

Теория автоматов и формальных языков

Регулярные языки

Лектор: Екатерина Вербицкая

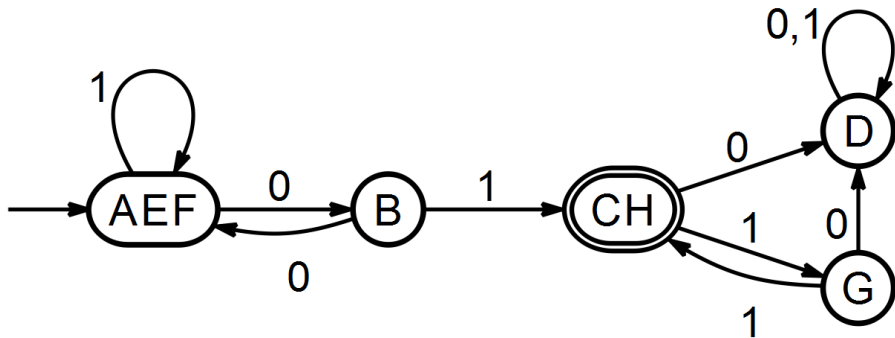
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»

20 сентября 2016г.

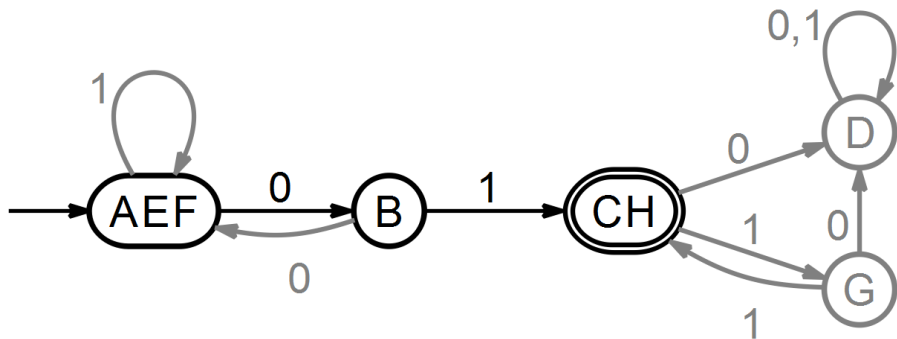
Конечный автомат — $\langle Q, \Sigma, \delta, q_0, F \rangle$

- $Q \neq \emptyset$ — конечное множество состояний
- Σ — Конечный входной алфавит
- δ — функция переходов
 - ▶ Детерминированный КА: отображение типа $Q \times \Sigma \rightarrow Q$
 - ▶ Недетерминированный КА: отображение типа $Q \times \Sigma \rightarrow 2^Q$
- $q_0 \in Q$ — начальное состояние
- $F \subseteq Q$ — множество конечных состояний

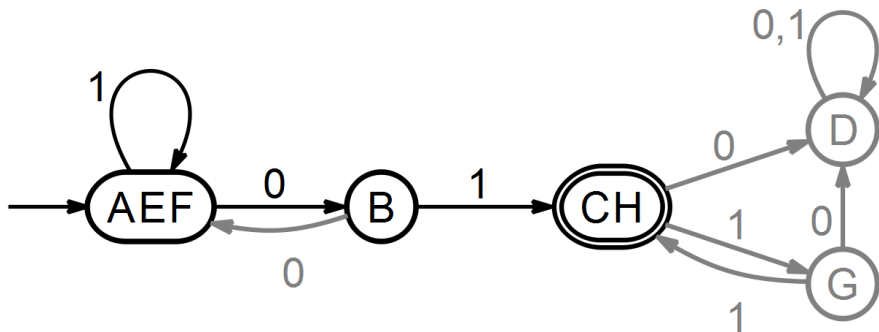
В предыдущей серии: ДКА



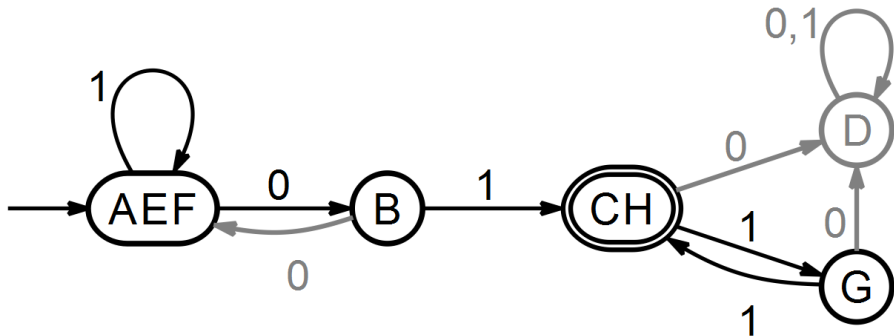
В предыдущей серии: распознавание слова ДКА



В предыдущей серии: распознавание слова ДКА

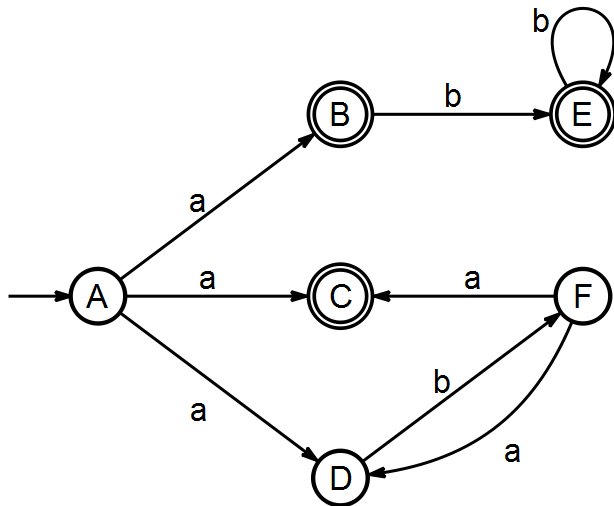


В предыдущей серии: распознавание слова ДКА



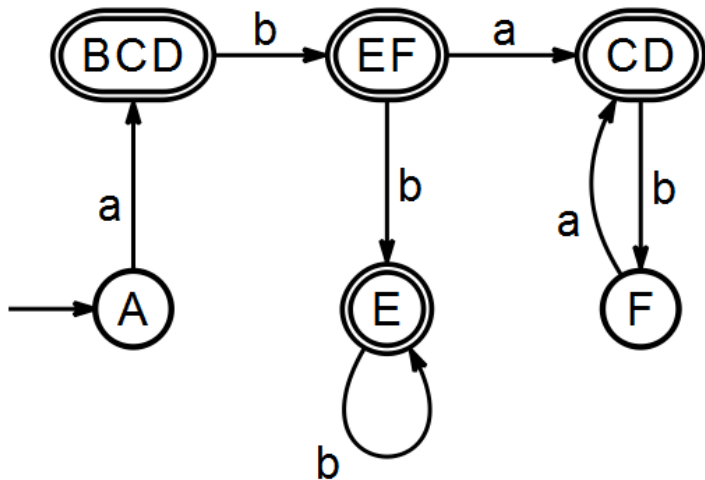
Слово распознается за $O(n)$

В предыдущей серии: распознавание слова НКА

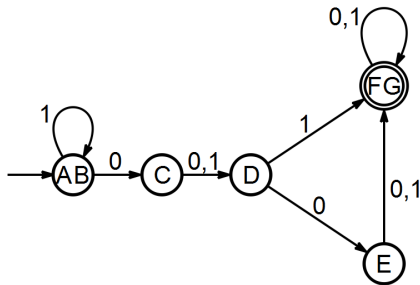
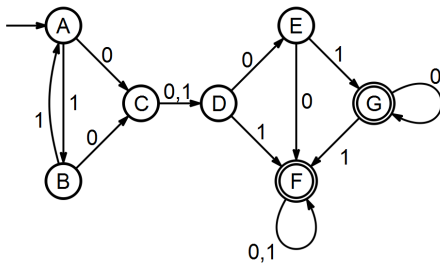


Слово распознается за ...

В предыдущей серии: детерминизация



В предыдущей серии: минимизация



Произведение автоматов

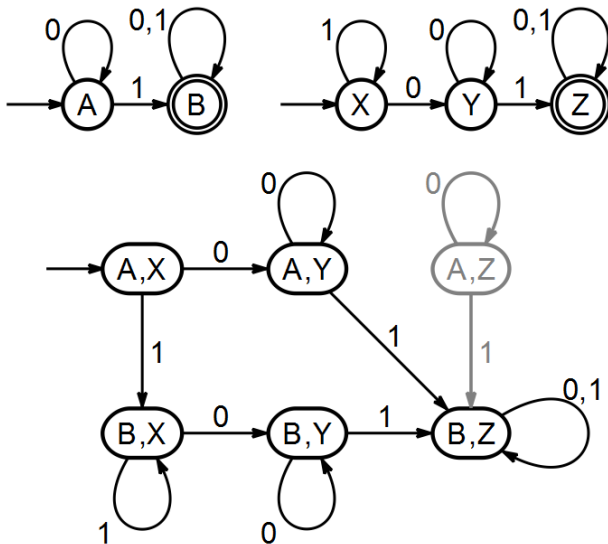
$A_1 = \langle \Sigma_1, Q_1, q_{10}, \delta_1, F_1 \rangle$ и $A_2 = \langle \Sigma_2, Q_2, q_{20}, \delta_2, F_2 \rangle$ — КА

Произведением автоматов назовем $A = \langle \Sigma, Q, q_0, \delta, F \rangle$, где

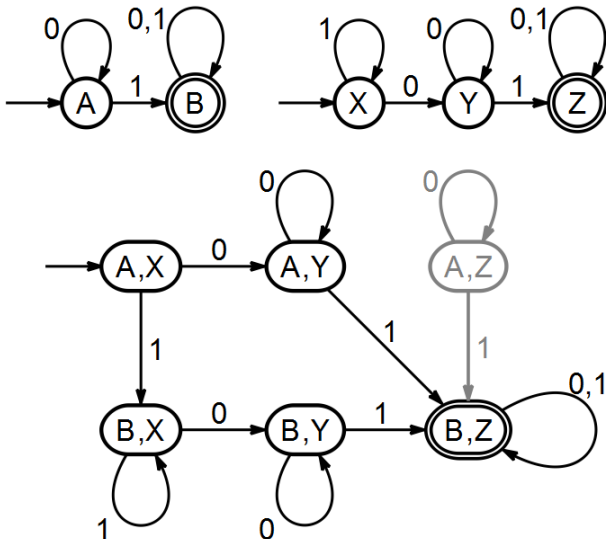
- $\Sigma = \Sigma_1 \cup \Sigma_2$
- $Q = Q_1 \times Q_2$
- $q_0 = (q_{10}, q_{20})$
- $F \subseteq Q$
 - ▶ $F = F_1 \times F_2$ — распознает **произведение** языков
 - ▶ $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$ — распознает **объединение** языков
 - ▶ $F = F_1 \times (Q_2 \setminus F_2)$ — распознает **разность** языков
- $\delta((q_1, q_2), c) = (\delta_1(q_1, c), \delta_2(q_2, c))$

Интуиция: ищем пути в двух автоматах одновременно

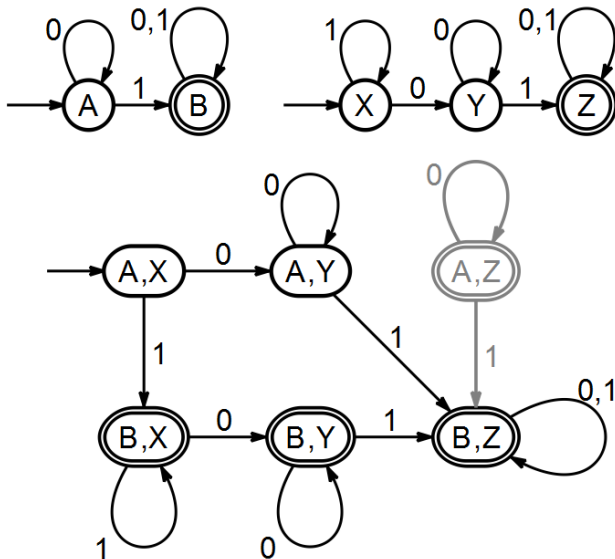
Произведение автоматов: пример



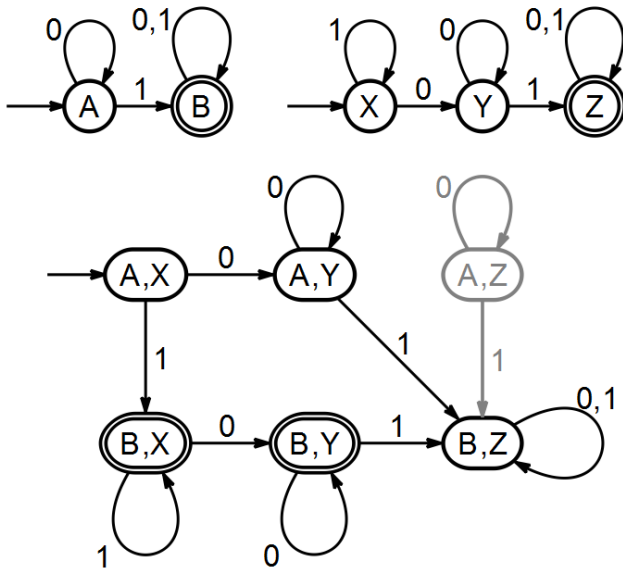
Пересечение языков



Объединение языков



Разность языков



Замкнутость автоматных языков относительно операций

Автоматные языки замкнуты относительно операций

- Объединения
- Пересечения
- Разности
- Дополнения

Регулярное множество (регулярный язык)

Регулярное множество в алфавите Σ определяется итеративно:

- \emptyset — регулярное множество в алфавите Σ
- $\{a\}$ — регулярное множество в алфавите Σ для каждого $a \in \Sigma$
- $\{\varepsilon\}$ — регулярное множество в алфавите Σ
- Если P и Q — регулярные множества в алфавите Σ , то регулярны
 - ▶ $P \cup Q$ (объединение)
 - ▶ PQ (конкатенация, $\{pq | p \in P, q \in Q\}$)
 - ▶ P^* (итерация: $P^* = \{\varepsilon\} \cup P \cup PP \cup PPP \cup \dots$)
- Ничто другое не является регулярным множеством в алфавите Σ
- Множество всех регулярных языков обозначим \mathbb{R}

Примеры регулярных языков

- Все конечные языки
 - ▶ $\{-2147483648, -2147483647, \dots, 2147483647\}$ — все 32-разрядные целые числа
- $L_a = \{a^k \mid k - \text{odd}\}$
- $L_b = \{b^l \mid l - \text{even}\}$
- $L_{ab} = \{a^k b^l \mid k - \text{odd}, l - \text{even}\} = L_a L_b$
- $L = \{a^*\} = L_a^*$

Регулярное выражение

Регулярное выражение — способ записи регулярного множества

- \emptyset — обозначает \emptyset
- a — обозначает $\{a\}$
- ε — обозначает $\{\varepsilon\}$
- Если p и q обозначают P и Q , то:
 - ▶ $p|q$ обозначает $P \cup Q$
 - ▶ pq обозначает PQ
 - ▶ p^* обозначает P^*

Примеры регулярных выражений

- $-2147483648 | -2147483647 | \dots | 2147483647$ — все 32-разрядные целые числа
- $a(aa)^* : L_a = \{a^k \mid k - odd\}$
- $(bb)^* : L_b = \{b^l \mid l - even\}$
- $a(aa)^*(bb)^* : L_{ab} = \{a^k b^l \mid k - odd, l - even\} = L_a L_b$
- $a^* : L = \{a^*\} = L_a^*$

Замкнутость регулярных языков относительно операций

Регулярные языки замкнуты ($A \in \mathbb{R}, B \in \mathbb{R} \Rightarrow A \diamond B \in \mathbb{R}$) относительно операций:

- Конкатенации ($L_1 L_2$), объединения ($L_1 \cup L_2$), итерации (L^*)
- Пересечения ($L_1 \cap L_2$), дополнения ($\neg L$), разности ($L_1 \setminus L_2$)
- Обращения ($L_{rev} = \{a_m, a_{m-1}, \dots, a_1 \mid a_1, a_2, \dots, a_m \in L\}$)
- Гомоморфизма цепочек (операция сохраняющая ε и конкатенацию)
- Обратного гомоморфизма цепочек

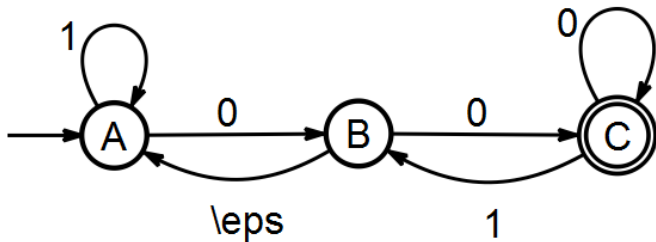
Теорема Клини

Теорема

Классы автоматных и регулярных языков эквивалентны

НКА с ε -переходами: почему бы и нет?

$$\delta: Q \times (\Sigma \cup \varepsilon) \rightarrow 2^Q$$

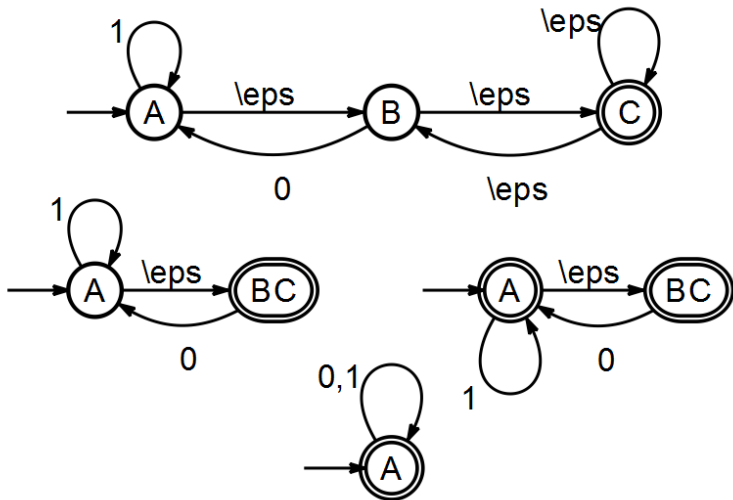


Ничего не поломалось?

Эквивалентность НКА с ε -переходами и НКА без ε -переходов

- НКА без ε -переходов — частный случай НКА с ε -переходами
- В обратную сторону — можно построить ε -замыкание
 - ▶ Транзитивное замыкание: для каждого подграфа, состоящего только из ε -переходов, делаем ε -замыкание
 - ▶ Добавление терминальных состояний: для ε -перехода из состояния u в v , где v — терминальное, добавляем u в терминальные
 - ▶ Добавление ребер: $\forall u, v, c, w. \delta(u, \varepsilon) = v, \delta(v, c) = w$, добавим переход $\delta(u, c) = w$
 - ▶ Устранение ε -переходов

ϵ -замыкание



Теорема Клини: доказательство \Leftarrow

Теорема

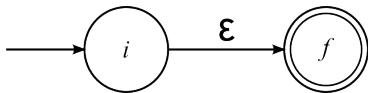
Классы автоматных и регулярных языков эквивалентны

Доказательство.

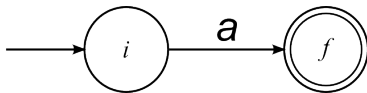
\Leftarrow : Построим по регулярному выражению КА (НКА с ε -переходами)



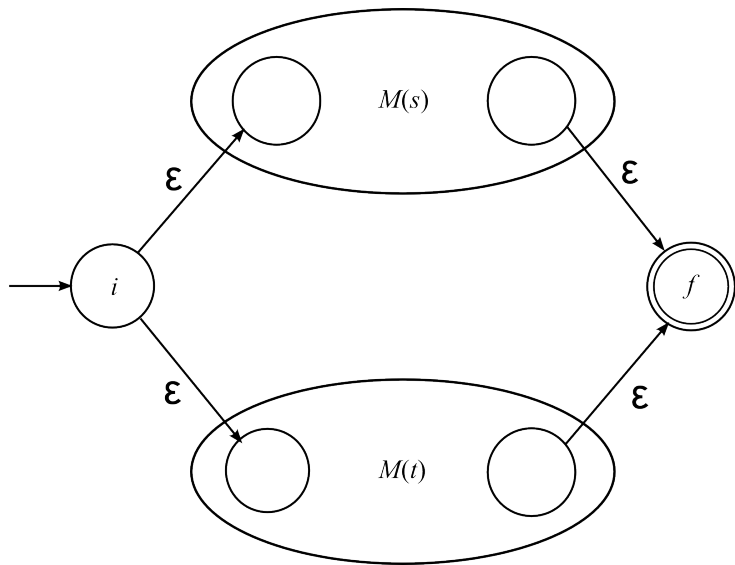
Построение КА по РВ: ε



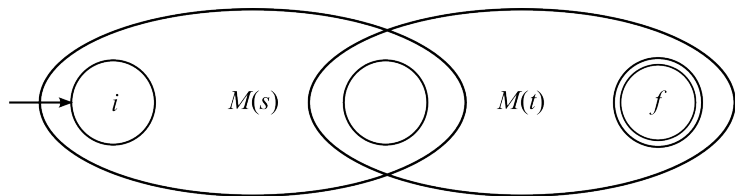
Построение КА по РВ: символ



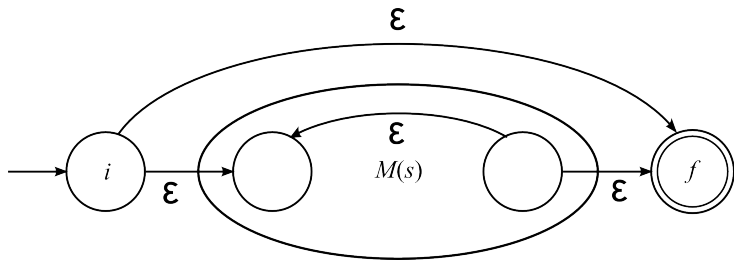
Построение КА по РВ: объединение $p|q$



Построение КА по РВ: конкатенация pq



Построение КА по РВ: итерация p^*



Теорема Клини: доказательство \Rightarrow

Теорема

Классы автоматных и регулярных языков эквивалентны

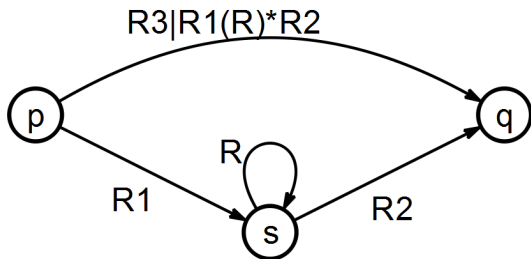
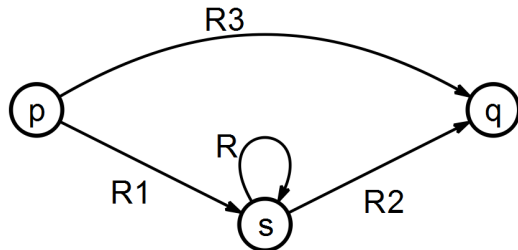
Доказательство.

\Rightarrow : Построим регулярное выражение по конечному автомату методом исключения состояний

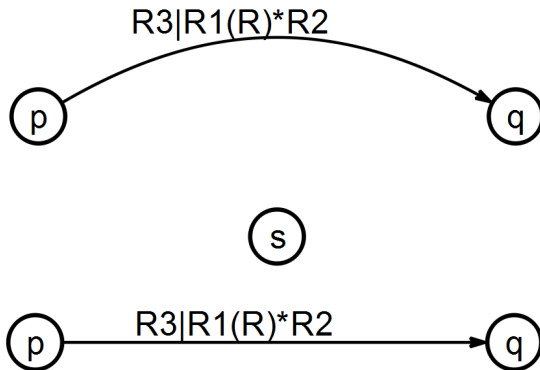
Идея: на ребрах пишем регулярные выражения, соответствующие путям между вершинами, последовательно исключаем состояния



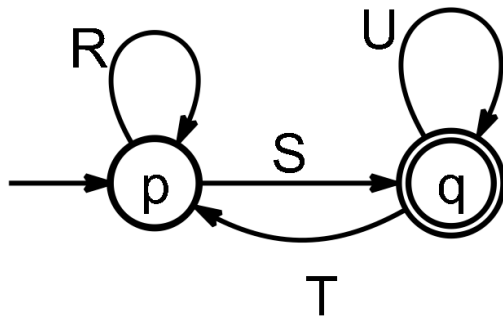
Исключение состояния s



Исключение состояния s : удаление ребер и вершины

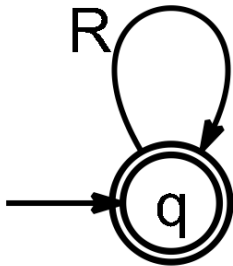


Исключение состояний: последний шаг



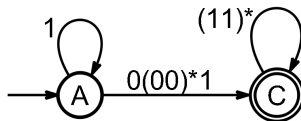
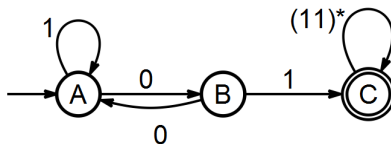
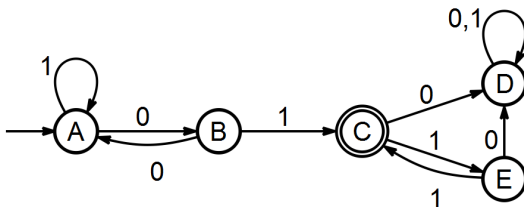
$$(R^* | SU^* T)^* SU^*$$

Исключение состояний: последний шаг



R^*

Исключение состояний: пример



$1*0(00)^*1(11)^*$

Свойства регулярных выражений

- $a|a = a$
- $a|\emptyset = a$
- $a|b = b|a$
- $a|(b|c) = (a|b)|c$
- $a(bc) = (ab)c$
- $\{\varepsilon\}a = a\{\varepsilon\} = \{\varepsilon\}$
- $\emptyset a = a\emptyset = \emptyset$
- $a(b|c) = ab|ac$
- $(a|b)c = ac|bc$
- $\{\varepsilon\}|aa^* \subseteq a^*$
- $\{\varepsilon\}|a^*a \subseteq a^*$
- $ab \subseteq b \Rightarrow a^*b \subseteq b$
- $ab \subseteq a \Rightarrow ab^* \subseteq a$

Регулярная грамматика

Праволинейная грамматика — грамматика, все правила которой имеют следующий вид:

- $A \rightarrow aB$ или $A \rightarrow a$, где $A, B \in V_N, a \in V_T$

Левوليнейная грамматика — грамматика, все правила которой имеют следующий вид:

- $A \rightarrow Ba$ или $A \rightarrow a$, где $A, B \in V_N, a \in V_T$

Регулярная грамматика

Праволинейная грамматика — грамматика, все правила которой имеют следующий вид:

- $A \rightarrow aB$ или $A \rightarrow a$, где $A, B \in V_N, a \in V_T$

Левوليнейная грамматика — грамматика, все правила которой имеют следующий вид:

- $A \rightarrow Ba$ или $A \rightarrow a$, где $A, B \in V_N, a \in V_T$

Теорема

Пусть L — формальный язык.

$\exists G_r$ — праволинейная грамматика, т.ч. $L = L(G_r) \Leftrightarrow \exists G_l$ — левوليнейная грамматика, т.ч. $L = L(G_l)$

Регулярная грамматика

Праволинейная грамматика — грамматика, все правила которой имеют следующий вид:

- $A \rightarrow aB$ или $A \rightarrow a$, где $A, B \in V_N, a \in V_T$

Левوليнейная грамматика — грамматика, все правила которой имеют следующий вид:

- $A \rightarrow Ba$ или $A \rightarrow a$, где $A, B \in V_N, a \in V_T$

Теорема

Пусть L — формальный язык.

$\exists G_r$ — праволинейная грамматика, т.ч. $L = L(G_r) \Leftrightarrow \exists G_l$ — левوليнейная грамматика, т.ч. $L = L(G_l)$

Регулярная грамматика — праволинейная или левوليнейная грамматика

Эквивалентность регулярной грамматики и НКА

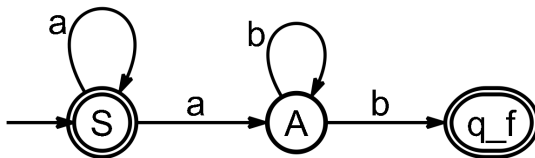
Алгоритм построения НКА $\langle Q, \Sigma, q_0, \delta, F \rangle$ по праволинейной грамматике $\langle V_T, V_N, P, S \rangle$

- $Q = V_N \cup \{q_f\}$
- $\forall (A \rightarrow aB) \in P. \delta(A, a) = B$
- $\forall (A \rightarrow a) \in P. \delta(A, a) = q_f$
- $q_0 = S$
- $\forall (B \rightarrow \varepsilon) \in P. B \in F$

Пример построения НКА по регулярной грамматике

$S \rightarrow aS \mid aA \mid \epsilon$

$A \rightarrow b \mid bA$

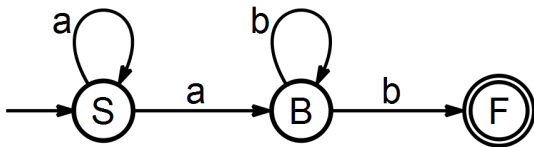


Эквивалентность регулярной грамматики и НКА

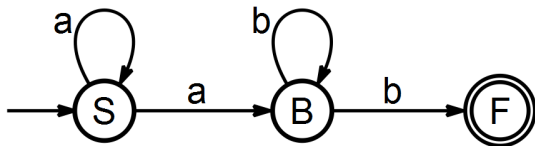
Алгоритм построения праволинейной грамматики $\langle V_T, V_N, P, S \rangle$ по НКА $\langle Q, \Sigma, q_0, \delta, F \rangle$

- $V_N = Q$
- $V_T = \Sigma$
- $\forall \delta(A, a) = B. (A \rightarrow aB) \in P$
- $\forall B \in F. (B \rightarrow \varepsilon) \in P$
- $S = q_0$
- Опционально: удалить ε -правила и бесполезные символы

Пример построения НКА по регулярной грамматике


$$S \rightarrow aS \mid aB$$
$$B \rightarrow bB \mid aF$$
$$F \rightarrow \varepsilon$$

Пример построения НКА по регулярной грамматике


$$S \rightarrow aS \mid aB$$
$$B \rightarrow bB \mid a$$

Лемма о разрастании (о накачке)

Теорема

L — регулярный язык над $\Sigma \Rightarrow \exists n. \forall \omega \in L, |\omega| > n$

$\exists x, y, z \in \Sigma^*. xyz = \omega, y \neq \varepsilon, |xy| \leq n,$

$\forall k \geq 0. xy^kz \in L$

Доказательство.

Строим автомат, распознающий L .

Обозначаем за n число состояний автомата.

Слово длины большей, чем n , обязано при разборе пройти через одно состояние дважды — получили цикл.

Метка цикла — искомое y , по циклу можно пройти сколько угодно раз.



Использование леммы о накачке

- $L = \{(^k)^k \mid k \geq 0\}$
- Предполагаем, что L — регулярный язык
- Берем n из леммы, рассматриваем слово $(^n)^n$
- Его можно разбить на $x y z$, $y \neq \varepsilon$, $|x y| \leq n$
- $|x y| \leq n \Rightarrow y = (^b, b > 0$
- Берем $k = 2$. $x y^k z = (^{n+b})^n$, что не принадлежит L
- Получили противоречие $\Rightarrow L$ не регулярен

- ДКА, НКА, НКА с ε -переходами, регулярные выражения, регулярные грамматики — все эти формализмы задают один класс (регулярных) языков и эквивалентны друг другу
- Проверка принадлежности слова регулярному языку осуществляется за $O(n)$ и не требует дополнительной памяти
- Класс регулярных языков обладает хорошими свойствами, прост и нагляден
- С помощью леммы о накачке можно доказать нерегулярность языка