Теория автоматов и формальных языков

Конечные автоматы

Лектор: Екатерина Вербицкая

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»

12 сентября 2019

В предыдущей серии

- Формальные языки повсюду. Язык — множество строк над алфавитом
- Существует множество способов описать язык
- Задачи теории формальных языков
  - Как представить язык?
  - Какие есть характеристики у разных представлений языка?
  - Как определить, принадлежит ли строка данному языку?

В предыдущей серии

- Формальная грамматика
  - ⟨Терминалы, Нетерминалы, Правила, Стартовый нетерминал⟩
- 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 детей узла
- Контекстно-свободная грамматика
  - все правила имеют вид $A \to \alpha$

В предыдущей серии: левосторонний и правосторонний вывод

$$\begin{array}{rcl} E & \to & E + E \mid N \\ N & \to & 0 \mid 1 \end{array}$$

Какой из нетерминалов раскрывать на данном шаге?

- Левосторонний вывод: раскрываем самый левый нетерминал
  - $\boldsymbol{E} \Rightarrow \boldsymbol{E} + E \Rightarrow \boldsymbol{N} + E \Rightarrow 1 + \boldsymbol{E} \Rightarrow 1 + \boldsymbol{E} + E \overset{2}{\Rightarrow} 1 + 0 + \boldsymbol{E} \overset{2}{\Rightarrow} 1 + 0 + 1$

В предыдущей серии: левосторонний и правосторонний вывод

$$\begin{aligned} E &\rightarrow E + E \mid N \\ N &\rightarrow 0 \mid 1 \end{aligned}$$

Какой из нетерминалов раскрывать на данном шаге?

- Левосторонний вывод: раскрываем самый левый нетерминал
  - $\boldsymbol{E} \Rightarrow \boldsymbol{E} + E \Rightarrow \boldsymbol{N} + E \Rightarrow 1 + \boldsymbol{E} \Rightarrow 1 + \boldsymbol{E} + E \overset{2}{\Rightarrow} 1 + 0 + \boldsymbol{E} \overset{2}{\Rightarrow} 1 + 0 + 1$
- Правосторонний вывод: раскрываем самый правый нетерминал
  - $\boldsymbol{E} \Rightarrow E + \boldsymbol{E} \Rightarrow E + \boldsymbol{N} \Rightarrow \boldsymbol{E} + 1 \Rightarrow E + \boldsymbol{E} + 1 \overset{2}{\Rightarrow} \boldsymbol{E} + 0 + 1 \overset{2}{\Rightarrow} 1 + 0 + 1$

В предыдущей серии: левосторонний и правосторонний вывод

$$\begin{array}{rcl} E & \to & E + E \mid N \\ N & \to & 0 \mid 1 \end{array}$$

Какой из нетерминалов раскрывать на данном шаге?

- Левосторонний вывод: раскрываем самый левый нетерминал
  - $\boldsymbol{E} \Rightarrow \boldsymbol{E} + E \Rightarrow \boldsymbol{N} + E \Rightarrow 1 + \boldsymbol{E} \Rightarrow 1 + \boldsymbol{E} + E \overset{2}{\Rightarrow} 1 + 0 + \boldsymbol{E} \overset{2}{\Rightarrow} 1 + 0 + 1$
- Правосторонний вывод: раскрываем самый правый нетерминал
  - $\boldsymbol{E} \Rightarrow E + \boldsymbol{E} \Rightarrow E + \boldsymbol{N} \Rightarrow \boldsymbol{E} + 1 \Rightarrow E + \boldsymbol{E} + 1 \overset{2}{\Rightarrow} \boldsymbol{E} + 0 + 1 \overset{2}{\Rightarrow} 1 + 0 + 1$
- Для каких грамматик левосторонний и правосторонний вывод любой строки совпадают?

Разбор самостоятельной

Построить 2 различных (левосторонних) вывода строки $1 + 0 + 1$

$$\begin{array}{rcl} E & \to & E + E \mid N \\ N & \to & 0 \mid 1 \end{array}$$

- $\boldsymbol{E} \Rightarrow \boldsymbol{E} + E \Rightarrow \boldsymbol{N} + E \Rightarrow 1 + \boldsymbol{E} \Rightarrow 1 + \boldsymbol{E} + E \overset{2}{\Rightarrow} 1 + 0 + \boldsymbol{E} \overset{2}{\Rightarrow} 1 + 0 + 1$

Разбор самостоятельной

Построить 2 различных (левосторонних) вывода строки $1 + 0 + 1$

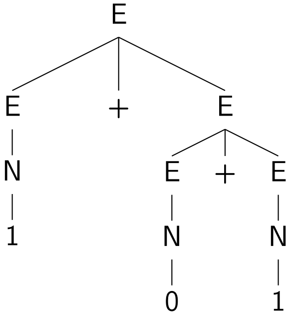
$$
\begin{array}{lcl}
E & \to & E + E \mid N \\
N & \to & 0 \mid 1
\end{array}
$$

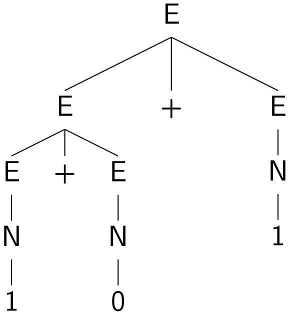
- $\boldsymbol{E} \Rightarrow \boldsymbol{E} + E \Rightarrow \boldsymbol{N} + E \Rightarrow 1 + \boldsymbol{E} \Rightarrow 1 + \boldsymbol{E} + E \overset{2}{\Rightarrow} 1 + 0 + \boldsymbol{E} \overset{2}{\Rightarrow} 1 + 0 + 1$
- $\boldsymbol{E} \Rightarrow \boldsymbol{E} + E \Rightarrow \boldsymbol{E} + E + E \Rightarrow \boldsymbol{N} + E + E \Rightarrow 1 + \boldsymbol{E} + E \overset{2}{\Rightarrow} 1 + 0 + \boldsymbol{E} \overset{2}{\Rightarrow} 1 + 0 + 1$

Построить 2 различных (левосторонних) вывода строки $1 + 0 + 1$

$$
\begin{aligned}
E &\rightarrow E + E \mid N \\
N &\rightarrow 0 \mid 1
\end{aligned}
$$

- $\boldsymbol{E} \Rightarrow \boldsymbol{E} + E \Rightarrow \boldsymbol{N} + E \Rightarrow 1 + \boldsymbol{E} \Rightarrow 1 + \boldsymbol{E} + E \overset{2}{\Rightarrow} 1 + 0 + \boldsymbol{E} \overset{2}{\Rightarrow} 1 + 0 + 1$
- $\boldsymbol{E} \Rightarrow \boldsymbol{E} + E \Rightarrow \boldsymbol{E} + E + E \Rightarrow \boldsymbol{N} + E + E \Rightarrow 1 + \boldsymbol{E} + E \overset{2}{\Rightarrow} 1 + 0 + \boldsymbol{E} \overset{2}{\Rightarrow} 1 + 0 + 1$





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

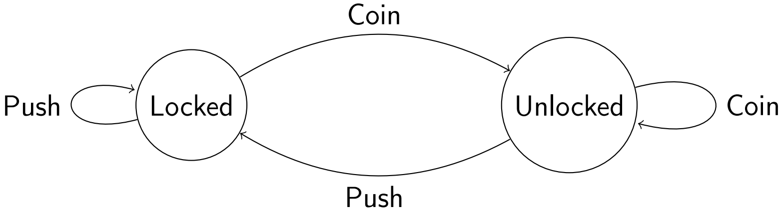
Теоретико-множественное доказательство невозможности описания языков

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

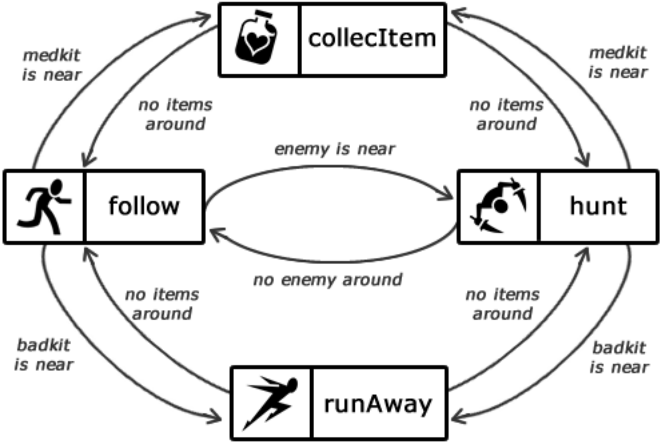
Нет.

- 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

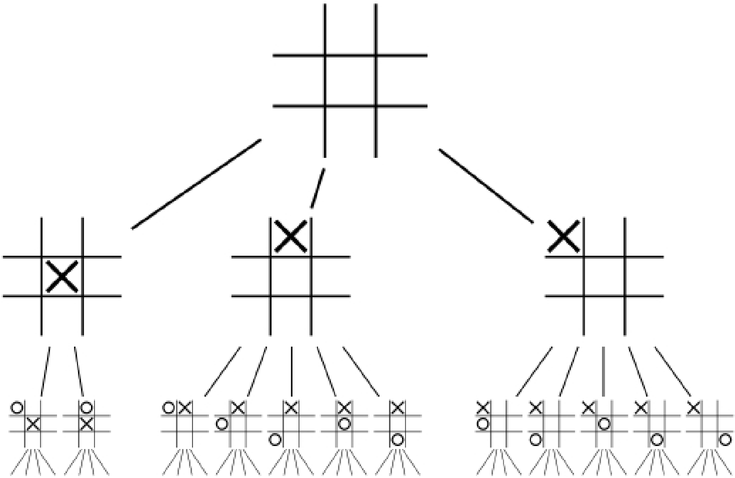
Конечные автоматы



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

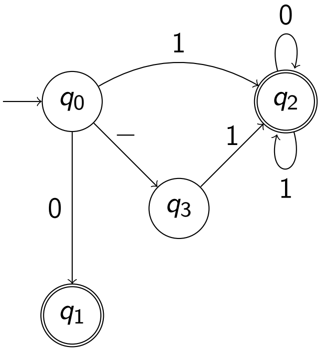
КА называется **полным**, если существует переход из каждого состояния по каждому символу алфавита

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

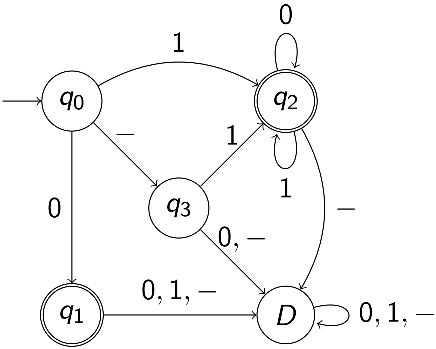
Пример конечного автомата

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

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



Пример полного конечного автомата



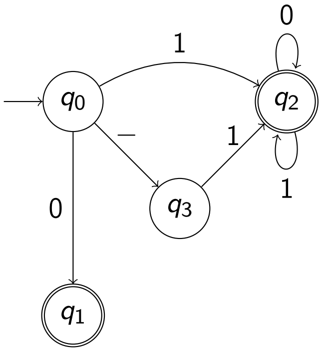
Путь в конечном автомате

- **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 $-\mathbf{110}$ 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

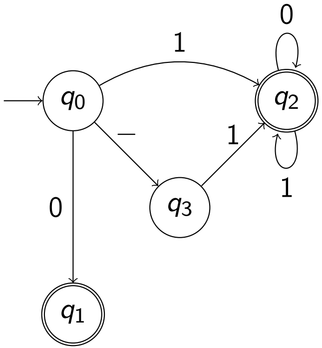
Цепочка ω **распознается** КА, если $\exists$ успешный путь с меткой ω

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

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 $\delta'(q_i, \omega) = t_1, \delta'(q_j, \omega) = t_2 \Rightarrow (t_1 \notin F \Leftrightarrow t_2 \in F)$
- q_i и q_j **эквивалентны** $(q_i \sim q_j)$, если $\forall \omega \in \Sigma^* : \delta'(q_i, \omega) = t_1, \delta'(q_j, \omega) = t_2 \Rightarrow (t_1 \in F \Leftrightarrow t_2 \in F)$
  - 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)
- $\sphericalangle \alpha = a_1 a_2 \dots a_k, a_i \in \Sigma : \langle s, \alpha \rangle \vdash^* \langle u, \varepsilon \rangle ; \langle s', \alpha \rangle \vdash^* \langle u', \varepsilon \rangle$
- $\sphericalangle \langle s, a_1 \rangle \vdash^* \langle l, \varepsilon \rangle ; \langle s', a_1 \rangle \vdash^* \langle l', \varepsilon \rangle$. s, s' эквивалентны $\Rightarrow l, l'$ эквивалентны
- Аналогично для всех $a_i \Mapsto u, u'$ эквивалентны
- $\Mapsto \forall q$ — 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

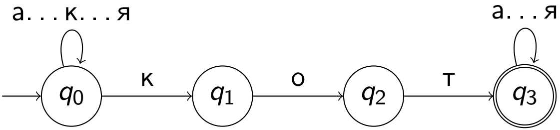
Недетерминированный КА

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

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

Недетерминированный КА: пример

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

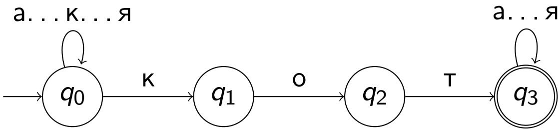


Распознавание слова НКА

- **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

$$
\begin{array}{rcl}
R(\alpha) & = & \{p \mid \langle s, \alpha \rangle \vdash^* \langle p, \varepsilon \rangle\}
\end{array}
$$

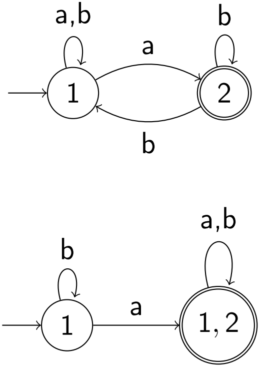
$$
\begin{array}{rcl}
R(\varepsilon) & = & \{q_0\} \\
R(\alpha c) & = & \{q \mid q \in \delta(p, c), p \in R(\alpha)\}
\end{array}
$$

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

Построение ДКА по НКА: алгоритм Томпсона

- 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 Томпсона 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 $(q_{d_0}, w_1 w_2 \dots w_m) \vdash (u_{d_1}, w_2 \dots w_m) \vdash^* (u_{d_m}, \varepsilon)$

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