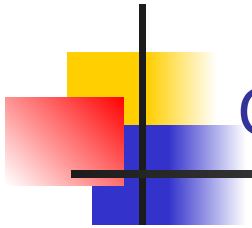
Автоматы и формальные языки

Карпов Юрий Глебович
профессор, д.т.н., зав.кафедрой
“Распределенные вычисления и компьютерные сети”
Санкт-Петербургского Политехнического университета

karpov@dcn.infos.ru



Структура курса

- Конечные автоматы-распознаватели – 4 л
  - Лекция 1. Формальные языки. Примеры языков. Грамматики. КА
  - Лекция 2. Теория конечных автоматов-распознавателей
  - Лекция 3. Трансляция автоматных языков
  - Лекция 4. Регулярные множества и регулярные выражения
- Порождающие грамматики Хомского – 3 л
- Атрибутные трансляции и двусмысленные КС-грамматики – 2 л
- Распознаватели КС-языков и трансляция – 6 л
- Дополнительные лекции 2 л

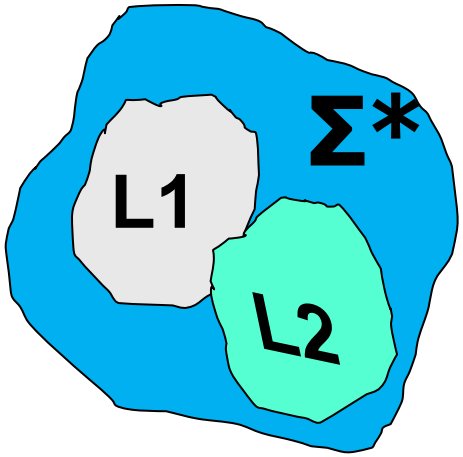
Формальные языки

Формальный язык – это некоторое множество конечных цепочек, построенных из символов конечного словаря.

$\Sigma = \{a, b\}$

$\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, \ldots, abbabb, \ldots\}$

$L1 = \{\varepsilon, a, ab, abb, \ldots, abbbb, \ldots\}$



Языки могут быть конечными и бесконечными.

Число всех возможных языков - континуум.

Основная проблема: как задать бесконечный язык конечной формальной моделью.

Любой конечный формализм, задающий язык, называется грамматикой.

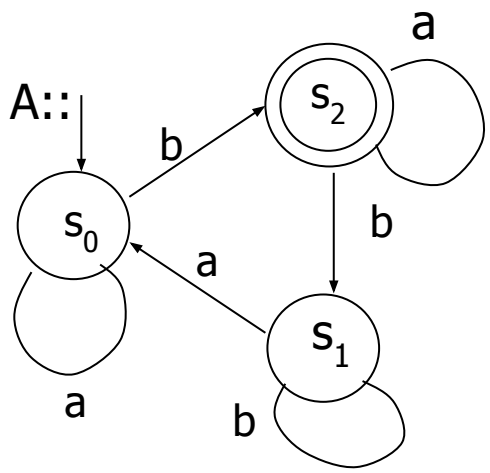
Автоматные языки

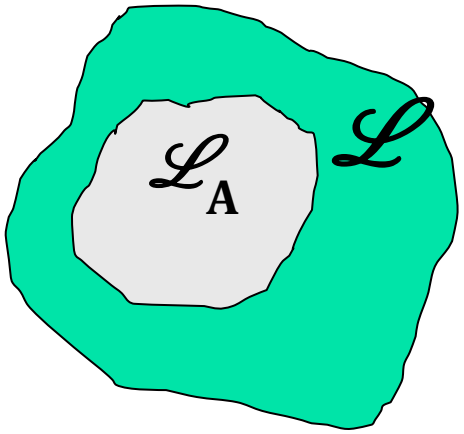
Конечный автомат может задавать язык.

Язык, для которого *существует* распознающий его конечный автомат, называется автоматным.

$aabaaa \in L_A$

$bbb \notin L_A$





Любой конечный автомат-распознаватель определяет какой-нибудь язык.

- Конечных автоматов-распознавателей бесконечное число. И автоматных языков тоже бесконечное число.
- Но существуют языки, которые не являются автоматными. Например, язык вложенных скобок $L=\{a^n b^n \mid n \geq 0\}$ – неавтоматный.

Представление конечных автоматов

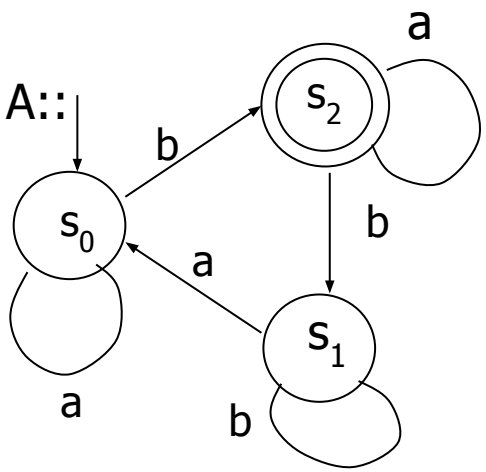
Формально:

$A=(S, \Sigma, s_0, \delta, F)$

S – множество состояний,
Σ - входной алфавит,
$s_0 \in S$ – начальное состояние s_0 ,
$\delta : S \times \Sigma \to S$ - функция переходов,
$F \subseteq S$ – множ финальных состояний.

$S = \{s_0, s_1, s_2\}$;
$\Sigma = \{a, b\}$;
$s_0 \in S$;
$\delta : \delta(s_0, a)=s_0; \delta(s_0, b)=s_2; \delta(s_1, a)=s_0;$
$\delta(s_1, b) = s_1; \ldots$
$F = \{s_2\}$.

Граф переходов:



$aaba \in L_A$;

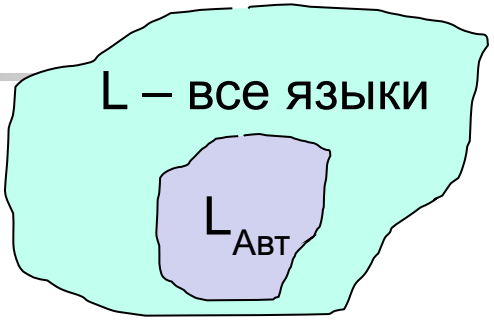
$aabbb \notin L_A$.

Таблица переходов:

	δ	
	x=a	x=b
s_0 –	s_0	s_2
s_1	s_0	s_1
s_2 +	s_2	s_1

$\delta(s_0, b) = s_2$

Необходимость изучения автоматных языков и конечных автоматов-распознавателей

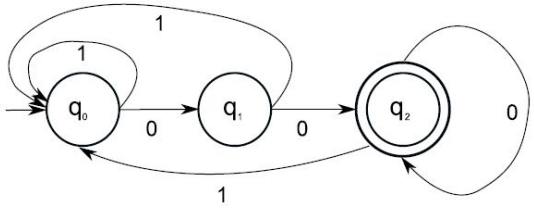


- Автоматными языками являются многие языки:
  - Специализированные языки
  - Языки управления ОС
  - Многие скриптовые языки
  - Фрагменты языков высокого уровня (имена, константы, комментарии, ...)
  - Все регулярные языки (т.е. языки, задаваемые регулярными выражениями)
  - … …
- Автоматные языки составляют лишь подкласс (довольно узкий) всех возможных языков
  - Примеры неавтоматных языков:
    - язык вложенных скобок (и любой язык, с вложенными конструкциями)
    - множество цепочек с одинаковым числом вхождений символов a и b
    - все языки программирования высокого уровня, естественные языки ....

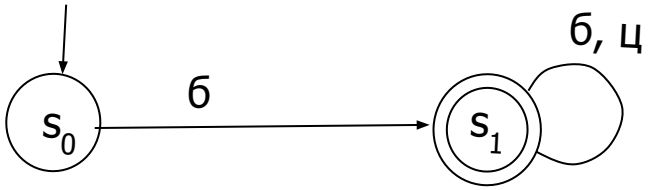
    Существует множество важных и интересных практических применений автоматных языков

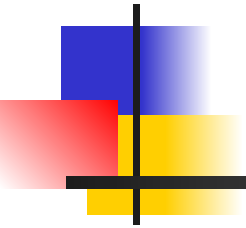
Примеры языков и распознающих их конечных автоматов

- Пример 1. Язык: цепочки из 0 и 1, заканчивающиеся на 00
  Например: 0010100, 0100, 000, ...



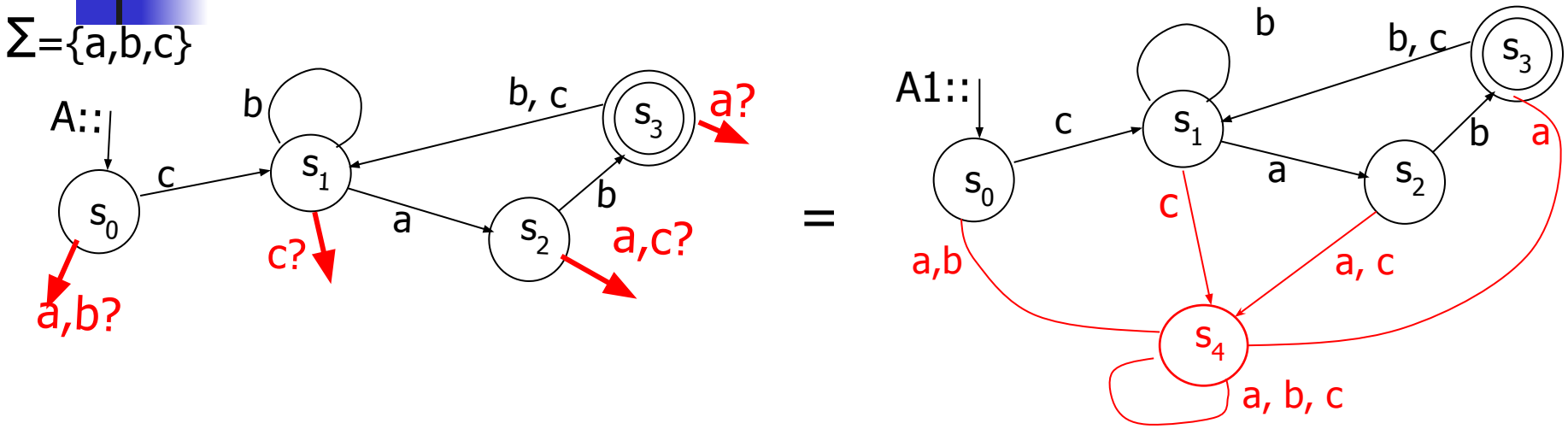
- Пример 2. Язык: все возможные идентификаторы
  Например: abba, a0012j, j23, …





Операции над автоматами: тримминг

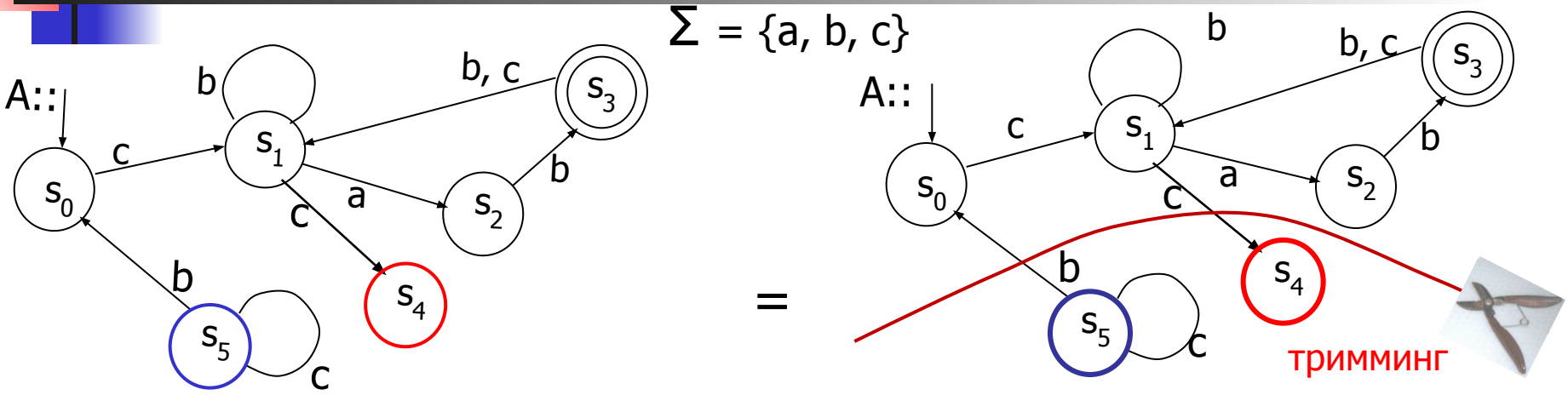
Полный конечный автомат



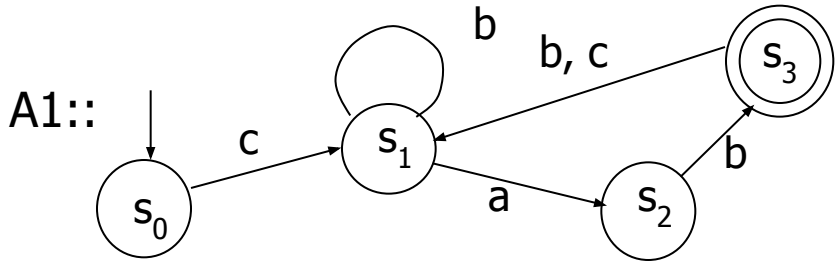
По умолчанию всегда существует – “sink state”, из которого не достижимо никакое финальное состояние.

- Если переходы в автомате-распознавателе под воздействием некоторых событий не определены, то это означает, что соответствующие цепочки входного языка ошибочны, они не допускаются, не принадлежат языку, распознаваемому этим автоматом.
- Для любых манипуляций с автоматом это состояние нужно вводить явно.
- Полученный полный автомат с точки зрения распознавания языка эквивалентен исходному автомату.

Приведение конечного автомата



- Операция “приведения” (trimming) конечного автомата: выбрасывание всех состояний, недостижимых из начального состояния, и тех, из которых недостижимо ни одно из финальных состояний (не кодостижимых).
- При этом язык, допускаемый автоматом, не меняется!



A1 – приведенный автомат

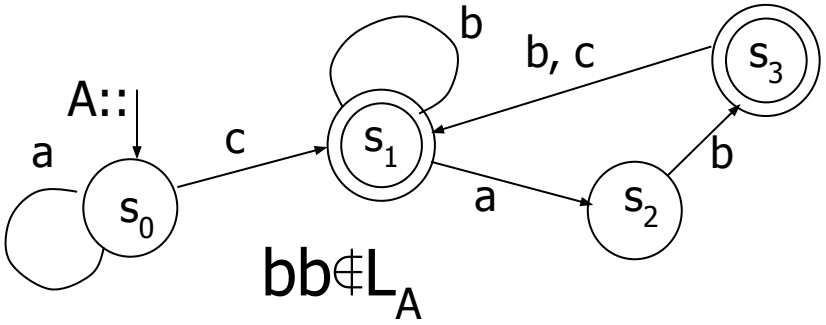
L(A) = L(A1)

- Автомат является приведенным (trimmed, “*подстриженным*”) тогда и только тогда, когда все его состояния достижимы и из любого состояния существует путь в какое-нибудь финальное состояние.

Распознавание дополнения автоматного языка

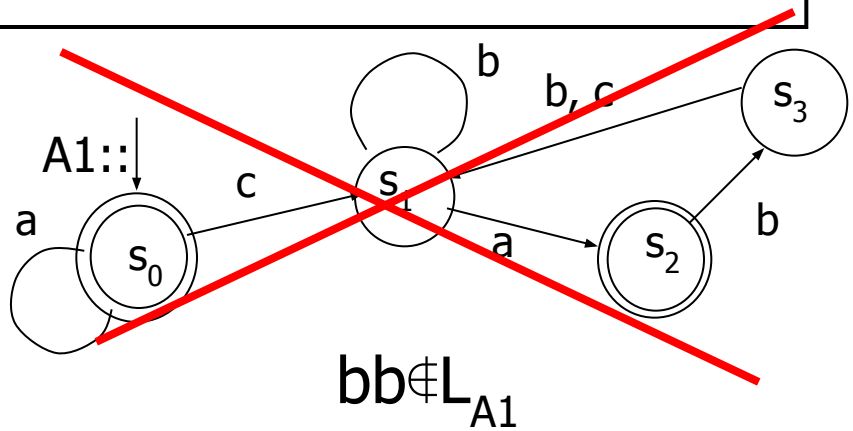
- КА A распознает цепочку, если она переводит его из начального в одно из финальных состояний. Язык L_A состоит из всех допускаемых цепочек.

Следовательно, все те цепочки, которые переводят автомат из начального состояния в одно из недопускающих состояний, являются цепочками языка, дополняющего L до Σ*.



$bb \notin L_A$

$\Rightarrow$??



$bb \notin L_{A1}$

L(A1) = (?) Σ*-L(A)

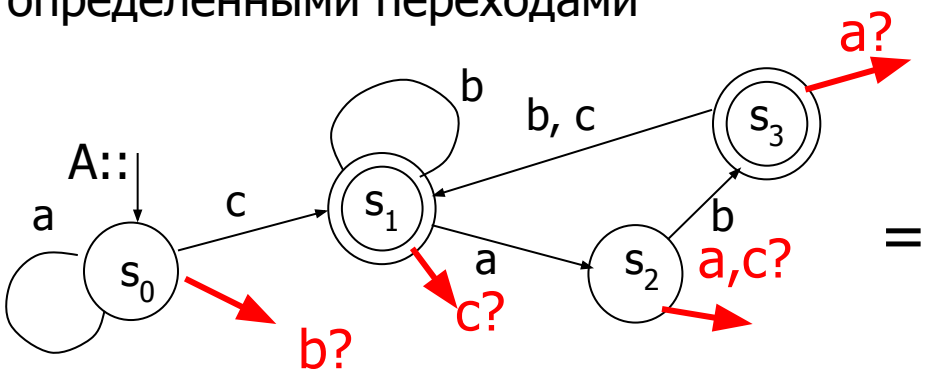
A1 - дополнение A?? НЕТ!

Где ошибка??

Такое построение неверно, потому что мы не учли все состояния и переходы, которые не обозначены на картинке! Например, цепочка bb не является цепочкой языка L(A), но она не допускается и построенным новым автоматом

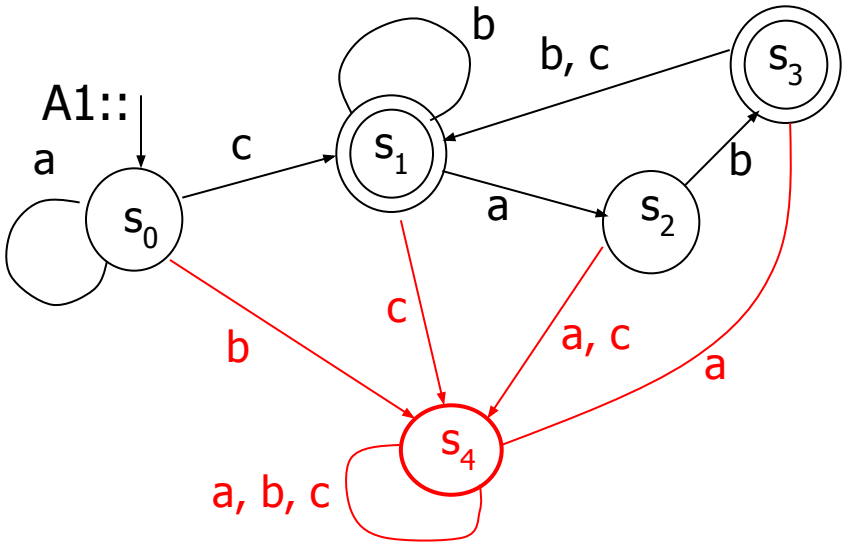
Распознаватель дополнения автоматного языка

1: Заданный КА с неполностью определенными переходами

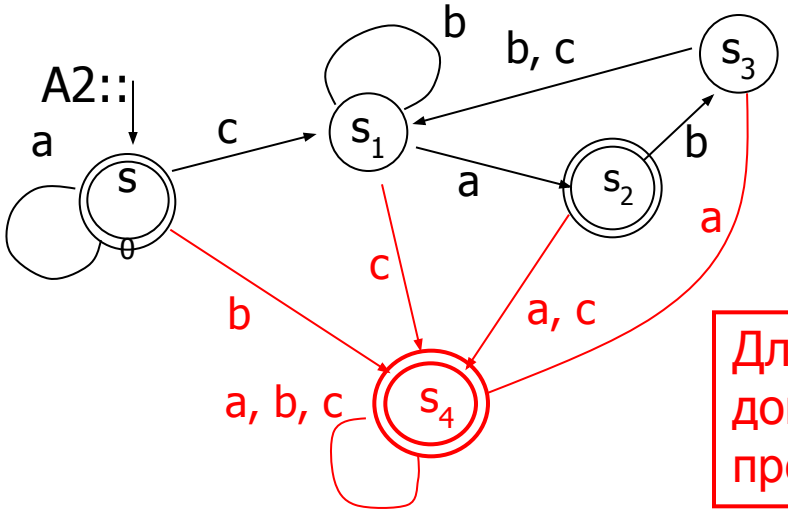


=

2: КА с полностью определенными переходами L(A1) = L(A)

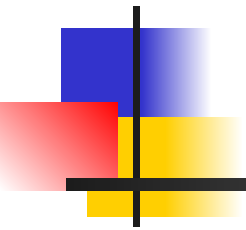


3: Финальные состояния делаем не финальными, и наоборот L(A2) = Σ*-L(A)



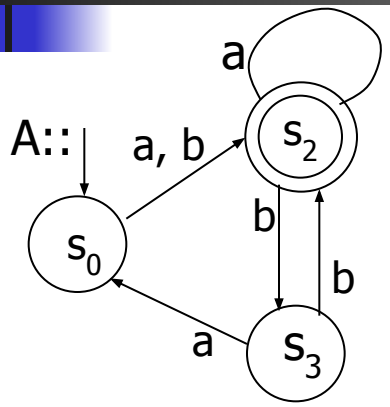
Все цепочки, которые **НЕ** переводят А в какое-нибудь финальное состояние, принадлежат множеству Σ-L_A, т.е. дополнению языка L_A

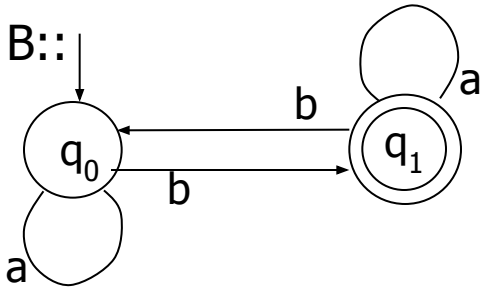
Для построения автомата, распознающего дополнение языка, в исходном автомате нужно представить ВСЕ состояния

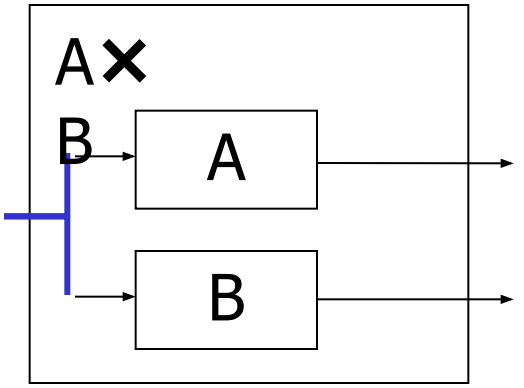


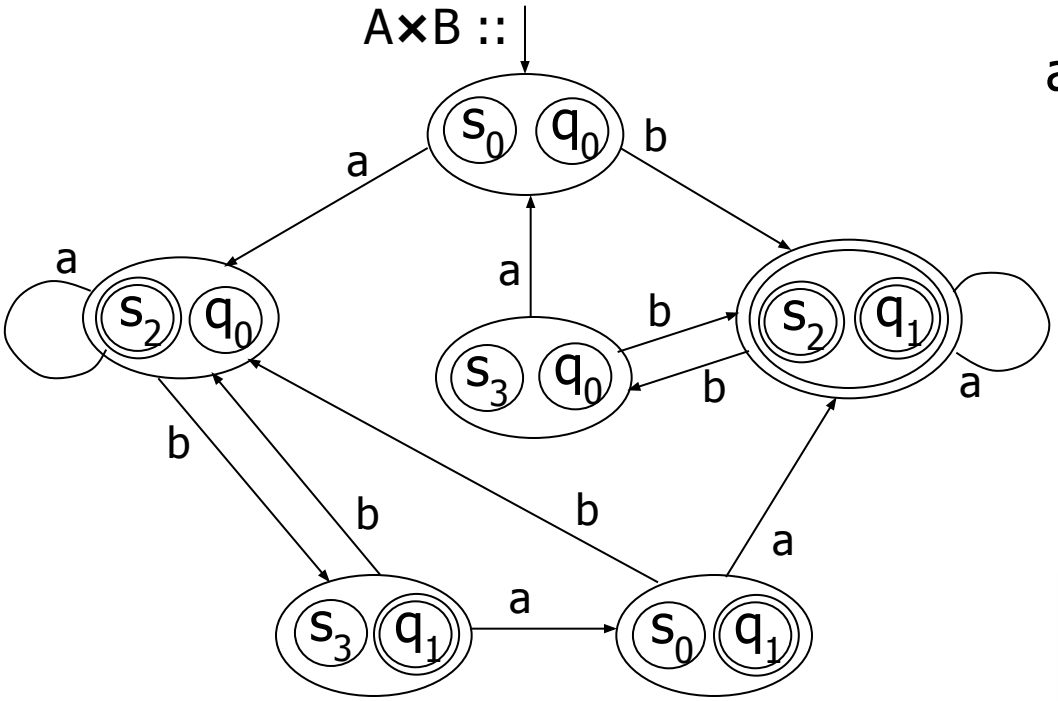
Синхронная композиция двух автоматов-распознавателей
Пересечение автоматных языков

Синхронная композиция автоматов – два автомата, работающие синхронно от одного входа









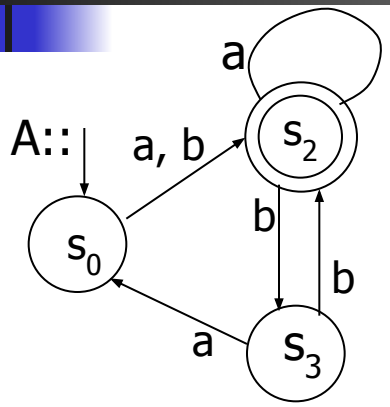
Если цепочка α привела автомат A×B в <s2, q1>, то α допускается и A, и B.

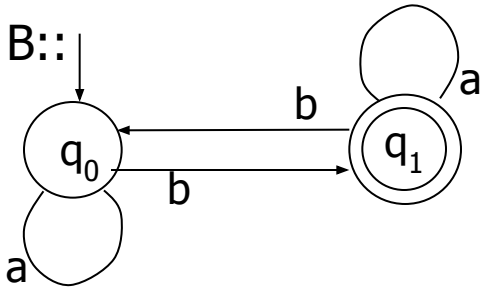
Какой язык допускает A×B

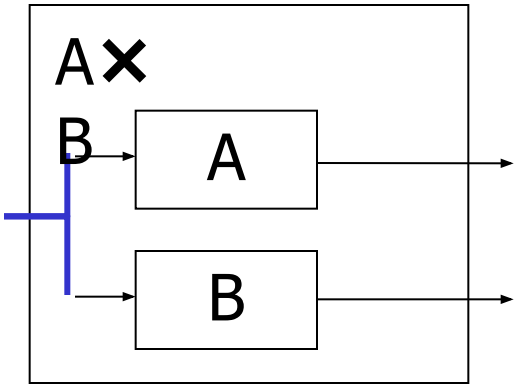
Синхронная композиция A×B автоматов A и B допускает язык, который является пересечением языков, допускаемых A и B.

$$L_{A\times B} = L_A \cap L_B$$

Синхронная композиция автоматов.







$$L_{A\times B} = L_A \cap L_B$$

$$\frac{s \text{ -a-> } s' \in \delta_A;\ q \text{ -a-> } q' \in \delta_B}{(s, q) \text{ -a-> } (s', q') \in \delta_{A\times B}}$$

Правило переходов A×B:

Если и из s, и из q есть переходы под воздействием a, то из (s, q) есть переход под воздействием a.

$$F_{A\times B} = F_A \times F_B$$

пары финальных состояний

Финальные состояния A×B:

Такие пары (s,q), что $s \in F_A$, $q \in F_B$.

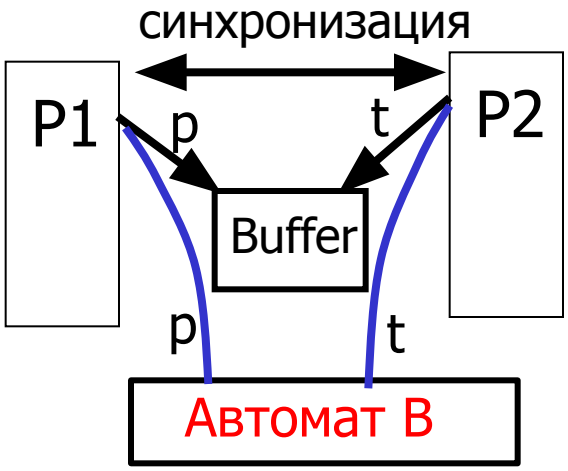
Максимальное число достижимых состояний синхронной композиции автоматов A и B равно $|S_A| \times |S_B|$.
Но могут быть и недостижимые состояния.

Пример. Статический анализ текста || программ

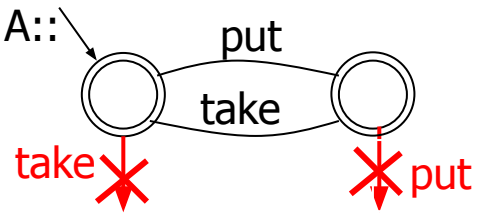
- Правильные цепочки событий обращения к буферу (*запись*, p от put и *выборка*, t от take) –

  язык (pt)*

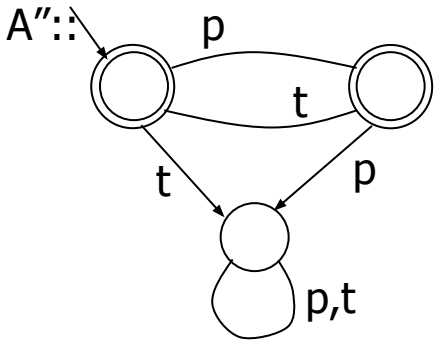
- Анализ статического кода параллельной программы, работающей с буфером, может выявить любую неправильную подцепочку в потоке операций



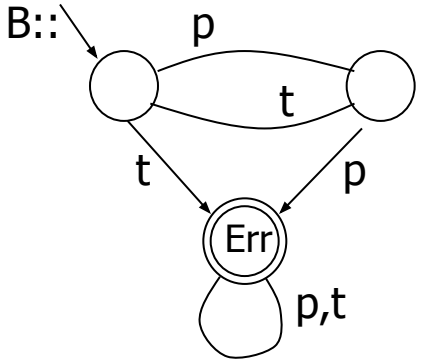
Правильные цепочки обращений к одноместному буферу:



Полный автомат A, определяющий все *правильные* цепочки обращений к буферу:



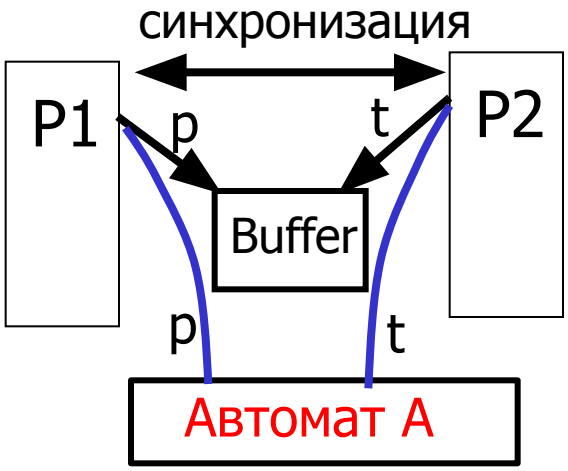
Автомат B – дополнение A, определяет все неправильные поведения системы процессов



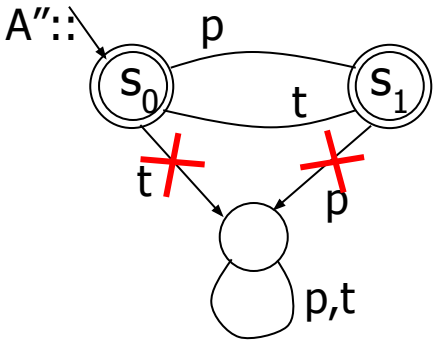
Синтез супервизоров – новая идея построения систем логического управления

- Автомат А может быть использован как супервизор – устройство управления, которое ограничивает функционирование системы процессов только правильными цепочками

язык (pt)*



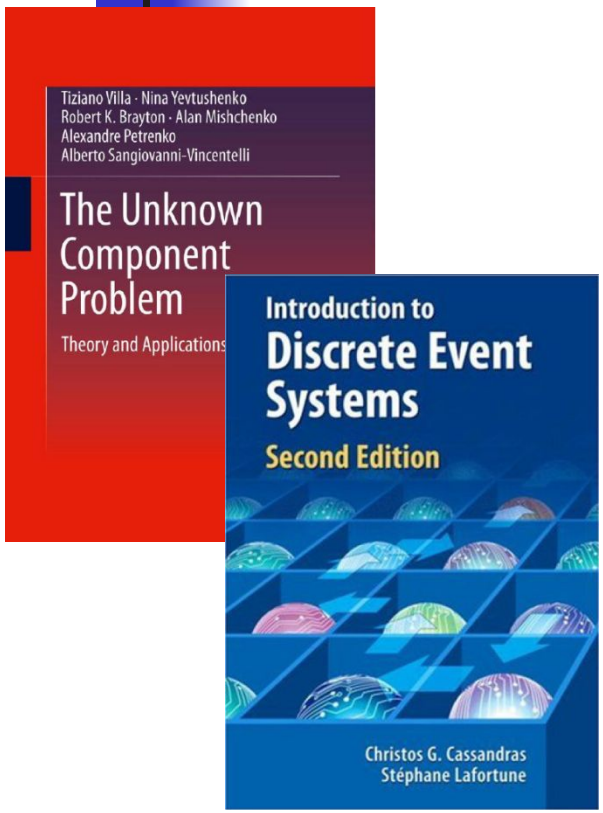
Полный автомат А, определяющий все *правильные* цепочки обращений к буферу:



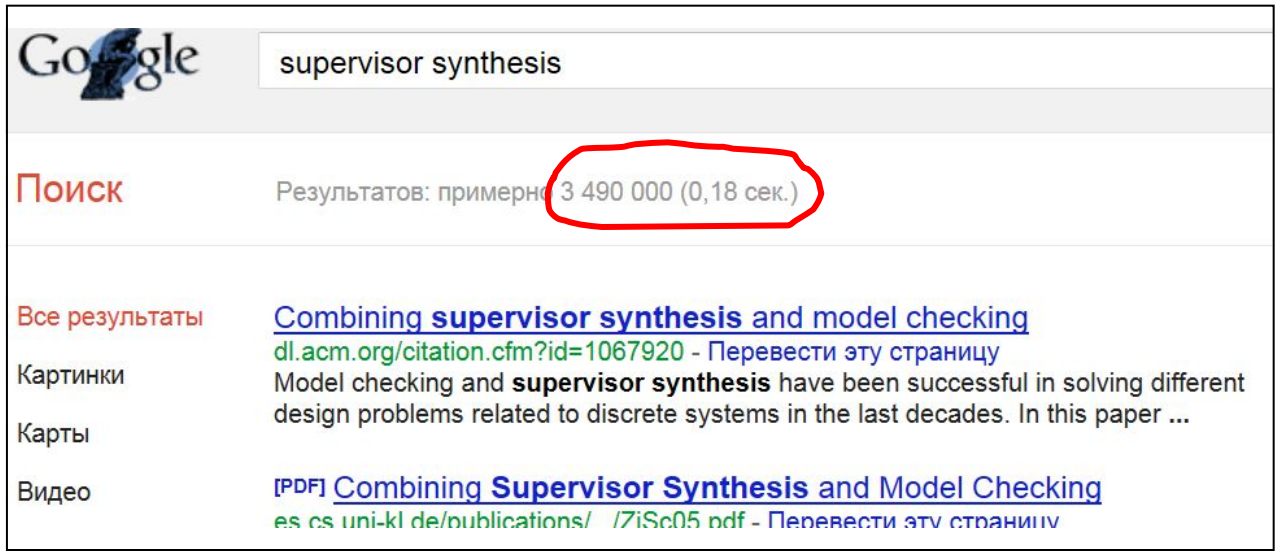
Супервизор А в каждом состоянии посылает процессам множество тех действий, которые допустимы в данном состоянии. Выполненные процессами действия перехватываются супервизором для того, чтобы перейти в новое состояние и в нем также ограничить возможные действия процессов (синхронизация).

Так осуществляется СИНХРОНИЗАЦИЯ параллельных процессов. Формальная основа – теория автоматных языков.

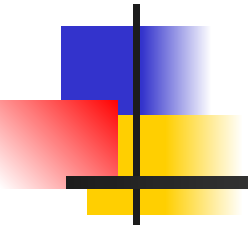
Синтез супервизоров - “горячая” область исследований



Монографии, миллионы публикаций, десятки ежегодных конференций



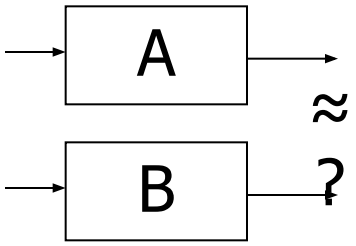
- INTERNATIONAL WORKSHOP ON DISCRETE EVENT SYSTEMS
- IEEE INT CONFERENCE ON EMERGING TECHNOLOGIES AND FACTORY AUTOMATION
- CONFERENCE OF IEEE INDUSTRIAL ELECTRONICS
- IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION
- …



Эквивалентность двух автоматов-распознавателей

Эквивалентность двух конечных автоматов-распознавателей – как проверить?

- **Определение.** Два конечных автомата-распознавателя эквивалентны, если языки, распознаваемые ими, совпадают

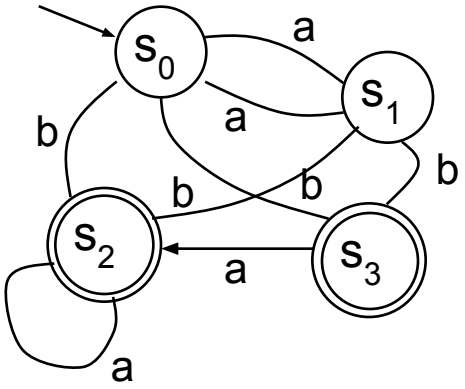


$A=(S_A, \Sigma_A, s_{0A}, \delta_A, F_A)$ $B=(S_B, \Sigma_B, s_{0B}, \delta_B, F_B)$

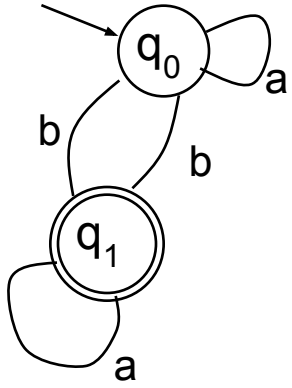
1. Требование: Одинаковые входные алфавиты, $\Sigma_A = \Sigma_B$
2. Вопрос: Совпадают ли множества входных цепочек, которые переводят автоматы из начального состояния в допускающее состояние?

$\Sigma = \{a, b, c\}$; $\Sigma^* = \{\varepsilon, a, b, c, aa, ab, ac, cc, bb, cba, cbbba, ...\}$

$A \approx B$ iff $(\forall \alpha \in \Sigma^*)\ [\delta^*_A(s_{0A}, \alpha) \in F_A \Leftrightarrow \delta^*_B(s_{0B}, \alpha) \in F_B]$

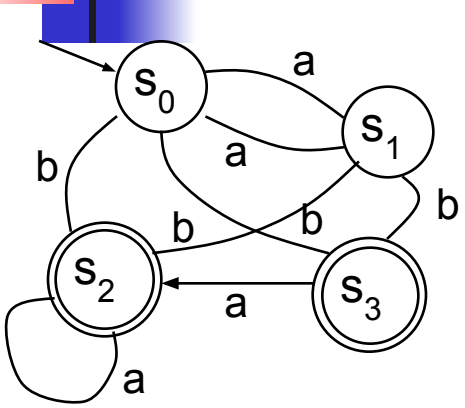


≈?

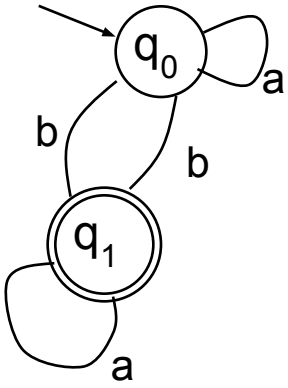


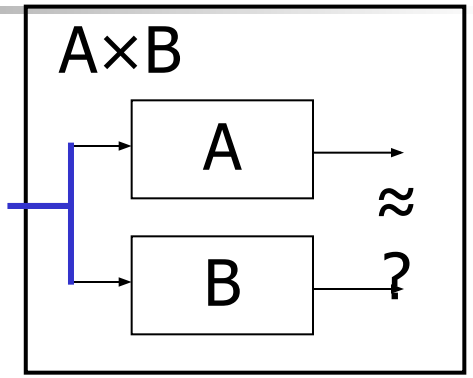
Эквивалентные автоматы под воздействием любой входной цепочки переходят либо оба в финальные, либо оба в нефинальные состояния

Эквивалентные автоматы-распознаватели. Теорема Мура



≈?





Два автомата-распознавателя эквивалентны тогда и только тогда, когда любая цепочка, допускаемая первым автоматом, допускается и вторым, И НАОБОРОТ.

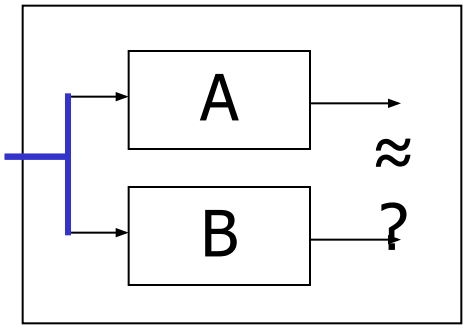
НО МЫ НЕ МОЖЕМ ПЕРЕБРАТЬ ВСЕ ВОЗМОЖНЫЕ ЦЕПОЧКИ !!

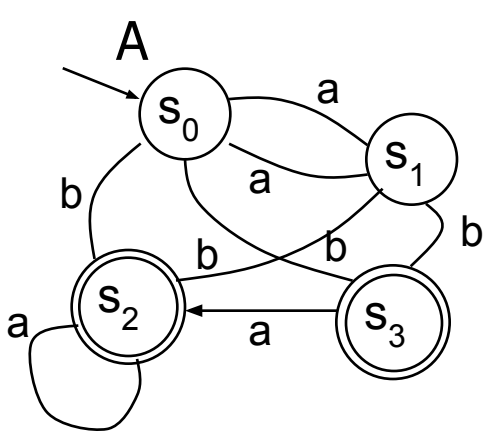
Что делать??

Теорема Мура. Проблема эквивалентности конечных автоматов разрешима.

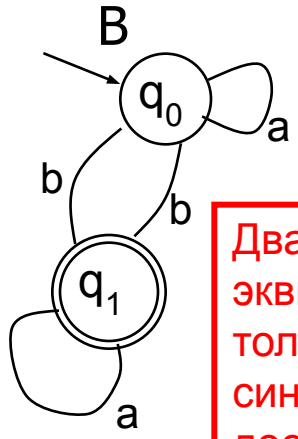
Доказательство. *Два автомата будут эквивалентными тогда и только тогда, когда в их синхронной композиции достижимы только такие пары состояний, в которых либо оба компонентные состояния принимающие (финальные), либо оба состояния не принимающие (не финальные). Поскольку число состояний синхронной композиции конечных автоматов КОНЕЧНО, мы можем все достижимые пары состояний проанализировать.*

Эквивалентные автоматы-распознаватели



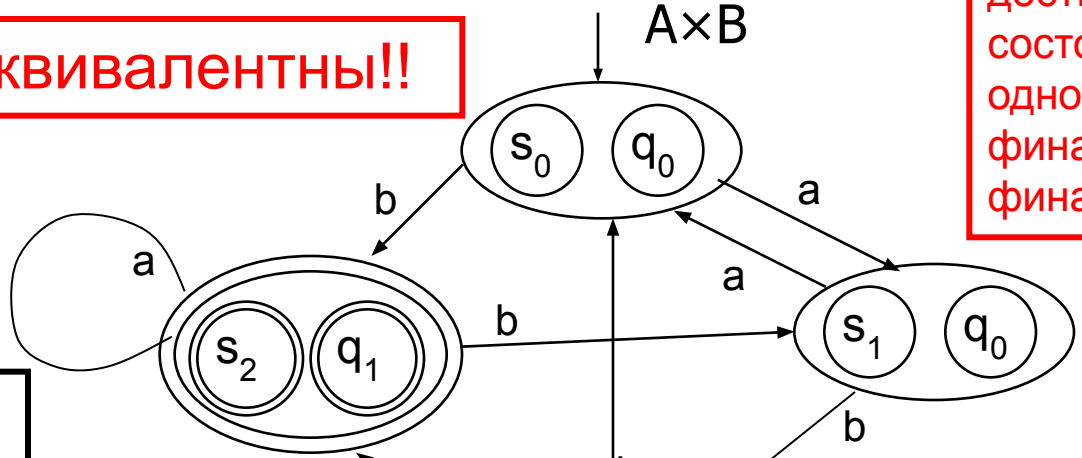


×

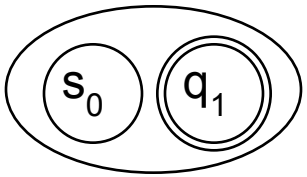


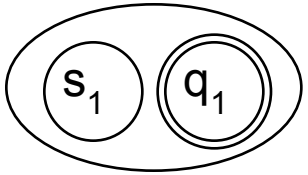
Два автомата будут эквивалентными тогда и только тогда, когда в их синхронной композиции достижимы только пары состояний, которые одновременно либо оба финальные, либо оба не финальные.

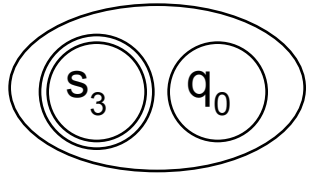
Автоматы эквивалентны!!

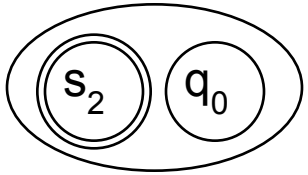


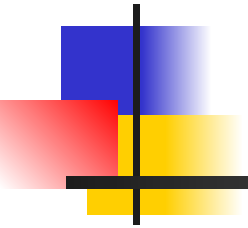
4 состояния недостижимы







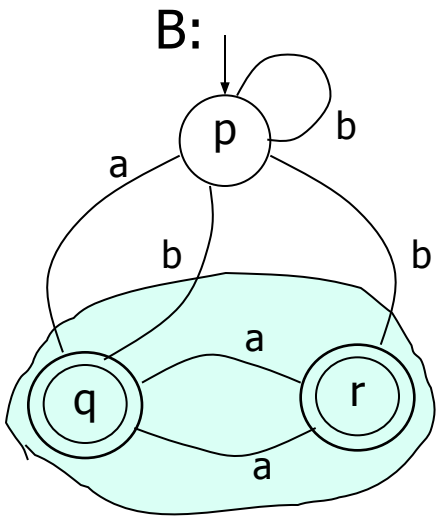




Минимизация конечных автоматов-распознавателей

Минимизация конечных автоматов-распознавателей

- Существует единственный (с точностью до изоморфизма, т.е. именования состояний) минимальный конечный автомат, распознающий данный автоматный язык.
- У КА есть его каноническое представление - это минимальный КА.
  - Для минимизации КА-распознавателя нужно найти группы (классы) эквивалентных состояний.



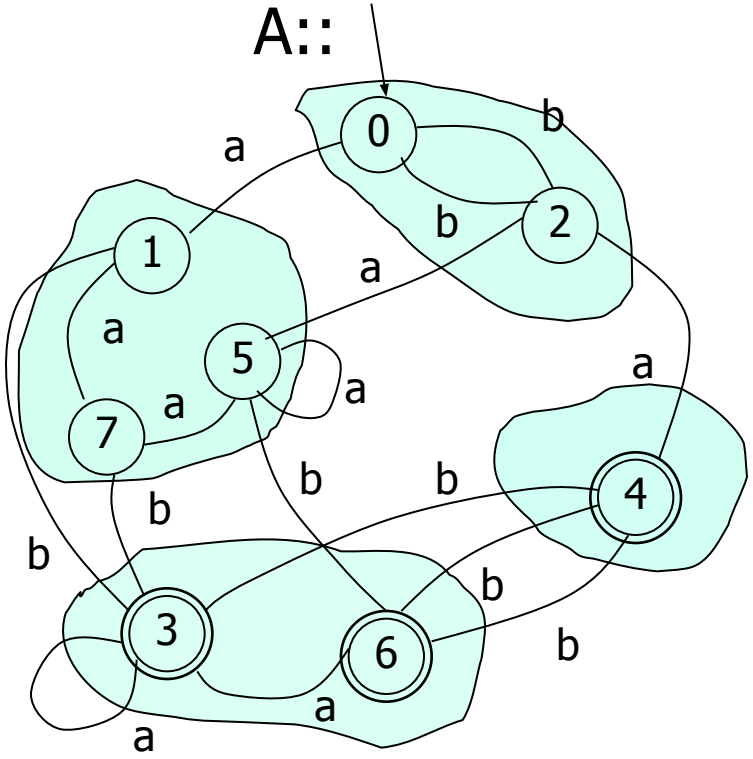
q, r – эквивалентны, если две копии автомата A, которые начинают свое функционирование с состояний q и r, невозможно различить: *любая подаваемая на вход автоматов входная цепочка либо обоими автоматами допускается, либо обоими автоматами не допускается*

Формальное определение эквивалентных состояний:

$$q \approx r \Leftrightarrow (\forall \alpha \in \Sigma^*)\ [\ \delta^*(q, \alpha) \in F \Leftrightarrow \delta^*(r, \alpha) \in F\ ]$$

НО мы не можем использовать это определение практически: мы не можем перебрать все возможные входные цепочки для каждой пары состояний: таких цепочек (поведений автоматов) бесконечное число

Пример неминимального автомата



Строим автомат таким образом: два эквивалентных состояния переходят (под воздействием одного и того же входа) в эквивалентные состояния.

- Подмножества состояний {0,2}; {1,5,7}; {3,6}; {4} – классы эквивалентных состояний. Как найти эту эквивалентность?

$$q \approx r \;\Leftrightarrow\; (\forall x \in \Sigma)\,[\,\delta(q, x) \approx \delta(r, x)\,]$$

эквивалентные состояния под воздействием одинаковых входов переходят в эквивалентные состояния.

Минимизация конечных автоматов - распознавателей

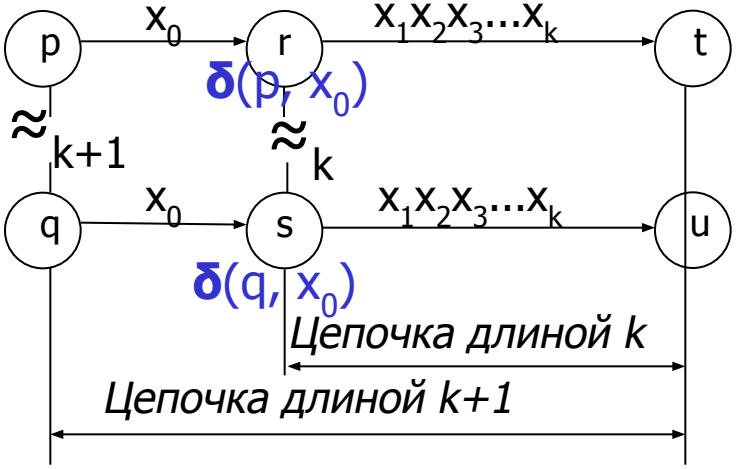
- Построим на множестве состояний автомата A разбиения $\pi_0, \pi_1, \ldots, \pi_\infty$, такие, что в один класс π_k попадают состояния, из которых цепочки длиной k одновременно допускаются или одновременно не допускаются.
- Такие состояния будем считать k-эквивалентными, т.е. $p \approx_k q$ (в отношении π_k)

  $p \approx_k q \Leftrightarrow (\forall \boldsymbol{\alpha} \in \boldsymbol{\Sigma}^*: |\boldsymbol{\alpha}| = k)\ [\ \boldsymbol{\delta}^*(p, \alpha) \in F \Leftrightarrow \boldsymbol{\delta}^*(q, \alpha) \in F\ ]$.

Алгоритм.

Шаг 1. Полагаем k=0: все допускающие состояния и все не допускающие состояния 0-эквивалентны. Т.е. строим два класса 0-эквивалентности: все допускающие состояния – один класс, все недопускающие – другой класс.

Шаг 2. От k к k+1. Очевидно, что $p \approx_{k+1} q \Leftrightarrow (\forall x \in \boldsymbol{\Sigma})\ \boldsymbol{\delta}(p,x) \approx_k \boldsymbol{\delta}(q,x)$.

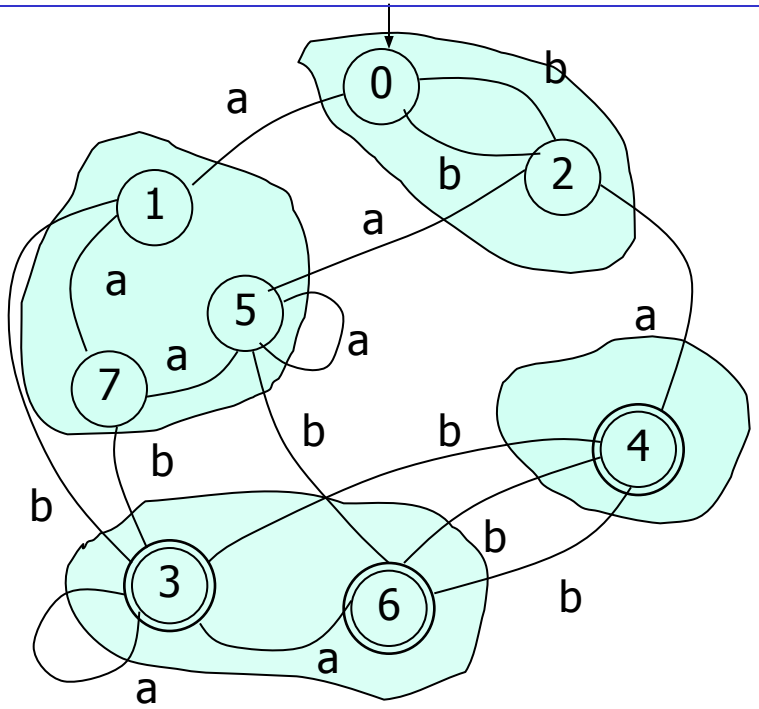


Состояния p и q k+1-эквивалентны, если и только если для любого $x \in \Sigma$ состояния $\boldsymbol{\delta}(p, x)$ и $\boldsymbol{\delta}(q, x)$ k-эквивалентны.

Состояния p и q находятся в одном блоке k+1-эквивалентности, если и только если для любого $x \in \Sigma$ состояния $\boldsymbol{\delta}(p,x)$ и $\boldsymbol{\delta}(q,x)$ находятся в одном и том же блоке предыдущей, k-эквивалентности .

Конечный автомат с эквивалентными состояниями

Если R_k – класс эквивалентных состояний по эквивалентности $п_k$, то все переходы из состояний класса R_k, помеченные одним и тем же входным символом, должны вести в состояния одного и того же класса предыдущей эквивалентности



	δ		$δ_{п0}$		$δ_{п1}$	
	x=a	x=b	x=a	x=b	x=a	x=b
0	1	2	A	A	D	C
1	7	3	A	B	D	E
2	5	0	A	A	D	C
3^+	3	4	B	B	E	F
4^+	2	6	A	B		
5	5	6	A	B	D	E
6^+	3	4	B	B	E	F
7	5	3	A	B	D	E

- п0 = {0, 1, 2, 5, 7}=A; {3, 4, 6}=B
- п1 = {0, 2}=C; {1, 5, 7}=D; {3, 6}=E; {4}=F
- п2 = {0, 2}; {1, 5, 7}; {3, 6}; {4} = п1 = п∞

Алгоритм построения эквивалентности π

- Так же, как и для автоматов-преобразователей, рассматриваем последовательность разбиений π_k на классы эквивалентности.
- Два состояния, s и q, π_k-эквивалентны, если под воздействием любой входной цепочки длиной не более k, автоматы попадают оба либо в пару финальных, либо в пару нефинальных состояний.
- π_0 всегда состоит из двух классов: все нефинальные состояния – один класс, все финальные состояния – другой.
- Пусть уже построено разбиение π_k. Два состояния, находящиеся в одном классе эквивалентности π_k, будут в одном классе эквивалентности π_{k+1} тогда и только тогда, когда для любого x состояния δ(s,x) и δ(q,x) будут в одном и том же классе предыдущего разбиения π_k.

$\pi_0 = \{ (0, 1, 2, 5, 7); (3, 4, 6) \}$

$\pi_1 = \{ (0, 2); (1, 5, 7); (3, 6); (4)\}$

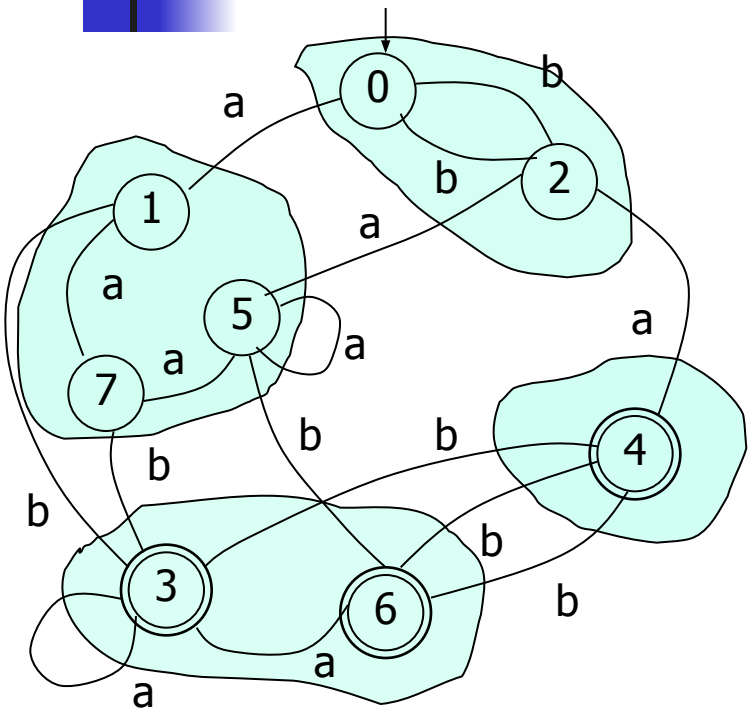
$\pi_2 = \pi_1$

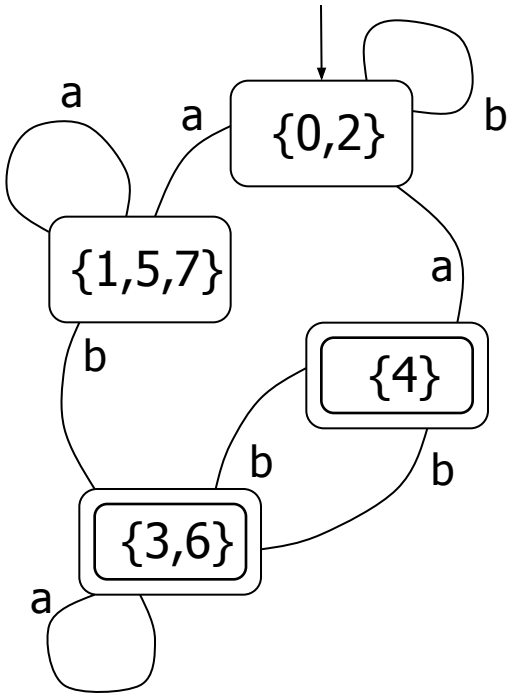
	δ	
	x=a	x=b
0 -	1	2
1	7	3
2	5	0
3+	3	4
4+	2	6
5	5	6
6+	3	4
7	5	3

3^+, 4^+ и 6^+ - финальные состояния

Эквивалентны:
0 и 2;
1, 5 и 7;
3 и 6;
4 само себе

Минимальный конечный автомат-распознаватель





Минимальный КА – единственный (с точностью до изоморфизма)

- п0 = {0, 1, 2, 5, 7}=A; {3, 4, 6}=B
- п1 = {0, 2}=C; {1, 5, 7}=D; {3, 6}=E; {4}=F
- п2 = {0, 2}; {1, 5, 7}; {3, 6}; {4} = п1 = п∞

	$\delta_{п\infty}$	
	x=a	x=b
$\{0,2\}^-$	**{1,5,7}**	{0,2}
{1,5,7}	**{1,5,7}**	{3,6}
$(3,6)^+$	{3,6}	{4}
$\{4\}^+$	{0,2}	{3,6}

Состояниями минимального автомата являются классы эквивалентности $п_\infty$ исходного автомата

Представление отношения эквивалентности матрицей

- Манипуляции с отношениями эквивалентности удобно выполнять с помощью матрицы инциденций, в которой в клетке <i, j> ставим N, если элементы i и j НЕ принадлежат одному и тому же блоку соответствующего разбиения

	5	4	3	2	1	0
0	N	N	N			
1	N	N	N			
2	N	N	N			
3	N			N	N	N
4	N			N	N	N
5		N	N	N	N	N

	5	4	3	2	1
0	N	N	N		
1	N	N	N		
2	N	N	N		
3	N				
4	N				

- Пусть π = {0, 1, 2}; {3, 4}; {5}
  – всего 6 элементов: {0, …, 5}
- Матрица симметричная, потому что если i и j не принадлежат одному блоку, то и j и i не принадлежат одному блоку.
- Диагональ в этой матрице излишняя: для любого элемента i пара <i, i> всегда принадлежит одному и тому же блоку
- Вывод: в матрице можно выбросить один треугольник и диагональ

Треугольная матрица отношения эквивалентности
π = {0, 1, 2}; {3, 4}; {5}

Эквивалентное представление алгоритма построения отношения эквивалентности

- ***Алгоритм: для каждой пары состояний (p,q) определяет, являются ли p и q эквивалентными***
- Строим треугольную матрицу пар
- Начальное заполнение: для каждой пары (p,q), для которой одно состояние заключительное, другое - нет, ставим N (нет)
- Многократно: Для каждой пары <p, q>, у которой не стоит N, проверяем, стоит ли N для пар <δ(p,x), δ(q,x)> хотя бы при одном x. Если стоит, то для пары <p, q> ставим N.
- Алгоритм повторяется, пока добавляется хотя бы одно N

	δ	
	x=a	x=b
0	1	2
1	7	3
2	5	0
3⁺	3	4
4⁺	2	6
5	5	6
6⁺	3	4
7	5	3

- п0 = { (0, 1, 2, 5, 7); (3, 4, 6) }

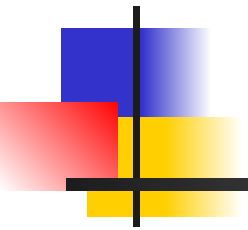
	7	6	5	4	3	2	1
0		N		N	N		
1		N		N	N		
2		N		N	N		
3	N		N				
4	N		N				
5		N					
6	N						

Начальная расстановка

	7	6	5	4	3	2	1
0	N	N	N	N	N		N
1		N		N	N	N	
2	N	N	N	N	N		
3	N		N	N			
4	N	N	N				
5		N					
6	N						

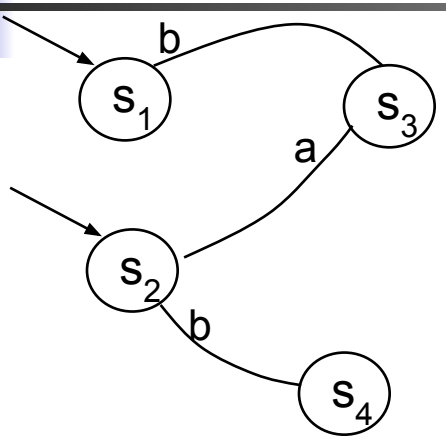
Второй просмотр (красным)
Третий не добавляет ничего

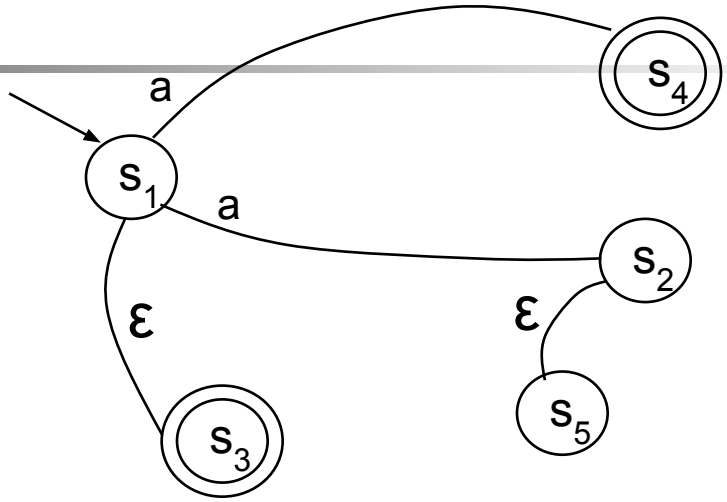
Эквивалентны:
0 и 2;
1, 5 и 7;
3 и 6;
4 само себе



Недетерминированные конечные автоматы-распознаватели

Недетерминированные конечные автоматы-распознаватели



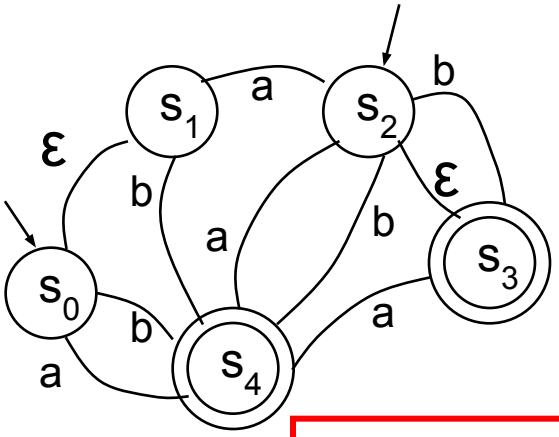


- Два начальных состояния, пустой переход, неоднозначный переход – как это понимать: как ошибки, коллизии?
- Что это за монстр? Что значит – несколько возможных переходов? Как его реализовывать? Автомат подбрасывает монету?

- Недетерминизм - *очень удобное свойство формальной модели, его можно определенным образом трактовать, даже и не реализовывая, ограничиваясь только формальными аналитическими преобразованиями.*
- Пустой переход. *В некоторых приложениях (например, построение конечно-автоматных систем логического управления) такие модели представляют системы с "невидимыми" извне событиями. Как и всегда в случае недетерминизма, какие-то ненаблюдаемые события могут действовать, а в нашей абстракции нам или невозможно, или неудобно их явно представлять.*

Основное правило для недетерминированных конечных автоматов-распознавателей

- Как понимать коллизии: а) несколько начальных состояний,
  б) несколько переходов, помеченных одним и тем же символом,
  в) переходы, помеченные пустым символом ε.



Какова реакция на aa? Она может быть РАЗНАЯ!

$s_2s_1s_0s_4$ – допускается.

$s_0s_4s_2$ – НЕ допускается.

В разных применениях можно понимать по-разному!

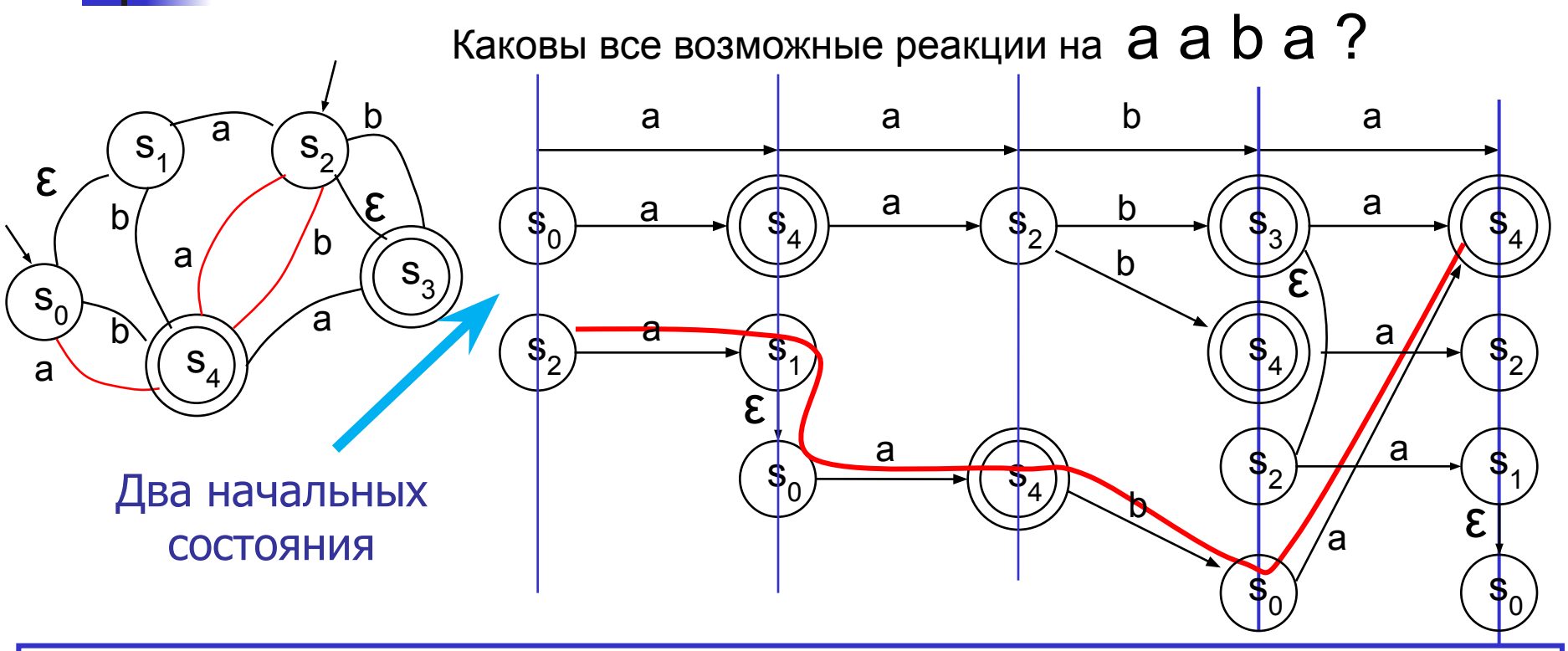
Конечный автомат распознаватель – это не модель устройства, обычно мы не реализуем его аппаратно.

Это формальная модель, которая удобна для задания языка!

Основное правило в теории формальных языков:

Цепочка допускается конечным автоматом, если существует путь, по которому эта цепочка переводит автомат из какого-нибудь начального состояния в одно из финальных состояний.

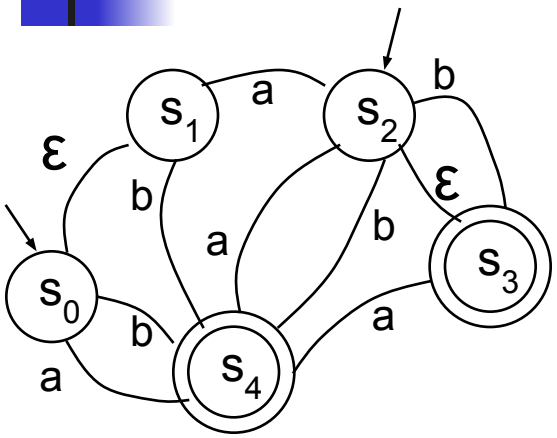
Как “работает” недетерминированный КА



Цепочка aaba допускается: s_2 -> s_1 -> s_0 -> s_4 -> s_0 -> s_4 – СУЩЕСТВУЕТ переход из одного из начальных состояний в одно из финальных состояний

Все возможные множества состояний можно считать новыми состояниями и моделировать работу КА с такими состояниями

Формальное определение недетерминированного КА

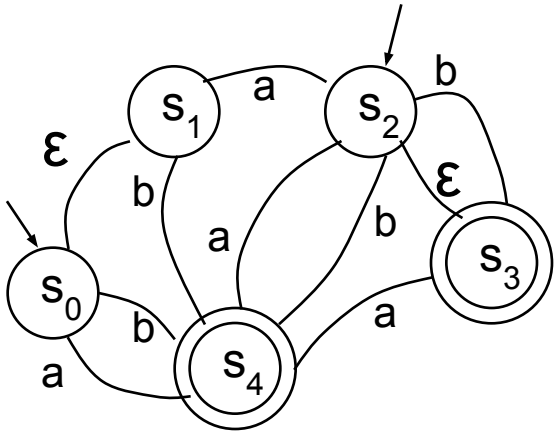


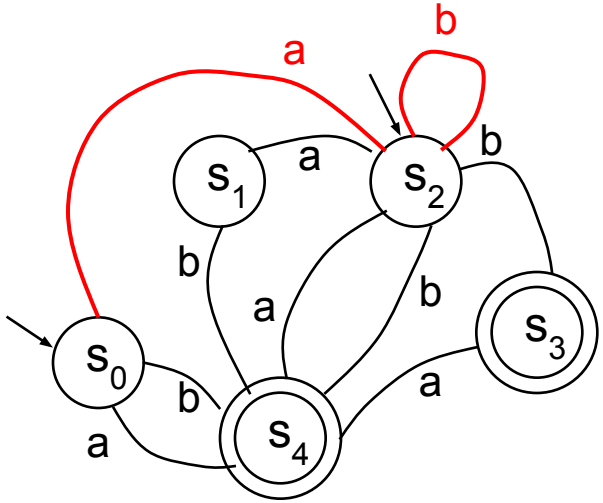
- Недетерминированный конечный автомат-распознаватель $A=(S, \Sigma, s_0, \delta, F)$, где:
  - S – конечное множество состояний
  - Σ – конечное множество входов
  - $S_0 \in S$ – множество начальных состояний
  - $\delta: S \times \Sigma \to 2^S$ – функция переходов
  - $F \subseteq S$ – множество финальных состояний

- Формальное задание автомата примера: $A=(S, \Sigma, s_0, \delta, F)$
  - $S=(s_0, s_1, s_2, s_3, s_4)$; $\Sigma = \{a, b\}$; $S_0=\{s_0, s_2\}$
  - $\delta(s_0, a)=\{s_4\}$;
    $\delta(s1, b)=\{s_4\}$;
    $\delta(s_2, a)=\{s_4\}$;
    $\delta(s_2, b)=\{s_2, s_3, s_4\}$;
    $\delta(s_3, a)=\{s_0, s_1, s_4\}$;
    ...
    $\delta(\{s_2, s_3, s_4\}, a) =\{s_0, s_1, s_3, s_4\}$;
    Спонтанные переходы $\delta(s_1, \varepsilon)=\{s_1, s_0\}$; $\delta(s_3, \varepsilon)=\{s_3, s_2\}$;
  - $F=\{s_3, s_4\}$.

2^S – это множество всех подмножеств множества S

Приведение к НДКА без ε-переходов





- Недетерминированный конечный автомат-распознаватель A=(S, Σ, s_0, δ, F), где:
  - S – конечное множество состояний
  - Σ – конечное множество входных символов
  - $S_0 \in S$ – множество начальных состояний
  - δ: $S \times \Sigma \to 2^S$ – функция переходов
  - $F \subseteq S$ – множество финальных состояний

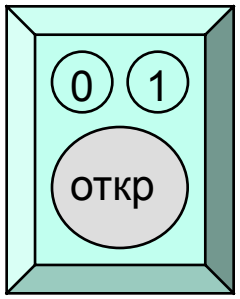
Для каждого состояния удобно определить множество его ε-преемников:

для s1 – это {s0};
для s3 – это {s2};
для остальных – пустые множества

Правило:

Если под воздействием a можем перейти в s_1, то под воздействием a можем перейти и во все состояния из множества ε-преемников состояния s_1

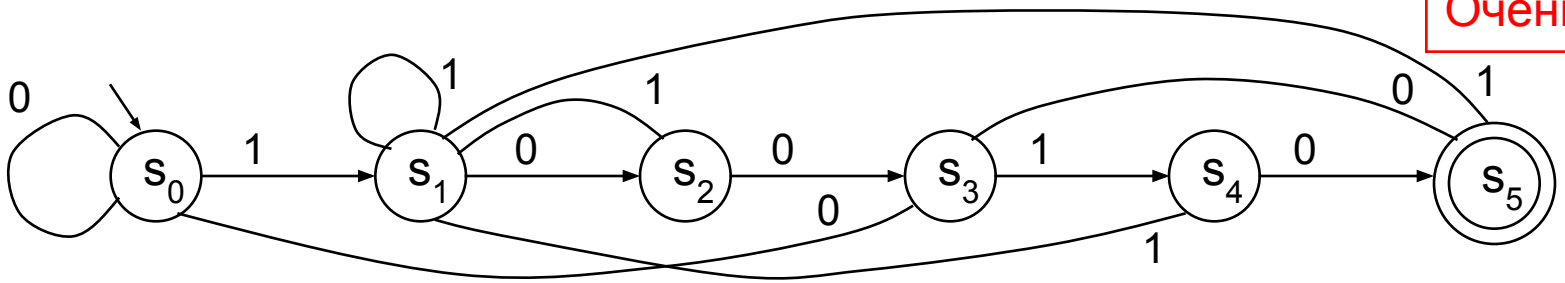
Чем удобны НДКА? Пример 1



- Проблема: построить кодовый замок.
- Замок должен открываться при наборе любого кода, заканчивающегося на 10010, например, 010011010010.
- Нажатие "откр" либо откроет дверь, либо, после неправильного набора кода, стукнет током

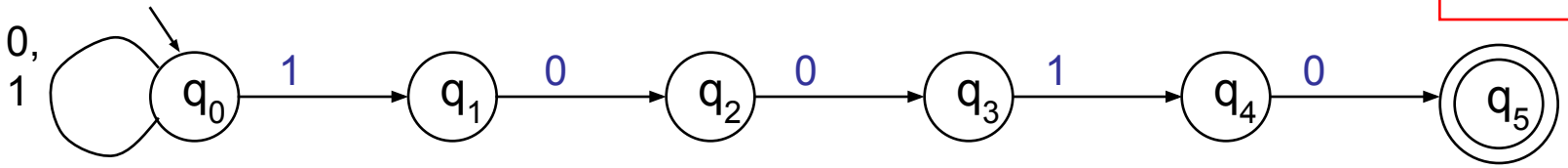
Строим детерминированный конечный автомат:

Очень сложно!



Строим НЕдетерминированный конечный автомат:

Очень просто!

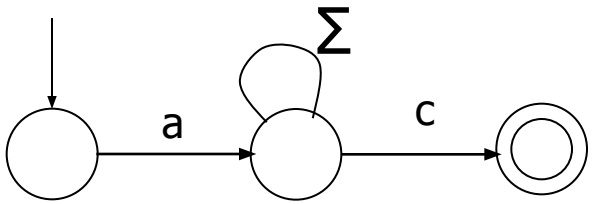


Откуда этот НДКА "знает", что цепочка закончится 10010?

Чем удобны НДКА? Пример 2

- Задача: Построить КА с входным алфавитом Σ = {a, b, c}, распознающий все цепочки, которые начинаются символом a и кончаются символом c

НЕдетерминированный автомат



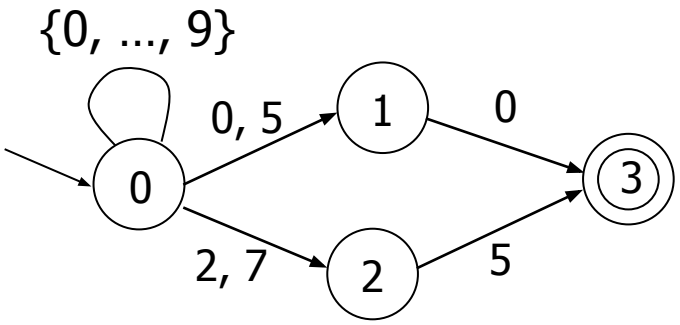
Очень просто!

Детерминированный автомат – построить сложно!

Чем удобны НДКА? Пример 3

- Задача: Построить КА с входным алфавитом Σ = {0, …, 9}, распознающий все числа, которые делятся на 25 (числа вводятся старшими разрядами вперед)
- Ясно, что это числа, заканчивающиеся на 00, 25, 50 и 75

НЕдетерминированный автомат

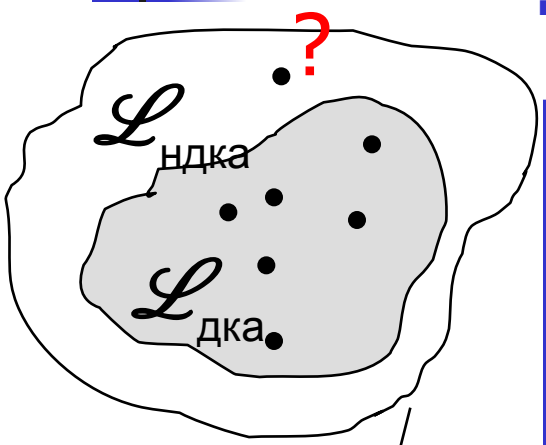


Очень просто!

Детерминированный автомат – СЛОЖНО!

- Замечание: Автомат распознает правильно только числа, состоящие не менее, чем из двух цифр
- Построить КА, распознающий также и число 0, как делящееся на 25

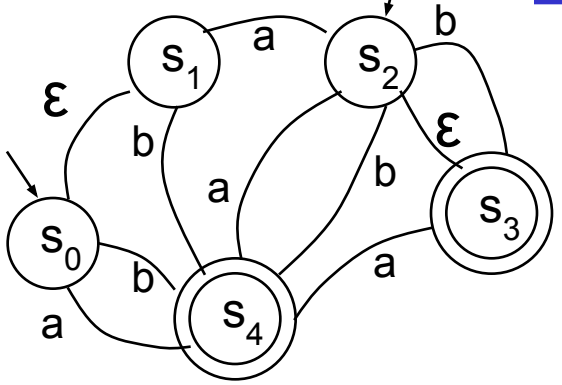
Распознающая мощность недетерминированных КА



- Ясно, что детерминированные конечные автоматы – подкласс недетерминированных КА.

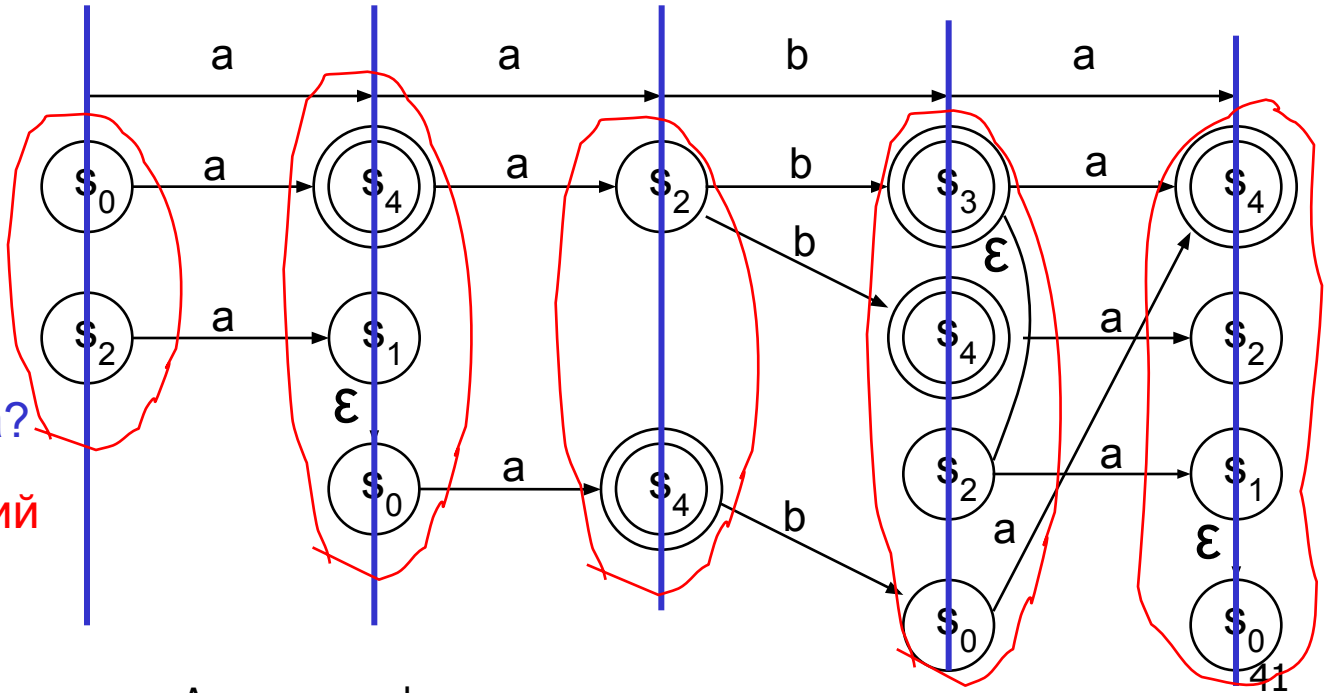
- Увеличиваются ли возможности КА по заданию языков при недетерминизме?
  (существует ли такой язык $L_{\text{НДКА}}$, который НЕ распознается никаким детерминированным КА, но распознается некоторым Недетерминированным КА?).

НЕТ!

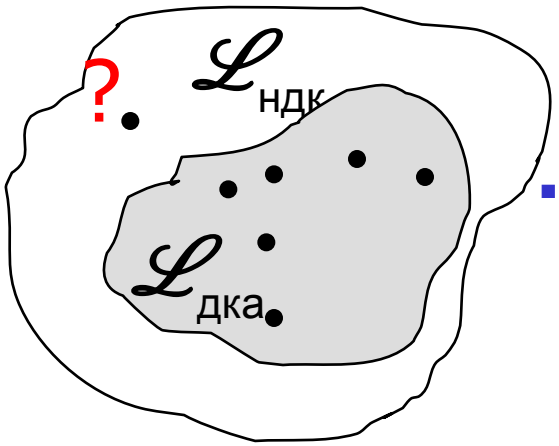


Каковы реакции на aaba?

Подмножества состояний недетерминированного автомата.



Распознающая мощность недетерминированных КА



- Теорема (Рабина-Скотта). Для любого недетерминированного конечного автомата существует эквивалентный ему детерминированный конечный автомат (т.е. автомат, допускающий тот же язык).
- Множество языков, распознаваемых НЕдетерминированными конечными автоматами совпадает с множеством языков, распознаваемых детерминированными конечными автоматами.

$$(\forall A \in NDFA)\,(\exists B \in DFA)\; A \approx B$$

или, что то же:

$$(\forall A \in NDFA)\,(\exists B \in DFA)\; L(A) = L(B)$$

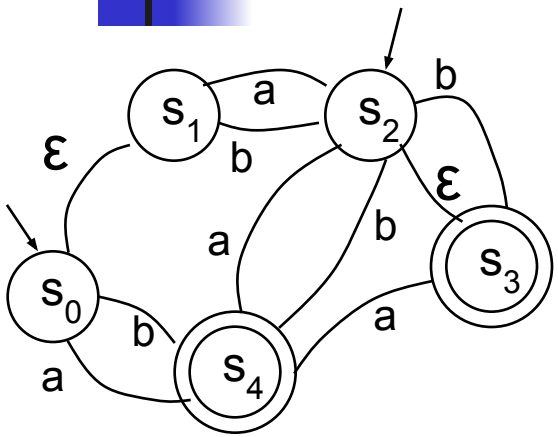
Доказательство КОНСТРУКТИВНОЕ:

В качестве состояний нового автомата берем подмножества состояний недетерминированного автомата (т.о., число состояний нового автомата в общем случае $O(2^{|S|})$, где $|S|$ - число состояний исходного недетерминированного автомата.

Начальное состояние нового автомата – ε-замыкание множества начальных состояний недетерминированного автомата.

Принимающим состоянием будет всякое ε-замыкание множества, включающего хотя бы одно принимающее состояние недетерминированного автомата.

От недетерминированного автомата к эквивалентному детерминированному автомату



δ	x=a	x=b
s0 -	s4	
s1		s2
s2-	{s0, s1}	{ s2, s3, s4}
s3+	s4	
s4+	s2	s0

Строим эквивалентный детерминированный КА:

	δ	x=a	x=b
q0-	{s0-, s2-}-	{s0, s1, s4} **q1**	{ s2, s3, s4} **q2**
q1+	{s0, s1, s4}+	{ s2, s4} **q3**	{s0 , s2} **q4**
q2+	{ s2, s3, s4}+	{s0,s1,s2,s4} **q5**	{s0,s2,s3,s4} **q6**
q3+	{ s2, s4}+	{s0, s1, s2} **q7**	{s0,s2,s3,s4} **q6**
q4	{ s0, s2}	{ s0, s1, s4} **q1**	{ s2, s3, s4} **q2**
q5+	{ s0, s1, s2, s4}+	{s0,s1,s2,s4} **q5**	{ s0,s2,s3,s4} **q6**
q6+	{ s0, s2, s3, s4}+	{s0,s1,s2,s4} **q5**	{s0,s2,s3,s4} **q6**
q7	{ s0, s1, s2}	{s0, s1, s4} **q1**	{ s2, s3, s4} **q2**

Множество начальных состояний – все те, где можем оказаться, не подавая входного символа.

Множество состояний, включающее хотя бы одно финальное состояние, является финальным.

Пример: переход из {s2, s4} по a: все те, куда можем попасть по a и a**ε.**

Минимизация алгоритмом “треугольника”

	q7	q6	q5	q4	q3	q2	q1
q0		N	N		N	N	N
q1	N			N			
q2	N			N			
q3	N						
q4		N	N				
q5	N						
q6	N						

Начальная расстановка

	7	6	5	4	3	2	1
0		N	N		N	N	N
1	N	N	N	N	N	N	
2	N			N	N		
3	N	N	N	N			
4		N	N				
5	N						
6	N						

	δ	x=a	x=b
$q0^-$	$\{s_0^-, s_2^-\}^-$	$\{s_0, s_1, s_4\}$ q1	$\{s_2, s_3, s_4\}$ q2
$q1^+$	$\{s_0, s_1, s_4\}^+$	$\{s_2, s_4\}$ q3	$\{s_0, s_2\}$ q4
$q2^+$	$\{s_2, s_3, s_4\}^+$	$\{s_0,s_1,s_2,s_4\}$ q5	$\{s_0,s_2,s_3,s_4\}$ q6
$q3^+$	$\{s_2, s_4\}^+$	$\{s_0, s_1, s_2\}$ q7	$\{s_0,s_2,s_3,s_4\}$ q6
q4	$\{s_0, s_2\}$	$\{s_0, s_1, s_4\}$ q1	$\{s_2, s_3, s_4\}$ q2
$q5^+$	$\{s_0, s_1, s_2, s_4\}^+$	$\{s_0,s_1,s_2,s_4\}$ q5	$\{s_0,s_2,s_3,s_4\}$ q6
$q6^+$	$\{s_0, s_2, s_3, s_4\}^+$	$\{s_0,s_1,s_2,s_4\}$ q5	$\{s_0,s_2,s_3,s_4\}$ q6
q7	$\{s_0, s_1, s_2\}$	$\{s_0, s_1, s_4\}$ q1	$\{s_2, s_3, s_4\}$ q2

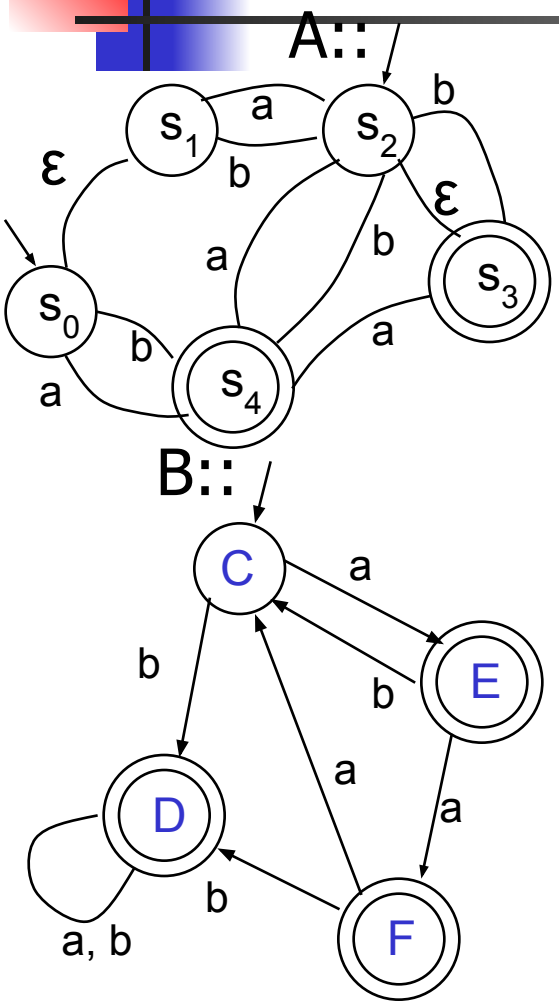
Начальное заполнение:
для каждой пары <p,q>, для которой $(p \in F) \oplus (q \in F)$, ставим N.

Многократно:
Для каждой пары <p, q>, у которой не стоит N, проверяем, стоит ли N для $<\delta(p,x), \delta(q,x)>$ хотя бы при одном x. Если стоит, то для пары <p, q> ставим N.

Алгоритм повторяется, пока добавляется хотя бы одно N.

Π_∞ = {q0, q4, q7}; {q2, q5, q6}; {q1}; {q3}

Минимизация построенного детерминированного автомата, эквивалентного исходному



	δ	x=a	x=b
$q0^-$	$\{s_0^-, s_2^-\}^-$	$\{s_0, s_1, s_4\}$ q1	$\{s_2, s_3, s_4\}$ q2
$q1^+$	$\{s_0, s_1, s_4\}^+$	$\{s_2, s_4\}$ q3	$\{s_0, s_2\}$ q4
$q2^+$	$\{s_2, s_3, s_4\}^+$	$\{s_0,s_1,s_2,s_4\}$ q5	$\{s_0,s_2,s_3,s_4\}$ q6
$q3^+$	$\{s_2, s_4\}^+$	$\{s_0, s_1, s_2\}$ q1	$\{s_0,s_2,s_3,s_4\}$ q6
q4	$\{s_0, s_2\}$	$\{s_0, s_1, s_4\}$ q1	$\{s_2, s_3, s_4\}$ q2
$q5^+$	$\{s_0, s_1, s_2, s_4\}^+$	$\{s_0,s_1,s_2,s_4\}$ q5	$\{s_0,s_2,s_3,s_4\}$ q6
$q6^+$	$\{s_0, s_2, s_3, s_4\}^+$	$\{s_0,s_1,s_2,s_4\}$ q5	$\{s_0,s_2,s_3,s_4\}$ q6
q7	$\{s_0, s_1, s_2\}$	$\{s_0, s_1, s_4\}$ q1	$\{s_2, s_3, s_4\}$ q2

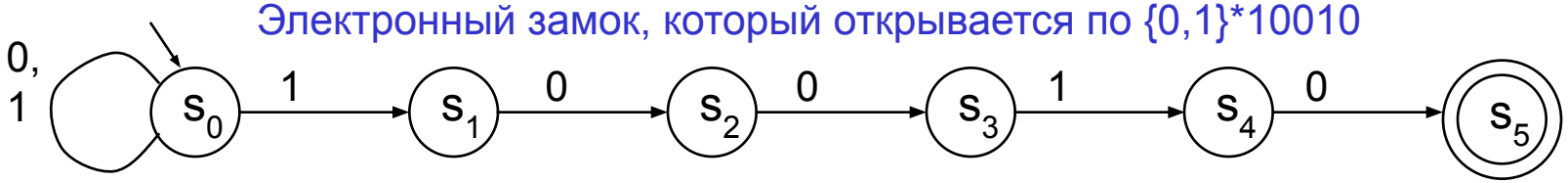
- п0 = {q0, q4, q7}=A; {q1, q2, q3, q5, q6}=B
- п1 = {q0, q4, q7}=C; {q2, q5, q6}=D; {q1}=E; {q3}=F
- п2 = {q0, q4, q7}; {q2, q5, q6}; {q1}; {q3} = п1 = $П_\infty$

C D E F

У детерминированного автомата B, эквивалентного заданному недетерминированному автомату A, число состояний может быть как больше, так и меньше, чем у A.

Пример перехода от недетерминированного автомата к эквивалентному детерминированному

Электронный замок, который открывается по {0,1}*10010

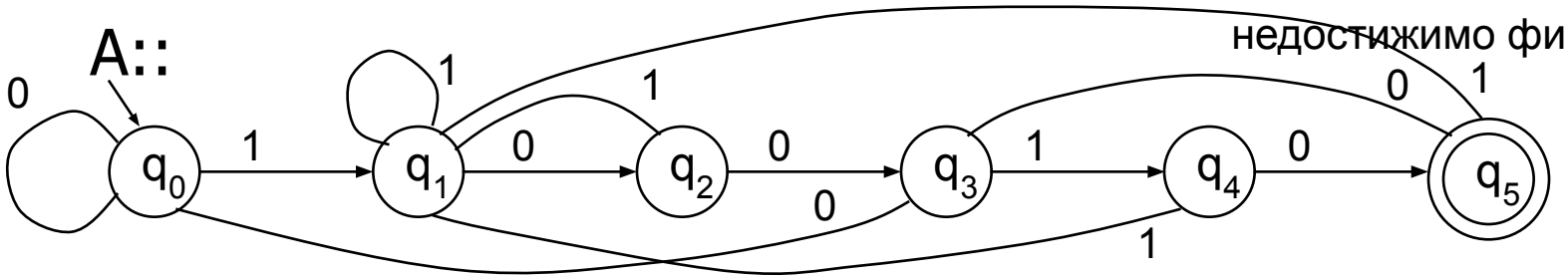


δ	x=0	x=1
s^-_0	s_0	$\{s_0, s_1\}$
s_1	s_2	Sош
s_2	s_3	Sош
s_3	Sош	s_4
s_4	s_5	Sош
s^+_5	Sош	s_0

	δ	x=0	x=1
q^-_0	$\{s^-_0\}$	$\{s_0\}$	$\{s_0, s_1\}$
q_1	$\{s_0,s_1\}$	$(s_0,s_2\}$	$\{s_0, s_1\}$
q_2	$\{s_0,s_2\}$	$(s_0,s_3\}$	$\{s_0, s_1\}$
q_3	$\{s_0,s_3\}$	$\{s_0\}$	$\{s_0, s_1, s_4\}$
q_4	$\{s_0, s_1, s_4\}$	$\{s_0, s_2, s_5\}$	$\{s_0, s_1\}$
q^+_5	$\{s_0, s_2, s_5\}$	$\{s_0, s_3\}$	$\{s_0, s_1\}$

Автомат А, который строили с большим трудом, построен автоматически из недетерминированного автомата

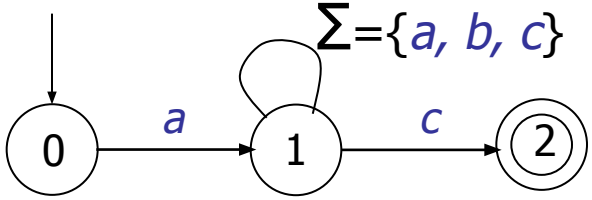
Sош – это ошибочное состояние, из которого недостижимо финальное



Зачем нужны недетерминированные КА? Пример 2

- Задача: Построить КА с входным алфавитом Σ = {*a*, *b*, *c*}, распознающий все цепочки, которые начинаются символом *a* и кончаются символом *c*.

НЕдетерминированный автомат



Очень просто!

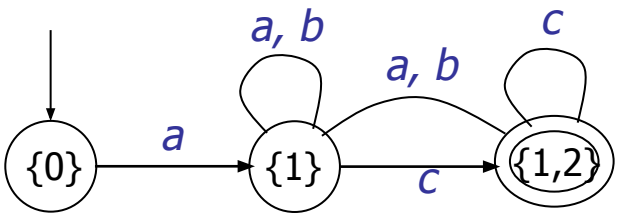
δ	*a*	*b*	*c*
0 -	1		
1	1	1	{1, 2}
2 +			

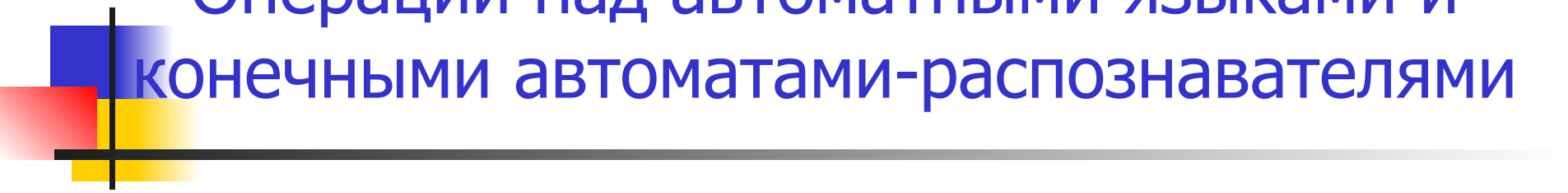
s^- - начальное состояние

s^+ - финальное состояние

Детерминированный автомат – СТРОИМ!

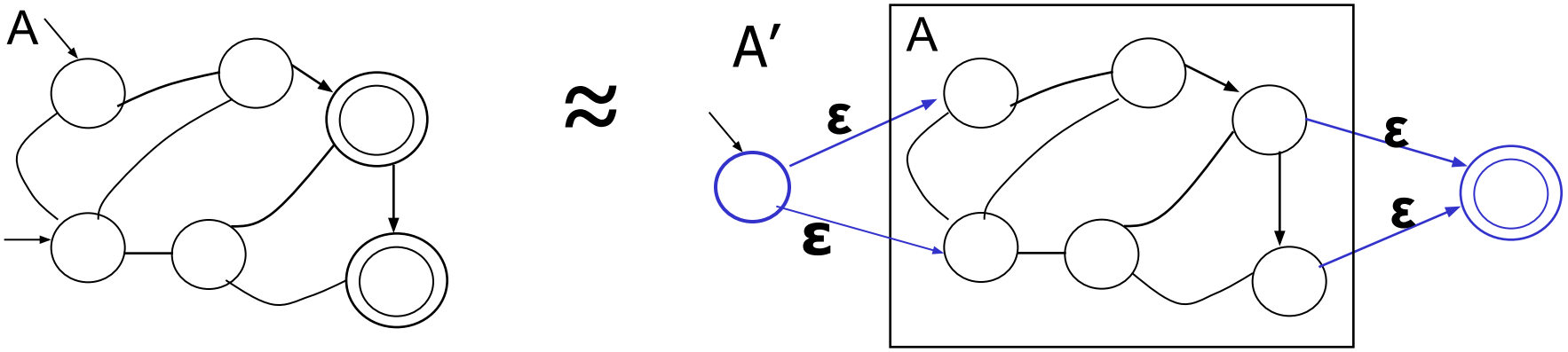
δ	*a*	*b*	*c*
{ 0 } -	{ 1 }	{ }	{ }
{ 1 }	{ 1 }	{ 1 }	{1, 2}
{1, 2} +	{ 1 }	{ 1 }	{1, 2}





Операции над автоматными языками и конечными автоматами-распознавателями

- Используя переход, помеченный пустым символом **ε**, ЛЮБОЙ конечно-автоматный распознаватель можно представить автоматом только с одним начальным и одним финальным состоянием.
- Из начального состояния переходы только выходят, в финальное – только входят.

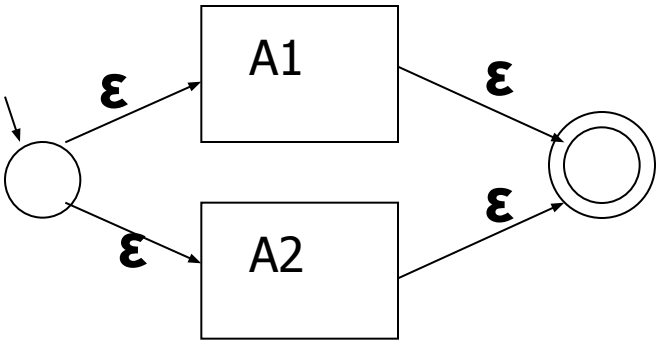


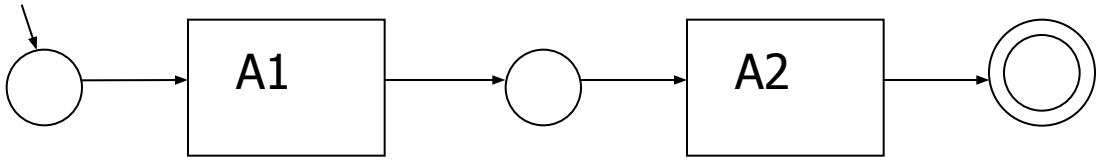
Очевидно, что автоматы A′ и A эквивалентны: они распознают один и тот же язык.

Операции над автоматными языками

- Теорема. Если L_1 и L_2 – автоматные языки, то автоматными являются также их объединение $L_1 \cup L_2$, пересечение $L_1 \cap L_2$. конкатенация $L_1 \cdot L_2$, дополнение $\Sigma - L_1$, итерация L_1^*

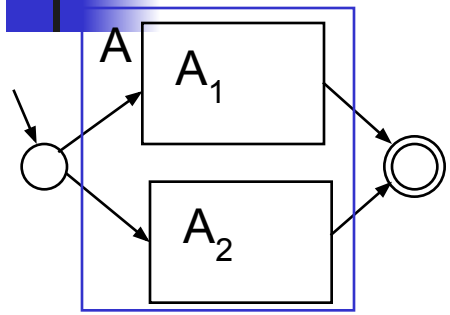
- Доказательство. Если L1 и L2 – автоматные языки, то для них существуют конечные автоматы, распознающие их. Пусть это автоматы A1 и A2. Можно считать, что A1 и A2 одно начальное и одно принимающее состояние

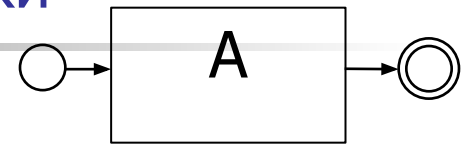




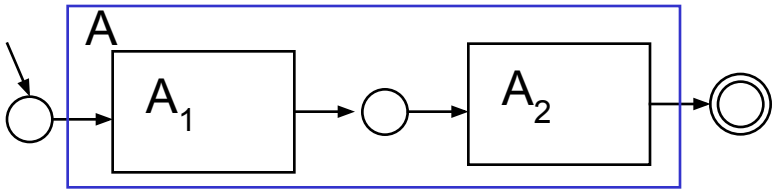
- Объединение двух автоматных языков L1 ∪ L2 допускает автомат A1 ∪ A2
- Пересечение двух автоматных языков L1 ∩ L2 допускает автомат A1 × A2
- Конкатенацию двух автоматных языков L1 L2 допускает автомат A1 · A2
- Дополнение $\Sigma - L_1$ автоматного языка L1 распознает автомат, полученный из A1 с помощью простой операции: непринимающие состояния A1 сделаем принимающими, а принимающие состояния A1 сделаем непринимающими

Операции над автоматами и автоматными языками, дающие в результате автоматные языки

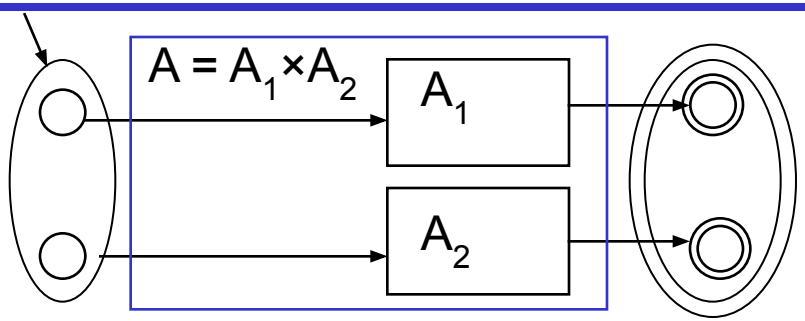




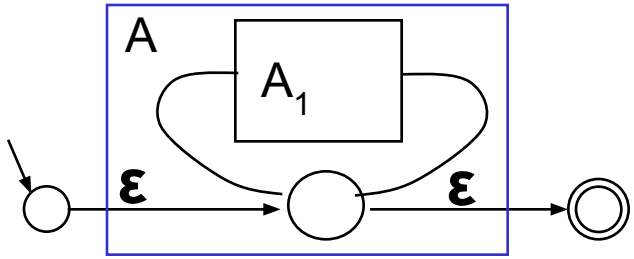
- $A = A_1 \cup A_2$
- $L_A = L_{A1} \cup L_{A2}$



- $A = A_1 \cdot A_2$
- $L_A = L_{A1} \cdot L_{A2}$



- $A = A_1 \times A_2$
- $L_{A1 \times A2} = L_{A1} \cap L_{A2}$



- $A = A_1^*$
- $L_A = L_{A1}^*$

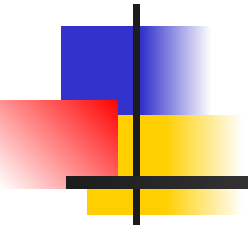
Резюме: операции над автоматными языками

- Теорема. Множество автоматных языков замкнуто относительно операций:
  - дополнения,
  - объединения,
  - пересечения,
  - конкатенации,
  - итерции (звезда Клини).

Иными словами, если L1 и L2 – автоматные языки, то автоматными будут также:

- дополнение $\Sigma - L1$;
- объединение $L1 \cup L2$;
- пересечение $L1 \cap L2$;
- конкатенация $L1 \cdot L2$;
- итерция (звезда Клини) $L1^*$

исходных языков.



Пример применения недетерминированных КА: Римская система счисления

Что написано на золотой медали Альфреда Нобиля?

MDCCCXXXIII - MDCCCXCVI

1833 - 1896

Зачем нужны недетерминированные КА? Пример

Римская система счисления - конечный язык.

I = 1
V = 5
X = 10
L = 50
C = 100
D = 500
M = 1000

IV = 4
IX = 9
XL = 40
XC = 90
CD = 400
CMXCVI = 996
0 – не представим!!!
(представим пустой строкой)

Примеры чисел

0	отсутствует
3	III
4	IV
8	VIII
32	XXXII
99	CXIX
665	DCLXV
889	DCCCLXXXIX
1989	MCMLXXXIX
2009	MMIX
3999	MMMCMXCIX

Каковы формальные правила (грамматика) построения римских чисел?

Построить детерминированный автомат довольно сложно.

Правила построения римских чисел: Википедия

I = 1
V = 5
X = 10
L = 50
C = 100
D = 500
M = 1000

IV = 4
LXXXVII = 87
XLIII = 43
XC = 90
CD = 400
CMXCII = 992

- Если меньшие цифры следуют за большими, то их значения суммируются.
- Только цифры, являющиеся степенью 10, могут повторяться до 3-х раз (т.е. V, L и D повторяться не могут), любые цифры могут отсутствовать.
- Меньшие цифры могут предшествовать большим. Значение числа получается вычитанием меньшей цифры из большей. Такое предшествование возможно в следующих случаях:
  - меньшая цифра может быть только степенью 10 (т.е. C, X, I);
  - ее значение может быть только либо одной пятой либо одной десятой значения большей (например, можно XL (=40) и XC (=90), но XM и XD – нельзя);
  - она либо первая цифра в числе, либо ей предшествует цифра, значение которой, по крайней мере, в 10 раз больше ее значения (например, MXL(=1040) можно, LXL – нельзя);
  - если за большей цифрой следует какая-либо цифра, она должна быть меньше, чем та, которая предшествует большей цифре.

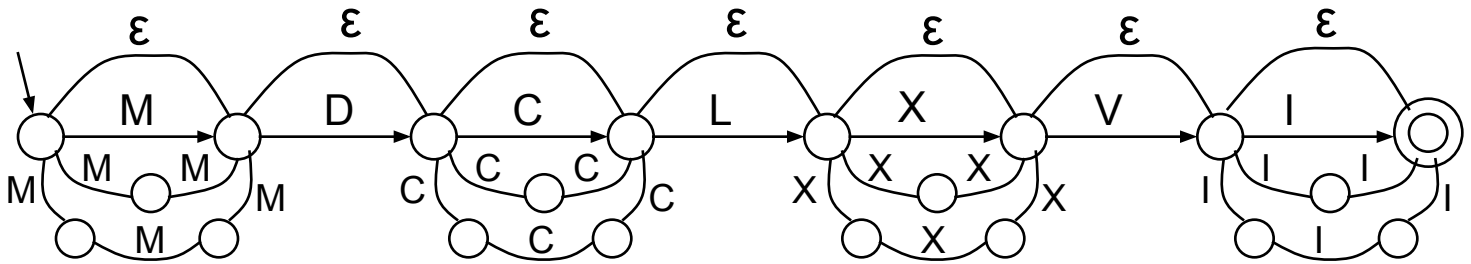
Правила построения римских чисел: формально (1)

I = 1
V = 5
X = 10
L = 50
C = 100
D = 500
M = 1000

IV = 4
LXXXVII = 87
XLIII = 43
XC = 90
CD = 400
CMXCII = 992

- Меньшие цифры обычно следуют за большими.

- Цифры, являющиеся степенью 10, могут повторяться до 3-х раз (т.е. V, L и D повторяться не могут), любые цифры могут отсутствовать.



Построить детерминированный автомат сложно.

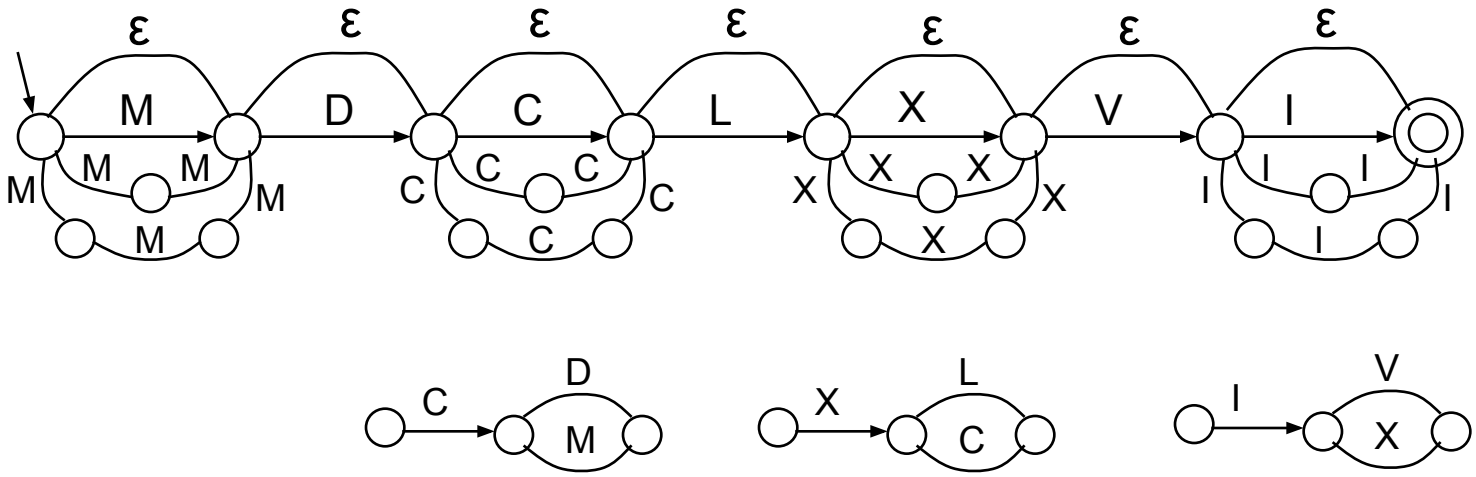
Строим недетерминированный.

Правила построения римских чисел: формально (2)

I = 1
V = 5
X = 10
L = 50
C = 100
D = 500
M = 1000

IV = 4
LXXXVII = 87
XLIII = 43
XC = 90
CD = 400
CMXCII = 992

- Меньшие могут предшествовать большим. Меньшая цифра:
  - может быть только степенью 10 (т.е. C, X, I);
  - ее значение может быть только либо одной пятой либо одной десятой значения большей.



Построить детерминированный автомат сложно.

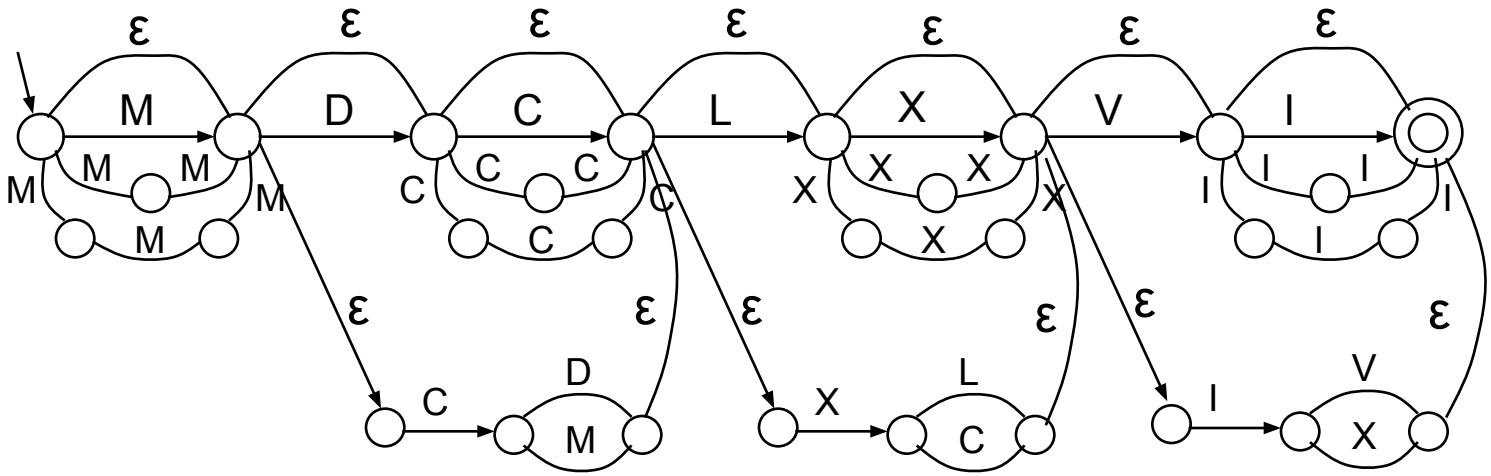
Строим недетерминированный.

Правила построения римских чисел: формально (3)

I = 1
V = 5
X = 10
L = 50
C = 100
D = 500
M = 1000

IV = 4
LXXXVII = 87
XLIII = 43
XC = 90
CD = 400
CMXCII = 992

- Меньшая цифра либо первая в числе, либо ей предшествует цифра, значение которой, по крайней мере, в 10 раз больше ее значения.
- Если за большей цифрой следует какая-либо цифра, она должна быть меньше, чем та, которая предшествует большей цифре.



Построить детерминированный автомат сложно.

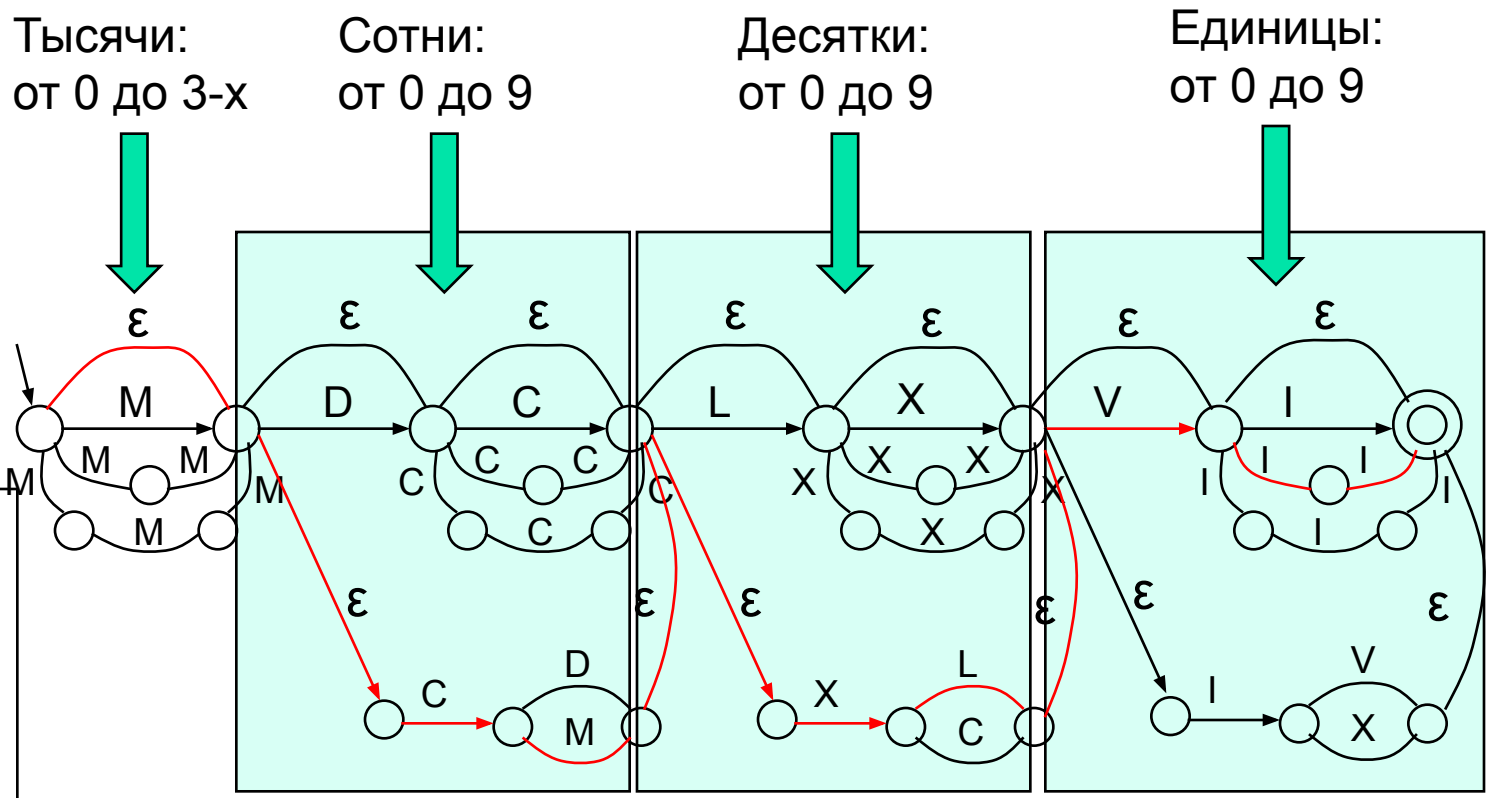
Построили недетерминированный конечный автомат.

Возможные значения римских чисел

Сколько всего цепочек? $4 \times 10 \times 10 \times 10 = 4000$

I = 1
V = 5
X = 10
L = 50
C = 100
D = 500
M = 1000

IV = 4
LXXXVII = 87
XLIII = 43
XC = 90
CD = 400
CMXCVII = 997



Все правильные цепочки языка – это числа в Римской системе счисления. Вывод - путь из начального в финальное состояние). Пример вывода □: CMXLVII = 947.

Правила построения римских чисел: формально (4)

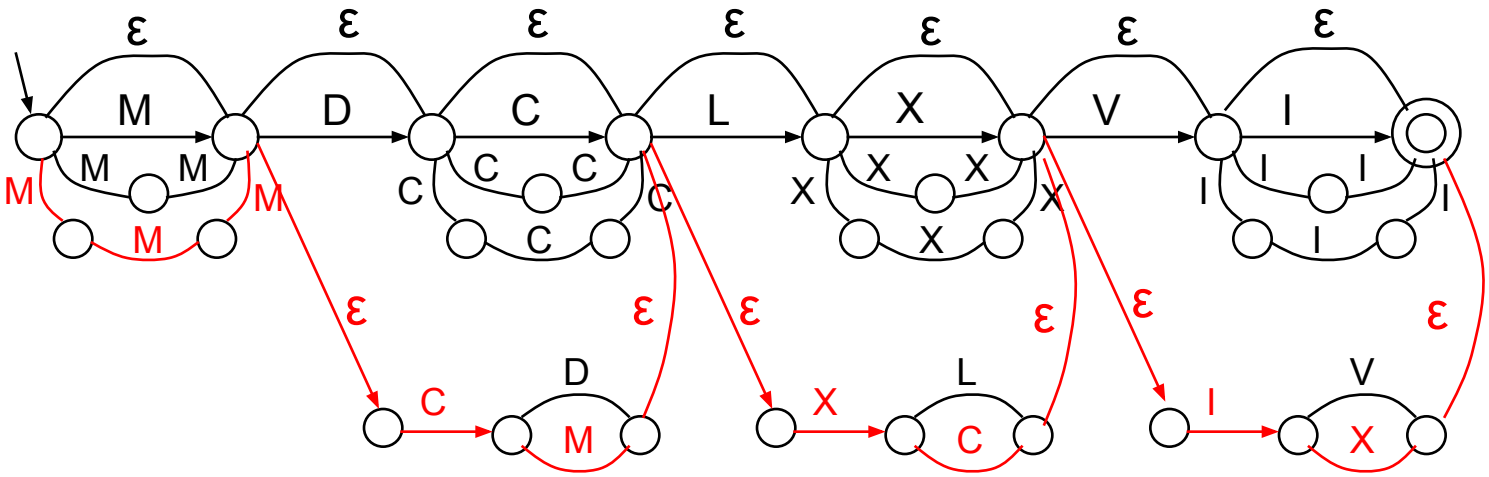
I = 1
V = 5
X = 10
L = 50
C = 100
D = 500
M = 1000

IV = 4
LXXXVII = 87
XLIII = 43
XC = 90
CD = 400
CMXCII = 992

Сколько всего цепочек? $4 \times 10 \times 10 \times 10 = 4000$

Минимальное число 0 – отсутствие символов.

Максимальное число MMMCMXCIX = 3999.



Римская система счисления неудобна.

Большая сложность выполнения арифметических операций:
XLVIII + V I = L I V – кто сможет запомнить?

Недетерминированные автоматы: общее понимание

- Недетерминированные автоматы нельзя рассматривать, как физические устройства, которые читают входные символы, принимают решение, "угадывают", в какое из состояний перейти, "обладают интуицией", чтобы выбрать "правильный путь", откуда-то "зная", какими будут следующие символы входного слова (как в [1]).
- Недетерминированный автомат-распознаватель языка – это модель, математическая абстракция, абстрактное порождение мысли, ее бессмысленно реализовывать. С ней можно выполнять некоторые формальные операции: анализ, эквивалентные преобразования.
- По каждому недетерминированному автомату-распознавателю можно построить эквивалентный ему детерминированный автомат, а такой автомат можно реализовать и программно, и аппаратно.

Недетерминированные автоматы являются формализмом, не распознающим, а порождающим языки.
Они позволяют удобно и просто задавать языки формально.

[1] Курс "Теория алгоритмов", С.Ю.Подзоров, НГУ.

Заключение

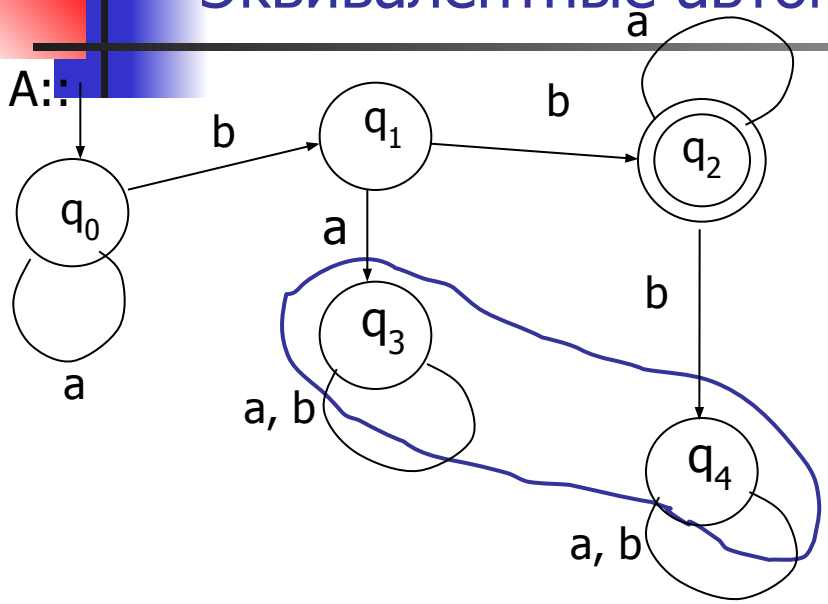
- Операции над КА-распознавателями:
  - “trimming” – приведение, т.е. выбрасывание недостижимых и некодостижимых состояний в КА;
  - по языку L, распознаваемому заданным КА, построение автомата, распознающего дополнение языка L,т.е. языка Σ* - L;
  - построение синхронной композиции двух КА-распознавателей = построение автомата, распознающего пересечение двух автоматных языков;
  - проверка эквивалентности двух КА-распознавателей;
  - минимизация КА-распознавателя;
  - построение по недетерминированному КА эквивалентного детерминированного КА-распознавателя;
  - построение КА, распознающего объединение и конкатенацию двух автоматных языков, заданных своими распознающими КА.

Заключение (2)

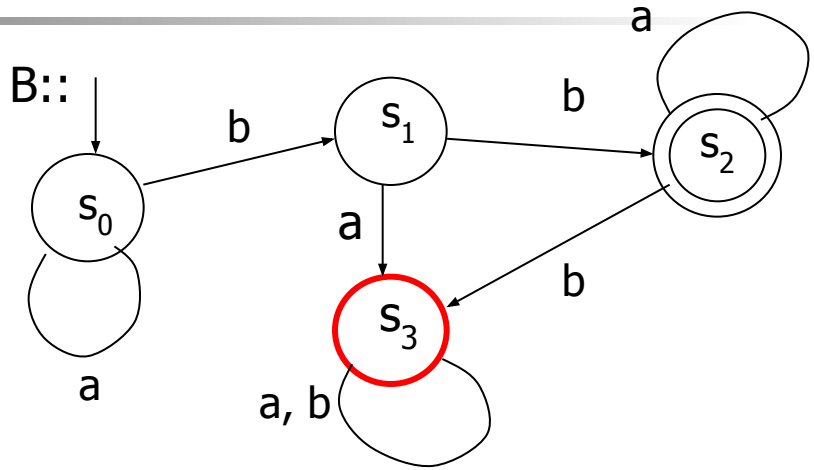
- Существует простой алгоритм проверки эквивалентности КА. Для такой проверки строится синхронная композиция автоматов.
- Синхронная композиция автоматов определяет пересечение двух автоматных языков.
- Конечные автоматы-распознаватели могут быть не минимальными. Алгоритм минимизации КА прост, он похож на алгоритм минимизации автоматов-преобразователей.
- Недетерминированный КА – это НЕ операционное устройство, которое действует, функционирует, принимая на вход цепочку. Это математическая модель, удобная для задания, для порождения языков.
- Недетерминизм КА не увеличивает возможностей распознавания языков: для любого недетерминированного КА существует эквивалентный ему детерминированный КА.
- Автоматные языки замкнуты относительно операций объединения, пересечения, конкатенации и дополнения.

Спасибо за внимание

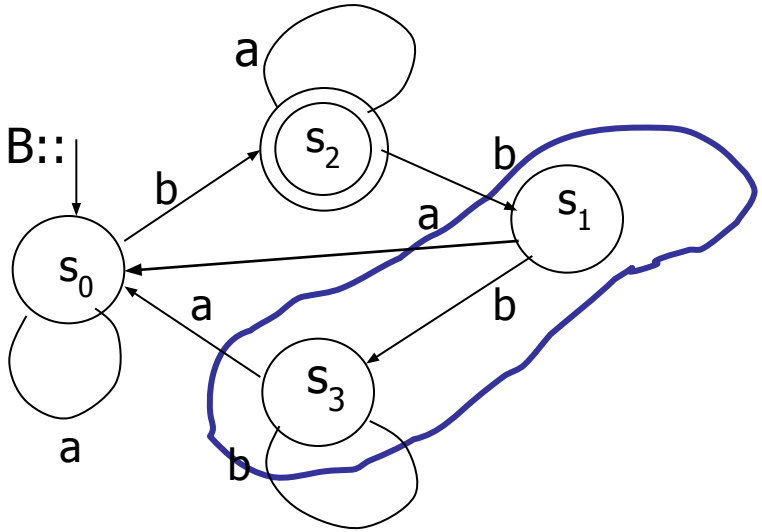
Эквивалентные автоматы-распознаватели: ПРИМЕРЫ



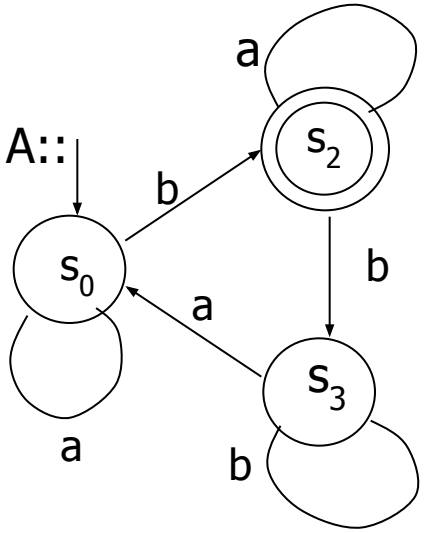
≈



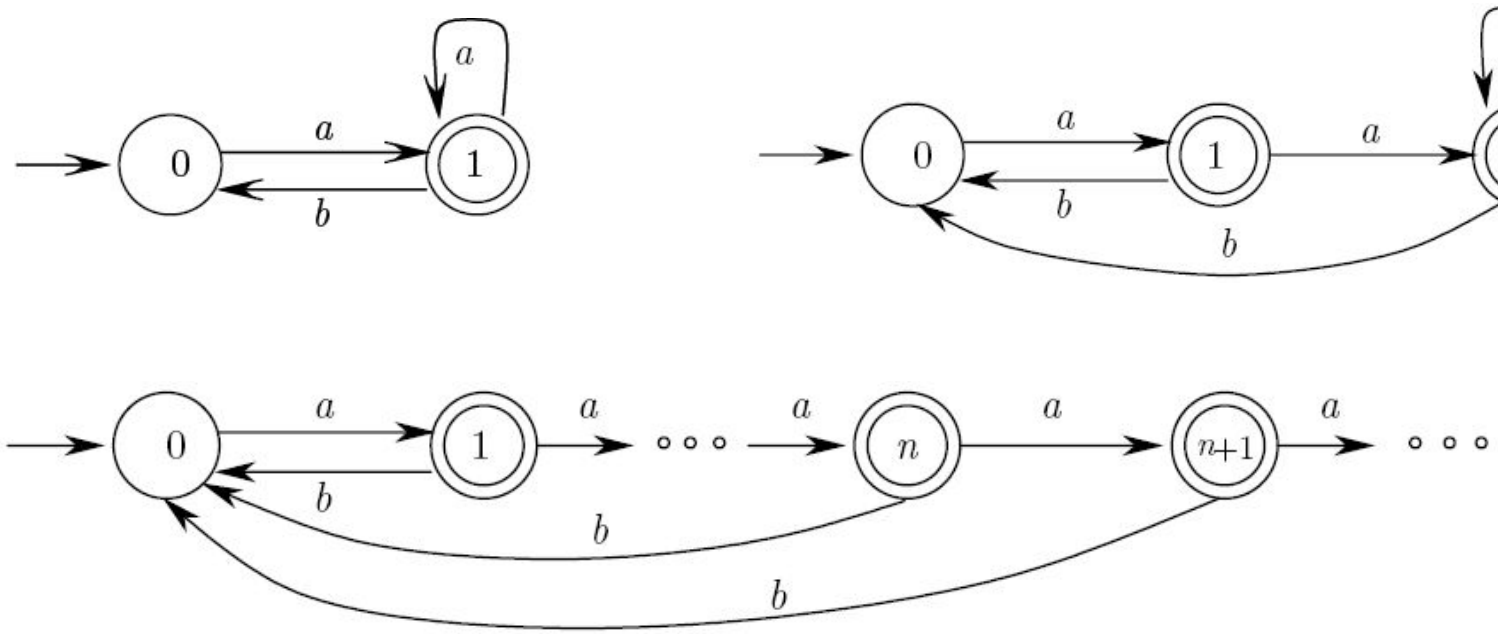
Все состояния, из которых НЕдостижимы
финальные состояние, эквивалентны



≈



Эквивалентные автоматы-распознаватели. ПРИМЕР



- Все три автомата-распознавателя эквивалентны
- Они распознают один и тот же язык
- Третий пример показывает, что автомат с бесконечным числом состояний может быть эквивалентным конечному автомату