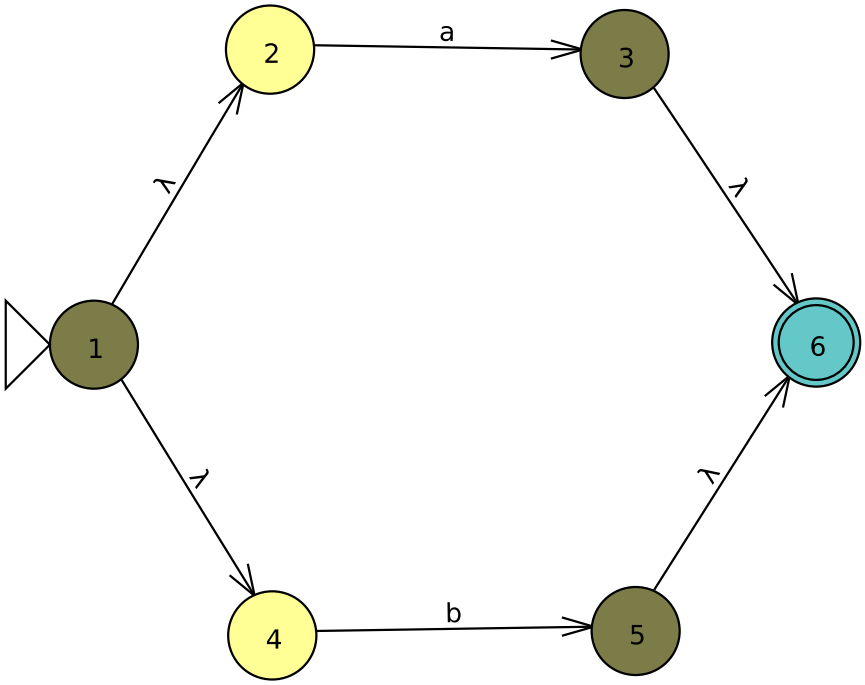


Рис. 15: НКА для r_3

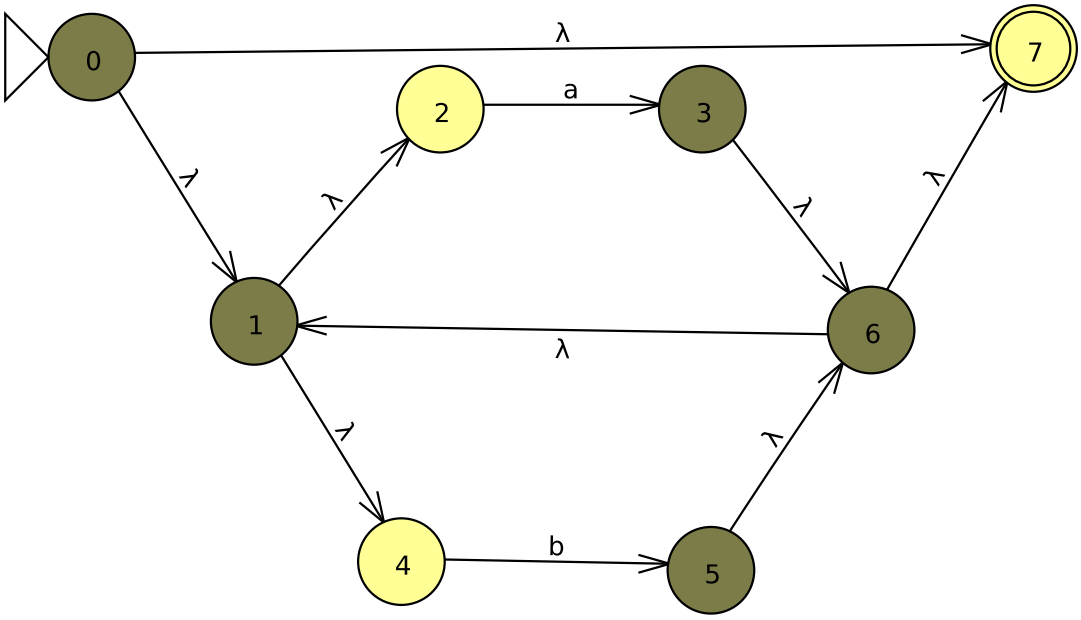
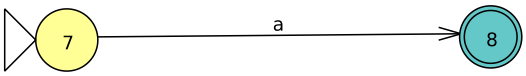


Рис. 16: НКА для r_5

НКА для регулярного выражения $r_4 = (r_3)$ имеет тот же вид, что и для выражения r_3. Конечный автомат для выражения $r_5 = (r_4)*$ приведен на рис. Рис. 16. Для его получения из НКА на Рис. 13 было использовано построение, показанное на Рис. 11.

Рассмотрим подвыражение r_6. Для него мы опять воспользуемся базисным построением, но с новыми состояниями. НКА для регулярного выражения r_6 имеет следующий вид



Для получения НКА для регулярного выражения $r_7 = r_5 r_6$ применяется построение, показанное на Рис. 12. Мы объединяем состояния 7 и 7', что дает НКА, показанный на Рис. 17.

Продолжая работу с новым НКА для двух подвыражений символа b — r_8 и r_9, — построим НКА для регулярного выражения (a+b)*abb, с которым уже встречались на Рис. 1.

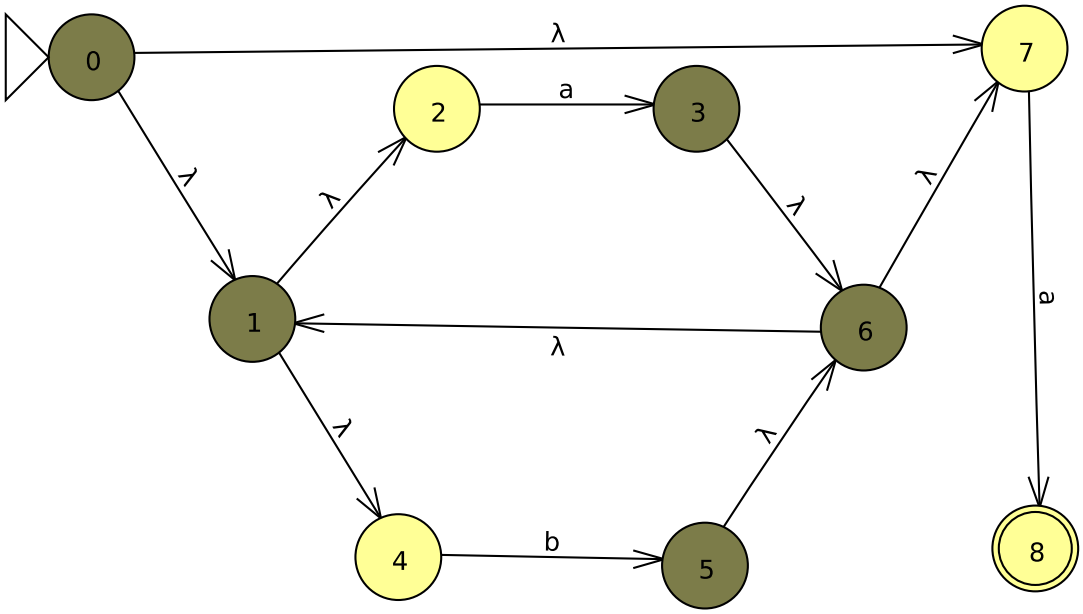


Рис. 17: НКА для r_7