Теория автоматов и формальных языков

Регулярные языки

Лектор: Екатерина Вербицкая

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»

13 сентября 2016г.

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
  - $\langle$Терминалы, Нетерминалы, Правила, Стартовый нетерминал$\rangle$
  - Однозначность (любая строка имеет единственный вывод) и неоднозначность
- 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 $A \to \alpha$

Конечный автомат

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

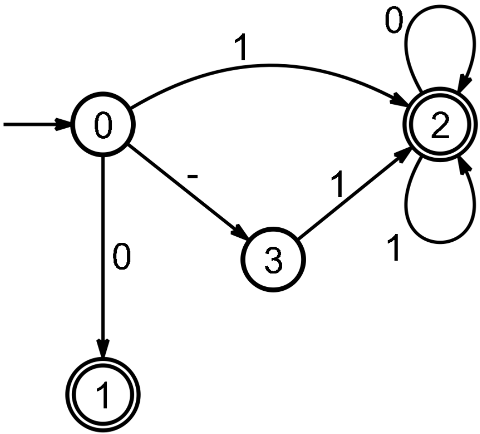
- $Q \neq \emptyset$ — конечное множество состояний
- Σ — Конечный входной алфавит
- δ — отображение типа $Q \times \Sigma \to Q$
- $q_0 \in Q$ — начальное состояние
- $F \subseteq Q$ — множество конечных состояний

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

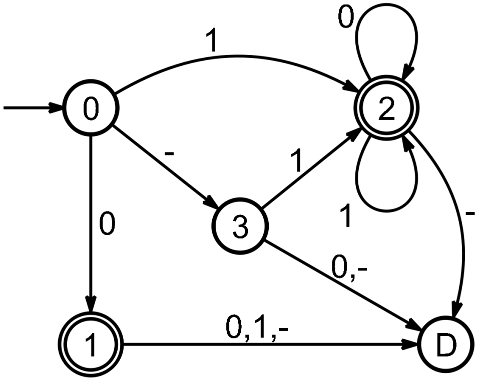
- Обычно добавляют "дьявольскую"вершину, она же сток.

Пример конечного автомата

$Q = \{0, 1, 2, 3\}, \Sigma = \{0, 1, -\}, q_0 = 0, F = \{1, 2\}$
$\delta(0, 0) = 1; \delta(0, 1) = 2; \delta(0, -) = 3; \delta(3, 1) = 2; \delta(2, 0) = 2; \delta(2, 1) = 2$



Пример полного конечного автомата

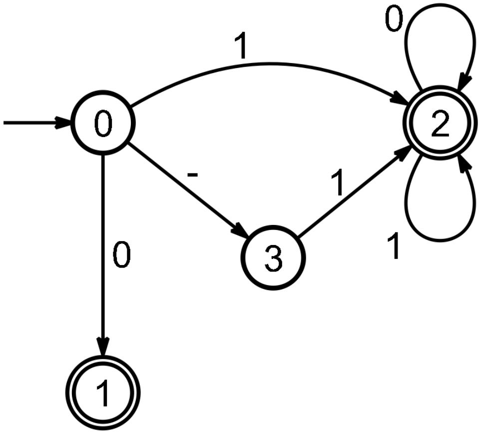


Путь в конечном автомате

- **Путь** — кортеж $\langle q_0, e_1, q_1, \ldots, e_n, q_n \rangle$
  - $n \geq 0$
  - $\forall i. e_i = \langle q_{i-1}, w_i, q_i \rangle \in \delta$
  - q_0 — **начало** пути
  - q_n — **конец** пути
  - $w_0, w_1, \ldots, w_n$ — **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

$\langle 0, \langle 0,' -', 3\rangle, 3, \langle 3,' 1', 2\rangle, 2, \langle 2,' 1', 2\rangle, 2, \langle 2,' 0', 2\rangle, 2\rangle$ — успешный путь с меткой “-110” длины 4



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 $\langle p, x, q \rangle \in \Delta$ 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

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:

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

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 $Q \neq \emptyset$:
  - 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)
- $\sphericalangle \alpha = a_1 a_2 \ldots a_k, a_i \in \Sigma : \langle s, \alpha \rangle \vdash^* \langle u, \varepsilon \rangle ; \langle s', \alpha \rangle \vdash^* \langle u', \varepsilon \rangle$
- $\sphericalangle \langle s, a_1 \rangle \vdash^* \langle l, \varepsilon \rangle ; \langle s', a_1 \rangle \vdash^* \langle l', \varepsilon \rangle$. s, s' эквивалентны $\Rightarrow l, l'$ эквивалентны
- Аналогично для всех $a_i \Rrightarrow u, u'$ эквивалентны
- $\Rrightarrow \forall q$ — 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 построение A_{min} в нем не может быть эквивалентных состояний. Противоречие

Недетерминированный КА

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

- $Q \neq \emptyset$ — конечное множество состояний
- Σ — Конечный входной алфавит
- δ — отображение типа $Q \times \Sigma \to 2^Q$
- $q_0 \in Q$ — начальное состояние
- $F \subseteq Q$ — множество конечных состояний

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

Алгоритм, определяющий допустимость слова

- $R(\alpha) = \{p | \langle s, \alpha \rangle \vdash^* \langle p, \varepsilon \rangle\}$
- $R(\varepsilon) = \{q_0\}$
- $R(\alpha c) = \{q | q \in \delta(p, c), p \in R(\alpha)\}$
- 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 $q' = \{t = \delta(s, c) | s \in q, c \in \Sigma\}$. 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: $\langle \Sigma, Q, q_0 \in Q, T \subset Q, \delta : Q \times \Sigma \to 2^Q \rangle$
  -

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. $\square$

Регулярная грамматика

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

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

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

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

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

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

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

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

Теорема

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

Регулярная грамматика

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

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

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

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

Теорема

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

Регулярная грамматика — праволинейная или леволинейная грамматика