# Об анализе расширенных регулярных языков с памятью

Дельман А.Д.

МГТУ им. Н.Э. Баумана

01.07.24

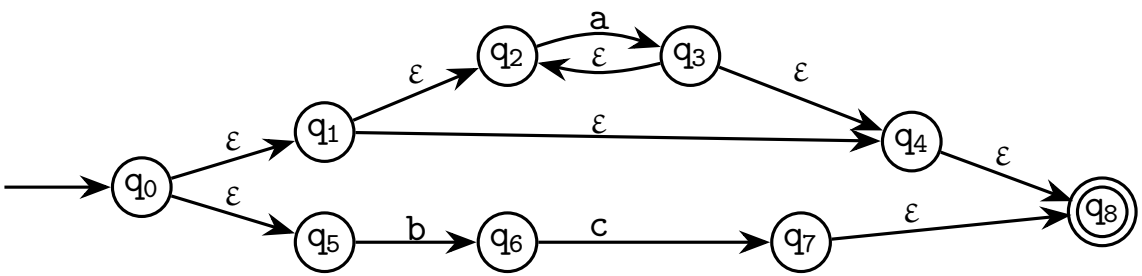
# Регулярные выражения и конечные автоматы

**Академические регулярные выражения $\mathcal{RE}$**

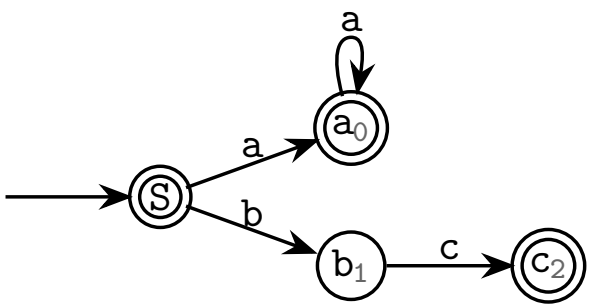
- $\alpha_1, \alpha_2 \in \mathcal{RE}, \alpha_1 \mid \alpha_2$ — описывает слова или из $\alpha_1$, или из $\alpha_2$ (альтернатива);
- $\alpha_1, \alpha_2 \in \mathcal{RE}, \alpha_1\alpha_2$ — описывает слова с префиксами из $\alpha_1$ и суффиксами из $\alpha_2$ (конкатенация);
- $\alpha \in \mathcal{RE}, \alpha^*$ - ноль или более конкатенаций $\alpha$ с собой (итерация Клини).

Регулярное выражение: a* | bc

**Автомат Томпсона**




**Автомат Глушкова**



# Обратные ссылки

Обратные ссылки (backreferences), как расширение регулярных языков, довольно распространено в современных библиотеках. Позволяют обращаться к ячейкам памяти, содержимое которых описывается подвыражениями.

- $\varepsilon$-семантика — ссылки на неинициализированную память считаются $\varepsilon$;
- $\varnothing$-семантика — ссылки на неинициализированную память недопустимы.

# Формализм CSY

## Формализация Кампеану–Саломаа–Ю

- Безымянные группы захвата:
  Каждая скобочная группа $(\ldots)$ считается группой захвата в память и автоматически нумеруется по очерёдности вхождений открывающих скобок в выражение.
- Ссылка $\backslash i$ на скобочную группу с номером $i$ может появиться, только если скобочная группа уже закрыта;
- $\varnothing$-семантика.

**Пример**

- $((a^*)b^*)\backslash 2\backslash 1$ описывает язык $a^n b^m a^{2n} b^m$;
- $((a^*)b^* \mid \backslash 1)^*\backslash 2$ некорректно в CSY.

# Формализм Шмида

**Определение**

*Reference word* (ref-слово) — регулярные выражения, расширенные специальными символами $[_\mathtt{i}, ]_\mathtt{i}, \&\mathtt{i} \mid \mathtt{i} \in \mathbb{N}$, где $\&\mathtt{i}$ — переменная, а $[_\mathtt{i}, ]_\mathtt{i}$ — скобки, выделяющие подвыражение для нее. При этом если выражение $[_\mathtt{i}\omega]_\mathtt{i}$ — ref-слово, то $\omega$ не содержит $\&\mathtt{i}$. Допускает несбалансированность именованных скобочных групп.

Далее рассматриваем выражения в $\varepsilon$-семантике и со сбалансированными квадратными скобками.

# MFA

**Конечный автомат с памятью — *MFA***

Пятерка $\langle Q, \Sigma, \delta, q_0, F \rangle$:

- $Q$ — множество состояний автомата;
- $\Sigma$ — входной (терминальный) алфавит автомата;
- $\delta$ — множество правил перехода. Переходы помечаются действиями над памятью: $o$ — открытие ячейки, $c$ — закрытие ячейки, $r$ — сброс до $\varepsilon$;
- $q_0 \in Q$ — начальное состояние, $F \subseteq Q$ — множество конечных состояний.

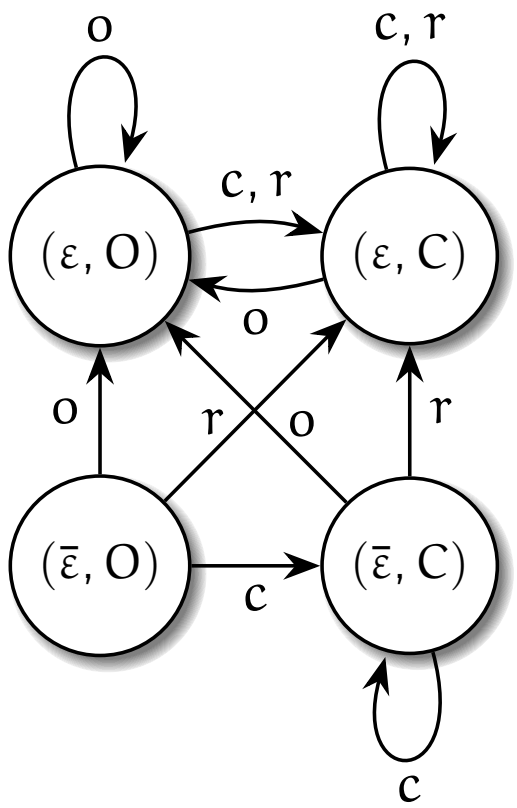
Операция сброса не применяется в оригинальной статье Шмида и необходима, чтобы сделать алгебру действий над памятью композиционно замкнутой.

# MFA состояния памяти

**Конфигурация *MFA***

- текущее состояние;
- содержимое памяти;
- конфигурации ячеек:

$$\begin{cases} \mathrm{O} & (\text{open}) \\ \mathrm{C} & (\text{closed}) \end{cases}$$

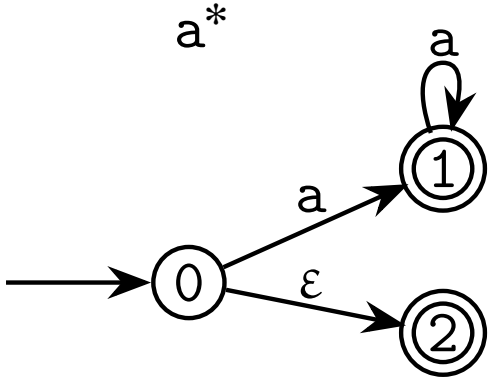



- Начальная конфигурация памяти: $\big(q_0, w, (\varepsilon, \mathrm{C}), \ldots, (\varepsilon, \mathrm{C})\big)$.
- Действия над ячейками памяти происходят до чтения с ленты.

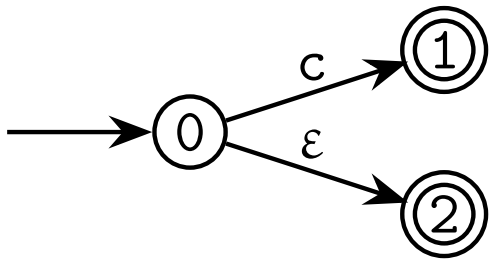
# Пример построения автомата по частям

Регулярное выражение: a*b[c |] : 1&1

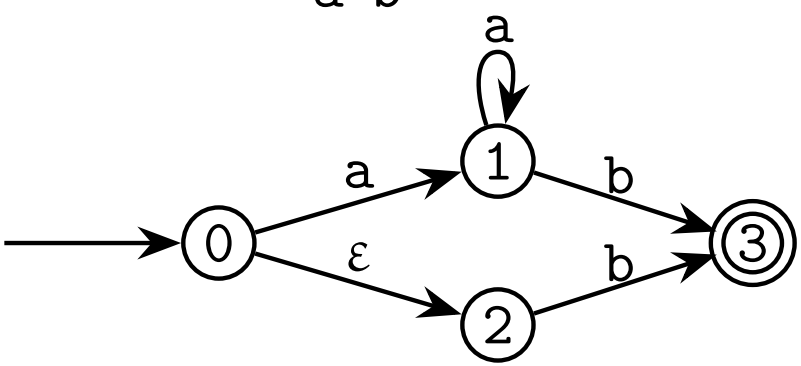
a*



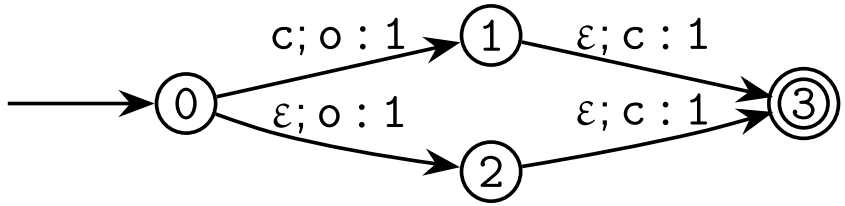

(c |)




a*b



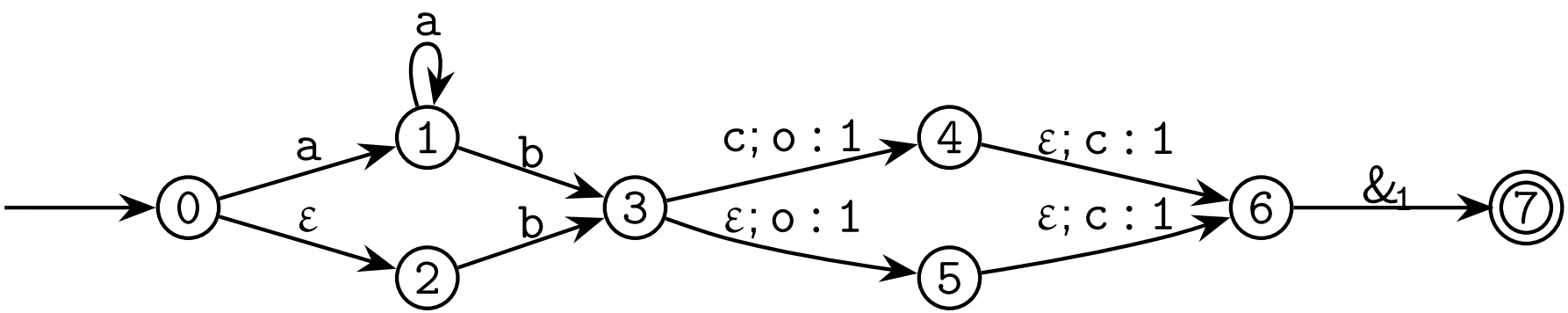

[c |] : 1



# Результат

Регулярное выражение: a*b[c |] : 1&1

Автомат:

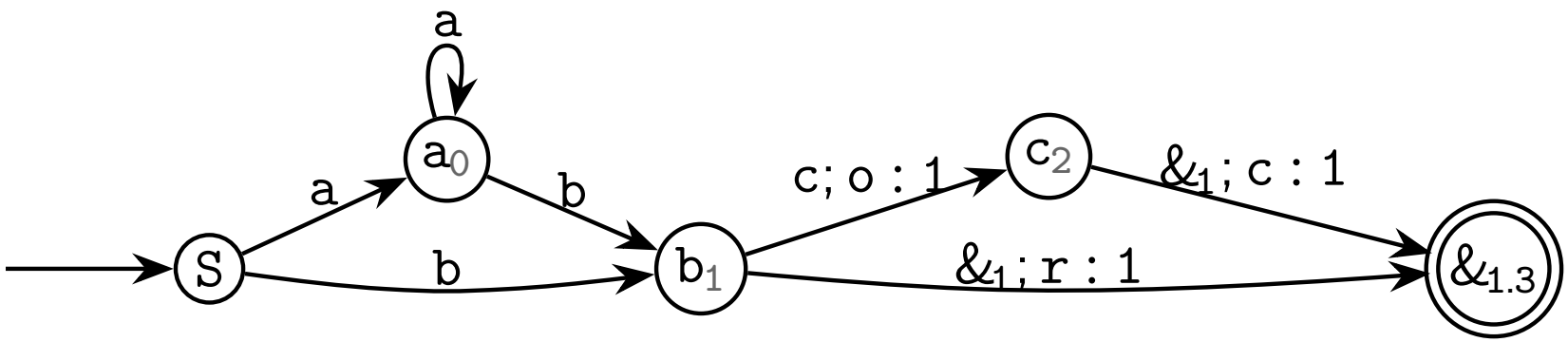


# Аналог автомата Глушкова

Регулярное выражение: $a^*b[c\ |] : 1\&1$

Линеаризованное регулярное выражение: $a_0^*b_1[c_2\ |] : 1\&_{1.3}$

Автомат:




Множество First:

$a_0,\ b_1$

Множество Last:

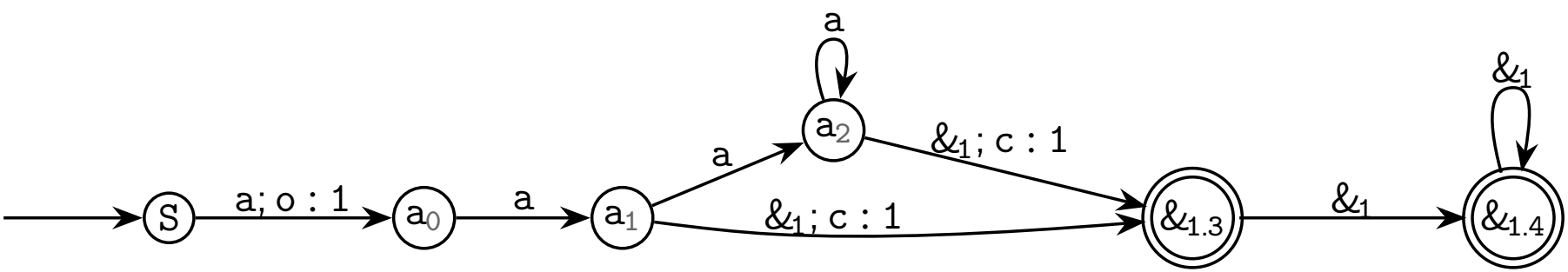
$\&_{1.3}$

Множество Follow:

$(a_0, a_0)\ (a_0, b_1)\ (b_1, c_2)\ (b_1, \&_{1.3})\ (c_2, \&_{1.3})$

# Пример *MFA*

Регулярное выражение: $[aaa^*] : 1\&1\&1^*$




При $\&1 = a^2$ распознает $a^{2(k+1)}$

$\&1 = a^3$ распознает $a^{3(k+1)}$

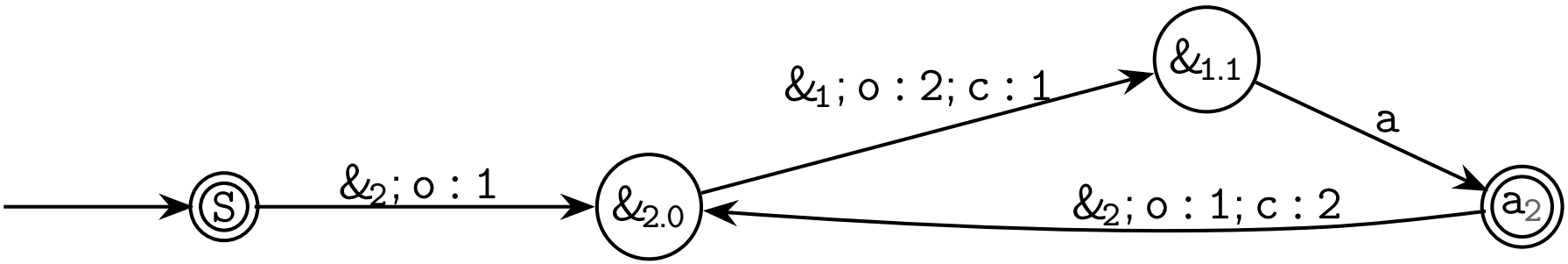
...

$\&1 = a^n$ распознает $a^{n(k+1)}$

...

# Пример *MFA*

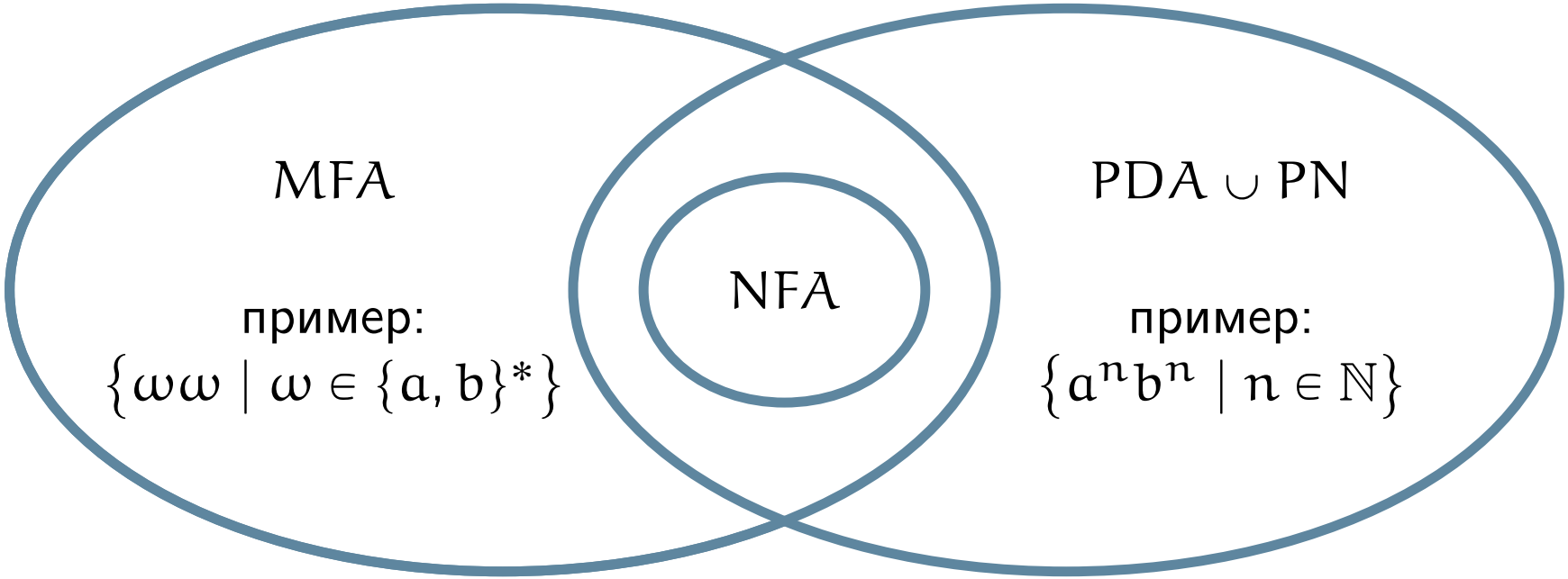
Регулярное выражение: $([\&2] : 1[\&1\mathrm{a}] : 2)^*$




Состояние памяти в финальном $\mathrm{a}_2$:

1-я итерация: $\langle(\varepsilon, \mathrm{C}), (\mathrm{a}, \mathrm{O})\rangle$, распознанное слово: $\mathrm{a}$
2-я: $\langle(\mathrm{a}, \mathrm{C}), (\mathrm{aa}, \mathrm{O})\rangle$, распознанное слово: $\mathrm{a} + \mathrm{aaa} = \mathrm{a}^4$
3-я: $\langle(\mathrm{aa}, \mathrm{C}), (\mathrm{aaa}, \mathrm{O})\rangle$, распознанное слово: $\mathrm{a}^4 + \mathrm{aaaaa} = \mathrm{a}^9$
...
n-я: $\langle(\mathrm{a}^n, \mathrm{C}), (\mathrm{a}^{n+1}, \mathrm{O})\rangle$, распознанное слово: $\mathrm{a}^{\sum_{k=0}^{n}(2k+1)} = \mathrm{a}^{(n+1)^2}$
...

# Языки *MFA*




- Существующие модели, такие как стековые автоматы и сети Петри, не подходят для описания языков *MFA*.

# Анализ регулярных языков

Некоторые разрешимые задачи:

- Эквивалентность *DFA*;
- Вложение языков;
- Эквивалентность *NFA*;
- Универсальность языка.

- Иногда на практике применяются более простые отношения, позволяющие оптимизировать анализ представлений.

# Сложность анализа расширенных языков

Образцы позволяют описывать подмножество языков ref-слов.

**Пример:** $\mathtt{aXbXcX} \to \mathtt{a}[(\mathtt{a} \mid \mathtt{b} \mid \mathtt{c})^*] : \mathtt{1b\&1c\&1}$

> *«Decision Problems for Patterns»*
> Jiang T., Salomaa A., Salomaa K., Yu S. (1993)

- Вложение языков образцов неразрешимо даже в малом алфавите.

> *«Inclusion of pattern languages and related problems»*
> Freydenberger D. D. (2011)

- Эквивалентность языков расширенных регулярных выражений неразрешима даже для малого числа ячеек памяти.

**Определение**

Labelled Transition System — тройка $\langle S, \Sigma, Q \rangle$, где $S$ — множество состояний, $\Sigma$ — множество меток, $Q$ — множество переходов (троек из $S \times \Sigma \times S$).

Любую машину состояний можно определить её графом переходов $\mathcal{T}$, то есть полному описанию всех возможных путей.

# Симуляция и бисимуляция *LTS*

**Определение**

Если $\precsim$ — симуляция для $LTS = \langle S, \Sigma, Q \rangle$, то

$$\forall p, q \in S(p \precsim q \Rightarrow (\exists p', a((p \xrightarrow{a} p') \Rightarrow \exists q'(q \xrightarrow{a} q' \ \& \ p' \precsim q')))$$

**Определение**

Если одновременно выполняются $p \precsim q$ и $q \precsim p$, то говорят, что $p$ и $q$ находятся в отношении бисимуляции (обозначается $p \sim q$).

Бисимуляция состояний в единственной *LTS* обобщается и на бисимуляцию между двумя разными *LTS*.

- Проверка бисимилярности графов переходов $\mathcal{T}_1$ и $\mathcal{T}_2$ при начальной конфигурации $\langle q_S, q'_S \rangle$ может быть представлена в виде игры между Атакующим и Защитником.

# Бисимуляция *NFA*

- Два *NFA* $\mathscr{A}_1$ и $\mathscr{A}_2$ бисимилярны ($\mathscr{A}_1 \sim \mathscr{A}_2$), если их графы переходов $\mathcal{T}(\mathscr{A}_1)$ и $\mathcal{T}(\mathscr{A}_2)$ бисимилярны.

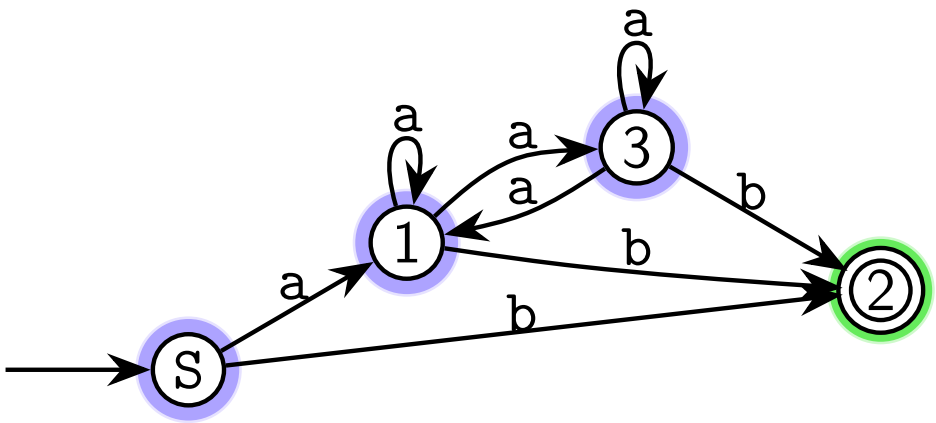
При построении бисимуляции $\mathcal{T}(\mathscr{A}_1) \sim \mathcal{T}(\mathscr{A}_2)$ нужно добавить ограничения на бисимуляцию начальных и конечных состояний.

Таким образом, для бисимуляции *NFA* $\mathscr{A}_1$ и $\mathscr{A}_2$ необходимы следующие условия:

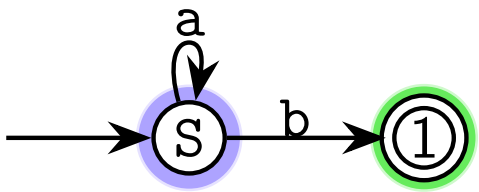
1. каждому состоянию $\mathscr{A}_1$ бисимилярно состояние $\mathscr{A}_2$, и наоборот;
2. стартовому состоянию $\mathscr{A}_1$ бисимилярно стартовое состояние $\mathscr{A}_2$;
3. каждому финальному состоянию $\mathscr{A}_1$ бисимилярно финальное состояние $\mathscr{A}_2$, и наоборот.

# Пример

Первое регулярное выражение: $(aa^*)^*b$

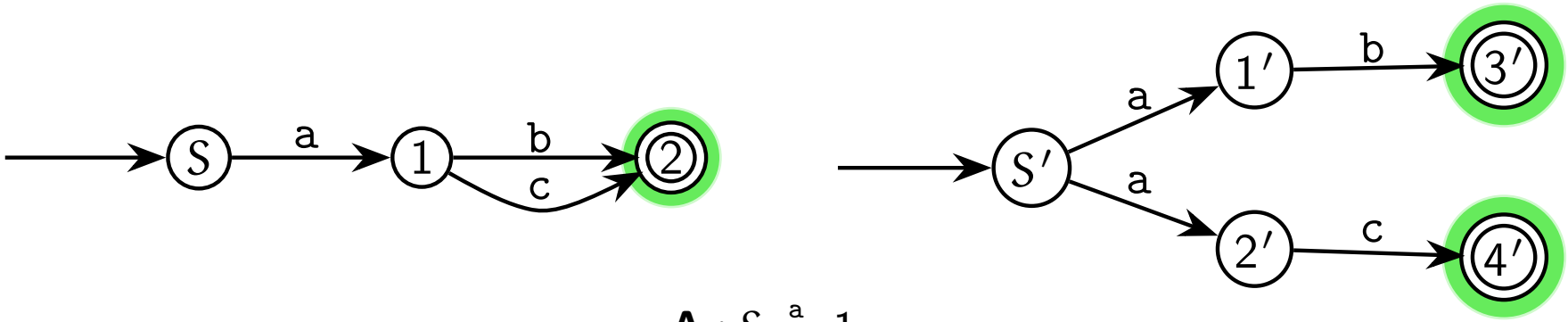


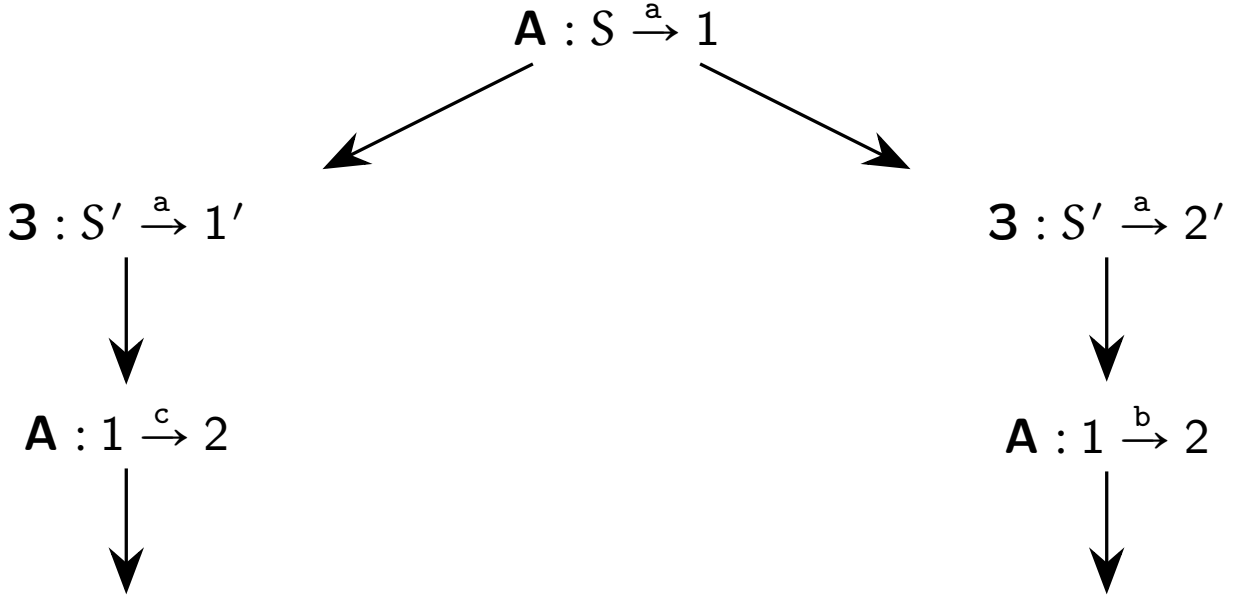

Второе регулярное выражение: $a^*b$



# Бисимуляция и эквивалентность

- Бисимилярные *NFA* распознают равные языки, но эквивалентность является менее сильным отношением.



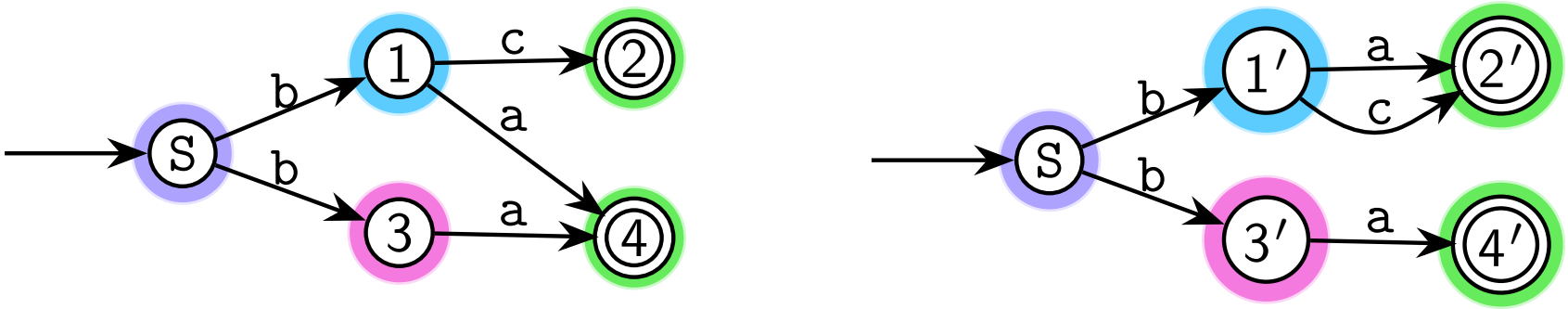




**З** проигрывает (не может перейти по c)

**З** проигрывает (не может перейти по b)

# Бисимуляция и равенство

Существования бисимуляции *NFA* недостаточно, чтобы гарантировать их буквальное равенство.




Представленные автоматы бисимилярны, но не равны.

# Слияние по бисимуляции

Бисимилярные состояния в автомате можно объединить,
при этом не изменив распознаваемый язык.

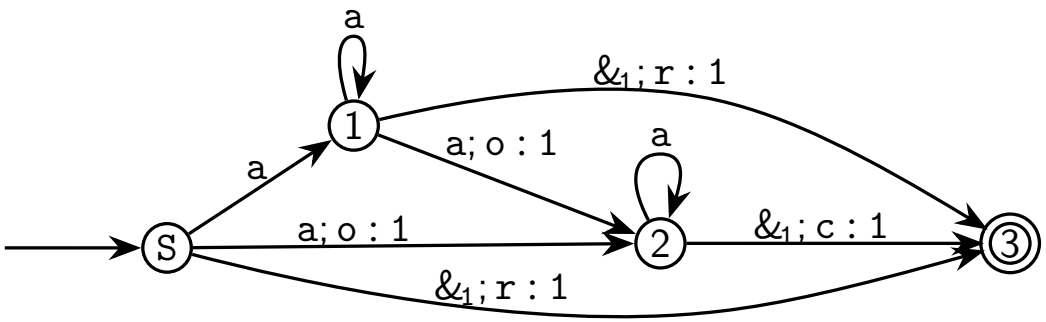
Это преобразование часто позволяет существенно упростить *NFA*.

- Для класса детерминированных конечных автоматов *DFA* такое слияние равносильно минимизации.

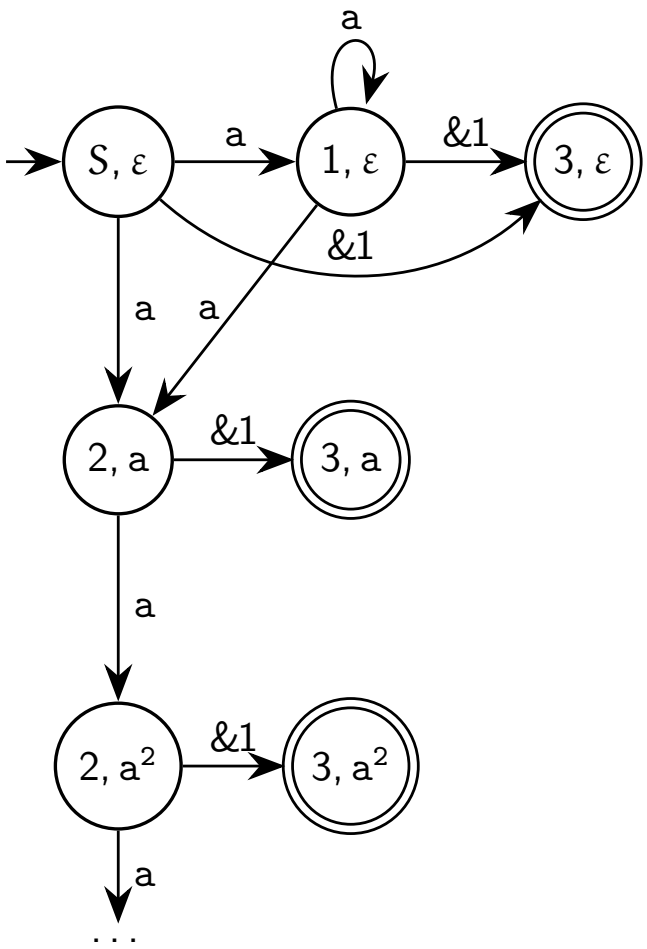
# Как определение переносится на MFA

- Все группы захвата используются: $[a^*] : 1[a^*] : 1\&1$ и т.п. исключаем;
- Переход по ссылке $\&i_{\mathscr{A}_1}$ можно симулировать только переходом по $\&i_{\mathscr{A}_2}$ с учетом содержимого.

Автомат по выражению $a^*[a^*] : 1\&1$:

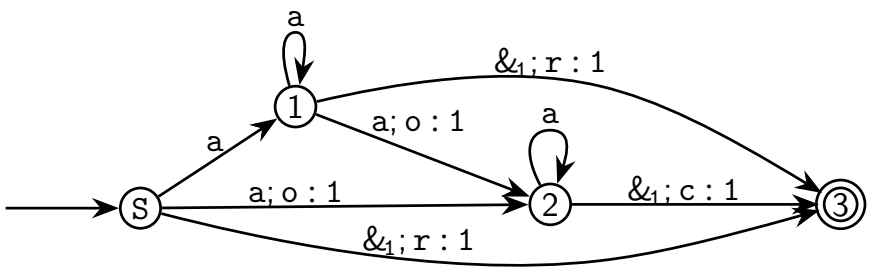



Граф переходов:

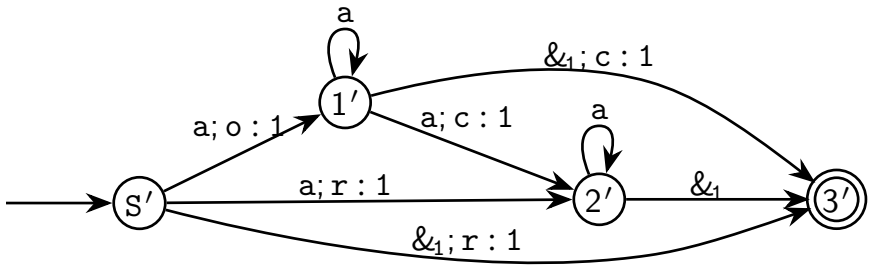


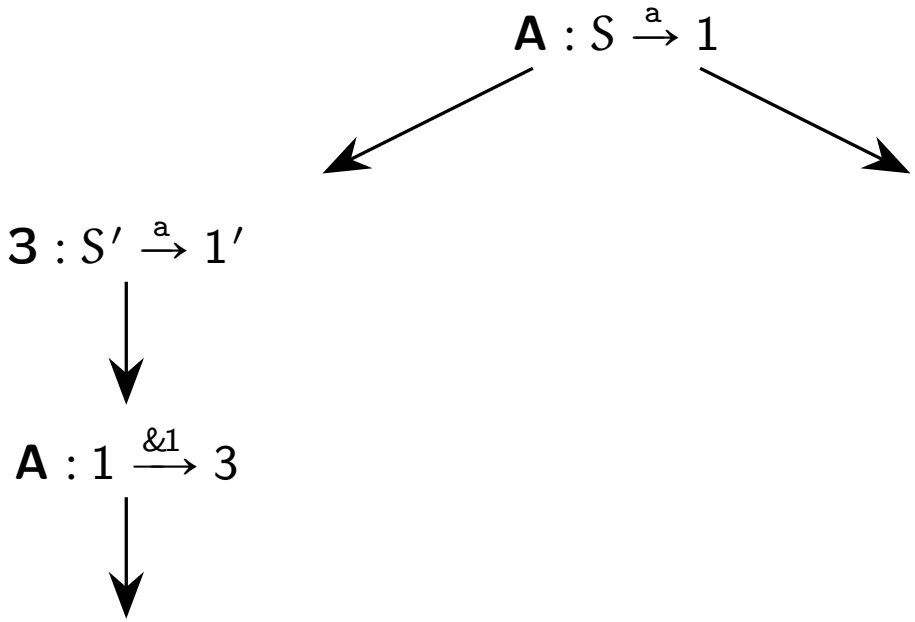
# Пример игры

Автомат по выражению a*[a*] : 1&1:




Автомат по выражению [a*] : 1a*&1:



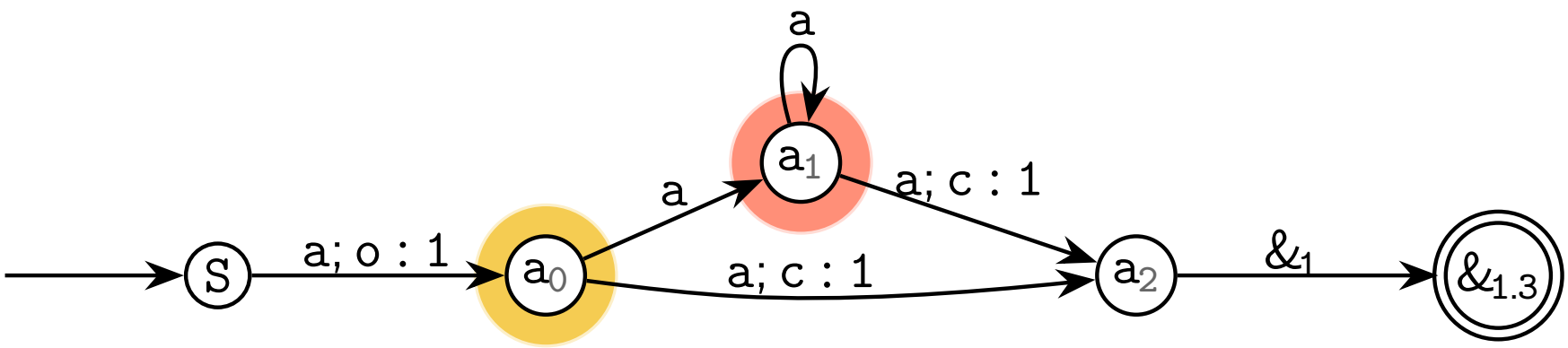




З проигрывает (не может сбросить &1)

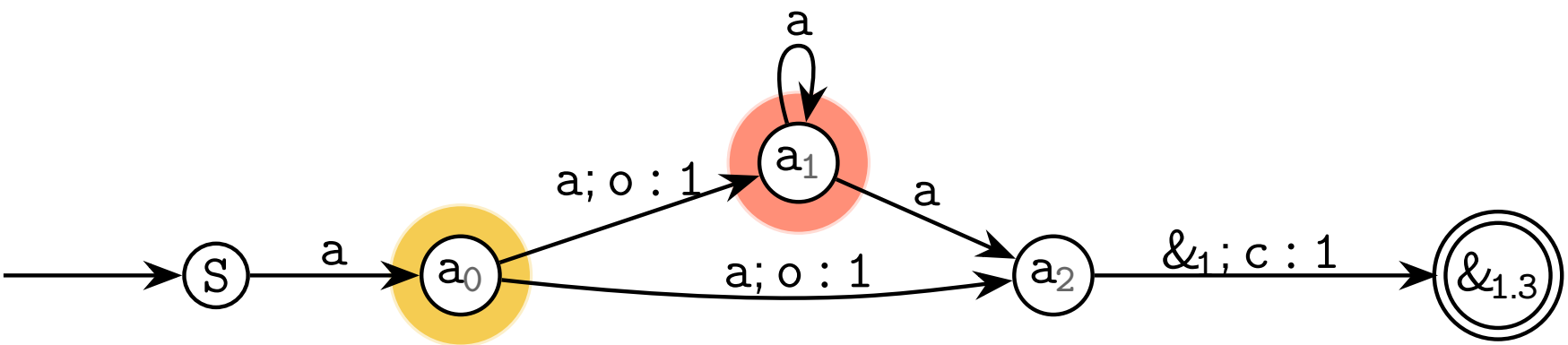
З проигрывает (не может поменять &1)

# Пример бисимилярных MFA

Первое регулярное выражение: $[aa^*] : 1a\&1$

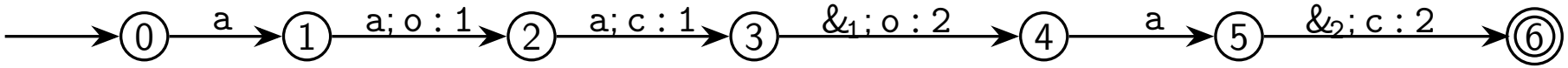


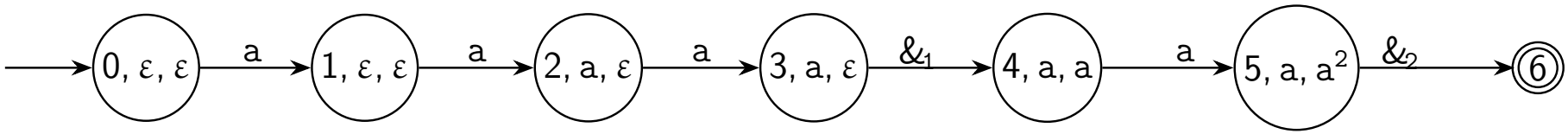

Второе регулярное выражение: $a[a^*a] : 1\&1$



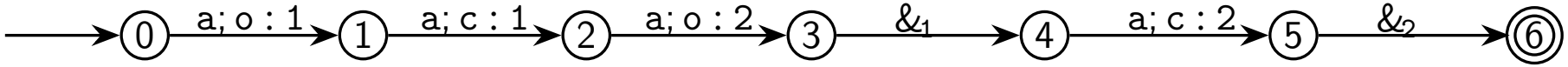
# Пример бисимилярных MFA, но сложнее

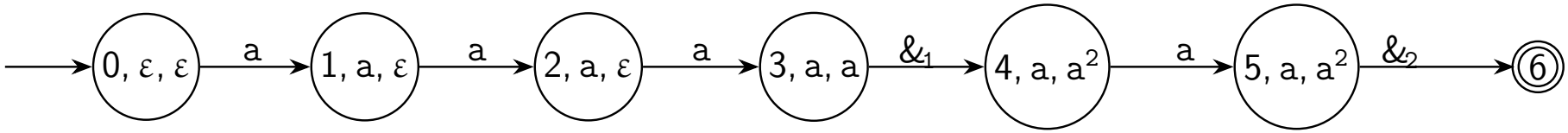
Первое регулярное выражение: $\mathtt{a[a] : 1a[\&1a] : 2\&2}$




Второе регулярное выражение: $\mathtt{[a] : 1a[a\&1] : 2a\&2}$

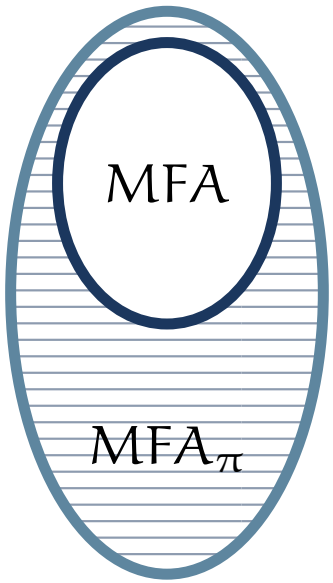


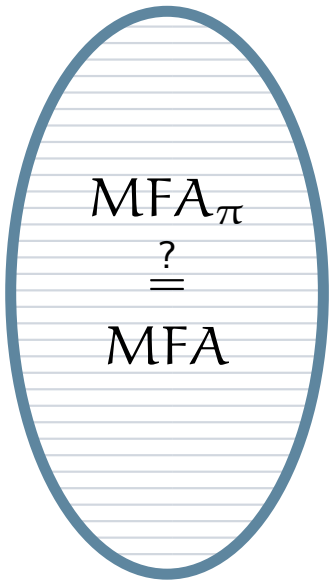
# Сложность сравнения переходов по ссылкам

Язык $\mathscr{L}(\mathrm{MFA}_\pi)$ состоит из слов, которые могут лежать с ячейках памяти.

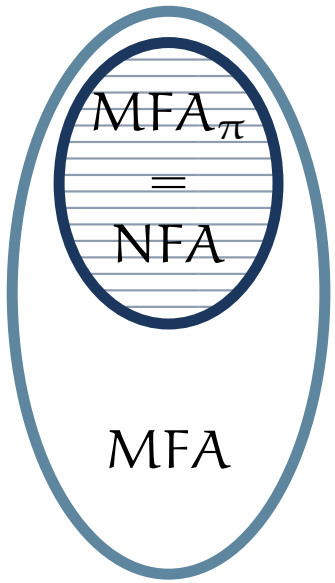
Рекурсивная память




Нерекурсивная память




Одна ячейка



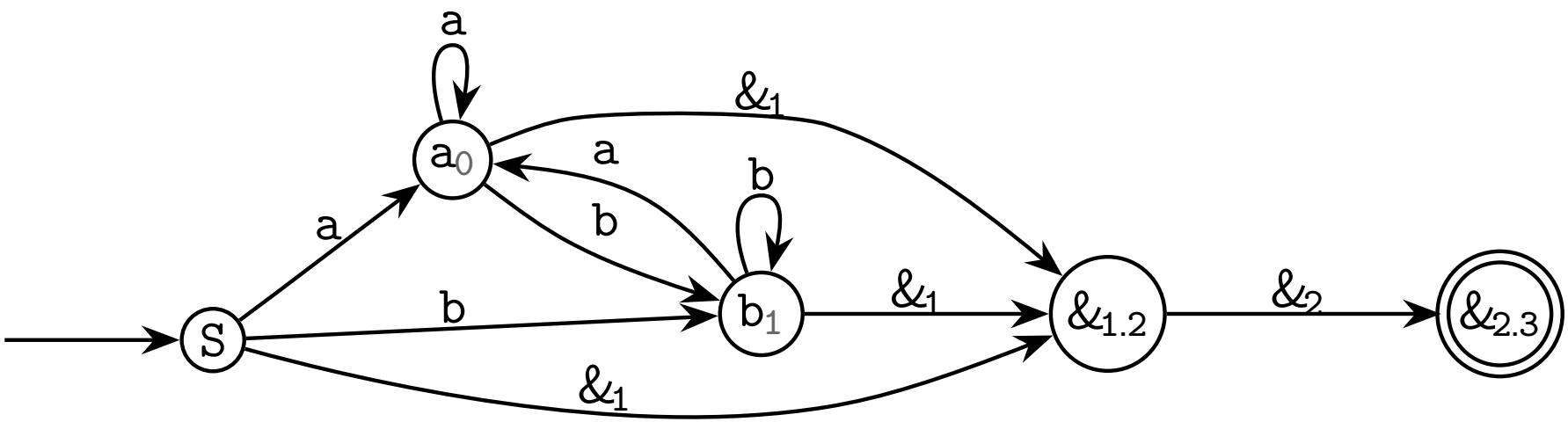

$\{a^n b^n \mid n \in \mathbb{N}\} \in \mathscr{L}(\mathrm{MFA}_\pi)$ в рекурсивном случае.
3-я ячейка хранит такой язык: $\left([_2 a\ \&1\ b]_2 [_1 a\ \&2\ b]_1\right)^* [_3 (\&1 \mid \&2)]_3$.

# Аппроксимации конечными автоматами

`Action-NFA` — $\pi_{\mathcal{M}}(\mathscr{A})$:

- Операции над памятью удаляются;
- Ссылки становятся элементами входного алфавита;
- `Action`-бисимуляция **необходима** для бисимуляции *MFA*.

Регулярное выражение: $([\mathtt{a}] : 1 \mid [\mathtt{b}] : 2)^* \,\&1\,\&2$

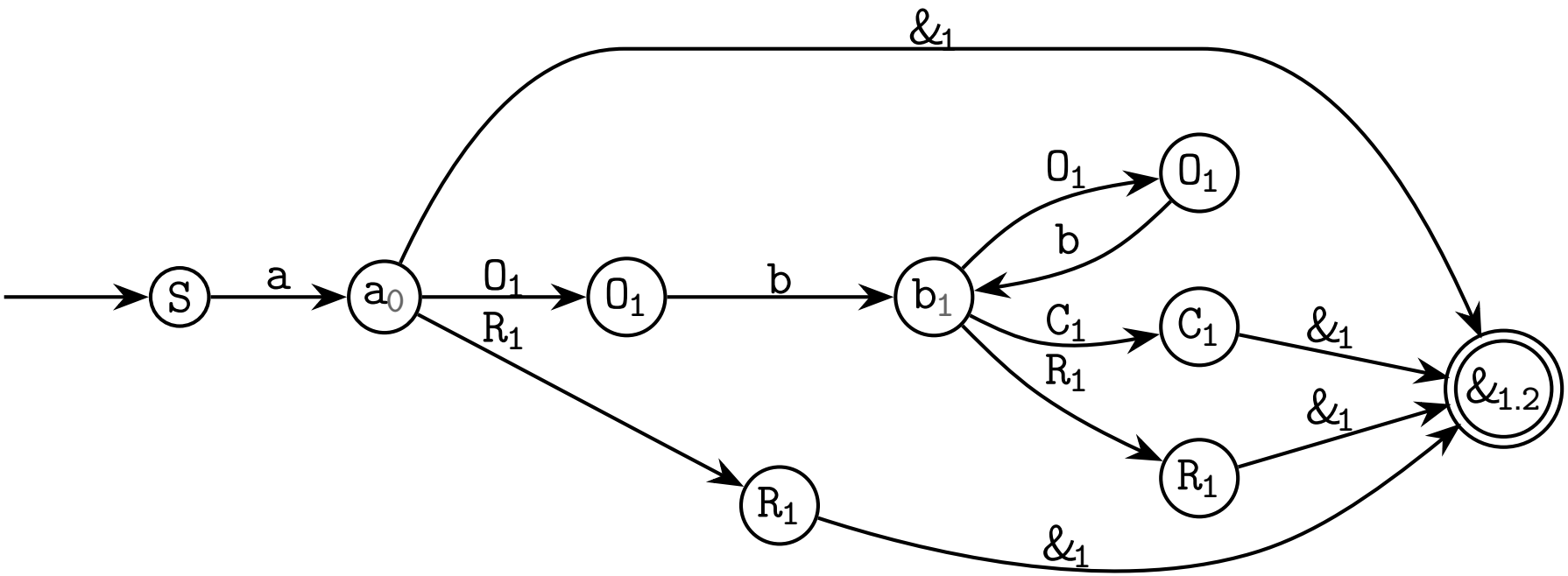


# Аппроксимации конечными автоматами

Symbolic-NFA — $\pi^{\mathcal{M}}(\mathscr{A})$:

- Операции над памятью считаются входными символами;
- Symbolic-бисимуляции **достаточно** для бисимуляции *MFA*.

Регулярное выражение: $\mathtt{a}[\mathtt{b}\,|] : 1^{*}\&1$



# Экспериментальный алгоритм

## Сравнение содержимого ячеек памяти

- Решение уравнений в словах. Простой пример:
  Для автоматов по $[\mathtt{aa}^*] : \mathtt{1a\&1}$ и $\mathtt{a}[\mathtt{a}^*\mathtt{a}] : \mathtt{1\&1}$ значения ссылок $\&1_{\mathscr{A}_1}$ и $\&1_{\mathscr{A}_2}$ всегда совпадают, так как $\forall \omega = \mathtt{a}^k$, $\mathtt{a}\omega = \omega\mathtt{a}$;
- Сравнение языков *NFA*.

Убедиться, что решающие состояния `Action`-бисимилярны и между ними отсутствуют перекрестные связи.

↓

Найти `Action`-бисимилярные состояния с входящими в них переходами по ссылкам с равными номерами.

↓

Каждому такому состоянию в одном автомате найти соответствующее по языку групп захвата в другом.

# Как такое можно тестировать?

## Фаззинг

- Параметризованный генератор расширенных регулярных выражений;
- Параметризованный генератор автоматов;
- Сравнительный парсинг строк по двум автоматам.

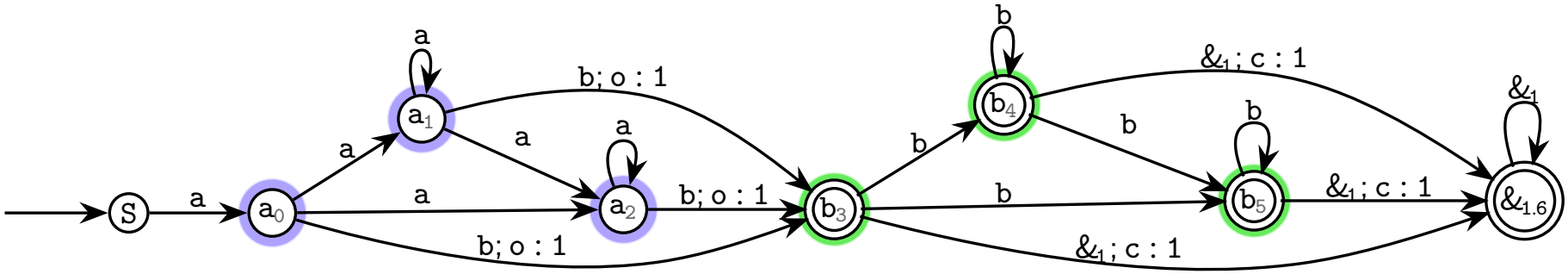
Отношения между аппроксимациями и бисимуляцией:

$$\left(\sim^{\mathcal{M}}\right) \subseteq \left(\sim\right) \subseteq \left(\sim_{\mathcal{M}}\right)$$

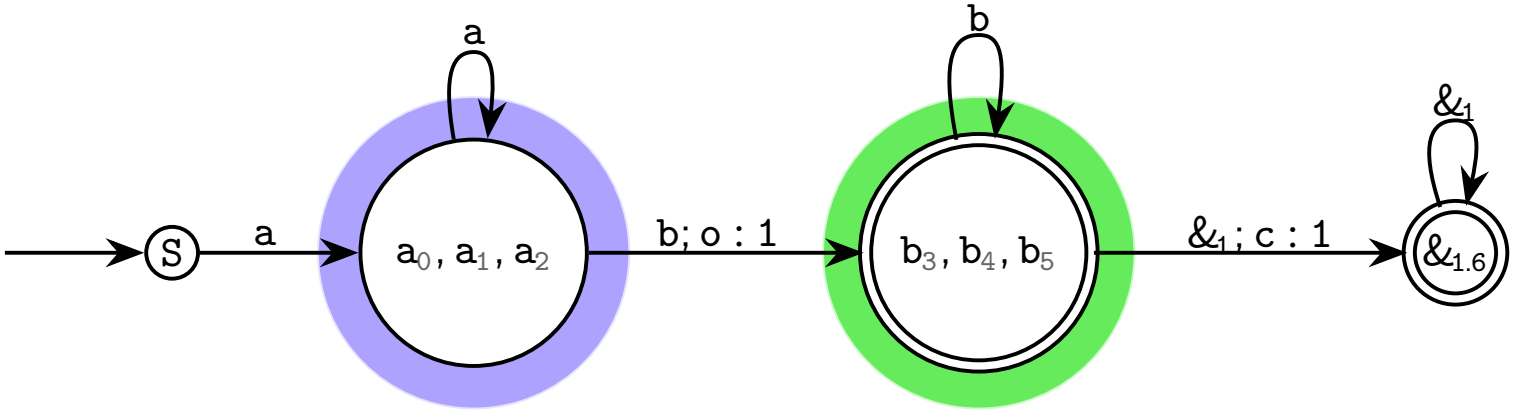
1. Сгенерировать $\mathscr{A}_1$ и $\mathscr{A}_2$;
2. Вычислить $\mathscr{A}_1 \sim \mathscr{A}_2$, $\pi_{\mathcal{M}}(\mathscr{A}_1) \sim \pi_{\mathcal{M}}(\mathscr{A}_2)$, $\pi^{\mathcal{M}}(\mathscr{A}_1) \sim \pi^{\mathcal{M}}(\mathscr{A}_2)$.
3. Если $\left(\pi_{\mathcal{M}}(\mathscr{A}_1) \sim \pi_{\mathcal{M}}(\mathscr{A}_2)\right) \neq \left(\pi^{\mathcal{M}}(\mathscr{A}_1) \sim \pi^{\mathcal{M}}(\mathscr{A}_2)\right)$, проверить вручную экспериментальный результат $\sim$.

# Пример слияния по бисимуляции

Исходный автомат:




Автомат после слияния:



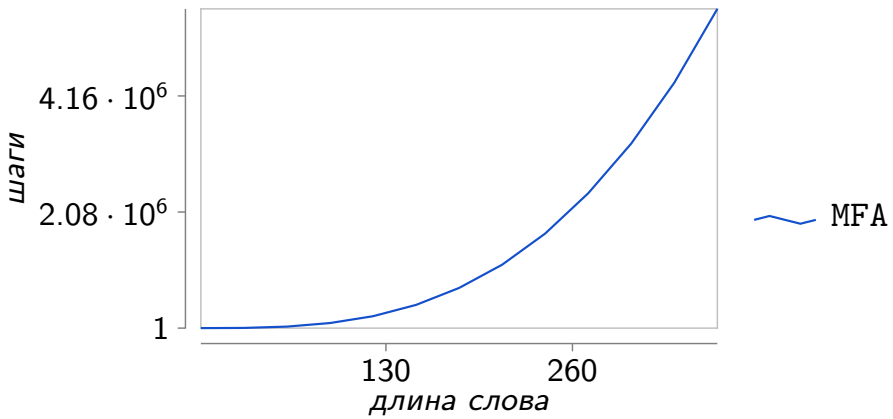

Классы эквивалентности по бисимуляции:
$\{\&_{1.6}\}; \{b_3,\ b_4,\ b_5\}; \{a_0,\ a_1,\ a_2\}; \{S\};$

# Разбор слов по *MFA*: исходный автомат

Слова порождаются выражением a*(bb)*c

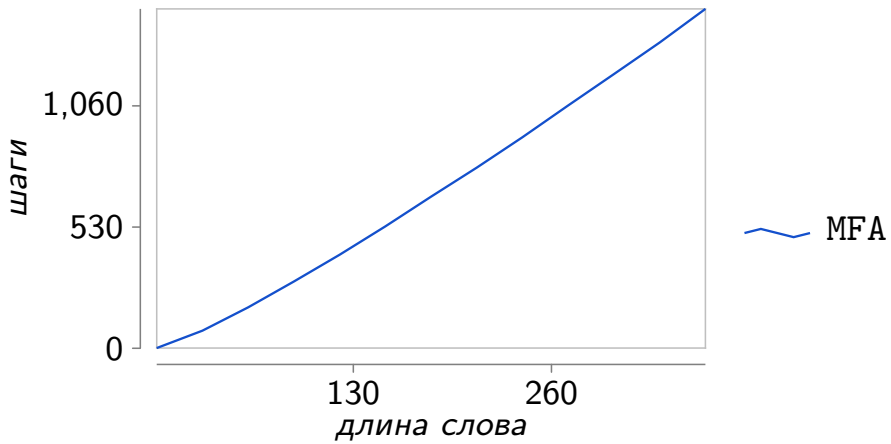
Шаг итерации: 10

| | Шаги | Длина строки | Время парсинга | Принадлежность языку |
|---|---|---|---|---|
| 1 | 1 | 1 | 0.000000 | false |
| 2 | 3446 | 31 | 0.033000 | false |
| 3 | 27051 | 61 | 0.266000 | false |
| 4 | 90886 | 91 | 0.885000 | false |
| 5 | 213381 | 121 | 2.000000 | false |
| 6 | 416226 | 151 | 3.893000 | false |
| 7 | 720211 | 181 | 6.786000 | false |
| 8 | 1136976 | 211 | 10.619000 | false |
| 9 | 1695561 | 241 | 15.859000 | false |
| 10 | 2417896 | 271 | 22.886000 | false |
| 11 | 3309451 | 301 | 31.140000 | false |
| 12 | 4399946 | 331 | 41.744000 | false |
| 13 | 5723221 | 361 | 54.087000 | false |



# Разбор слов по *MFA*: оптимизированный автомат

| | Шаги | Длина строки | Время парсинга | Принадлежность языку |
|---|---|---|---|---|
| 1 | 1 | 1 | 0.000000 | false |
| 2 | 77 | 31 | 0.000000 | false |
| 3 | 179 | 61 | 0.001000 | false |
| 4 | 292 | 91 | 0.002000 | false |
| 5 | 409 | 121 | 0.003000 | false |
| 6 | 533 | 151 | 0.004000 | false |
| 7 | 663 | 181 | 0.006000 | false |
| 8 | 790 | 211 | 0.007000 | false |
| 9 | 923 | 241 | 0.008000 | false |
| 10 | 1062 | 271 | 0.009000 | false |
| 11 | 1199 | 301 | 0.010000 | false |
| 12 | 1337 | 331 | 0.012000 | false |
| 13 | 1484 | 361 | 0.013000 | false |



# Спасибо за внимание!