Теория автоматов и формальных языков

Конечные автоматы

Лектор: Екатерина Вербицкая

НИУ-ВШЭ

12 сентября 2022

В предыдущей серии

- Контекст, в котором возникают формальные языки
- Метаязыки: БНФ, синтаксические диаграммы, формальные грамматики
- Вывод: транзитивное и рефлексивное замыкание отношения выводимости
  - Левосторонний (на каждом шаге заменяем самый левый нетерминал) и правосторонний
- Дерево вывода
  - Дерево: листья соответствуют терминалам, внутренние вершины — нетерминалам; для каждого внутреннего узла существует правило грамматики, правая часть которого совпадает с метками потомков узла
- Контекстно-свободная грамматика
  - все правила имеют вид $A \to \alpha$

Теоретико-множественное доказательство невозможности описания языков

Правда ли, что любой бесконечный язык можно представить конечным описанием?

Теоретико-множественное доказательство невозможности описания языков

Правда ли, что любой бесконечный язык можно представить конечным описанием?

Нет.

- Конечное описание — предложение над некоторым алфавитом (подразумеваемая интерпретация которого связывает его с описываемым языком)

Теоретико-множественное доказательство невозможности описания языков

Правда ли, что любой бесконечный язык можно представить конечным описанием?

Нет.

- Конечное описание — предложение над некоторым алфавитом (подразумеваемая интерпретация которого связывает его с описываемым языком)
- Любой язык является не более, чем счетным; соответственно существует не более, чем счетное множество конечных описаний

Теоретико-множественное доказательство невозможности описания языков

Правда ли, что любой бесконечный язык можно представить конечным описанием?

Нет.

- Конечное описание — предложение над некоторым алфавитом (подразумеваемая интерпретация которого связывает его с описываемым языком)
- Любой язык является не более, чем счетным; соответственно существует не более, чем счетное множество конечных описаний
- Множество всех языков над данным алфавитом не является счетным, так как множество всех подмножеств счетного множества более, чем счетно

Теоретико-множественное доказательство невозможности описания языков

Правда ли, что любой бесконечный язык можно представить конечным описанием?

Нет.

- Конечное описание — предложение над некоторым алфавитом (подразумеваемая интерпретация которого связывает его с описываемым языком)
- Любой язык является не более, чем счетным; соответственно существует не более, чем счетное множество конечных описаний
- Множество всех языков над данным алфавитом не является счетным, так как множество всех подмножеств счетного множества более, чем счетно
- Итого, конечных описаний меньше, чем языков; соответственно не для всех бесконечных языков существует конечное описание

Устройство компилятора ((очень) упрощенно)

- Препроцессор
- Лексический анализ
- Синтаксический анализ
- Семантический анализ
- Оптимизации
- Генерация кода

Лексический синтаксис

Определяет, как последовательность символов разбить на последовательность лексем

- Лексема — последовательность символов алфавита, имеющая собственный смысл
- Лексический анализ — процесс разбора последовательности символов на последовательность токенов
- Токен — лексема + служебная информация
  - Тип токена
  - Значение лексемы
  - Позиции начала и конца лексемы во входном потоке

Стандартные типы токенов

- Идентификатор: `ident`, `HumanBeing`, `x1`
  - Последовательность цифр и букв, начинающаяся на букву[1]
  - Не может быть ключевым словом[2]
- Ключевое слово: `for`, `if`, `begin`
- Разделитель: `,`, `(`, `)`, `[`, `]`
- Оператор: `+`, `:=`, `/=`
- Литерал
  - Число: `123.45e-6`
  - Булева константа: `true`
  - Строка: `"Hello world"`
  - Символ: `'c'`
- Комментарий
  - Однострочный: `// Отсюда и до конца строки`
  - Многострочный: `/* Весь текст между скобками */`

[1] Могут быть другие определения

[2] Не во всех языках

Ссылки на описания синтаксиса языков

- https://www.gnu.org/software/kawa/Lexical-syntax.html
- https://www.scala-lang.org/files/archive/spec/2.13/01-lexical-syntax.html
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Lexical_grammar
- https://docs.oracle.com/javase/specs/jls/se8/html/jls-3.html
- https://www.haskell.org/onlinereport/lexemes.html

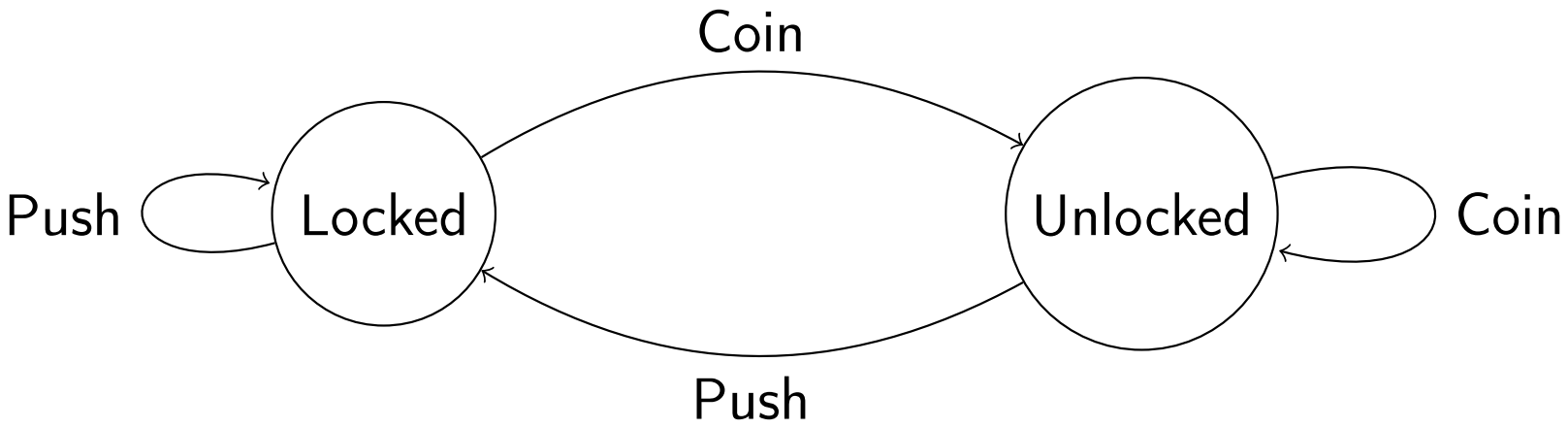
Лексер (лексический анализатор)

Программа, осуществляющая лексический анализ

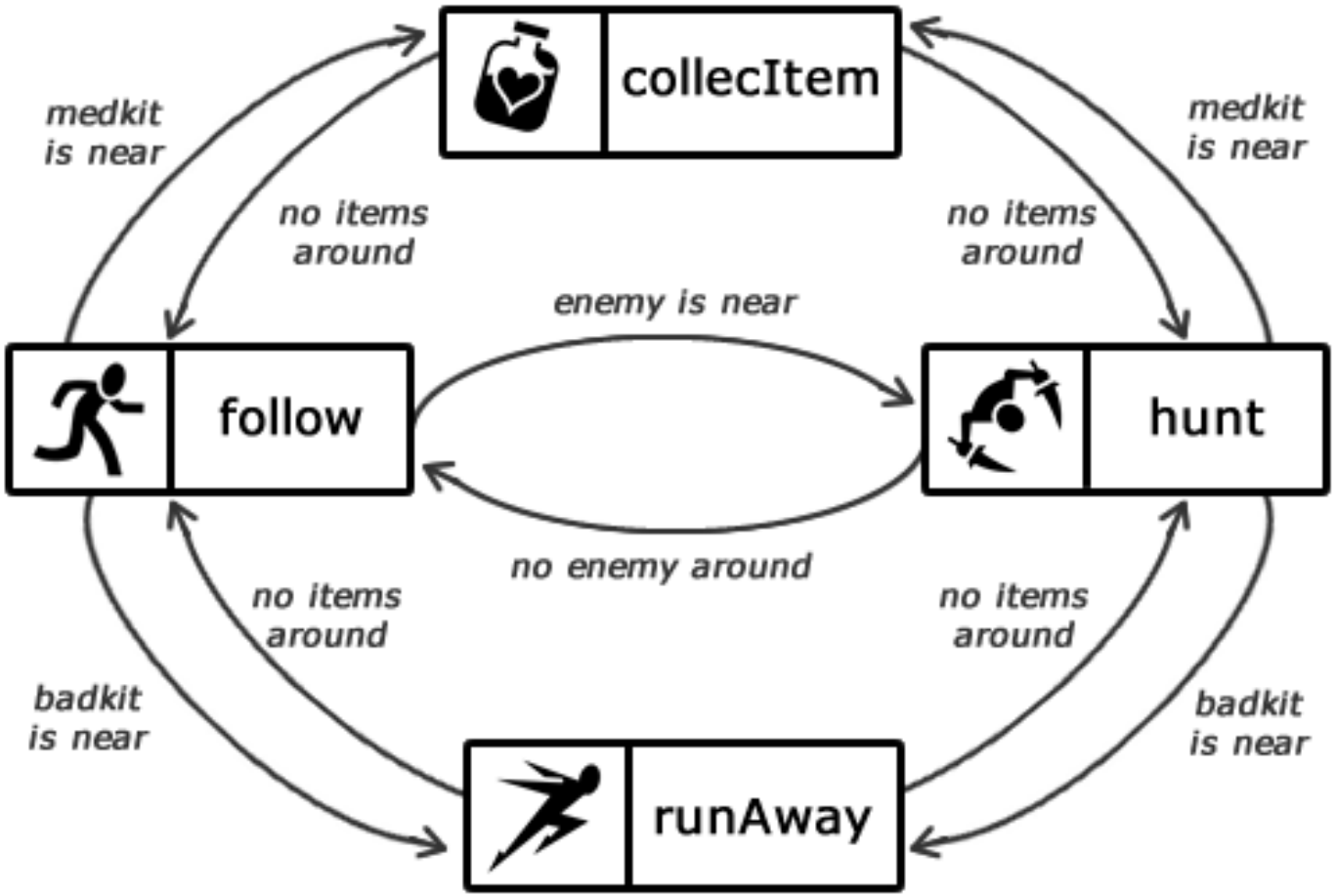
- На вход — строка
- На выход — последовательность токенов
- Фильтрует пробелы и комментарии (или нет)
- Следит за индентацией в чувствительных к индентации языках
- Сохраняет позиции лексем во входной строке
- Сообщает об ошибках

Часто лексический синтаксис описывается *автоматными* языками

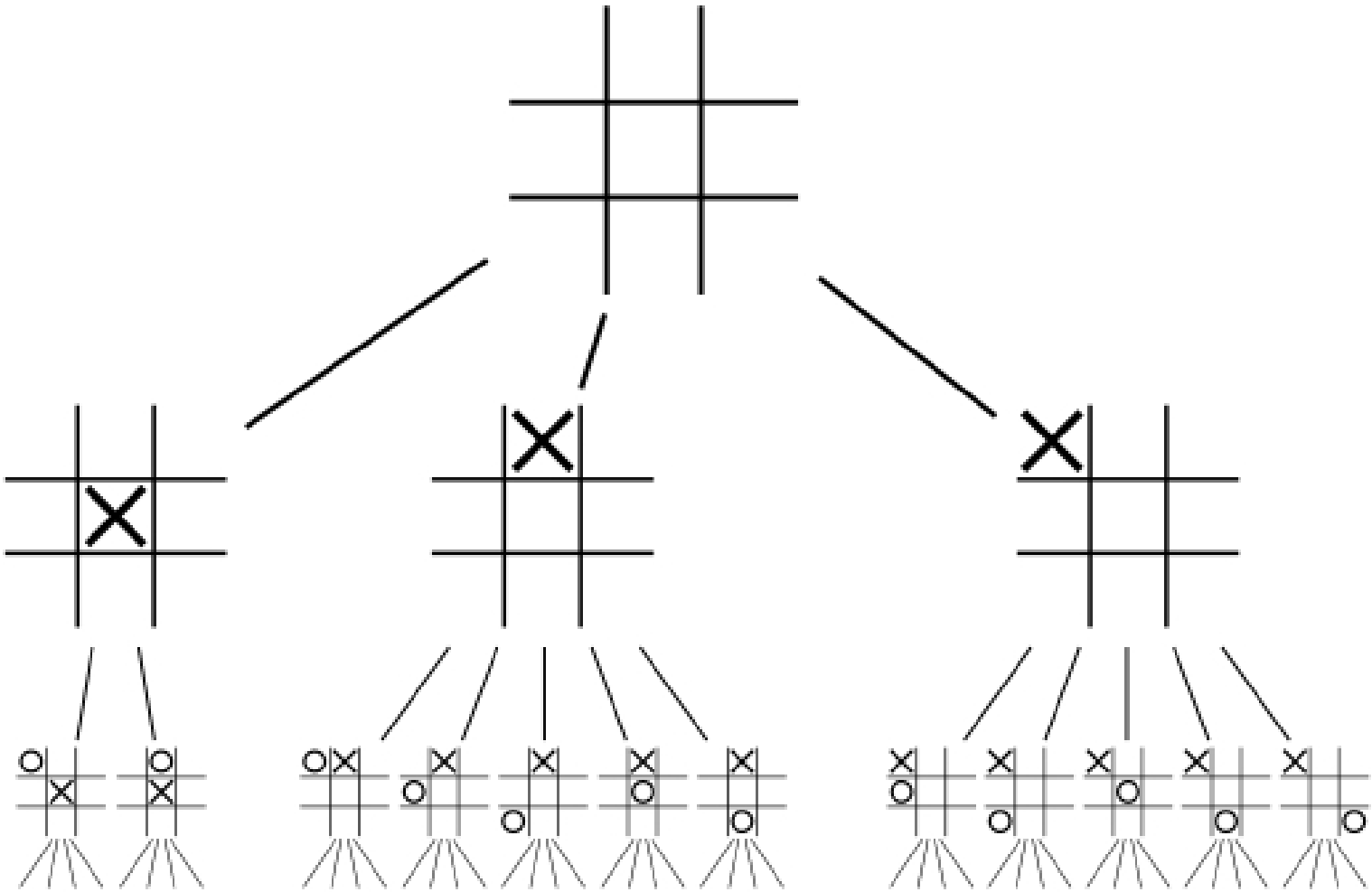
Конечные автоматы



Конечные автоматы



Конечные автоматы



Конечный автомат

(Детерминированный) конечный автомат — $\langle Q, \Sigma, \delta, q_0, F \rangle$

- $Q \neq \varnothing$ — конечное множество состояний
- Σ — Конечный входной алфавит
- δ — отображение типа $Q \times \Sigma \to Q$
  - $\delta(q_i, x) = q_j$
- $q_0 \in Q$ — начальное состояние
- $F \subseteq Q$ — множество конечных состояний

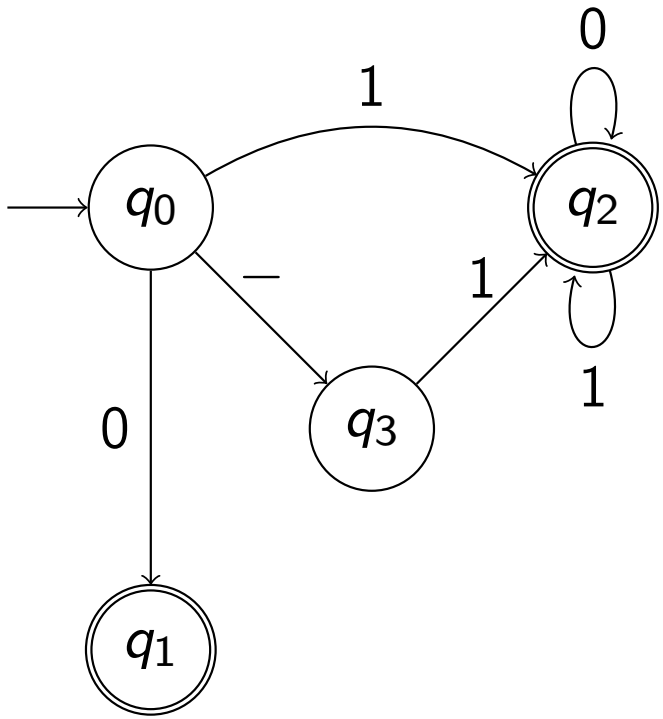
КА называется **полным**, если существует переход из каждого состояния по каждому символу алфавита

- Обычно добавляют “дьявольскую” вершину, она же сток.

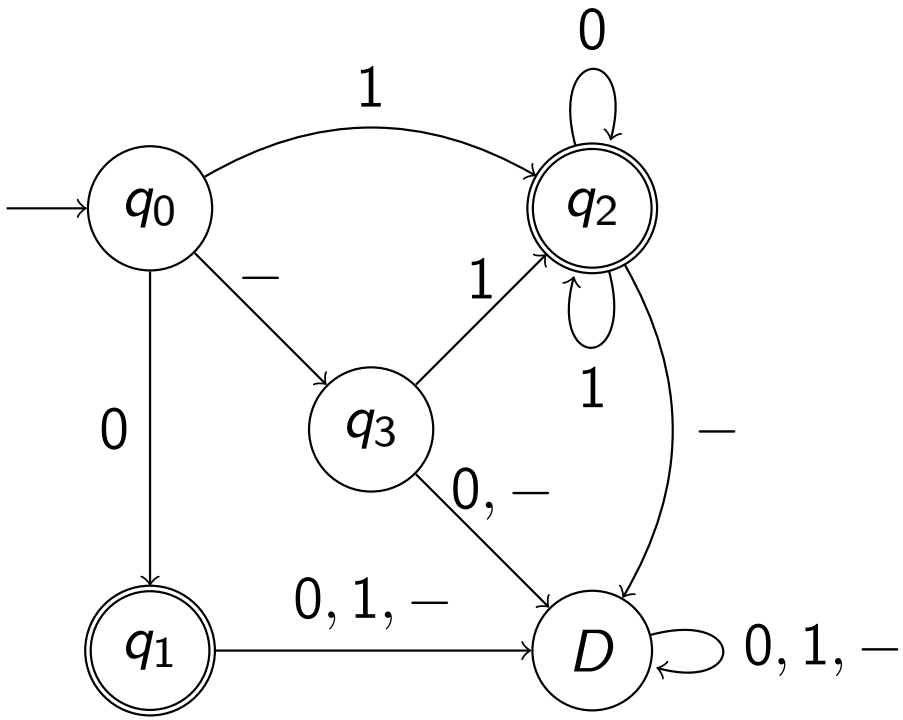
Пример конечного автомата

$$Q = \{q_0, q_1, q_2, q_3\}, \Sigma = \{0, 1, -\}, q_0 = q_0, F = \{q_1, q_2\}$$

$$\begin{aligned}
\delta(q_0, 0) &= q_1 \\
\delta(q_0, 1) &= q_2 \\
\delta(q_0, -) &= q_3 \\
\delta(q_2, 0) &= q_2 \\
\delta(q_2, 1) &= q_2 \\
\delta(q_3, 1) &= q_2
\end{aligned}$$



Пример полного конечного автомата



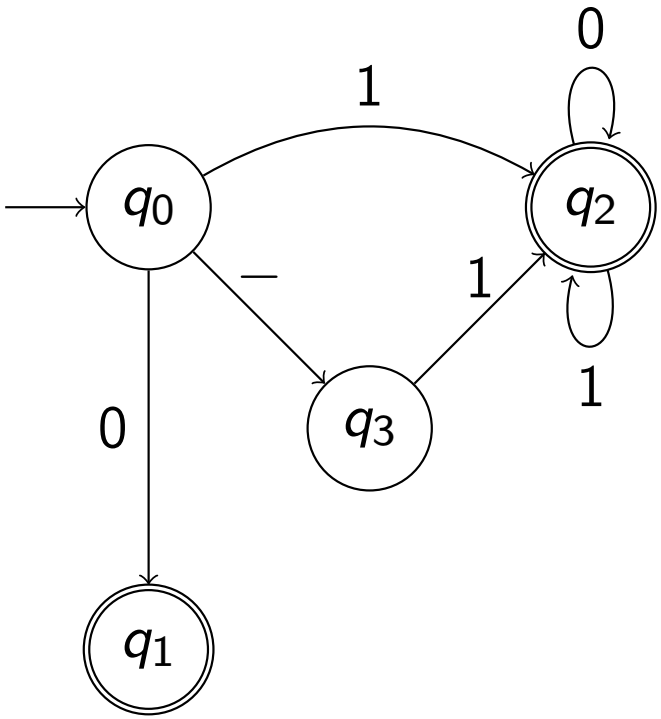
Путь в конечном автомате

- **Путь** — кортеж $\langle q_0, e_1, q_1, \ldots, e_n, q_n \rangle$
  - $n \geq 0$
  - $\forall i : e_i = \langle q_{i-1}, w_i, q_i \rangle$, где $\delta(q_{i-1}, w_i) = q_i$
  - q_0 — **начало** пути
  - q_n — **конец** пути
  - $w_1, w_2, \ldots, w_n$ — **метка** пути
  - n — **длина** пути
- Путь **успешен**, если q_0 — начальное состояние, а $q_n \in F$
- Состояние q **достижимо** из состояния p, если существует путь из состояния p в состояние q

Пример пути

Успешный путь с меткой -110 длины 4

$$\langle q_0, \langle q_0, -, q_3 \rangle, q_3, \langle q_3, 1, q_2 \rangle, q_2, \langle q_2, 1, q_2 \rangle, q_2, \langle q_2, 0, q_2 \rangle, q_2 \rangle$$



Такт работы КА (шаг)

- **Конфигурация (Мгновенное описание)** КА — $\langle q, \omega \rangle$, где $q \in Q, \omega \in \Sigma^*$
- **Такт работы** — бинарное отношение $\vdash$: если $\delta(p, x) = q$ и $\omega \in \Sigma^*$, то $\langle p, x\omega \rangle \vdash \langle q, \omega \rangle$
- Бинарное отношение $\vdash^*$ — рефлексивное, транзитивное замыкание $\vdash$

Распознавание слова конечным автоматом

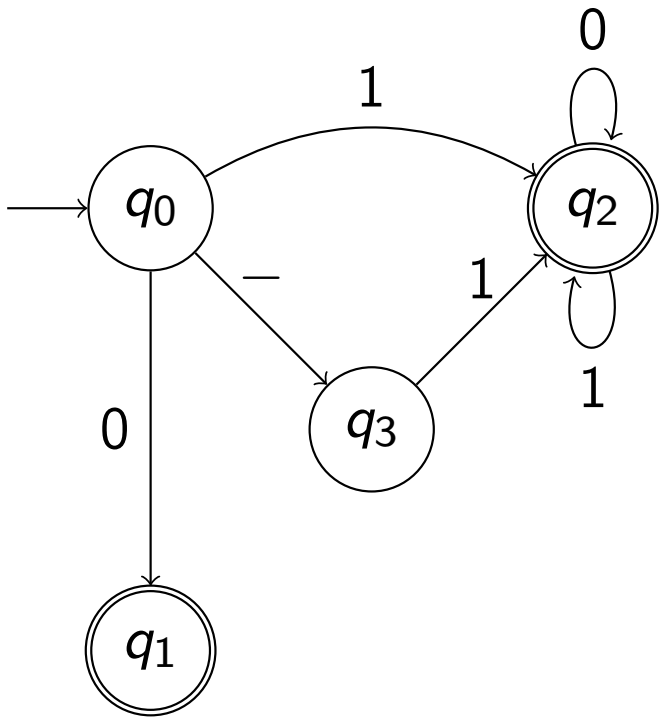
Цепочка ω **распознается** КА, если $\exists$ успешный путь с меткой ω

Язык, распознаваемый конечным автоматом:
$\{\omega \in \Sigma^* \mid \exists p$ — успешный путь с меткой $\omega\}$

Распознавание слова конечным автоматом: пример

$$\{\ldots, -110, -101, -100, -11, -10, -1, 0, 1, 10, 11, 100, 101, 110, \ldots\}$$

Язык всех целых чисел в двоичной записи



Распознавание слова конечным автоматом

Теорема

Рассмотрим конечный автомат $M = \langle Q, \Sigma, \delta, q_0, F \rangle$.
Слово $\omega \in \Sigma^*$ *принадлежит языку* $L(M) \Leftrightarrow \exists q \in F : \langle q_0, \omega \rangle \vdash^* \langle q, \varepsilon \rangle$.

Распознавание слова конечным автоматом

Обобщаем функцию перехода:

- $\delta'(q, \varepsilon) = q$
- $\delta'(q, x\alpha) = \delta'(\delta(q, x), \alpha)$, где $x \in \Sigma, \alpha \in \Sigma^*$

Теорема

Цепочка ω ***распознается*** *КА* $\langle Q, \Sigma, \delta, q_0, F \rangle \Leftrightarrow \exists p \in F : \delta'(q_0, \omega) = p$

Язык, распознаваемый конечным автоматом:

$$\{\omega \in \Sigma^* \mid \exists p \in F : \delta'(q_0, \omega) = p\}$$

Свойство конкатенации строк

Теорема

$\langle q_1, \alpha \rangle \vdash^* \langle q_2, \varepsilon \rangle, \langle q_2, \beta \rangle \vdash^* \langle q_3, \varepsilon \rangle \rightarrow \langle q_1, \alpha\beta \rangle \vdash^* \langle q_3, \varepsilon \rangle$

Эквивалентность конечных автоматов

Конечные автоматы A_1 и A_2 **эквивалентны**, если распознают один и тот же язык

Как проверить что автоматы эквиваленты?

Проверка на эквивалентность автоматов

- Запустить одновременный обход в ширину двух автоматов
- Каждый переход должен приводить в терминальные или нетерминальные вершины в обоих автоматах соответственно

Минимальный конечный автомат

Минимальный конечный автомат — автомат, имеющий наименьшее число состояний, распознающий тот же язык, что и данный

Классы эквивалентности

Отношение эквивалентности — рефлексивное, симметричное, транзитивное отношение

- xRx
- $xRy \Leftrightarrow yRx$
- $xRy, yRz \rightarrow xRz$

Теорема

$\forall R$ — отношение эквивалентности на множестве S
Можно разбить S на k непересекающихся подмножеств $I_1 \dots I_k$, т.ч. $aRb \Leftrightarrow a, b \in I_j$

Множества $I_1 \dots I_k$ называются **классами эквивалентности**

Эквивалентные состояния

- $\omega \in \Sigma^*$ **различает** состояния q_i и q_j, если
  $\delta'(q_i, \omega) = t_1, \delta'(q_j, \omega) = t_2 \to (t_1 \notin F \Leftrightarrow t_2 \in F)$
- q_i и q_j **эквивалентны** $(q_i \sim q_j)$, если
  $\forall \omega \in \Sigma^* : \delta'(q_i, \omega) = t_1, \delta'(q_j, \omega) = t_2 \to (t_1 \in F \Leftrightarrow t_2 \in F)$
  - Является отношением эквивалентности

Лемма

$\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle, p_1, p_2, q_1, q_2 \in Q, q_i = \delta(p_i, c)$
$\omega \in \Sigma^*$ *различает* q_1 *и* q_2. *Тогда* $c\omega$ *различает* p_1 *и* p_2

Доказательство

$\delta'(p_i, c\omega) = \delta'(\delta(p_i, c), \omega) = \delta'(q_i, \omega) = t_i$

Алгоритм минимизации КА

TLDR: разбиваем состояния на классы эквивалентности, которые делаем новыми состояниями

Алгоритм минимизации КА

Q — очередь

marked — таблица размером $n \times n$ (n — количество состояний КА).

Помечаем в таблице пары неэквивалентных состояний и кладем их в очередь

Алгоритм минимизации КА

- Если автомат не полный — дополнить дьявольской вершиной
- Строим отображение δ^{-1} — обратные ребра
- Находим все достижимые из стартового состояния
- Добавляем в Q и отмечаем в *marked* пары состояний, различимые ε
- Можем пометить пару (u, v), если $\exists c \in \Sigma : (\delta(u, c), \delta(v, c))$. Для этого, пока $Q \neq \varnothing$:
  - Извлекаем (u, v) из Q
  - $\forall c \in \Sigma$ перебираем $(\delta^{-1}(u, c), \delta^{-1}(v, c))$ — если пара не помечена, помечаем и кладем в очередь
- В момент опустошения Q непомеченные пары являются эквивалентными
- За проход по таблице выделяем классы эквивалентности
- За проход по таблице формируем новые состояния и переходы

Алгоритм минимизации КА

- Стартовое состояние — класс эквивалентности, которому принадлежит стартовое состояние исходного КА
- Конечные состояния — классы эквивалентности, которым принадлежат конечные состояния исходного КА

Алгоритм минимизации КА: корректность

- Пусть в результате применения алгоритма к КА A получили КА A_{min}. Покажем, что этот автомат минимальный и единственный с точностью до изоморфизма
- Пусть $\exists A' : A'$ и A эквивалентны, но количество состояний A' меньше, чем у A_{min}
- Стартовые состояния $s \in A_{min}$ и $s' \in A'$ эквивалентны (КА допускают один язык)
- $\lhd \alpha = a_1 a_2 \dots a_k, a_i \in \Sigma : \langle s, \alpha \rangle \vdash^* \langle u, \varepsilon \rangle; \langle s', \alpha \rangle \vdash^* \langle u', \varepsilon \rangle$
- $\lhd \langle s, a_1 \rangle \vdash^* \langle l, \varepsilon \rangle; \langle s', a_1 \rangle \vdash^* \langle l', \varepsilon \rangle$. s, s' эквивалентны $\rightarrow l, l'$ эквивалентны
- Аналогично для всех $a_i \Vdash u, u'$ эквивалентны
- $\Vdash \forall q$ — состояние $A_{min} \exists q'$ — эквивалентное состояние A'
- Состояний A' меньше, чем состояний $A_{min} \rightarrow$ 2 состояниям A_{min} соответствует 1 состояние $A' \rightarrow$ они эквивалентны. Но по построению A_{min} в нем не может быть эквивалентных состояний. Противоречие

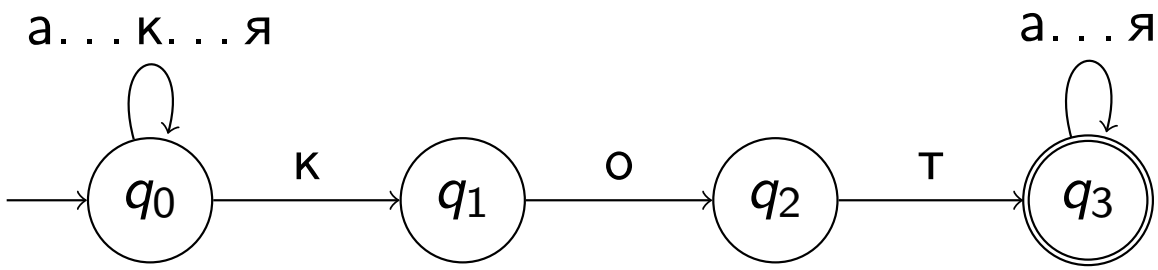
Недетерминированный КА

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

- $Q \neq \varnothing$ — конечное множество состояний
- Σ — Конечный входной алфавит
- δ — отображение типа $Q \times \Sigma \to 2^Q$
  - $\delta(q_i, x) = \{q_{j_0}, \ldots, q_{j_k}\}$
- $q_0 \in Q$ — начальное состояние
- $F \subseteq Q$ — множество конечных состояний

Недетерминированный КА: пример

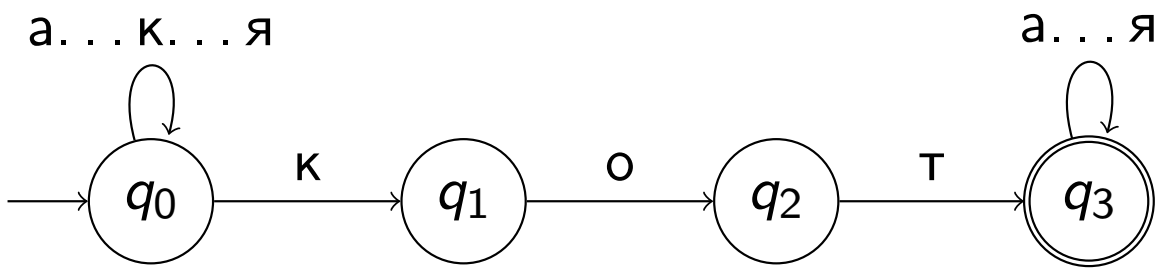
$$
\begin{aligned}
\delta(q_0, \text{а}) &= q_0 \\
&\dots \\
\delta(q_0, \text{к}) &= q_0 \\
&\dots \\
\delta(q_0, \text{я}) &= q_0 \\
\delta(q_0, \text{к}) &= q_1 \\
\\
\delta(q_1, \text{о}) &= q_2 \\
\\
\delta(q_2, \text{т}) &= q_3 \\
\\
\delta(q_3, \text{а}) &= q_3 \\
&\dots \\
\delta(q_3, \text{я}) &= q_3
\end{aligned}
$$



Распознавание слова НКА

- **Конфигурация (Мгновенное описание)** КА — $\langle q, \omega \rangle$, где $q \in Q, \omega \in \Sigma^*$
- **Такт работы** — бинарное отношение $\vdash$: если $q \in \delta(p, x)$ и $\omega \in \Sigma^*$, то $\langle p, x\omega \rangle \vdash \langle q, \omega \rangle$
- Бинарное отношение $\vdash^*$ — рефлексивное, транзитивное замыкание $\vdash$
- НКА **допускает** слово α, если $\exists t \in F : \langle s, \alpha \rangle \vdash^* \langle t, \varepsilon \rangle$
- **Язык НКА** $L(A) = \{\omega \in \Sigma^* \mid \exists t \in F : \langle s, \omega \rangle \vdash^* \langle t, \varepsilon \rangle\}$
- ДКА — частный случай НКА

Недетерминированный КА: пример



{**кот**, с**кот**, **кот**лета, мя**кот**ь, антре**кот**. . . }

Алгоритм, определяющий допустимость слова

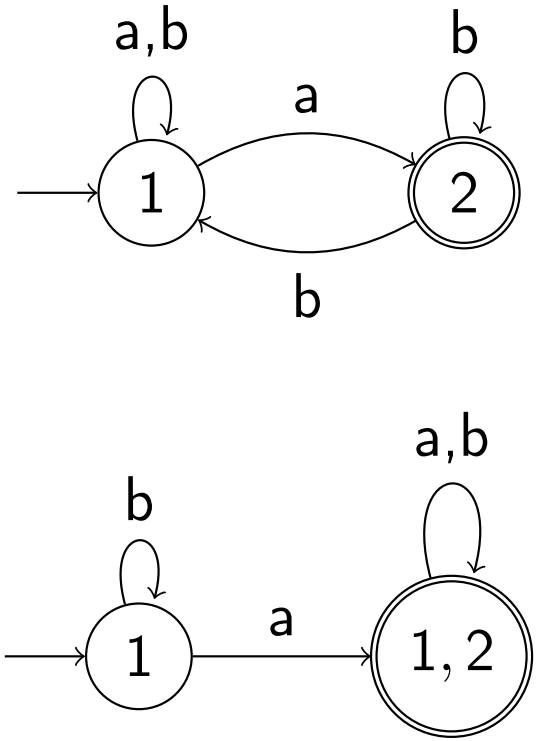
$$
\begin{aligned}
R(\alpha) &= \{p \mid \langle q_0, \alpha \rangle \vdash^* \langle p, \varepsilon \rangle\} \\
\\
R(\varepsilon) &= \{q_0\} \\
R(\alpha c) &= \{q \mid q \in \delta(p, c), p \in R(\alpha)\}
\end{aligned}
$$

НКА допускает слово $\alpha \Leftrightarrow \exists t \in F : t \in R(\alpha)$

Построение ДКА по НКА

- Помещаем в *Queue* множество $\{q_0\}$
- Пока очередь не пуста, выполняем:
  - $q = Queue.pop()$
  - Строим множество $q' = \{t = \delta(s, c) \mid s \in q, c \in \Sigma\}$. Если $q' \notin Queue$, добавить его в очередь. Каждое такое множество — новая вершина ДКА; добавляем переходы по соответствующим символам
  - Если во множестве есть хотя бы одна вершина, являющаяся терминальной в данном НКА, то соответствующая вершина ДКА будет конечной
- Результат: $\langle \Sigma, Q_d, q_{d_0} \in Q_d, F_d \subset Q_d, \delta_d : Q_d \times \Sigma \to Q_d \rangle$
  - $Q_d = \{q_d \mid q_d \subset 2^Q\}$
  - $q_{d_0} = \{q_0\}$
  - $F_d = \{q \in Q_d \mid \exists p \in F : p \in q\}$
  - $\delta_d(q, c) = \{\delta(a, c) \mid a \in q\}$

Детерминизация НКА: пример



Эквивалентность языков, распознаваемых ДКА и НКА

Теорема

ДКА и НКА распознают один и тот же класс языков

Доказательство.

$\Rightarrow$: очевидно

$\Leftarrow$: Рассмотрим произвольный НКА и покажем, что алгоритм строит по нему эквивалентный ДКА.

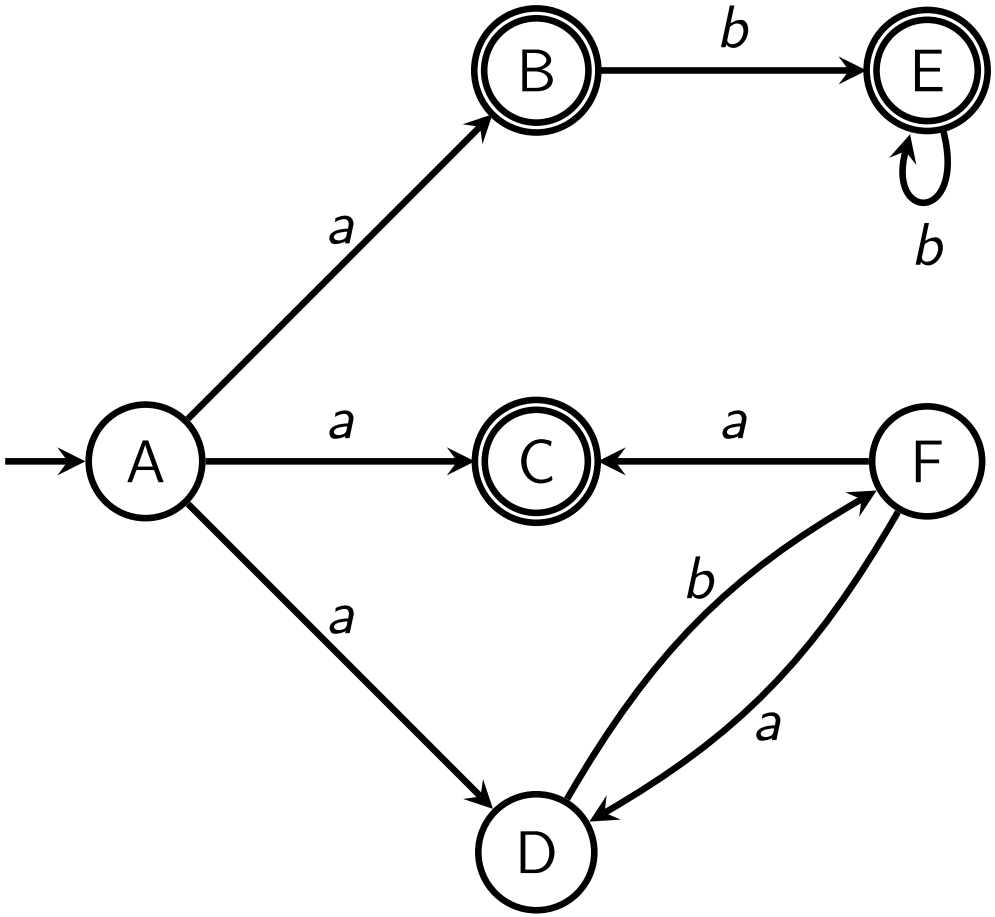
$\forall q \in q_d, \forall c \in \Sigma, \forall p \in \delta(q, c) : p \in \delta_d(q_d, c)$

Рассмотрим $\langle q_0, w_1 w_2 \dots w_m \rangle \vdash \langle u_1, w_2 \dots w_m \rangle \vdash^* \langle u_m, \varepsilon \rangle, u_m \in F$

$\forall i : u_i \in u_{d_i}$, где $(q_{d_0}, w_1 w_2 \dots w_m) \vdash (u_{d_1}, w_2 \dots w_m) \vdash^* (u_{d_m}, \varepsilon)$

$\Vdash u_m \in u_{d_m}$ $\square$

Распознание слова НКА



Слово распознается за $O(|\omega| \sum_{t \in Q} \sum_{c \in \Sigma} |\delta(t, c)|)$

Произведение автоматов

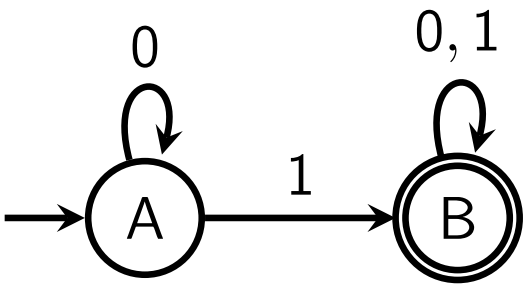
$A_1 = \langle \Sigma_1, Q_1, q_{1_0}, \delta_1, F_1 \rangle$ и $A_2 = \langle \Sigma_2, Q_2, q_{2_0}, \delta_2, F_2 \rangle$ — КА

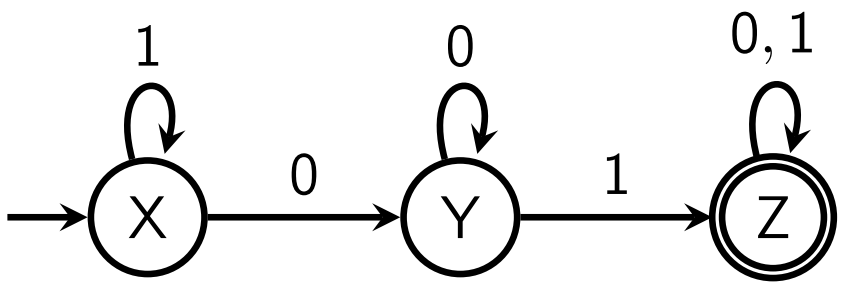
Произведением автоматов назовем $A = \langle \Sigma, Q, q_0, \delta, F \rangle$, где

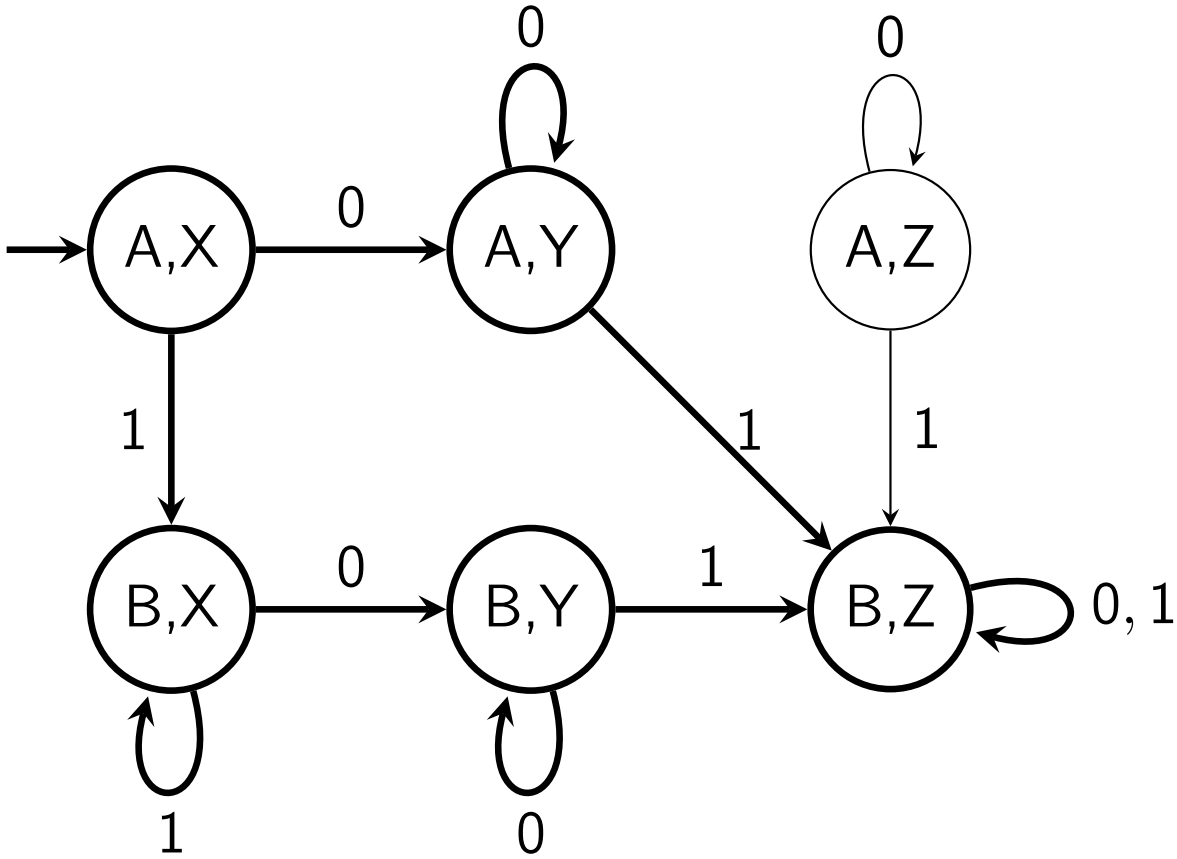
- $\Sigma = \Sigma_1 \cup \Sigma_2$
- $Q = Q_1 \times Q_2$
- $q_0 = (q_{1_0}, q_{2_0})$
- $F \subseteq Q$
  - $F = F_1 \times F_2$ — распознает **пересечение** языков
  - $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$ — распознает **объединение** языков
  - $F = F_1 \times (Q_2 \setminus F_2)$ — распознает **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

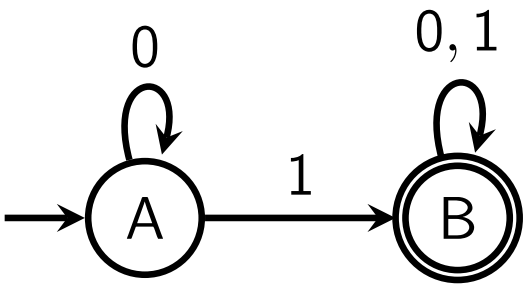
Произведение автоматов: пример

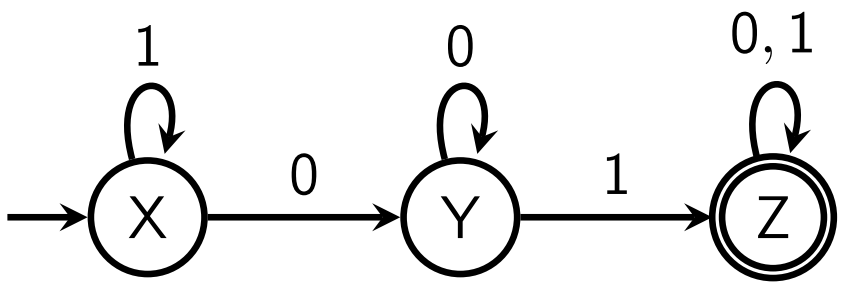


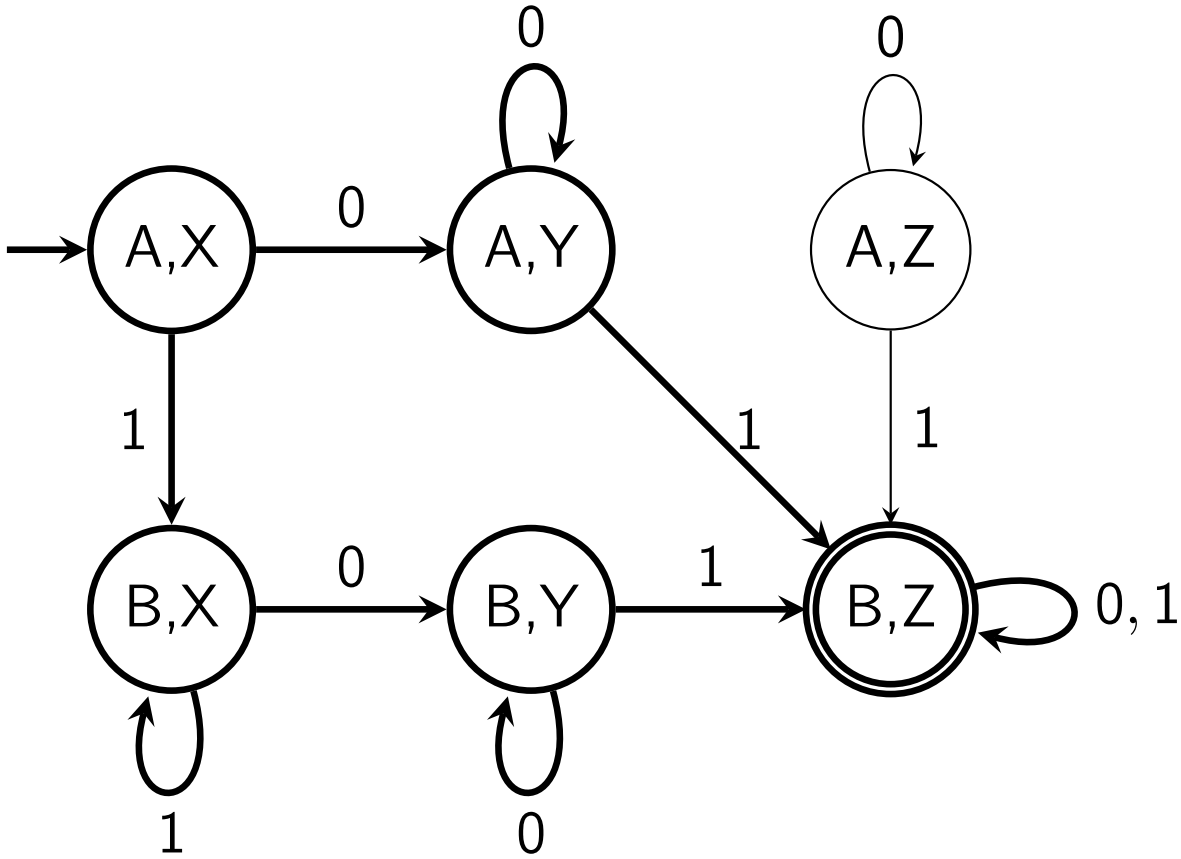




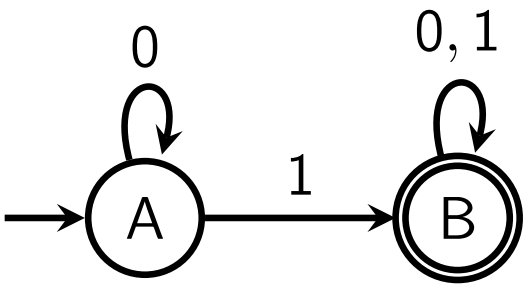
Пересечение языков

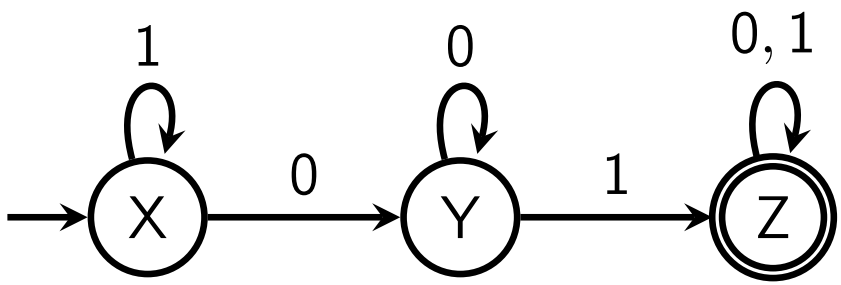


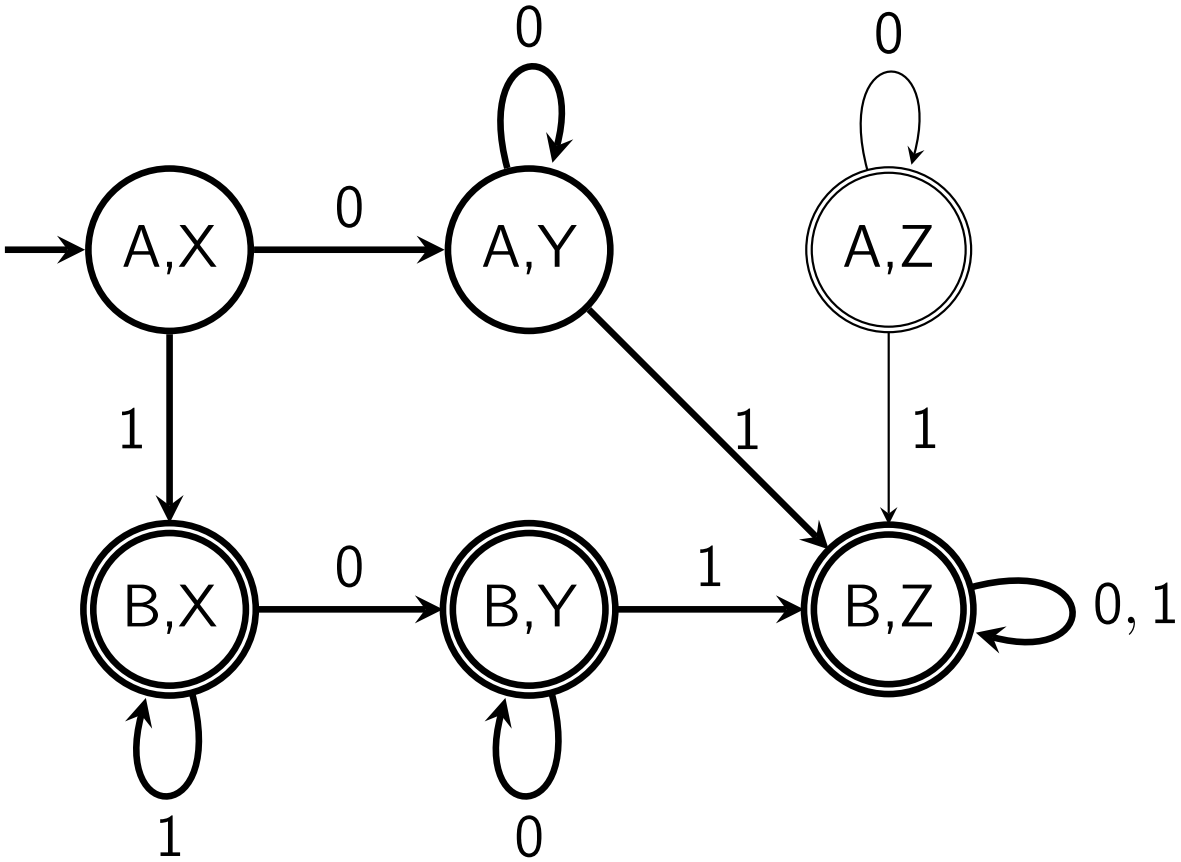




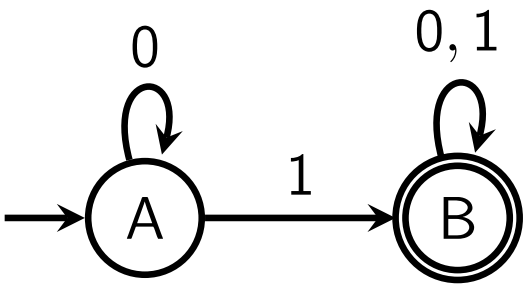
Объединение языков

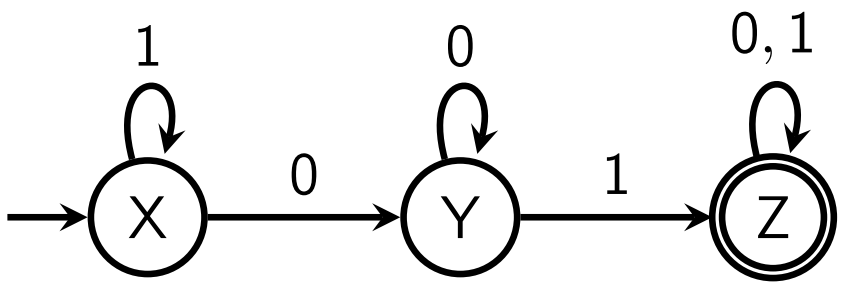


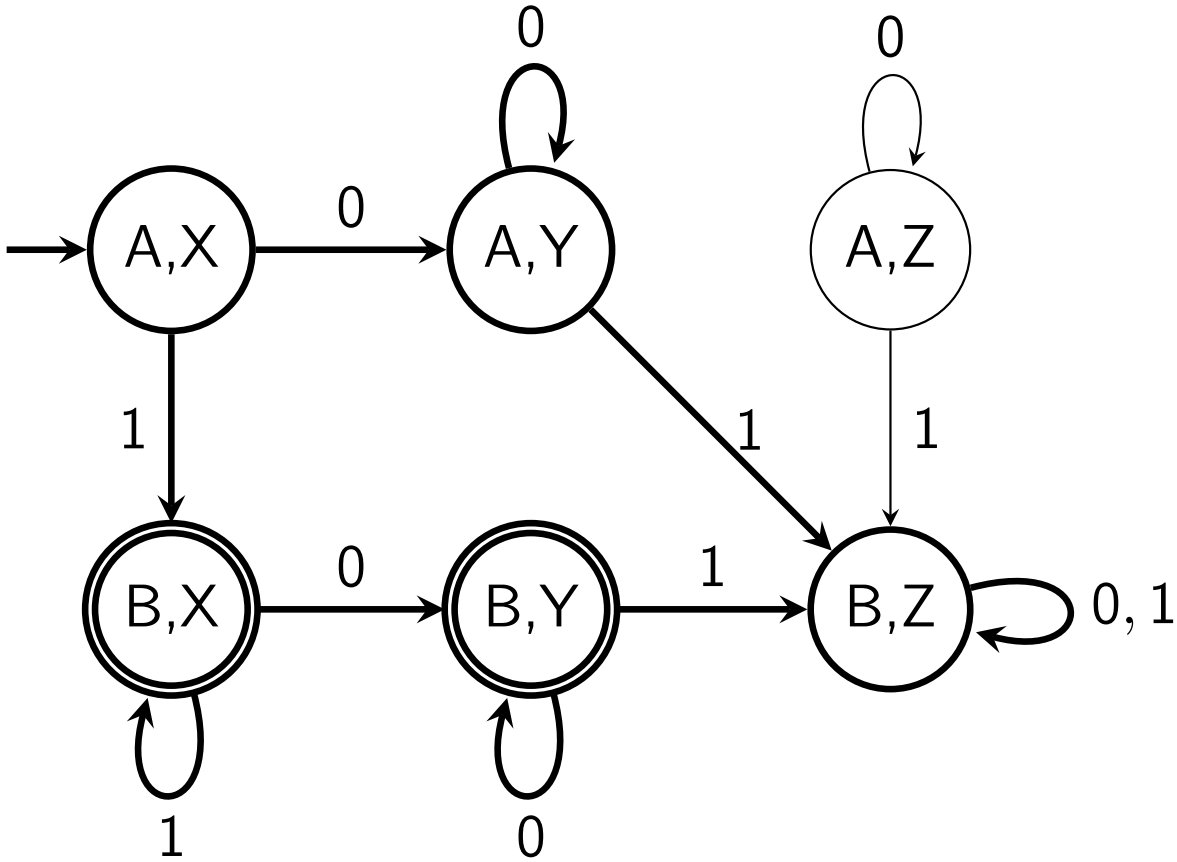




Разность языков







Замкнутость автоматных языков относительно операций

Автоматные языки замкнуты относительно операций:

- Объединения
- Пересечения
- Разности
- Дополнения
  - $\overline{X} = \Sigma^* \setminus X$

Регулярное множество (регулярный язык)

Регулярное множество в алфавите Σ определяется итеративно:

- $\varnothing$ — 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)
  - $PQ = \{pq \mid p \in P, q \in Q\}$ (конкатенация)
  - $P^* = \{\varepsilon\} \cup P \cup PP \cup PPP \cup \ldots$ (итерация)
- Ничто другое не является регулярным множеством в алфавите Σ
- Множество всех регулярных языков обозначим $\mathbb{R}$

Примеры регулярных языков

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

Регулярное выражение

Регулярное выражение — способ записи регулярного множества

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

Примеры регулярных выражений

- $-2147483648 \mid -2147483647 \mid \cdots \mid 2147483647$ — 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 ($L_{rev} = \{\omega^R = a_m a_{m-1} \dots a_1 \mid a_1 a_2 \dots a_m = \omega \in L\}$)
- Гомоморфизма цепочек ϕ
  - $\phi(\varepsilon) = \varepsilon$
  - $\phi(\alpha\beta) = \phi(\alpha)\phi(\beta)$
- Обратного гомоморфизма цепочек

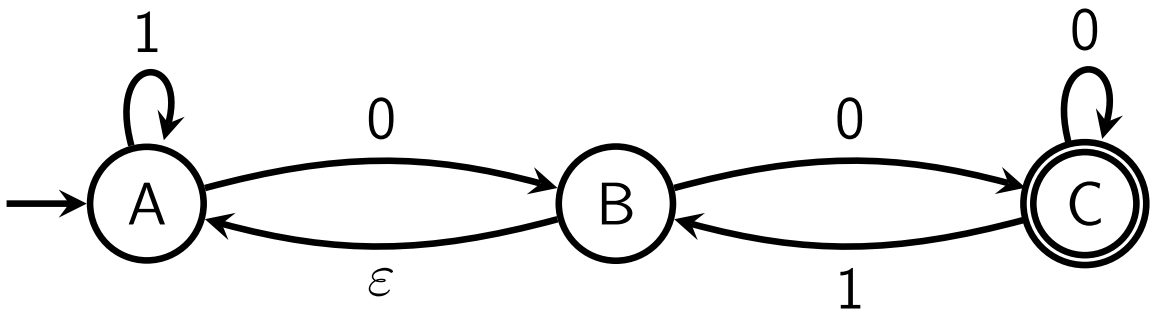
Теорема Клини

Теорема

Классы автоматных и регулярных языков эквивалентны

НКА с ε-переходами: почему бы и нет?

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

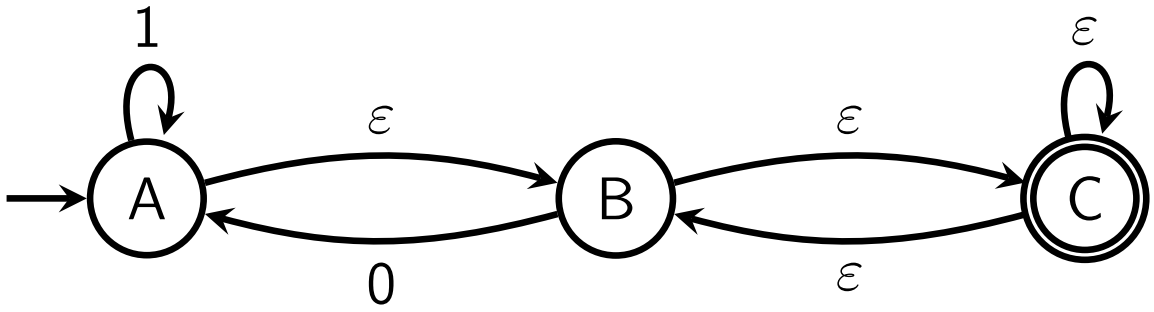


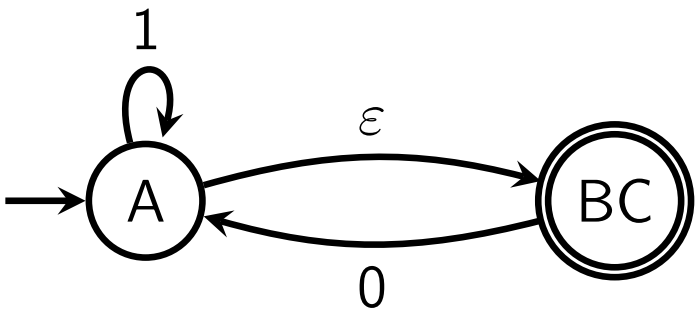
Ничего не поломалось?

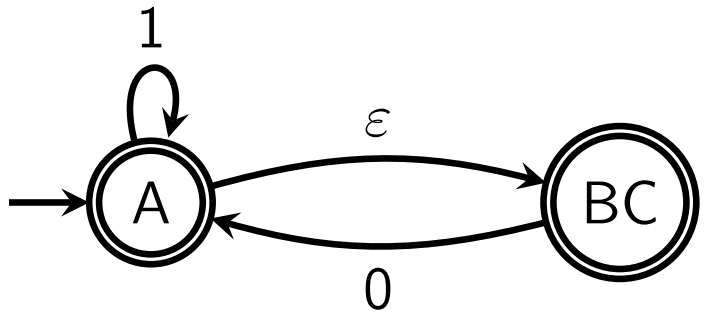
Эквивалентность НКА с ε-переходами и НКА без ε-переходов

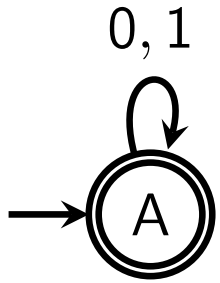
- НКА без ε-переходов — частный случай НКА с ε-переходами
- В обратную сторону — можно построить ε-замыкание
  - Транзитивное замыкание: для каждого подграфа, состоящего только из ε-переходов, делаем ε-замыкание
  - Добавление терминальных состояний: для ε-перехода из состояния u в v, где v — терминальное, 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 $\Leftarrow$

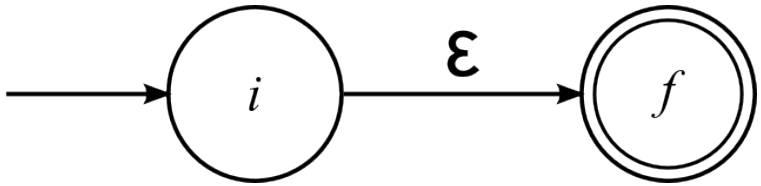
Теорема

Классы автоматных и регулярных языков эквивалентны

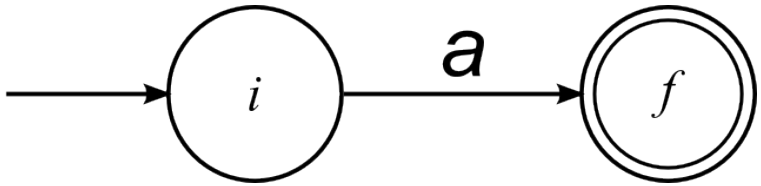
Доказательство.

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

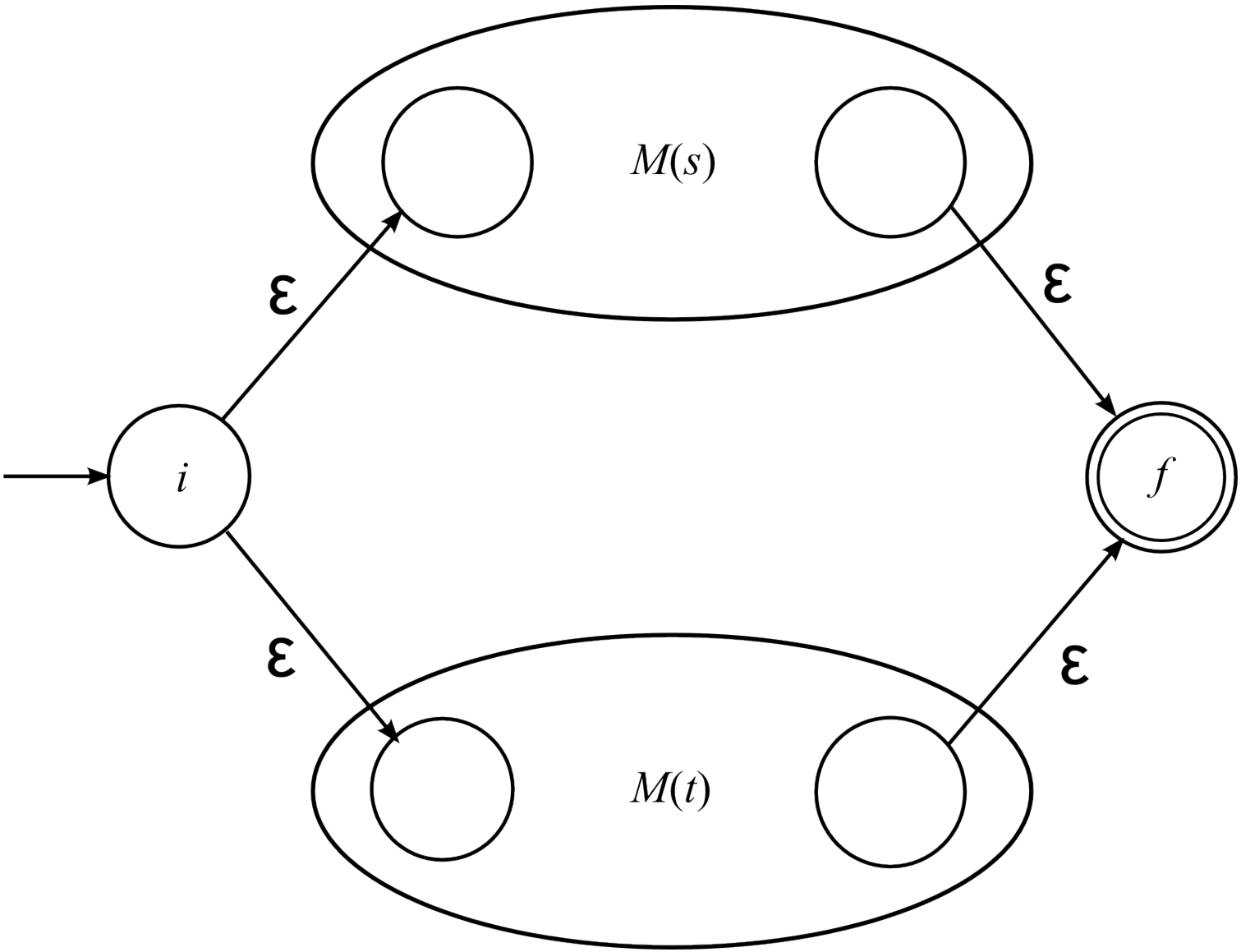
Построение КА по РВ: ε



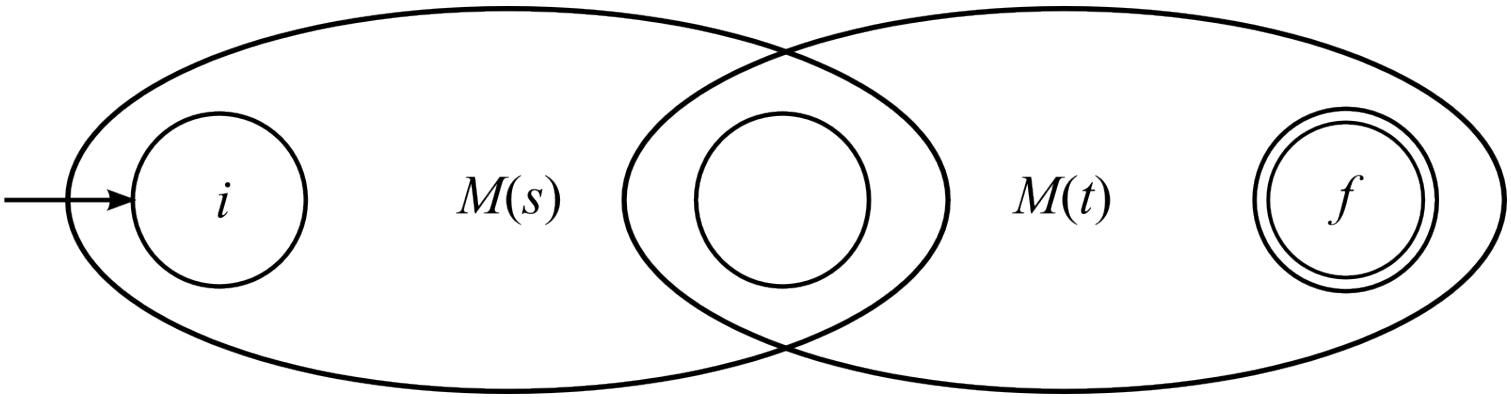
Построение КА по РВ: символ



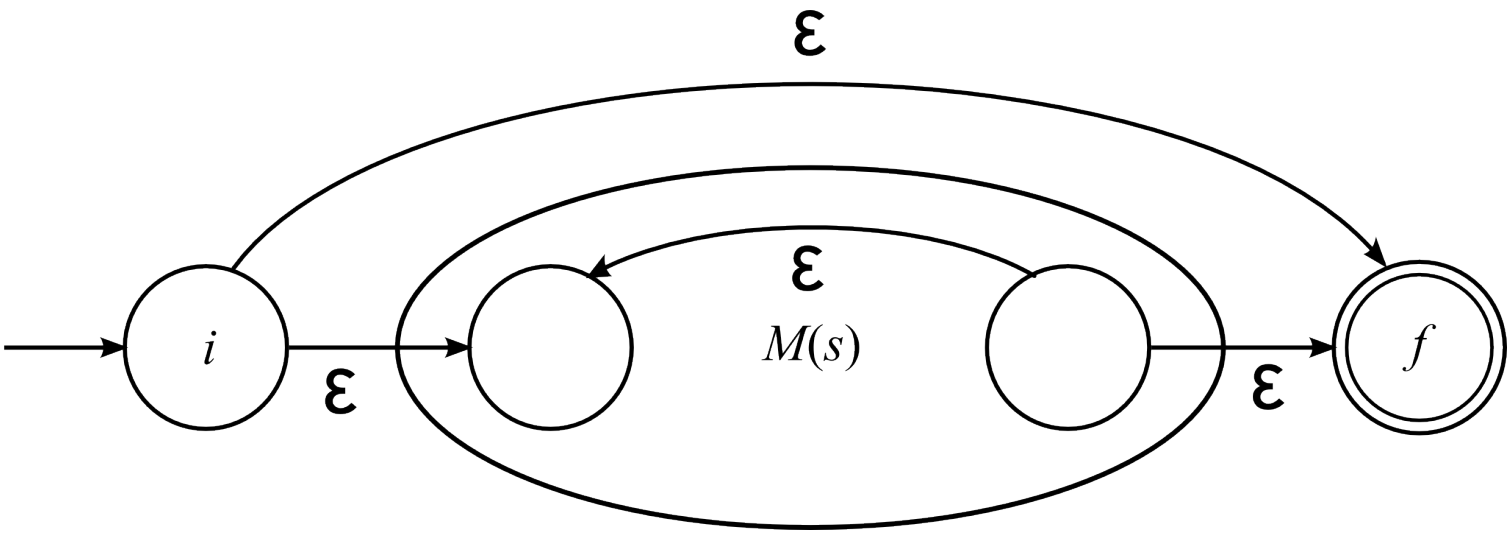
Построение КА по РВ: объединение $p \mid q$



Построение КА по РВ: конкатенация *pq*



Построение КА по РВ: итерация p^*



Теорема Клини: доказательство $\Rightarrow$

Теорема

Классы автоматных и регулярных языков эквивалентны

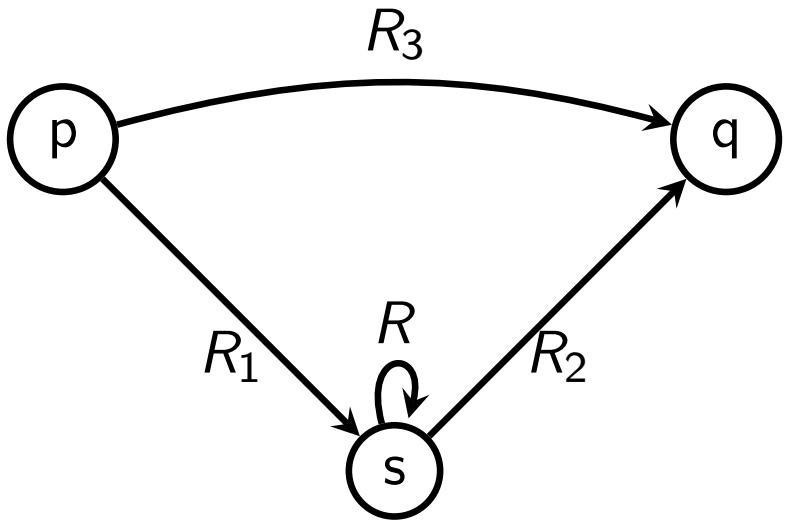
Доказательство.

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

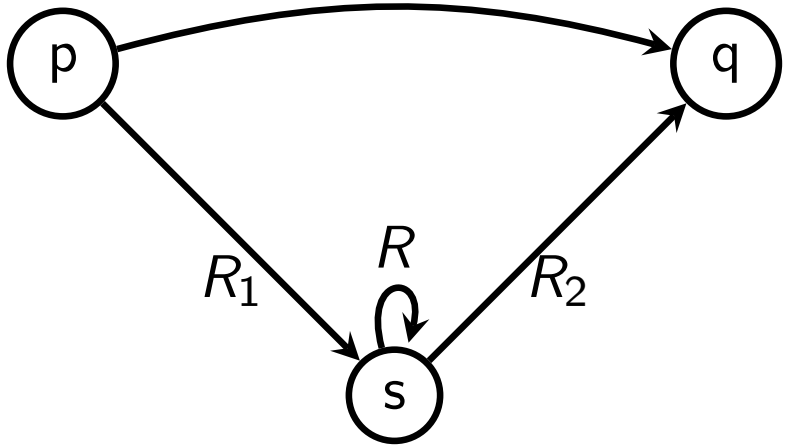
Идея: на ребрах пишем регулярные выражения, соответсвующие путям между вершинами, последовательно исключаем состояния

□

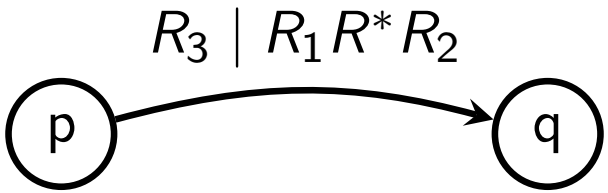
Исключение состояния s

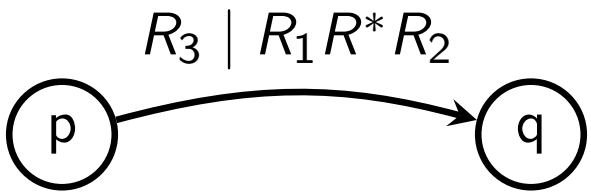


$$R_3 \mid R_1 R^* R_2$$

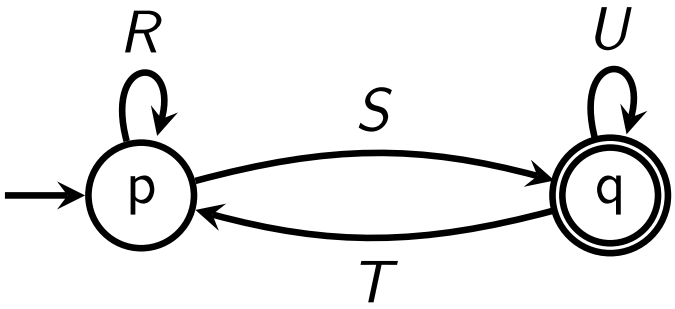


Исключение состояния s: удаление ребер и вершины



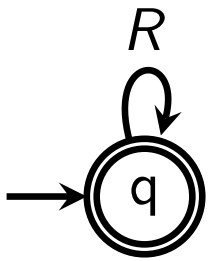


Исключение состояний: последний шаг



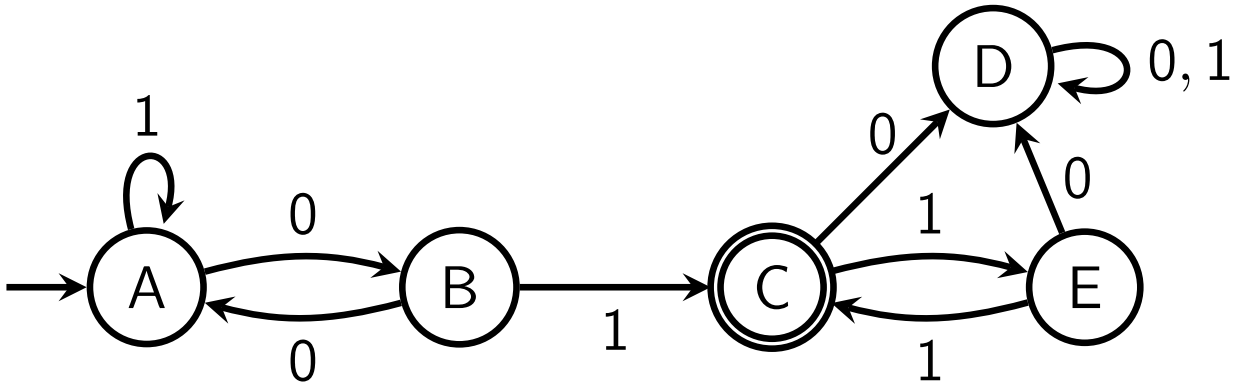
$$(R^* \mid SU^*T)^*SU^*$$

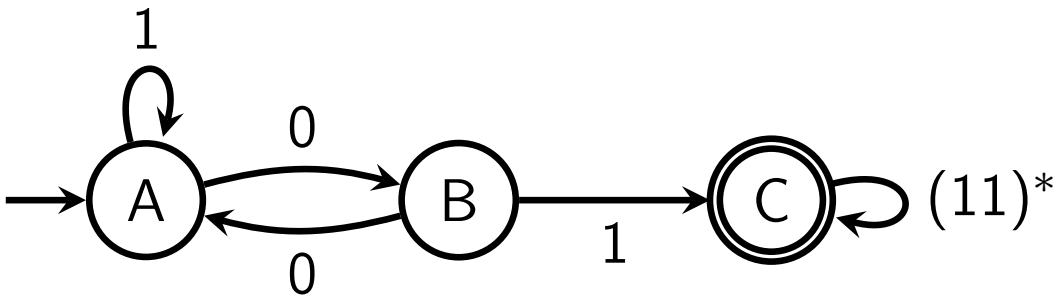
Исключение состояний: последний шаг

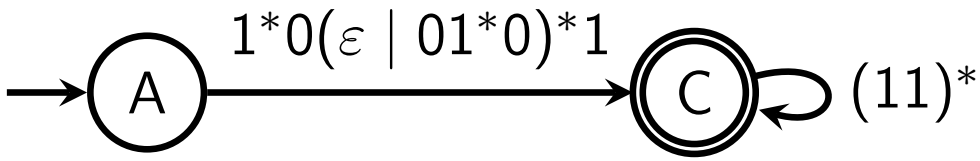


R^*

Исключение состояний: пример







$$1^*0(\varepsilon \mid 01^*0)^*1(11)^*$$

Свойства регулярных выражений

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

Регулярная грамматика

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

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

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

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

Регулярная грамматика

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

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

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

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

Теорема

Пусть L — формальный язык.
$\exists G_r$ *— праволинейная грамматика, т.ч.* $L = L(G_r) \Leftrightarrow$
$\exists G_l$ *— леволинейная грамматика, т.ч.* $L = L(G_l)$

Регулярная грамматика

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

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

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

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

Теорема

Пусть L — формальный язык.
$\exists G_r$ *— праволинейная грамматика, т.ч.* $L = L(G_r) \Leftrightarrow$
$\exists G_l$ *— леволинейная грамматика, т.ч.* $L = L(G_l)$

Регулярная грамматика — праволинейная или леволинейная грамматика

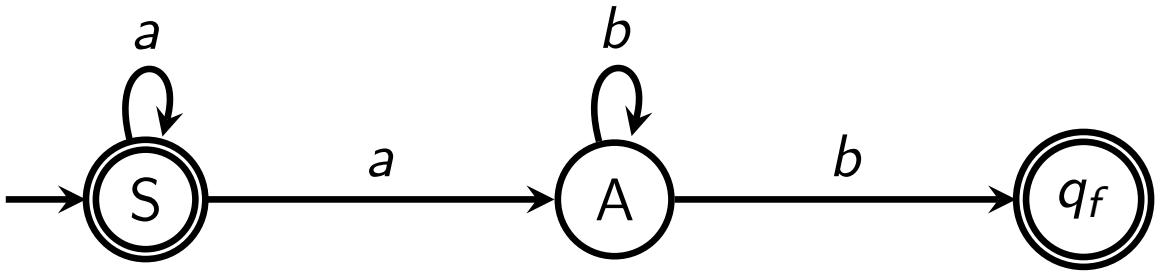
Эквивалентность регулярной грамматики и НКА

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

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

Пример построения НКА по регулярной грамматике

$$S \to aA \mid aS \mid \varepsilon$$
$$A \to bA \mid b$$

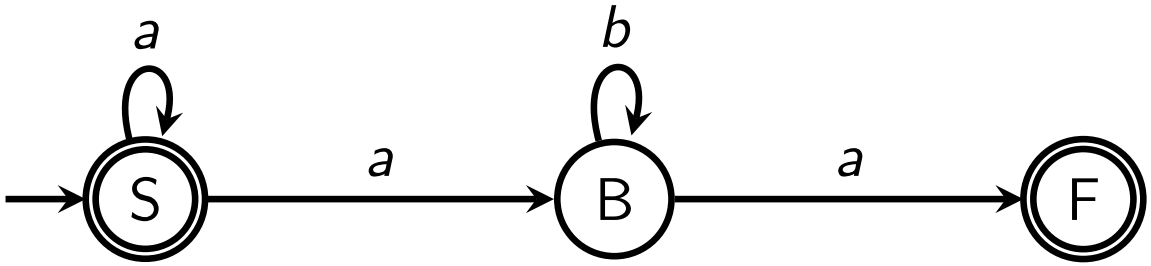


Эквивалентность регулярной грамматики и НКА

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

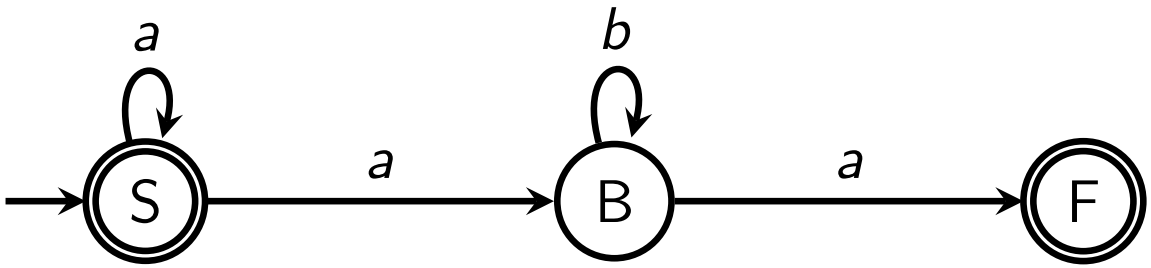
- $V_N = Q$
- $V_T = \Sigma$
- $\forall \delta(A, a) = B : (A \to aB) \in P$
- $\forall B \in F : (B \to \varepsilon) \in P$
- $S = q_0$
- Опционально: удалить ε-правила и бесполезные символы

Пример построения регулярной грамматики по НКА



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

Пример построения регулярной грамматики по НКА



$$S \to aB \mid aS \mid \varepsilon$$
$$B \to bB \mid a$$

Лемма о разрастании (о накачке)

Теорема

L — *регулярный язык над* $\Sigma \Rightarrow \exists n : \forall \omega \in L, |\omega| > n$
$\exists x, y, z \in \Sigma^* : xyz = \omega, y \neq \varepsilon, |xy| \leq n,$
$\forall k \geq 0 : xy^k z \in L$

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, значит выполняется лемма о накачке
- Берем n из леммы, рассматриваем слово $(^n)^n$
- Его можно разбить на $xyz, y \neq \varepsilon, |xy| \leq n$
- $|xy| \leq n \Rightarrow y = (^b, b > 0$
- Берем $k = 2 : xy^k z = (^{n+b})^n$, что не принадлежит L
- Получили противоречие $\Rightarrow L$ не регулярен

Использование леммы о накачке

$$L = \{a^{k^2} \mid k \geq 0\}$$

- Предполагаем, что L — регулярный язык, значит выполняется лемма о накачке
- Берем n из леммы, рассматриваем слово $a^{(n+1)^2}$
  - Слово a^{n^2} не подойдет, потому что $|a^{n^2}| = n$, где $n = 1$
- Его можно разбить на $xyz, y \neq \varepsilon, |xy| \leq n$
- Берем $k = 0 : xy^0z = xz$
- $n^2 < n^2 + n + 1 = (n+1)^2 - n \leq |xz| \leq (n+1)^2 - 1 < (n+1)^2$
- Длина слова xz находится между двумя соседними полными квадратами, поэтому это слово не в L
- Получили противоречие $\Rightarrow L$ не регулярен

Резюме

- 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