Теория автоматов

Лекция 4: Недетерминированные конечные автоматы

Дьулустан Никифоров

Кафедра ИТ
Северо-Восточный Федеральный Университет

Осень 2024

Регулярные языки: Конкатенация

Theorem

Множество регулярных языков замкнуто под операцией конкатенации, т.е.
A_1 *и* A_2 *— регулярные языки* $\Rightarrow$ $A_1 A_2$ *— регулярный язык.*

Proof:

Попробуем также построить M, который симулирует действия автоматов M_1 и M_2. Но теперь все сложнее! Почему?

Регулярные языки: Конкатенация

Theorem

Множество регулярных языков замкнуто под операцией конкатенации, т.е.
A_1 *и* A_2 *— регулярные языки* $\Rightarrow$ $A_1 A_2$ *— регулярный язык.*

Proof:

Попробуем также построить M, который симулирует действия автоматов M_1 и M_2. Но теперь все сложнее! Почему?

M должна сначала принять какую-то часть строки согласно машине M_1, а потом принять оставшуюся часть строки согласно M_2. Но где провести такую границу? Непонятно...

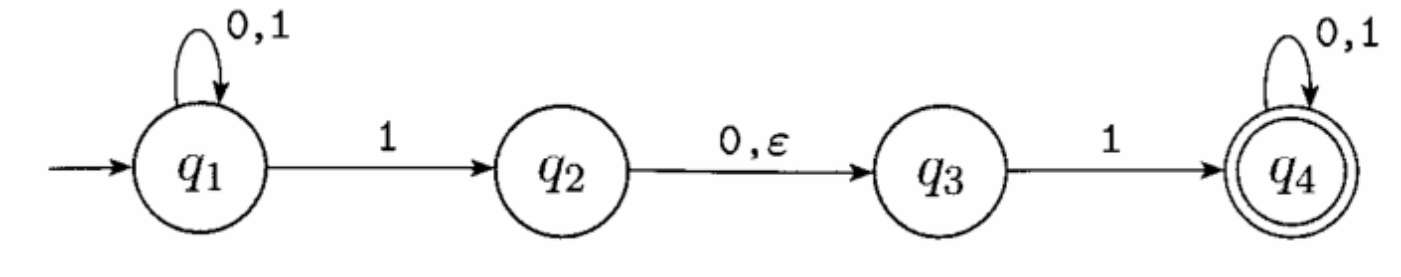
Недетерминированные Конечные Автоматы

Чтобы справиться с такой проблемой, нам надо ввести новое понятие — nondeterminism (недетерменированность).

- Конечные автоматы, которые были у нас до этого — называются **Deterministic Finite Automata (DFA)** / Детерминированные конечные автоматы (ДКА).
- Введем новое определение конечных автоматов — **Nondeterministic Finite Automata (NFA)** / Недетерминированные конечные автоматы (НКА).
- Теперь мы не требуем, чтобы для каждого состояния для каждого символа была ровно одна стрелка. Стрелок может быть 0, 1 или больше.
- Также у нас есть стрелки, обозначенные через ε — это стрелки, которые “бесплатно” переходят в другое состояние.

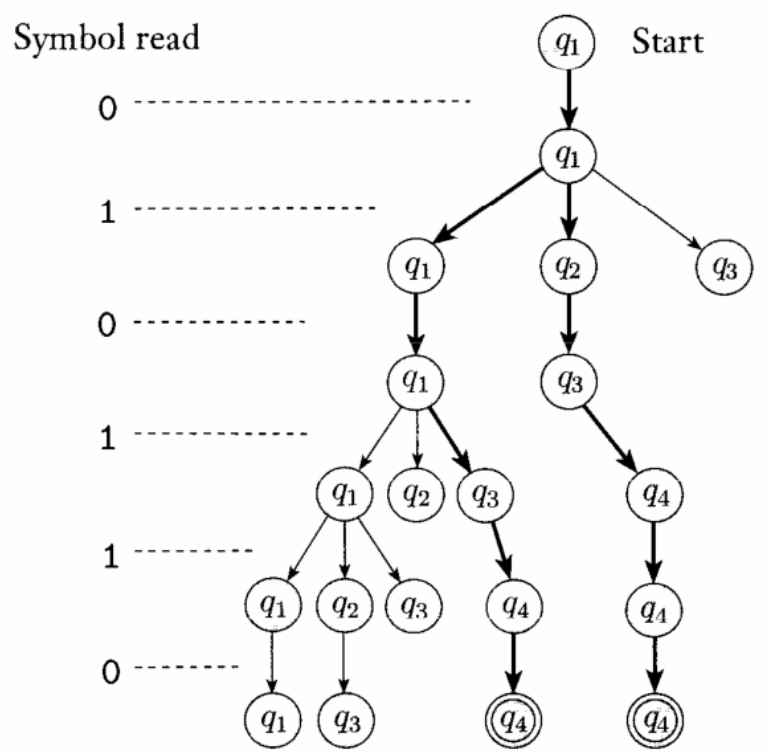
Недетерминированные Конечные Автоматы

- Можно сказать, автомат делает вычисления на параллельных процессах. Автомат принимает строку, если принимает ее хотя бы в одном из исполняемых процессов.



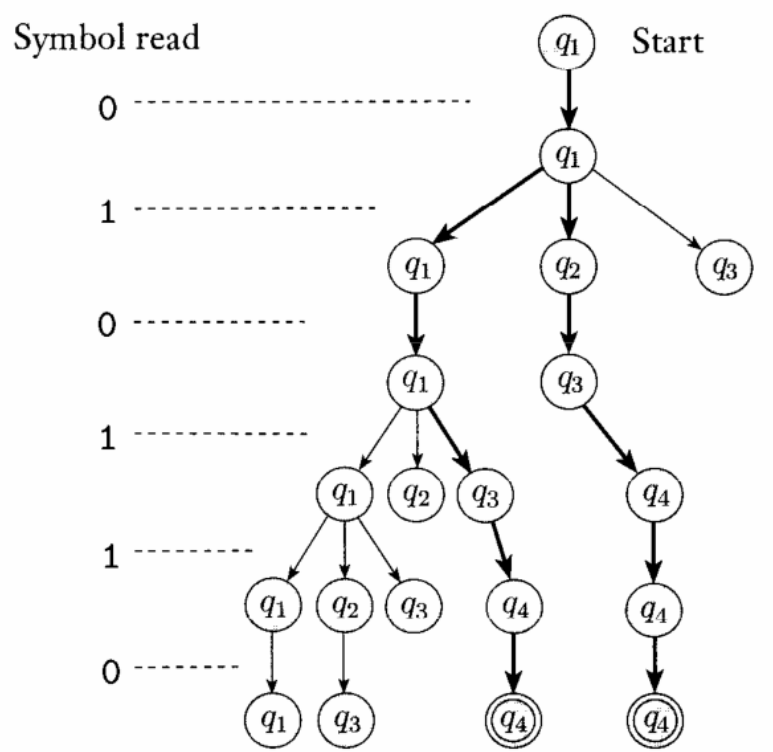
Давайте скормим этому автомату строку 010110.

Недетерминированные Конечные Автоматы



Какой язык распознается этой машиной?

Недетерминированные Конечные Автоматы



Какой язык распознается этой машиной? Строки, которые содержат 101 или 11 как подстроку.

ДКА и НКА

Детерминированные Конечные Автоматы (ДКА)
Недетерминированные Конечные Автоматы (НКА)

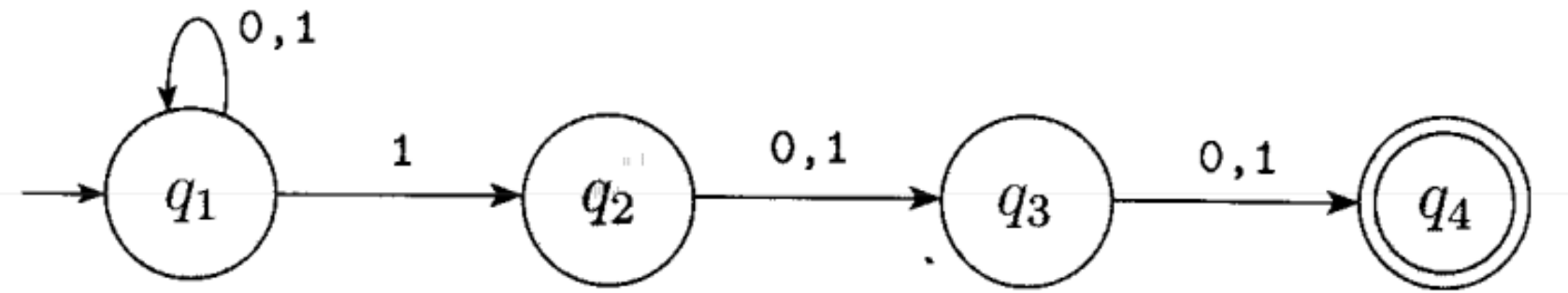
- Кажется, что мы усложнили понятие конечных автоматов с введением нондетерминизма.
- На самом деле, все наоборот! Строить и понимать НКА на деле гораздо проще, чем ДКА.
- НКА содержит *гораздо* более концентрированную информацию, чем ДКА (экспоненциально больше).

НКА: примеры

Пусть A — язык бинарных строк таких, что третий с конца символ — это единичка.
Можно легко построить НКА для этого языка:

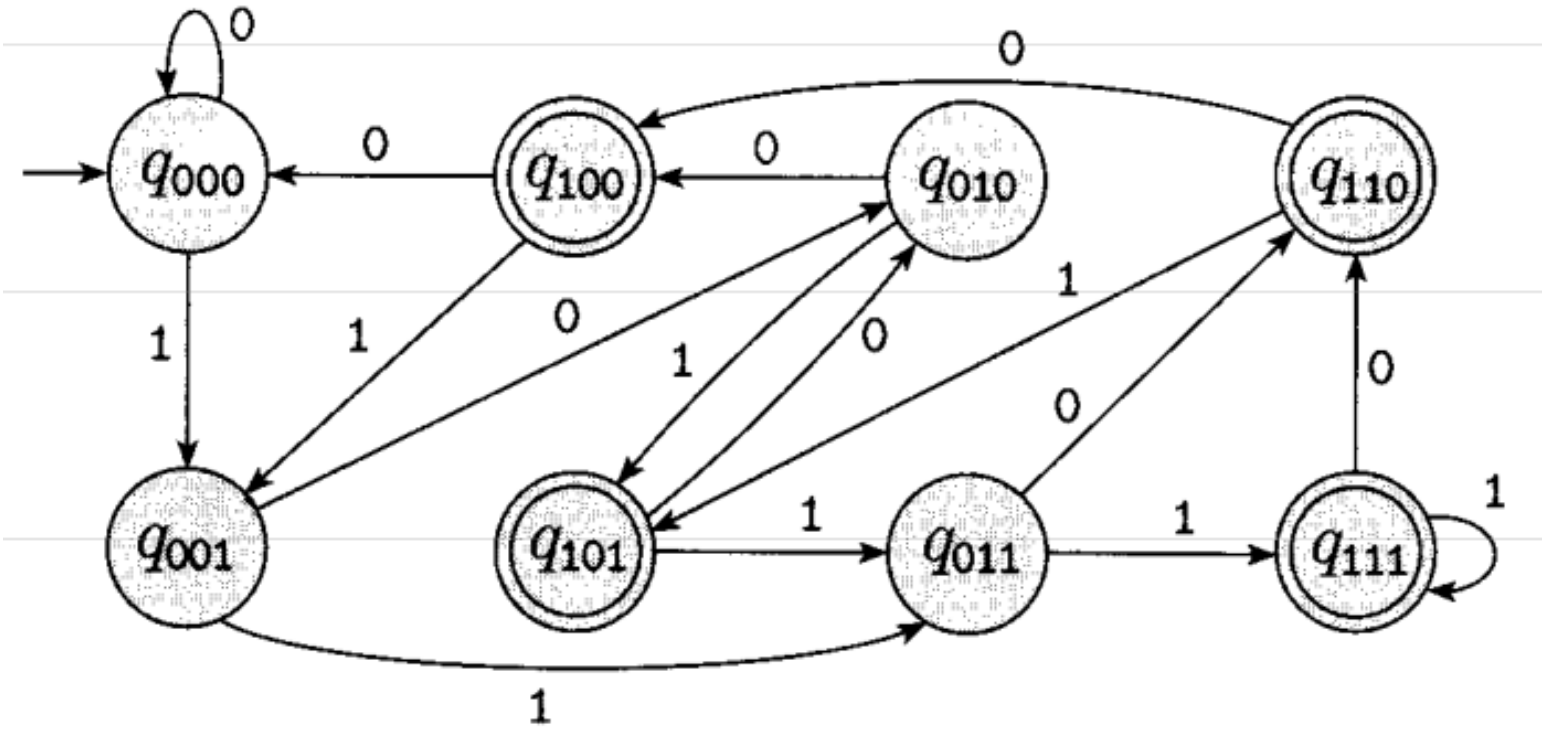
НКА: примеры

Пусть A — язык бинарных строк таких, что третий с конца символ — это единичка.
Можно легко построить НКА для этого языка:



НКА: примеры

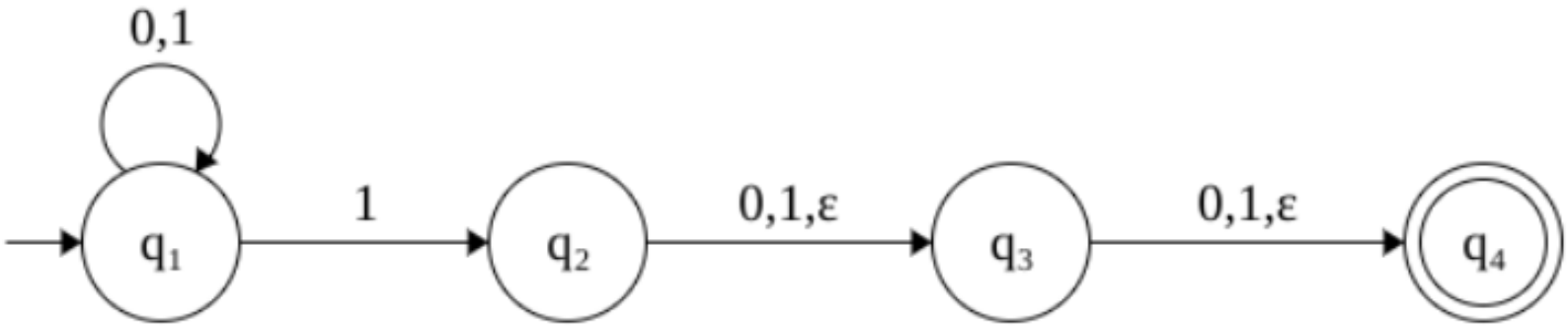
ДКА, который распознает тот же язык:



Уже не айс.

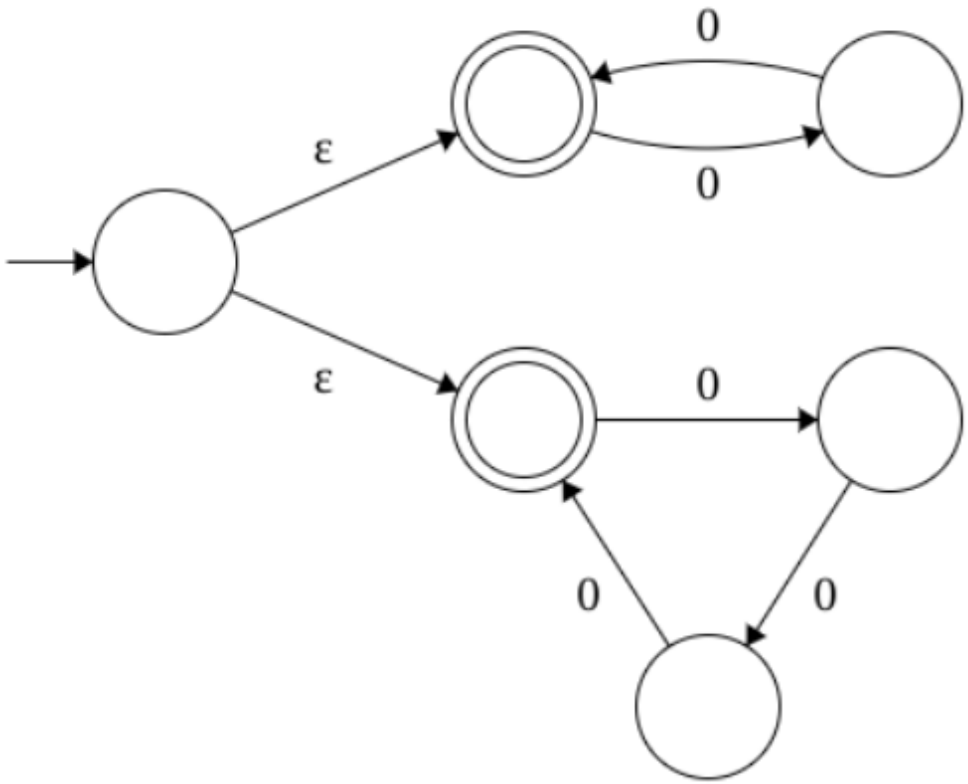
НКА: примеры

Слегка изменим предыдущий НКА:



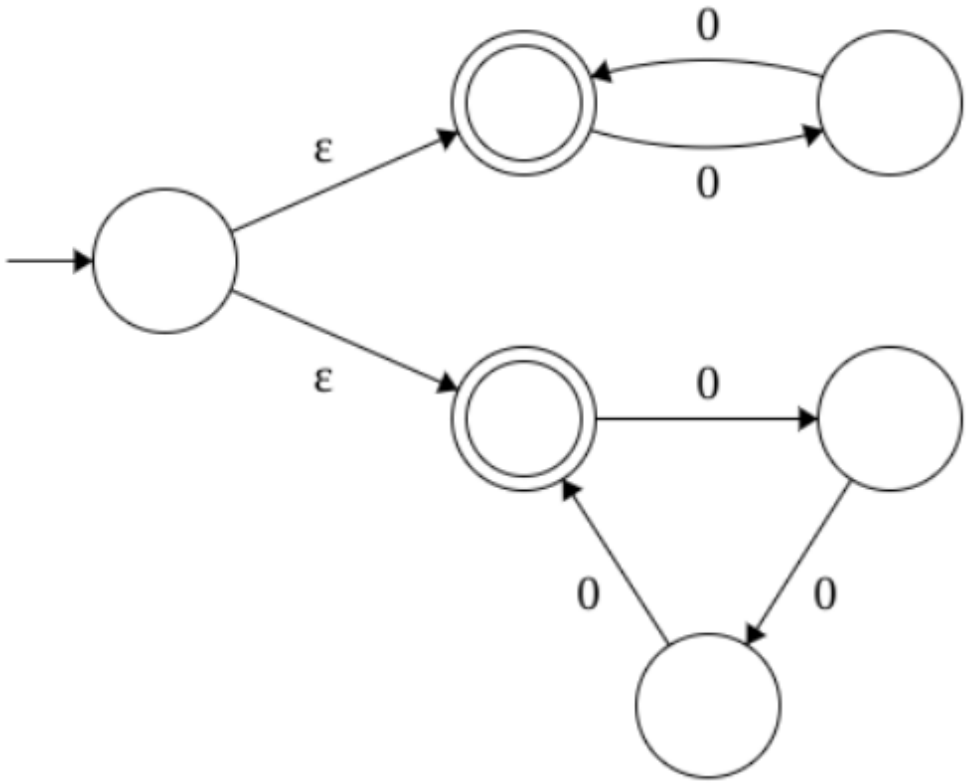
- Что это за язык?
- А как будет теперь выглядеть ДКА для этого языка?

НКА: примеры



Какой язык?

НКА: примеры



Какой язык?

Строки из нулей, количество которых делится на 2 или на 3.
А что так просто!

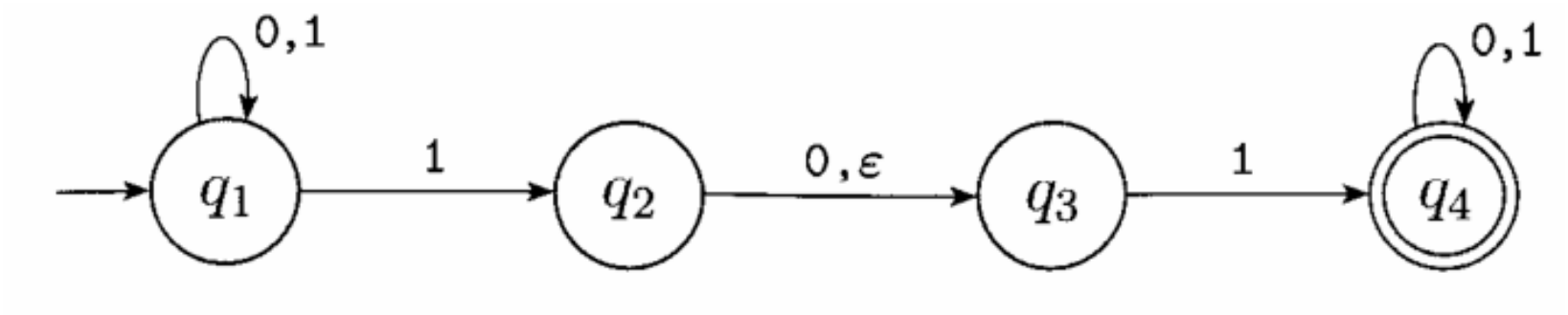
НКА: definition

Definition

Недетерминированный конечный автомат (nondeterministic finite automaton) — это 5-tuple $(Q, \Sigma, \delta, S, F)$, где

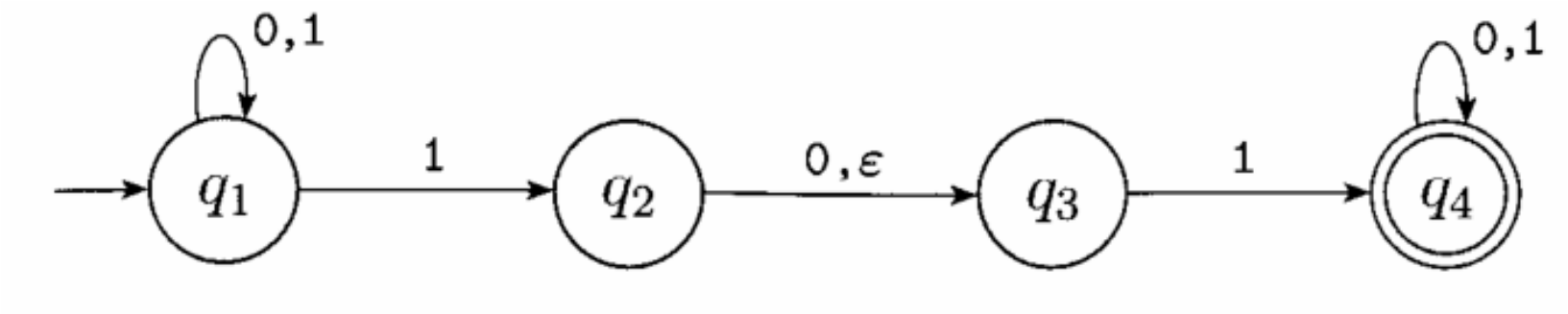
- Q — конечное множество, называемое множеством **состояний (states)**,
- Σ — конечное множество, называемое **алфавитом (alphabet)**,
- $\delta : Q \times (\Sigma \cup \varepsilon) \to 2^Q$ — **функция перехода (transition function)**,
- $S \in Q$ — **начальное состояние (start state)**,
- $F \subseteq Q$ **множество принимающих состояний (set of accept states)**.

НКА: формальное описание автомата



$(Q, \Sigma, \delta, S, F)$:

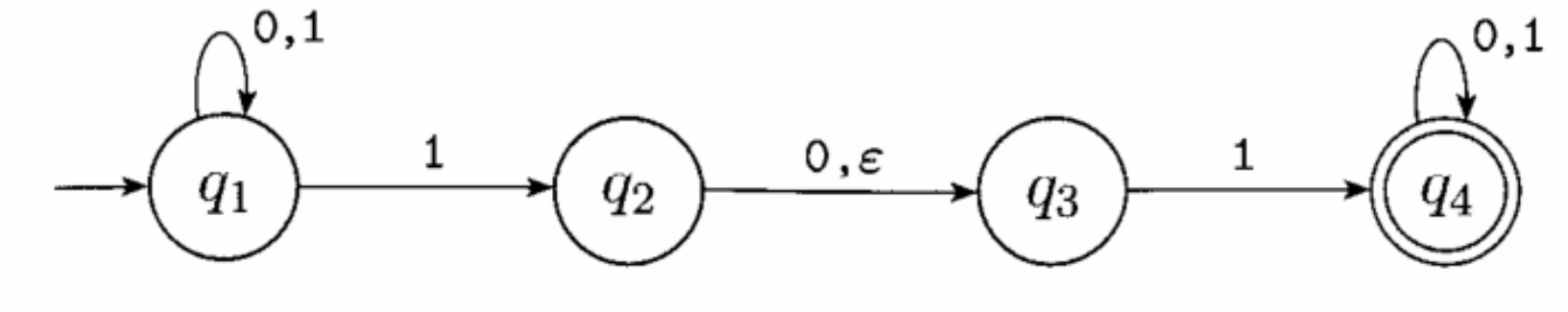
НКА: формальное описание автомата



$(Q, \Sigma, \delta, S, F)$:

- $Q = \{q_1, q_2, q_3, q_4\}$,
- $\Sigma = \{0, 1\}$,
- $S = q_1$,
- $F = \{q_4\}$,

НКА: формальное описание автомата



$\delta()$:

	0	0	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$

НКА: вычисление автомата

Definition

Пусть $M = (Q, \Sigma, \delta, S, F)$ — НКА, $w = w_1 w_2 \dots w_n$ — строка над Σ.

Тогда **вычислением (computation)** автомата M над строкой w называется последовательность

$s_0 \xrightarrow{b_1} s_1 \xrightarrow{b_2} s_2 \xrightarrow{b_3} \dots \xrightarrow{b_{k-1}} s_{k-1} \xrightarrow{b_k} s_k$, где:

- $b_1 b_2 \dots b_k = w_1 w_2 \dots w_n$ (некоторые $b_i = \varepsilon$, т.е. пустые символы);
- s_0 — начальное состояние автомата;
- $s_i \in \delta(s_{i-1}, b_i)$ для $i = 1, \dots, k$.

M **принимает (accepts)** w, если последнее состояние в *некотором* вычислении M над w является принимающим.

Конечные Автоматы: язык автомата

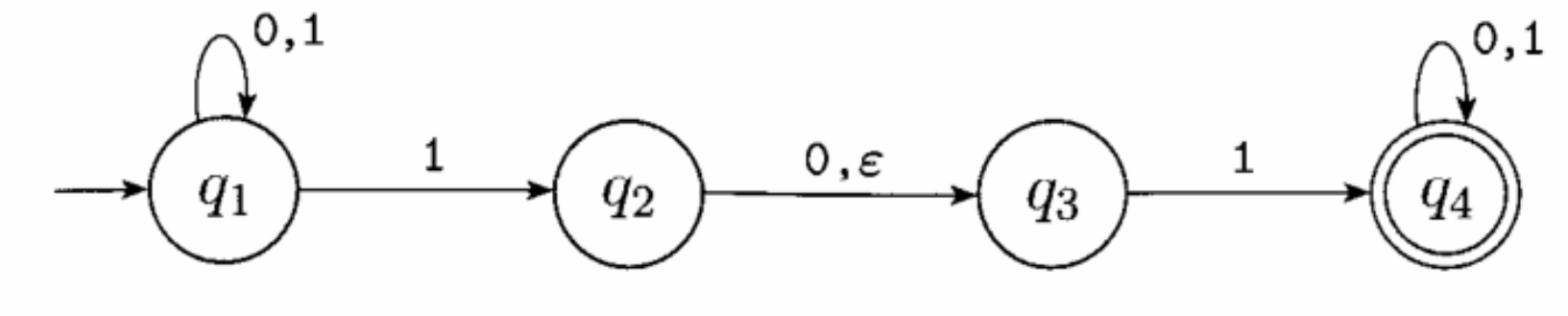
- Если A — это множество всех строк, *принимаемых* автоматом M, то мы говорим, что **M распознает/принимает (recognizes/accepts) A**.
- **A — язык (language)** автомата **M**, обозначается **A=L(M)**.
- Обратите внимание, всегда можно сказать, что M принимает пустую строку.

Конечные Автоматы: язык автомата

- **принимающее вычисление** — это вычисление, в котором автомат полностью обрабатывает строку и в конце оказывается в *принимающем* состоянии.
- **непринимающее вычисление** — это вычисление, в котором автомат полностью обрабатывает строку и в конце оказывается в *непринимающем* состоянии.
- **умирающее вычисление** — это вычисление, в котором автомат «умирает» в процессе обработки строки.

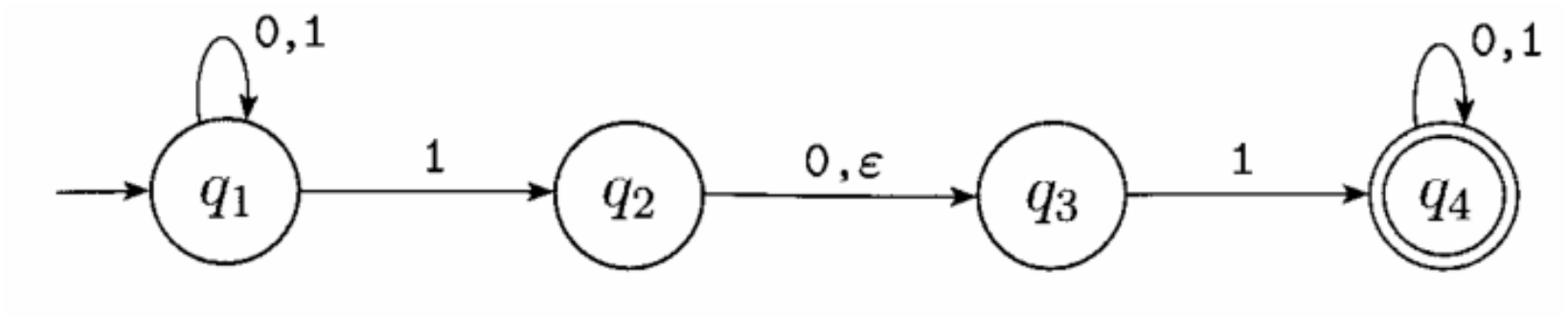
НКА принимает строку, если найдётся хотя бы одно принимающее вычисление для неё.

НКА: вычисление автомата



Построим **принимающее вычисление** этого автомата на строке 00110100:

НКА: вычисление автомата



Построим **принимающее вычисление** этого автомата на строке 00110100:

$$q_1 \xrightarrow{0} q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3 \xrightarrow{1} q_4 \xrightarrow{0} q_4 \xrightarrow{0} q_4$$

Автомат принимает строку, так как последнее состояние этого вычисления — это q_4, являющийся принимающим.

Другое принимающее вычисление:

$$q_1 \xrightarrow{0} q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{\varepsilon} q_3 \xrightarrow{1} q_4 \xrightarrow{0} q_4 \xrightarrow{1} q_4 \xrightarrow{0} q_4 \xrightarrow{0} q_4$$

НКА: вычисление автомата

непринимающее вычисление на 00110100:

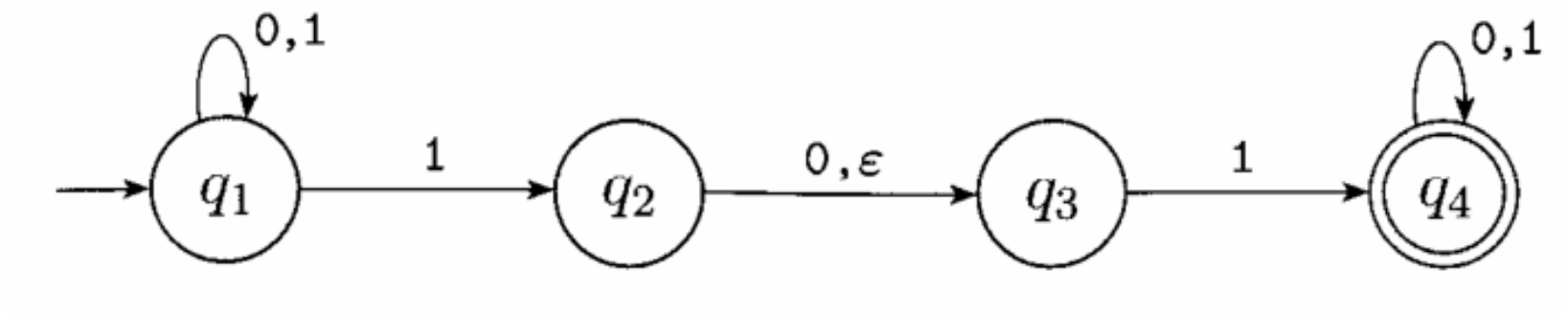
$$q_1 \xrightarrow{0} q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_1 \xrightarrow{\varepsilon} q_1 \xrightarrow{1} q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_1 \xrightarrow{0} q_1 \xrightarrow{0} q_1$$

умирающее вычисление на 00110100:

$$q_1 \xrightarrow{0} q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_1 \xrightarrow{1} \text{death}$$

Если построим **непринимающее/умирающее вычисление** на строке — это еще не означает, что автомат не принимает строку. Возможно, можно сделать другое вычисление, которое принимает — надо тщательно проверять!

НКА: вычисление автомата



Язык автомата:

$L(M) = \{w \in \{0,1\}^* \mid w$ содержит подстроку 11 или 101$\}$.

НКА vs ДКА

- Кажется, мы очень существенно расширили возможности конечных автоматов, когда мы добавили недетерминизм $\Rightarrow$ НКА мощнее, чем ДКА, верно?

НКА vs ДКА

- Кажется, мы очень существенно расширили возможности конечных автоматов, когда мы добавили недетерминизм $\Rightarrow$ НКА мощнее, чем ДКА, верно?
- Оказывается, нет!
- Понятно, что любой язык, распознаваемый НКА, также распознается ДКА (ДКА — это частный случай НКА).
- Главный вопрос: есть ли языки, распознаваемые НКА, но не распознаваемые ДКА?

НКА без ε = ДКА

Theorem

Если N — НКА без ε-переходов, то существует ДКА, который распознает язык $L(N)$.

Proof: Пусть $N = (Q, \Sigma, \delta, S, F), L(N) = A$.
Мы построим ДКА $M = (Q', \Sigma, \delta', S', F')$ такой, что $L(M) = A$.

НКА без ε = ДКА

Theorem

Если N — НКА без ε-переходов, то существует ДКА, который распознает язык $L(N)$.

Proof: Пусть $N = (Q, \Sigma, \delta, S, F), L(N) = A$.
Мы построим ДКА $M = (Q', \Sigma, \delta', S', F')$ такой, что $L(M) = A$.

- $Q' = 2^Q$ — каждое подмножество состояний N будет отдельным состоянием в M,
- для $R \in Q'$ и $a \in \Sigma$, пусть
  $\delta'(R, a) = \{q \in Q \mid q \in \delta(r, a)$ для некоторого $r \in R\}$,
  По-другому говоря, $\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$.
- $S' = \{S\}$,
  $F' = \{R \in Q' \mid R \cap F \neq \emptyset\}$.

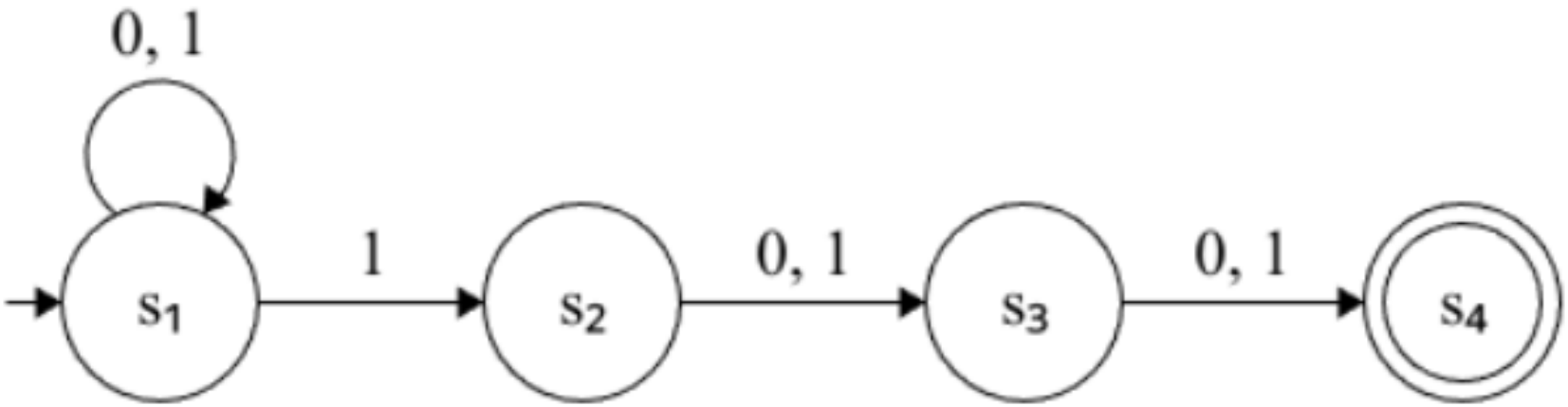
НКА без ε = ДКА

Практичный алгоритм, как перевести НКА N без ε-переходов в ДКА M:

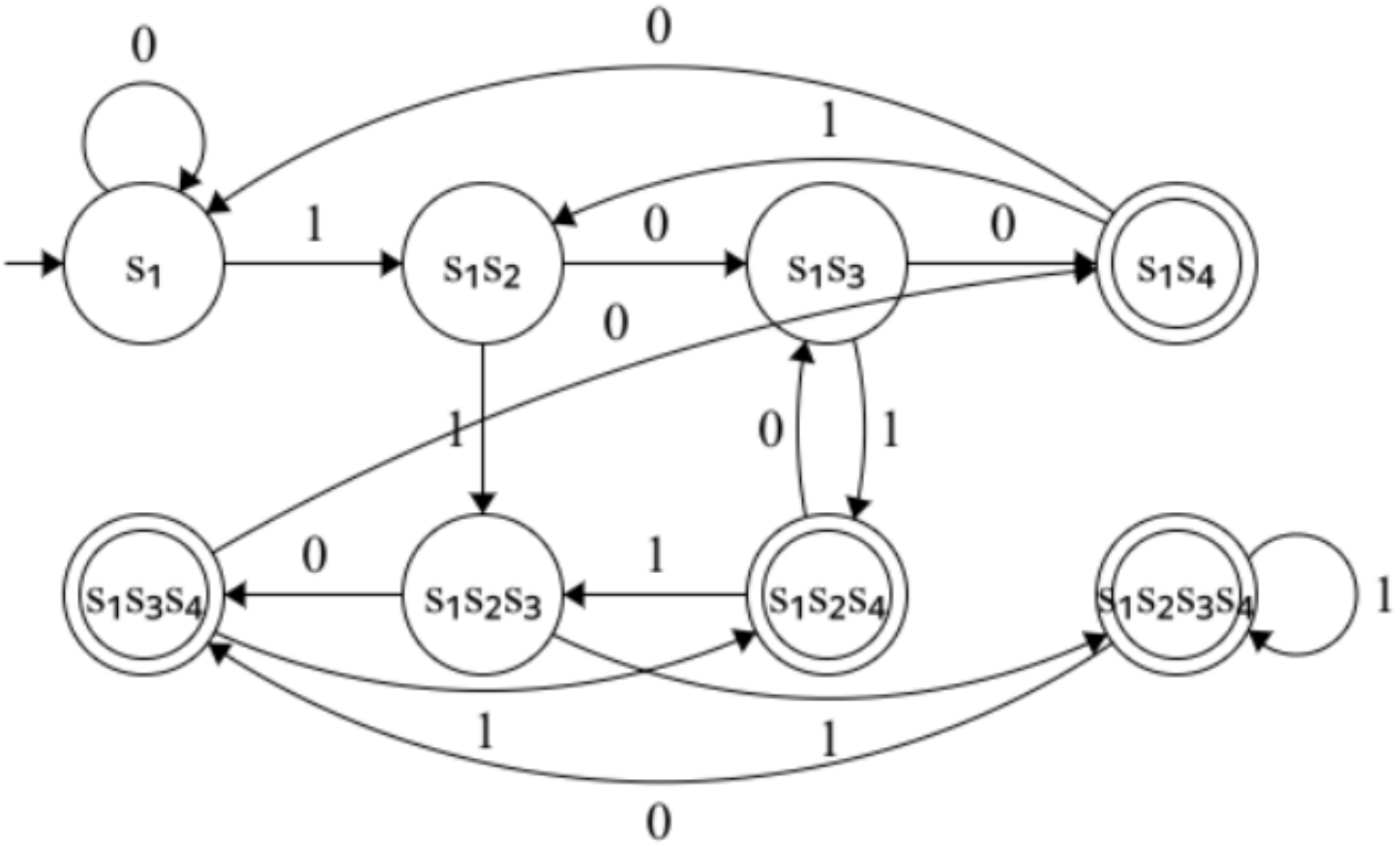
1. Создать начальное состояние такое же, как и у НКА.
2. Добавить нехватающий переход в какое-нибудь из существующих состояний: посмотреть в какие состояния $(*)$ идет переход (в N) $\rightarrow$ направить переход в то состояние M, которое представляет собой множество тех состояний $(*)$ из N (если такое состояние еще не существует, создать ее); может быть, что такого перехода из этого состояния вообще не существует в N — в этом случае надо делать переход в состояние, представляющее пустое множество состояний $(\emptyset)$.
3. Продолжать делать шаг 2, пока все созданные состояния не будут иметь все переходы.
4. Сделать принимающими те состояния (из M), которые содержат внутри хотя бы одно принимающее состояние из N.

НКА без ε = ДКА: Пример 1

Применим этот алгоритм для такого автомата:



НКА без ε = ДКА: Пример 1



НКА без ε = ДКА: Пример 2

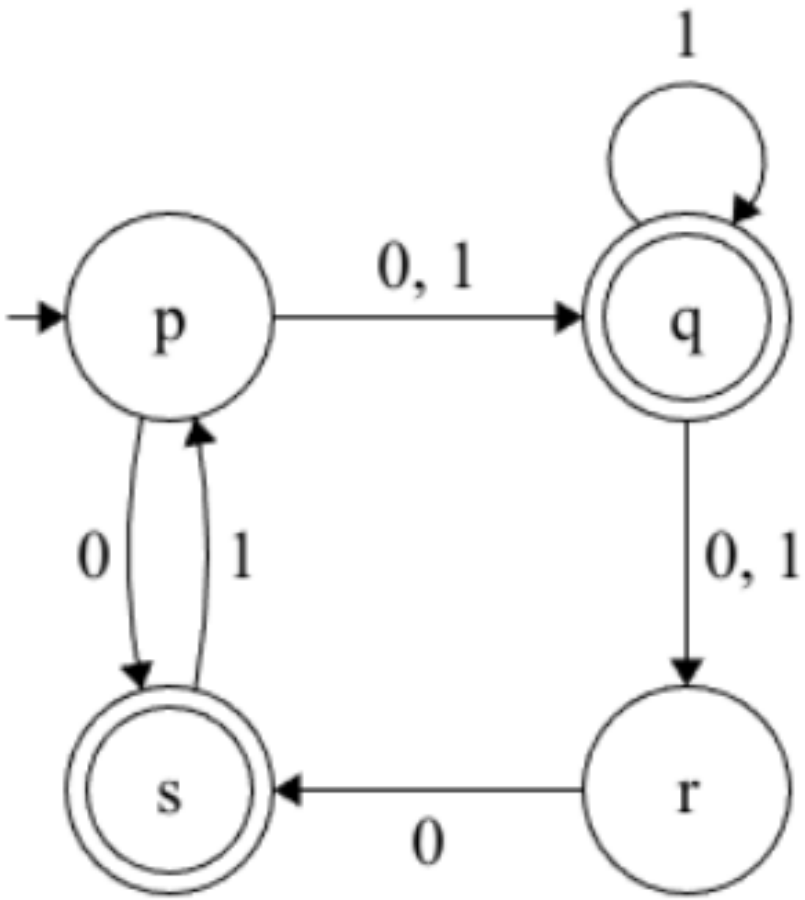
Задача: Переведите в эквивалентный ДКА следующие НКА (через $*$ обозначены принимающие состояния, через $\rightarrow$ начальное состояние)

	0	1
$\rightarrow p$	$\{q, s\}$	$\{q\}$
$*q$	$\{r\}$	$\{q, r\}$
r	$\{s\}$	$\{p\}$
$*s$	$\emptyset$	$\{p\}$

НКА без ε = ДКА: Пример 2

Решение:

Сам НКА:



НКА без ε = ДКА: Пример 2

Эквивалентный ему ДКА: