Теория формальных языков

Тух Игорь и Ютман Михаил
по лекциям Михаила Эдуардовича Дворкина

16 июня 2018 г.

Содержание

1. Введение

См. Хопкрафт, Мотвани, Ульман. Введение в теорию автоматов, языков и вычислений

Эти типы стали есть на складе

***Определение* 1.1.** Зафиксируем конечное множество символов (алфавит) Σ и скажем, что слово $w_1 \dots w_n$ – это конечная последовательность символов из алфавита. Язык L – это множество слов (над алфавитом Σ).

Замечание. Язык L не обязан быть конечным.

Например, рассмотрим язык всех синтаксически корректных фраз на русском языке, тогда предложения будет словами, а кириллица, знаки препинания и пробелы – алфавитом. Заметим, что прапраправнук (сколько угодно раз пра) – корректное слово на русском. Значит, таких фраз бесконечное количество.

***Определение* 1.2.** Σ^* – это множество всех слов над Σ. ε – стандартное обозначение пустого слова.

Замечание. Множество всех слов над алфавитом – это счетное множество. Любой язык, как его подмножество, либо конечный, либо счетный (т.е. не более, чем счетный). Над пустым алфавитом есть два различных языка $\{\varepsilon\}$ и $\varnothing$.

Замечание. Пусть есть некоторая задача с ответом ДА или НЕТ (входные данные – слово над алфавитом). Тогда есть соответствие между некоторым языком и множеством слов, на которых ответ ДА. Задач с ответом ДА/НЕТ несчетное число. А различных программ на C++ счетное. Значит, существуют неразрешимые ДА/НЕТ задачи.

2. Детерминированные Конечные Автоматы (DFA)

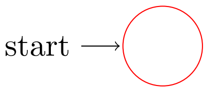
2.1. Детерминированные Конечные Автоматы

***Определение* 2.1.** Детерминированный конечный автомат $A = (\Sigma, Q, q_o, T, \delta)$, где

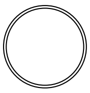
1. Σ – конечный алфавит;
2. Q – конечное множество состояний;
3. $q_o \in Q$ – начальное состояние;
4. $T \subseteq Q$ – множество терминальных состояний;
5. $\delta : Q \times \Sigma \to Q$ – функция переходов.

Обозначение.

Далее в конспекте так будет обозначаться стартовые состояния:

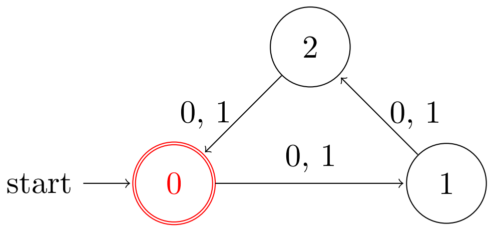


А вот так терминальные:



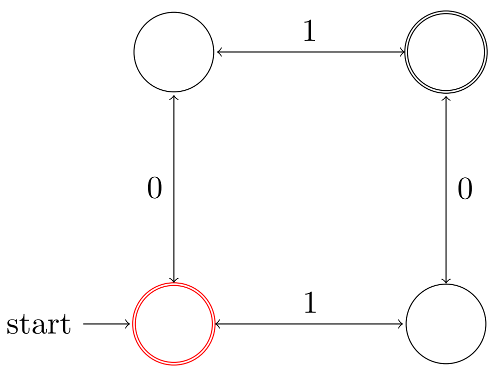
Примеры.

1. Зафиксируем алфавит $\{0, 1\}$. $L = \{w \mid |w| \vdots 3\}$



Переход по любому символу происходит по указанной стрелке. Значение в вершине совпадает с остатком по модулю 3 слова. Терминальное состояние совпадает с начальным.

2. $\Sigma = \{0, 1\}$



Как несложно понять, распознаются только слова четной длины.

***Определение* 2.2.** Обобщенная функция переходов $\hat{\delta} : Q \times \Sigma^* \to Q$, причем $\hat{\delta}(q, \varepsilon) = q$ и $\hat{\delta}(q, xw) = \hat{\delta}(\hat{\delta}(q, x), w)$. И тогда язык принимаемый ДКА $L(A) = \{w \in \Sigma^* : \hat{\delta}(q_o, w) \in T\}$.

Замечание. Проверка принадлежности слова w языку $L(A)$ может быть осуществлена за $\mathcal{O}(|w|)$.

2.2. Минимизация ДКА

Задача минимизации: $A_1 \to A_2$ так, что $L(A_1) = L(A_2)$ и $|Q(A_2)| \to \min$.

***Определение* 2.3.** Состояния p и q – различимые, если $\exists w \in \Sigma^* \;\; \hat{\delta}(p, w) \in T \oplus \hat{\delta}(q, w) \in T$.

Алгоритм (поиска всех пар различимых состояний)**.**

1. $p \in T, q \notin T \Rightarrow (p, q)$ – различима;
2. $\hat{\delta}(p, x) = u$ и $\hat{\delta}(q, x) = v$. Тогда (u, v) различима $\Rightarrow (p, q)$ – различима.

Нужно запустить DFS или BFS из вершин из первого пункта. Доказательство корректности очевидно по индукции. Время работы будет $\mathcal{O}(|Q|^2)$ (ребер вообще говоря, $|Q|^2 \cdot |\Sigma|$).

Лемма. "p неразличима с q" является отношением эквивалентности.

Доказательство. Рефлексивность, симметричность и транзитивность очевидны. □

Алгоритм (Минимизации ДКА)**.**

Рассмотрим класс эквивалентности. Будем строить автомат A_2. Новые состояния соответствуют классам эквивалентности. Начальным состоянием будет класс начального состояния. Каждый класс либо состоит только из терминальных, либо только из нетерминальных, на основании чего определяем терминальность в новом автомате. Несложно заметить, что не существует двух переходов по одной и той же букве из двух состояний из одного класса эквивалентности в два разных класса (тогда эти две вершины были бы различимы). Оставляем только состояния, достижимые из начального. Получили автомат, принимающий те же слова. Автомат меньшего размера получить нельзя, так как все классы эквивалентности нужны.

Замечание. Можно легко обобщить понятие различимости на вершины двух разных автоматов (пара состоит из одной вершины из одного автомата и еще одной из другой). Тогда аналогично найдем все различимые пары вершины в этих двух автоматах. Проверка на эквивалентность этих двух автоматов заключается в проверке различимости стартовых состояний.

2.3. Правые контексты

***Определение* 2.4.** Правым контекстом называется функция $C_L^R : \Sigma^* \to 2^{\Sigma^*}$, $C_L^R(w) = \{u \in \Sigma^* : wu \in L\}$

Аналогично можно определить левые контексты.

Замечание. $w \in L \Leftrightarrow \varepsilon \in C_L^R(w)$

Замечание. $\forall w \in \Sigma^* \;\; C_L^R(w)$ – различные множества правых контекстов. Количество $= |Q_{minA}|$.

***Определение* 2.5.** Тупиковым (дьявольским) состоянием ДКА называется состояние, все переходы из которого ведут в него же самого.

Замечание. Тупиковые состояния на картинке обычно не рисуют.

3. Недетерминированные конечные автоматы с ε-переходами

3.1. Недетерминированные конечные автоматы с ε-переходами

***Определение* 3.1.** Недетерминированный конечный автомат (НКА) A $= (\Sigma, Q, q_0, T, \delta)$, где

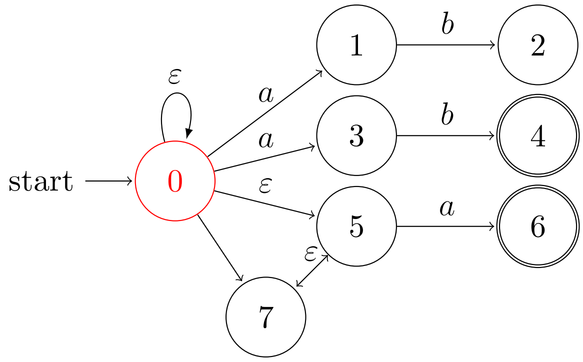
1. Σ – конечный алфавит
2. Q – конечное множество
3. $q_0 \in Q$ – начальное состояние
4. $T \subseteq Q$ – множество терминальных состояний
5. $\delta : Q \times (\{\varepsilon\} \cup \Sigma) \to 2^Q$ – функция переходов (по состоянию и символу получаем множество состояний, в которое мы можем перейти)

НКА принимает строку w и выдаёт "Да", если есть хотя бы один путь по строке w из начальной вершины в терминальную.

***Определение* 3.2.** ε-переход – это переход, который можно спонтанно сделать в любой момент чтения слова.

Число переходов, возможно, экспоненциально больше, чем у детерминированного, но тем не менее конечно.

Пример. Рассмотрим следующий автомат:

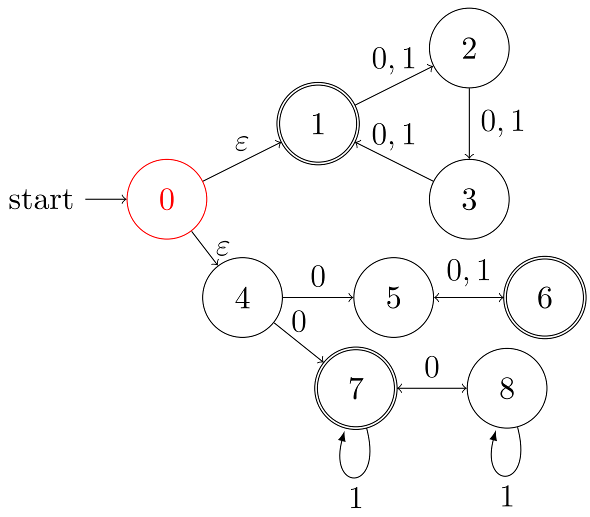


В этот автомат слово "a" можно принять перейдя из 0 в 3, а можно перейти по ε-переходу из 0 в 5, потом походить миллион раз туда-обратно по ε-переходу между 5 и 7, а потом из 5 перейти в 6, и таким образом, это слово распознаётся автоматом.

***Определение* 3.3.** Язык, распознаваемый НКА – это все такие слова, по которым существует хотя бы один путь из стартовой вершины в терминальную.

$L(A) = \{w \in \Sigma^* : \exists$ путь из q_0 в $q \in T$, на ребрах которого написано слово $w\}$

Пример.



Этот автомат разпознаёт следующий язык: $L(A) = \{$слова длины, кратной трём $\cup$ слова чётной длины, начинающиеся с нуля $\cup$ слова с нечётным числом нулей, начинающиеся с нуля$\}$.

То есть мы сначала выбираем, каким образом мы будем принимать наше слово, а потом пытаемся его принять, то есть "хвост" у нас детерминированный.

Надо подумать, как запрограммировать правильный выбор ветки автомата, которая принимает наше слово.

Первое, что приходит в голову – это каждый раз перебирать все возможные варианты, куда пойти, тогда на каждом ходу у нас $\leqslant |Q|$ рёбер, значит нам придётся рассмотреть $O(|Q|^{|w|})$ вариантов (это очень много, не говоря уже о том, что возможны циклы по ε).

Можем вместо того, чтобы перебирать все пути смотреть на множество достижимых по данной строке вершин, и при добавлении новой буквы обновлять это множество, а потом обновлять ещё раз с учётом всех ε-переходами.

***Определение* 3.4.** ε-замыкание состояния q – это множество состояний, достижимых из q только по ε-переходам.

Это множество можно предподсчитать с помощью dfs для каждой вершины. И теперь, когда мы собираемся считать очередной символ, мы добавим вместо вершины сразу её ε-замыкание (это решает проблему с ε-переходами).

Чтобы ещё сильнее оптимизировать, можно использовать вместо булева массива bitset, а также для каждого состояния и каждого символа предподсчитать побитовое "ИЛИ" *eps*-замыканий всех вершин, в которые из данной вершины есть переход по данному символу.

Таким образом, на предподсчёт у нас уйдёт $O(|Q|^2)$ времени, а именно $|Q|^2$ на построение замыканий и ещё $|\Sigma||Q|^2$ на описанный выше предподсчёт, а про размер алфавита мы сказали, что он константный.

И с помощью этого предподсчёта мы добились того, что у нас переход по очередной букве работает за $O(|Q|^2)$, но такая оценка почти никогда не достигается, поэтому на деле алгоритм работает примерно за $|w|$ (но это не точно).

Заметим, что если программа хранит всегда конечное число информации и читает слово слева направо, то она, по сути, является детерминированным конечным автоматом.

Если же мы посмотрим на все возможные замыкания состояний в ε-НКА, то среди них окажется всего $2^{|Q|}$ различных, то есть константа. То есть программа, которая осуществляет переходы по ε-НКА является детерминированным конечным автоматом.

3.2. Детерминизация ε-НКА

Мы уже догадались, что ε-НКА эквивалентен некоторому ДКА. Давайте формально объясним, как сделать из конкретного ε-НКА A_1 эквивалентный ему ДКА A_2, то есть такой, что $L(A_2) = L(A_1)$.

Алгоритм. Детерминизации ε-НКА

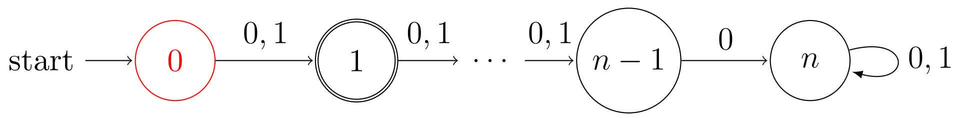
Построим все необходимые части ДКА:

1. $\Sigma_2 = \Sigma_1$, то есть алфавит берём такой же
2. $Q_2 = 2^{Q_1}$, то есть смотрим на множество всех достижимых по данной строке состояний
3. $q_{02} = \varepsilon\text{-closure}(q_{01})$, то есть вместо стартового состояния смотрим на его ε-замыкание.
4. $T_2 = \{S \subset Q : S \cap T \neq \varnothing\}$, то есть все подмножества множества состояний, содержащие хотя бы одно терминальное состояние
5. $\delta_2(S, a) = \bigcup\limits_{p \in S, p \to q} \varepsilon\text{-closure}(q)$, то есть объединение ε-замыканий по всем вершинам, в которые мы можем перейти по этому символу из какого-то состояния текущего множества достижимых

Замечание. Таким образом, множество рапознаваемых ε-НКА и ДКА языков совпадает, потому что по ε-НКА мы научились получать ДКА, а ДКА сам является НКА. Здесь всё вполне логично, так как, по факту, определения отличаются только тем, что у ε-НКА функция перехода возвращает не одно состояние, а множество состояний, и ещё наличие ε-переходов.

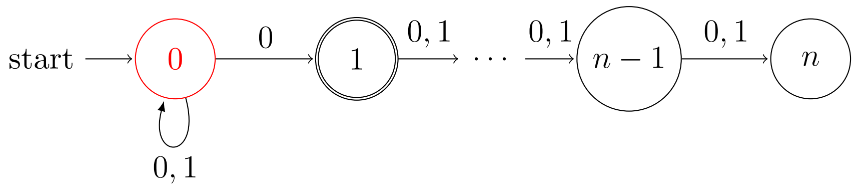
Оптимизации: состояние входит всегда со своим замыканием, поэтому можно делать только dfs из начального состояния, для которого мы хотим узнать замыкание.

Пример. $L_1 = \{w : w_n = 0\}$



И ДКА и НКА для этого языка имеют размер n.

Пример. $L = \{w : w_{|w|-n+1} = 0\} = L_1^R$. ε-НКА:



В ДКА для того же языка $|Q| \geq 2^n$ (можно доказать с помощью числа правых контекстов, но мы не будем). Почувствуйте разницу с $n+1$ состоянием в НКА. Это плохой пример, на котором алгоритм детерминизации экспоненциально взрывной.

Мы умеем делать в НКА каждый переход в худшем случае за $|Q|^2$. Тем временем в ДКА переход – это просто посмотреть число в прямоугольном массиве. Но мы знаем, тем не менее, что если мы захотим построить ДКА, то у нас может получиться слишком много состояний, и мы просто в память не влезет эта табличка.

Идея: попробуем детерминизировать НКА, и если за мало операций получилось детерминизировать, то будем работать с ДКА, а если не повезло, то будем работать с НКА.

3.3. Произведение конечных автоматов

***Определение* 3.5.** Пусть есть два автомата: A и B, тогда произведением автоматов называется автомат $A \times B$, у которого:

1. $\Sigma = \Sigma$, то есть алфавит такой же
2. $Q = Q_A \times Q_B$, множество пар состояний
3. $q_0 = (q_{0A}, q_{0B})$, пара из двух начальных
4. $\delta((p, q), a) = (\delta_A(p, a), \delta_B(q, a))$, то есть переходим по символу в обоих автоматах

В зависимости от того, как мы выберем терминальные состояния, мы можем получить автоматы, распознающие следующие языков:

1. $L_1 \cap L_2$, для этого возьмём $T = T_A \times T_B$
2. $L_1 \cup L_2$, для этого возьмём $T = T_A \times Q_B \cup Q_A \times T_B$

Замечание. Когда мы в ДКА инвертировали множество терминальных состояний, мы получали, автомат, который распознаёт дополнение исходного языка. В ε-НКА такой фокус не пройдёт. Если мы так сделаем, мы получим непонятно что.

4. Регулярные выражения

4.1. Академические регулярные выражения

***Определение* 4.1** (Академические регулярные выражения)**.**

Регулярное выражение R	$L(R)$
$\varnothing$	$\varnothing$
ε	$\{\varepsilon\}$
$a \quad (\forall a \in \Sigma)$	$\{a\}$
$R_1 \vert R_2$	$L(R_1) \cup L(R_2)$
R_1R_2	$\{w \in \Sigma^* \mid w = xy, x \in L(R_1), y \in L(R_2)\}$
R^*	$\{w \in \Sigma^* \mid w = x_1x_2 \dots x_n, n \in \mathbb{Z}_{0+}, \forall x_i \in L(R)\}$
(R)	$L(R)$

Последние четыре операции перечислены в порядке возрастания приоритета. Последняя операция называется "замыкание Клини". Получили рекурсивное определение, т.е. выражение, полученное в результате применения нескольких правил выше и будет называться академическим регулярным выражением.

Примеры.

1. $(0|1)^*$ – язык, состоящий из всех слов над алфавитом $\{0, 1\}$
2. $(0|1)^*1001^*$ – язык из слов, последняя группа нулей которых имеет длину два
3. $(00|1)^*$ – язык, состоящий их слов, все группы нулей которых имеют четную длину

Замечание.

1. $RR* = R_+$
2. $RR(\varepsilon|R)(\varepsilon|R) = R\{2-4\}$
3. Пример из промышленных (расширенных) регулярных выражений: $(00+)\backslash 1+$ – слова составной длины из нулей

Теорема 4.1 (Клини)**.** Множество языков, задаваемых академическими регулярными выражениями совпадает с множество языков, задаваемыми конечными автоматами.

Доказательство.

1. Докажем, что регулярные языки $\subseteq$ автоматные.
   $\forall R \to \varepsilon-$НКА $A: \ L(A) = L(R)$.
   Далее, аналог индукции:
   Итак, будем строить автомат, в котором будут выполнены следующие условия:

   (a) ровно одно терминальное состояние

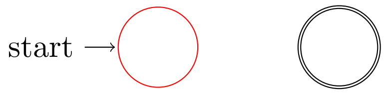
   (b) в начальное состоянии нет переходов

(c) из терминального состояния нет переходов

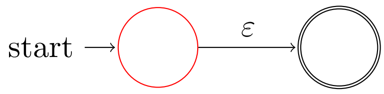
(d) начальное и терминальное состояние не совпадают

Итак, построение:

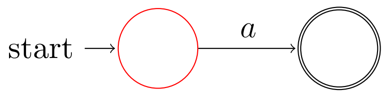
1) Автомат, задающий пустой язык:



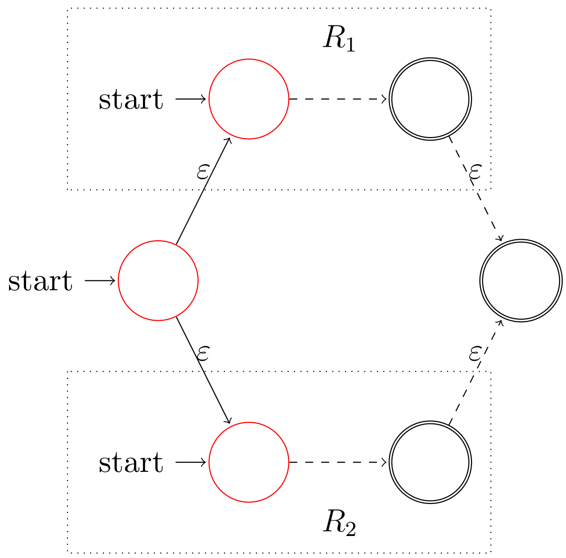
2) Автомат, задающий пустое слово:



3) Автомат, задающий одну букву a:

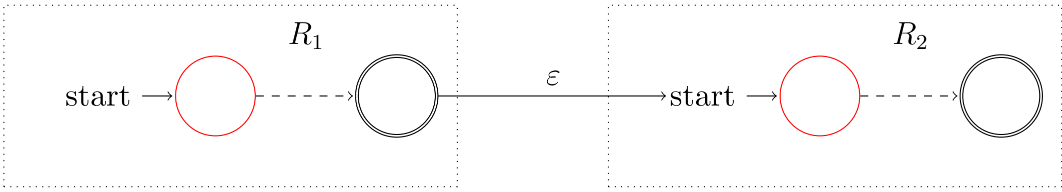


4) Автомат, задающий $R_1|R_2$ (для которых уже есть свои автоматы)



Т.е. берем старые автоматы для R_1 и R_2, после чего добавляем новое стартовое состояние, из которого добавляем переходы по ε в стартовые состояния старых автоматов, добавляем новое терминальное состояние, в которое добавляем переходы по ε из старых терминальных состояний (по построению таких ровно два). Заметим, что теперь мы должны сказать, что старые стартовые и терминальные состояния больше не являются таковыми, так как их должно быть по одному (в точности новые). Тогда все условия будут выполнены.

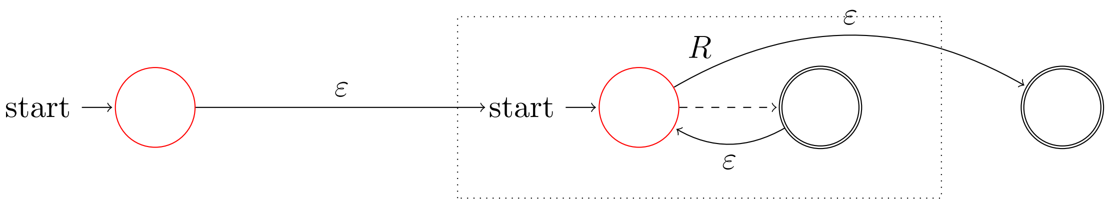
5) Автомат, принимающий R_1R_2



Т.е. берем старые автоматы для R_1 и R_2, добавляем ε-переход из терминального состояния R_1 в стартовое R_2. После чего говорим, что единственное стартовое состояние – стартовое состояние R_1, а единственное терминальное – терминальное состояние R_2.

Замечание. На самом деле, можно просто схлопнуть терминальное состояние R_1 и стартовое R_2.

6) Автомат, принимающий R^*



Т.е. добавляем новое терминальное состояние, ε-переход из терминального состояния автомата, распознающего R в стартовое состояние, а так же из стартового состояния ε-переход в новое терминальное состояние. Для того, чтобы выполнить условие (b), добавим новое стартовое состояние и ε-переход из него в стартовое состояние автомата, распознающего R. Т.е. терминальное и стартовое состояние у нас новые.

Таким образом, мы разберем регулярное выражение (формально, построим дерево разбора, применим индукцию), после чего получим автомат с $\mathcal{O}(|R|)$ состояниями. Причем данный алгоритм работает также за линейное время.

2. Автоматные языки $\subseteq$ регулярные.
ДКА $A \to$ АРВ $R : L(R) = L(A)$
$R_{i,j,k}$ – АРВ, задающее язык всех слов, переводящих автомат A из состояния i в состояние j, использующая промежуточные состояния только меньшие k (в данном автомате состояния $\{0, 1 \ldots, n-1\}$)

Слой $k = 0$

1) $i = j$ $\quad R_{i,i,0} = \varepsilon|a_1|a_2|\ldots|a_k$, где $a_1, a_2, \ldots, a_m$ – буквы, по которым есть петля $i \to i$.

2) $i \neq j$ $\quad R_{i,j,0} = a_1|a_2|\ldots|a_k$, где $a_1, a_2, \ldots, a_m$ – буквы, по которым есть переход $i \to j$ и $\varnothing$, если таких ребер нет.

Слой $k > 0$

$R_{i,j,k} = R_{i,j,k-1}|R_{i,k-1,k-1}R^*_{k-1,k-1,k-1}R_{k-1,j,k-1}$

Ответ

$R = R_{q_0,t_1,n}|R_{q_0,t_2,n}|\ldots|R_{q_0,t_{|T|},n}$ или $\varnothing$, если терминальных состояний нет.
Тогда $L(R) = L(A)$.
Оценим время работы: $\mathcal{O}^*(4^n)$. Размер ответа такой же.

□

регулярные = автоматные

4.2. Лемма о разрастании

Лемма. L – регулярный $\Rightarrow \exists n \in \mathbb{N}$, что $\forall w \in L$, такого, что $|w| \geq n$, $\exists x, y, z \in \Sigma^*$ такие, что
$$\begin{cases} w = xyz \\ y \neq \epsilon \\ |xy| \leqslant n \\ \forall iin\mathbb{Z}_{0+} xy^i z \in L \end{cases}.$$

Доказательство. A – ДКА, $L(A) = L$, $n = |Q|$

TODO: картинка автомата

Так как длина больше, чем $|Q|$, то мы где-то зациклимся. Возьмём в качестве y первый цикл, и всё получится.

□

Пример. L = ПСП. Докажем, что он нерегулярный

Доказательство. Пусть не так, тогда возьмём последовательность из n открывающих, а затем n закрывающих скобок. Для неё существуют соответствующие x, y, z из Леммы. Но строка y состоит только из открывающих скобок исходя из условий Леммы. Таким образом, если мы повторим строку y больше раз, то получится не ПСП. Получили противоречие. □

4.3. Динамическое программирование по ДКА

Пример. Дан регулярный язык, найти кратчайшее слово, принадлежащее ему.

TODO: картинка графа

Пример. Количество слов длины l в L.

$a_{q,i}$ – количество слов длины i, переводящих A из q_0 в q.

Чтобы пересчитать эту величину, нужно просуммировать ячейки из предыдущего слоя (по длине) по всем входящим в состояние рёбрам.

У нас есть правило, по которому линейной комбинацией из столбца выводится следующий, поэтому можно посчитать n-ый столбец даже для очень больших n с помощью возведения матрицы в степень.

Ответ – это сумма элементов столбца, соответствующих терминальным вершинам.

5. Формальные грамматики

Нужны более хорошие способы описания языков, так как автоматы могут распознать очень мало языков. Для этого (и лингвистами в том числе) используются конструкции, которые называются "формальными грамматиками"

5.1. Контекстно-свободные грамматики

Пример.

$S \to AB$

$S \to ABC$

$A \to DA$

$A \to Person$

$B \to sits$

$C \to atatable$

$D \to livefunny$

***Определение* 5.1.** Формальная грамматика – это четвёрка $G = (N, T, S, P)$, где:

T – терминальные символы (алфавит)

N – нетерминальные символы (синтаксические категории)

$S \in N$ – стартовый символ

P – множество продукций

***Определение* 5.2.** КС-грамматики: P – множество продукций имеет вид $N \to (N \cup T)^*$, то есть $P \in 2^{N \times (N \cup T)^*}$ – конечное

***Определение* 5.3.** Из слова w выводится слово u за один ход ($w \Rightarrow u$), где $u, w \in (N \cup T)^*$, если $w = \alpha B \gamma$, $u = \alpha\beta\gamma$, $B \to \beta \in P$.

***Определение* 5.4.** Из слова w выводится слово u ($w \Rightarrow^* u$), если $\exists w_0, w_1, \ldots, w_n$, где $w_0 = w, w_n = u$ и $\forall 0 \le i \le n - 1 w_i \Rightarrow w_{i+1}$.

***Определение* 5.5.** $L(G) = w \in T^* : S \Rightarrow^* w$

Обозначение.

стартовый символ – S

нетерминалы – заглавные латинские буквы

терминалы – строчные или цифры

вместо $A \to \alpha, A \to \beta$ пишем $A \to \alpha|\beta$

Пример.

$S \to 0S|\varepsilon$, $S \Rightarrow 0S \Rightarrow 00S \Rightarrow 000S \Rightarrow 000, L(G) = 0^*$.

$S \to 0S|1S|\varepsilon$, $L(G) = (0|1)^*$, однозначная

$S \to 0S|1S|S0|s1|\varepsilon$, неоднозначная (одна строка может быть получена по-разному)

$S \to 0S1|\varepsilon$, $L(G) = 0^n 1^n$

$S \to \varepsilon|SS|\,(S)$, $S \Rightarrow SS \Rightarrow (S)\,S \Rightarrow ((S))\,S \Rightarrow (())\,S \Rightarrow (())\,(S) \Rightarrow (())\,()$ – ПСП

***Определение* 5.6.** Из w левосторонне выводится u за один ход ($w \Rightarrow^*_{lm}$) , если $w = \alpha B \gamma$, $u = \alpha\beta\gamma$, $B \to \beta \in P$ и $\alpha \in T^*$.

***Определение* 5.7.** Из w парвосторонне выводится u за один ход ($w \Rightarrow^*_{rm}$), если $w = \alpha B\gamma$, $u = \alpha\beta\gamma$, $B \to \beta \in P$ и $\gamma \in T^*$.

Однозначная грамматика для ПСП: $(S)\,S|\varepsilon$

5.2. Дерево разбора

Пример. Дерево разбора однозначной граматики для ПСП, строка $(())\,()$ **TODO:** картинка

Грамматика однозначна, если для неё есть ровно одно дерево разбора, оно же дерево левостороннего и правостороннего вывода. Таким образом, если слово левосторонне выводится однозначно, то и правосторонне тоже и наоборот.

Теорема 5.1. Регулярные $\subsetneq$ КС-языки

Доказательство. $\neq$: 0^n1^n разпознаётся КС-грамматикой, не распознаётся автоматом

$\subset$: ДКА $A \to$ КС-грамматика G.

$N = Q$, $q \to ap$, где $p = \delta(q, a)$.

□

5.3. Преобразования КС-грамматик

- **Меняем стартовый символ**

Грамматика обладает следующим свойством: $\varepsilon \in L(G)$.
По договоренности, вводятся стартовый символ $S \to \varepsilon|S'$, а из $S' \Rightarrow^* L(G) \setminus \{\varepsilon\}$.
После этого можно считать, что $\varepsilon \notin L$.

***Определение* 5.8.** $A \in N$ – ε-порождающий, если $A \Rightarrow^* \varepsilon$.

- **Избавление от ε-порождающих**

1) $A \to \varepsilon$ удаляем

2) $A \to A_1 \ldots A_n$, $A_i \in N \cup T$. Некоторое из A_i могли порождать ε. Но, может быть язык не совпадает, с тем, что было раньше. Вместе с ней добавляем все продукции вида
$A \to$ подпоследовательность $A_1 \ldots A_n$, только если все удаленные символы – ε-порождающие терминалы и эта подпоследовательность не пустая.

Пример.

$$\begin{cases} S \to aBCD \\ S \to BB \\ B \to \varepsilon|b \\ C \to \varepsilon|c \\ D \to d \end{cases}$$
$$\to$$

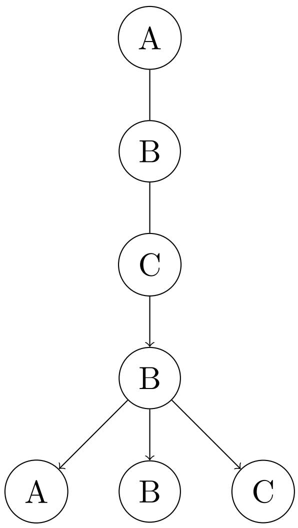
$$\begin{cases} S' \to aD|aBD|aCD|aBCD \\ S' \to BB|B \\ B \to b \\ C \to c \\ D \to d \\ S \to S'|\varepsilon \end{cases}$$

Время работы $\mathcal{O}(|G| \cdot 2^{max\ right\ part\ len})$

- **Удаление цепных продукций**

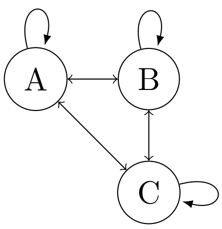
$A \to B; A, B \in N$
Построим транзитивное замыкание отношения "есть продукция $A \to B$"
Для любого ребра $A \Rightarrow^* B$ и для любой продукции $B \to \gamma$ получим продукцию $A \to \gamma$

Пример.



Рассмотрим следующие правила.

Преобразуем в



Время работы $\mathcal{O}(\frac{|N|^3}{w} + |G| \cdot |N|)$

- **Удаление бесполезных нетерминалов**

1) Удаление непорождающих нетерминалов $A \in N$ – непорождающий, если $\nexists w \in T^* : A \Rightarrow^* w$. Реализационно – фазами или очередями с событиями.

2) Оставляем только достижимые из S нетерминалы.

Пример.

$$\begin{cases} (\checkmark)S \to AB|B \\ (-)A \to Aa|AC \\ (\checkmark)B \to b \\ (\checkmark)C \to A|S \end{cases}$$

Время работы $\mathcal{O}(|G|)$

- **Удаление длинных правых частей**

$A \to A_1 \dots A_n, A_i \in (N \cup T), n \leq 3$
Добавляем новые нетерминалы $\mathcal{O}(|G|)$
$A \to A_1 B_1$
$B_1 \to A_2 B_2$
$B_2 \to A_3 B_3$
$\dots$
$B_n \to A_{n-1} A_n$

- **Удаление терминалов в правых частях**

Хотим терминал в правой части только в $A \to a$
$\forall$ терминала a, создаем уникальный нетерминал A, продукцию $A \to A$ и $\to \dots a \dots$ меняем на $\to \dots A \dots$.

Резюмируем:
$A \to BC$
$A \to a$
все нетерминалы – полезные (достижимые из стартового, порождающие)

***Определение* 5.9.** G записанная в этом виде – К.С. грамматика в нормальной форме Хомского.

5.4. Алгоритм Кока-Янгера-Касами (CYK)

G – КС-грамматика в НФ Хомского, $w \in T^*$, хотим понять, $w \in L(G)$ или нет.

Алгоритм (Кока-Янгера-Касами)**.** $M_{i,j,A}$ – верно ли, что $A \Rightarrow^* w_{i \dots j}$, где $i, j, \in [1; |w|], A \in N$.

1) $i = j$, $M_{i,i,A} = $ [Есть продукция $A \to w_i$]

2) $i < j$, $M_{i,j,A} = \bigvee\limits_{A \to B,C} \bigvee\limits_{k=i}^{j-1} M_{i,k,B} \wedge M_{k+1,j,C}$

Время работы $\mathcal{O}(|w|^3 \cdot |G|)$.

Замечание. Можно применить идею четырех русских.

6. Конечные автоматы с магазинной памятью

МП-автомат (Pushdown automata). Бывают детерминированные и недетерминированные (по умолчанию недетерминированные). Промежуточное звено между ДКА и машиной Тьюринга.

У нас есть стек, который представляет из себя строчку с указателем на верхушку. Можем обращаться к верхушке, делать push и pop.

***Определение* 6.1.** МП-автомат – это семёрка $A = (\Sigma, \Gamma, Q, q_0, z_0, T, \delta)$, где Q, Σ, Γ – конечные, $q_0 \in Q, z_0 \in \Gamma, T \subseteq Q, \delta : Q \times \Sigma \times \Gamma \to 2^{Q \times \Gamma^*}$. Все возможные значения δ являются конечными множествами.

Γ – это стековый алфавит, z_0 – символ на верхушке стека.

***Определение* 6.2.** Моментальное описание МП-автомата – это $(w, \alpha, q) \in \Sigma^* \times \Gamma^* \times Q$.

***Определение* 6.3.** $(xw, g\alpha, q) \vdash (w, \gamma\alpha, p)$, если $(p, \gamma) \in \delta(q, x, g)$.

xw – то, что предстоит обработать $(x \in \Sigma \cup \{\varepsilon\})$, $g\alpha$ – то, что сейчас лежит на стеке, $(g \in \Gamma)$.

***Определение* 6.4.** $\vdash^*$ (аналогично $\Rightarrow^*$).

Принципы принятия слова:

1. По терминальным состояниям, $w \in L(A) \Leftrightarrow \exists \gamma \in \Gamma^*, p \in T : (w, z_0, q_0) \vdash^* (\varepsilon, \gamma, p)$.
2. По пустому стеку, $w \in L(A) \Leftrightarrow \exists p \in Q : (w, z_0, q_0) \vdash^* (\varepsilon, \varepsilon, p)$.

Замечание. Из определения можно удалить T, что мы и сделаем.

Обозначение. $(p, \gamma) \in \delta(q, x, g)$ **TODO:** картинка с двумя вершинами q и p и между ними стрелочка, на которой написано $x, q/\gamma$.

Пример. НКА A $\to$ МП-автомат B

$L(A) = L(B)$ по терминальным состояниям.

Регулярные языки $\subseteq$ МП-автоматные.

Пример. $L = \{0^n 1^{2n}\}$ **TODO:** картинка автомата Пока считываем нолики, кладём по два символа z_0. Когда начинаем считывать единички, снимаем по одному символу z_0.

В то время, как $\{0^n 1^n 2^n\}$ не МП-автоматный.

Теорема 6.1. Множество слов, принимаемое МП-автоматом по терм. состоянию равномощно множеству, принимаемому по пустому стеку.

Доказательство. **TODO:** картинка □

Теорема 6.2. МП-автоматы и КС-грамматики распознают одинаковые языки

Доказательство. **МП-автомат** A $\leftarrow$ **КС-грамматика** G

S $\Rightarrow^{lm}$ терминал $N \Rightarrow w$.

$\Sigma =$ терминалы

$\Gamma = N \cup \Sigma \cup \{z_0\}$

$Q = \{q_0\}$

$\forall A \to v, \forall t \in \Sigma$

по пустому стеку

МП-автомат $A \leftarrow$ **КС-грамматика** G

Автомат берём по пустому стеку.

$N = Q \times \Gamma \times Q$.

$[pgq]$ – это синтаксическая категория, которая описывает все следующие слова (сейчас будет объяснено, какие).

TODO: Картинка "жизнь стека"

Был ровно один момент, когда данная буква была снята со стека.

$[pgq]$ – это синтаксическая категория, которая описывает все слова, которые могут снять со стека символ g, переведя автомат из состояния p в состояние q.

Взяли любое ребро, которое считывает x и переводит g в γ.

Слова из той синтаксической категории имеют следующий вид $[pgq] \to x[q\gamma_1 r_1][r_1\gamma_2 r_2] \dots [r_{k-1}\gamma_k r_k$ $\forall r_1, r_2, \dots, r_k \in Q^k$.

Нам нужен стартовый символ S, добавляем правило $S \to [q_0 z_0 q_0]|[q_0 z_0 q_1]| \dots |[q_0 z_0 q_{last}]$.

Тут нет никаких противоречий, потому что чтобы снять что-то со стека нас должны быть переходы к пустому γ.

Форма всех продукций выглядит так: $A \to tN_1N_2 \dots N_k$ (ослабленная нормальная форма Грейбаха). Если бы было $\leqslant 2$ нетерминалов, то это называлось бы нормальной формой Грейбаха.

□

6.1. Детерминированные конечные автоматы с магазинной памятью

***Определение* 6.5.** Все тоже самое (т.е. $(\Sigma, \Gamma, Q, T, \delta, z_o, q_o)$), но
$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to Q \times \Gamma^* \cup \{\perp\}$ и, если $\delta(q, \epsilon, g) \to (p, \gamma)$, то нет $\delta(q, a, g) \to (,)$

***Определение* 6.6.** Язык называется префиксным, если $\forall w \neq v \ \in L$ не верно, что w – префикс v.

Теорема 6.3. Язык принимается ДМП по пустому стеку $\Leftrightarrow$ **TODO:** система L принимается ДМП по терминальному состоянию L – префиксный

Доказательство. **TODO:** доказательство по видео, не успеваю

$\Rightarrow$ Покажем префиксность. Теперь покажем, что он принимается ДМП по терминальному состоянию.

$\Leftarrow$ План: строим, избавляемся от "барахла" (получили пустой стек раньше времени, лечим вторым дном) □

Теорема 6.4. Регулярные языки $\subsetneq$ ДМП-языки по терминальному состоянию $\subsetneq$ КС-языки

Доказательство.

1) Регулярные языки $\subsetneq$ ДМП-языки по терминальному состоянию
   Вложение: Берем дно, кладем обратно. $\neq \{0^n 1^{2n}\}$ не является регулярным по лемме о накачке.

2) ДМП-языки по терминальному состоянию $\subsetneq$ КС-языки
Рассмотрим язык $L = \{0^n 1^n\} \cup \{0^n 1^2 n\}$ Покажем, что ДМП-язык.
От противного. Тогда язык принимается ДМП A: $L(A) = L$ Делаем полную копию A, единичку меняем на двоечки (буквы языка), а также проведем ребро из каждого терминального состояния проведем переход по ϵ в его копию. Получился МП-автомат B. Покажем, что $L(B) = \{0^n 1^n\} \cup \{0^n 1^{2n}\} \cup \{0^n 1^n 2^n\}$. То, что он все это принимает очевидно получается разбором случаев (но **TODO:**). Покажем, что не принимает ничего другого. Если мы приняли какое-то слово в первой версии, то очевидно. Пусть в какой-то момент перешли вниз. Значит, сверху были в $0^n 1^n$ или $0^n 1^{2n}$. Мысленно перенесем путь во второй версии в первую версию и все получится. Позже мы докажем, что этот язык не КС-свободный. Мы знаем, что МП-автоматы эквиваленты КС-грамматикам, тогда покажем, что исходный язык не ДМП.

□

Лемма (о накачке для КС-языков)**.** Пусть L – КС-язык, тогда $\exists n \in \mathbb{N} : \forall w \in L : |w| \geq n : \exists u, v, x, y, z \in \Sigma^* :$

$$\begin{cases} w = uvxyz \\ vy \neq \epsilon \\ |vxy| \leq n \\ \forall i \in \mathbb{N}_0 uv^i xy^i z \in L \end{cases}$$

Доказательство.
Перейдем в нормальную форму Хомского. $n = 2^{|N|}$, пусть $|w| \geq n$. Возьмем самый длинный путь. Его длинна будет хотя бы $|N| + 1$. По принципу Дирихле есть хотя бы два одинаковых нетерминала. Возьмем самую нижнюю такую пару. Верхний из этих двух терминалов породит строку длиной не более, чем $2^{|N|}$. Остальные части на **TODO:** картинке. □

Пример. $\{0^n 1^n\} \cup \{0^n 1^{2n}\} \cup \{0^n 1^n 2^n\}$ не является КС. От противного. Рассмотрим n из леммы о накачке и слово $0^{2n} 1^{2n} 2^{2n}$. Откачаем, но vy не содержит по крайней мере одну цифру. Значит, получим слово не из языка. Противоречие.

6.2. Иерархия Хомского

0) Произвольные грамматики

1) Контекстно-зависимые

2) Контекстно-свободные

3) Регулярные

***Определение* 6.7.**
Праволинейной грамматикой назовем такую грамматику, что $A \to a$ и $A \to aB$.
Левоолинейной грамматикой назовем такую грамматику, что $A \to a$ и $A \to Ba$.
Тогда регулярные грамматики это леволинейные $\cap$ праволинейные. Есть прямое соответствие между ДКА и грамматиками. Т.е. то, что задается леволинейными задается и парволинейными и наоборот.

***Определение* 6.8.**
Контекстно-зависимые грамматики. $\alpha B \gamma \to \alpha \beta \gamma$, где $\alpha, \gamma \in (\Sigma \cup N)^*, B \subset N, \beta \in (\Sigma \cup N)^+$

***Определение* 6.9.**

Неукорачивающая грамматика, если все продукции имею вида$\alpha \to \beta$, где α содержит хотя бы один нетерминал, $|\alpha| \le |\beta|$

Замечание. $\alpha_1 \dots \alpha_n \to \beta_1 \dots \beta_m$, где $m \ge n$. Заменим на

$$\begin{cases} \alpha_1 \dots \alpha_n \to A\alpha_1 \dots \alpha_n \\ A_1\alpha_2 \dots \alpha_n \to A_1A_2 \dots \alpha_n \\ \dots \\ A_1A_2 \dots A_{n-1}\alpha_n \to A_1A_2 \dots A_n \\ A_1A_2 \dots A_n \to \beta_1A_2 \dots A_n \\ \beta_1 \dots \beta_{n-1}A_n \to \beta_1\beta_2 \dots \beta_m \end{cases}$$

Теорема 6.5.

Любую КС-зависимую (неукорачивающую) грамматику можно привести к Н.Ф. Куроды:
$AB \to CD$
$A \to BC$
$A \to a$
TODO: тут протек мозг и расплавленный корпус ноута (утомленные солнцем). Но вообще, похоже на привидение к Хомскому.

***Определение* 6.10.**

Произвольные грамматики: $\alpha \to \beta$ в α есть ≥ 1 нетерминал.

Замечание.

Любую произвольную грамматику можно привести к Н.Ф. Куроды для произвольных грамматик: (все тоже самое, но еще легально $A \to \epsilon$)
Доказательство: введем дополнительный символ Z, полечим укорачиваемость, потом $Z \to \epsilon$.

Теорема 6.6.

Произвольные формальные грамматики эквиваленты МТ (Тьюринг-полны).

Доказательство.

1. $\Rightarrow$ Ставим S на ленту, едим на лево, выбираем правило (недетерминировано, но НМТ эквиваленты ДМТ), делаем $\alpha \to \beta$.
2. $(ac, q) \to (bc, p)$
   TODO: посмотреть видео, записать нормально.

□