[-НКА](#_ftq4kzc6j8q8)

[Моноиды и автоматы.](#_nhi98xahc4fx)

[Распознаваемость моноидами и автоматами.](#_kn6wkajzvk6g)

[Минимальные автоматы](#_rnar49cfmp2)

Сайт, который помогает рисовать: <http://madebyevan.com/fsm/>

Теория автоматов

* *Все очень хорошо, но плохо*
* *Нет, все очень плохо  
  @Антон Толстов*

**Опр.** Алфавитом будем называть любое конечное множество.  
Обозначение:

— буквы.

**Опр.** Слово над алфавитом - конечная последовательность букв.

Слово может быть пустым, обозначается .

**Опр.** - свободная полугруппа над (т.е. множество непустых слов с операцией конкатенации).

Конкатенация — некоммутативна! (зато ассоциативна)

**Опр.**  — свободный моноид

**Опр.**  - число букв в слове

**Опр.** Язык - множество

**Операции над языками:**

1. Как над множествами: дополнение (универсальным, видимо, считается ).
2. Произведение. Пусть

**Опр.**

Пусть

Тогда - префикс

- суффикс

**Опр.**

Пусть

Тогда - подслово (фактор)

**Прим.:** При конкатенации строки самой на себя раз допустима запись в виде степени.

То есть,

**Опр.** Левое (правое) частное:

Пусть есть - языки над алфавитом .

**Опр.** Итерация Клини:

Пусть . Тогда:

Обозначение:   
Если язык состоит из одного слова , будем так его и обозначать.

В частности, язык из одного пустого слова обозначим (а не ).

Обозначение:

будем писать просто как

Можно заметить, что - это как раз и будет итерация клини от .  
То есть, обозначения корректны.

Можно ещё писать

А можно , и так мы и будем делать в дальнейшем.

**Опр.** Регулярное выражение над алфавитом :

Выражение одного из следующих видов:

1. - регулярное выражение (задающее язык из пустого слова)
2. - регулярное выражение (задающее язык из единственного слова, состоящего из единственной буквы )
3. Пусть - регулярные выражения.  
   Тогда - регулярное выражение ( задаёт язык, задаёт язык, - объединение этих языков)  
    - регулярное выражение (аналогично, здесь произведение двух языков)  
    - регулярное выражение (итерация клини над языком)
4. Других регулярных выражений нет.

Обозначение: - класс регулярных языков над алфавитом .

**Опр.** Язык называется **регулярным**, если он может быть задан регулярным выражением

**Замечание:**

- нерегулярный язык.

**Опр.** Детерминированный конечный автомат (ДКА):

ДКА называется , где:

- множество состояний данного автомата (конечное),

- алфавит,

- функция переходов, , (по текущей позиции и букве - переходим в другую позицию)

- начальное состояние,

- множество конечных состояний,

Автомат можно рассматривать в виде ориентированного графа, вершины которого соответствуют состояниям, каждому ребру поставлена в соответствие некоторая буква из . Если из вершины в ведет ребро, которому поставлена в соответствие буква , то .

Заметим, что из любого состояния по любой букве есть переход.  
Если он не указан - подразумевается, что переход ведет в состояние ошибки.

Также автомат можно задать таблицей переходов, где по горизонтали отмечены буквы, по вертикали состояния, на перекрестье - состояние, в которое приведет от данных состояния и буквы.

**Опр.** Функция переходов на :

1. Пусть .  
   Пусть , где - буква.  
   Тогда

**Опр.** Пусть - ДКА (убрали входное и выходное состояние).

- синхронизируемый, если

Слово называют синхронизирующим.

**Задачка от Ананичева:**В городе Кентербере есть старинный замок а в нём живёт привидение.

Привидение умеет:  
1) Молчать.  
2) Хохотать

3) Ухать

4) Топать ногами

Посетители замка умеют:  
1) Включать музыку

2) Хлопать дверями

|  |  |  |
| --- | --- | --- |
|  | Хлопать дверями | Слушать музыку |
| Молчит | ухает | топает |
| Хохочет | топает | замолкает |
| Ухает | ухает | хохочет |
| Топает | топает | хохочет |

Нужно научиться заставить его замолчать из любого начального состояния.  
То есть, привести последовательность действий “хлопать дверями”/”слушать музыку”, которая из любого состояния приводит в состояние молчания.

Решение: 1 раз хлопнуть дверью, 2 раза слушать музыку, жить долго и счастливо и умереть в один день.

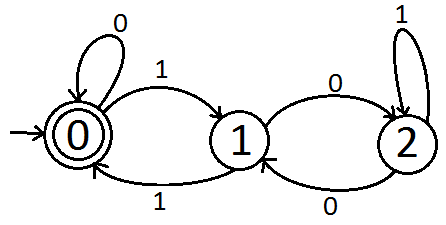
**Опр.** Слово распознается(принимается, accepted) автоматом , если

.

**Опр.** Язык, распознаваемый ДКА — это множество слов, которые принимает (распознает) автомат .

— язык распознаётся машиной .

Пример:

****

*(0 - начальное и конечное состояние)*

— это язык слов над , который представляет собой числа в двоичной записи: .

дают в остатке при делении на три 0

дают в остатке при делении на три 1

дают в остатке при делении на три 2

**Доказательство** (индукция по длине слова)**:**

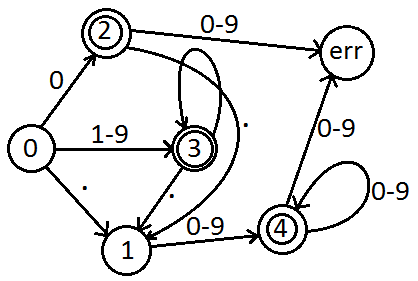
Б.И.

Ш.И.

Сл. 1

Пример 2.

L - язык десятичных чисел без знака xxx.yyy



**Опр.** Состояние смерти - такое, что по любой букве оно переходит в себя же.

Договоримся не обозначать его на иллюстрациях-графов, будем считать, что если для некоторой пары вершины и буквы соответствующий переход не изображен, то этот переход ведет на смерть.

**Опр.** Язык называется рациональным если существует ДКА, распознающий этот язык, т.е.

Обозначается

Связь рациональных языков и операций над языками:

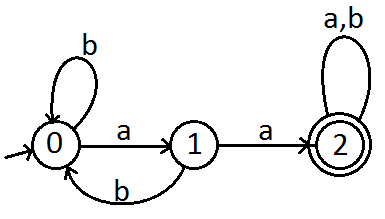
Пусть языки

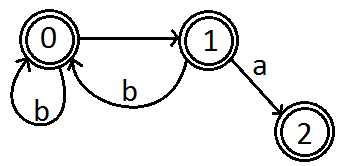
Тогда:

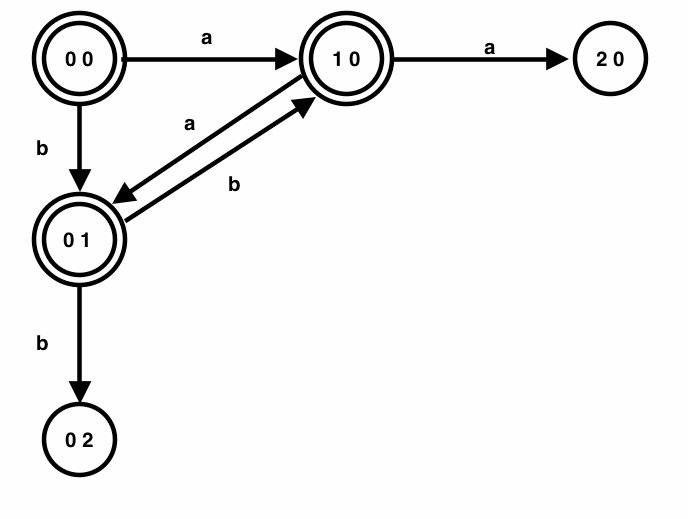
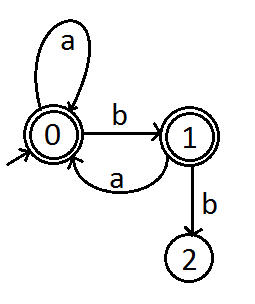
1. **Доказательство:**  
   Так как - рациональный, то   
   Построим КДА   
   Очевидно, распознает .
2. **Доказательство:**  
   Пусть распознают , т.е.:  
     
     
   Построим автомат как декартово произведение :  
   .  
     
   .  
   Нетрудно видеть, что распознает .

L - язык слов над , в кот. нет

= нет , = есть





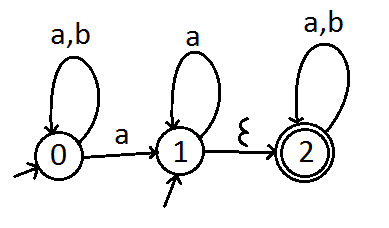


Суть:

Могут быть несколько входных состояний.

Из одного состояния по одной букве могут вести сразу несколько ребер, включая 0.

А еще ребра могут быть подписаны , тогда по ним можно переходить в любой момент.



**Опр.** Слово принимается автоматом, если хотя бы одно из состояний, в которые приводит слово, является конечным.

**Опр.** -НКА называется

- множество состояний.

- входной алфавит.

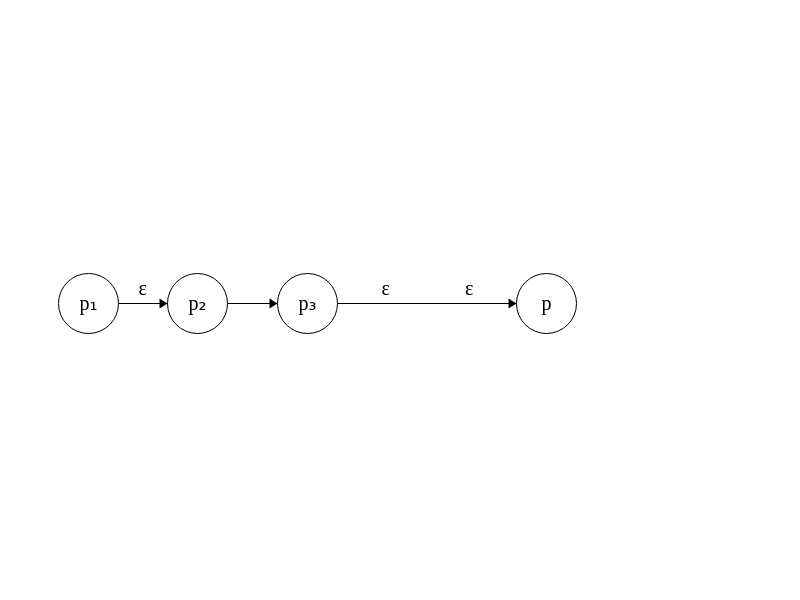
- множество переходов.

- множество входных состояний, .

- множество выходных состояний, .

- состояние в которое приходим

**Опр.** - замыкание - все состояния, в которые мы можем попасть из , двигаясь только по -ребрам (рисунок).



-то что на картинке только словами(как попасть в состояние )

**Опр.** По индукции, как действует автомат на

БИ:

ШИ:

и . (Здесь мы используем последний шаг, т.к. лежит в замыкании )

Этот автомат нужен, чтобы он принимал или не принимал слова.

Untitled Diagram.png

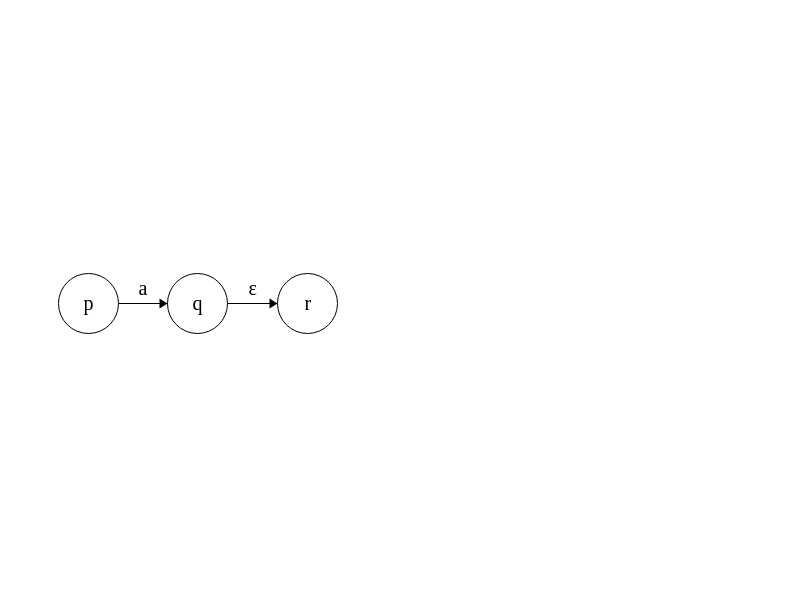
**Опр.** -НКА принимает слово ,￼ если

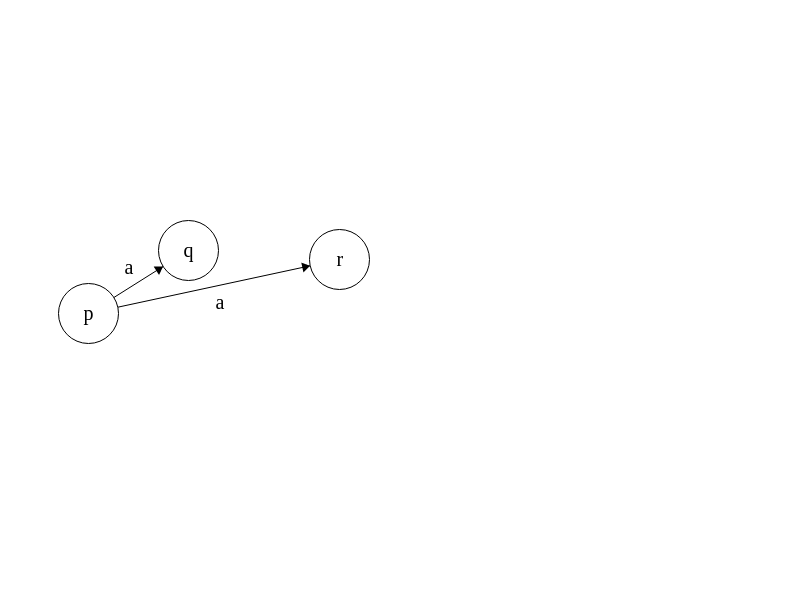
— множество всех слов, которые принимает автомат.

**ЕЁ любимое Опр.**  и — эквивалентны, если

**Лемма** Пусть -НКА. Тогда существует эквивалентный ему НКА с тем же множеством состояний.

**Доказательство:** Че происходит:





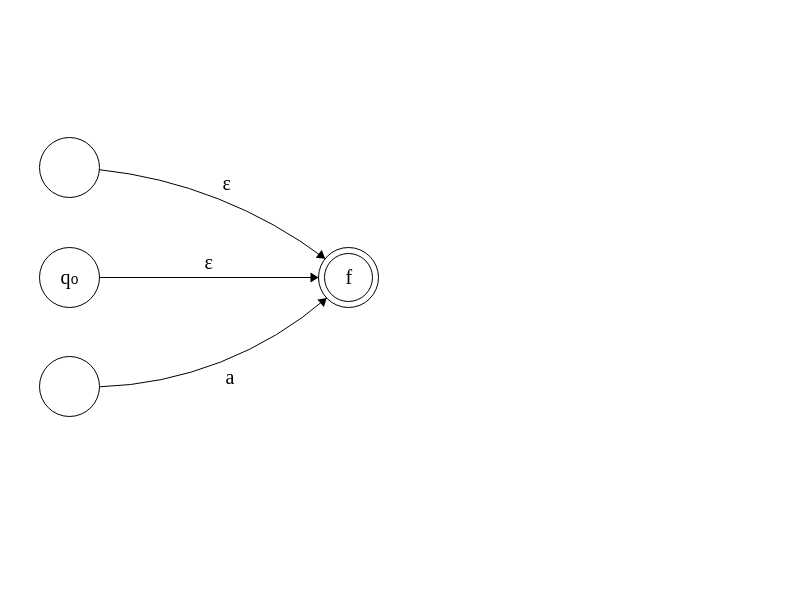
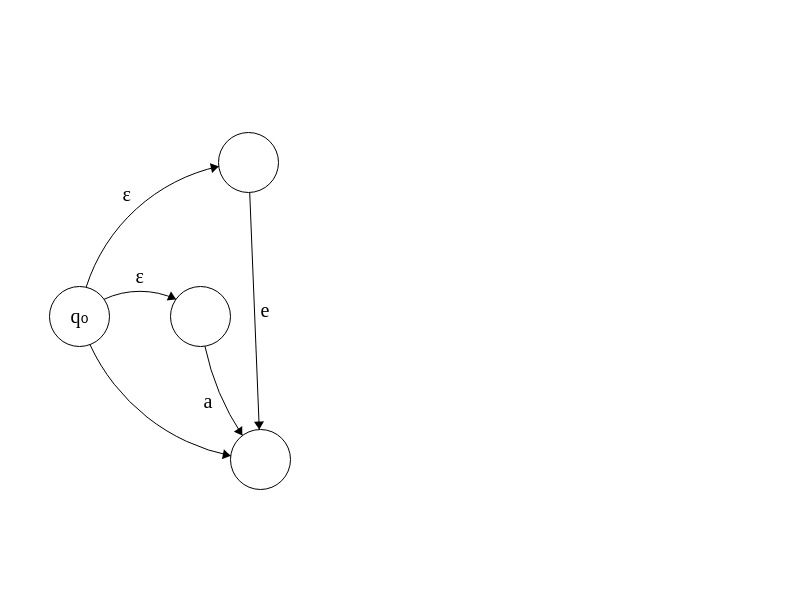
Строим .

Будем считать, что (халтурка)

- множество - переходов в А

(мы будем считать, что это множество транзитивно, то-есть) .

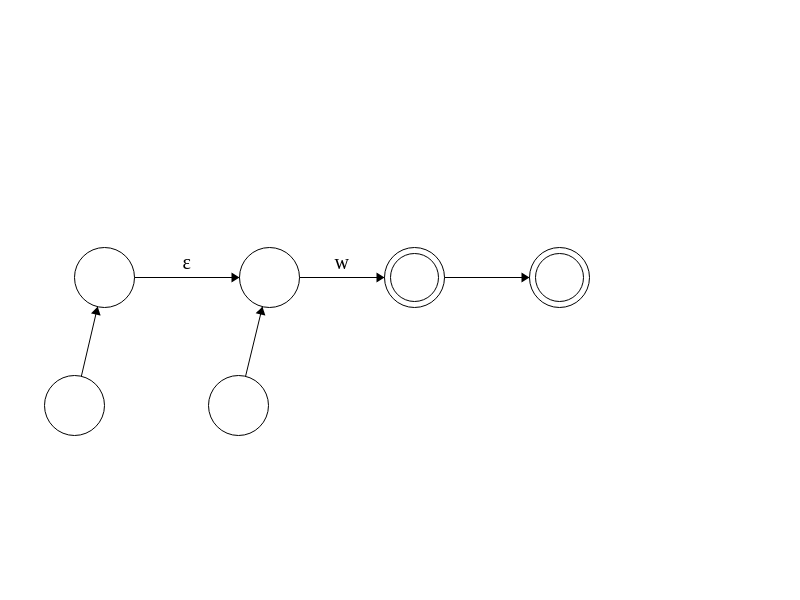
- из множества переходов исключаем переход



- В новом автомате. А в старом автомате прочитали и пришли в конечное .

**Доказать: .**

На рисунке: Кто-то из может быть .



Идея: если есть НКА то можно построить ДКА по нему

**Теорема Рабина-Скотта. (Незнание = казнь через повешение + расстрел на 4 курсе)**

Пусть - НКА, тогда существует эквивалентный ему ДКА

**Доказательство:**

, где - одно состояние

Идея детерминирования автомата: мы объединяем некоторые множества состояний в одно состояние .

Untitled Diagram (1).png

— все те состояния в которые мы можем попасть по букве .

1. Покажем, что - множество всех тех состояний, в которое мы можем попасть из прочитав

БИ не доказываем.

ШИ. По длине .

— прочитали и попали в состояние, потом читаем

.

— из состояния мы попадаем по букве в .

+

, где и



**Пример 1.** (Пример построения по )

Дано НКА без переходов по - для простоты. (это )

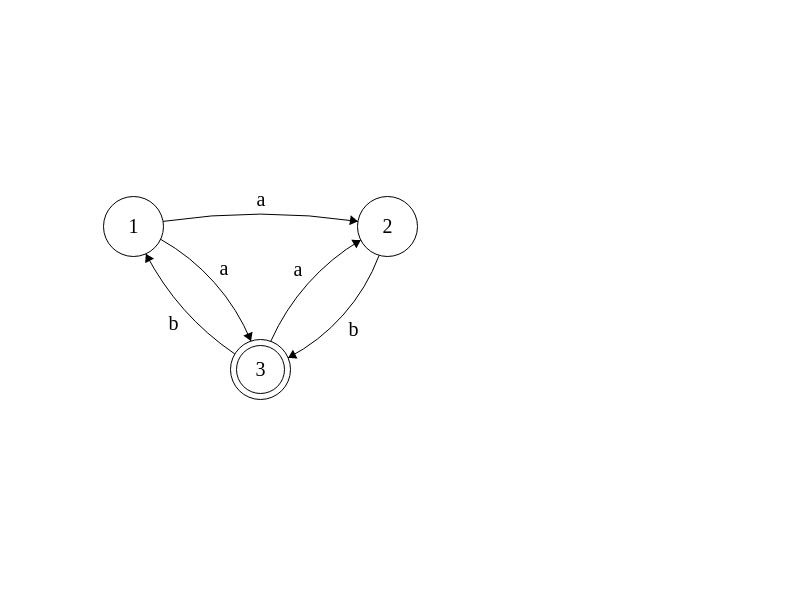
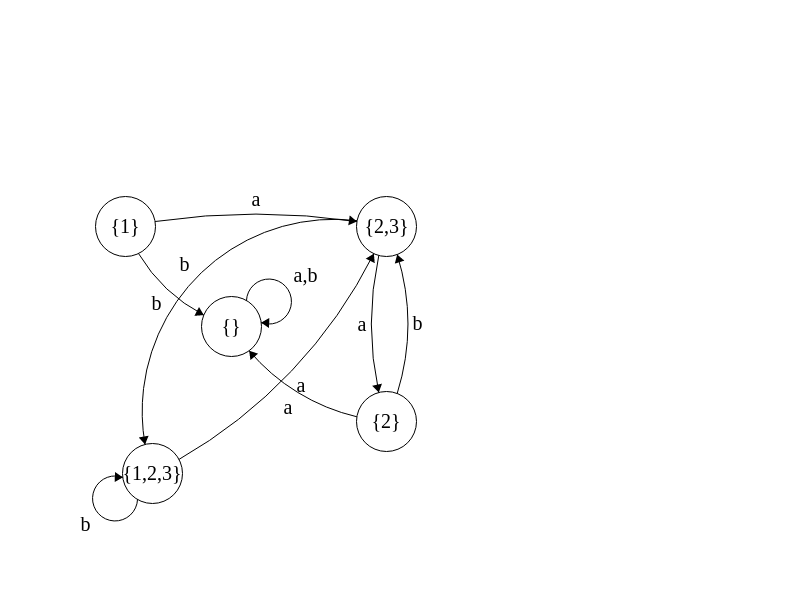


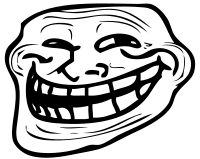
Таблица переходов в

|  |  |  |
| --- | --- | --- |
| Мн-во сост |  |  |
|  |  |  |
|  |  |  |
|  |  | {2, 3} |
|  | {2, 3} | {2, 3, 1} |
|  |  |  |

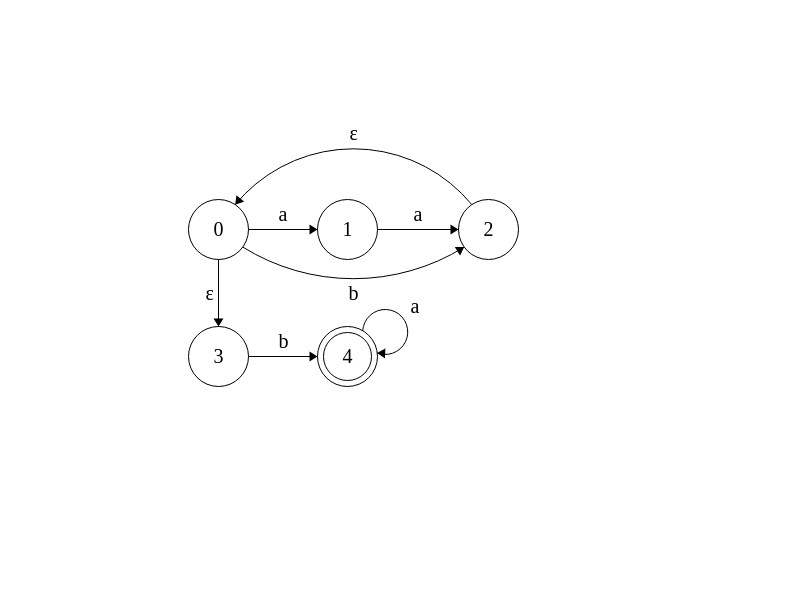
Рисунок автомата В на множествах



**Пример 2.** Как же детерминировать -НКА



(тут опять рисунок , похоже, рисунка не будет)

****

Тут есть какая-то идея и она очень идейная…

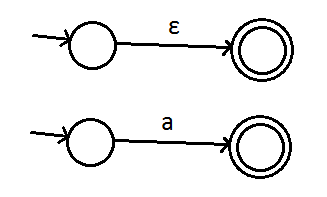
|  |  |  |
| --- | --- | --- |
| Множ. сост. | а | b |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

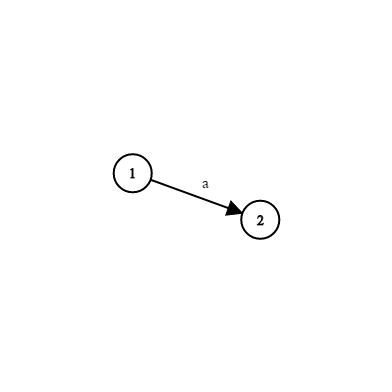
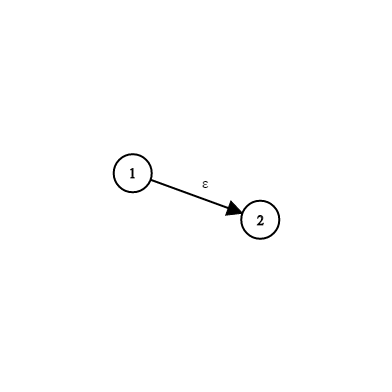
**Замечание: (Следствие из теоремы)**  Чтобы доказать, что язык рационален

Достаточно построить -НКА, который распознает язык

**Теорема (Клини).**

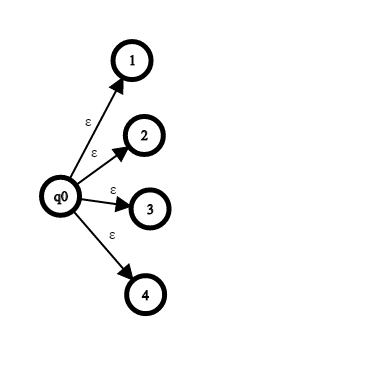
**Доказательство:**

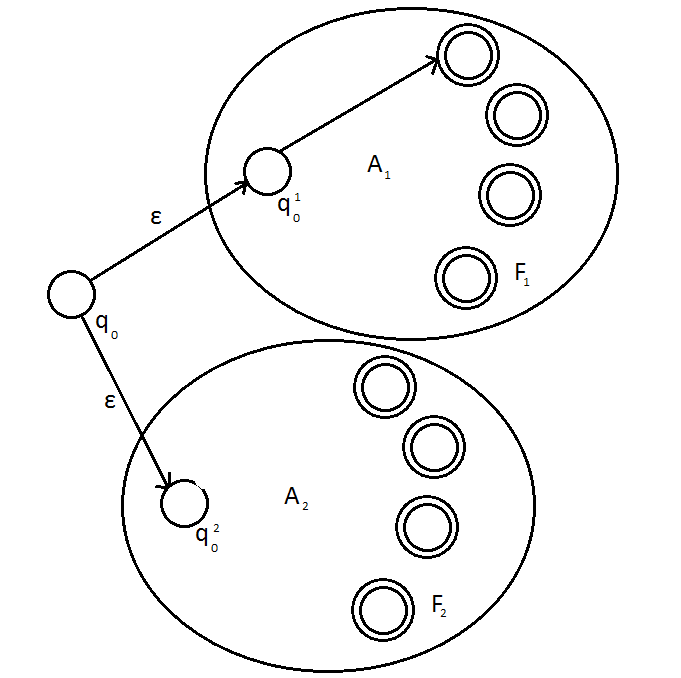




1)По определению. Докажем, что и - регулярные выражения.

2) Пусть есть , , , , ,





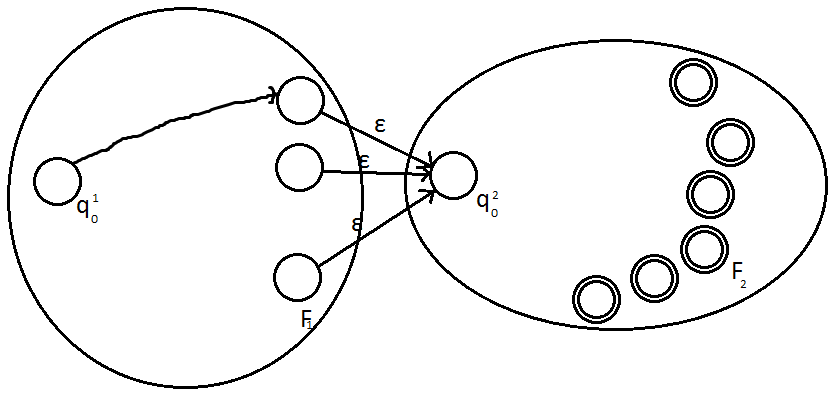
“Если попадаем в первый автомат, во второй никогда в жизни не попадем”

Пусть , , Б.о.о: , , ,

, Б.о.о. ,

Тогда

Запишем произведение двух языков

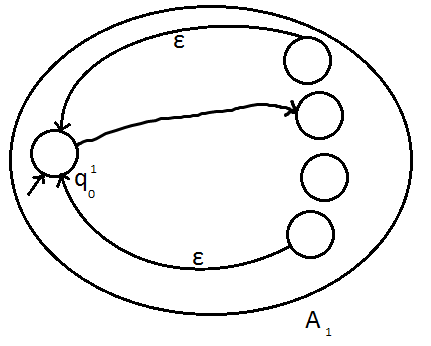


,

,

)

:



\*рисовашки :3\*

Итак. -НКА

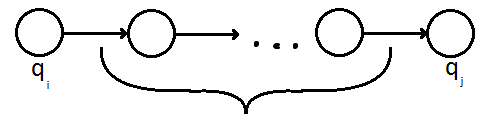
-ДКА

Покажем, что язык, распознаваемый ДКА - регулярный.

Пусть - ДКА

Докажем, что

Все промежуточные состояния должны иметь номера от до .



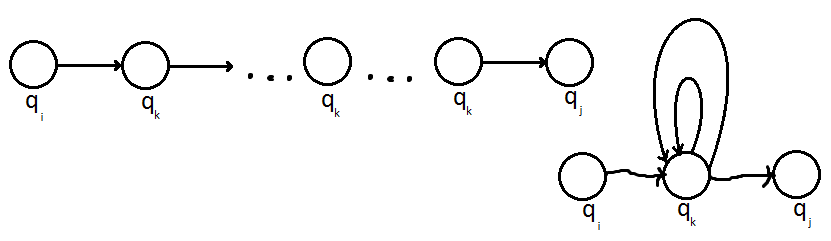
- без промежуточных состояний

- промежуточные состояния любые(в т.ч и 0)

- это хотим доказать

Докажем индукцией по параметру .  
Б.И.:

\*тут рисуночек - петля \*

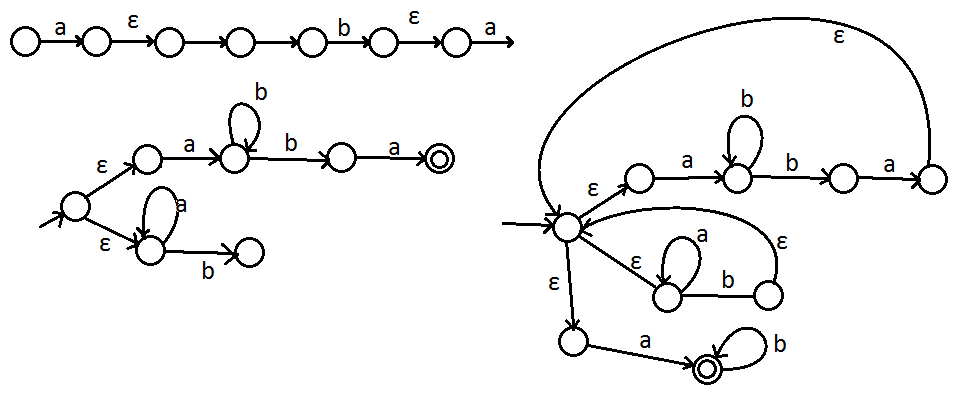


Ш. И. Предположим, что

тут всё регулярные выражения

**Примеры**

1. “Как построить по регулярному выражению?”  
   , , , , , , ,



1. “Как по автомату построить язык, который он распознает?”

**Лемма** фикс,

Тогда

Доказательство:

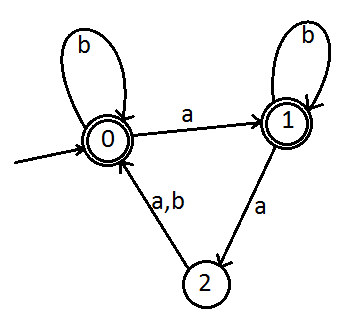
Пусть - решение уравнения

Докажем, что

О/п : но

Б.о.о -кратчайшее, ( потому как быть не может)

(тут пришли к противоречию о форме )



1. “Построить по

“Тут рисуночек, к которому не стоит применять теорему Клини”

, ,

(три уравнения-система)

1q3r5u89

Iuyt1r `\

=-0kolm.x

, ,

,

1. Если есть язык - можно из него получить НКА ( состояний), из которого по теореме Рабина-Скотта можно получить ДКА (состояний)
2. регулярное выражение(которых может быть много)

Минимальный ДКА (среди всех, задающих некоторый язык) - ДКА с минимальным количеством состояний.  
Проблема - как построить минимальный ДКА?

## Моноиды и автоматы.

Пусть есть множество с заданной операцией .  
 - моноид, если это полугруппа, в которой есть нейтральный элемент.  
Более строго:

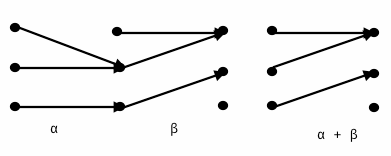
1. - нейтральный :

Возьмём

Рассмотрим множество всех преобразований .

Известно, что - полугруппа относительно операции композиции.

На самом деле, это даже моноид, потому что тождественное преобразование - это единица.



Только что мы вспомнили некоторые факты из алгебры и дискретной математики. А теперь давайте вернёмся к автоматам.

Пусть дан автомат - ДКА.

)

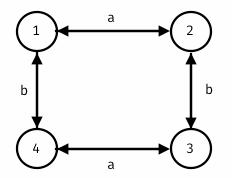
Если зафиксировать некоторое , то .

Также, если , то аналогично можно при фиксированном сказать, что

- нейтральный элемент.

- композиция преобразований.  
( - моноид переходов

**Пример:**



Давайте поисследуем этот автомат.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
|  | 1 | 2 | 3 | 4 |
|  | 2 | 1 | 4 | 3 |
|  | 4 | 3 | 2 | 1 |
|  | 1 | 2 | 3 | 4 |
|  | 1 | 2 | 3 | 4 |
|  | 3 | 4 | 1 | 2 |
|  | 3 | 4 | 1 | 2 |
|  | 4 | 3 | 2 | 1 |
|  |  |  |  |  |

Можно заметить, что все квадраты уничтожаются.  
Значит разных элементов всего 4.

Составляем таблицу Кэли:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| ab |  |  |  |  |

Это называется моноидом переходов.

Вопрос:  
Моноид переходов произвольного ДКА - конечный или бесконечный?  
Ответ:  
Конечный, ведь существует лишь конечное количество преобразований.

## 

## 

## Распознаваемость моноидами и автоматами.

**Опр.** Пусть есть два моноида: ,

Тогда - гомоморфизм, если

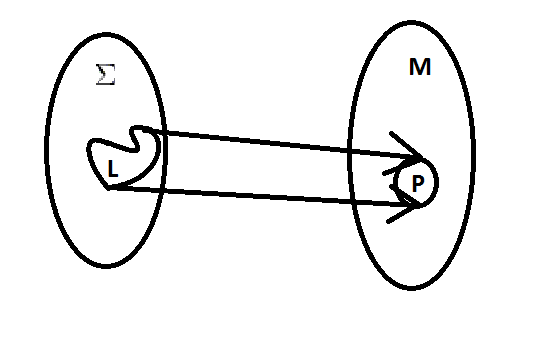
можно продолжить(как мы и сделали в случае автоматов) единственным образом до гомоморфизма :

**Опр.**   
Пусть есть:  
-моноид, .

- гомоморфизм.

Тогда:

распознаётся тройкой , если



**Теорема 1.**Пусть и .

Тогда распознаётся [он конечен!]

**Док-во:**

Положим

- гомоморфизм.

очевидно (по определению)

Пусть

Это равенство выполняется .

Значит, имеем право подставить любое состояние, которое нам нравится.

Подставим .

.

**Теорема 2**

Пусть и распознается конечным моноидом

Тогда ДКА , такой что .

**Док-во:**

Строим автомат

распознается .

Осталось доказать, что

Возьмём

(- конечное)

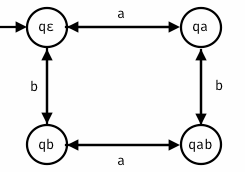
Язык рационален он распознаётся конечным моноидом.

распознаётся конечным моноидом.

Рационален - значит распознаётся ДКА.

**Пример.**  
Используем предыдущий пример

Был моноид из 4 элементов:

Значит будет 4 состояния   
****

Куда мы перейдём по букве из ?  
Нужно взять и умножить на .

Значит

Аналогично

Собственно, мы построили автомат.

**Предложение 1**

Пусть - гомоморфизм.

Тогда

**Док-во:**  
-конечный моноид.

распознает

- распознает

**Предложение 2**

Пусть

Тогда ,

**Док-во:**

(вспоминаем что такое )

распознаётся

Докажем, что распознают

## 

## 

## Минимальные автоматы

**Опр.**

- автомат левых частных

**Предложение 1**

Автомат распознает L

Доказательство:

**Опр.**

- достижимое (*accessible*)

Если ,

**Опр.** Автомат достижим(trim), если все его состояния достижимы.

**Предложение 2**

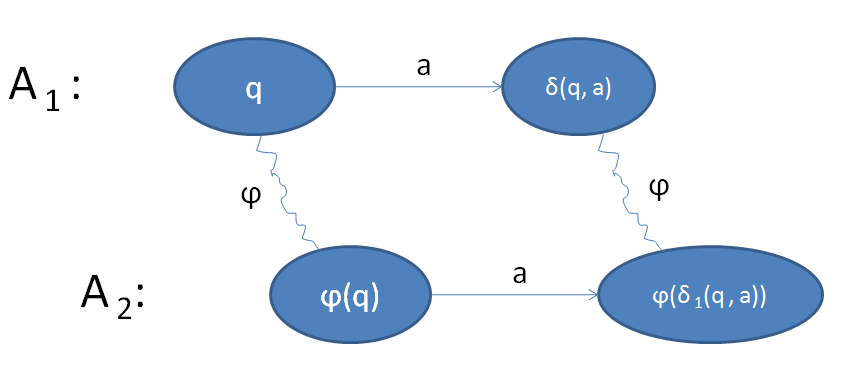
- достижим

## 

## 

## Гомоморфизм автоматов

*“Слышишь слово “гомоморфизм”-беги.”*



Отображение - гомоморфизм автоматов, если

**Теорема:** (Myhill-Nerode)

Пусть - ДКА распознает и он достижим.

Тогда - гомоморфизм автоматов.

и

1. - сюръекция

**Доказательство:**

Определим бинарное отношение на

*“Ща мы будем доказывать много пунктов.”*

1. Докажем, что - отображение.  
     
    - достижим   
   От противного,
2. Докажем, что - гомоморфизм автоматов.  
    хотим доказать:   
     
     
     
     
   Доказали.
3. Докажем, что - сюръекция  
   Докажем, что у неё есть прообраз:   
     
   Возьмем
4. Действительно,
5. ,

Свойство сюръекции:

- сюръекция, .

**Следствие:**

**Опр.** ДКА называется минимальным, если в нем наименьшее число состояний среди тех автоматов, распознающих тот же язык.

- минимальный по количеству состояний ДКА, распознающий тот же язык

**Пример:**

Построить минимальный автомат для языка

(Это язык всех слов, содержащий как подстроку )

**Замечание:**

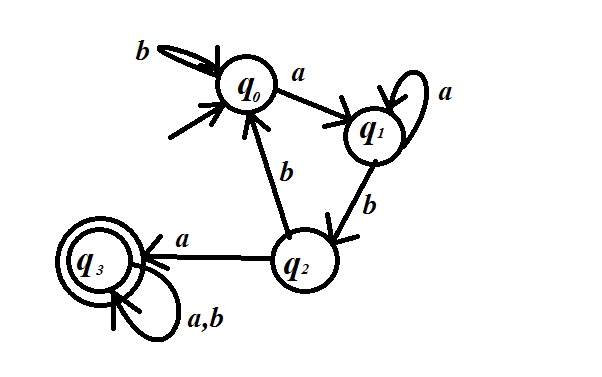
- новое, потому что слово принадлежит , но не принадлежит .

Значит это состояние уже было и оно нам не нужно.

- опять не нужно

- новое, т.к. не принадлежит и .

- новое, т.к. пустое слово не принадлежит



— моноиды.

**Утверждение:**

— конгруэнция

**Опр.** — моноид; — конгруэнция.

Рассмотрим , (*превратим в моноид*)

Определим операцию:

- класс эквивалентности

(по определению)

Докажем, что это корректно.

и .

Тогда для первой эквивалентности и для второй домножим на

.

Значит, .

Нейтральный элемент:

изоморфно

Синтаксический моноид языка L.

**Опр.**

Утв. - конгурэнция.

.

- синтаксическая конгруэнция L.

— синтаксический моноид языка L.

Рассмотрим множество слов

Значит,

Утверждение.

**Утв.**

**Доказательство.**

Введем

**Теорема.** (о гомоморфизме, без доказательства)

— сюръективный гомоморфизм**.**

Следовательно, изоморфно .

Продолжаем прошлое доказательство

Докажем, что

.

.

**,** в том числе и для начального состояния .

Следовательно,

.

Нужно доказать, что

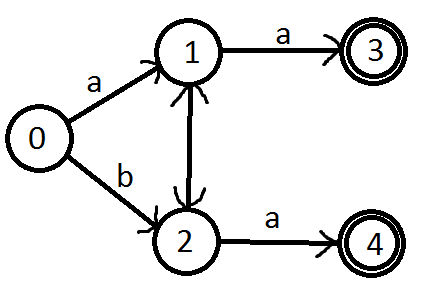
Тогда , .

Другой алгоритм минимизации автомата

Что у нас есть? Есть теорема:

Теорема : — ДКА, распознающий .

Тогда — сюръективный гомоморфизм.



, .

Надо уметь склеивать состояния, но так, чтоб автомат не ломался. Сломанный автомат — это плохо.

**Опр.** - ДКа

— конгруэнция автоматов.

— отношение эквивалентности.

Нам нужна стабильность.

.

.

Рассмотрим

.

— гомоморфизм.

Докажем, что — конгруэнция автоматов.

**Опр.**

конгруэнця

— факторавтомат.

— определение.

**Докажем корректность.**

, по определению конгруэнции

**Теорема.**

Пусть — автомат достижим. Пусть , — конгруэнция. Тогда .

**Доказательство**

,

**Алгоритм.**

Идея.

Пусть — построено.

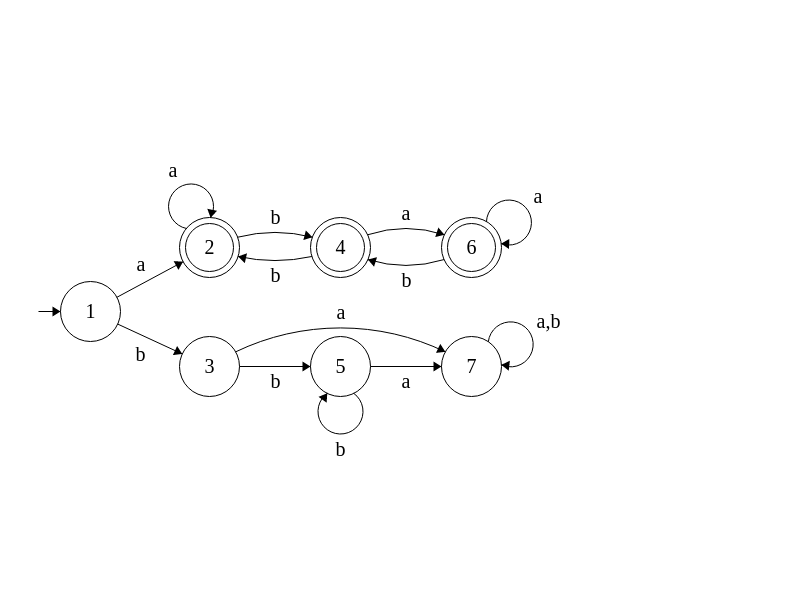
эквивалентно системе:

Останавливаемся при

Покажем, что если — конгруэнция

— конгруэнция.

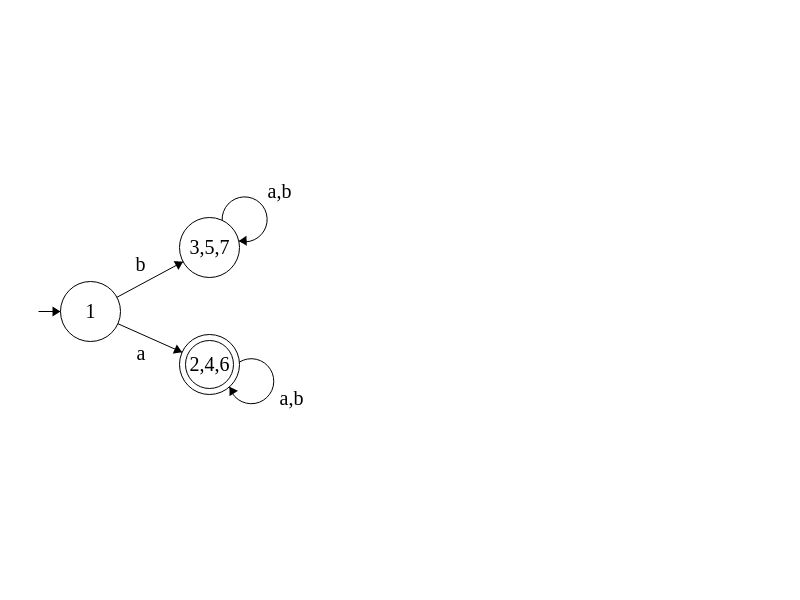
*Пример 1.*



|  |  |  |
| --- | --- | --- |
|  | 1 3 5 7 | 2 4 6 |
| a | 2 7 7 7 | 2 6 6 |
| b | 3 5 5 7 | 4 2 4 |

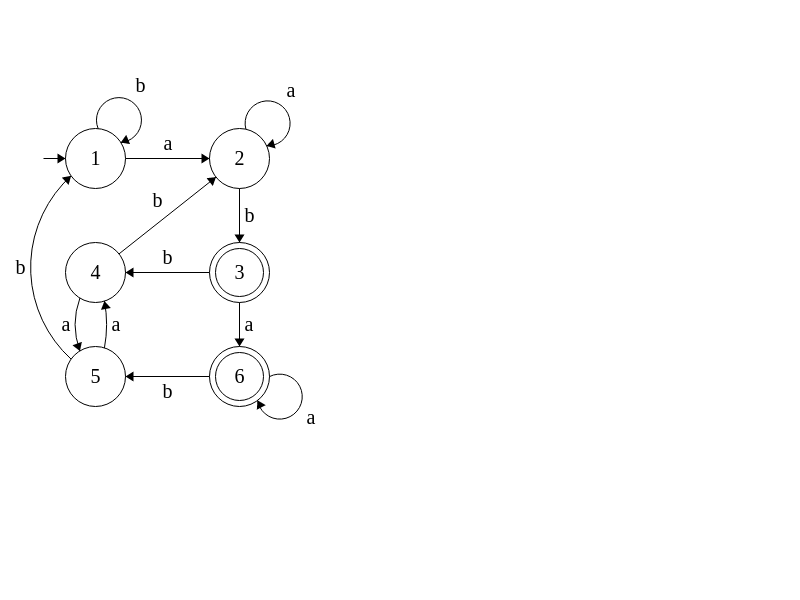
|  |  |  |
| --- | --- | --- |
|  | 1| 3 5 7 | 2 4 6 |
| a | 2| 7 7 7 | 2 6 6 |
| b | 3| 5 5 7 | 4 2 4 |

Тк классы переходов совпадают, значит

, конечное состояние: 2,4,6

В итоге этот автомат равен регулярному выражению “*a(a+b)\**”.

**Пример 2.**



|  |  |  |
| --- | --- | --- |
|  | 1 2 4 5 | 3 6 |
| a | 2 2 5 4 | 6 6 |
| b | 1 3 2 1 | 4 5 |

Делим по букве b

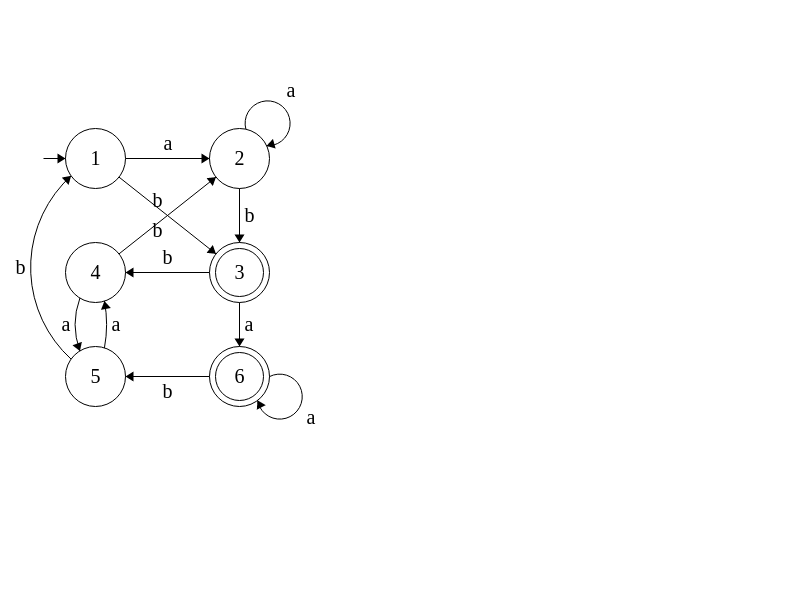
|  |  |  |
| --- | --- | --- |
|  | 2| 1 4 5 | 3 6 |
| a | 2| 2 5 4 | 6 6 |
| b | 3| 1 2 1 | 4 5 |

|  |  |  |
| --- | --- | --- |
|  | 2| 1| 4| 5 | 3| 6 |
| a | 2| 2| 5| 4 | 6| 6 |
| b | 3| 1| 2| 1 | 4| 5 |

Факторавтомат совпадает с исходным, то есть это минимальный  
С помощью этого алгоритма можно доказать что автомат минимальный

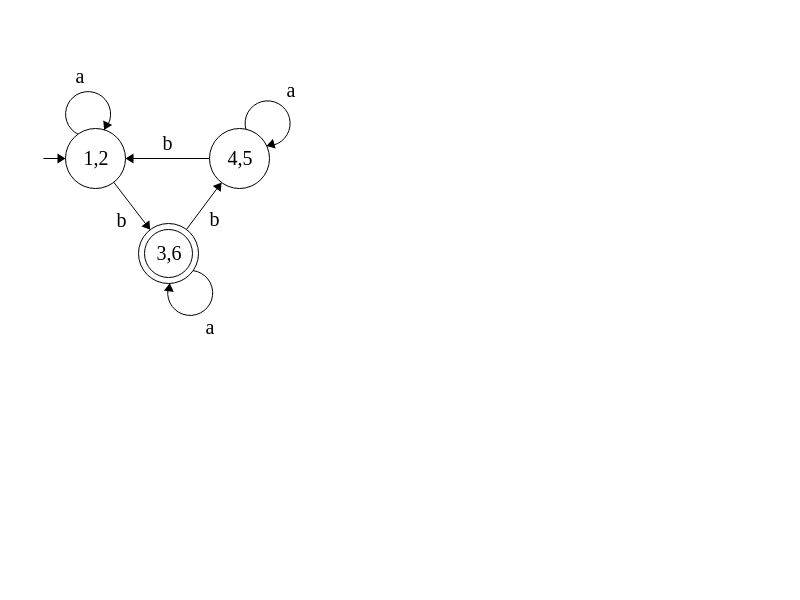
Каждое состояние находится в своем классе. Или что то подобное

**Пример 3.**

****

|  |  |  |
| --- | --- | --- |
|  | 1 2 4 5 | 3 6 |
| a | 2 2 5 4 | 6 6 |
| b | 3 3 2 1 | 4 5 |

|  |  |  |
| --- | --- | --- |
|  | 1 2| 4 5 | 3 6 |
| a | 2 2| 5 4 | 6 6 |
| b | 1 3| 2 1 | 4 5 |

Тогда у нас 3 состояния:  


Это минимальный автомат. Было 6 стало 3.

При этом у нас только одно выходное состояние : (3, 6)

**Рассмотрим:**

— лол нет.  
Почему? А потому.

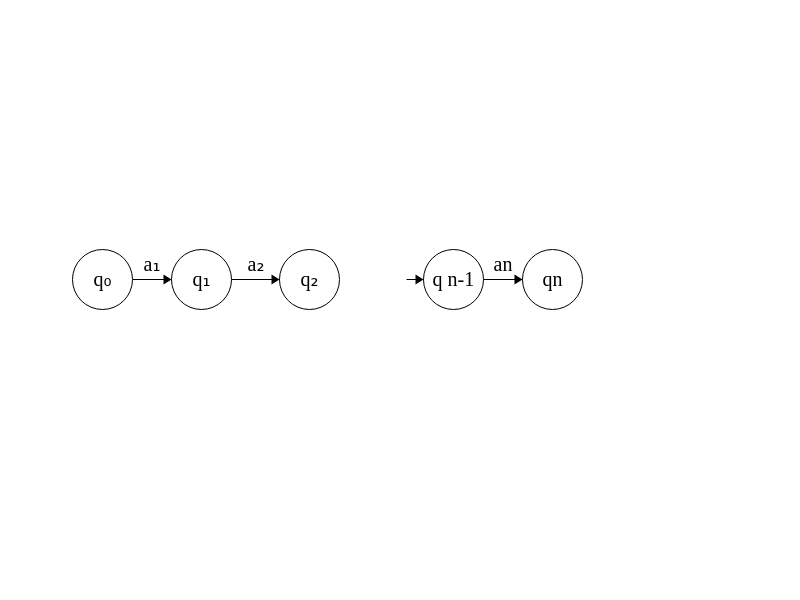
**Лемма(**О накачке для рациональных языков) - *pumping lemma.*

Пусть , тогда

**Доказательство.**

1 случай. — конечный язык, тогда в этом есть самое длинное слово все выполняется

2 случай. — бесконечный язык,



У нас (n+1) состояние в этом пути . Будем считать что i, j - наименьшие с таким свойством. На пути , где это префикс x, - суффикс z, a это слово y. Получили xyz. Так вот, на пути есть только 2 повторяющиеся сост. , значит

Пример 1. Доказать, что — не рациональный.

От противного. Пусть рациональный, то выполняется лемма о накачке, значит существует слово из этого языка, для которого условия леммы о накачке не выполняются

1. Выбрать хороший пример. Берём . Тогда . Должно быть размещение .

Берем k=2

Пример 2.

О/п. рациональное => лемма о накачке

контрпример

- слово

**Что должен знать каждый уважающий себя студент:**

1. Т. Клинни Она строит автоматы - детерминированные
2. Есть — НКА Детерминированный
3. ДКА минимальный
4. \* регулярные выражения, значит ли это, что ?????

## 

## 

## 

## Трансдьюсеры

*Суть: есть автомат. На каждую букву, которую вы ему даете, автомат дает букву вам. При этом вы не видите, в каком состоянии находитесь, т.е. автомат - черный ящик.*

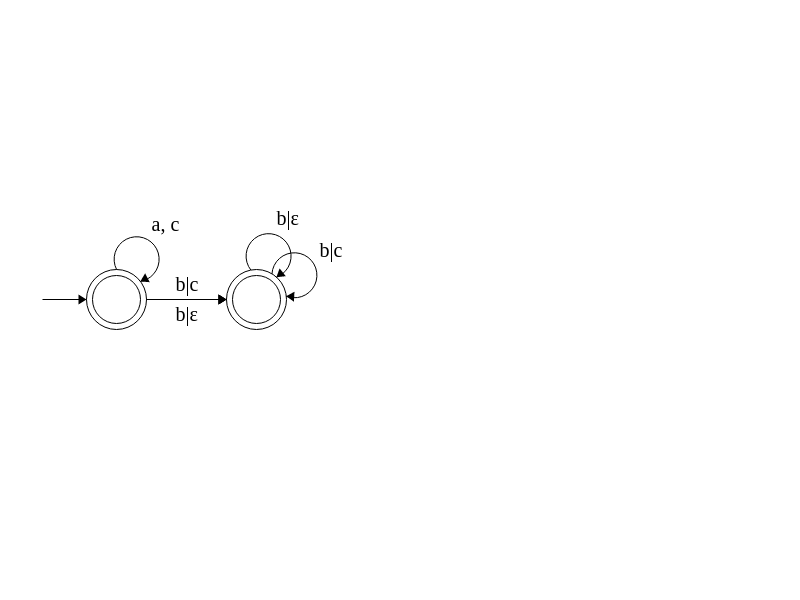
Пример: Вход: Выход: все заменим на .

Пример 2:

в ответ

Пример 3: Вход: Выход:

Как выглядит рисунок трансдьюсера:



**Опр.**

Трансдьюсер(Transducer) - набор

где:

- Входной алфавит

- Выходной алфавит

- Функция переходов автомата; *(как и раньше, обобщаем на )*

- Выходы автомата;

*(т.е. выход - набор из состояния, буквы, которую вы автомату “скормили” и буквы, которую вы на это получаете)*

(Остальные символы имеют прежний смысл)

**Опр.** Трансдьюсер распознает (принимает) пару , если .

**Опр.** Пусть .

распознает , если распознает

**Опр.** Назовем рациональным, если трансдьюсер, распознающий этот язык.

## Рациональные отношения

**Опр.** Проекии:

**Замечание:**

Пусть - рационально

**Опр.** Рассмотрим отношение ,

- *отношение равенства*.

Докажем, что оно рационально.

Доказывается построением тривиального трансдьюсера (что принимает, то и отдает)

Следовательно .

**Опр.**

Возьмем произвольные

Определим на них некоторые операции:

Эти операции называют *рациональными*. Потому что для рациональных аргументов возвращают рациональный результат.

А вот пересечение и дополнение рациональными операциями не являются:

Пример: -рациональны, но - не рациноальны

Рассмотрим первую проекцию пересечения:

Не рациональный язык! Тогда и не рационален.

НО! [x2]

В конкретном случае пересечение все же может оказаться рациональным.

*Учебник по трансдьюсерам:*

*Juhani Karhumåki (это автор)*

**Пример**

Пусть - отношение равенства.

Дополнение к нему - отношение неравенства. *(социального, кек, кек[2], кек[мишаня])*

Оно рационально, можно построить автомат.

**Утверждение:**

Пусть - рациональный язык над .

- гомоморфизм (т.е. )

Внезапное наблюдение - слово есть произведение букв

Если вы еще живы, продолжайте читать…

Рассмотрим следующее отношение:

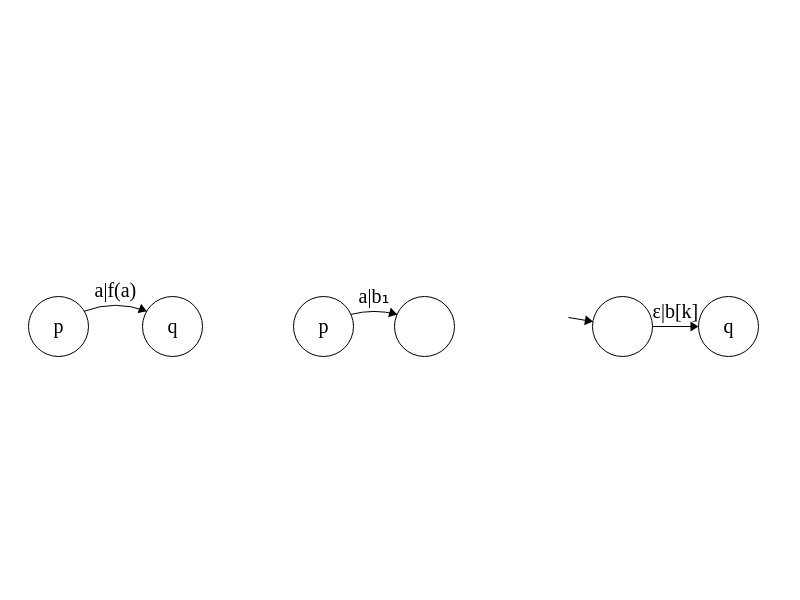
Докажем, что оно рационально.

Так как - рационален, то - ДКА, распознающий .

Сделаем из этого автомата трансдьюсер.

Пусть в этом автомате есть пара состояний , соединенные ребром .

Пусть .  
Добавим в наш автомат несколько состояний, чтобы получилось так:

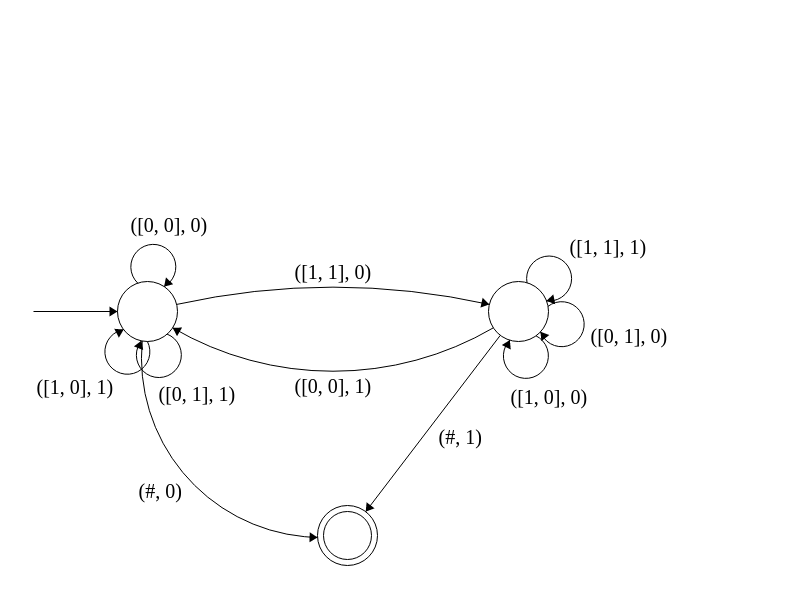


А что еще умеет трансдьюсер? И кто его этому научил?

Он умеет складывать! Давайте научим его этому.

Двоичные числа(записаны задом наперед, одинаковой длины, заканчиваются на спец символ #) *(не нравятся наши двоичные числа - придумай свои)*  
Тогда на вход мы получаем набор пар двоичных цифр и # в конце.

И вот как выглядит нужный трансъдьюсер:



А вот пример трансдьюсера, возводящего в квадрат двоичные числа вида:

То есть их квадраты имеют вид:

- должен быть рациональным по теореме Нива (которую мы еще не доказали). Увы, на самом деле он не рациональный(((. Можно доказать по лемме о накачке.

**Теорема** Нива

Отношение рационально тогда и только тогда, когда:

алфавит, рациональный язык

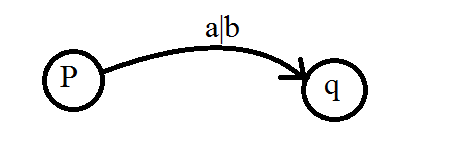
1. *, где 1 и 2 - гомоморфизмы*

**Доказательство:**

=>:

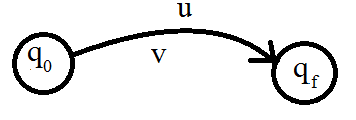
Пусть отношение рационально. Тогда , распознающий .

Выберем алфавит



Пусть - множество пар букв, между которыми нет перехода.

- множество “правильных” переходов



-начинающиеся в начальном состоянии.

-заканчивающиеся в конечном состоянии

- слова, которые начинаются в начальном состоянии, заканчиваются в конечном и все переходы корректны.

.

-

Рассмотрим слово

<=

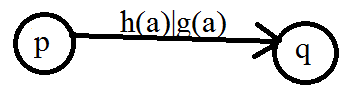
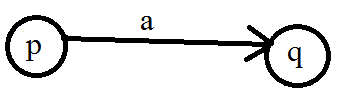
Пусть:

алфавит, рациональный язык

1. *, где 1 и 2 - гомоморфизмы*

Докажем, что рационально, т.е. Для него можно построить трансдьюсер.

Рассмотрим некоторую пару . Нам нужно, чтобы автомат распознавал пару . Для этого достаточно, чтобы для каждой буквы автомат распознавал пару .



Пусть:

при этом (без ограничения общности).

Возьмем пару вершин такую, что переводит в . Добавим промежуточные состояния так, чтобы пара переводили в (где ).

Получим трансдьюсер, распознающий .

**Теорема доказана.**

**Следствие:**

Пусть рациональный

Тогда - рациональный.

**Док-во:**

для некоторого .

-рациональный

рациональный язык рациональный

## 

## 

## Вопросы разрешимости

**Проблема 1:**

Верно ли, что если , то можно эффективно проверить, что ?

Допустим, мы умеем задавать вопрос, принадлежит ли какое-нибудь слово. Тогда, получив ответ да, мы будем знать, что не пуст. Получив же ответ нет, мы не продвинемся в решении нашей проблемы(((.

**Проблема 2:** Пусть языки . Верно ли, что ?

**Проблема 3:** Пусть языки . Верно ли, что ?

**Теорема**

Проблемы 1, 2, 3 - разрешимы.

**Док-во:**

Проблема 1:  
Построим автомат, распознающий . Теперь вопрос сводится к тому, достижима ли из . Это мы проверять умеем.

Проблема 2:

Заметим, что множества равны, если их симметрическая разность (являющаяся рациональным языком) пуста. Свели к проблеме 1.

Проблема 3:

, если . Свели к проблеме 1.

Напоминаем определение синхронизируемого автомата:

**Опр.** Пусть - ДКА (убрали входное и выходное состояние).

- синхронизируемый, если

Слово называют синхронизирующим.

Пусть . Обозначим .

Тогда - синхронизируемый - переводит все состояния в 1 единственное

Возникает вопрос, как проверить, что автомат синхронизируемый?

**Теорема**

Существует алгоритм, проверяющий, синхронизируем ли автомат.

**Док-во:**

Пусть - ДКА.

Построим автомат (где - булеан ).

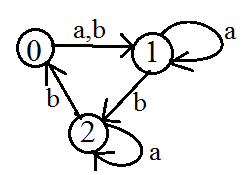
Возьмем вот так:

Тогда - синхр .

:

Тогда исходная проблема эквивалентна тому, чтобы проверить, что в из позиции достижимо какое-нибудь одноэлементное множество.

**Пример:**



Задачу и поиске синхронизирующего слова здесь стоит решать, рассматривая автомат с 8 состояниями - элементами булеана множества состояний исходного автомата, и проверяя, достижима ли какое-то состояние из .

**Утверждение:**

Автомат - синхронизируемый

*(т.е. Любые два состояния автомата можно “склеить” каким-нибудь словом)*

**Доказательство:**

:

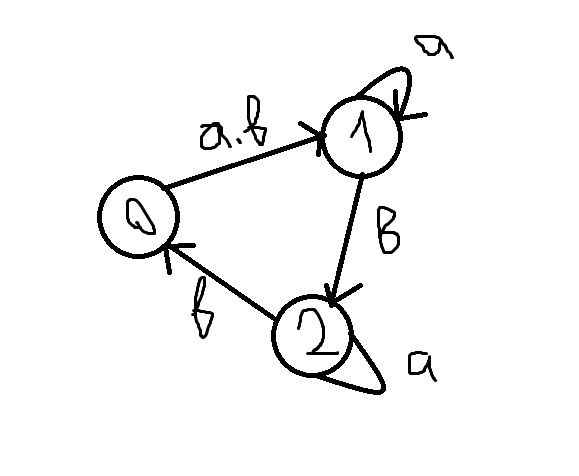
Очевидно, если автомат синхронизируем, то любые его два состояния можно “склеить” (т.к. Их все можно склеить).

:

Построим синхронизирующее слово:

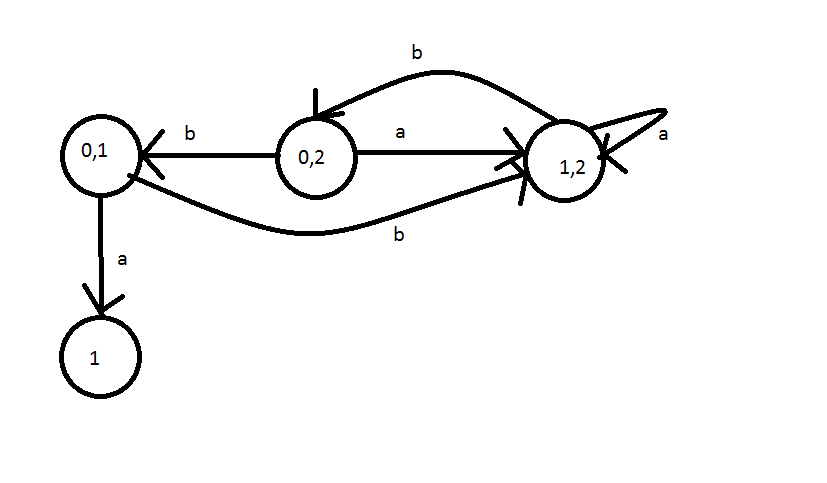
Выберем пару состояний в , для них найдется “склеивающее” слово , тогда объявим , при этом в состояний меньше, чем в . Если в одно состояние, то автомат синхронизирован, иначе в вновь найдется пара различных состояний и слово для них. Получаем , в котором меньше состояний, чем в . И так далее. В итоге получим множество с одним состоянием и итоговое синхронизирующее слово будет .

**Пример:**



В данном случае сначала склеим 0 и 1 словом , затем 1 и 2 словом . В итоге синхронизирующее слово .

**Другой пример:** (Для самостоятельного решения :D)



### Алгоритм Ахо-Корасик

**Задача:**

Задано - текст

, где - шаблон. (т.е. тоже некоторые слова)

Требуется найти множество пар , таких, что начиная с позиции в тексте начинается шаблон .

**Частный случай:**

(Алгоритм Кнута-Морриса-Пратта) *(Вспоминаем Скрипты, первый семестр)*

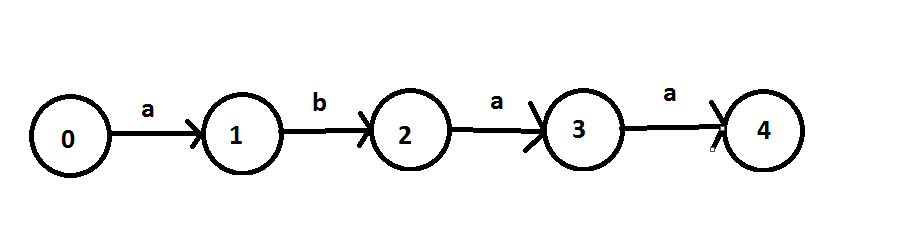
Шаблон единственный, т.е.

Построим автомат, который будет выдавать номера начал вхождений.

**Пример:**

, . (по методу пристального взгляда: 2 вхождения, с 3 и 6 позиций)

Пример автомата, распознающего :



Однако пока непонятно, что делать при получении “ошибки” при чтении шаблона .

Введем функцию ошибки: - состояние, в которое мы переходим по ошибке на позиции шаблона .

Очевидно, . (Если на первом же символе случилась ошибка, ничего не остается, кроме как начать “прикладывать” шаблон заново).

Если , то возникает ситуация, когда нужно “сдвинуть” шаблон. Для этого нужно найти наибольший суффикс уже прочитанного префикса шаблона, являющийся также и префиксом шаблона. (Так как нам нужно “сдвинуть” шаблон вправо по тексту на наименьшее расстояние, при котором уже “прочитанная” часть шаблона совпадет с текстом.)

Найдем для всех позиций шаблона функцию ошибки по индукции:

Б.И.

Ш.И. Перешли из в по букве , для функция ошибки известна: .

Если , то .

Иначе: обозначим . Повторим: если , то . Иначе… И далее рекурсивно, пока не найдем или не дойдем до нуля, тогда .

**Лемма:**

Пусть

Тогда - наидлиннейший суффикс, который является префиксом

(Другими словами, алгоритм поиска, приведенный выше, работает и корректно определяет функцию ошибки.)

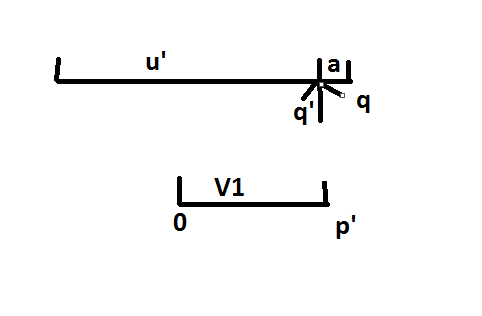
- наидлиннейший суффикс, который является префиксом

**Доказательство:**

По индукции.

Б.И.: очевидна.

Ш.И.: “Прочитали” префикс шаблона , дальше идет буква .

Пусть , - наидлиннейший суффикс префикса шаблона , являющийся также и префиксом.

Если , то наидлиннейший префикс и суффикс. При этом и , ч.т.д.

Иначе:

Получаем, что - не префикс . Берем как префикс до позиции . И так далее…

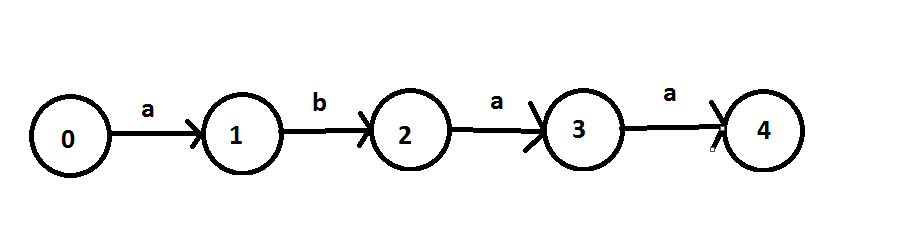
**Пример:**

Функция ошибки:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
|  | 0 | 0 | 0 | 1 | 1 |

Построим же наконец автомат для решения нашей задачи.

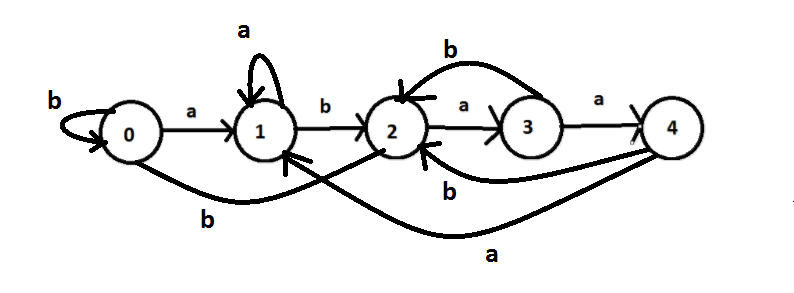
По индукции, разумеется. (Т.е. Дорисовываем ребра к автомату, нарисованному выше)



Б.И. , если было не определено. (В нашем примере из нуля по идем в нуль).

Ш.И. Если , то .

Получится вот такой автомат:



Теперь мы готовы вернуться к исходной задаче (алгоритму Ахо-Корасик, где не обязательно 1).

**Пример:**

Сначала нарисуем “ветвящийся” автомат, который умеет распознавать эти шаблоны, но пока еще не умеет корректно обрабатывать ошибки.

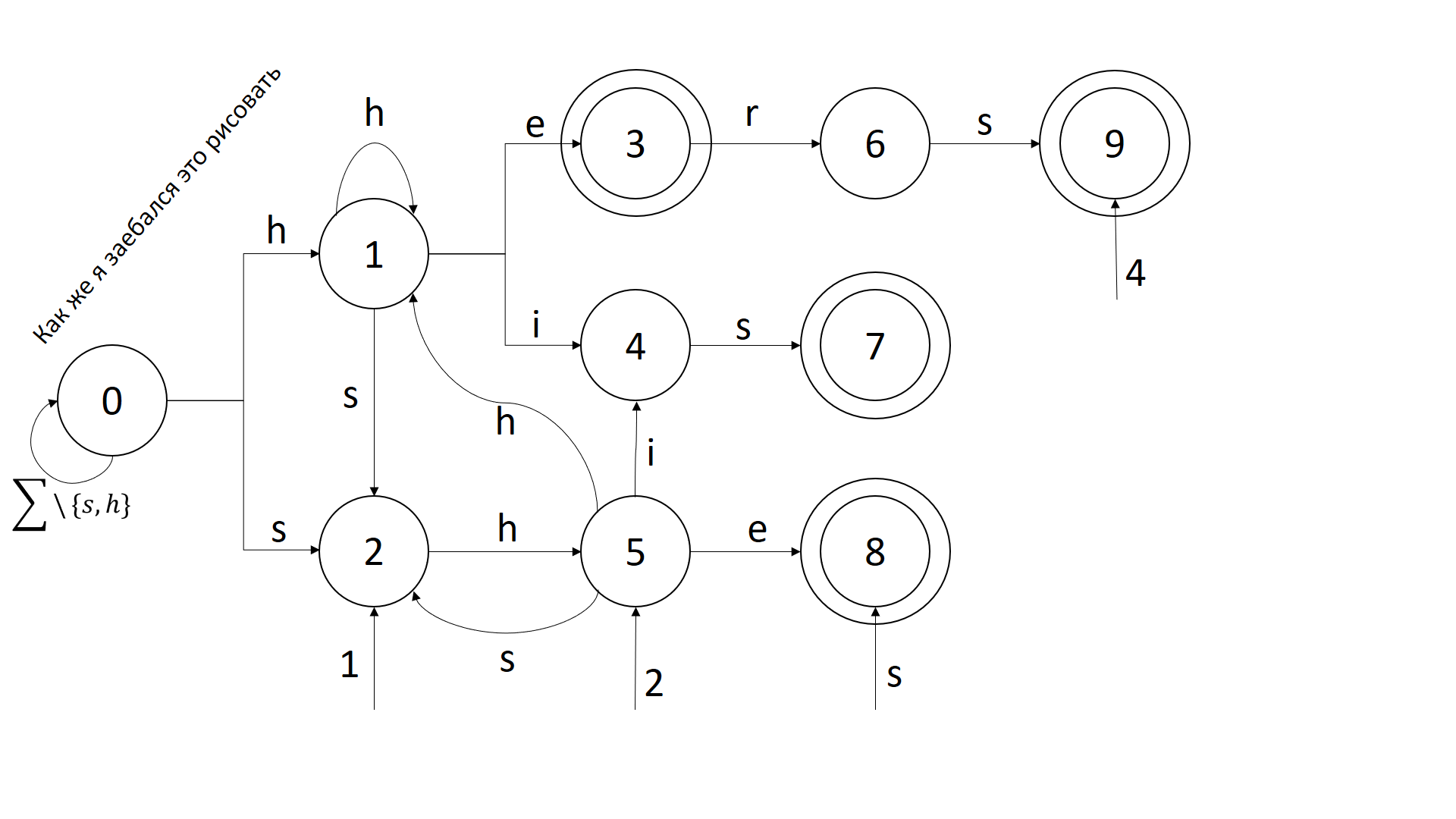
Теперь нужно определить функцию ошибки. Разница с предыдущим случаем весьма небольшая:

,

И слово является суффиксом и префиксом некоторого шаблона.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| f | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 2 |

выдает слова , которые +



**Утверждение:**

существует НКА, для которого алгоритм детерминирования строит ДКА с состояниями, где все состояния достижимы из начального.

**Док-во:**

Просто приводим пример для каждого n.

Возьмем любое натуральное , пусть наш автомат имеет состояния с номерами

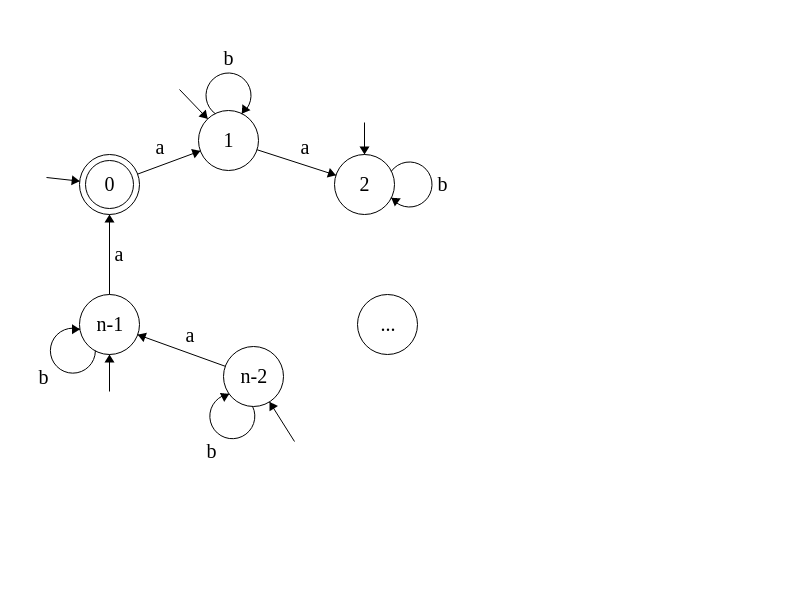
.

Объявим все состояния входными.

Пусть буква переводит любое состояние в ;

буква в нуле ведет в состояние смерти, а во всех остальных случаях делает “петлю”.

Вот так:

****

Покажем, что такой автомат - искомый.

**Доказательство:**

Пусть  **-** автомат, получающийся из в результате детерминирования.

Тогда его входное состояние - . (так как все состояния у входные).

Покажем, что из достижимо любое состояние . (т.е. Любое подмножество состояний ).

Индукцией по - количеству элементов, которые можно выбрасывать из , гарантированно получая достижимое состояние.

Б.И.: .

Действительно, достижимо из по пустому слову

Ш.И.: Пусть - достижимо.

Покажем что мы можем выкинуть любой элемент , оставив множество достижимым.

Случай 1: (т.е. нуль можно выкинуть).

Случай 2:

1. - Не определено

(т.е. Применяли букву , циклично “смещая” наше множество по часовой стрелке, пока не совпало с нулем, затем выкинули нуль и опять сместили наше множество состояний по часовой стрелке так, чтобы все элементы, кроме выкинутого, снова “вернулись на свое место”)

## Вопросы сложности

Пусть:

- Рациональный язык

число состояний в минимальном ДКА, распознающим

(SC - State Complexity)

**Напоминаем:**

**Теорема** (Майхилла-Нероуда (???)):

1. Автомат х частных:

**Утверждение 1:**

Пусть - рациональный язык, в минимальном автомате которого состояний.

Тогда

(т.е. в минимальном автомате, распознающим дополнение к столько же состояний)

**Доказательство:**

Пусть - распознает

Возьмем - распознает

Докажем, что минимальный.

От противного: если есть автомат с меньшим числом состояний, распознающий , то взяв к нему “автомат с чертой”” по тому же правилу, получим автомат, распознающий и с меньшим числом состояний, чем у . Противоречие.

**Утверждение 2:**

Пусть - рациональные языки.

Тогда .

(это только верхняя граница! Фактически значение может быть хоть нулем).

**Доказательство:**

Оценка очевидна.

**Примечание:**

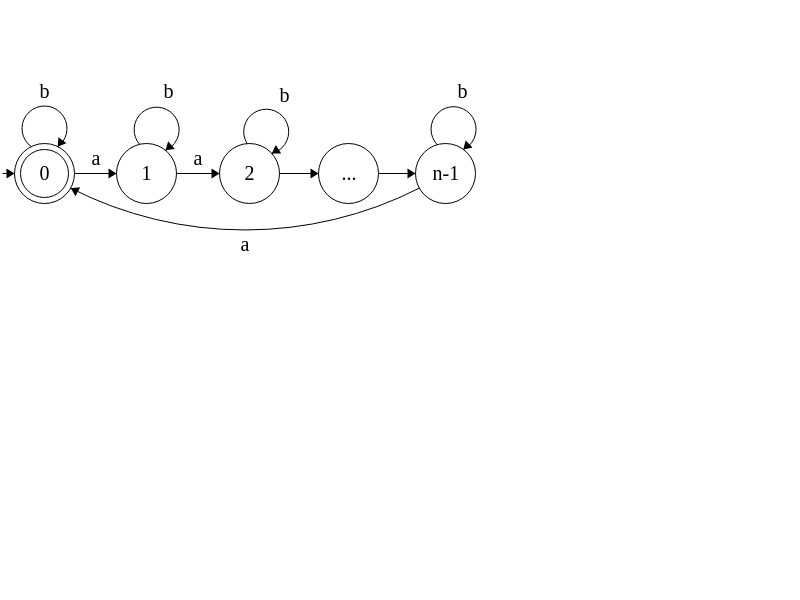
Значениедействительно достигается:

Пусть

Покажем, что:

Построим автомат из состояний, распознающий .

Вот он:

****

Тогда

Покажем что

Положим .

Покажем, что кол-во классов по этому отношению не может быть меньше . (Тогда мы получим, что в автомате не менее состояний.)

Действительно, слова все обязаны лежать в разных классах, так как для любой пары можно взять , после чего окажется, что , что может лежать в только лишь при условии, что , т.е. .

**Доказано.**

**Утверждение 3:**

Если - рациональные языки

Тогда

**Доказательство:**

Так как , то по утверждениям 1 и 2 ч.т.д.

**Утверждение 4:**

- ПРАВИЛЬНО СТЕПЕНЬ НАПИСАЛ??

кол-во выходных состояний в автомате для

Введем несколько определений:

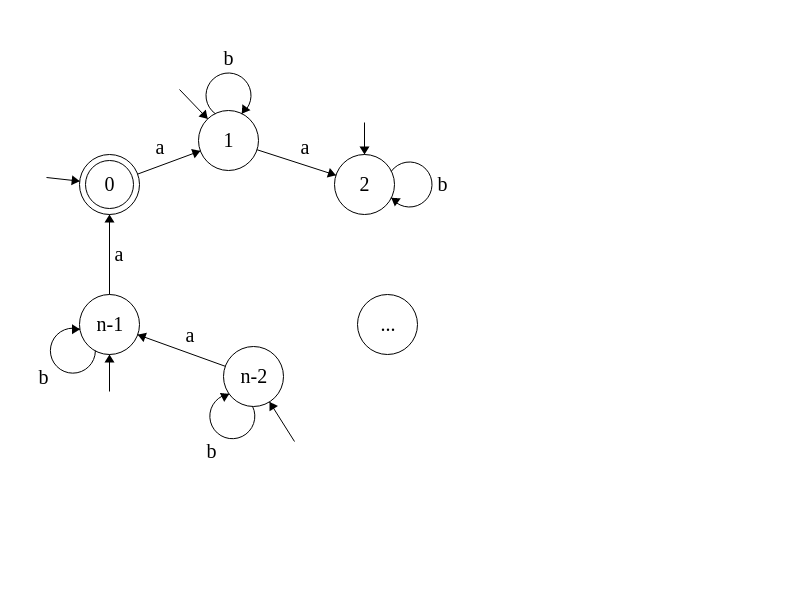
- операция разворота слова или языка

**Утверждение 5:**

**Доказательство:**

Очевидно :D. Ну в самом деле, чтобы построить НКА, распознающий , достаточно “развернуть все стрелки” в автомате, распознающем . После чего детермининируем это чудо и получим заявленную в утверждении оценку.

Пример, на котором достигается максимум, тот же, что и в утверждении из предыдущей темы:

****

**Предположение:**

- ДКА, минимальный для .

**Доказательство:**

Пусть - минимальный ДКА, распознающий .

По теореме Майхилла-Нероуда: найдется сюръекция , для которого:

Покажем, что инъективно (т.е. если образы совпадают, то и прообразы совпадают).

Действительно:

Покажем, что: , где .

Пусть

В автомате мы можем по букве попасть в .

По теореме Майхилла-Нероуда, в автомате из множества по попасть в некоторое , и в автомате найдется , по ведущее в некоторую , из которой по какому-то можно попасть в , т.е. .

То есть: .

Получаем , т.е. из по можно попасть в , откуда по в .

Получаем . Прообразы равны, ура.

---- The End of The Land ----