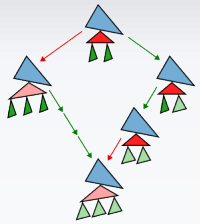
Теорема Клини. Окончание. Минимизация ДКА

Теория формальных языков

2023 г.



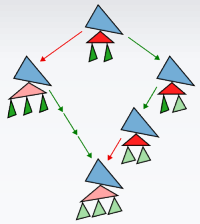
Недетерминированные КА

Определение

Недетерминированный конечный автомат (NFA) — это пятёрка $\mathscr{A} = \langle Q, \Sigma, q_0, F, \delta \rangle$, где:

- Q — множество состояний;
- Σ — алфавит терминалов;
- δ — множество правил перехода вида $\langle q_i, (a_i|\varepsilon), M_i \rangle$, где $q_i \in Q$, $a_i \in \Sigma$, $M_i \in 2^Q$;
- $q_0 \in Q$ — начальное состояние;
- $F \subseteq Q$ — множество конечных состояний.

Сокращаем: $\langle q_1, a, q_2 \rangle \in \delta \Leftrightarrow \langle q_1, a, M \rangle \in \delta \ \& \ q_2 \in M$.

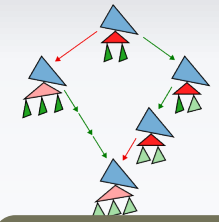


Недетерминированные КА

Определение

Недетерминированный конечный автомат (NFA) — это пятёрка $\mathscr{A} = \langle Q, \Sigma, q_0, F, \delta \rangle$.

- $q \xrightarrow{\varepsilon} q' \Leftrightarrow (q = q') \vee \exists p_1, \ldots, p_k (\langle q, \varepsilon, p_1 \rangle \in \delta \;\&\; \langle p_k, \varepsilon, q' \rangle \in \delta \;\&\; \forall i, 1 \leqslant i < k \langle p_i, \varepsilon, p_{i+1} \rangle \in \delta)$.
- $q \xrightarrow{a} q' \Leftrightarrow \exists p, p' (q \xrightarrow{\varepsilon} p \;\&\; \langle p, a, p' \rangle \in \delta \;\&\; p' \xrightarrow{\varepsilon} q')$.
- $q \xrightarrow{a_1 \ldots a_k} q' \Leftrightarrow \exists p_1, \ldots, p_{k-1} (q \xrightarrow{a_1} p_1 \;\&\; p_{k-1} \xrightarrow{a_k} q' \;\&\; \forall i, 1 \leqslant i < k-1 (p_i \xrightarrow{a_{i+1}} p_{i+1}))$.



Недетерминированные КА

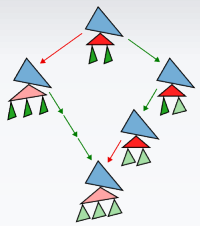
Определение

Недетерминированный конечный автомат (NFA) — это пятёрка $\mathscr{A} = \langle Q, \Sigma, q_0, F, \delta \rangle$.

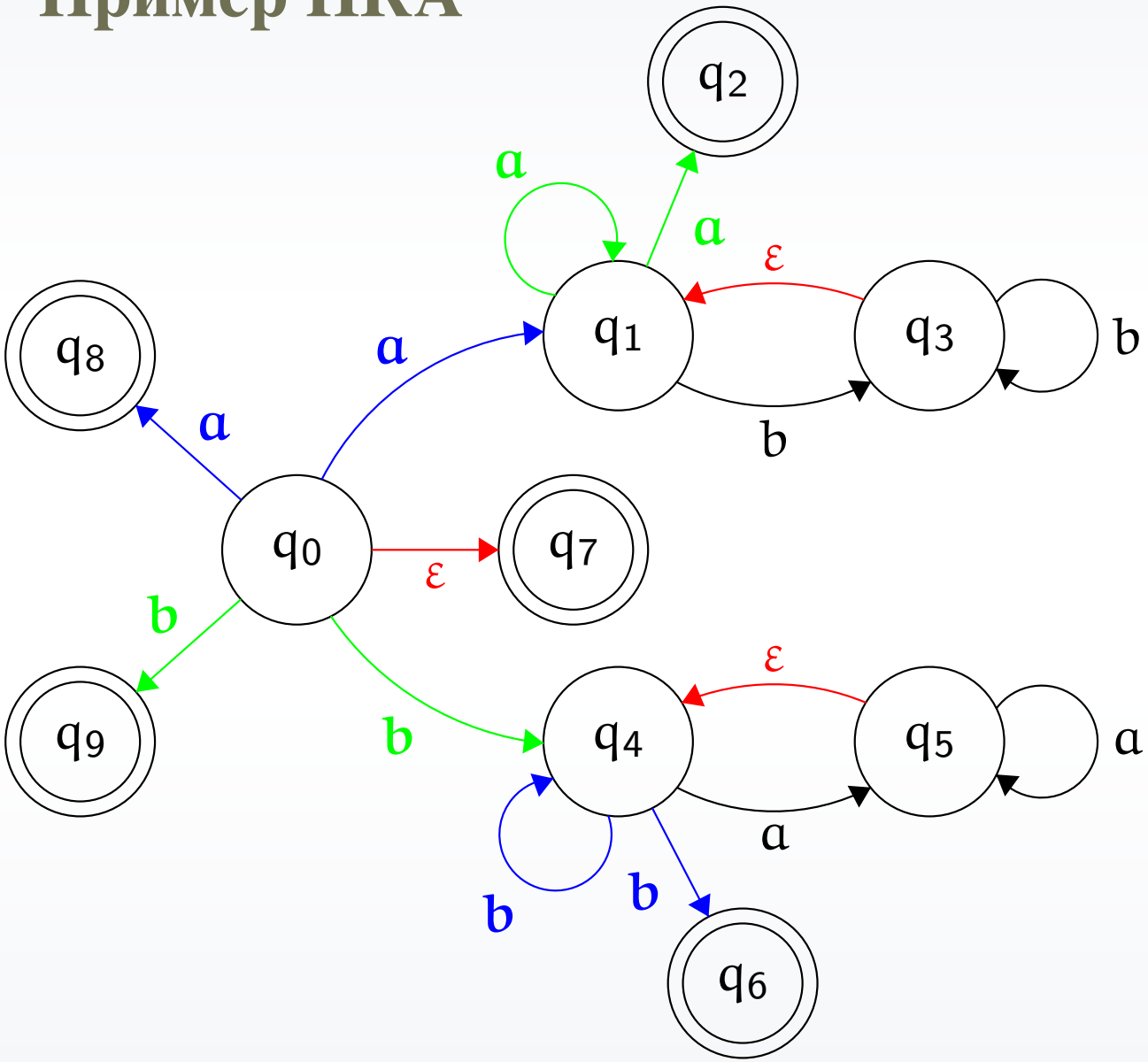
- $q \xrightarrow{\varepsilon} q' \Leftrightarrow (q = q') \vee \exists p_1, \ldots, p_k (\langle q, \varepsilon, p_1 \rangle \in \delta \;\&\; \langle p_k, \varepsilon, q' \rangle \in \delta \;\&\; \forall i, 1 \leqslant i < k \langle p_i, \varepsilon, p_{i+1} \rangle \in \delta)$.
- $q \xrightarrow{a} q' \Leftrightarrow \exists p, p' (q \xrightarrow{\varepsilon} p \;\&\; \langle p, a, p' \rangle \in \delta \;\&\; p' \xrightarrow{\varepsilon} q')$.
- $q \xrightarrow{a_1 \ldots a_k} q' \Leftrightarrow \exists p_1, \ldots, p_{k-1} (q \xrightarrow{a_1} p_1 \;\&\; p_{k-1} \xrightarrow{a_k} q' \;\&\; \forall i, 1 \leqslant i < k-1 (p_i \xrightarrow{a_{i+1}} p_{i+1}))$.

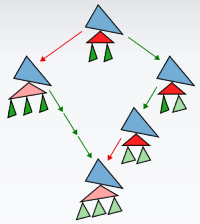
Определение

Язык $\mathscr{L}$, распознаваемый НКА $\mathscr{A}$ — это множество слов $\{w \mid \exists q \in F (q_0 \xrightarrow{w} q)\}$.



Пример НКА





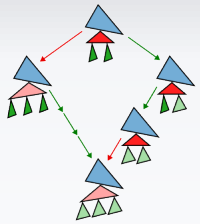
Детерминированный КА

Определение

Детерминированный конечный автомат (DFA) — это пятёрка $\mathscr{A} = \langle Q, \Sigma, q_0, F, \delta \rangle$, где:

- Q — множество состояний;
- Σ — алфавит терминалов;
- δ — множество правил перехода вида $\langle q_i, a_i, q_j \rangle$, где $q_i, q_j \in Q$, $a_i \in \Sigma$, причём $\forall q_i, a_i \exists q_j (\langle q_i, a_i, q_j \rangle \in \delta \;\&\; \forall q_k (\langle q_i, a_i, q_k \rangle \in \delta \Rightarrow q_k = q_j))$;
- $q_0 \in Q$ — начальное состояние;
- $F \subseteq Q$ — множество конечных состояний.

ε-переходов нет $\Rightarrow q \xrightarrow{a} q' \Leftrightarrow \langle q, a, q' \rangle \in \delta$.



Детерминированный КА

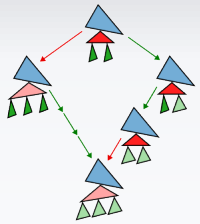
Определение

Детерминированный конечный автомат (DFA) — это пятёрка $\mathscr{A} = \langle Q, \Sigma, q_0, F, \delta \rangle$, где:

- Q — множество состояний;
- Σ — алфавит терминалов;
- δ — множество правил перехода вида $\langle q_i, a_i, q_j \rangle$, где $q_i, q_j \in Q$, $a_i \in \Sigma$, причём $\forall q_i, a_i \exists q_j (\langle q_i, a_i, q_j \rangle \in \delta \;\&\; \forall q_k (\langle q_i, a_i, q_k \rangle \in \delta \Rightarrow q_k = q_j))$;
- $q_0 \in Q$ — начальное состояние;
- $F \subseteq Q$ — множество конечных состояний.

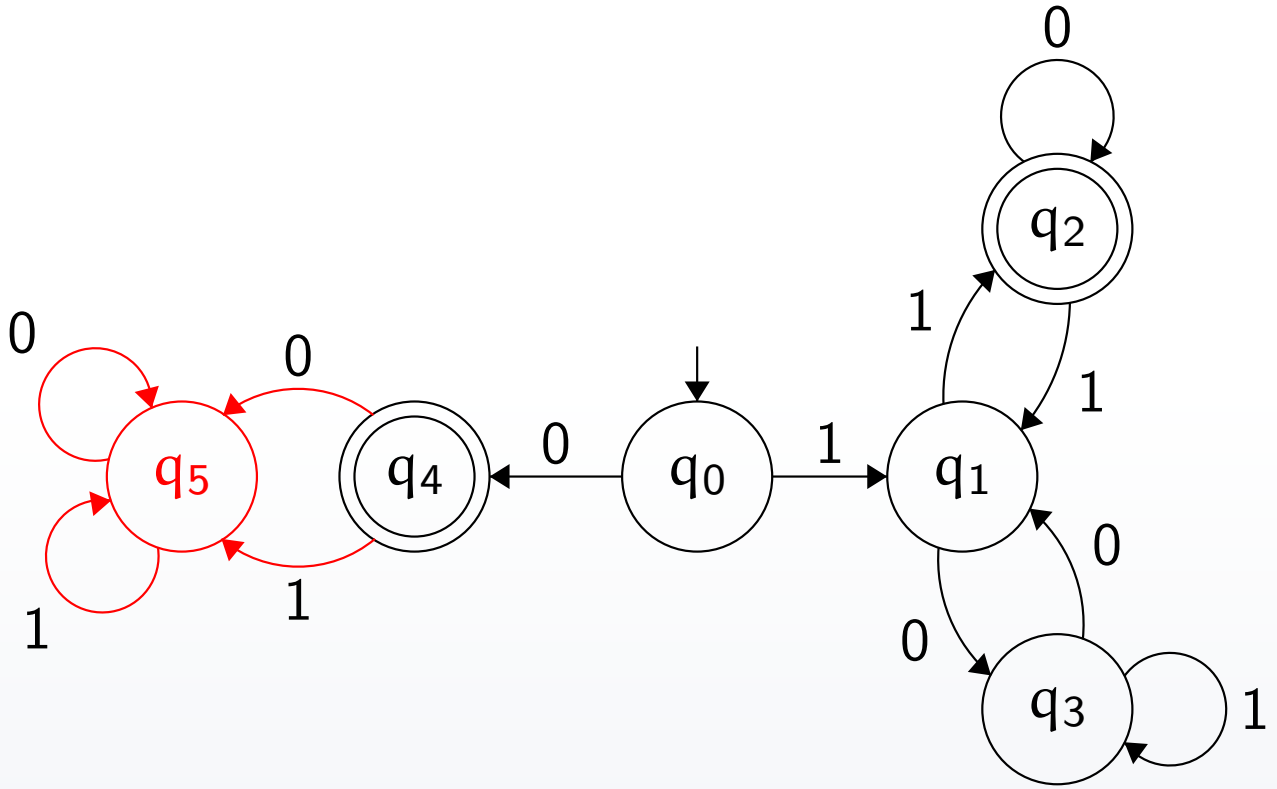
ε-переходов нет $\Rightarrow q \xrightarrow{a} q' \Leftrightarrow \langle q, a, q' \rangle \in \delta$.

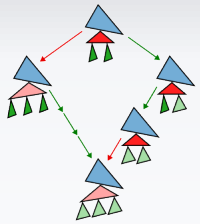
Язык $\mathscr{L}$, распознаваемый $\mathscr{A}$ — это множество слов $\{w \mid \exists q \in F (q_0 \xrightarrow{w} q)\}$.



Sink/trap state (состояние–ловушка)

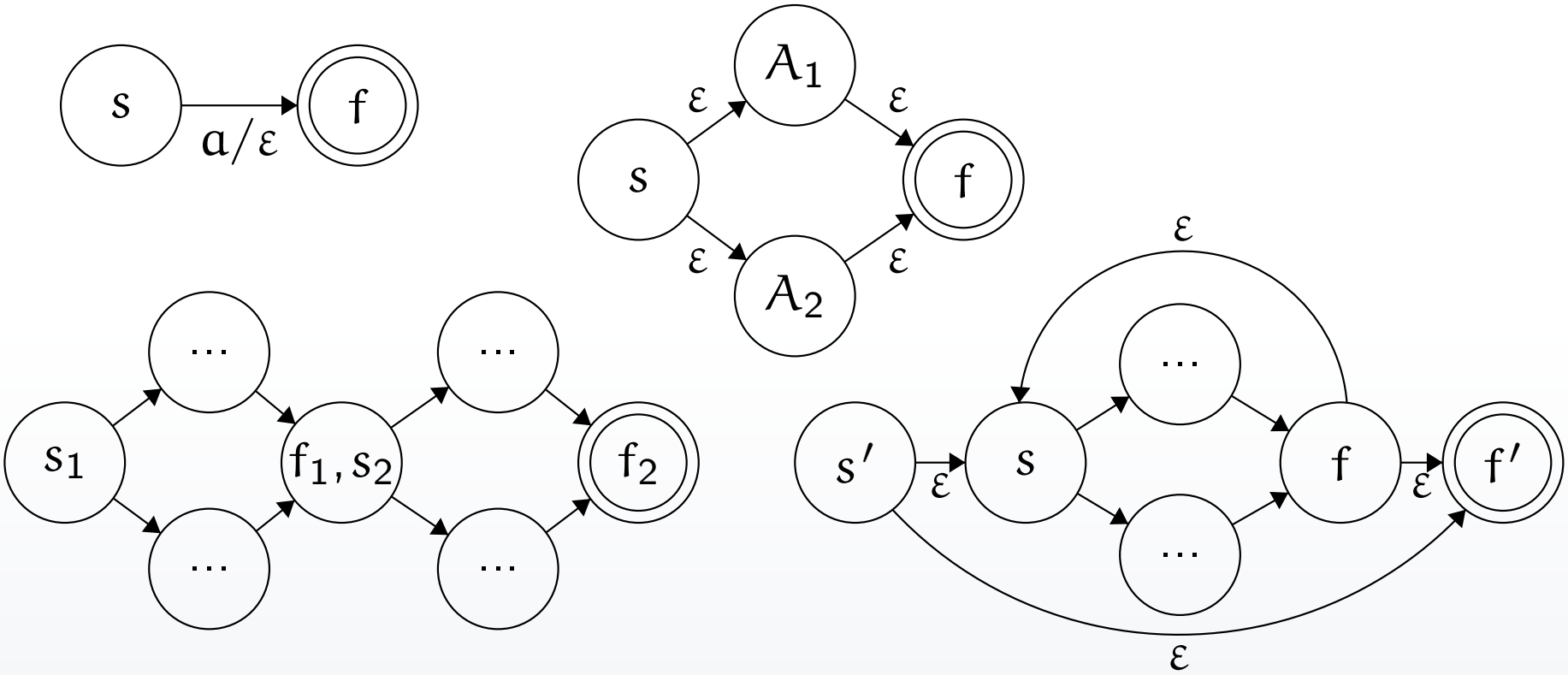
«Ловушка» — не конечное состояние с переходами лишь в себя. Нужны для корректного задания ДКА, но иногда по умолчанию не описываются.

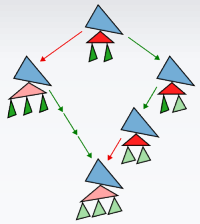




Автомат Томпсона

- Единственное начальное состояние
- Единственное конечное состояние
- Не больше двух переходов из каждого состояния



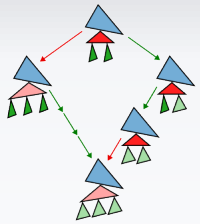


Матрица НКА

НКА можно описать посредством трёх матриц: вектор начальных состояний (в классическом определении оно одно); матрица переходов между состояниями; и вектор конечных состояний. Если принять первое состояние НКА стартовым, и допустить отсутствие переходов в стартовое состояние, то НКА представится следующей матрицей.

$$\left\langle \begin{pmatrix} 1 & \dots 0 \end{pmatrix} \begin{pmatrix} 0 & J_1 & \dots & J_n \\ 0 & F_{1,1} & \dots & F_{1,n} \\ & & \dots & \\ 0 & F_{n,1} & \dots & F_{n,n} \end{pmatrix} \begin{pmatrix} c \\ U_1 \\ U_2 \\ \dots \\ U_n \end{pmatrix} \right\rangle$$

Тогда естественно строить объединение, конкатенацию и итерацию НКА посредством матричных операций.

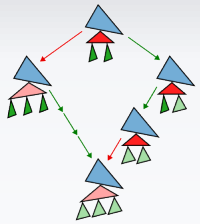


Матрица НКА

$$A = \left\langle (1 \dots 0) \quad \begin{pmatrix} 0 & J_1 & \dots & J_n \\ 0 & F_{1,1} & \dots & F_{1,n} \\ & & \dots & \\ 0 & F_{n,1} & \dots & F_{n,n} \end{pmatrix} \quad \begin{pmatrix} c \\ u_1 \\ \dots \\ u_n \end{pmatrix} \right\rangle$$

$$B = \left\langle (1 \dots 0) \quad \begin{pmatrix} 0 & K_1 & \dots & K_m \\ 0 & G_{1,1} & \dots & G_{1,m} \\ & & \dots & \\ 0 & G_{m,1} & \dots & G_{m,m} \end{pmatrix} \quad \begin{pmatrix} d \\ v_1 \\ \dots \\ v_m \end{pmatrix} \right\rangle$$

$$A \mid B = \left\langle (1 \dots 0 \dots 0) \quad \begin{pmatrix} 0 & J_1 \dots & J_n & K_1 & \dots K_m \\ 0 & F_{1,1} \dots & F_{1,n} & 0 & \dots 0 \\ & \dots & & & \\ & \dots & & & \\ 0 & F_{n,1} \dots & F_{n,n} & 0 & \dots 0 \\ 0 & 0 \dots & 0 & G_{1,1} & \dots G_{1,m} \\ 0 & 0 \dots & 0 & & \dots \\ 0 & 0 \dots & 0 & G_{m,1} & \dots G_{m,m} \end{pmatrix} \quad \begin{pmatrix} c \mid d \\ u_1 \\ \dots \\ u_n \\ v_1 \\ \dots \\ v_m \end{pmatrix} \right\rangle$$

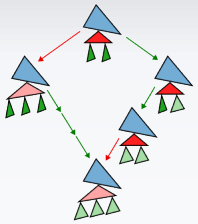


Матрица НКА

$$A = \left\langle (1 \dots 0) \quad \begin{pmatrix} 0 & J_1 & \dots & J_n \\ 0 & F_{1,1} & \dots & F_{1,n} \\ & & \dots & \\ 0 & F_{n,1} & \dots & F_{n,n} \end{pmatrix} \quad \begin{pmatrix} c \\ u_1 \\ \dots \\ u_n \end{pmatrix} \right\rangle$$

$$B = \left\langle (1 \dots 0) \quad \begin{pmatrix} 0 & K_1 & \dots & K_m \\ 0 & G_{1,1} & \dots & G_{1,m} \\ & & \dots & \\ 0 & G_{m,1} & \dots & G_{m,m} \end{pmatrix} \quad \begin{pmatrix} d \\ v_1 \\ \dots \\ v_m \end{pmatrix} \right\rangle$$

$$A \cdot B = \left\langle (1 \dots 0 \dots 0) \quad \begin{pmatrix} 0 & J_1 \dots & J_n & c \cdot K_1 & \dots c \cdot K_m \\ 0 & F_{1,1} \dots & F_{1,n} & u_1 \cdot K_1 & \dots u_1 \cdot K_m \\ & \dots & & & \\ & \dots & & & \\ 0 & F_{n,1} \dots & F_{n,n} & u_n \cdot K_1 & \dots u_n \cdot K_m \\ 0 & 0 \dots & 0 & G_{1,1} & \dots G_{1,m} \\ 0 & 0 \dots & 0 & & \dots \\ 0 & 0 \dots & 0 & G_{m,1} & \dots G_{m,m} \end{pmatrix} \quad \begin{pmatrix} c \cdot d \\ u_1 \cdot d \\ \dots \\ u_n \cdot d \\ v_1 \\ \dots \\ v_m \end{pmatrix} \right\rangle$$



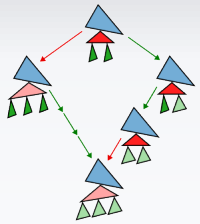
Матрица НКА

$$A = \left\langle (1\ldots0) \begin{pmatrix} 0 & J_1 & \ldots & J_n \\ 0 & F_{1,1} & \ldots & F_{1,n} \\ & & \ldots & \\ 0 & F_{n,1} & \ldots & F_{n,n} \end{pmatrix} \begin{pmatrix} c \\ u_1 \\ \ldots \\ u_n \end{pmatrix} \right\rangle$$

$$B = \left\langle (1\ldots0) \begin{pmatrix} 0 & K_1 & \ldots & K_m \\ 0 & G_{1,1} & \ldots & G_{1,m} \\ & & \ldots & \\ 0 & G_{m,1} & \ldots & G_{m,m} \end{pmatrix} \begin{pmatrix} d \\ v_1 \\ \ldots \\ v_m \end{pmatrix} \right\rangle$$

$$A^* = \left\langle (1\ldots0) \begin{pmatrix} 0 & J_1 & \ldots & J_n \\ 0 & H_{1,1} & \ldots & H_{1,n} \\ & & \ldots & \\ 0 & H_{n,1} & \ldots & H_{n,n} \end{pmatrix} \begin{pmatrix} 1 \\ u_1 \\ \ldots \\ u_n \end{pmatrix} \right\rangle$$

Здесь $H_{i,j} = (U_i \cdot J_j) \mid F_{i,j}$. Таким образом, можно определить автомат для всех стандартных операций над базовыми автоматами, и конструкция рекурсивно распространяется на все регулярные выражения.



Линеаризация

Определение

Если регулярное выражение $\mathtt{r} \in \mathcal{RE}$ содержит $\mathtt{n}$ вхождений букв алфавита Σ, тогда линеаризованное регулярное выражение $\mathtt{Linearize}(\mathtt{r})$ получается из $\mathtt{r}$ приписыванием $\mathtt{i}$-ой по счёту букве, входящей в $\mathtt{r}$, индекса $\mathtt{i}$.

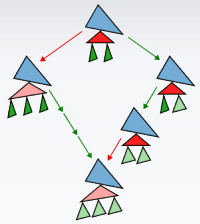
Пример

Рассмотрим регулярное выражение:

$$(\mathtt{ba} \mid \mathtt{b})\mathtt{aa}(\mathtt{a} \mid \mathtt{ab})^*$$

Его линеаризованная версия:

$$(\mathtt{b}_1\mathtt{a}_2 \mid \mathtt{b}_3)\mathtt{a}_4\mathtt{a}_5(\mathtt{a}_6 \mid \mathtt{a}_7\mathtt{b}_8)^*$$



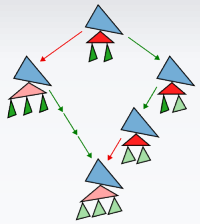
Множества `First`, `Last`, `Follow`

Определение

Пусть $\mathtt{r} \in \mathcal{RE}$, тогда:

- множество `First` — это множество букв, с которых может начинаться слово из $\mathscr{L}(\mathtt{r})$ (если $\varepsilon \in \mathscr{L}(\mathtt{r})$, то оно формально добавляется в `First`);
- множество `Last` — это множество букв, которыми может заканчиваться слово из $\mathscr{L}(\mathtt{r})$;
- множество $\mathtt{Follow}(\mathtt{c})$ — это множество букв, которым может предшествовать c. Т.е. $\left\{\mathtt{d} \in \Sigma \mid \exists w_1, w_2 (w_1 \mathtt{c} \mathtt{d} w_2 \in \mathscr{L}(\mathtt{r}))\right\}$.

Множество `Follow` в теории компиляции обычно определяется иначе — это множество символов, которые могут идти за выводом из определённого нетерминального символа. Два этих определения можно унифицировать, если рассматривать каждую букву в r как «обёрнутую» (в смысле, например, н.ф. Хомского).



First, Last, Follow — пример

Построим указанные множества для регулярного выражения $\mathtt{r} = (\mathtt{ba} \mid \mathtt{b})\mathtt{aa}(\mathtt{a} \mid \mathtt{ab})^*$.

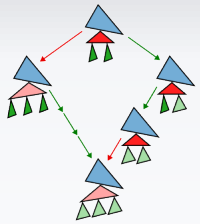
Начнём с исходного регулярного выражения.

Исходное регулярное выражение

- $\mathtt{First}(\mathtt{r}) = \{\mathtt{b}\}$.
- $\mathtt{Last}(\mathtt{r}) = \{\mathtt{a}, \mathtt{b}\}$.
- $\mathtt{Follow}_{\mathtt{r}}(\mathtt{a}) = \{\mathtt{a}, \mathtt{b}\}$; $\mathtt{Follow}_{\mathtt{r}}(\mathtt{b}) = \{\mathtt{a}\}$.

Хотя данные множества описывают, как устроены слова из $\mathscr{L}(\mathtt{r})$ локально, однако они не исчерпывают всей информации о языке, поскольку разные вхождения букв в регулярное выражения никак не различаются.

Например, по множествам First и Last можно предположить, что $\mathtt{b} \in \mathscr{L}(\mathtt{r})$, хотя это не так.



First, Last, Follow — пример

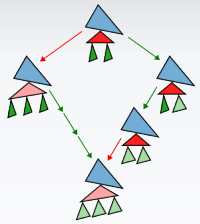
Построим указанные множества для регулярного выражения $r = (ba \mid b)aa(a \mid ab)^*$.

Вспомним, что $r_{Lin} = (b_1a_2 \mid b_3)a_4a_5(a_6 \mid a_7b_8)^*$.

Линеаризованное выражение

- $First(r_{Lin}) = \{b_1, b_3\}$.
- $Last(r_{Lin}) = \{a_5, a_6, b_8\}$.
- $Follow_{r_{Lin}}(b_1) = \{a_2\}$; $Follow_{r_{Lin}}(a_2) = \{a_4\}$;
  $Follow_{r_{Lin}}(b_3) = \{a_4\}$; $Follow_{r_{Lin}}(a_4) = \{a_5\}$;
  $Follow_{r_{Lin}}(a_5) = \{a_6, a_7\}$; $Follow_{r_{Lin}}(a_6) = \{a_6, a_7\}$;
  $Follow_{r_{Lin}}(a_7) = \{b_8\}$; $Follow_{r_{Lin}}(b_8) = \{a_6, a_7\}$.

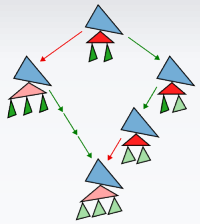
В описании данных множеств содержится исчерпывающая информация о языке $\mathscr{L}(r_{Lin})$.



Конструкция автомата Глушкова

Алгоритм построения $\texttt{Glushkov}(r)$

- Строим линеаризованную версию r: $r_{\mathsf{Lin}} = \texttt{Linearize}(r)$.
- Ищем $\texttt{First}(r_{\mathsf{Lin}})$, $\texttt{Last}(r_{\mathsf{Lin}})$ и $\texttt{Follow}_{r_{\mathsf{Lin}}}(c)$ для всех $c \in \Sigma_{r_{\mathsf{Lin}}}$.
- Все состояния автомата, кроме начального (назовём его S), соответствуют буквам $c \in \Sigma_{r_{\mathsf{Lin}}}$.
- Из начального состояния строим переходы в те состояния, для которых $c \in \texttt{First}(r_{\mathsf{Lin}})$. Переходы имеют вид $S \xrightarrow{c} c$.
- Переходы из состояния c соответствуют элементам d множества $\texttt{Follow}_{r_{\mathsf{Lin}}}(c)$ и имеют вид $c \xrightarrow{d} d$.
- Конечные состояния — такие, что $c \in \texttt{Last}(r_{\mathsf{Lin}})$, а также S, если $\varepsilon \in \mathscr{L}(R)$.
- Теперь стираем разметку, построенную линеаризацией, на переходах автомата. Конструкция завершена.



Пример автомата Глушкова

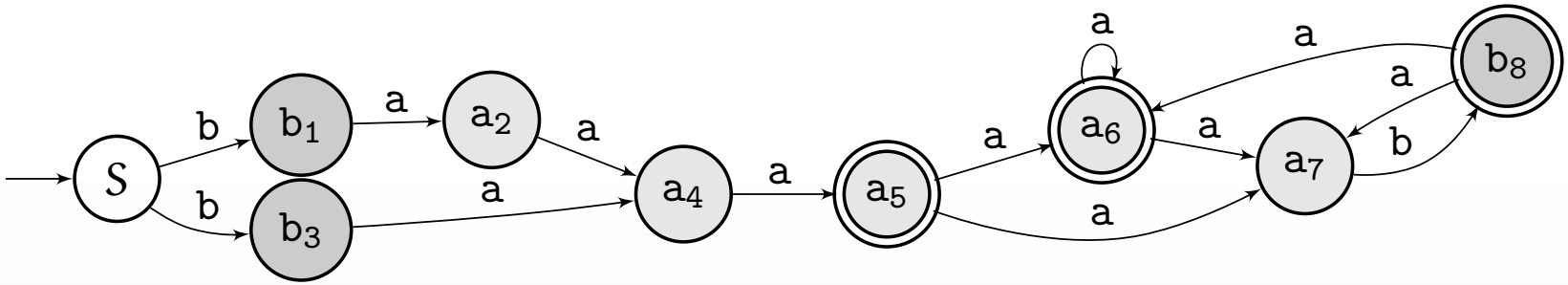
Исходное регулярное выражение:

$$(\mathtt{ba} \mid \mathtt{b})\mathtt{aa}(\mathtt{a} \mid \mathtt{ab})^*$$

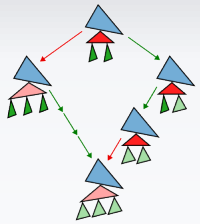
Линеаризованное регулярное выражение:

$$(\mathtt{b}_1\mathtt{a}_2 \mid \mathtt{b}_3)\mathtt{a}_4\mathtt{a}_5(\mathtt{a}_6 \mid \mathtt{a}_7\mathtt{b}_8)^*$$

Автомат Глушкова:



Подграфы, распознающие регулярные выражения, являющиеся подструктурами исходного, не имеют общих вершин. Это свойство автомата Глушкова используется в реализациях `match`-функций некоторых библиотек регулярных выражений.



Свойства автомата Глушкова

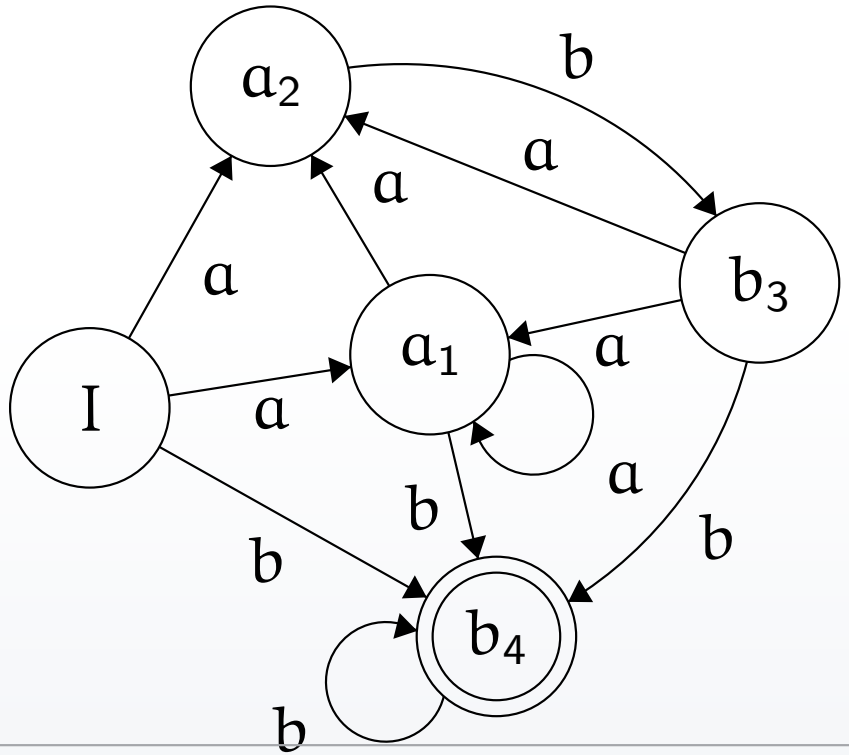
- Не содержит ε-переходов.
- Число состояний равно длине регулярного выражения (без учёта регулярных операций), плюс один (стартовое состояние).
- В общем случае недетерминированный.

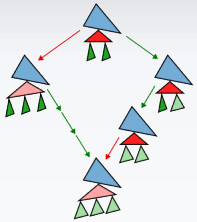
Примечание

Для 1-однозначных регулярных выражений r автомат $\texttt{Glushkov}(r)$ является детерминированным. Эту его особенность активно используют в современных библиотеках регулярных выражений, например, в RE2. Выигрыш может получиться колоссальным: например, $\texttt{Thompson}((a^*)^*)$ является экспоненциально неоднозначным, а $\texttt{Glushkov}((a^*)^*)$ однозначен и детерминирован!

Построим НКА, распознающий $(a \mid (ab))^* b^+$.

- Линеаризуем: $E' = (a_1 \mid (a_2 b_3))^* b_4^+$.
- Порождаем множества First, Last, Next:
  $\mathsf{First}(E') = \{a_1, a_2, b_4\}$
  $\mathsf{Last}(E') = \{b_4\}$
  $\mathsf{Next}(E') = \{a_1 a_1, a_1 a_2, a_2 b_3, b_3 a_1, b_3 a_2, a_1 b_4, b_3 b_4, b_4 b_4\}$
- Строим конечный автомат:

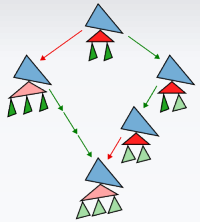




Удаление ε-переходов

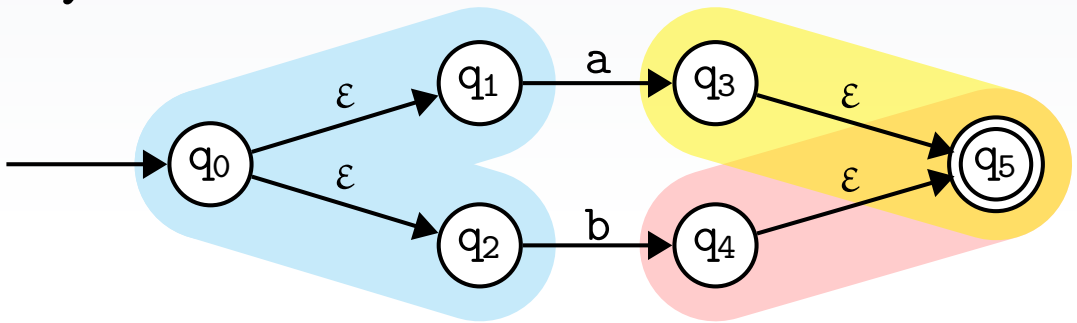
- Один из шагов детерминизации.
- Может пониматься в разных смыслах: удаление только переходов без изменения числа состояний, и построение состояний, замкнутых относительно ε-переходов (то есть аналогично детерминизации, но только по ε-переходам).

Результаты этих преобразований будут различны, причём первое сохраняет недетерминированные переходы, а второе может их детерминизировать.

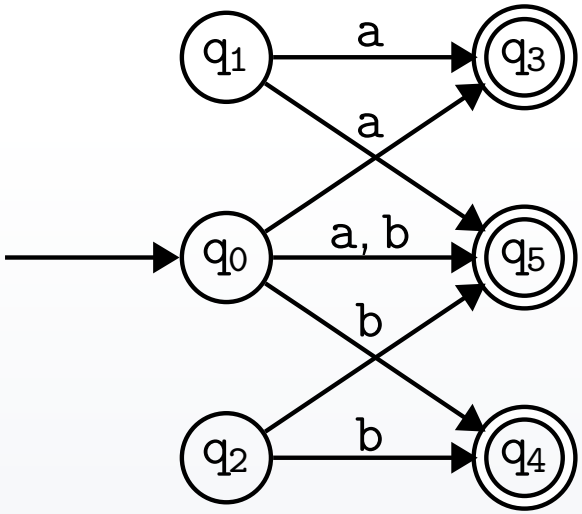


Удаление ε-переходов

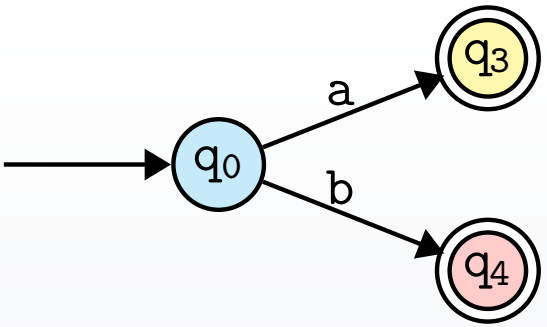
Рассмотрим следующий автомат.

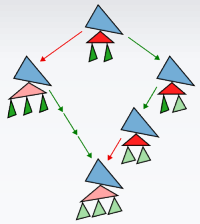


ε-замыкание q_0 — это $\{q_0, q_1, q_2\}$. ε-замыкание q_3 — это $\{q_3, q_5\}$.
ε-замыкание q_4 — это $\{q_4, q_5\}$. Остальные состояния ε-замкнуты собой.
Результат первого преобразования:



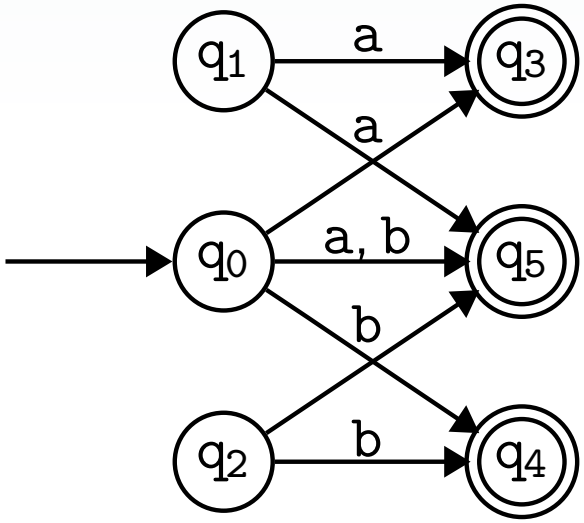
Результат второго преобразования:



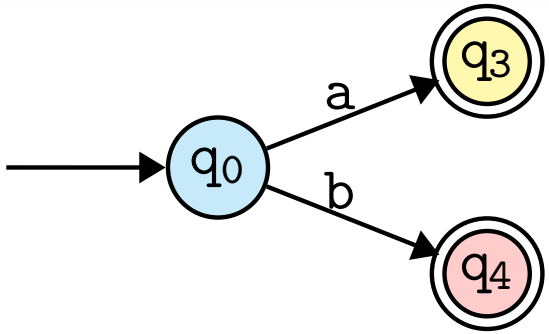


Удаление ε-переходов

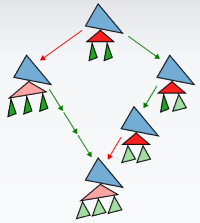
Результат первого преобразования:



Результат второго преобразования:

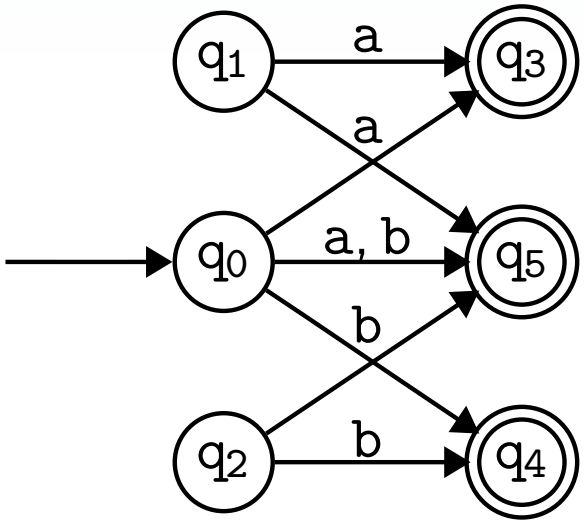


В первом случае q_3 и q_4 стали финальными, потому что из них есть путь по ε-переходам в финальное состояние. Дополнительно добавились переход из q_0 в q_5 по a (поскольку такой путь есть из ε-достижимого из q_0 состояния q_1) и аналогичный переход в q_5 по b. После чего все ε-переходы были удалены. Для завершения построения, следует ещё удалить недостижимые состояния q_1 и q_2. Автомат остался недетерминированным.

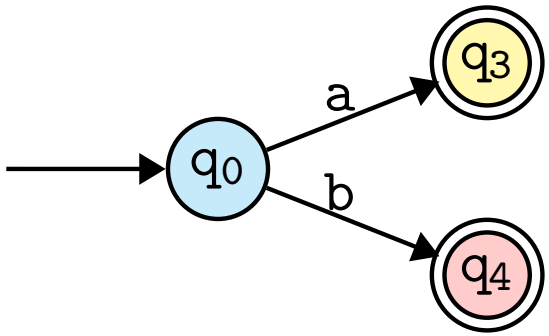


Удаление ε-переходов

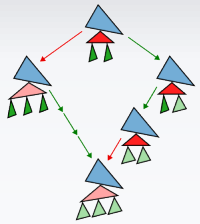
Результат первого преобразования:



Результат второго преобразования:



Во втором случае ε-замыкания состояний исходного автомата сразу же рассматривались как состояния нового автомата. Это привело к тому, что удалось сэкономить одно состояние, и результат оказался детерминированным.



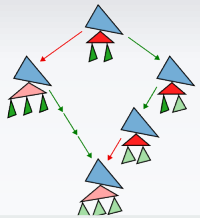
Производные $\mathcal{RE}$

Множество $a^{-1}U = \{w \mid aw \in U\}$ называется производным Бзрозовски множества U относительно a. Если $\varepsilon \in a^{-1}U$, тогда a распознаётся выражением U.

Λ_E положим равным $\{\varepsilon\}$, если $\varepsilon \in E$, и пустым множеством иначе.

- $a^{-1}\varepsilon = \varnothing$, $a^{-1}\varnothing = \varnothing$;
- $a^{-1}a = \{\varepsilon\}$, $a^{-1}b = \varnothing$;
- $a^{-1}(\Phi \mid \Psi) = a^{-1}(\Phi) \cup a^{-1}(\Psi)$;
- $a^{-1}(\Phi\ \Psi) = a^{-1}(\Phi)\Psi \cup \Lambda_\Phi\, a^{-1}(\Psi)$;
- $a^{-1}(\Phi^*) = a^{-1}(\Phi)\Phi^*$.

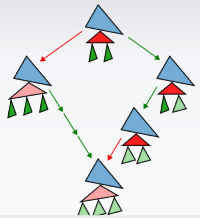
С помощью последовательного взятия производных можно свести задачу $w \in \mathscr{L}(R)$ к задаче $\varepsilon \in w^{-1}R$. На этом построен ещё один способ преобразования $\mathcal{RE}$ к автомату.



Пример преобразования

Рассмотрим всё то же выражение $(a \mid (ab))^* b^+$. Построим по нему автомат с помощью производных Брзозовски.

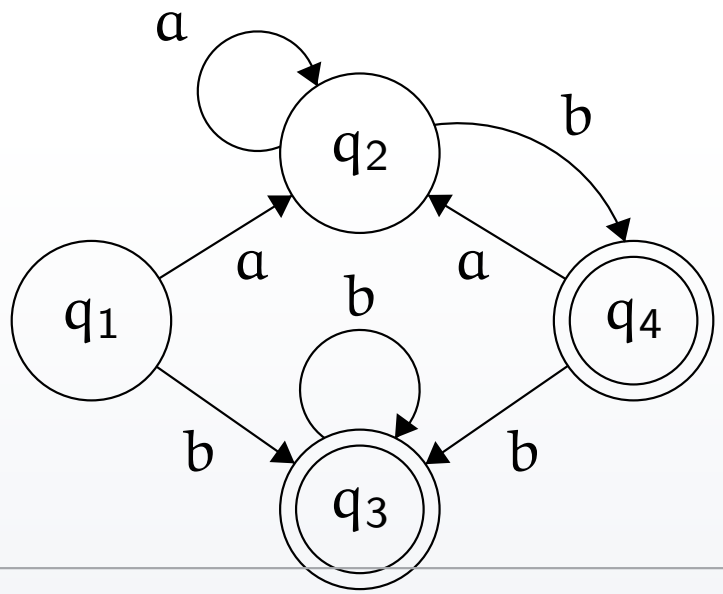
- $a^{-1}(a \mid (ab))^* b^+ = (a^{-1}(a \mid (ab))^*)b^+ \cup (a^{-1}b^+)$, но второе очевидно пусто, поэтому $a^{-1}(a \mid (ab))^* b^+ = (\varepsilon \mid b)\ (a \mid (ab))^* b^+$;
- $b^{-1}(a \mid (ab))^* b^+ = (b^{-1}(a \mid (ab))^*)b^+ \cup (b^{-1}b^+)$, и здесь как раз пусто первое, поэтому производная равна b^*.
- $a^{-1}b^* = \varnothing$; $b^{-1}b^* = b^*$.
- $a^{-1}((\varepsilon \mid b)\ (a \mid (ab))^* b^+)$ вынуждает первую альтернативу в $(\varepsilon|b)$ и порождает само себя.
- $b^{-1}((\varepsilon \mid b)\ (a \mid (ab))^* b^+)$ порождает $(a \mid (ab))^* b^+ \mid b^*$.
- $a^{-1}((a \mid (ab))^* b^+ \mid b^*)$ порождает $(\varepsilon \mid b)\ (a \mid (ab))^* b^+$, $b^{-1}((a \mid (ab))^* b^+ \mid b^*)$ порождает b^*.
- Переходы замкнулись. Осталось собрать производные в состояния автомата.

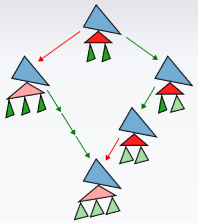


Пример преобразования

Рассмотрим всё то же выражение $(a \mid (ab))^* b^+$. Построим по нему автомат с помощью производных Брзозовски.

Состояние	Производная
q_1	$(a \mid (ab))^* b^+$
q_2	$(\varepsilon \mid b)\ (a \mid (ab))^* b^+$
q_3	b^*
q_4	$(a \mid (ab))^* b^+ \mid b^*$



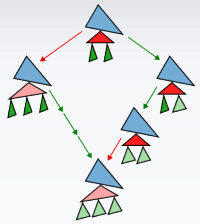


Частичные производные Антимирова

$\alpha_c(R)$ — это регулярное выражение R' такое, что если $w \in \mathscr{L}(R')$, то $c\,w \in \mathscr{L}(R)$. Обратное не обязательно выполняется. Вычислить частичные производные можно по следующему рекурсивному алгоритму.

$$
\begin{array}{lcl}
\alpha_c(c) & = & \{\varepsilon\} \\
\alpha_c(c') & = & \varnothing \\
\alpha_c(\varepsilon) & = & \varnothing \\
\alpha_c(r_1\,r_2) & = & \begin{cases} \{r\,r_2 \mid r \in \alpha_c(r_1)\} \cup \alpha_c(r_2) & \text{если } \varepsilon \in \mathscr{L}(r_1) \\ \{r\,r_2 \mid r \in \alpha_c(r_1)\} & \text{иначе} \end{cases} \\
\alpha_c(\bot) & = & \varnothing \\
\alpha_c(r_1|r_2) & = & \alpha_c(r_1) \cup \alpha_c(r_2) \\
\alpha_c(r*) & = & \{r'r* \mid r' \in \alpha_c(r)\}
\end{array}
$$

Автомат Антимирова аналогичен автомату Брзозовски, но состояния представляют собой элементы α_w, а не δ_w. Упрощать по ACI состояния не требуется — их множество и так конечно.



Пример автомата Антимирова

Положим $R_1 = (ab|b)^*ba$.

Тогда (производные, равные пустому множеству, здесь опущены):

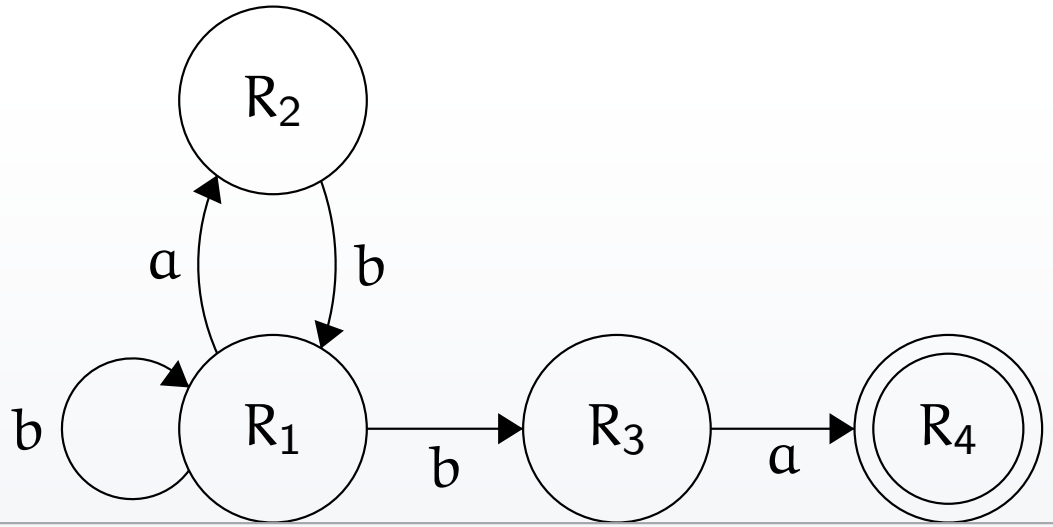
$\alpha_a(R_1) = \{b(ab|b)^*ba\}$ — положим $R_2 = b(ab|b)^*ba$

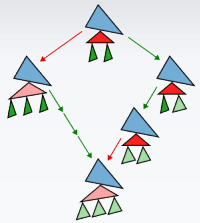
$\alpha_b(R_1) = \{(ab|b)^*ba,\ a\}$ — положим $R_3 = a$

$\alpha_b(R_2) = \{(ab|b)^*ba\}$ — тут ничего нового

$\alpha_a(R_3) = \{\varepsilon\}$ — положим $R_4 = \varepsilon$

Соответствующий автомат имеет состояния R_i и один недетерминированный переход.





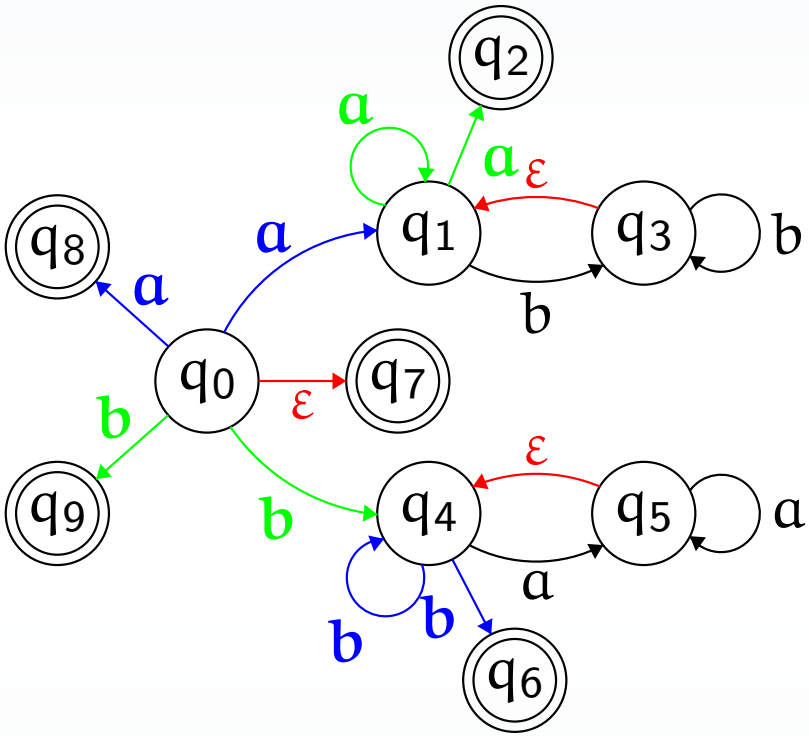
Детерминизация НКА

От $\mathscr{A}$ к $D(\mathscr{A})$

Состояния DFA $D(\mathscr{A})$ — это состояния $m_i \in 2^Q$, где Q — состояния NFA $\mathscr{A}$.

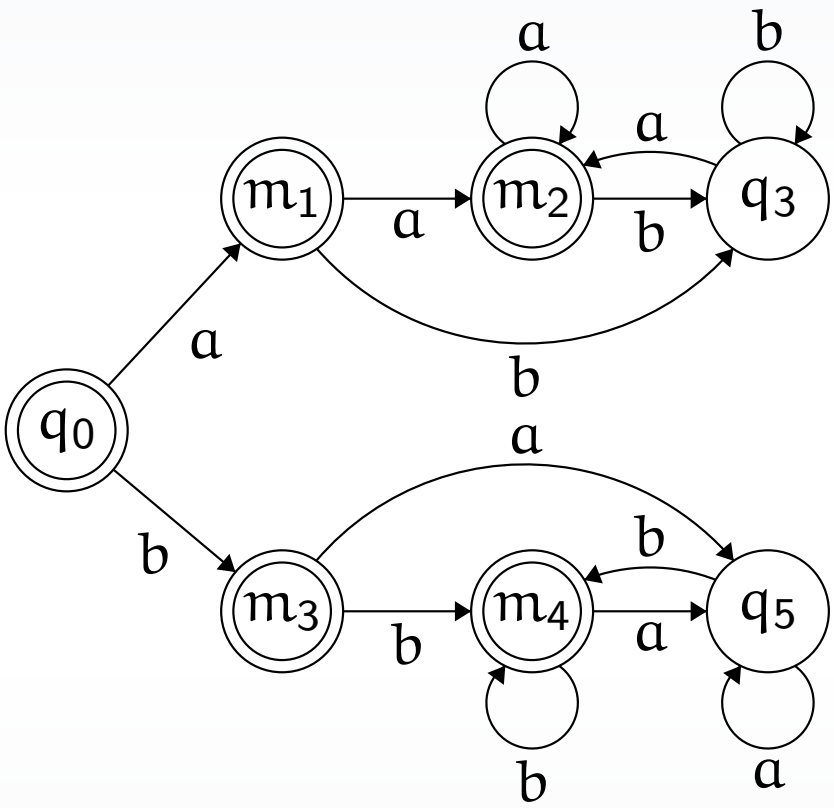
- $m_0 = \{q_i \mid q_0 \xrightarrow{\varepsilon} q_i\}$;
- $m_i \in F_D \Leftrightarrow \exists q_i, q_j \{q_i \in m_i \;\&\; q_j \in F(\mathscr{A}) \;\&\; q_i \xrightarrow{\varepsilon} q_j\}$;
- $\langle m, a, m' \rangle \in \delta_D \Leftrightarrow m' = \{q_i \mid \exists q_j \in m (q_j \xrightarrow{a} q_i)\}$.

Пример детерминизации

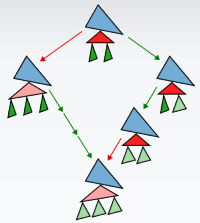


- $\{q_0\} \xrightarrow{a} \{q_1, q_8\}$,
  $\{q_0\} \xrightarrow{b} \{q_4, q_9\}$;
- $\{q_1, q_8\} \xrightarrow{a} \{q_1, q_2\}$,
  $\{q_1, q_8\} \xrightarrow{b} \{q_3\}$; $\{q_1, q_8\} \sim m_1$.
- $\{q_1, q_2\} \xrightarrow{a} \{q_1, q_2\}$,
  $\{q_1, q_2\} \xrightarrow{b} \{q_3\}$; $\{q_1, q_2\} \sim m_2$.
- $\{q_3\} \xrightarrow{a} \{q_1, q_2\}$, $\{q_3\} \xrightarrow{b} \{q_3\}$;
- $\{q_4, q_9\} \xrightarrow{b} \{q_4, q_6\}$,
  $\{q_4, q_9\} \xrightarrow{a} \{q_5\}$; $\{q_4, q_9\} \sim m_3$;
- $\{q_4, q_6\} \xrightarrow{b} \{q_4, q_6\}$,
  $\{q_4, q_6\} \xrightarrow{a} \{q_5\}$; $\{q_4, q_6\} \sim m_4$.
- $\{q_5\} \xrightarrow{b} \{q_4, q_6\}$, $\{q_5\} \xrightarrow{a} \{q_5\}$.

Пример детерминизации



- $\{q_0\} \xrightarrow{a} \{q_1, q_8\}$,
  $\{q_0\} \xrightarrow{b} \{q_4, q_9\}$;
- $\{q_1, q_8\} \xrightarrow{a} \{q_1, q_2\}$,
  $\{q_1, q_8\} \xrightarrow{b} \{q_3\}$; $\{q_1, q_8\} \sim m_1$.
- $\{q_1, q_2\} \xrightarrow{a} \{q_1, q_2\}$,
  $\{q_1, q_2\} \xrightarrow{b} \{q_3\}$; $\{q_1, q_2\} \sim m_2$.
- $\{q_3\} \xrightarrow{a} \{q_1, q_2\}$, $\{q_3\} \xrightarrow{b} \{q_3\}$;
- $\{q_4, q_9\} \xrightarrow{b} \{q_4, q_6\}$,
  $\{q_4, q_9\} \xrightarrow{a} \{q_5\}$; $\{q_4, q_9\} \sim m_3$;
- $\{q_4, q_6\} \xrightarrow{b} \{q_4, q_6\}$,
  $\{q_4, q_6\} \xrightarrow{a} \{q_5\}$; $\{q_4, q_6\} \sim m_4$.
- $\{q_5\} \xrightarrow{b} \{q_4, q_6\}$, $\{q_5\} \xrightarrow{a} \{q_5\}$.



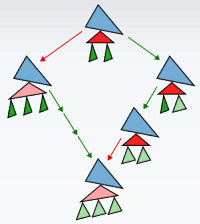
Замыкания регулярных языков

Гомоморфизм над свободной полугруппой (множеством слов) полностью определяется значениями на буквах, поскольку по определению $h(a_1 \circ a_2 \circ \cdots \circ a_n) = h(a_1) \circ h(a_2) \circ \cdots \circ h(a_n)$. Здесь $\circ$ —конкатенация.

Утверждение

Пусть $\mathscr{L}$ — регулярный язык над Σ. Тогда регулярны:

- язык $\Sigma^* \setminus \mathscr{L}$;
- для любого гомоморфизма h язык $\{h(w) \mid w \in \mathscr{L}\}$;
- для любого гомоморфизма h язык $\{w \mid h(w) \in \mathscr{L}\}$.



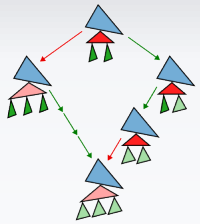
Замыкания регулярных языков

Утверждение

Пусть $\mathscr{L}$ — регулярный язык над Σ. Тогда регулярны:

- язык $\Sigma^* \setminus \mathscr{L}$;
- для любого гомоморфизма h язык $\{h(w) \mid w \in \mathscr{L}\}$;
- для любого гомоморфизма h язык $\{w \mid h(w) \in \mathscr{L}\}$.

Рассмотрим ДКА $\mathscr{A} = \langle Q, \Sigma, q_0, F, \delta \rangle$, распознающий $\mathscr{L}$.
Построим $\mathscr{A}' = \langle Q, \Sigma, q_0, Q \setminus F, \delta \rangle$. Тогда
$w \notin \mathscr{L} \Leftrightarrow w \in \mathscr{L}(\mathscr{A}')$.



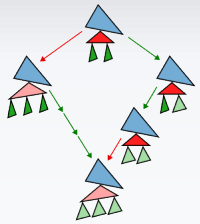
Замыкания регулярных языков

Утверждение

Пусть $\mathscr{L}$ — регулярный язык над Σ. Тогда регулярны:

- язык $\Sigma^* \setminus \mathscr{L}$;
- для любого гомоморфизма h язык $\{h(w) \mid w \in \mathscr{L}\}$;
- для любого гомоморфизма h язык $\{w \mid h(w) \in \mathscr{L}\}$.

Рассмотрим регулярное выражение R такое, что $\mathscr{L}(R) = \mathscr{L}$. Заменим в нём все $a_i \in \Sigma$ на $h(a_i)$. Полученное таким образом выражение R' также регулярно, причём $\mathscr{L}(R') = h(\mathscr{L})$.



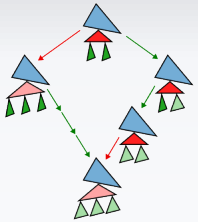
Замыкания регулярных языков

Утверждение

Пусть $\mathscr{L}$ — регулярный язык над Σ. Тогда регулярны:

- язык $\Sigma^* \setminus \mathscr{L}$;
- для любого гомоморфизма h язык $\{h(w) \mid w \in \mathscr{L}\}$;
- для любого гомоморфизма h язык $\{w \mid h(w) \in \mathscr{L}\}$.

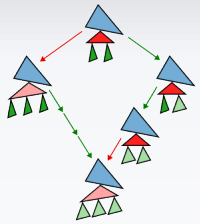
Рассмотрим ДКА $\mathscr{A} = \langle Q, \Sigma, q_0, F, \delta \rangle$, распознающий $\mathscr{L}$.
Построим $\mathscr{A}' = \langle Q, \Sigma, q_0, F, \delta' \rangle$ такой, что
$\langle q_i, a, q_j \rangle \in \delta' \Leftrightarrow q_i \xrightarrow{h(a)} q_j$ в исходном автомате $\mathscr{A}$.



Примеры

Рассмотрим язык $\mathscr{L}' = \{a^n b^m \mid n \neq m\}$.

Предположим, $\mathscr{L}'$ регулярен. Тогда $a^* b^* \setminus \mathscr{L}' = \{a^n b^n\}$ также регулярен, а мы знаем, что это не так. $\perp$



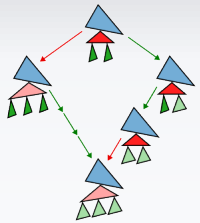
Примеры

Рассмотрим язык $\mathscr{L}' = \{a^n b^m \mid n \neq m\}$.

Предположим, $\mathscr{L}'$ регулярен. Тогда $a^* b^* \setminus \mathscr{L}' = \{a^n b^n\}$ также регулярен, а мы знаем, что это не так. $\perp$

Рассмотрим язык $\mathscr{L}^f = \{(abaabb)^n b^n\}$.

Попытка доказать его нерегулярность леммой о накачке породит перебор по накачиваемым строкам $(abaabb)^+$, $(abaabb)^* a$, $(abaabb)^* ab$,$(abaabb)^* aba$, $(abaabb)^* abaa$, ... Рассмотрим гомоморфизм $h(a) = abaabb$, $h(b) = b$. $h^{-1}(\mathscr{L}^f) = \{a^n b^n\}$, который был бы регулярен, если бы L^f был регулярен. $\perp$



Labelled Transition Systems

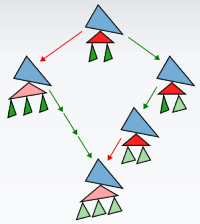
Понятие бисимуляции возникло в контексте систем размеченных переходов (LTS).

Определение

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

LTS похожи на конечные автоматы, но допускают бесконечные множества S и Q. Кроме того, в LTS нет начальных и финальных состояний.

Трансформационный моноид также строится в контексте LTS, то есть без учёта финальности состояний. Поэтому из ДКА, по которому строится трансформационный моноид, предварительно удаляются все ловушки, иначе в нём могут появиться правила переписывания, не имеющие никакого отношения к языку ДКА.



Симуляция и бисимуляция

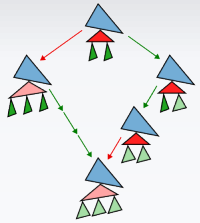
Определение

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

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

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

Можно считать, что если $p \precsim q$, то множество путей в LTS, стартующих в p, вкладывается в множество путей с началом в q. Бисимуляция состояний в единственной LTS легко обобщается и на бисимуляцию между двумя разными LTS. Поскольку в них нет начальных состояний, и они не обязаны быть связными, можно рассматривать несколько LTS как одну LTS с несколькими компонентами и искать бисимуляцию между элементами этих компонент.



Бисимилярность НКА

Чтобы определить отношение бисимуляции на конечных автоматах, к отношению бисимуляции на LTS нужно добавить ограничения на бисимуляцию начальных и конечных состояний. Более точно, для бисимуляции НКА $\mathscr{A}_1$ 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.

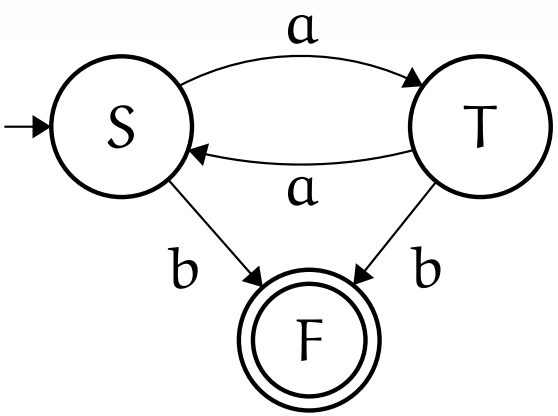
Лемма

Бисимилярные НКА распознают равные языки.

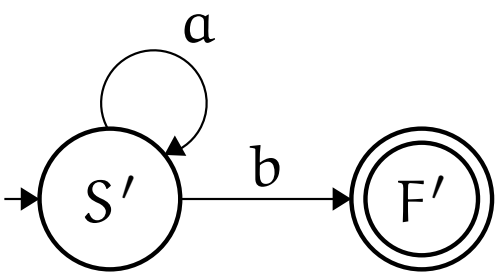
Пример бисимилярных НКА

Рассмотрим следующие два автомата, распознающие язык a^*b.

Автомат $\mathscr{A}_1$:



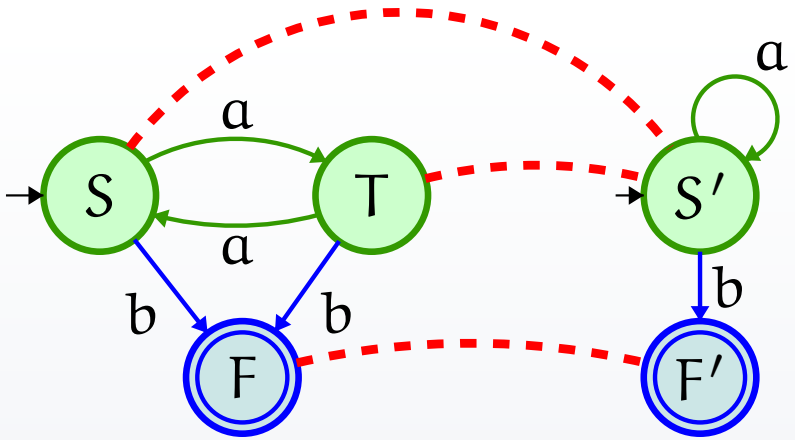
Автомат $\mathscr{A}_2$:

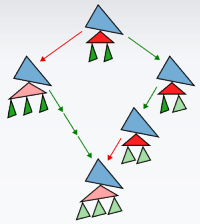


Их бисимуляция:

$$\{\langle S, S'\rangle, \langle T, S'\rangle, \langle F, F'\rangle\}$$

Состояния S и T бисимилярны одному и тому же состоянию S'.





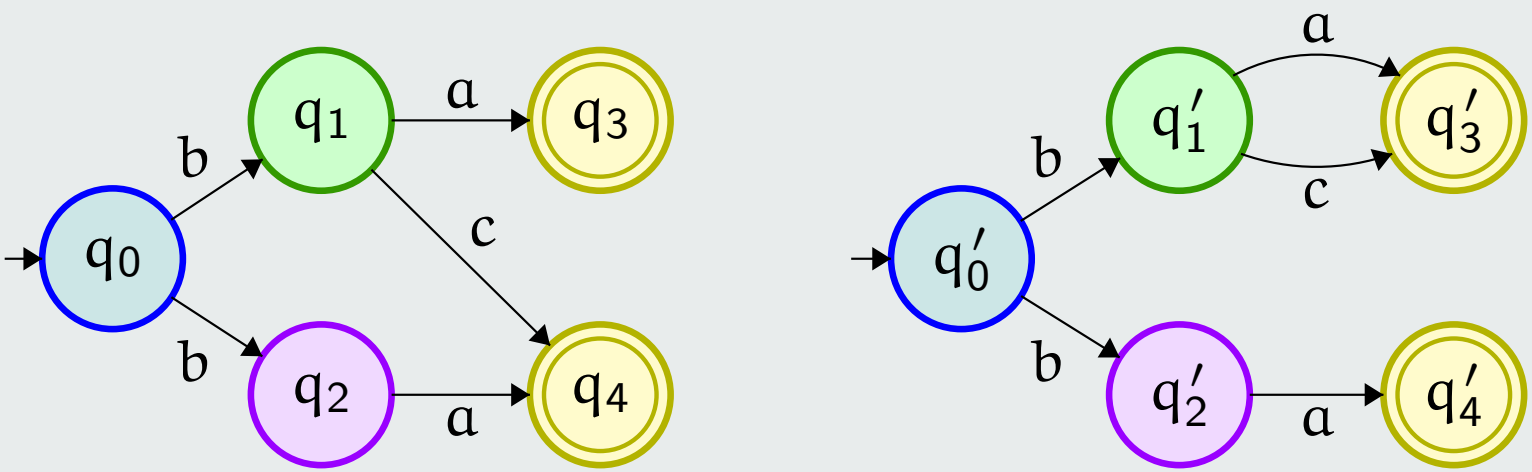
Бисимуляция и равенство

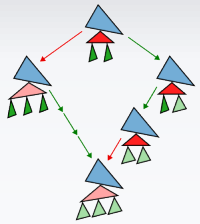
В равных НКА состояния бисимилярны, однако только условия существования бисимуляции и биекции бисимилярных состояний недостаточно, чтобы гарантировать равенство.

Пример неравных бисимилярных НКА

(автор примера: А. Д. Дельман)

Следующие два автомата бисимилярны и имеют одинаковое число состояний, однако не равны:



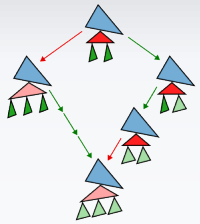


Бисимилярность состояний в НКА

Определение

Состояния q_i q_j в НКА $\mathscr{A}$ бисимилярны ($q_i \sim_{\mathscr{A}} q_j$), если они связаны LTS-бисимуляцией и имеют одинаковую финальность в $\mathscr{A}$.

- С учётом определения выше, бисимуляцию НКА можно переформулировать как отношение бисимуляции состояний НКА такое, что стартовые состояния бисимилярны.
- Отношение $\sim_{\mathscr{A}}$ имеет важное свойство: бисимилярные состояния в автомате можно объединить без изменения его семантики. Это преобразование часто позволяет существенно упростить НКА.



Бисимилярность состояний в НКА

Определение

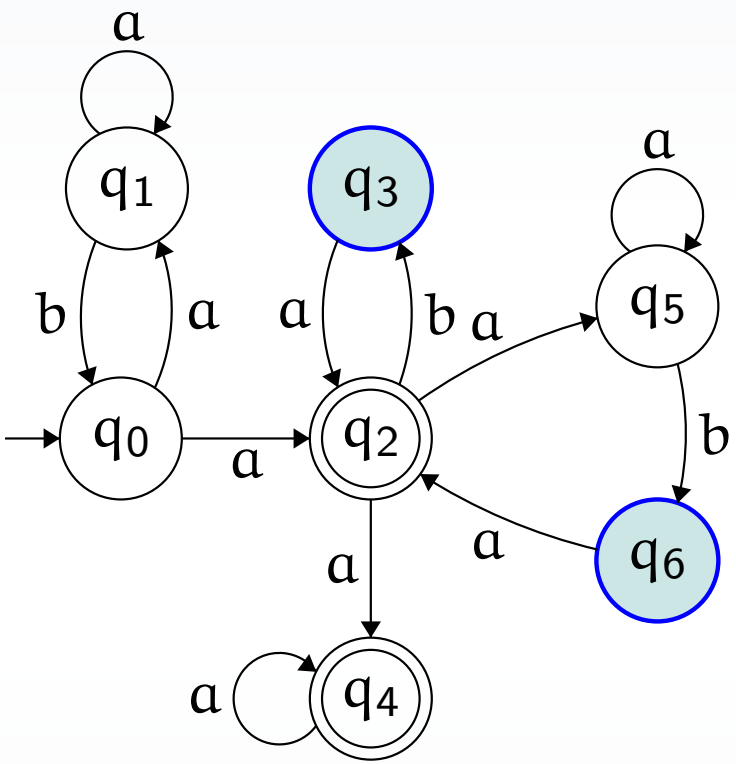
Состояния q_i q_j в НКА $\mathscr{A}$ бисимилярны ($q_i \sim_{\mathscr{A}} q_j$), если они связаны LTS-бисимуляцией и имеют одинаковую финальность в $\mathscr{A}$.

Пример

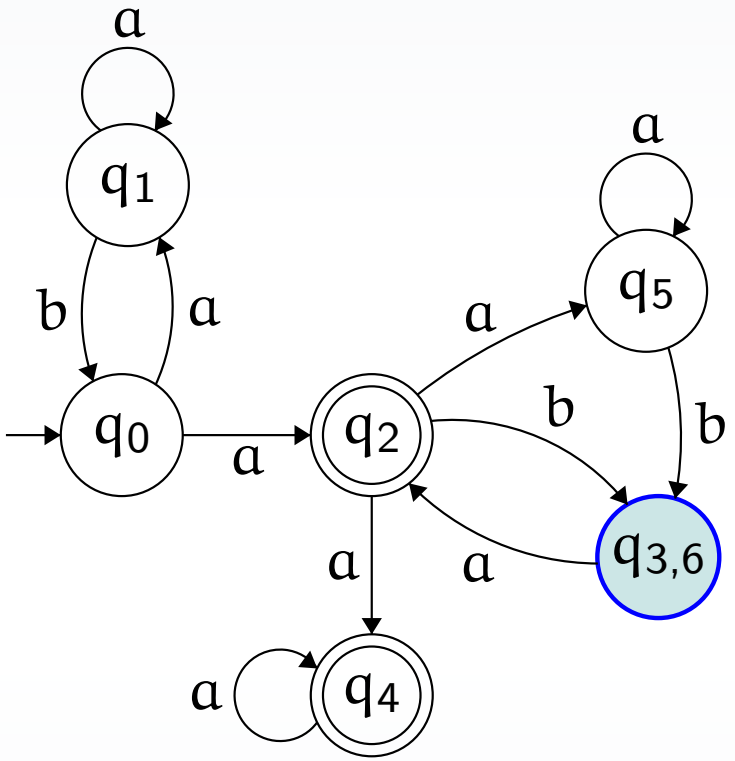
Все состояния-ловушки в любом полном автомате (т.е. с явно присутствующими переходами по всем буквам алфавита) бисимилярны друг другу. Все финальные состояния без переходов из них (кроме как в ловушки) также бисимилярны.

Пример слияния по бисимуляции

Исходный автомат:



Итоговый автомат:

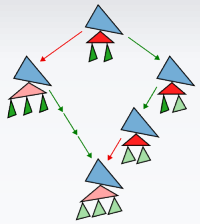


Бисимуляция: $\left\{\{q_3, q_6\}, \bigcup_{i\neq 3 \& i\neq 6} \{q_i\}\right\}$

Кроме q_3 и q_6, все состояния не бисимилярны никаким другим.

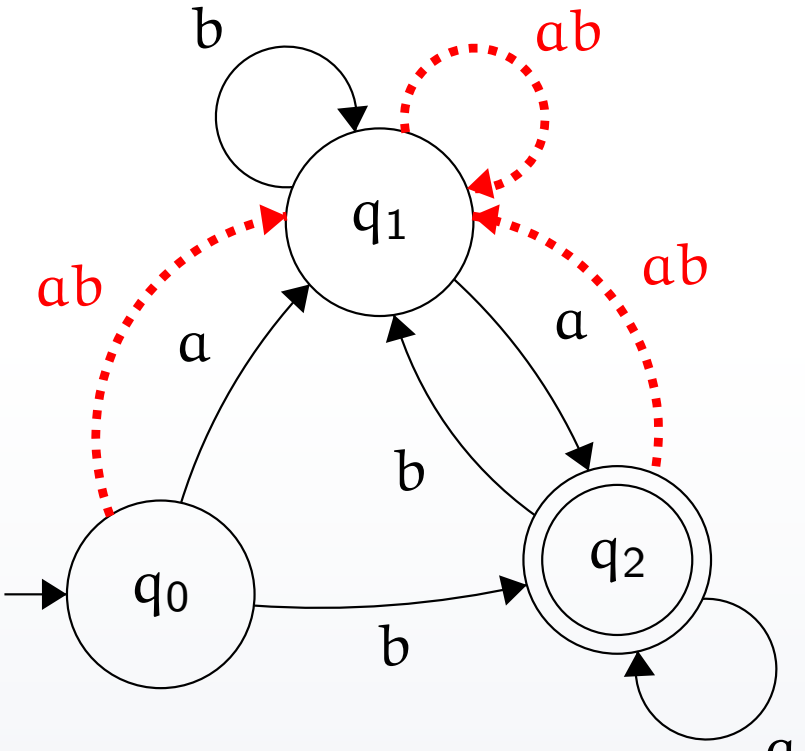
Например, $q_1 \not\sim q_5$, поскольку $q_1 \xrightarrow{b} q_0$, $q_5 \xrightarrow{b} q_6$, но $q_0 \xrightarrow{a} q_1$ (и q_1 — не финальное), а из q_6 есть переход только в финальное состояние q_2.

Трансформационный моноид



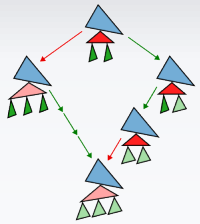
Функции переходов по слову в ДКА

Правила перехода в ДКА $\mathscr{A}$ над алфавитом Σ и множеством состояний Q определяются функцией $\Sigma \times Q \to Q$. Если специализировать её по первому аргументу, получится функция $F_\xi : Q \to Q$ ($\xi \in \Sigma$). Эту функцию можно продолжить на строки, положив $F_\xi \circ F_\eta = F_{\eta\xi}$.



Пусть мы строим функцию переходов по слову ab в автомате $\mathscr{A}$. Сначала определим функции F_a, F_b, определяющие его поведение на буквах a и b. Тогда поведение переходов на слове ab получится композицией F_a и F_b.

	q_0	q_1	q_2
a	q_1	q_2	q_2
b	q_2	q_1	q_1
ab	q_1	q_1	q_1



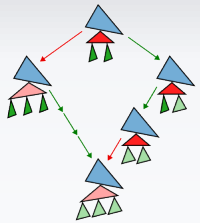
Функции переходов по слову в ДКА

Свойства множества функций переходов ДКА $\mathscr{A}$

- Существует единичная функция F_ε такая, что $F_\varepsilon \circ F_\xi = F_\xi \circ F_\varepsilon = F_\xi$.
- Композиция $\circ$ ассоциативна.

Таким образом, функции переходов по словам из Σ^* в ДКА $\mathscr{A}$ образуют моноид относительно композиции.

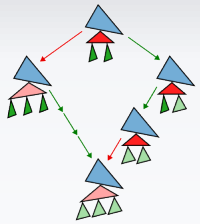
Если ДКА представлен в краткой (trim) форме, некоторые переходы могут вести «в никуда». На самом деле они ведут в (единственное!) состояние-ловушку, существование которого неявно подразумевается. Однако наличие нескольких ловушек в ДКА повлечёт ошибки при построении функции переходов.



Определение и свойства

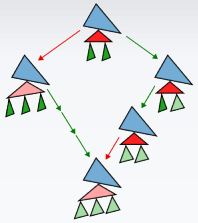
Определение

Трансформационный моноид $\mathcal{M}_{\mathscr{A}}$ для ДКА $\mathscr{A}$ — это моноид функций F_ξ таких, что $F_\xi(q_i) = q_j \Leftrightarrow (q_i \xrightarrow{\xi} q_j$ в $\mathscr{A})$. Иначе можно сказать, что трансформационный моноид $\mathcal{M}_{\mathscr{A}}$ определяется множеством классов эквивалентности $\{w \mid w \in \Sigma^+\}$ таким, что $w_i = w_j \Leftrightarrow F_{w_i} = F_{w_j}$.

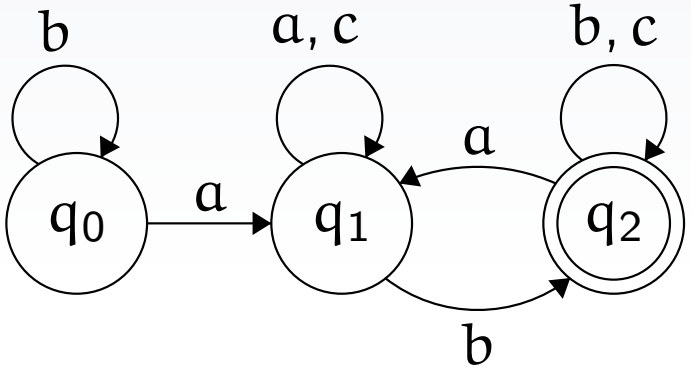


Определение и свойства

- $\mathcal{M}_{\mathscr{A}}$ определяется фактормножеством классов эквивалентности и правилами переписывания, задающими эквивалентность. ε обычно не включается в множество w_i.
- Поскольку множество функций F_{w_i} в случае ДКА конечно, то $\mathcal{M}_{\mathscr{A}}$ содержит конечное число классов эквивалентности (верно и обратное: каждый такой моноид определяет ДКА).
- Трансмоноид строится для ДКА без ловушек; переход в ловушку обозначается в таблице переходов просто прочерком.
- Для единообразия записи трансформаций и перестановок в алгебре, в таблице переходов пишут только номера состояний $\mathscr{A}$.

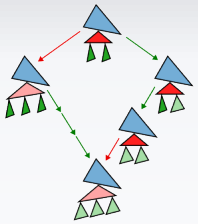


Построение трансф. моноида

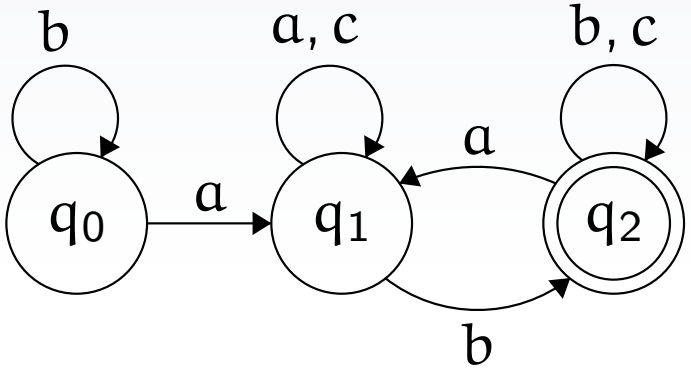


Определим соответствие между буквами и множествами переходов по ним и будем расширять этот список новыми словами в лексикографическом порядке. Если очередное слово задаёт такую же трансформацию, как и уже рассмотренное, порождаем соответствующее правило переписывания.

Классы эквивалентности				**Правила переписывания**
	0	1	2	
a	1	1	1	
b	0	2	2	
c	—	1	2	



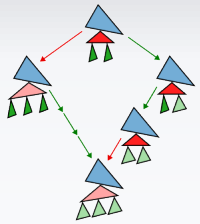
Построение трансф. моноида



Классы эквивалентности			
	0	1	2
a	1	1	1
b	0	2	2
c	—	1	2
ab	2	2	2
bc	—	2	2
ca	—	1	1

Правила переписывания	
$aa \to a$	$ac \to a$
$ba \to a$	$bb \to b$
$cb \to bc$	$cc \to c$
$abc \to ab$	$bca \to ca$
$cab \to bc$	

Всего классов эквивалентности: 6



Синтаксический моноид

Определим отношение синтаксической конгруэнтности слов:

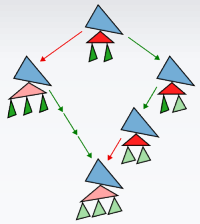
$$w_i \sim_{\mathscr{L}} w_j \Leftrightarrow \forall x, y (x\, w_i\, y \in \mathscr{L} \Leftrightarrow x\, w_j\, y \in \mathscr{L})$$

Синтаксический моноид $\mathscr{M}(\mathscr{L})$ — это множество его классов эквивалентности относительно $\sim_{\mathscr{L}}$. То есть такая полугруппа с единицей над $w \in \Sigma^*$, что $w_i = w_j \Leftrightarrow w_i \sim_{\mathscr{L}} w_j$ (равенство здесь понимается в алгебраическом смысле: как возможность преобразовать w_i и w_j к одному и тому же слову).

Лемма

Синтаксический моноид регулярного языка $\mathscr{L}$ совпадает с трансф. моноидом минимального ДКА, его распознающего.

Синтаксический моноид (так же, как и минимальный ДКА) — атрибут *языка*, а трансф. моноид — атрибут *конкретного ДКА*.



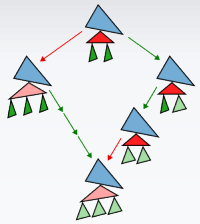
Суффиксная конгруэнтность

Предшествующие понятия рассматривали структуру переходов автомата без учёта начальных и конечных состояний, хотя неявно они использовались, чтобы удалить недостижимые состояния и состояния-ловушки при подготовке к построению моноида. Однако если чуть-чуть специализировать отношение $\sim_{\mathscr{L}}$, положив возможные префиксы пустыми, получится отношение эквивалентности, напрямую зависящее от положения стартовых и финальных состояний в минимальном ДКА.

Определим отношение эквивалентности по Нероуду как:

$$w_i \equiv_{\mathscr{L}} w_j \Leftrightarrow \forall y (w_i\, y \in \mathscr{L} \Leftrightarrow w_j\, y \in \mathscr{L})$$

Обозначение $\sim_{\mathscr{L}}$ может использоваться в литературе как в смысле синтаксической конгруэнции, так и в смысле эквивалентности по Нероуду. Лучше дополнительно уточнить.



Критерий регулярности языка

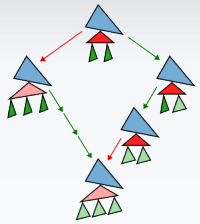
Теорема Майхилла–Нероуда

Язык $\mathscr{L}$ регулярен тогда и только тогда, когда множество классов эквивалентности по $\equiv_{\mathscr{L}}$ конечно.

$\Rightarrow$: Пусть $\mathscr{L}$ регулярен. Тогда он порождается некоторым DFA $\mathscr{A}$ с конечным числом состояний N. Значит, множество $\left\{q_i \mid q_0 \xrightarrow{w} q_i\right\}$ конечно, а для каждых двух w_1, w_2 таких, что $q_0 \xrightarrow{w_1} q_i$ и $q_0 \xrightarrow{w_2} q_i$, выполняется $w_1 \equiv_{\mathscr{L}} w_2$.

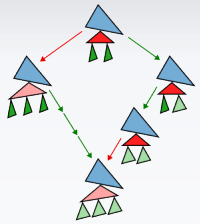
$\Leftarrow$: Пусть все слова в Σ^* принадлежат N классам эквивалентности $A_1, \ldots, A_n$ по $\equiv_{\mathscr{L}}$. Построим по ним DFA $\mathscr{A}$, распознающий $\mathscr{L}$. Классы A_i сопоставим состояниям:

- Начальным объявим класс эквивалентности A_0 такой, что $\varepsilon \in A_0$.
- Конечными объявим такие A_j, что $\forall w \in A_j (w \in \mathscr{L})$.
- Если $w \in A_i$, $w\,a_k \in A_j$, тогда добавляем в δ правило $\langle A_i, a_k, A_j \rangle$.

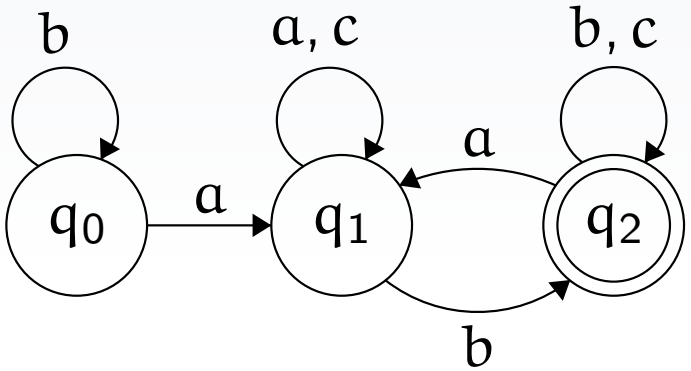


Трансформационный моноид и $\equiv_{\mathscr{L}}$

- Классы эквивалентности по Майхиллу–Нероуду можно извлечь из трансформационного моноида минимального ДКА для языка $\mathscr{L}$: они являются подмножеством факторслов, которые переводят стартовое состояние в какое-то другое (непустое) состояние.
- Для каждой пары таких факторслов w_i и w_j, переводящих стартовое состояние в разные состояния q_i и q_j, в синтаксическом моноиде обязательно найдётся различающий суффикс (т.е. класс эквивалентности u такой, что $w_i\,u \in \mathscr{L}$ & $w_j\,u \notin \mathscr{L}$, либо наоборот).
- Если в ДКА существует ловушка (возможно, неявная), то в трансформационном моноиде найдётся класс эквивалентности, переводящий в неё стартовое состояние. Таких классов может быть несколько, но с точки зрения эквивалентности по Майхиллу–Нероуду, они не различаются.



Пример



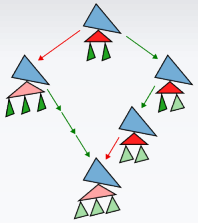
Включим в число факторслов ε и выделим по одному факторслову для каждого состояния q_i, переводящему стартовое слово в q_i. Для каждого из них определим множество суффиксов, которые оставляют слова этих классов в языке. После этого достаточно собрать вместе все суффиксы и префиксы и выкинуть из полученной таблицы дубли столбцов.

Факторслова–префиксы

префикс	$0 \to ?$	суффиксы
ε	0	ab
a	1	b, ab
ab	2	ε, b, ab

Таблица классов эквивалентности

	ε	b	ab
ε	−	−	+
a	−	+	+
ab	+	+	+

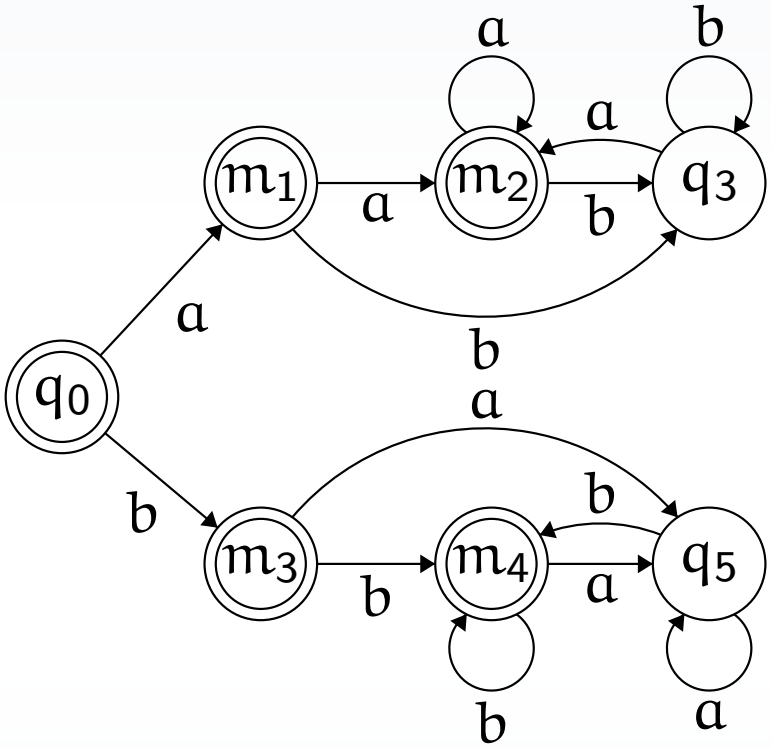


Минимизация ДКА

1. Построим таблицу всех двухэлементных множеств $\{q_i, q_j\}$, $q_i, q_j \in Q$.
2. Пометим все множества $\{q_i, q_j\}$ такие, что одно из q_i, q_j из F, а второе нет.
3. Пометим все множества $\{q_i, q_j\}$ такие, что $\exists a(q_i \xrightarrow{a} q_1' \;\&\; q_j \xrightarrow{a} q_2' \;\&\; \{q_1', q_2'\}$ — помеченная пара$)$.
4. Продолжаем шаг 3, пока не будет появляться новых помеченных пар.

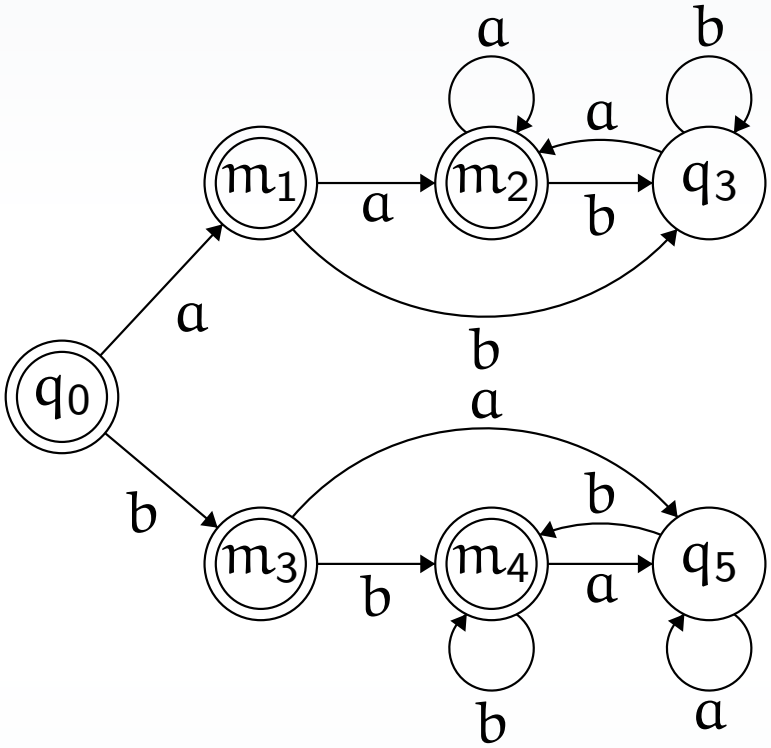
Пары, оставшиеся непомеченными, можно объединить.

Пример минимизации



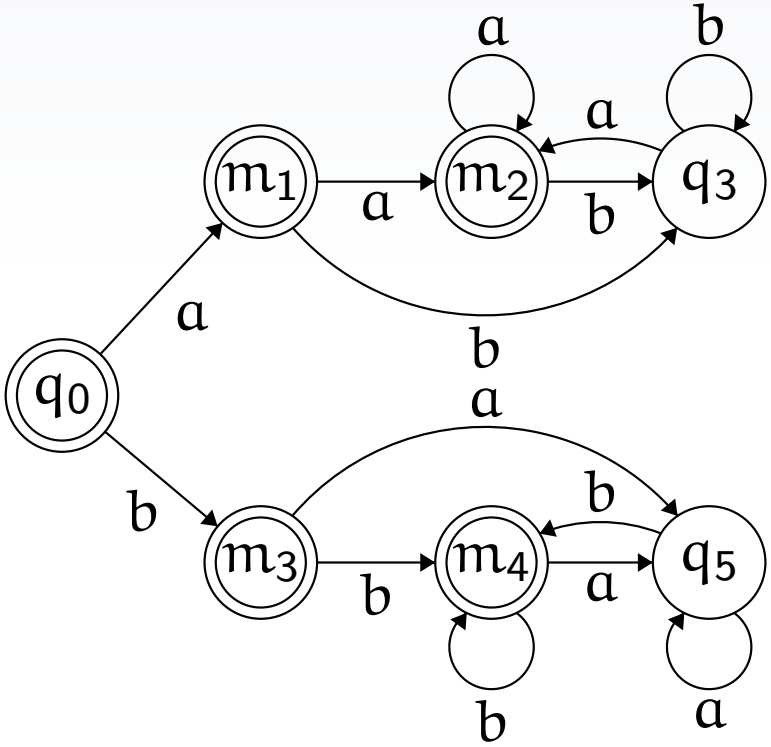
m_1						
m_2						
q_3						
m_3						
m_4						
q_5						
	q_0	m_1	m_2	q_3	m_3	m_4

Пример минимизации



m_1						
m_2						
q_3	✓	✓	✓			
m_3				✓		
m_4				✓		
q_5	✓	✓	✓		✓	✓
	q_0	m_1	m_2	q_3	m_3	m_4

Пример минимизации

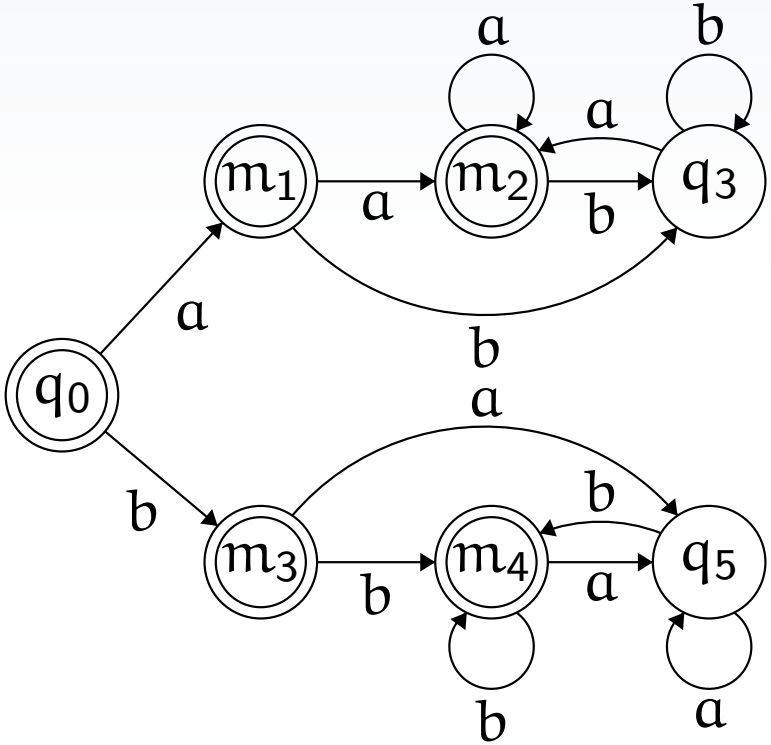


m_1						
m_2						
q_3	✓	✓	✓			
m_3				✓		
m_4				✓		
q_5	✓	✓	✓		✓	✓
	q_0	m_1	m_2	q_3	m_3	m_4

$q_0 \xrightarrow{a} m_1, m_1 \xrightarrow{a} m_2$ $\quad q_0 \xrightarrow{b} m_3, m_1 \xrightarrow{b} q_3$ $\quad \{m_1, m_2\} \xrightarrow{a} m_2$ $\quad \{m_1, m_2\} \xrightarrow{b} q_3$

$q_0 \xrightarrow{a} m_1, m_2 \xrightarrow{a} m_2$ $\quad q_0 \xrightarrow{b} m_3, m_2 \xrightarrow{b} q_3$

Пример минимизации

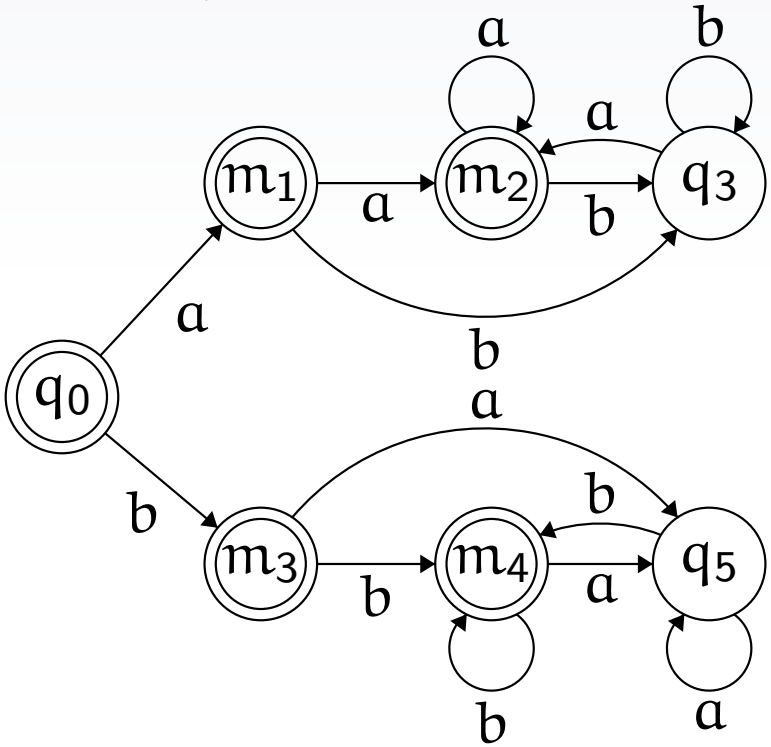


m_1	✓					
m_2	✓					
q_3	✓	✓	✓			
m_3				✓		
m_4				✓		
q_5	✓	✓	✓		✓	✓
	q_0	m_1	m_2	q_3	m_3	m_4

$q_0 \xrightarrow{a} m_1, m_3 \xrightarrow{a} q_5$ $q_0 \xrightarrow{a} m_1, m_4 \xrightarrow{a} q_5$ $m_1 \xrightarrow{a} m_2, m_3 \xrightarrow{a} q_5$

$m_2 \xrightarrow{a} m_2, m_3 \xrightarrow{a} q_5$ $m_1 \xrightarrow{a} m_2, m_4 \xrightarrow{a} q_5$ $m_2 \xrightarrow{a} m_2, m_4 \xrightarrow{a} q_5$

Пример минимизации

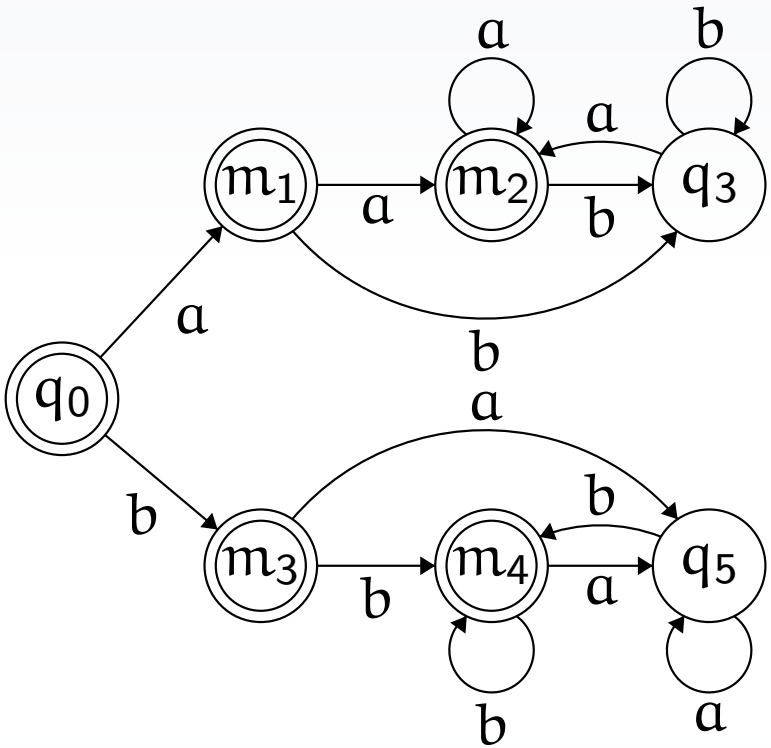


m_1	✓					
m_2	✓					
q_3	✓	✓	✓			
m_3	✓	✓	✓	✓		
m_4	✓	✓	✓	✓		
q_5	✓	✓	✓		✓	✓
	q_0	m_1	m_2	q_3	m_3	m_4

$\{m_3, m_4\} \xrightarrow{a} q_5$ $\{m_3, m_4\} \xrightarrow{b} m_4$

$q_3 \xrightarrow{a} m_2, q_5 \xrightarrow{a} m_4$

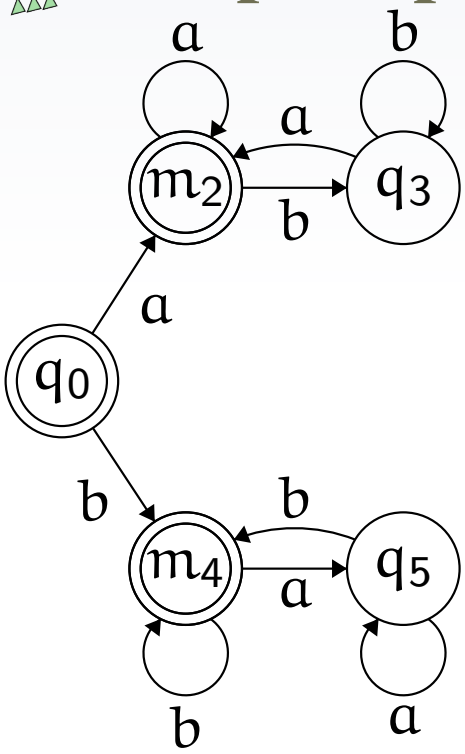
Пример минимизации



m_1	✓					
m_2	✓					
q_3	✓	✓	✓			
m_3	✓	✓	✓	✓		
m_4	✓	✓	✓	✓		
q_5	✓	✓	✓	✓	✓	✓
	q_0	m_1	m_2	q_3	m_3	m_4

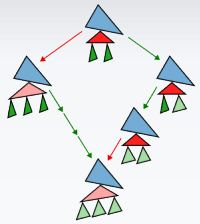
Можно объединить состояния m_1 и m_2 и состояния m_3 и m_4.

Пример минимизации



m_1	✓					
m_2	✓					
q_3	✓	✓	✓			
m_3	✓	✓	✓	✓		
m_4	✓	✓	✓	✓		
q_5	✓	✓	✓	✓	✓	✓
	q_0	m_1	m_2	q_3	m_3	m_4

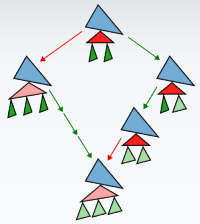
Меньше чем пятью состояниями не обойтись. Рассмотрим слова ε, a, b, ab, ba. Каждые два из них различаются по $\equiv_{\mathscr{L}}$ при выборе одного из трёх z: ε, a или b.



Связь M.–N. и производных

Пусть $w^{-1}U$ — это производная U по w, т.е. $\{v \mid wv \in U\}$.
Тогда выполнено $x \equiv_U y \Leftrightarrow x^{-1}U = y^{-1}U$.

- Количество производных (как языков) регулярного языка конечно.
- Конструкция Брзозовки порождает минимальный DFA.



Связь M.–N. и производных

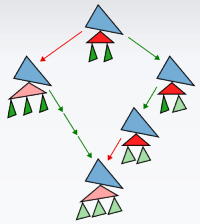
Пусть $w^{-1}U$ — это производная U по w, т.е. $\{v \mid wv \in U\}$.
Тогда выполнено $x \equiv_U y \Leftrightarrow x^{-1}U = y^{-1}U$.

- Количество производных (как языков) регулярного языка конечно.
- Конструкция Брзозовки порождает минимальный DFA.

Но проблема с правилами переписывания (ACI):

- $(w_1 \mid w_2) \mid w_3 = w_1 \mid (w_2 \mid w_3)$
- $w_1 \mid w_2 = w_2 \mid w_1$
- $w \mid w = w$

Пример проблемного выражения: $\mathtt{a}^*\mathtt{a}^*$.



Применение теоремы М.–N.

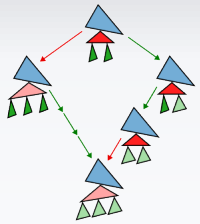
Задача

Дан язык $\mathscr{L}$. Показать, что он не регулярен, пользуясь теоремой Майхилла–Нероуда.

Стандартный подход

1. Подобрать бесконечную последовательность префиксов $w_1, \ldots, w_n, \ldots$.
2. Подобрать бесконечную последовательность суффиксов $z_1, \ldots, z_n, \ldots$, такую, что $w_i +\!\!+ z_i \in \mathscr{L}$.
3. Доказать, что в таблице конкатенаций все строки различны (значит, $\forall i, j\, \exists k (w_i z_k \in \mathscr{L} \;\&\; w_j z_k \notin \mathscr{L})$).

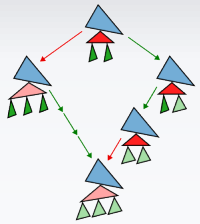
Диагональная конструкция (условие $w_i +\!\!+ z_i \in \mathscr{L}$) — одна из многих возможных, обычно она довольно удобна.



Диагональная конструкция

Рассмотрим язык $L = \{a^n b^n\}$. Положим $w_i = a^i$, $z_i = b^i$. Тогда таблица конкатенаций w_i, z_j будет выглядеть следующим образом. Здесь $+$ — это то же, что «$\in \mathscr{L}$», $-$ читаем как «$\notin \mathscr{L}$».

	$z_1 = b$	$z_2 = b^2$	$z_3 = b^3$	$\dots$	$z_n = b^n$	$\dots$
$w_1 = a$	$+$	$-$	$-$		$-$	
$w_2 = a^2$	$-$	$+$	$-$		$-$	
$w_3 = a^3$	$-$	$-$	$+$		$-$	
$\dots$			$\dots$			
$w^n = a^n$	$-$	$-$	$-$		$+$	
$\dots$						



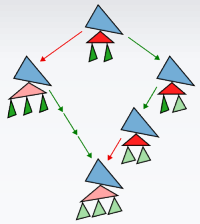
Доказательство минимальности

Так же можно обосновывать минимальность DFA. Рассмотрим минимальный автомат из примера выше. Его язык — слова в $\{a, b\}^*$, начинающиеся и заканчивающиеся одной и той же буквой. Построим таблицу классов эквивалентности по $w_i \in \{\varepsilon, a, b, ab, ba\}$.

	ε	a	b
ε	$+$	$+$	$+$
a	$+$	$+$	$-$
b	$+$	$-$	$+$
ab	$-$	$+$	$-$
ba	$-$	$-$	$+$

В этой таблице все строчки различны, значит, выбранные w_i действительно лежат в различных классах эквивалентности, и DFA, распознающий язык $\mathscr{L}$, не может иметь меньше пяти состояний.

При доказательстве минимальности DFA достаточно подобрать $[\log_2 n] + 1$ различающих суффиксов z_i, где n — число состояний автомата.

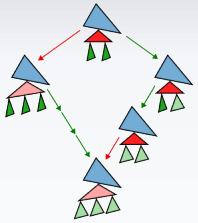


О порождении новых алгоритмов

Допустим существование нескольких начальных состояний у КА. Пусть $\mathscr{A}$ — НКА. Тогда $\mathtt{det}(\mathtt{reverse}(\mathtt{det}(\mathtt{reverse}(\mathscr{A}))))$ — минимальный ДКА, эквивалентный $\mathscr{A}$.

Многие алгоритмы для порождения малых (не минимальных) NFA являются комбинациями нескольких базовых операций.

- Обращение автомата
- Детерминизация
- Удаление ε-правил
- Минимизация
- Разметка



Несколько конструкций

- Автомат Глушкова: $\mathrm{rmeps}(\mathrm{Th}(\mathrm{R}))$;
- Автомат Антимирова: $\mathrm{rmeps}(\mathrm{deannote}(\mathrm{minimize}(\mathrm{rmeps}(\mathrm{annote\_eps}(\mathrm{Th}(\mathrm{R}))))))$;
- Автомат Илия–Ю: $\mathrm{deannote}(\mathrm{minimize}(\mathrm{rmeps}(\mathrm{annote}(\mathrm{Th}(\mathrm{R})))))$.