

Теория формальных языков

Тух Игорь и Ютман Михаил
по лекциям Михаила Эдуардовича Дворкина

16 июня 2018 г.

Содержание

1. Введение	1
2. Детерминированные Конечные Автоматы (DFA)	2
2.1 Детерминированные Конечные Автоматы	2
2.2 Минимизация ДКА	3
2.3 Правые контексты	3
3. Недетерминированные конечные автоматы с ε-переходами	5
3.1 Недетерминированные конечные автоматы с ε -переходами	5
3.2 Детерминизация ε -НКА	7
3.3 Произведение конечных автоматов	8
4. Регулярные выражения	9
4.1 Академические регулярные выражения	9
5. Ещё про детерминированные конечные автоматы	13
5.1 Лемма о разрастании	13
5.2 Динамическое программирование по ДКА	14
6. Формальные грамматики	15
6.1 Контекстно-свободные грамматики	15
6.2 Дерево разбора	16
6.3 Преобразования КС-грамматик	17
6.4 Алгоритм Кока-Янгера-Касами (СЮК)	19
7. Конечные автоматы с магазинной памятью	20
7.1 Детерминированные конечные автоматы с магазинной памятью	21
7.2 Иерархия Хомского	22

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 \emptyset .

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)

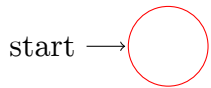
2.1. Детерминированные Конечные Автоматы

Определение 2.1. Детерминированный конечный автомат $A = (\Sigma, Q, q_0, T, \delta)$, где

1. Σ – конечный алфавит;
2. Q – конечное множество состояний;
3. $q_0 \in Q$ – начальное состояние;
4. $T \subseteq Q$ – множество терминальных состояний;
5. $\delta : Q \times \Sigma \rightarrow Q$ – функция переходов.

Обозначение.

Далее в конспекте так будет обозначаться стартовые состояния:

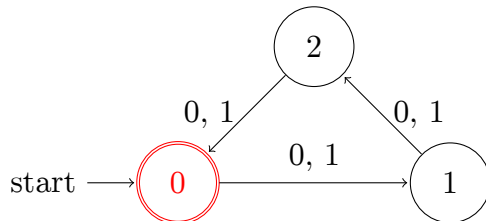


А вот так терминальные:



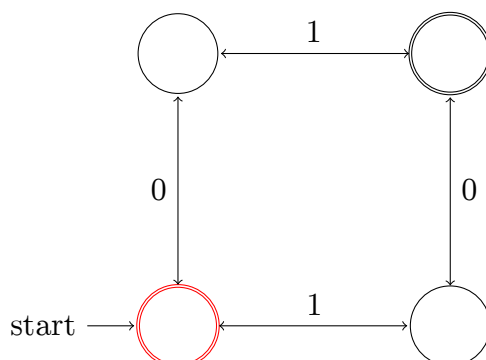
Примеры.

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



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 $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$, причем $\hat{\delta}(q, \varepsilon) = q$ и $\hat{\delta}(q, xw) = \hat{\delta}(\hat{\delta}(q, x), w)$. И тогда язык принимаемый ДКА $L(A) = \{w \in \Sigma^* : \hat{\delta}(q_0, w) \in T\}$.

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

2.2. Минимизация ДКА

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

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

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

Доказательство. Рефлексивность, симметричность и транзитивность очевидны. □

Алгоритм (поиска всех пар различимых состояний).

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

Рассмотрим граф, построенный на парах состояний. Для пар из пункта 1 мы знаем, что они различимы. Более того, понятно, что для каждой различимой пары (p, q) существует такая пара $(\hat{\delta}(p, w), \hat{\delta}(q, w))$, где w взято из определения различимой пары (*). Тогда построим граф на таких парах, где будет проведено ребро $(p, q) \rightarrow (u, v)$, если $\exists a \in \Sigma : \delta(u, a) = p$ и $\delta(v, a) = q$. Тогда для нахождения всех различимых пар, достаточно запустить *DFS* или *BFS* в таком графе от вершин из пункта 1, помечая все достигнутые пары, как различимые. Понятно, что если пара различимая, то по (*), мы найдем все такие пары. Время работы будет $\mathcal{O}(|Q|^2)$, так как в этом графе $|Q|^2$ вершин и $|Q|^2 \cdot |\Sigma|$ ребер.

Замечание. Можно не разворачивать ребра, а просто запустить ленивую динамику.

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 $C_L^R : \Sigma^* \rightarrow 2^{\Sigma^*}$, $C_L^R(w) = \{u \in \Sigma^* : wu \in L\}$

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) – это пятёрка $A = (\Sigma, Q, q_0, T, \delta)$, где

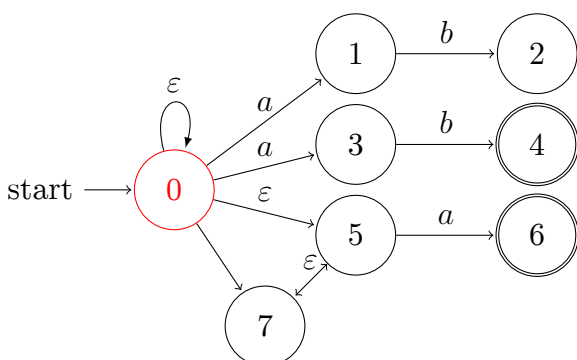
1. Σ – конечный алфавит
2. Q – конечное множество
3. $q_0 \in Q$ – начальное состояние
4. $T \subseteq Q$ – множество терминальных состояний
5. $\delta : Q \times (\{\varepsilon\} \cup \Sigma) \rightarrow 2^Q$ – функция переходов (по состоянию и символу получаем множество состояний, в которое мы можем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

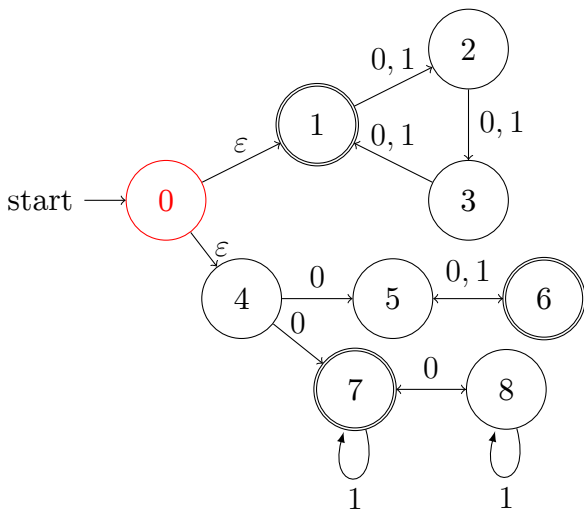


В этот автомат слово "a" можно принять перейдя из 0 в 3, а можно перейти по ε -переходу из 0 в 5, потом походить миллион раз туда-обратно по ε -переходу между 5 и 7, а потом из 5 перейти в 6, и таким образом, это слово распознаётся автоматом.

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

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

Пример.



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

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 $\leq |Q|$ 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" ε -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:

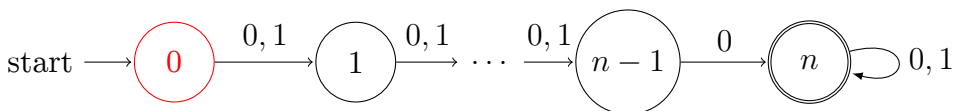
1. $\Sigma_2 = \Sigma_1$, то есть алфавит берём такой же
2. $Q_2 = 2^{Q_1}$, то есть смотрим на множество всех достижимых по данной строке состояний
3. $q_{02} = \varepsilon\text{-closure}(q_{01})$, то есть вместо стартового состояния смотрим на его ε -замыкание.
4. $T_2 = \{S \subset Q : S \cap T \neq \emptyset\}$, то есть все подмножества множества состояний, содержащие хотя бы одно терминальное состояние
5. $\delta_2(S, a) = \bigcup_{p \in S, p \rightarrow q} \varepsilon\text{-closure}(q)$, 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 распознаваемых 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 получится не полная экспонента, то есть набор множеств достижимых вершин, который может получиться скорее всего будет сильно меньше, чем просто все подмножества.

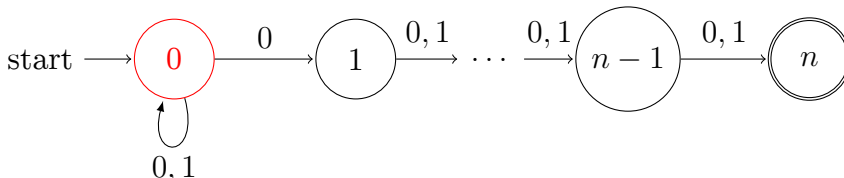
Также dfs для подсчета замыкания можно запускать только из достижимых вершин, и только когда мы в них первый раз заходим. Получается такая ленивая ДП.

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	$L(R)$
\emptyset	\emptyset
ε	$\{\varepsilon\}$
$a \quad (\forall a \in \Sigma)$	$\{a\}$
$R_1 R_2$	$L(R_1) \cup L(R_2)$
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. Предпоследняя операция называется "замыкание Клини". Получили рекурсивное определение, т.е. выражение, полученное в результате применения нескольких правил выше и будет называться академическим регулярным выражением.

Примеры.

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

Замечание.

- $RR^* = R_+$
- $RR(\varepsilon|R)(\varepsilon|R) = R\{2-4\}$
- 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 множеством языков, задаваемых конечными автоматами.

Доказательство.

- Докажем, что регулярные языки \subseteq автоматные.

$$\forall R \rightarrow \varepsilon\text{-НКА } A : L(A) = L(R).$$

Далее, аналог индукции:

Итак, будем строить автомат, в котором будут выполнены следующие условия:

- ровно одно терминальное состояние
- в начальном состоянии нет переходов

- (с) из терминального состояния нет переходов
- (d) начальное и терминальное состояние не совпадают

Итак, построение:

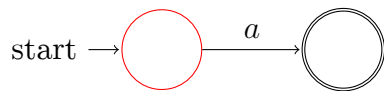
- 1) Автомат, задающий пустой язык:



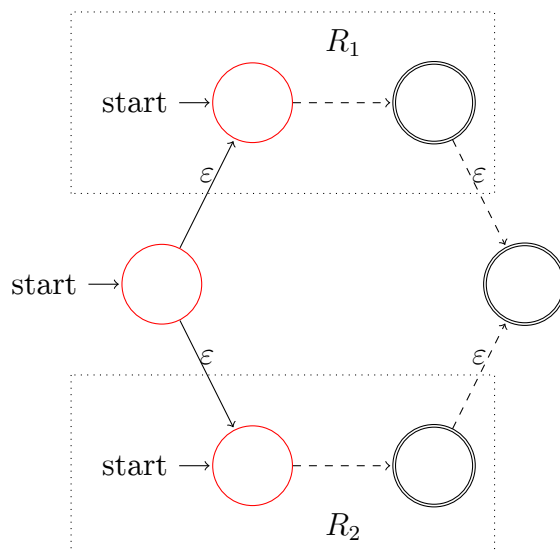
- 2) Автомат, задающий пустое слово:



- 3) Автомат, задающий одну букву a :

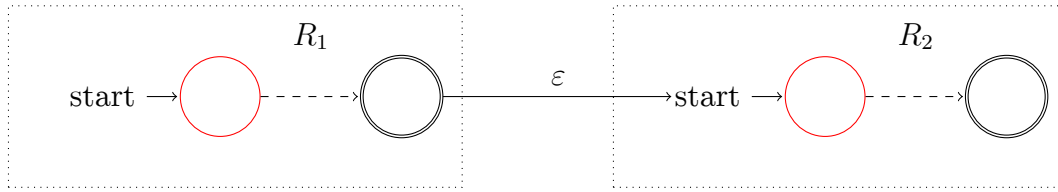


- 4) Автомат, задающий $R_1|R_2$ (для которых уже есть свои автоматы)



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.

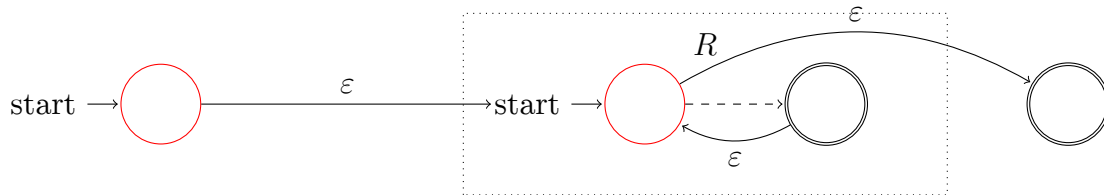
5) Автомат, принимающий R_1R_2



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 $A \rightarrow \text{APB}$ $R : L(R) = L(A)$

$R_{i,j,k}$ – APB, задающее язык всех слов, переводящих автомат A из состояния i в состояние j , использующая промежуточные состояния только меньшие k (в данном автомате состояния $\{0, 1 \dots, n - 1\}$)

Слой $k = 0$

- 1) $i = j$ $R_{i,i,0} = \epsilon | a_1 | a_2 | \dots | a_k$, где a_1, a_2, \dots, a_m – буквы, по которым есть петля $i \rightarrow i$.
- 2) $i \neq j$ $R_{i,j,0} = a_1 | a_2 | \dots | a_k$, где a_1, a_2, \dots, a_m – буквы, по которым есть переход $i \rightarrow j$ и \emptyset , 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

$R = R_{q_0, t_1, n} | R_{q_0, t_2, n} | \dots | R_{q_0, t_{|T|}, n}$ или \emptyset , если терминальных состояний нет.

Тогда $L(R) = L(A)$.

Оценим время работы: $\mathcal{O}^*(4^n)$. Размер ответа такой же.

□

5. Ещё про детерминированные конечные автоматы

5.1. Лемма о разрастании

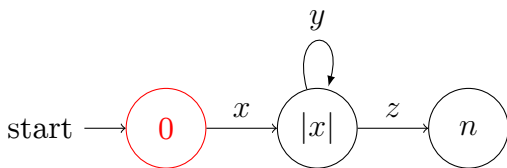
Лемма. Для любого регулярного языка L существует $n \in \mathbb{N}$, что любое слова $w \in L$ длины $\geq n$ представляется в виде xyz , где $x, y, z \in \Sigma^*$ и выполняются следующие условия:

1. $y \neq \varepsilon$
2. $|xy| \leq n$
3. $\forall i \in \mathbb{Z}_+ \cup \{0\}$ верно, что $xy^iz \in L$

Доказательство. Так как язык L регулярный, то для него есть ДКА A , такой, что $L(A) = L$. Тогда в качестве n нам подойдёт $|Q|$.

Если мы возьмём строку длины больше, чем $|Q|$, и будем последовательно переходить по её символам в автомате, то к тому моменту, как мы пройдем $|Q|$ символов, мы в каком-то состоянии побываем дважды.

Заметим, что если бы мы прошли по циклу ещё сколько угодно раз или же вовсе не прошли бы, слово не перестало бы лежать в языке. Значит, мы можем взять самый первый цикл, в который мы попали в качестве y , а его предпериод в качестве x , тогда $|xy| < |Q|$, а так как чтобы заиклиться мы должны были пройти по каким-то символам, то $y \neq \varepsilon$.



□

Замечание. Любой конечный язык регулярен, так как можно взять n такой, что любое слово имеет длину $< n$, например, $n := \max_{w \in L} |w| + 1$

Пример. Язык Дика, ака $L = \text{ПСП}$. Докажем, что он нерегулярный

Доказательство. Предположим, что он регулярен, тогда по лемме о накачке существует n с вышеописанными свойствами. Возьмём последовательность из n открывающих, а затем n закрывающих скобок. Для неё существуют соответствующие x, y, z из Леммы. Но так как $|xy| \leq n$, то y состоит только из открывающих скобок, причём по условию Леммы y не пустая. А значит при $i = 2$ в строке xy^iz получится больше открывающих скобок, чем закрывающих, то есть это будет не ПСП. Получили противоречие. □

Доказательство. (С помощью правых контекстов)

Если язык регулярен, то количество его различных правых контекстов равно количеству состояний в минимальном автомате, то есть конечно. Посмотрим, что можно сказать про правые контексты в языке Дика.

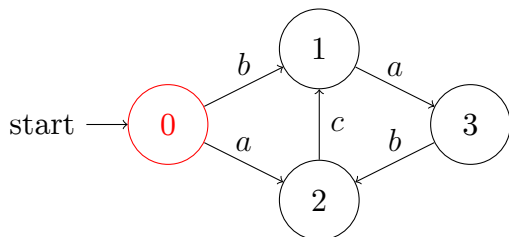
w_n – строка из n открывающих скобок. Правый контекст для w_n содержит строку из n закрывающих скобок, но при этом ни для какого $m \neq n$ правый контекст w_m эту строку не содержит по определению ПСП.

Таким образом, для всех w_n правые контексты различаются, значит их бесконечное число, а значит язык Дика нерегулярен. □

5.2. Динамическое программирование по ДКА

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

Запускаем bfs, выходим, когда пришли в терминальное состояние + восстановление ответа.



Кратчайшее слово в языке, распознаваемом этим автоматом – ba .

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 значения ДП из предыдущего по длине слоя для всех состояний, из которых есть ребро в q .

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.

6. 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"

6.1. Контекстно-свободные грамматики

Пример.

$$\begin{aligned} S &\rightarrow AB \\ S &\rightarrow ABC \\ A &\rightarrow DA \\ A &\rightarrow Person \\ B &\rightarrow sits \\ C &\rightarrow atatable \\ D &\rightarrow livefunny \end{aligned}$$

Определение 6.1. Формальная грамматика – это четвёрка $G = (N, T, S, P)$, где

1. T – терминальные символы (алфавит)
2. N – нетерминальные символы (синтаксические категории)
3. $S \in N$ – стартовый символ
4. P – множество продукций

Определение 6.2. (для КС-грамматик)

Множество продукций P – это **конечное** подмножество множества отображений вида $N \rightarrow (N \cup T)^*$.

Определение 6.3. Слово u выводится из слова w за один ход ($w \Rightarrow u$), где $u, w \in (N \cup T)^*$, если $w = \alpha B \gamma$, $u = \alpha \beta \gamma$ и $B \rightarrow \beta \in P$, то есть u получается из w применением продукции к нетерминалу B .

Определение 6.4. Слово u выводится из слова w ($w \Rightarrow^* u$), если $\exists w_0, w_1, \dots, w_n$, где $w_0 = w, w_n = u$ и $\forall 0 \leq i \leq n - 1$ верно, что $w_i \Rightarrow w_{i+1}$.

Определение 6.5. Грамматика называется однозначной, если для любого слова, выводимого из стартового символа существует единственный путь вывода.

Определение 6.6. $L(G) = \{ w \in T^* : S \Rightarrow^* w \}$

Обозначение.

стартовый символ – S

нетерминалы – заглавные латинские буквы

терминалы – строчные или цифры

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

Построим по ДКА A КС-грамматику. Для этого предъявим все элементы КС-грамматики из определения

1. Σ такой же
2. $N = Q$
3. $S = q_0$
4. P – это все правила вида $q \rightarrow ap$, где $p = \delta(q, a)$.

□

6.3. Преобразования КС-грамматик

Мотивация: хотим понять, подходит ли слово под данную грамматику или нет. Пока мы этого не умеем. Хочется так преобразовать грамматику, чтобы было достаточно легко ответить на поставленную задачу. Итак, совершим следующие преобразования:

- **Меняем стартовый символ**

Грамматика обладает следующим свойством: $\varepsilon \in L(G)$.

По договоренности, вводятся стартовый символ $S \rightarrow \varepsilon | S'$, а из $S' \Rightarrow^* L(G) \setminus \{\varepsilon\}$.

После этого можно считать, что $\varepsilon \notin L$.

Определение 6.11. $A \in N$ – ε -порождающий, если $A \Rightarrow^* \varepsilon$.

- **Избавляемся от ε -порождающих**

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

Пример.

$$\begin{aligned} & \left\{ \begin{array}{l} S \rightarrow aBCD \\ S \rightarrow BB \\ B \rightarrow \varepsilon | b \\ C \rightarrow \varepsilon | c \\ D \rightarrow d \end{array} \right. \\ & \quad \rightarrow \\ & \left\{ \begin{array}{l} S' \rightarrow aD | aBD | aCD | aBCD \\ S' \rightarrow BB | B \\ B \rightarrow b \\ C \rightarrow c \\ D \rightarrow d \\ S \rightarrow S' | \varepsilon \end{array} \right. \end{aligned}$$

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

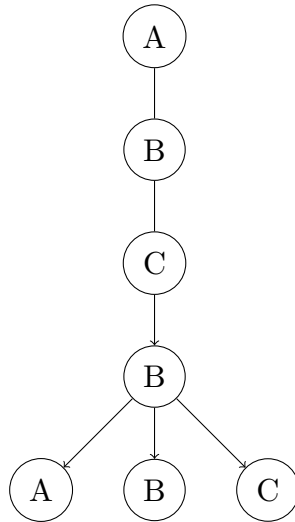
• Удаляем цепные продукции

$A \rightarrow B; A, B \in N$

Построим транзитивное замыкание отношения "есть продукция $A \rightarrow B$ "

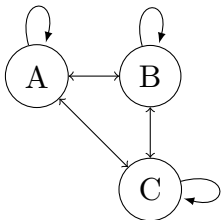
Для любого ребра $A \Rightarrow^* B$ и для любой продукции $B \rightarrow \gamma$ получим продукцию $A \rightarrow \gamma$

Пример.



Рассмотрим следующие правила.

Преобразуем в



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

• Удаляем бесполезные нетерминалы

- 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.

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

Время работы $\mathcal{O}(|G|)$

- Удаляем длинные правые частей

$$A \rightarrow A_1 \dots A_n, A_i \in (N \cup T), n \leq 3$$

Добавляем новые нетерминалы $\mathcal{O}(|G|)$

$$A \rightarrow A_1 B_1$$

$$B_1 \rightarrow A_2 B_2$$

$$B_2 \rightarrow A_3 B_3$$

...

$$B_n \rightarrow A_{n-1} A_n$$

- Удаляем терминалы в правых частях

Хотим терминал в правой части только в $A \rightarrow a$

\forall терминала a , создаем уникальный нетерминал A , продукцию $A \rightarrow A$ и $\rightarrow \dots a \dots$ меняем на $\rightarrow \dots A \dots$

Резюмируем:

$$A \rightarrow BC$$

$$A \rightarrow a$$

все нетерминалы – полезные (достижимые из стартового, порождающие)

Определение 6.12. G записанная в этом виде – К.С. грамматика в нормальной форме Хомского.

6.4. Алгоритм Кока-Янгера-Касами (СҮК)

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 $i, j \in [1; |w|]$, $A \in N$.

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

$$2) i < j, M_{i,j,A} = \bigvee_{A \rightarrow BC} \bigvee_{k=i}^{j-1} M_{i,k,B} \wedge M_{k+1,j,C}$$

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

Замечание. Можно применить идею четырех русских.

7. 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 7.1. МП-автомат – это семёрка $A = (\Sigma, \Gamma, Q, q_0, z_0, T, \delta)$, где Q, Σ, Γ – конечные, $q_0 \in Q, z_0 \in \Gamma, T \subseteq Q, \delta : Q \times \Sigma \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$. Все возможные значения δ являются конечными множествами.

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

Определение 7.2. Моментальное описание МП-автомата – это $(w, \alpha, q) \in \Sigma^* \times \Gamma^* \times Q$.

Определение 7.3. $(xw, g\alpha, q) \vdash (w, \gamma\alpha, p)$, 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 7.4. \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 $x, g/\gamma$.

Пример. НКА $A \rightarrow$ МП-автомат B

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

Регулярные языки \subseteq МП-автоматные.

Пример. $L = \{0^n 1^{2^n}\}$ **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 7.1. Множество слов, принимаемое МП-автоматом по терм. состоянию равносильно множеству, принимаемому по пустому стеку.

Доказательство. **TODO:** картинка □

Теорема 7.2. 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

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

$$Q = \{q_0\}$$

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

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 , переводя 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 $[pgq] \rightarrow x[q\gamma_1r_1][r_1\gamma_2r_2] \dots [r_{k-1}\gamma_kr_k]$
 $\forall r_1, r_2, \dots, r_k \in Q^k$.

Нам нужен стартовый символ S , добавляем правило $S \rightarrow [q_0z_0q_0][q_0z_0q_1] \dots [q_0z_0q_{last}]$.

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

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

7.1. Детерминированные конечные автоматы с магазинной памятью

Определение 7.5. Все тоже самое (т.е. $(\Sigma, \Gamma, Q, T, \delta, z_0, q_0)$), но $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow Q \times \Gamma^* \cup \{\perp\}$ и, если $\delta(q, \epsilon, g) \rightarrow (p, \gamma)$, то нет $\delta(q, a, g) \rightarrow (,)$

Определение 7.6. Язык называется префиксным, если $\forall w \neq v \in L$ не верно, что w – префикс v .

Теорема 7.3. 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 7.4. 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. $\neq \{0^n1^{2n}\}$ не является регулярным по лемме о накачке.

2) ДМП-языки по терминальному состоянию \subsetneq КС-языки

Рассмотрим язык $L = \{0^n 1^n\} \cup \{0^n 1^{2n}\}$. Покажем, что ДМП-язык.

От противного. Тогда язык принимается ДМП A : $L(A) = L$. Делаем полную копию A , единичку меняем на двоички (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 длина 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.

7.2. Иерархия Хомского

- 0) Произвольные грамматики
- 1) Контекстно-зависимые
- 2) Контекстно-свободные
- 3) Регулярные

Определение 7.7.

Праволинейной грамматикой назовем такую грамматику, что $A \rightarrow a$ и $A \rightarrow aB$.
 Леволинейной грамматикой назовем такую грамматику, что $A \rightarrow a$ и $A \rightarrow Ba$.
 Тогда регулярные грамматики это леволинейные \cap праволинейные. Есть прямое соответствие между ДКА и грамматиками. Т.е. то, что задается леволинейными задается и праволинейными и наоборот.

Определение 7.8.

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

Определение 7.9.

Неукорачивающая грамматика, если все продукции имеют вид $\alpha \rightarrow \beta$, где α содержит хотя бы один нетерминал, $|\alpha| \leq |\beta|$

Замечание. $\alpha_1 \dots \alpha_n \rightarrow \beta_1 \dots \beta_m$, где $m \geq n$. Заменяем на

$$\left\{ \begin{array}{l} \alpha_1 \dots \alpha_n \rightarrow A\alpha_1 \dots \alpha_n \\ A_1\alpha_2 \dots \alpha_n \rightarrow A_1A_2 \dots \alpha_n \\ \dots \\ A_1A_2 \dots A_{n-1}\alpha_n \rightarrow A_1A_2 \dots A_n \\ A_1A_2 \dots A_n \rightarrow \beta_1A_2 \dots A_n \\ \beta_1 \dots \beta_{n-1}A_n \rightarrow \beta_1\beta_2 \dots \beta_m \end{array} \right.$$

Теорема 7.5.

Любую КС-зависимую (неукорачивающую) грамматику можно привести к Н.Ф. Куроды:

$$AB \rightarrow CD$$

$$A \rightarrow BC$$

$$A \rightarrow a$$

TODO: тут протек мозг и расплавленный корпус ноута (утомленные солнцем). Но вообще, похоже на приведение к Хомскому.

Определение 7.10.

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

Замечание.

Любую произвольную грамматику можно привести к Н.Ф. Куроды для произвольных грамматик: (все тоже самое, но еще легально $A \rightarrow \epsilon$)

Доказательство: введем дополнительный символ Z , получим укорачиваемость, потом $Z \rightarrow \epsilon$.

Теорема 7.6.

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 $\alpha \rightarrow \beta$.

2. $(ac, q) \rightarrow (bc, p)$

TODO: посмотреть видео, записать нормально.

□