



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

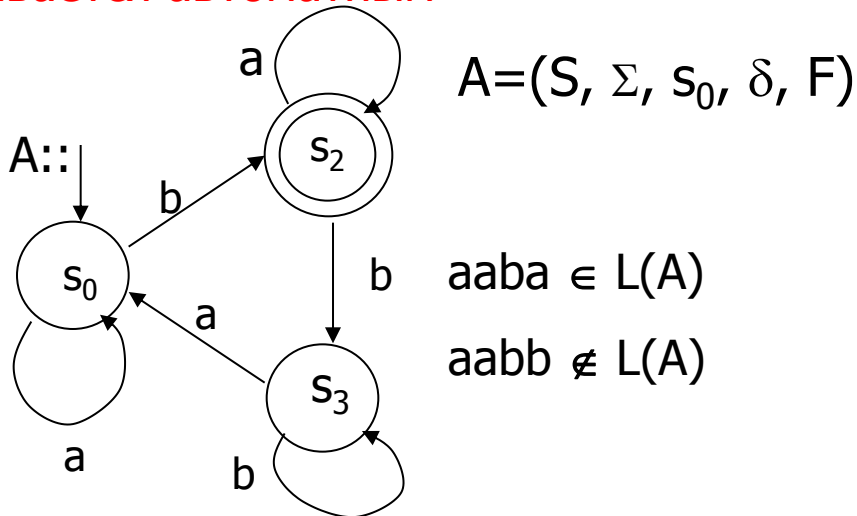


Структура курса

- 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

Автоматные языки и конечные автоматы

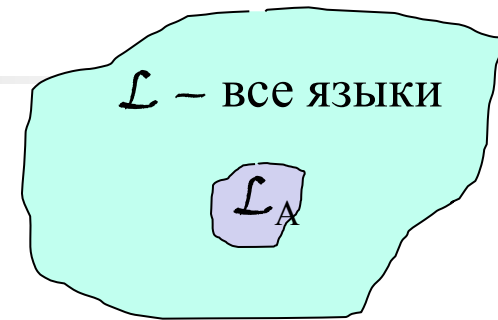
Любой конечный автомат-распознаватель определяет какой-нибудь язык
Язык, для которого существует распознающий его конечный автомат,
называется автоматным



S – множество состояний
 Σ – входной алфавит
 s_0 – начальное состояние
 δ – функция переходов
 $F \subseteq S$ – множество финальных состояний

- Конечных автоматов-распознавателей бесконечное число. И автоматных языков тоже бесконечное число. Но есть языки, которые не являются автоматными
- Неавтоматные языки – это такие, для которых не существует распознающего их конечного автомата. Это **по своей структуре** более сложные языки. Лемма о накачке позволяет доказать, что конкретный язык не автоматный
- **Примеры неавтоматных языков:**
 - язык вложенных скобок (и любой язык, допускающий вложенные конструкции),
 - множество цепочек с одинаковым числом вхождений символов a и b ,
 - все языки программирования высокого уровня, естественные языки,

Необходимость изучения автоматных языков и конечных автоматов-распознавателей

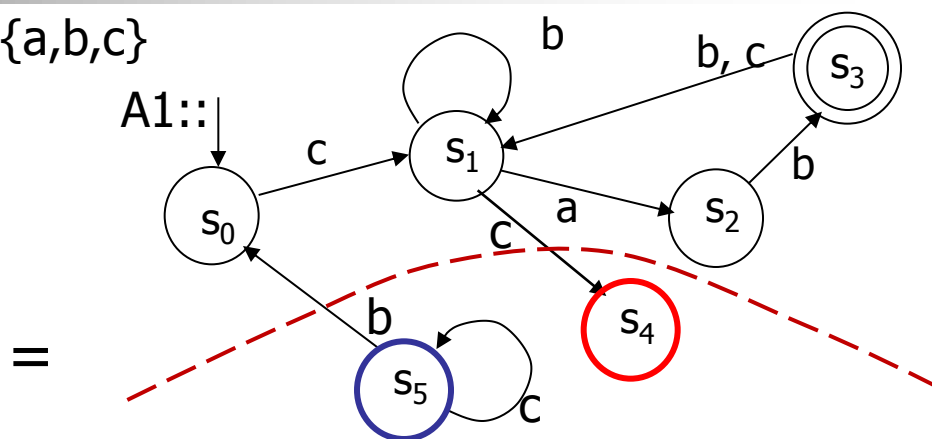
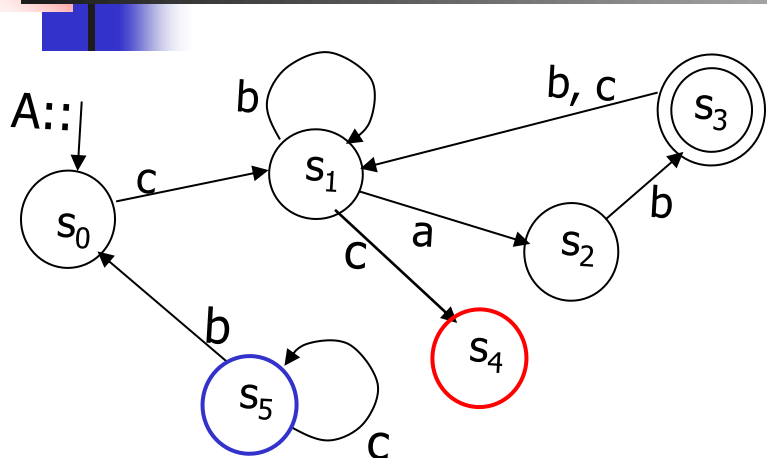


- Автоматные языки составляют лишь подкласс (довольно узкий) всех возможных языков
- В то же время, автоматными языками являются многие языки:
 - Специализированные языки,
 - Языки управления ОС,
 - Многие скриптовые языки,
 - Фрагменты языков высокого уровня (имена, константы, комментарии, ...) ,
 - Все регулярные языки (т.е. языки, задаваемые регулярными выражениями),
 -
- Существует множество важных и интересных практических применений автоматных языков

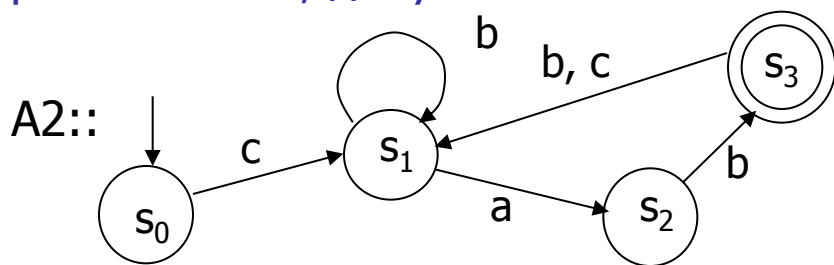


Операции над автоматами: тримминг

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!



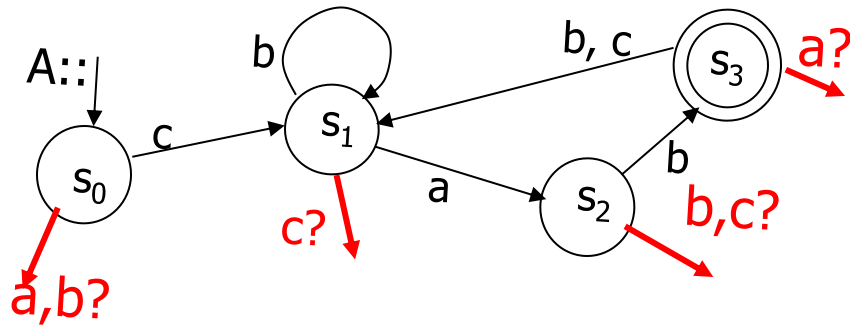
A2 – приведенный автомат

$$L(A) = L(A2)$$

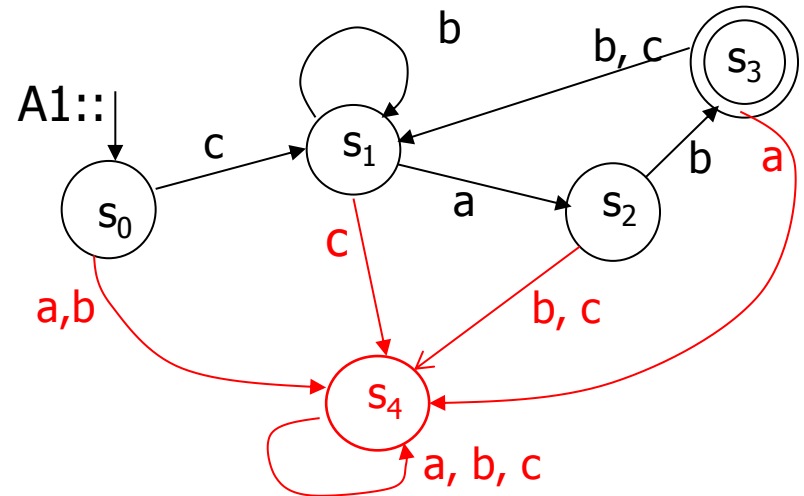
- Автомат является приведенным (trimmed, “*подстриженным*”) тогда и только тогда, когда все его состояния достижимы и из любого состояния существует путь в какое-нибудь финальное состояние

Полный конечный автомат

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



=

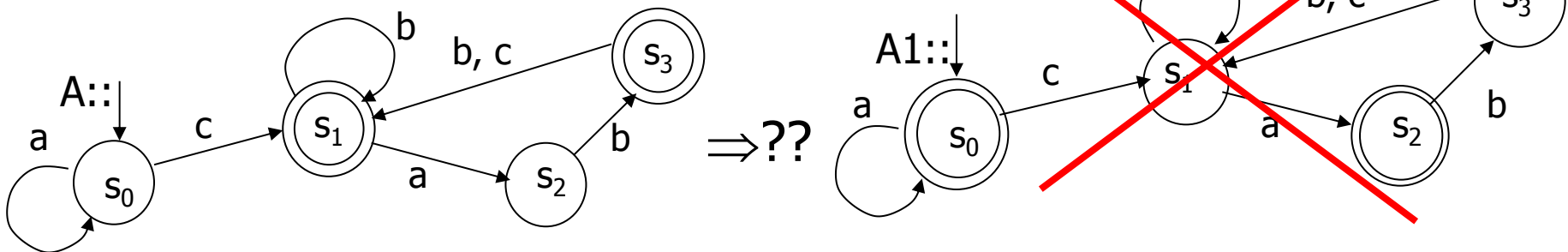


- Если переходы в автомате-распознавателе под воздействием некоторых событий не определены, то это означает, что соответствующие цепочки входного языка ошибочны, они не допускаются, не принадлежат языку, распознаваемому этим автоматом
- Для построения полного автомата вводим новое **недопускающее** состояние, и все "повисшие" переходы связываем с ним
- Полученный полный автомат с точки зрения распознавания языка эквивалентен исходному автомату

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^*



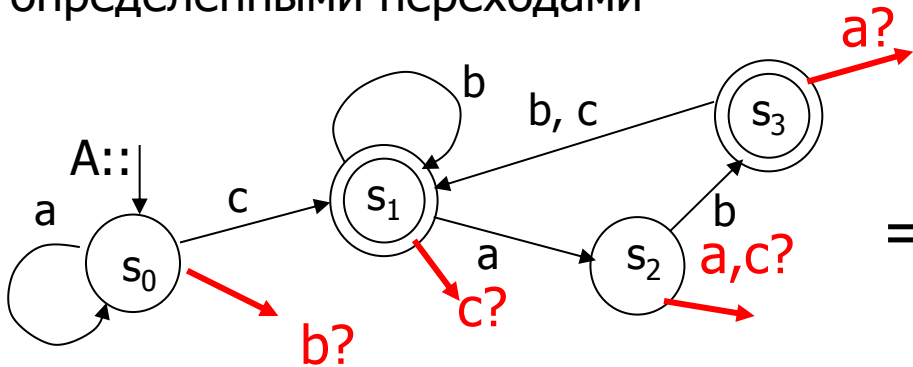
$$L(A1) = (?) \Sigma^* - L(A)$$

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

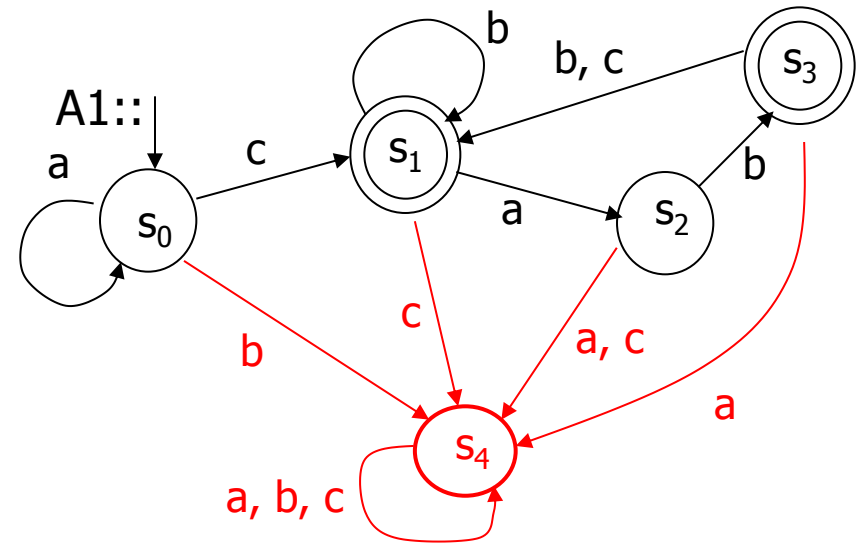
Где ошибка??

Распознавание дополнения автоматного языка

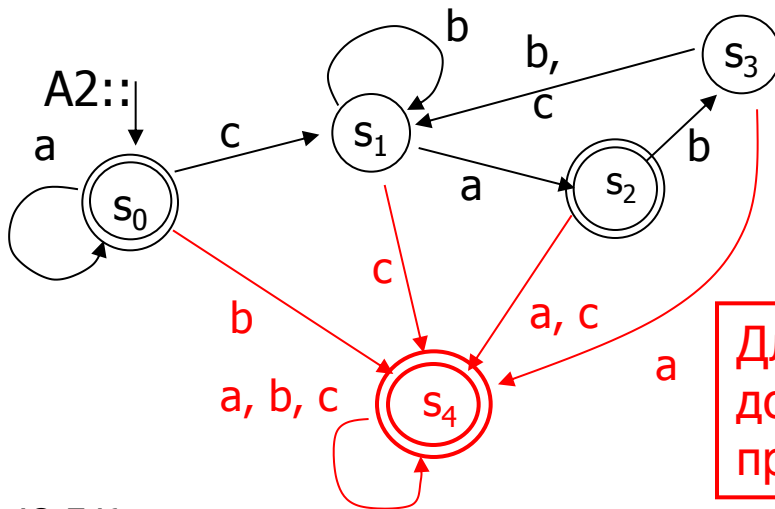
1: Заданный КА с неполностью определенными переходами



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

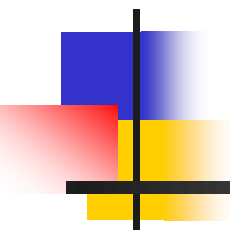


3: Финальные состояния делаем не финальными, и наоборот $L(A2) = \Sigma^* - L(A)$



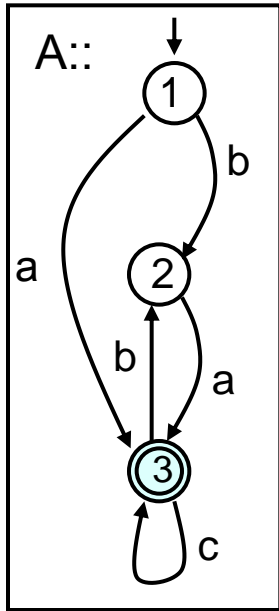
Все цепочки, которые **НЕ** переводят A в заключительное состояние, принадлежат множеству $\Sigma - L_A$, т.е. дополнению языка L_A

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

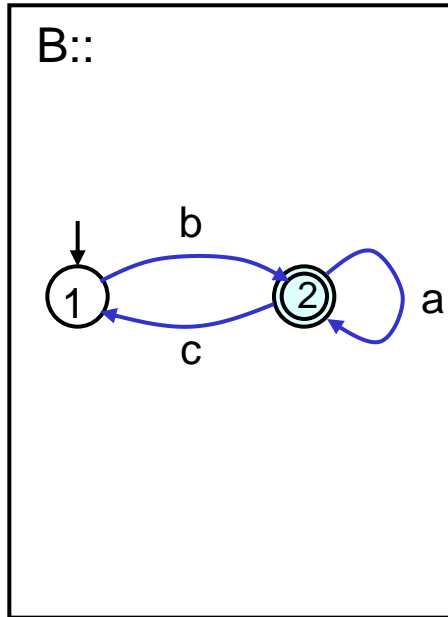


Операции над автоматами: синхронная и асинхронная композиция

Как найти пересечение автоматных языков?



×

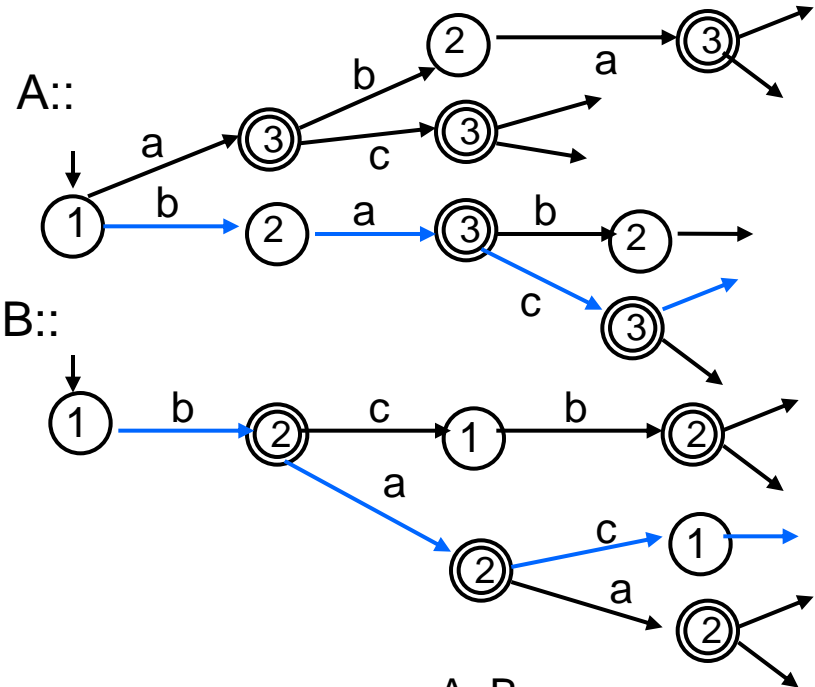


$L_A = \{ba, \text{acc}, \text{bacba}, \dots\}$

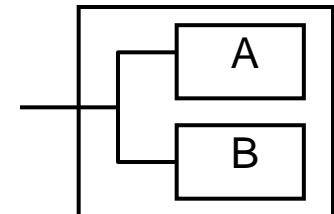
$L_B = \{b, \text{ba}, \text{bacb}, \text{bacba}, \dots\}$

$L_A \cap L_B = ?$ – непросто!

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

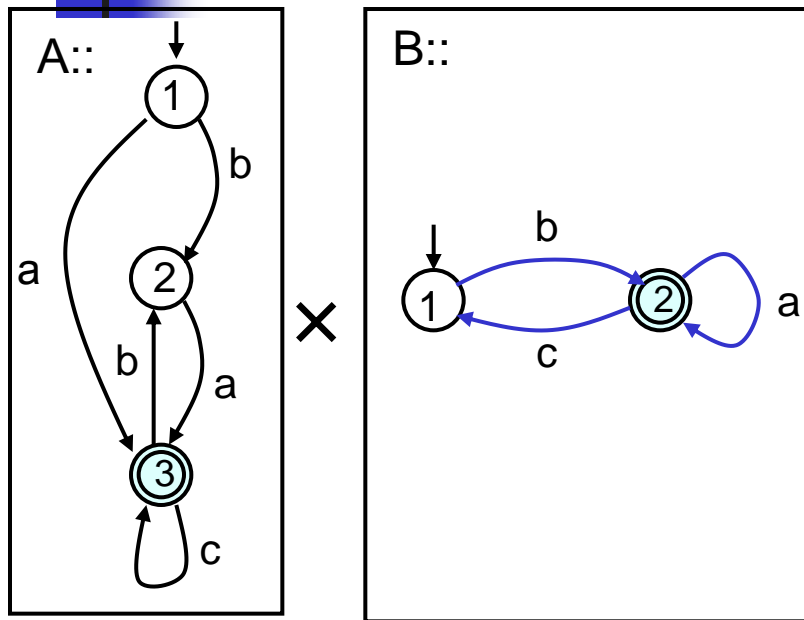


$A \times B$

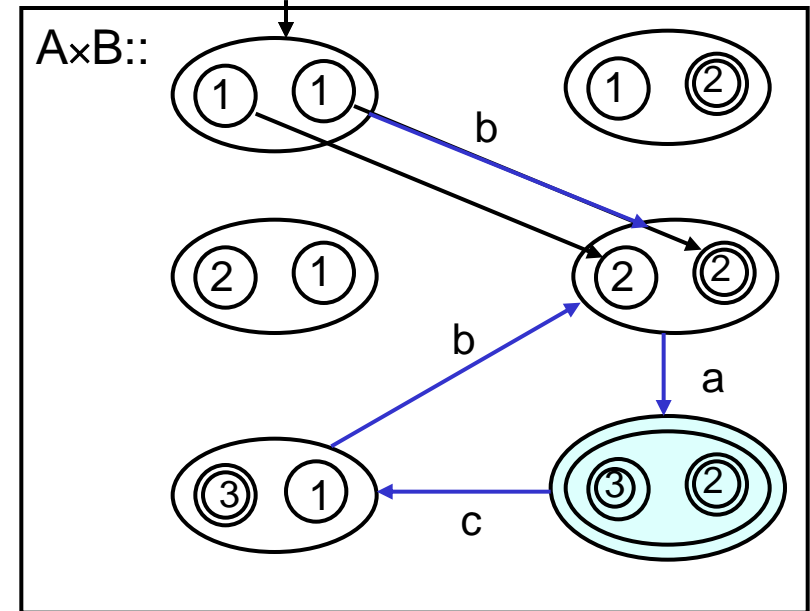


Язык, являющийся пересечением автоматных языков, можно определить, построив синхронную композицию конечных автоматов

Как найти пересечение автоматных языков?



=

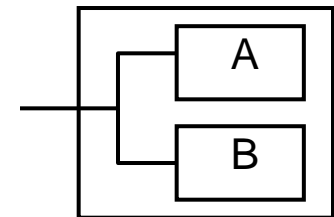


$A \times B$

$L_A = \{ba, acc, bacba, \dots\}$

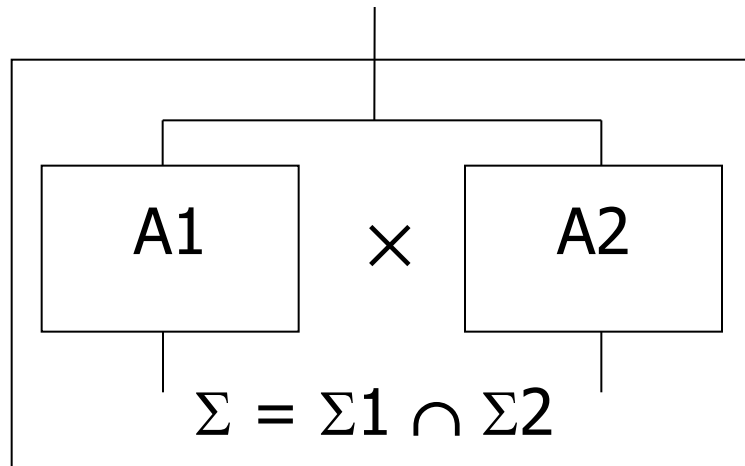
$L_B = \{b, ba, bacb, bacba, \dots\}$

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

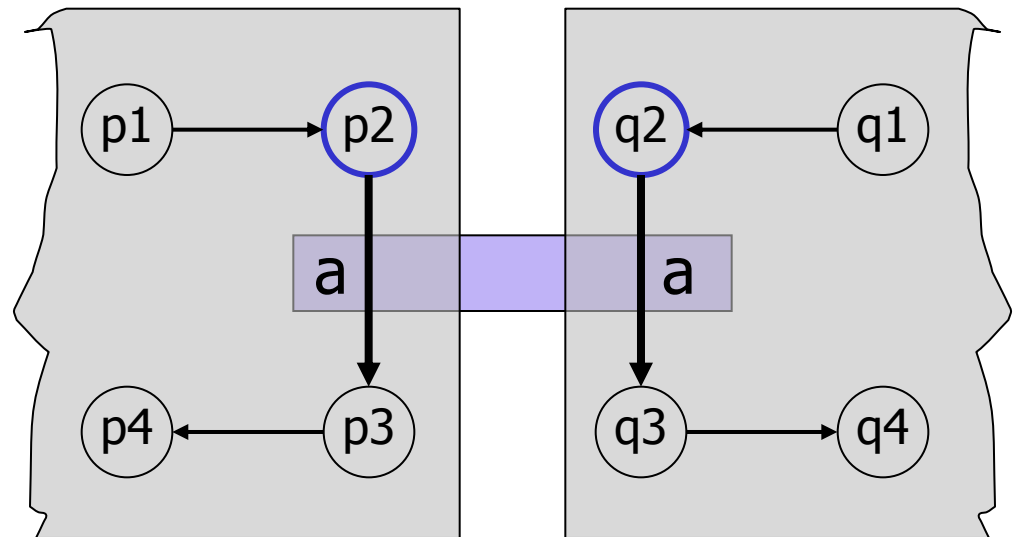


- Синхронная композиция автоматов – это два автомата, “стоящие рядом”, т.е. синхронно переходящие из одного состояния в другое под воздействием одинаковых входных сигналов
- Язык, допускаемый $A \times B$ – это пересечение языков, допускаемых A и B

Синхронное взаимодействие (рандеву)



$$L(A1 \times A2) = L(A1) \cap L(A2)$$



Общие действия выполняются по рандеву

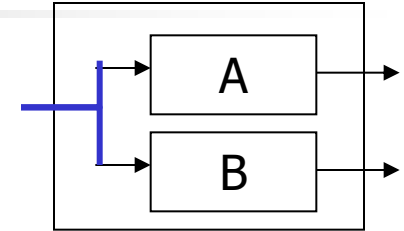
- Синхронное произведение двух автоматов дает пересечение языков, допускаемых этими автоматами
- Если синхронную коммуникацию по совпадающим событиям реализовать в железе, то для реализации автомата, допускающего пересечение языков, можно использовать формальную абстрактную модель синхронизации - синхронную композицию. Это сделано в новой теории "синтез супервизоров"

Формальное определение синхронной композиции

$$A = (S_A, \Sigma, s_{0A}, \delta_A, F_A)$$

$$B = (S_B, \Sigma, s_{0B}, \delta_B, F_B)$$

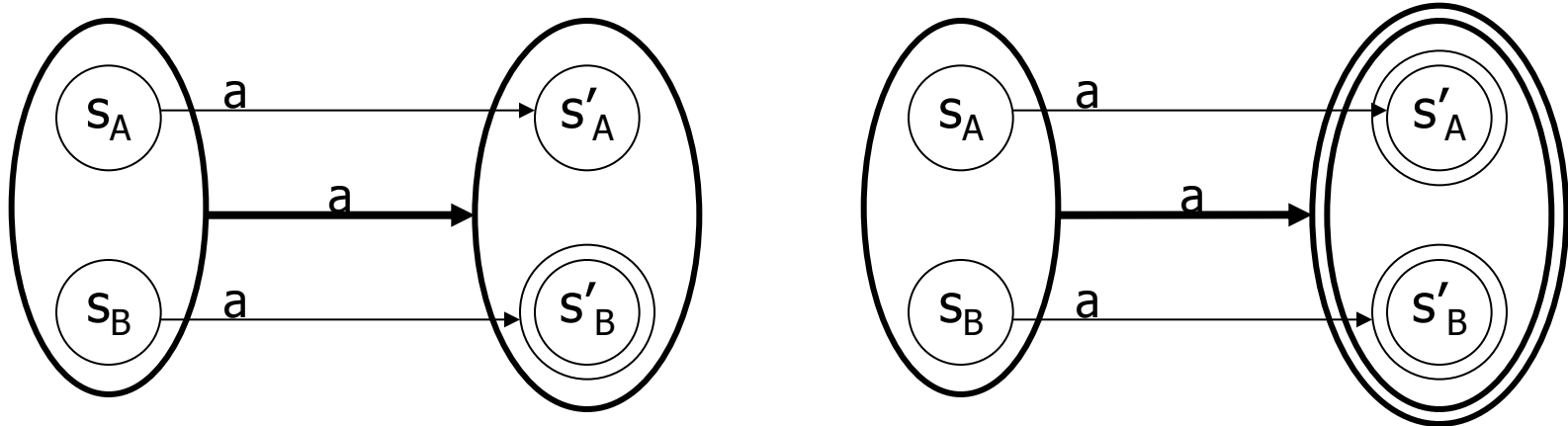
$$A \times B = (S_A \times S_B, \Sigma, (s_{0A}, s_{0B}), \delta_{AB}, F_A \times F_B)$$



$(\forall s_A \in S_A) (\forall s_B \in S_B) (\forall a \in \Sigma) \delta_{AB}((s_A, s_B), a) = (\delta_A(s_A, a), \delta_B(s_B, a))$ если оба перехода определены

$M \times N$ = множество всех пар элементов $\{ (m, n) \mid m \in M, n \in N \}$

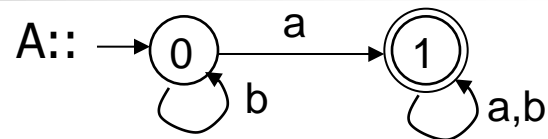
Операция взаимодействия “рандеву”, синхронная коммуникация: два автомата выполняют переход одновременно под воздействием одного сигнала



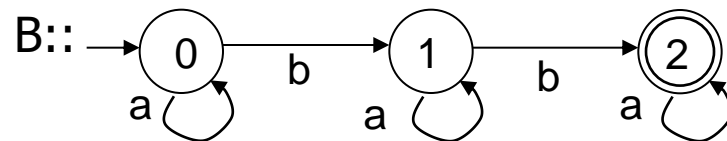
Состояние (s_A, s_B) – финальное, если оба, и s_A и s_B - финальные

Распознавание языков с помощью синхронной композиции автоматов

- Построить КА, распознающий язык над $\Sigma=\{a, b\}$, в цепочках которого **не менее одного символа a**.

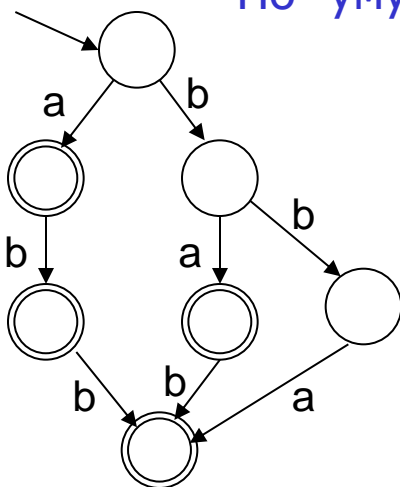


- Построить КА, распознающий язык над $\Sigma=\{a, b\}$, в цепочках которого **точно два символа b**



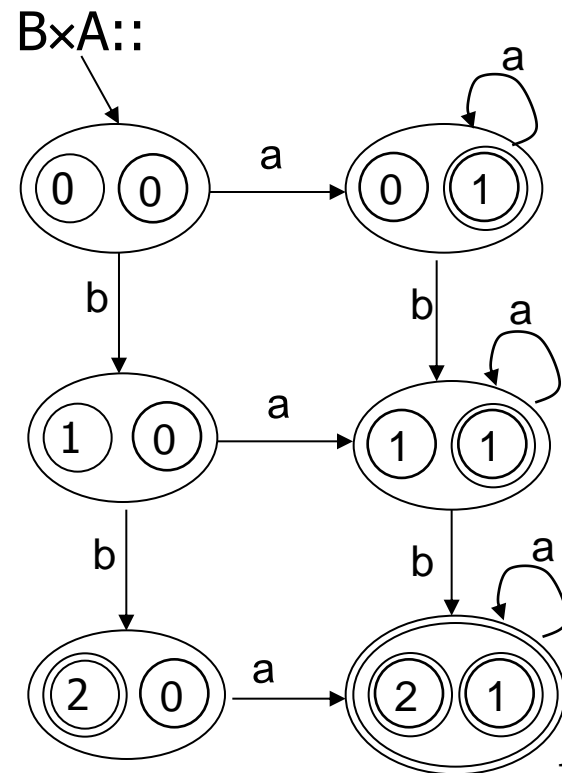
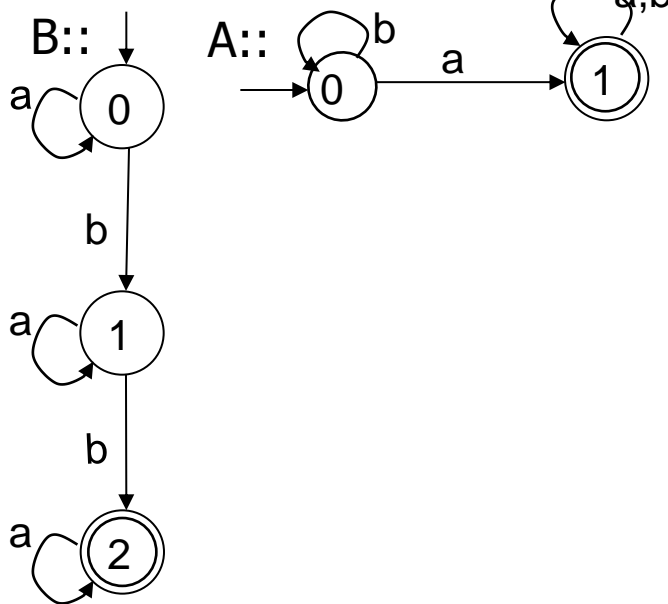
- Построить КА, распознающий язык над $\Sigma=\{a, b\}$, в цепочках которого **не менее одного символа a И точно два символа b**

Интуитивно:



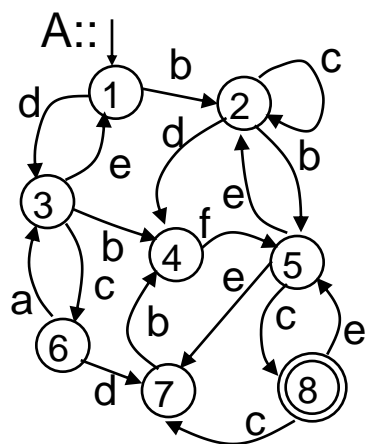
Неверно!!

По "уму":



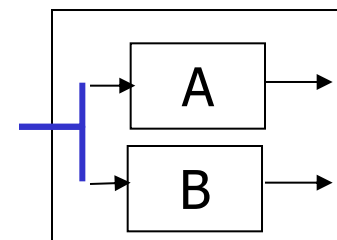
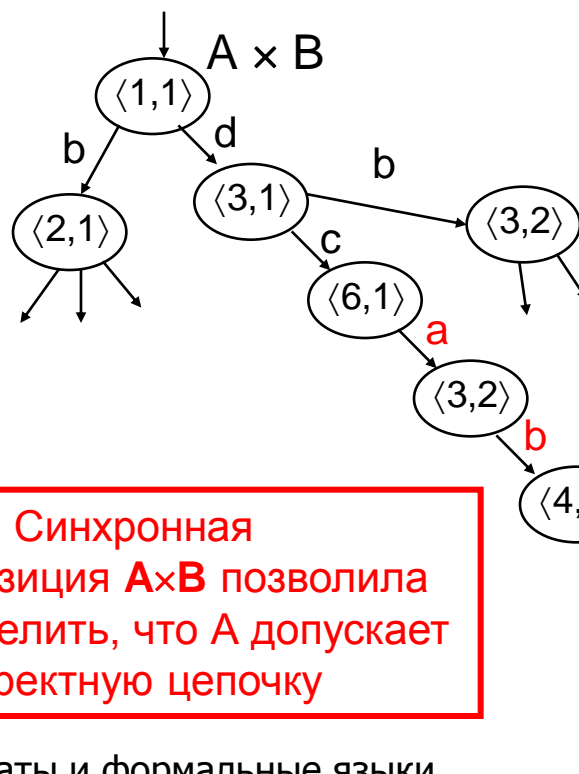
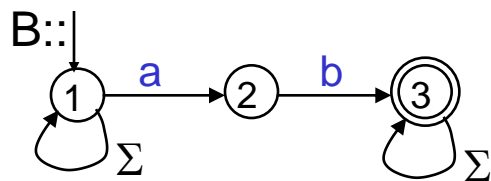
Верификация: проверка пересечения автоматных языков

Как доказать, что среди **бесконечного** множества всех возможных цепочек, допускаемых автоматом A, нет таких, которые имеют ab в середине (т.е. это **НЕПРАВИЛЬНЫЕ** цепочки).



- Нужно проверить, пересекаются ли L_A и L_B , т.е. существует ли хотя бы одна входная цепочка, которая допускается и A, и B?
- По автоматам A и B построим их синхронную композицию.
- **Теорема.** Языки L_A и L_B пересекаются тогда и только тогда, когда в синхронной композиции $A \times B$ достижимо финальное состояние

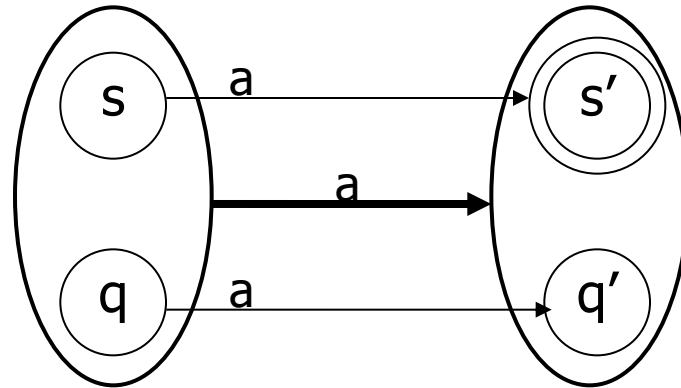
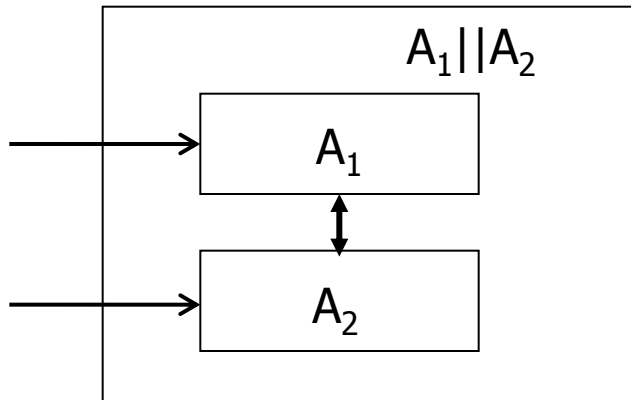
Построим автомат B, который допускает **все возможные НЕПРАВИЛЬНЫЕ** цепочки. B определяет все цепочки, включающие ab как подцепочку



ЕСТЬ! Синхронная композиция $A \times B$ позволила определить, что A допускает некорректную цепочку

Параллельная (асинхронная) композиция (интерливинг независимых действий автоматов)

- Пусть $A_1 = (S, \Sigma_1, \delta_1, s_0, F_1)$ и $A_2 = (Q, \Sigma_2, \delta_2, q_0, F_2)$ – два автомата

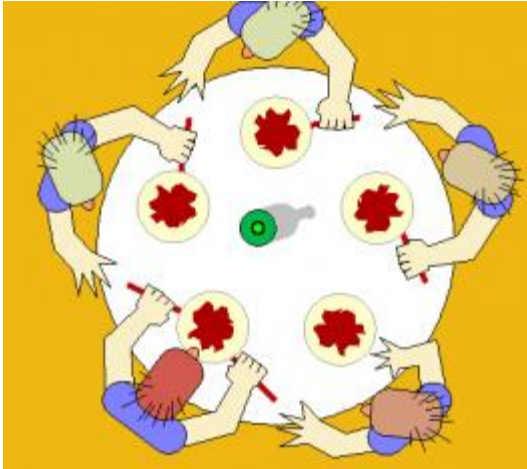


- Определение :** Параллельной композицией A_1 и A_2 ($A_1 || A_2$) называется автомат $A_1 || A_2 = (S \times Q, \Sigma_1 \cup \Sigma_2, \delta, (s_0, q_0), F_1 \times F_2)$, где:

общие	$\delta((s, q), a) = (s', q')$ если $a \in \Sigma_1 \cap \Sigma_2$ и $\delta_1(s, a) = s', \delta_2(q, a) = q'$;
локальные A_1	$\delta((s, q), a) = (s', q)$ если $a \in \Sigma_1 - \Sigma_2$ и $\delta_1(s, a) = s'$;
локальные A_2	$\delta((s, q), a) = (s, q')$ если $a \in \Sigma_2 - \Sigma_1$ и $\delta_2(q, a) = q'$;

В параллельной (асинхронной) композиции автоматов общие действия выполняются одновременно, локальные действия выполняются независимо

Пример параллельной композиции: анализ системы || процессов (обедающие философы)



- Классическая задача параллельного программирования, предложенная Э.Дейкстрой как метафора для прояснения проблем синхронизации параллельных процессов при использовании ими общих ресурсов
- Пять философов сидят за круглым столом, каждый думает. Перед каждым из них – тарелка со спагетти. Между каждой парой философов лежит вилка (общий ресурс) .
Спагетти можно есть только двумя вилками!
- Когда философ проголодается, он берет последовательно две вилки (если они свободны): сначала левую вилку, потом правую вилку, ест, кладет обе вилки и опять думает
- Правила поведения за столом едины для всех, все философы действуют по одному алгоритму.
- Проблема: существует дедлок (блокировка)

Блокировка в системе процессов, последовательно захватывающих ресурсы

Философ 0



Вилка 0



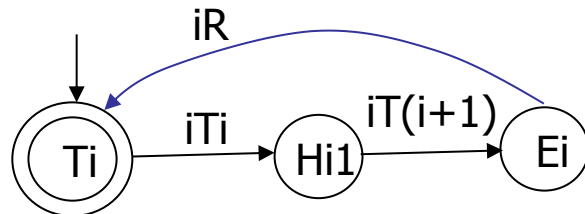
Вилка 1



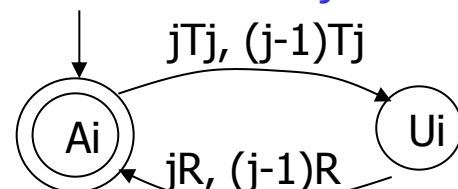
Философ 1



Философ i

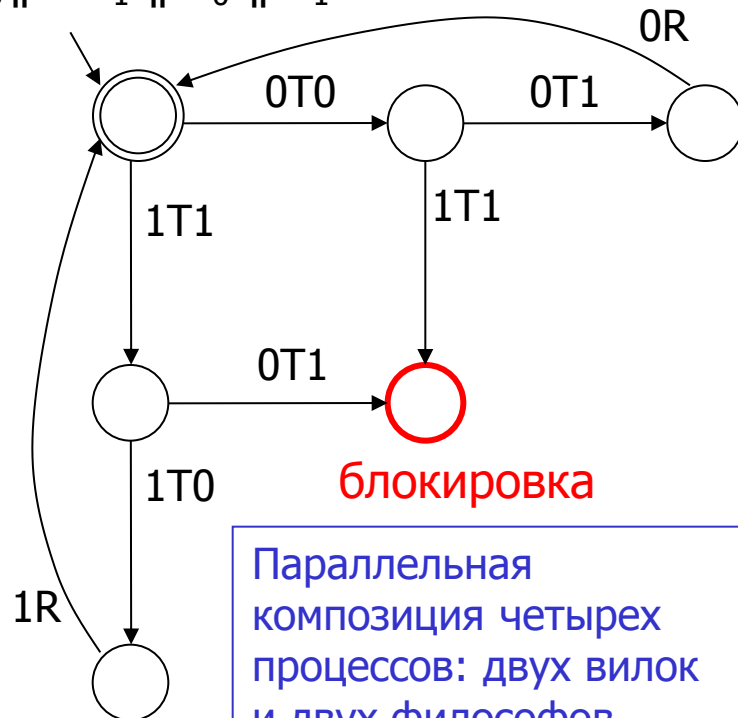


Вилка j



$Phil_0 \parallel Phil_1 \parallel F_0 \parallel F_1 ::$

$i \ T \ k$ – философ i
берет вилку k
 $j \ R$ – философ j
возвращает
обе вилки



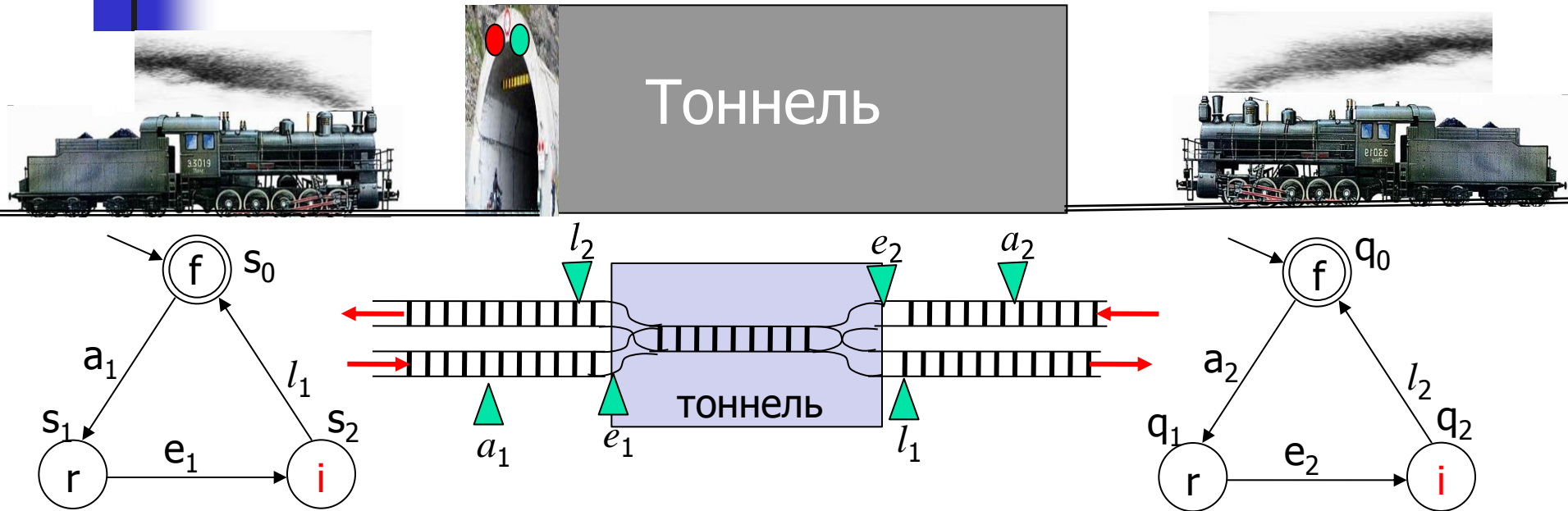
блокировка

Параллельная
композиция четырех
процессов: двух вилок
и двух философов

Анализ проблемы с пятью философами
требует построения \parallel композиции 10
процессов: 5 процессов-философов и
5 процессов-вилок.

Общее число состояний $\sim 3^5 \times 2^5 \sim 8000$

Управление входом в критический интервал. Как описать формально?



Состояния:

- f – поезд едет вне тоннеля (free)
- r – поезд - перед тоннелем (ready)
- i – поезд едет в тоннеле (inside)

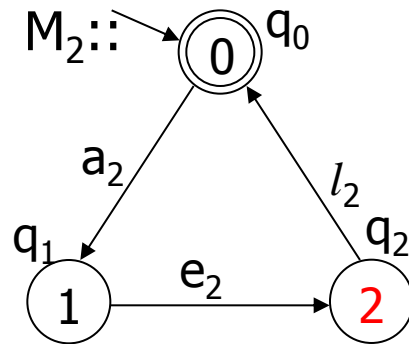
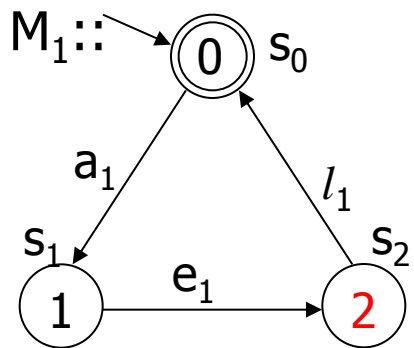
События:

- a – arrive – прибытие к тоннелю
- e – enter – вход в тоннель
- l – leave – выход из тоннеля

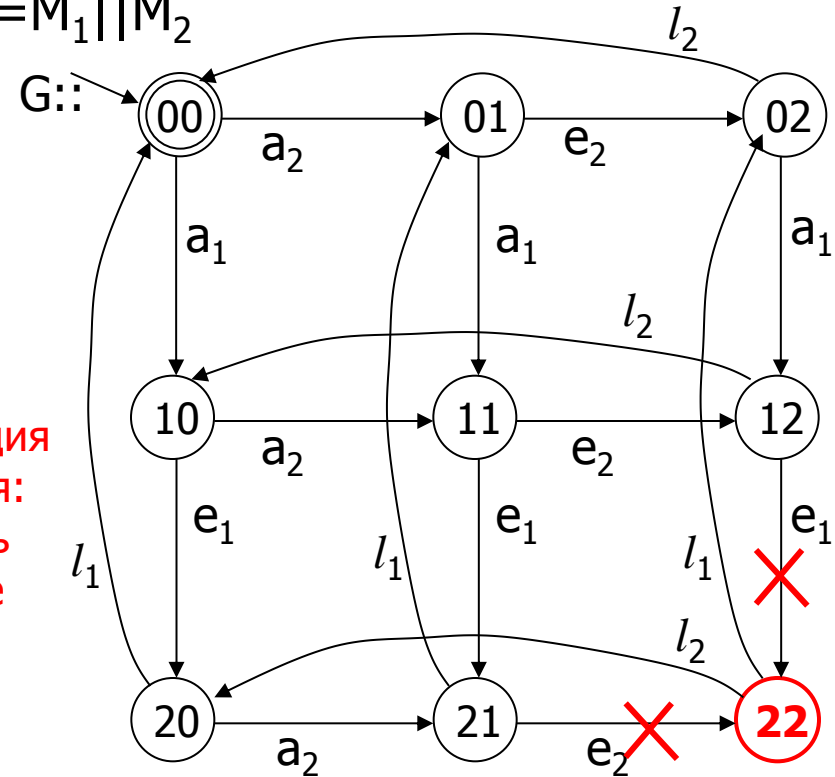
В тоннеле – однопутка, критический интервал, ресурс, который может использоваться только одним поездом

Управление входом в тоннель – запрет выполнения действий в системе отдельными процессами до наступления определенных событий (например, пока встречный поезд не выйдет из тоннеля)

События параллельно функционирующих процессов описываются их параллельной композицией



$$G = M_1 || M_2$$



У процессов НЕТ общих действий, все действия выполняются независимо

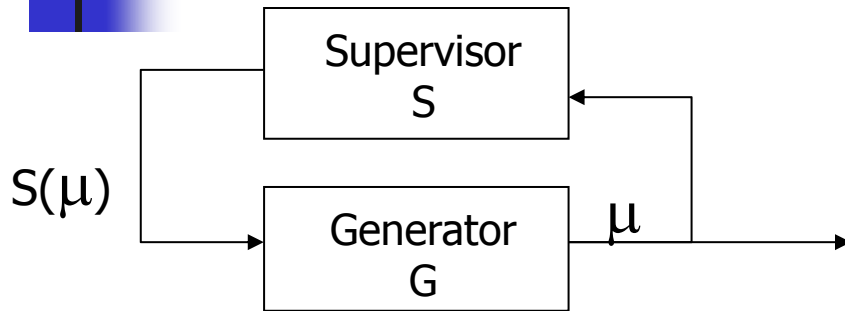
События:

а – arrive – прибытие к тоннелю
е – enter – вход в тоннель
l – leave – выход из тоннеля

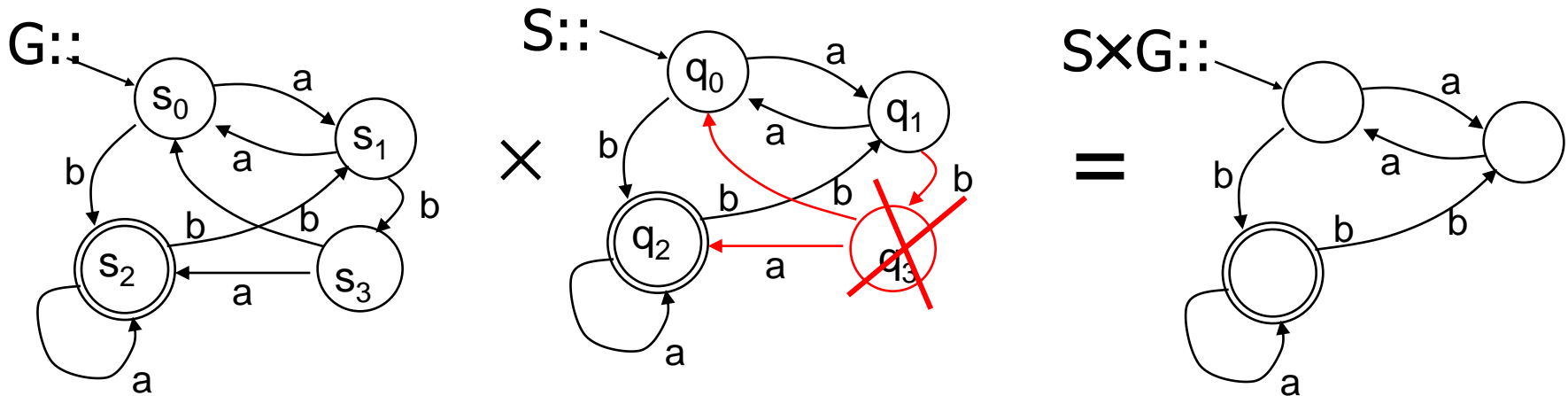
Спецификация требования:
исключить состояние $\langle 2, 2 \rangle$

некоординируемое поведение двух процессов приводит к аварии

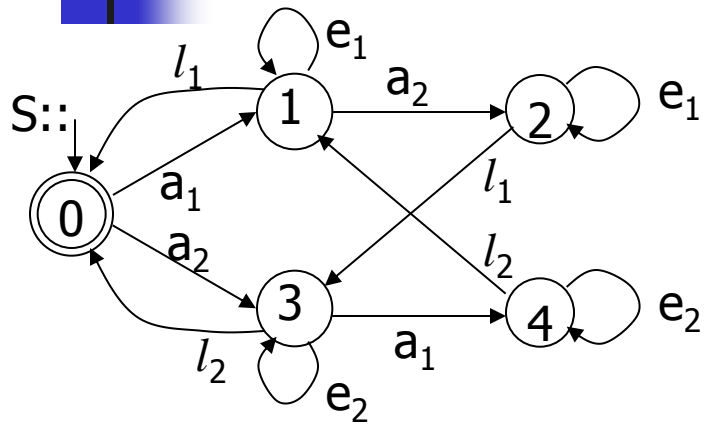
Использование синхронной композиции процессов для ограничения языка



- Систему рассматриваем, как генератор последовательностей событий, т.е. языков. Для того, чтобы запретить выполнение некоторых событий в конкретном состоянии генератора нужно просто не иметь этого события в супервизоре, работающем синхронно с генератором



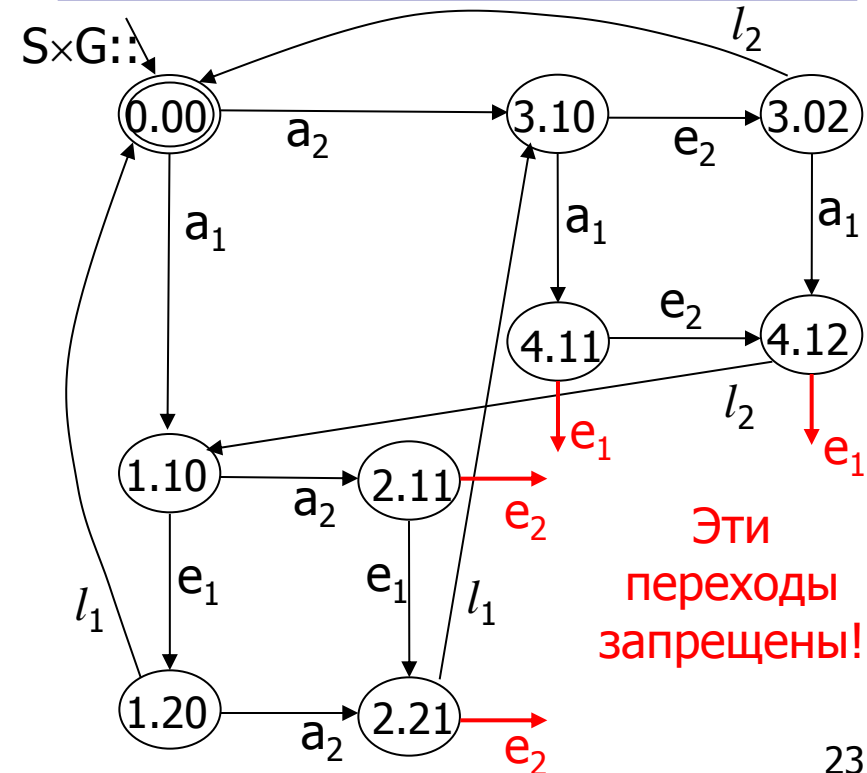
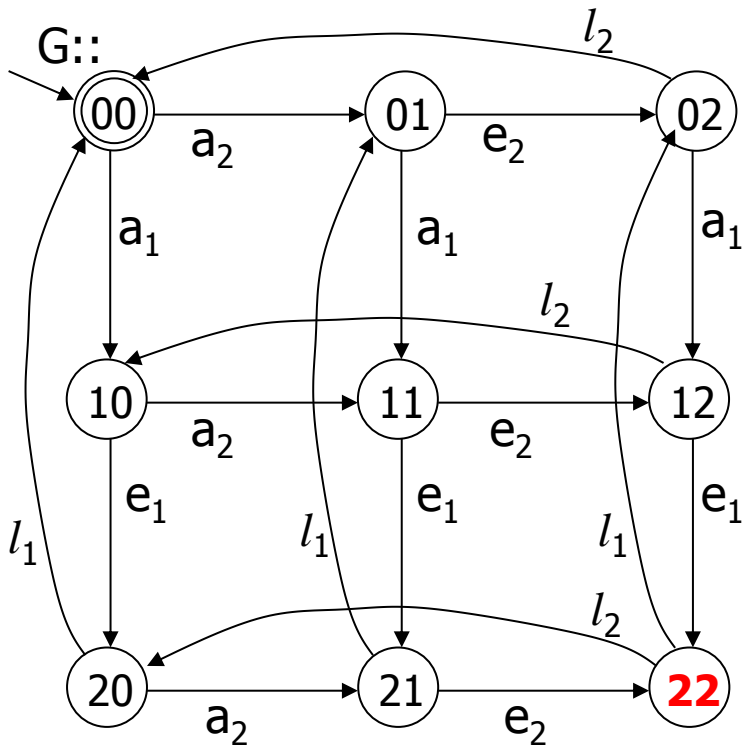
Синхронная композиция и ограничение языка. Управление супервизором: по справедливости



События:

а – arrive – прибытие к тоннелю
е – enter – вход в тоннель
l – leave – выход из тоннеля

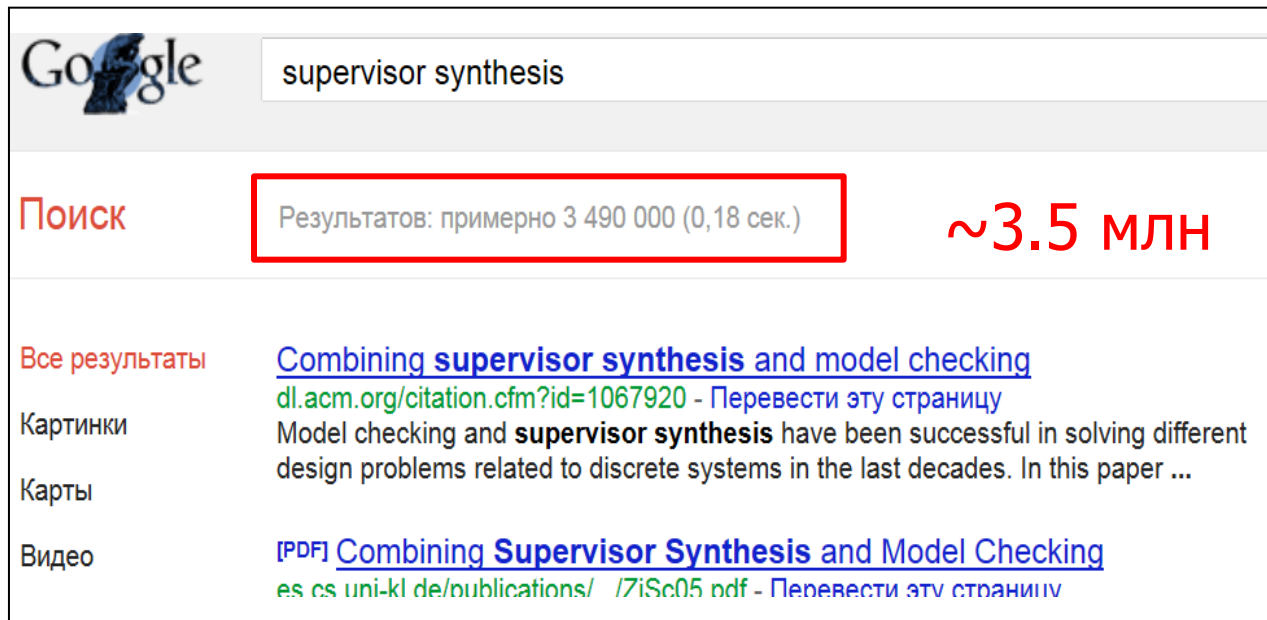
В качестве супервизора выберем S ,
для управления поездами реализуем
синхронную композицию $S \times G$



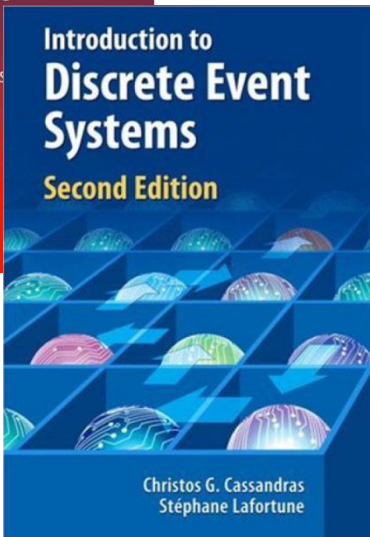
Эти
переходы
запрещены!

Supervisor synthesis - “горячая” область исследований

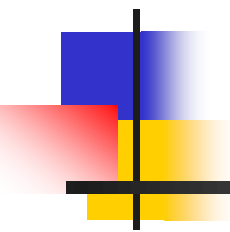
Монографии (недавние), миллионы публикаций,
десятки ежегодных конференций



Google search results for "supervisor synthesis". The search bar shows the query "supervisor synthesis". The results section shows "Поиск" (Search) and "Результатов: примерно 3 490 000 (0,18 сек.)" (Results: approximately 3 490 000 (0.18 sec.)). To the right, it says "~3.5 млн". Below the search bar, there are links to "Все результаты" (All results), "Картинки" (Images), "Карты" (Maps), and "Видео" (Videos). The first result is "Combining supervisor synthesis and model checking" with a link to dl.acm.org/citation.cfm?id=1067920 and a note "Перевести эту страницу" (Translate this page). The second result is "[PDF] Combining Supervisor Synthesis and Model Checking" with a link to es.cs.uni-kl.de/publications/171Sc05.pdf and a note "Перевести эту страницу" (Translate this page).



- 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
- ...



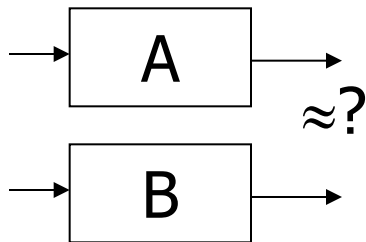
Эквивалентность двух автоматов- распознавателей

Эквивалентность двух конечных автоматов-распознавателей – как проверить?

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

$$A = (S_A, \Sigma_A, s_{0A}, \delta_A, F_A)$$

$$B = (S_B, \Sigma_B, s_{0B}, \delta_B, F_B)$$

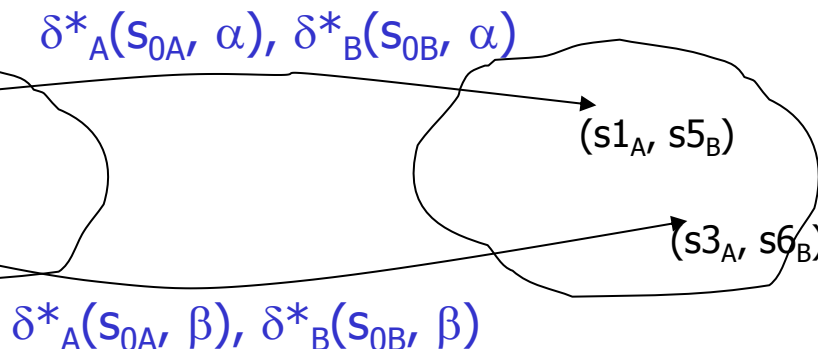


1. **Требование:** Одинаковые входные алфавиты, $\Sigma_A = \Sigma_B$
2. **Вопрос:** Совпадают ли множества входных цепочек, которые переводят автоматы из начального состояния в допускающее состояние?

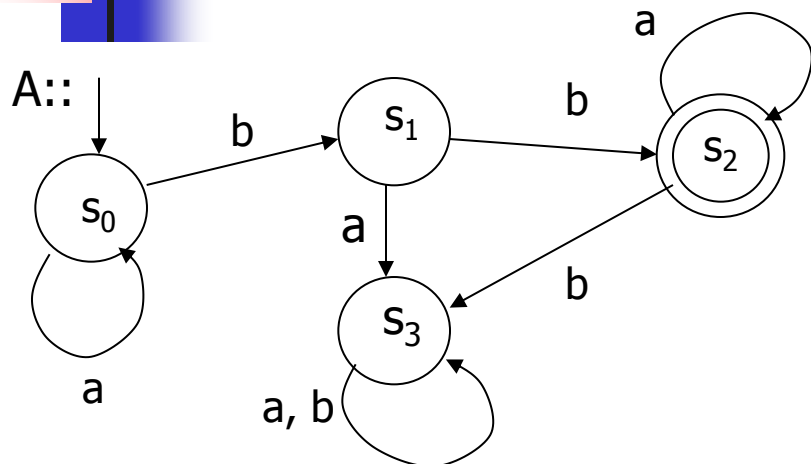
$$\Sigma = \{a, b, c\}; \quad \Sigma^* = \{\varepsilon, a, b, c, aa, ab, ac, cc, bb, cba, cbbba, \dots\}$$

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

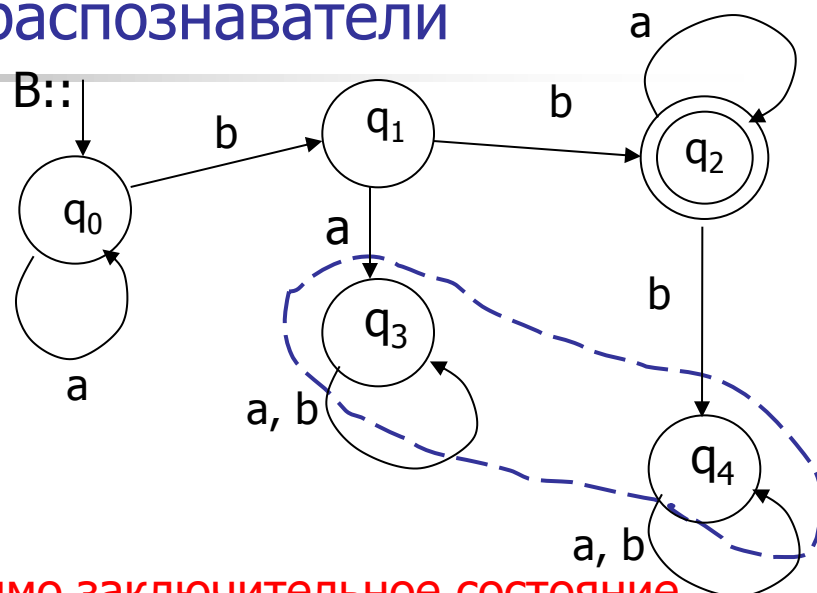
Эквивалентные автоматы под воздействием одинаковых входных цепочек переходят оба в допускающие или оба в недопускающие состояния



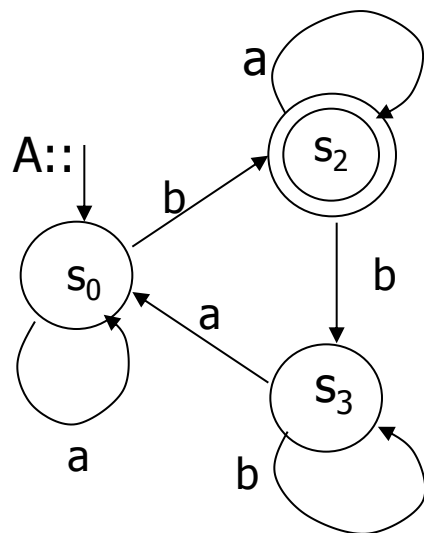
Эквивалентные автоматы-распознаватели



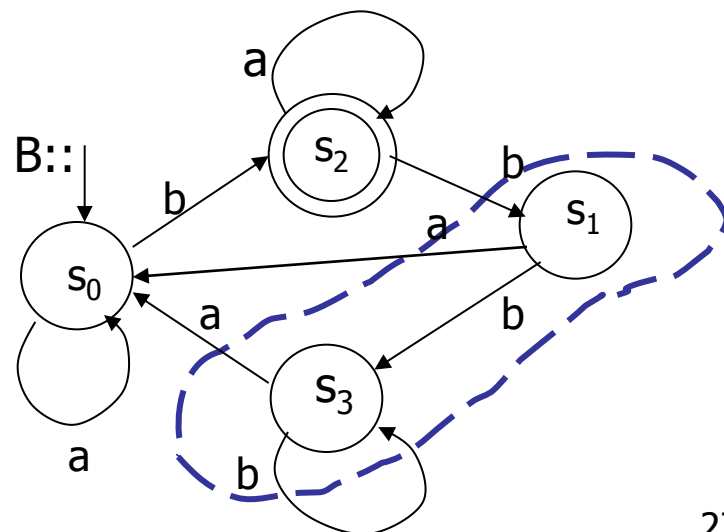
\approx



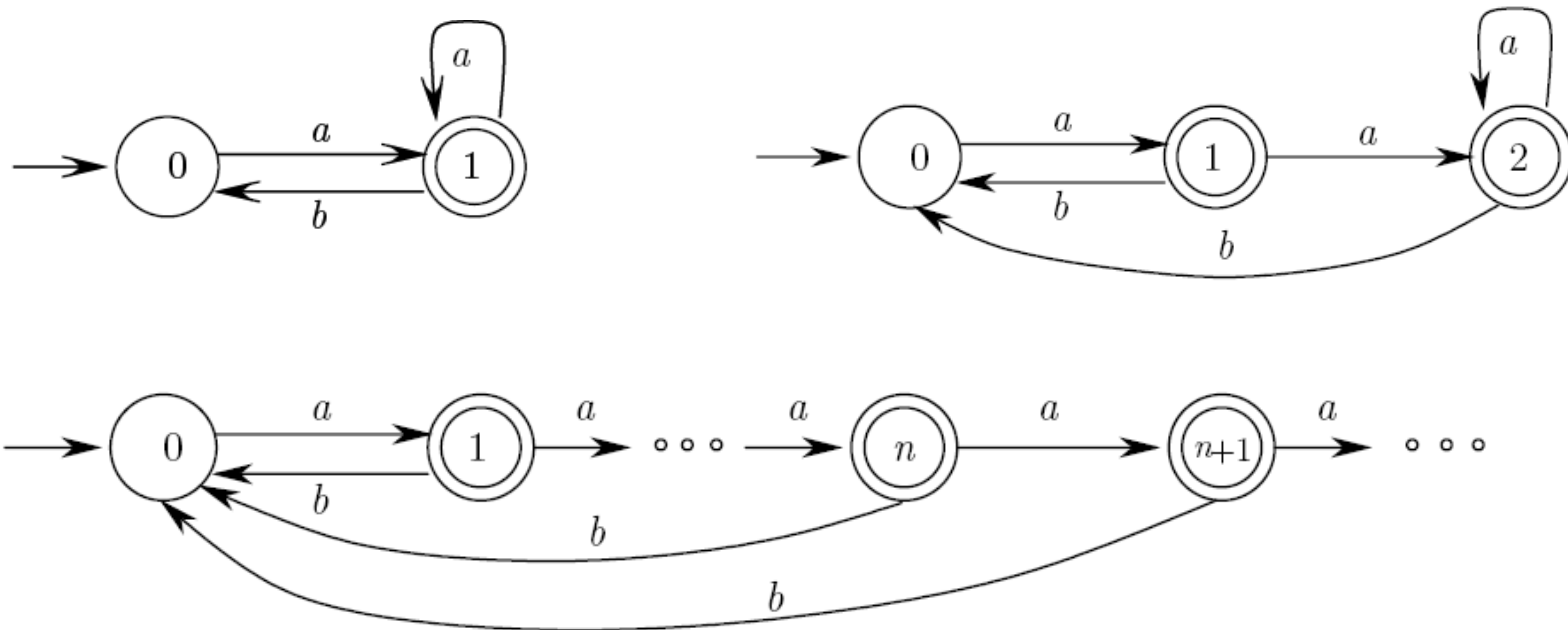
Все состояния, из которых не достижимо заключительное состояние,
эквивалентны



\approx



Эквивалентные автоматы-распознаватели

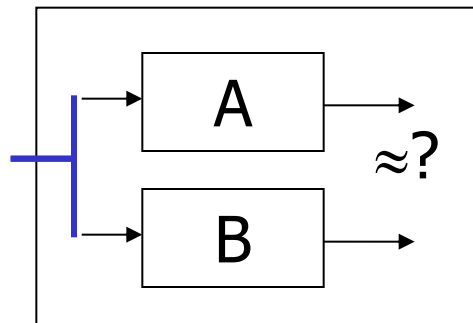


- Все три автомата-распознавателя эквивалентны
- Они распознают один и тот же язык
- Третий пример показывает, что автомат с бесконечным числом состояний может быть эквивалентным конечному автомату

Эквивалентные автоматы-распознаватели.

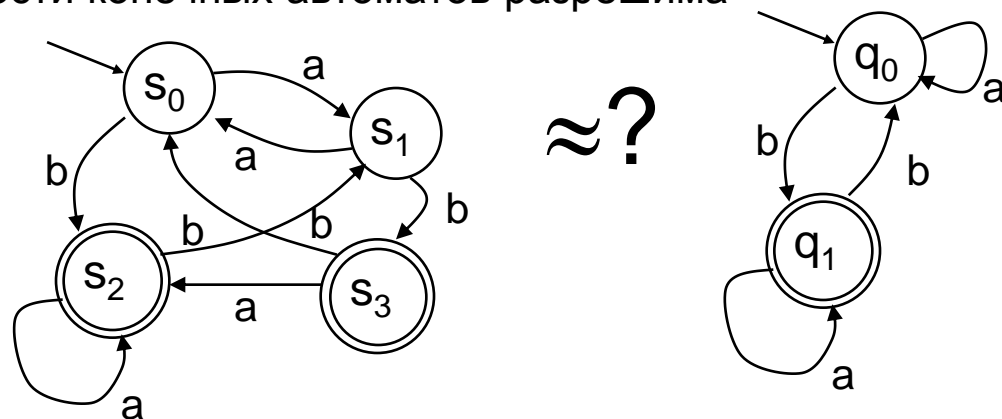
Теорема Мура

Теорема Мура. Проблема эквивалентности конечных автоматов разрешима



$$s \xrightarrow{a} s' \in \delta_A; \quad q \xrightarrow{a} q' \in \delta_B$$

$$(s, q) \xrightarrow{a} (s', q') \in \delta_{A \times B}$$

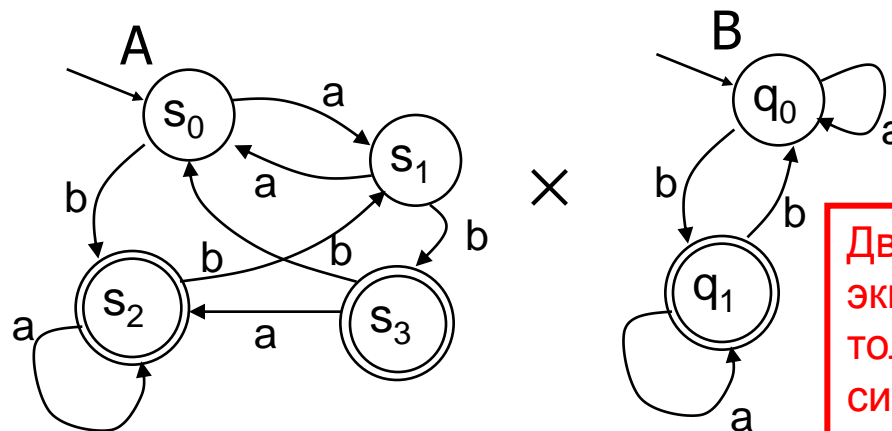
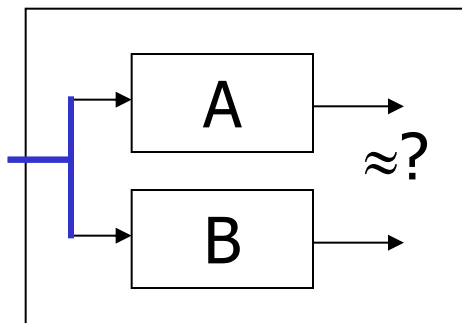


**Правило перехода синхронной
композиции автоматов**

Доказательство. Построим синхронную композицию автоматов. Синхронная композиция автоматов определяет пересечение языков. Синхронная композиция конечных автоматов имеет конечное число состояний и ее можно анализировать

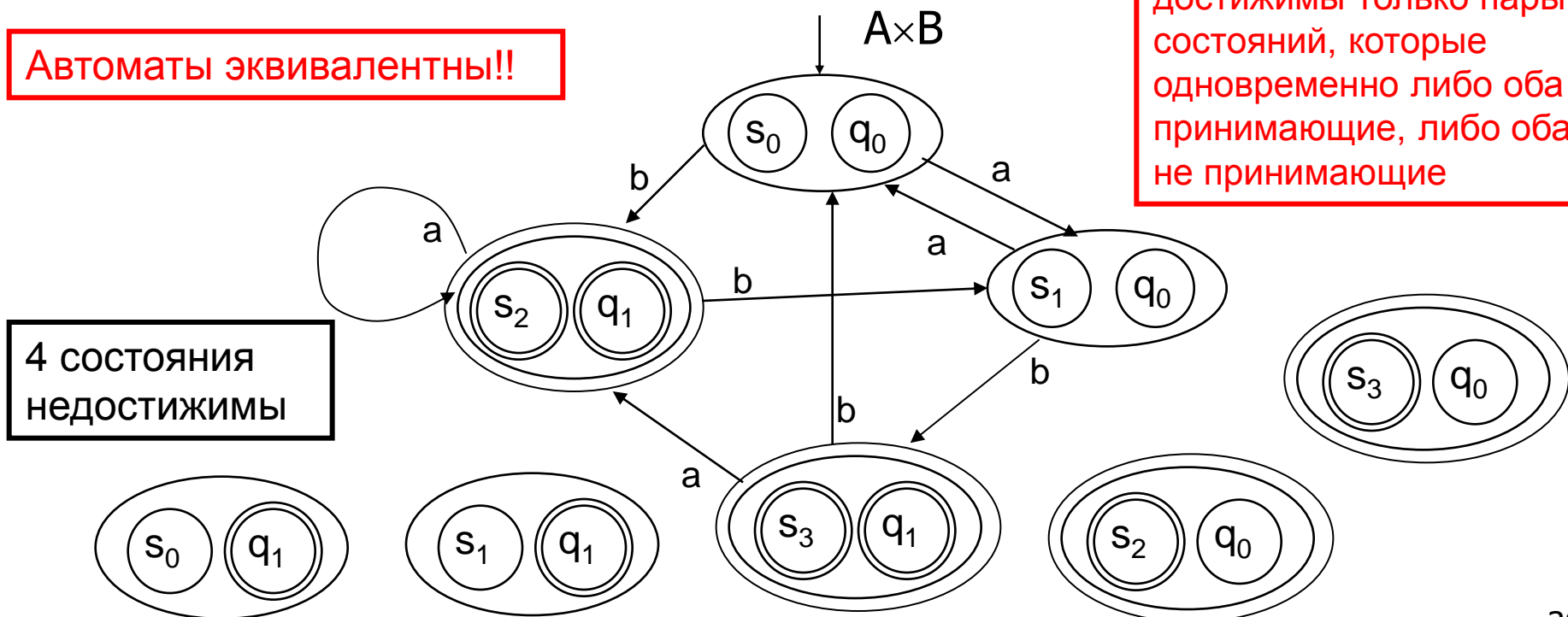
Теорема. Два автомата будут эквивалентными тогда и только тогда, когда в их синхронной композиции достижимы только такие пары состояний, которые одновременно либо оба принимающие, либо оба не принимающие

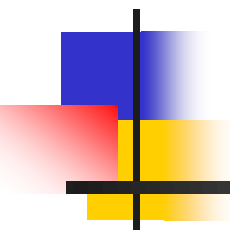
Эквивалентные автоматы-распознаватели



Автоматы эквивалентны!!

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

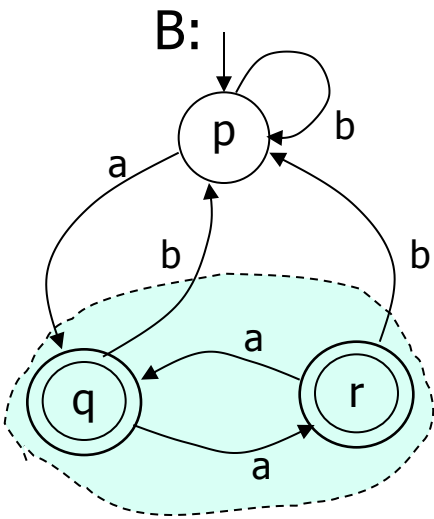




Минимизация конечных автоматов- распознавателей

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 q и с r , невозможно различить: *любая подаваемая на вход автоматов входная цепочка либо обоими автоматами допускается, либо обоими автоматами не допускается*

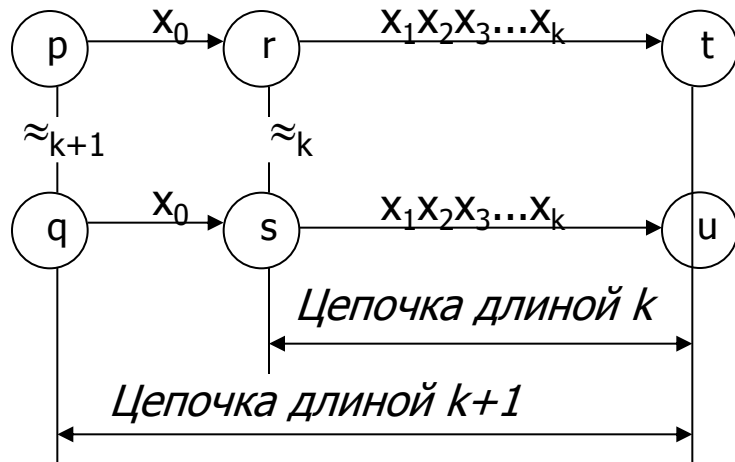
Формальное определение эквивалентных состояний:

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

Мы не можем использовать это определение практически: мы не можем перебрать все возможные входные цепочки для каждой пары состояний: таких цепочек бесконечное число

Минимизация конечных автоматов - распознавателей

- Построим на множестве состояний автомата A разбиения $\pi_0, \pi_1, \dots, \pi_\infty$, такие, что в один **класс разбиения** π_k попадают состояния, из которых входные цепочки **длиной** k одновременно допускаются или одновременно не допускаются
- Такие состояния будем считать k -эквивалентными (в отношении \approx_k)
$$p \approx_k q \Leftrightarrow (\forall \alpha \in \Sigma^*: |\alpha| \leq k) [\delta^*(p, \alpha) \in F \Leftrightarrow \delta^*(q, \alpha) \in F]$$
- При $k=0$: любые два допускающие состояния и любые два не допускающие состояния 0-эквивалентны
- Пусть $k \geq 0$. Очевидно, что $p \approx_{k+1} q \Leftrightarrow (\forall x \in \Sigma) \delta(p, x) \approx_k \delta(q, x)$

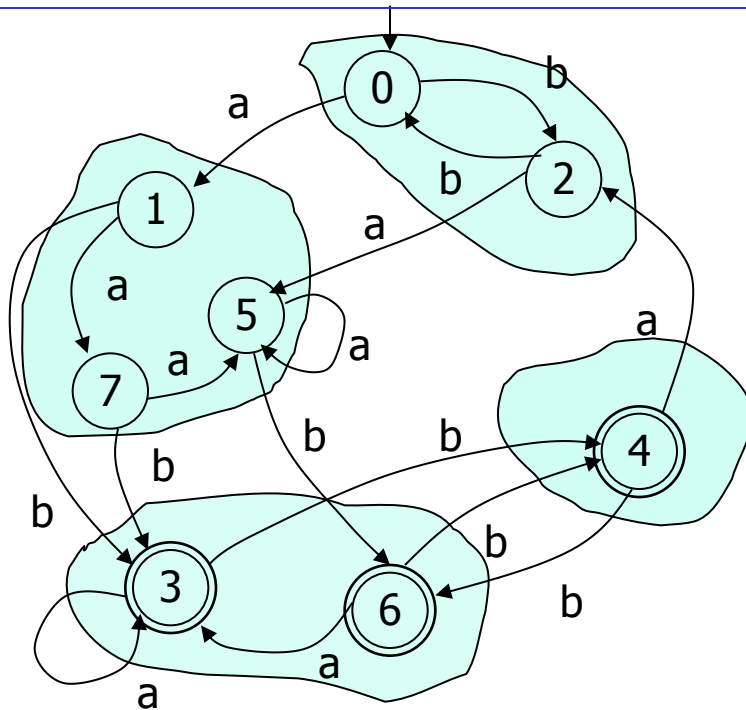


Два состояния, p и q , $k+1$ -эквивалентны, если и только если для любого $x \in \Sigma$, два состояния $\delta(p, x)$ и $\delta(q, x)$ k -эквивалентны

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

Конечный автомат с эквивалентными состояниями

Если R_k – класс эквивалентных состояний по эквивалентности π_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_{π_0}		δ_{π_1}	
	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

- $\pi_0 = \{0, 1, 2, 5, 7\} = A$; $\{3, 4, 6\} = B$
- $\pi_1 = \{0, 2\} = C$; $\{1, 5, 7\} = D$; $\{3, 6\} = E$; $\{4\} = F$
- $\pi_2 = \{0, 2\}$; $\{1, 5, 7\}$; $\{3, 6\}$; $\{4\} = \pi_1 = \pi_\infty$

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 допускающих, либо в пару недопускающих состояний
- Очевидно, что π_0 состоит всегда из двух классов: все недопускающие, и все допускающие состояния
- 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 $\delta(s,x)$ и $\delta(q,x)$ будут в одном и том же классе предыдущего разбиения π_k .

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

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

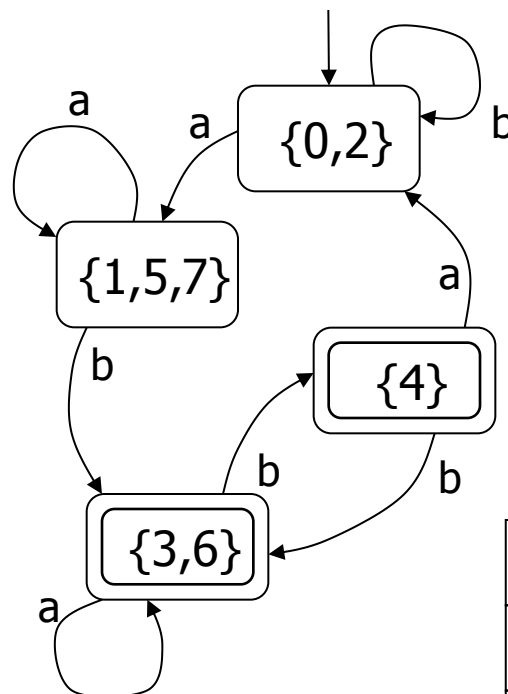
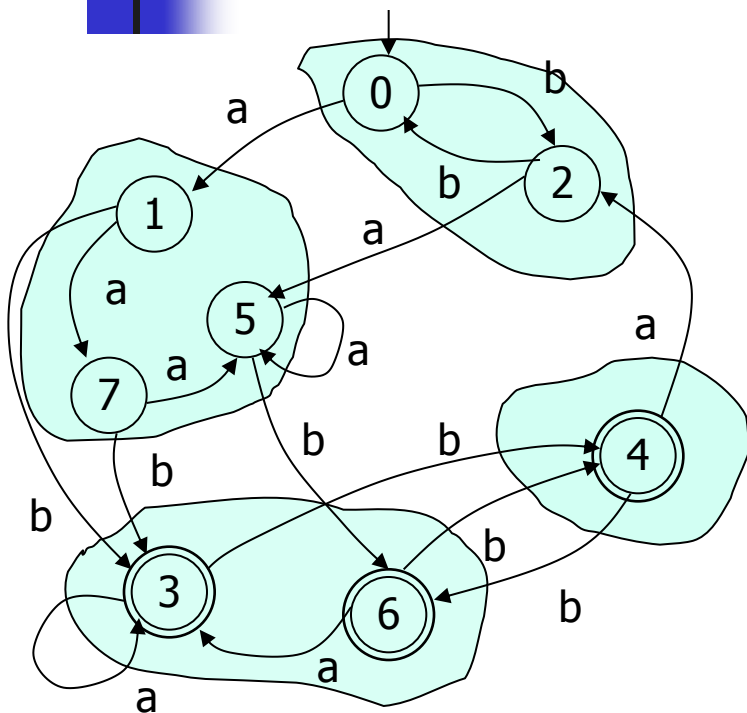
$$\pi_2 = \pi_1$$

	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+ -
допускающие
состояния

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

Минимальный конечный автомат-распознаватель



- $\pi_0 = \{0, 1, 2, 5, 7\} = A; \{3, 4, 6\} = B$
- $\pi_1 = \{0, 2\} = C; \{1, 5, 7\} = D; \{3, 6\} = E; \{4\} = F$
- $\pi_2 = \{0, 2\}; \{1, 5, 7\}; \{3, 6\}; \{4\} = \pi_1 = \pi_\infty$

δ_{π_∞}		
	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 π_∞ исходного автомата

Представление отношения эквивалентности матрицей

- Манипуляции с отношениями эквивалентности удобно выполнять с помощью матрицы инцидентий, в которой в клетке $\langle i, j \rangle$ 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				

Результирующая треугольная матрица для отношения эквивалентности $\pi = \{0, 1, 2\}; \{3, 4\}; \{5\}$

- Пусть $\pi = \{0, 1, 2\}; \{3, 4\}; \{5\}$
– всего 6 элементов: $\{0, \dots, 5\}$
- 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 $\langle i, i \rangle$ всегда принадлежит одному и тому же блоку
- Вывод:** в матрице можно выбросить один треугольник и диагональ

Эквивалентное представление алгоритма построения отношения эквивалентности на множестве состояний

- **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 $\langle p, q \rangle$, у которой не стоит N, проверяем, стоит ли N для пар $\langle \delta(p, x), \delta(q, x) \rangle$ хотя бы при одном x . Если стоит, то для пары $\langle p, q \rangle$ ставим N.
- Алгоритм повторяется, пока добавляется хотя бы одно N

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

	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

	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						

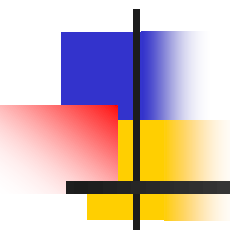
Начальная
расстановка

	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

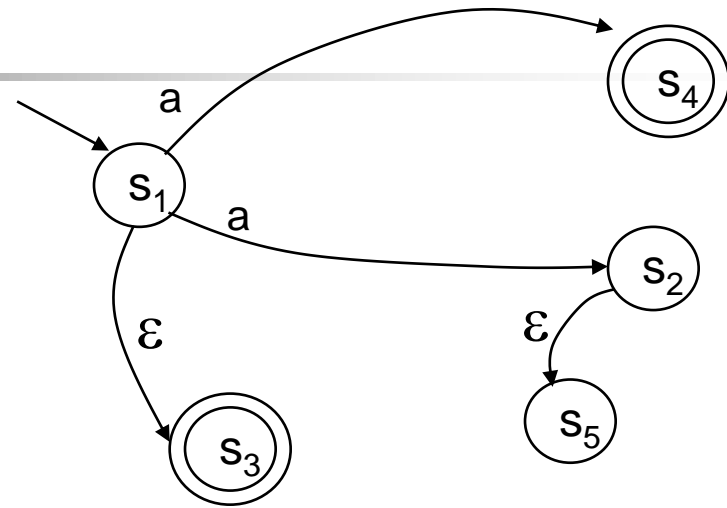
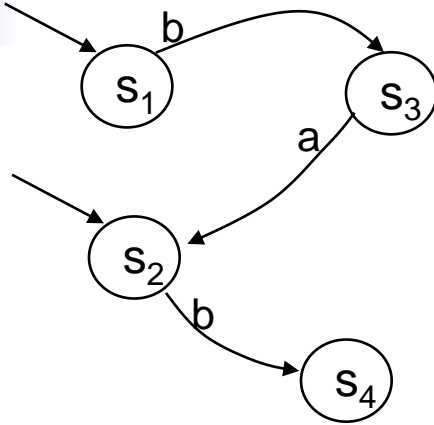
Эквивалентны:

0 и 2;
1, 5 и 7;
3 и 6;
4 само себе



Недетерминированные конечные автоматы-распознаватели

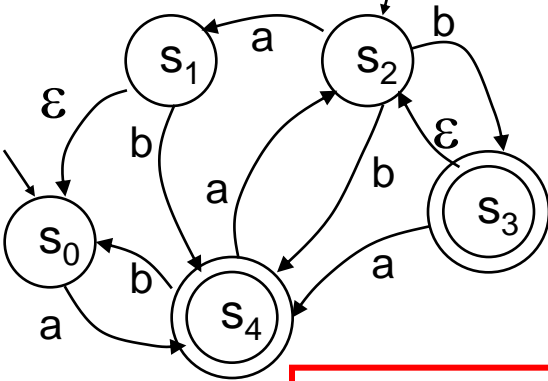
Недетерминированные конечные автоматы-распознаватели



- 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
 - 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

Недетерминированные конечные автоматы-распознаватели

- 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 ? Может быть РАЗНАЯ

$s_2s_1s_0s_4$ – допускается

$s_0s_4s_2$ – НЕ допускается

В разных применениях можно понимать по-разному

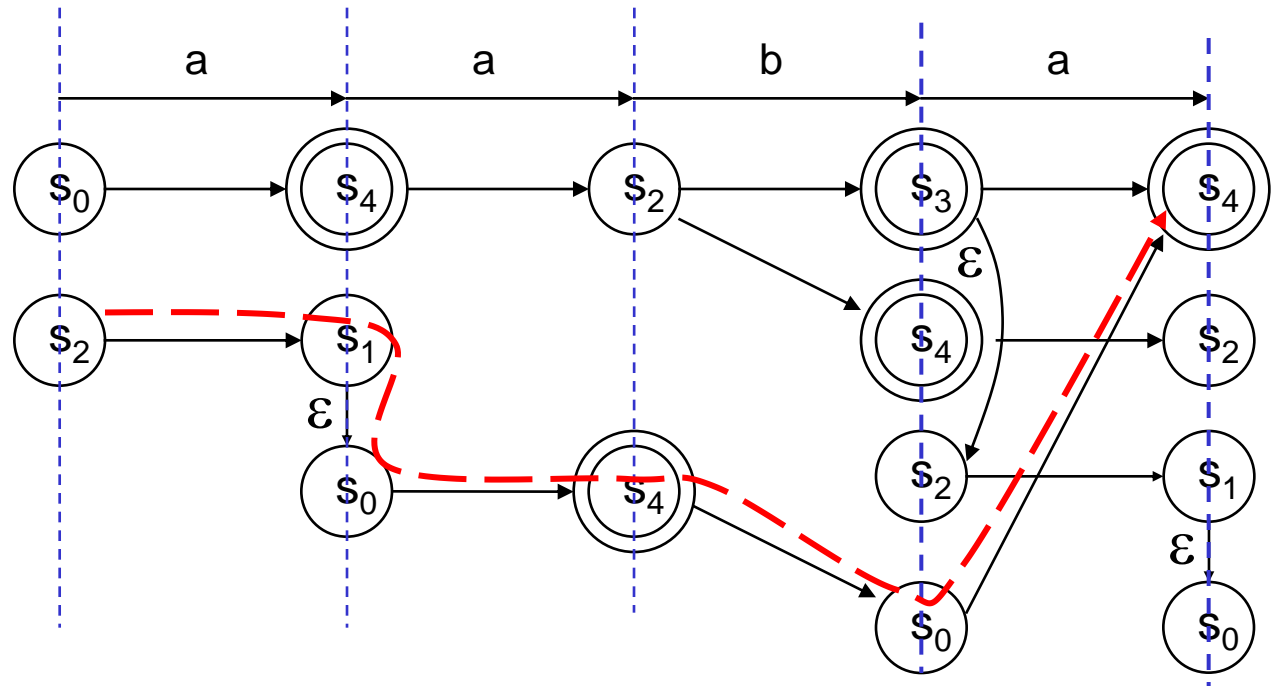
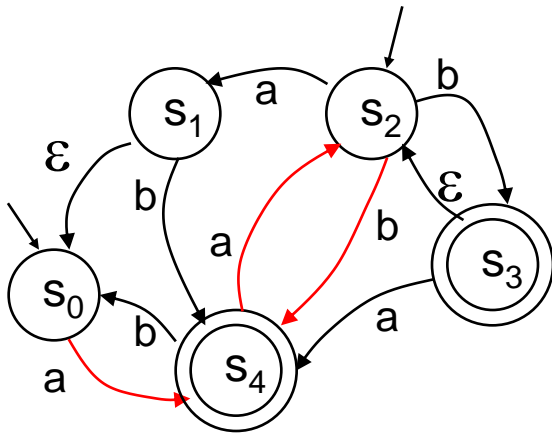
Основное правило в теории формальных языков:

Конечный автомат распознаватель – это не модель устройства, мы не реализуем его аппаратно. Это просто формальная модель, которая удобна для конечного задания языка

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

Недетерминированные конечные автоматы-распознаватели

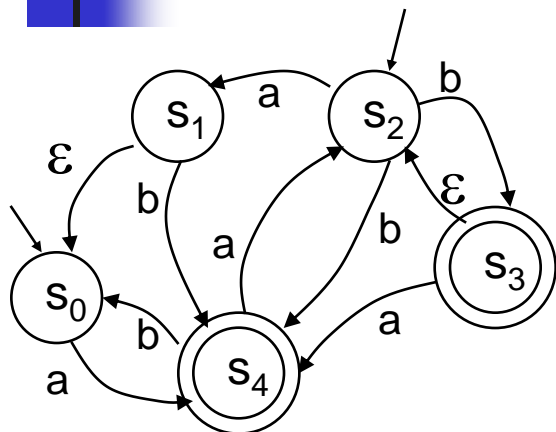
Каковы все возможные реакции на ааба?



Цепочка ааба допускается: $s_2 \rightarrow s_1 \rightarrow s_0 \rightarrow s_4 \rightarrow s_0 \rightarrow s_4$

В принципе, все возможные множества состояний можно считать новыми состояниями и моделировать работу КА с такими состояниями

Формальное определение НДКА



- 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
 - $\delta: S \times \Sigma \rightarrow 2^S$ – функция переходов
 - $F \subseteq S$ – множество финальных состояний

- Формальное задание автомата примера: $A=(S, \Sigma, s_0, \delta, F)$

- $S=\{s_0, s_1, s_2, s_3, s_4\}$; $\Sigma = \{a, b\}$; $S_0=\{s_0, s_2\}$

- $\delta(s_0, a)=\{s_4\}$;
 $\delta(s_1, b)=\{s_4\}$;
 $\delta(s_2, a)=\{s_4\}$;
 $\delta(s_2, b)=\{s_2, s_3, s_4\}$;
 $\delta(s_3, a)=\{s_0, s_1, s_4\}$;

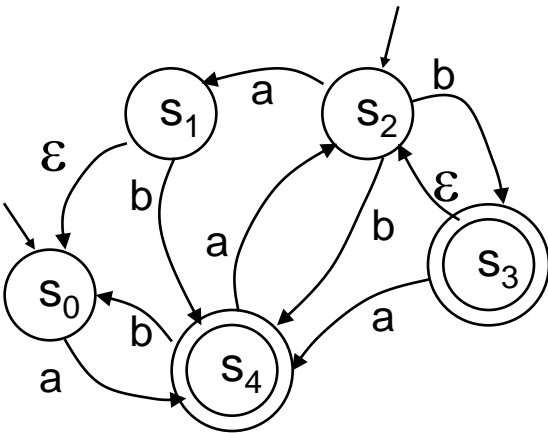
...

$\delta(\{s_2, s_3, s_4\}, a) = \{s_0, s_1, s_3, s_4\}$;

Спонтанные переходы $\delta(s_1, \epsilon)=\{s_1, s_0\}$; $\delta(s_3, \epsilon)=\{s_3, s_2\}$;

- $F=\{s_3, s_4\}$.

Приведение к НДКА без ε -переходов



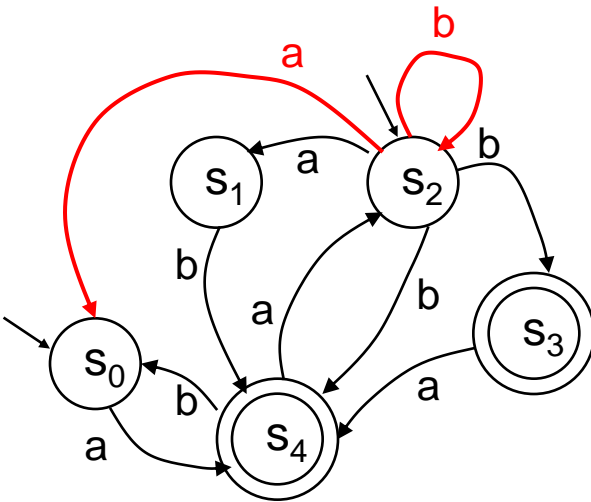
- Недетерминированный конечный автомат-распознаватель $A=(S, \Sigma, s_0, \delta, F)$, где:
 - S – конечное множество состояний
 - Σ – конечное множество входных символов
 - $s_0 \in S$ – множество начальных состояний
 - $\delta: S \times \Sigma \rightarrow 2^S$ – функция переходов
 - $F \subseteq S$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

Для каждого состояния удобно определить множество его ε -преемников:

для s_1 – это $\{s_0\}$;

для s_3 – это $\{s_2\}$;

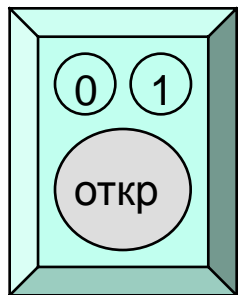
для остальных – пустые множества



Правило:

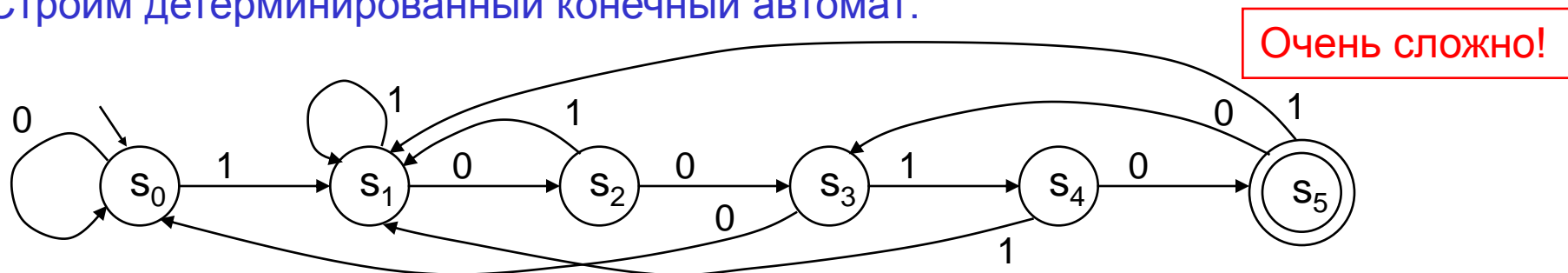
Если под воздействием **a** можем перейти в s_1 , то под воздействием **a** можем перейти и во все состояния из множества ε -преемников состояния s_1

Чем удобны недетерминированные КА? Пример 1

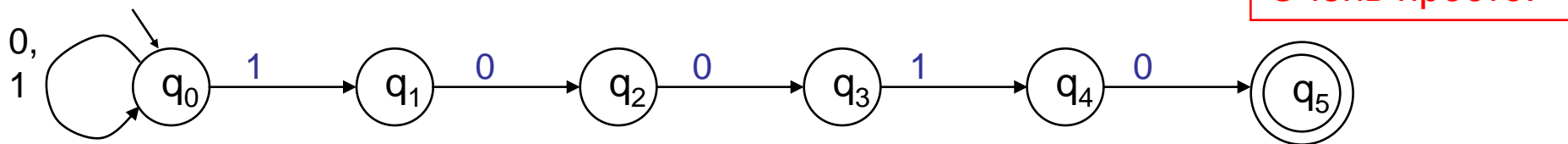


- **Проблема:** построить кодовый замок.
- Замок должен открываться при наборе любого кода, заканчивающегося на **10010**, например, 0100110**10010**.
- Нажатие "**откр**" либо откроет дверь, либо, после неправильного набора кода, стукнет током

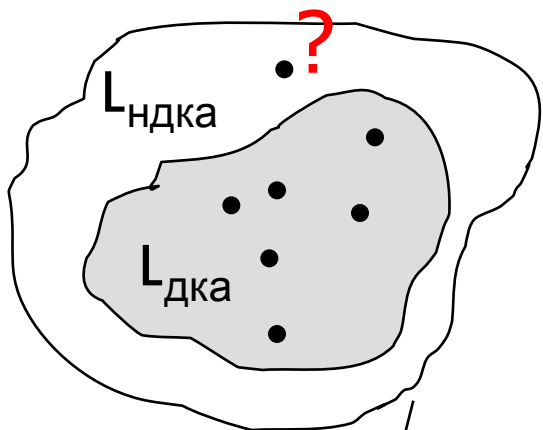
Строим детерминированный конечный автомат:



Строим НЕдетерминированный конечный автомат:

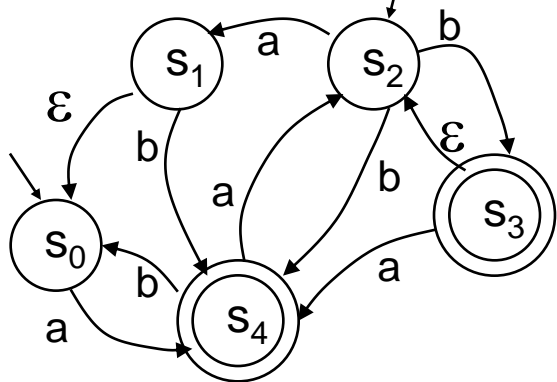


Распознающая мощность недетерминированных КА



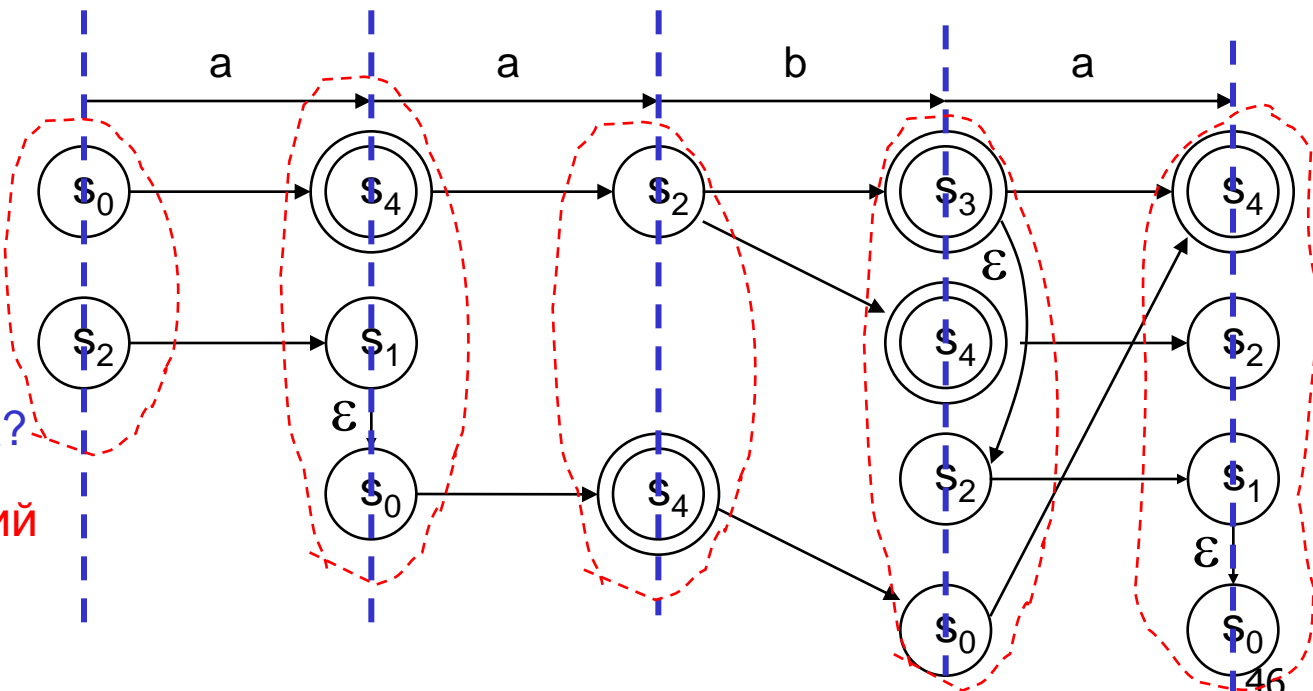
- Ясно, что детерминированные конечные автоматы – подкласс недетерминированных КА

- Увеличиваются ли возможности КА по заданию языков при недетерминизме? **НЕТ!**

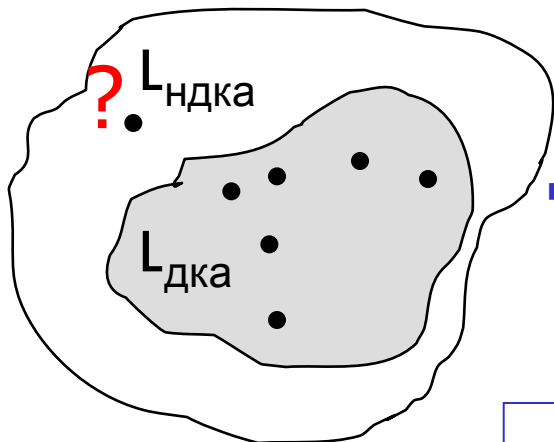


Каковы реакции на ааба?

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

$$(\forall A \in \text{NFA}) (\exists B \in \text{DFA}) A \approx B$$

$$(\forall A \in \text{NFA}) (\exists B \in \text{DFA}) L(A) = L(B)$$

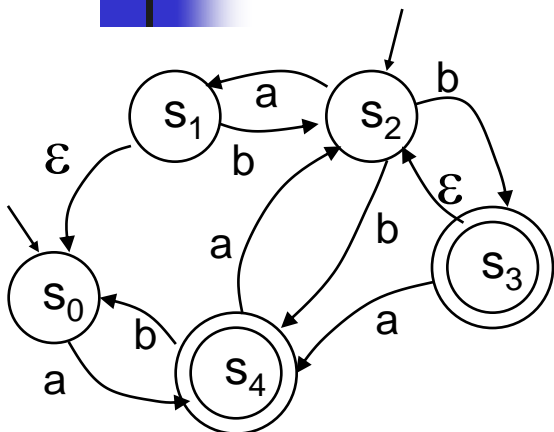
Доказательство

В качестве состояний нового автомата можно взять **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	x=ε
s_0^-	s_4		
s_1		s_2	s_0
s_2^-	s_1	$\{s_3, s_4\}$	
s_3^+	s_4		s_2
s_4^+	s_2	s_0	

	δ	x=a	x=b
$q0^-$	$\{s_0^-, s_2^-\}^-$	$\{s_0, s_1, s_4\} q_1$	$\{s_2, s_3, s_4\} q_2$
$q1^+$	$\{s_0, s_1, s_4\}^+$	$\{s_2, s_4\} q_3$	$\{s_0, s_2\} q_4$
$q2^+$	$\{s_2, s_3, s_4\}^+$	$\{s_0, s_1, s_2, s_4\} q_5$	$\{s_0, s_2, s_3, s_4\} q_6$
$q3^+$	$\{s_2, s_4\}^+$	$\{s_0, s_1, s_2\} q_1$	$\{s_0, s_2, s_3, s_4\} q_6$
$q4$	$\{s_0, s_2\}$	$\{s_0, s_1, s_4\} q_1$	$\{s_2, s_3, s_4\} q_2$
$q5^+$	$\{s_0, s_1, s_2, s_4\}^+$	$\{s_0, s_1, s_2, s_4\} q_5$	$\{s_0, s_2, s_3, s_4\} q_6$
$q6^+$	$\{s_0, s_2, s_3, s_4\}^+$	$\{s_0, s_1, s_2, s_4\} q_5$	$\{s_0, s_2, s_3, s_4\} q_6$
$q7$	$\{s_0, s_1, s_2\}$	$\{s_0, s_1, s_4\} q_1$	$\{s_2, s_3, s_4\} q_2$

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

Пример: переход из $\{s_2, s_4\}$ по a: все те, куда можем попасть по a и aε

Минимизация алгоритмом "треугольника"

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

q_0^-

q_1^+

q_2^+

q_3^+

q_4

q_5^+

q_6^+

q_7

δ	$x=a$	$x=b$
$\{s_0^-, s_2^-\}^-$	$\{s_0, s_1, s_4\} \text{ } q_1$	$\{s_2, s_3, s_4\} \text{ } q_2$
$\{s_0, s_1, s_4\}^+$	$\{s_2, s_4\} \text{ } q_3$	$\{s_0, s_2\} \text{ } q_4$
$\{s_2, s_3, s_4\}^+$	$\{s_0, s_1, s_2, s_4\} \text{ } q_5$	$\{s_0, s_2, s_3, s_4\} \text{ } q_6$
$\{s_2, s_4\}^+$	$\{s_0, s_1, s_2\} \text{ } q_1$	$\{s_0, s_2, s_3, s_4\} \text{ } q_6$
$\{s_0, s_2\}$	$\{s_0, s_1, s_4\} \text{ } q_1$	$\{s_2, s_3, s_4\} \text{ } q_2$
$\{s_0, s_1, s_2, s_4\}^+$	$\{s_0, s_1, s_2, s_4\} \text{ } q_5$	$\{s_0, s_2, s_3, s_4\} \text{ } q_6$
$\{s_0, s_2, s_3, s_4\}^+$	$\{s_0, s_1, s_2, s_4\} \text{ } q_5$	$\{s_0, s_2, s_3, s_4\} \text{ } q_6$
$\{s_0, s_1, s_2\}$	$\{s_0, s_1, s_4\} \text{ } q_1$	$\{s_2, s_3, s_4\} \text{ } q_2$

Начальное заполнение:

для каждой пары $\langle p, q \rangle$, для которой $(p \in F) \oplus (q \in F)$, ставим N

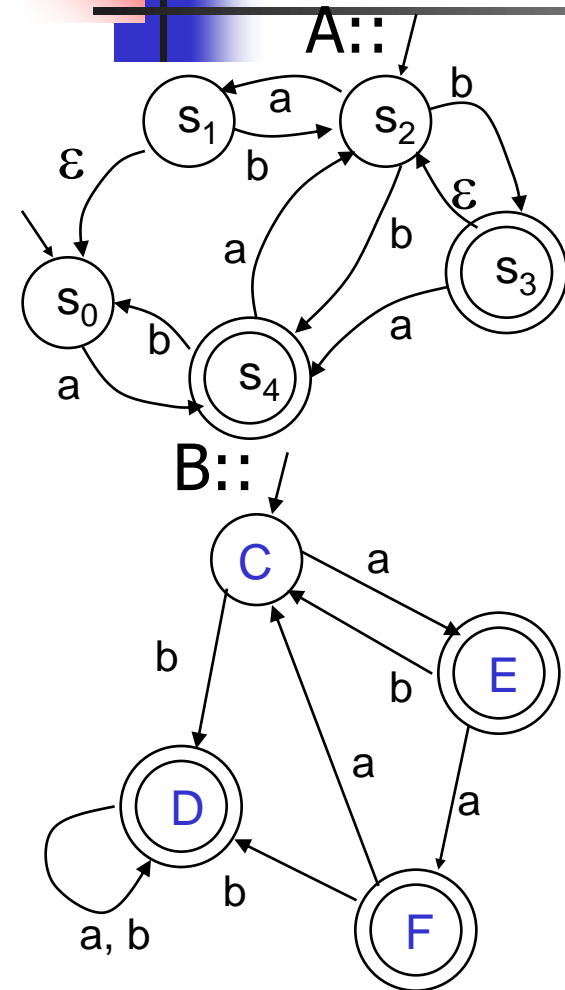
Многократно:

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

Алгоритм повторяется, пока добавляется хотя бы одно N

$$\pi_\infty = \{q_0, q_4, q_7\}; \{q_2, q_5, q_6\}; \{q_1\}; \{q_3\}$$

Минимизация построенного детерминированного автомата, эквивалентного исходному



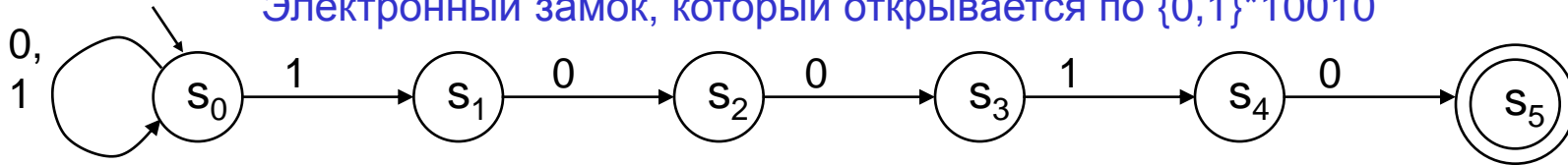
	δ	$x=a$	$x=b$
$q0^-$	$\{s_0^-, s_2^-\}^-$	$\{s_0, s_1, s_4\}$ q_1	$\{s_2, s_3, s_4\}$ q_2
$q1^+$	$\{s_0, s_1, s_4\}^+$	$\{s_2, s_4\}$ q_3	$\{s_0, s_2\}$ q_4
$q2^+$	$\{s_2, s_3, s_4\}^+$	$\{s_0, s_1, s_2, s_4\}$ q_5	$\{s_0, s_2, s_3, s_4\}$ q_6
$q3^+$	$\{s_2, s_4\}^+$	$\{s_0, s_1, s_2\}$ q_1	$\{s_0, s_2, s_3, s_4\}$ q_6
$q4$	$\{s_0, s_2\}$	$\{s_0, s_1, s_4\}$ q_1	$\{s_2, s_3, s_4\}$ q_2
$q5^+$	$\{s_0, s_1, s_2, s_4\}^+$	$\{s_0, s_1, s_2, s_4\}$ q_5	$\{s_0, s_2, s_3, s_4\}$ q_6
$q6^+$	$\{s_0, s_2, s_3, s_4\}^+$	$\{s_0, s_1, s_2, s_4\}$ q_5	$\{s_0, s_2, s_3, s_4\}$ q_6
$q7$	$\{s_0, s_1, s_2\}$	$\{s_0, s_1, s_4\}$ q_1	$\{s_2, s_3, s_4\}$ q_2

- $\pi_0 = \{q_0, q_4, q_7\}=A; \{q_1, q_2, q_3, q_5, q_6\}=B$
- $\pi_1 = \{q_0, q_4, q_7\}=C; \{q_2, q_5, q_6\}=D; \{q_1\}=E; \{q_3\}=F$
- $\pi_2 = \{q_0, q_4, q_7\}; \{q_2, q_5, q_6\}; \{q_1\}; \{q_3\} = \pi_1 = \pi_\infty$

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

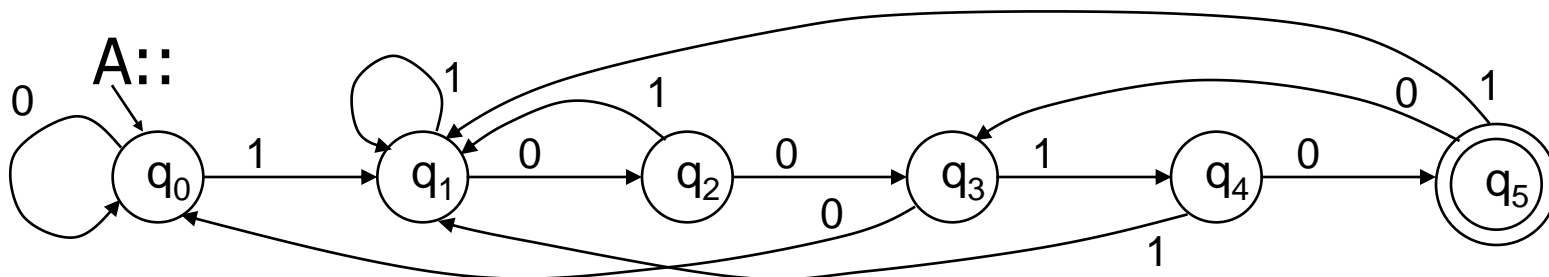
Электронный замок, который открывается по $\{0,1\}^*10010$

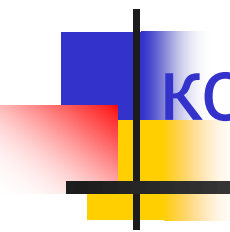


δ	x=0	x=1
s_0^-	s_0	$\{s_0, s_1\}$
s_1	s_2	
s_2	s_3	
s_3		s_4
s_4	s_5	
s_5^+		s_0

δ	x=0	x=1
q_0^-	$\{s_0\}$	$\{s_0, s_1\}$
q_1	$\{s_0, s_1\}$	$\{s_0, s_1\}$
q_2	$\{s_0, s_2\}$	$\{s_0, s_1\}$
q_3	$\{s_0, s_3\}$	$\{s_0, s_1, s_4\}$
q_4	$\{s_0, s_1, s_4\}$	$\{s_0, s_1\}$
q_5^+	$\{s_0, s_2, s_5\}$	$\{s_0, s_1\}$

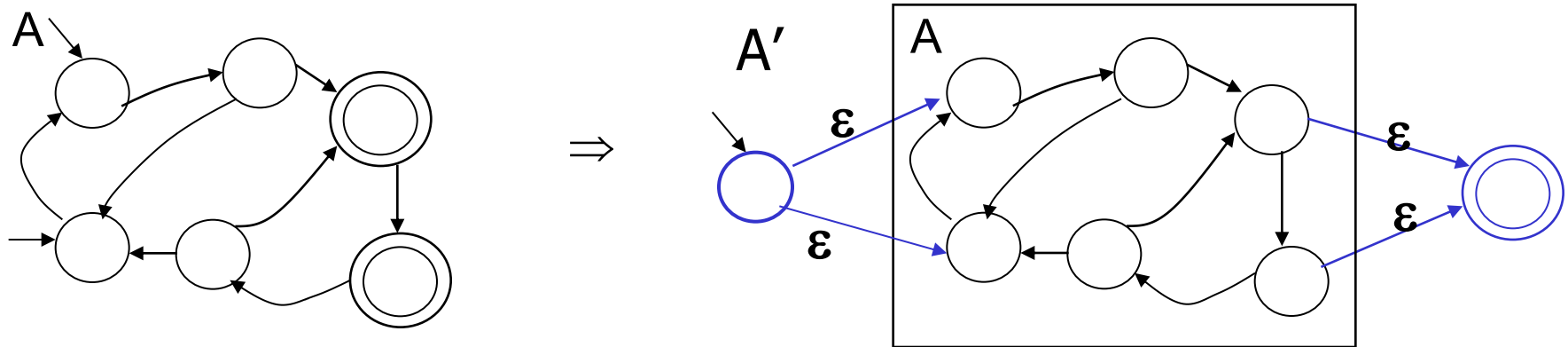
Автомат А, который строили с большим трудом, построен автоматически из недетерминированного автомата





Операции над автоматными языками и конечными автоматами-распознавателями

- Используя переход, помеченный пустым символом, ЛЮБОЙ КА-распознаватель можно представить автоматом только с одним начальным и одним финальным состоянием, из начального состояния переходы только выходят, в финальное – только входят

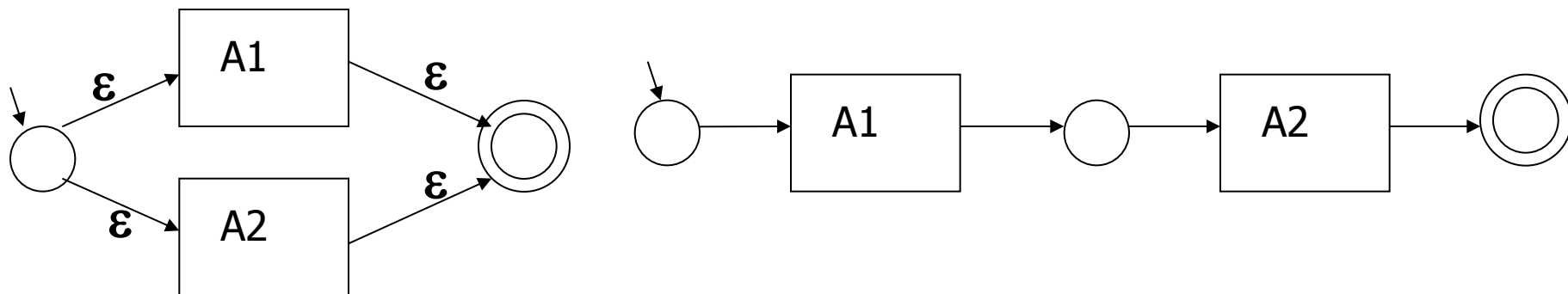


Очевидно, что автоматы A' и A эквивалентны: они распознают один и тот же язык

Операции над автоматными языками

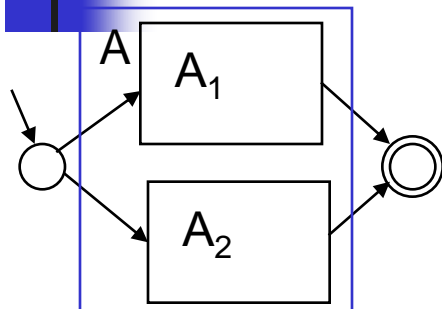
- **Теорема.** Если L_1 и L_2 – автоматные языки, то автоматными являются также их объединение $L_1 \cup L_2$, пересечение $L_1 \cap L_2$, конкатенация $L_1 \bullet L_2$, дополнение $\Sigma - L_1$, итерация L_1^*

■ **Доказательство.** Если L_1 и L_2 – автоматные языки, то для них существуют конечные автоматы, распознающие их. Пусть это автоматы A_1 и A_2 . Можно считать, что A_1 и A_2 одно начальное и одно принимающее состояние



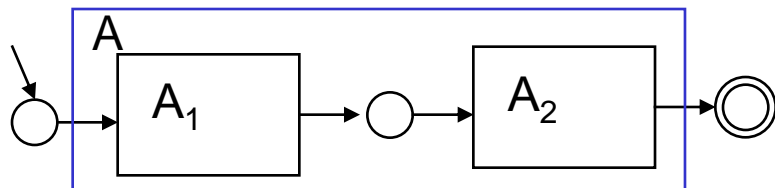
- Объединение двух автоматных языков $L_1 \cup L_2$ допускает автомат $A_1 \cup A_2$
- Пересечение двух автоматных языков $L_1 \cap L_2$ допускает автомат $A_1 \times A_2$
- Конкатенацию двух автоматных языков $L_1 L_2$ допускает автомат $A_1 \bullet A_2$
- Дополнение $\Sigma - L_1$ автоматного языка L_1 распознает автомат, полученный из A_1 с помощью простой операции: непринимающие состояния A_1 сделаем принимающими, а принимающие состояния A_1 сделаем непринимающими

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



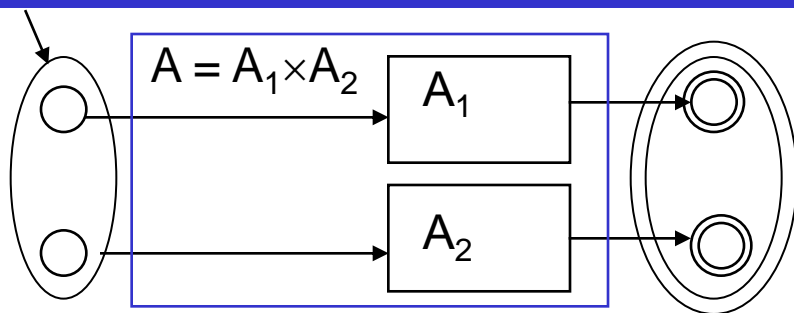
- $A = A_1 \cup A_2$

- $L_A = L_{A_1} \cup L_{A_2}$



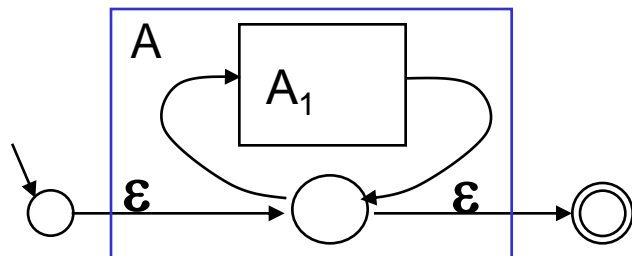
- $A = A_1 \bullet A_2$

- $L_A = L_{A_1} \bullet L_{A_2}$



- $A = A_1 \times A_2$

- $L_{A_1 \times A_2} = L_{A_1} \cap L_{A_2}$



- $A = A_1^*$

- $L_A = L_{A_1}^*$



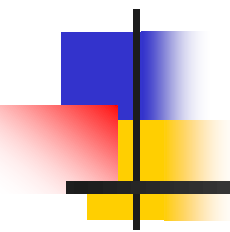
Резюме: операции над автоматными языками

- **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 $L1 \bullet L2$
- итерция (звезда Клини) $L1^*$

ИСХОДНЫХ ЯЗЫКОВ



Пример применения недетерминированных КА: Римская система счисления

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)

Каковы формальные правила
(грамматика) построения римских чисел?

Построить детерминированный автомат
довольно сложно

Примеры чисел

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

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 большему. 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

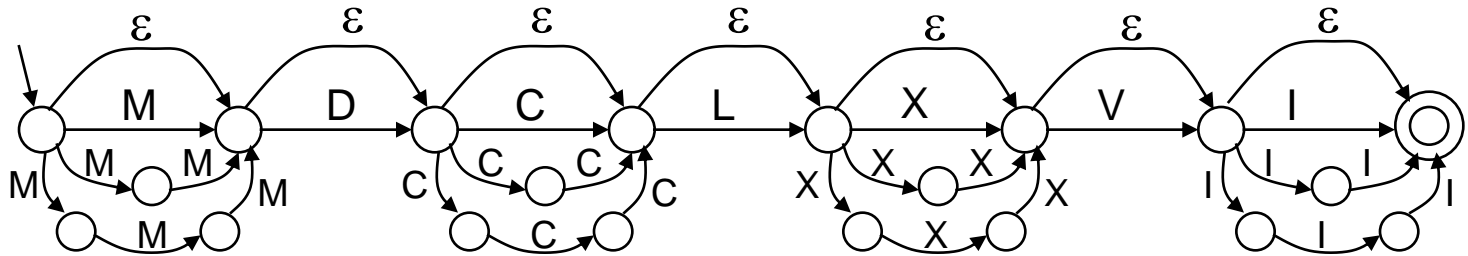
Правила построения римских чисел: формально (1)

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



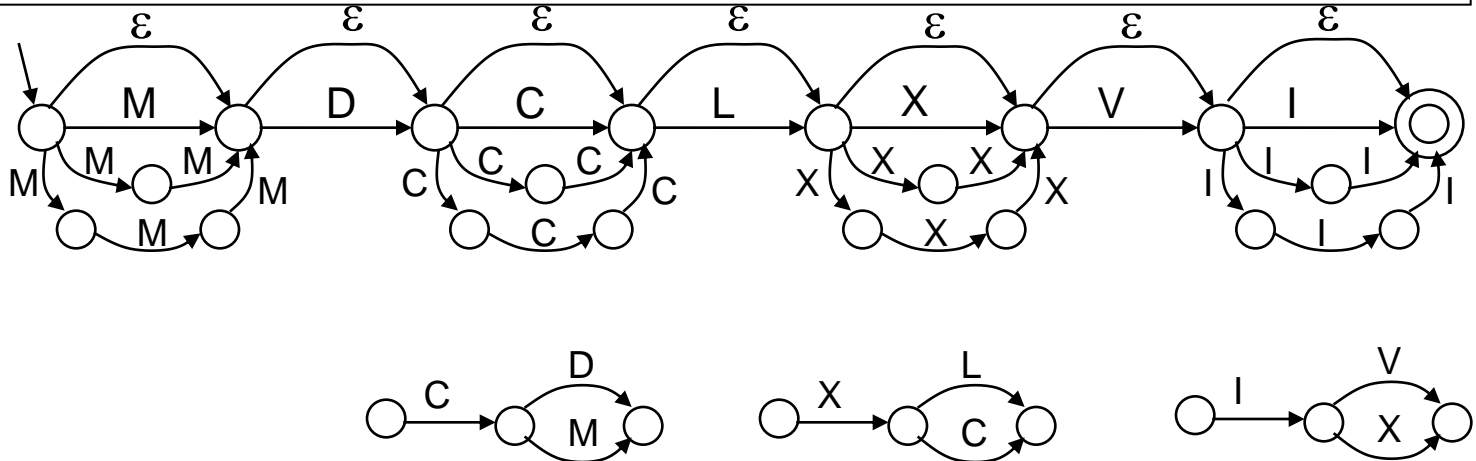
Построить детерминированный автомат сложно
Строим недетерминированный

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



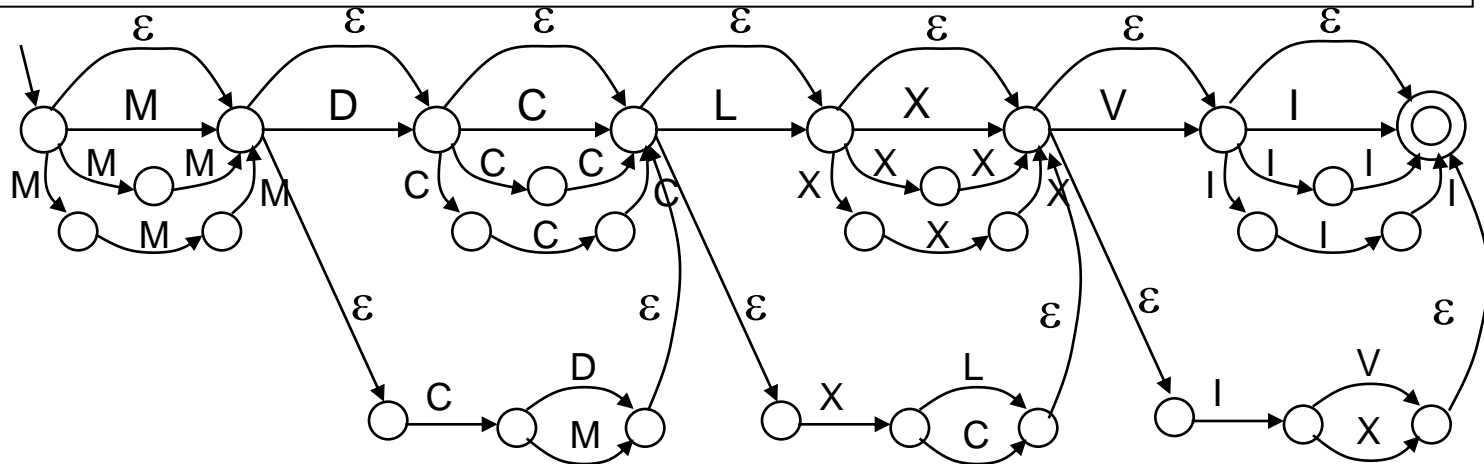
Построить детерминированный автомат сложно
Строим недетерминированный

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

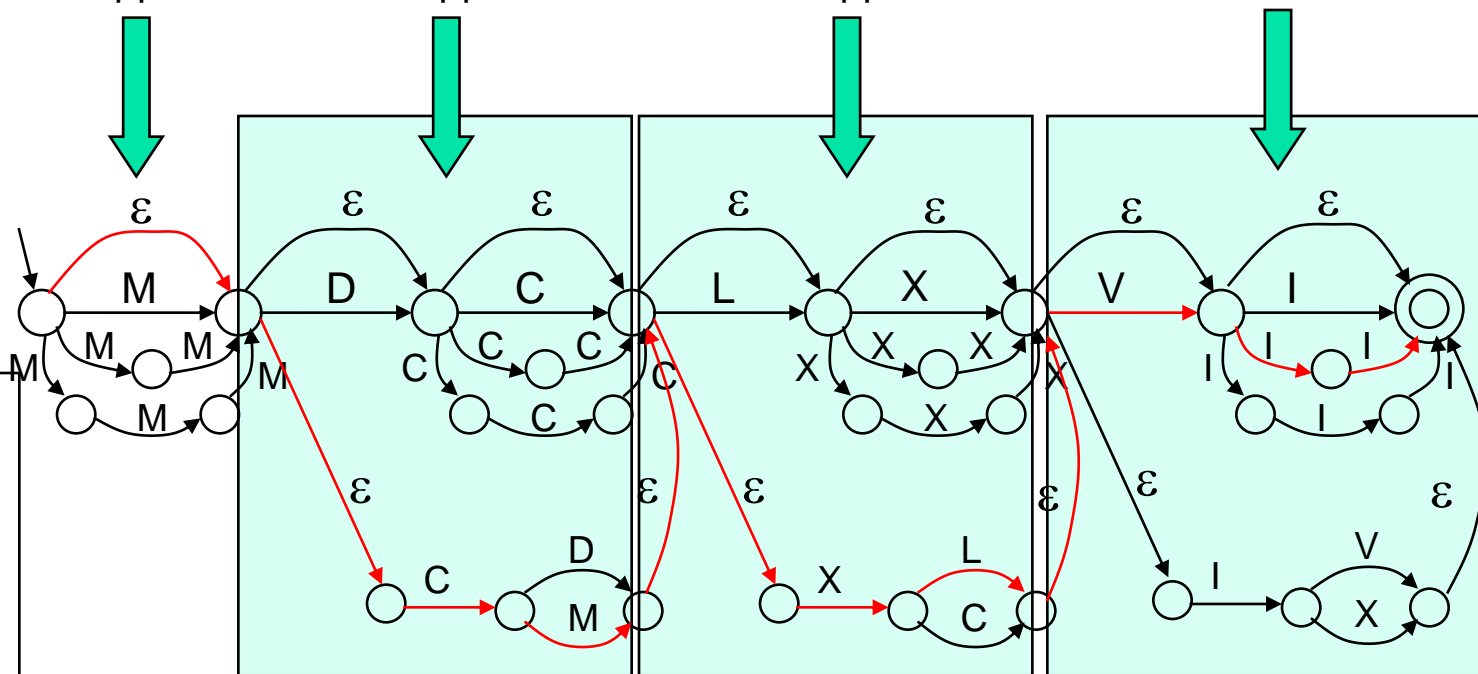
IV = 4
LXXXVII = 87
XLIII = 43
XC = 90
CD = 400
CMXCVII = 997

Тысячи:
от 0 до 3-х

Сотни:
от 0 до 9

Десятки:
от 0 до 9

Единицы:
от 0 до 9



Пример вывода (правильные цепочки языка, т.е. числа в Римской системе счисления): любой путь в заключительное состояние – **CMXLVII** – 947

Правила построения римских чисел: формально (4)

Сколько всего цепочек?

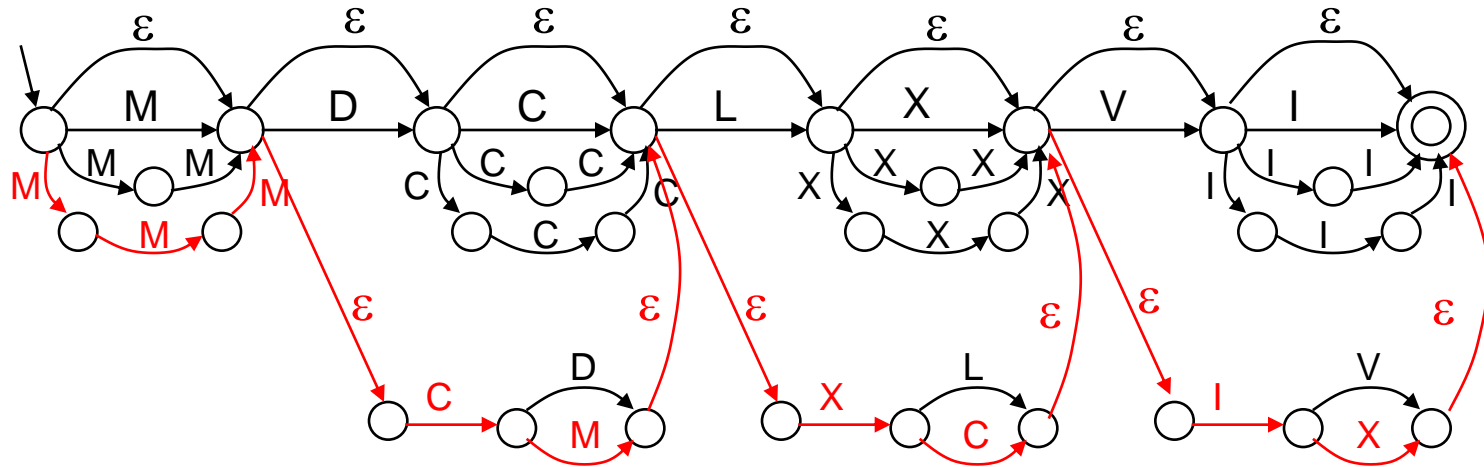
$$4 \times 10 \times 10 \times 10 = 4000$$

Минимальное число 0 – отсутствие символов

Максимальное число **MMMCMXCIX = 3999**

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



Большая сложность выполнения операций:
XLVIII + VI = LIV – 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 либо программно, либо 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 $\Sigma^* - L$
 - построение **синхронной композиции** двух КА-распознавателей = построение автомата, распознающего пересечение двух автоматных языков
 - **асинхронная композиция** двух КА – модель параллельных процессов
 - 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