Теория формальных языков

Тух Игорь и Ютман Михаил
по лекциям Михаила Эдуардовича Дворкина

17 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)

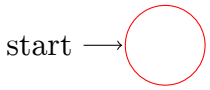
2.1. Детерминированные Конечные Автоматы

***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.

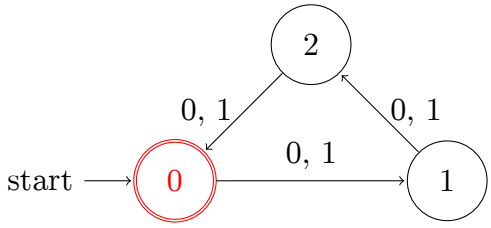
Далее в конспекте так будет обозначаться стартовые состояния:



А вот так терминальные:

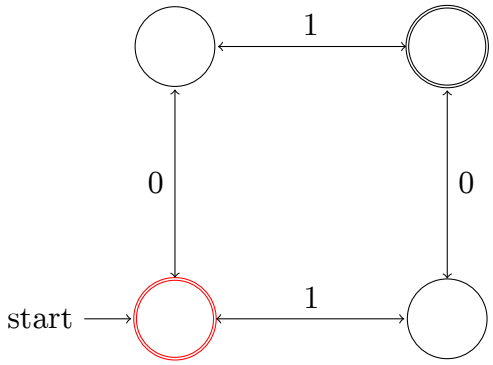
Примеры.

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 $\exists w \in \Sigma^* \;\; \hat{\delta}(p, w) \in T \oplus \hat{\delta}(q, w) \in T$.

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

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

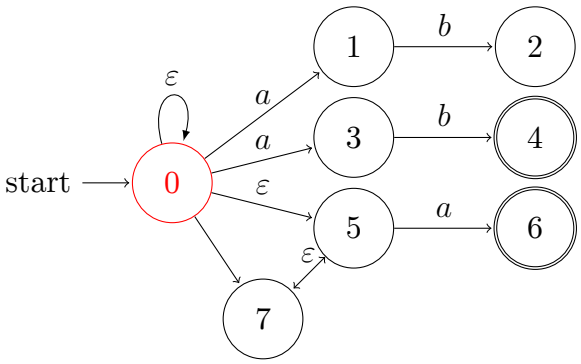
1. Σ – конечный алфавит
2. Q – конечное множество
3. $q_0 \in Q$ – начальное состояние
4. $T \subseteq Q$ – множество терминальных состояний
5. $\delta : Q \times (\{\varepsilon\} \cup \Sigma) \to 2^Q$ – функция переходов (по состоянию и символу получаем множество состояний, в которое мы можем перейти)

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:

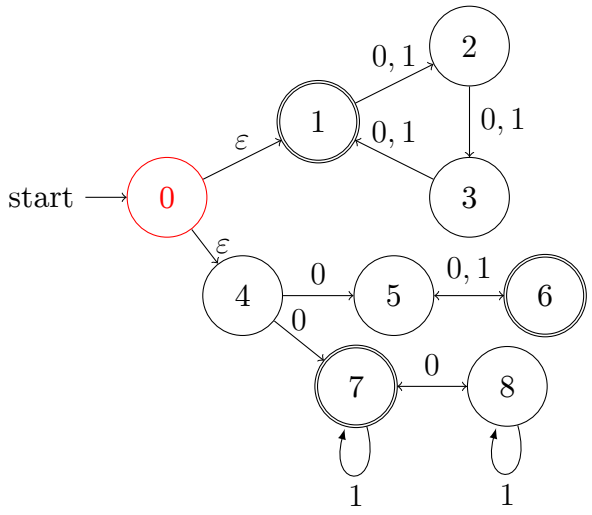


В этот автомат слово "a" можно принять перейдя из 0 в 3, а можно перейти по ε-переходу из 0 в 5, потом походить миллион раз туда-обратно по ε-переходу между 5 и 7, а потом из 5 перейти в 6, и таким образом, это слово распознаётся автоматом.

***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:

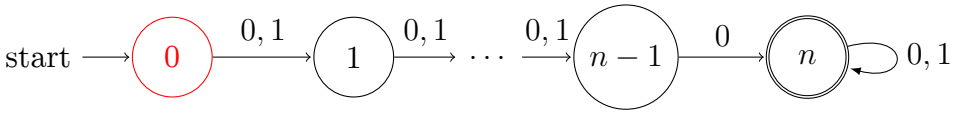
1. $\Sigma_2 = \Sigma_1$, то есть алфавит берём такой же
2. $Q_2 = 2^{Q_1}$, то есть смотрим на множество всех достижимых по данной строке состояний
3. $q_{02} = \varepsilon\text{-closure}(q_{01})$, то есть вместо стартового состояния смотрим на его ε-замыкание.
4. $T_2 = \{S \subset Q : S \cap T \neq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

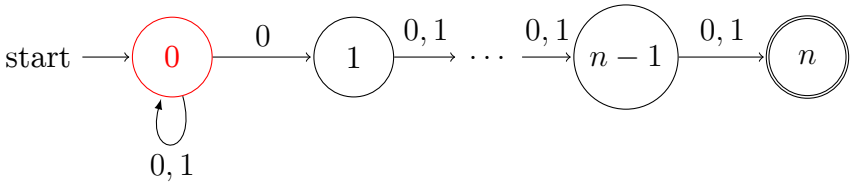
Также dfs для подсчета замыкания можно запускать только из достижимы вершин, и только когда мы в них первый раз заходим. Получается такая ленивая ДП.

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	$L(R)$
$\varnothing$	$\varnothing$
ε	$\{\varepsilon\}$
$a \quad (\forall a \in \Sigma)$	$\{a\}$
$R_1 \mid R_2$	$L(R_1) \cup L(R_2)$
$R_1 R_2$	$\{w \in \Sigma^* \mid w = xy, x \in L(R_1), y \in L(R_2)\}$
R^*	$\{w \in \Sigma^* \mid w = x_1 x_2 \dots x_n, n \in \mathbb{Z}_{0+}, \forall x_i \in L(R)\}$
(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.

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

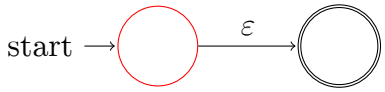
(c) из терминального состояния нет переходов

(d) начальное и терминальное состояние не совпадают

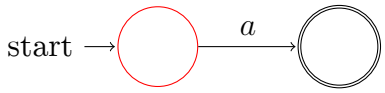
Итак, построение:

1) Автомат, задающий пустой язык:

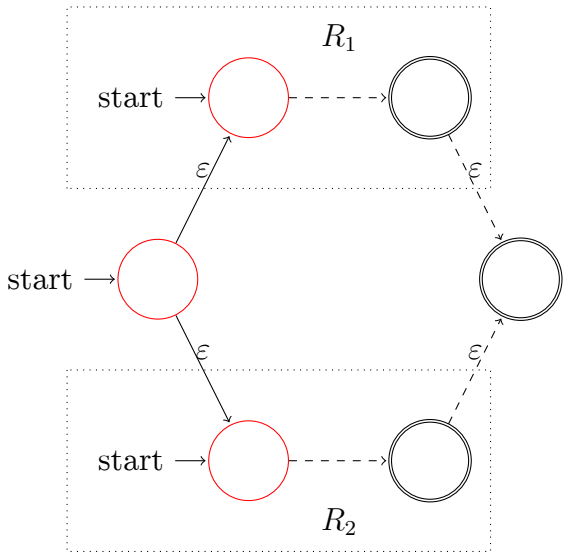
2) Автомат, задающий пустое слово:



3) Автомат, задающий одну букву a:

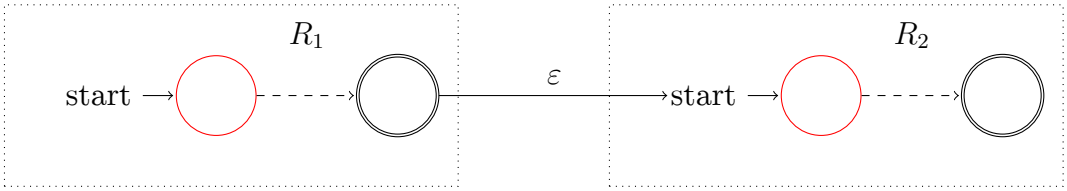


4) Автомат, задающий $R_1|R_2$ (для которых уже есть свои автоматы)



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.

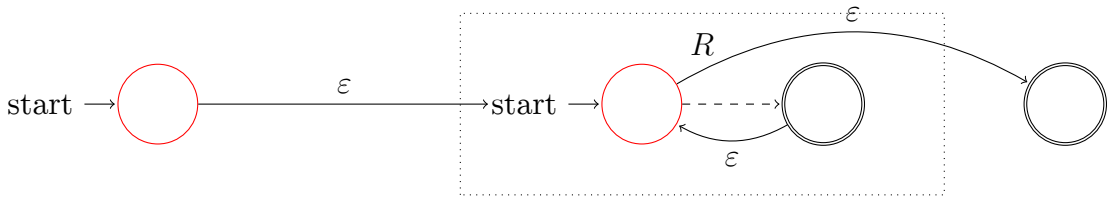
5) Автомат, принимающий $R_1 R_2$



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 $\{0, 1 \dots, n-1\}$)

Слой $k = 0$

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

□

5. Ещё про детерминированные конечные автоматы

5.1. Лемма о разрастании

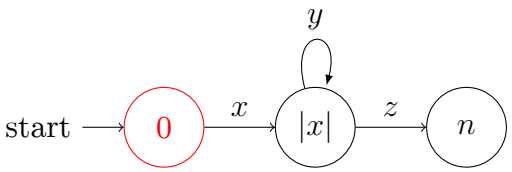
Лемма. Для любого регулярного языка L существует $n \in \mathbb{N}$, что любое слова $w \in L$ длины $\geq n$ представляется в виде xyz, где $x, y, z \in \Sigma^*$ и выполняются следующие условия:

1. $y \neq \varepsilon$
2. $|xy| \leqslant n$
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 пройдём $|Q|$ 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 зациклиться 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, $n := \max\limits_{w \in L} |w| + 1$

Пример. Язык Дика, aka $L =$ ПСП. Докажем, что он нерегулярный

Доказательство. Предположим, что он регулярный, 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 регулярный, 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.

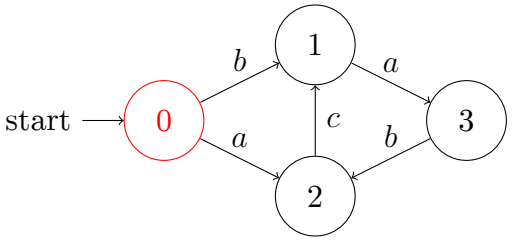
w_n – строка из n открывающих скобок. Правый контекст для w_n содержит строку из n закрывающих скобок, но при этом ни для какого $m \neq n$ правый контекст w_m эту строку не содержит по определению ПСП.

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.

$$S \to AB$$
$$S \to ABC$$
$$A \to DA$$
$$A \to Person$$
$$B \to sits$$
$$C \to atatable$$
$$D \to livefunny$$

***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 $N \to (N \cup T)^*$.

***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 $B \to \beta \in P$, то есть u получается из w применением продукции к нетерминалу B.

***Определение* 6.4.** Слово u выводится из слова w ($w \Rightarrow^* u$), если $\exists w_0, w_1, \ldots, w_n$, где $w_0 = w, w_n = u$ и $\forall 0 \le i \le n-1$ 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 $A \to \alpha, A \to \beta$ пишем $A \to \alpha|\beta$

Пример.

В грамматике $G = S \to 0S|\varepsilon$ выведем слово 000: $S \Rightarrow 0S \Rightarrow 00S \Rightarrow 000S \Rightarrow 000$. Этой грамматике соответствует язык $L(G) = 0^*$.

Грамматике $G = S \to 0S|1S|\varepsilon$ соответствует язык $L(G) = (0|1)^*$, и такая грамматика однозначная.

Грамматика $G = S \to 0S|1S|S0|S1|\varepsilon$ неоднозначная, так как строка 0 может быть получена двумя способами: $S \Rightarrow 0S \Rightarrow 0$ и $S \Rightarrow S0 \Rightarrow 0$.

Грамматике $G = S \to 0S1|\varepsilon$ соответствует язык $L(G) = 0^n1^n$

В грамматике $G = S \to \varepsilon|SS|\,(S)$ выведем строку $(())\,()$: $S \Rightarrow SS \Rightarrow (S)\,S \Rightarrow ((S))\,S \Rightarrow (())\,S \Rightarrow (())\,(S) \Rightarrow (())\,()$. Эта грамматика соответствует языку Дика.

***Определение* 6.7.** Слово u левосторонне выводится из слова w за один ход ($w \Rightarrow_{lm} u$) , если $w = \alpha B\gamma$, $u = \alpha\beta\gamma$, $B \to \beta \in P$ и $\alpha \in T^*$.

***Определение* 6.8.** Слово u левосторонне выводится из слова w ($w \Rightarrow^*_{lm} u$), если $\exists w_0, w_1, \ldots, w_n$, где $w_0 = w, w_n = u$ и $\forall 0 \le i \le n-1$ верно, что $w_i \Rightarrow_{lm} w_{i+1}$.

***Определение* 6.9.** Слово u правосторонне выводится из слова w за один ход ($w \Rightarrow_{rm} u$) , если $w = \alpha B\gamma$, $u = \alpha\beta\gamma$, $B \to \beta \in P$ и $\gamma \in T^*$.

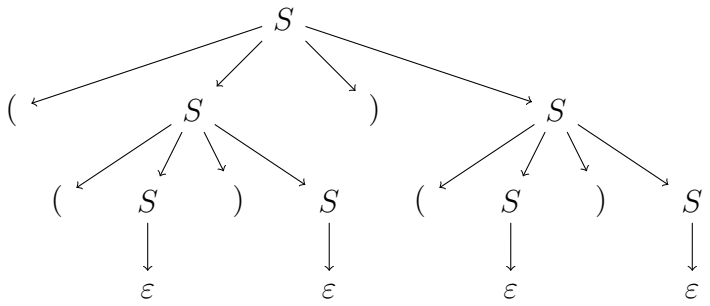
***Определение* 6.10.** Слово u правосторонне выводится из слова w ($w \Rightarrow^*_{rm} u$), если $\exists w_0, w_1, \ldots, w_n$, где $w_0 = w, w_n = u$ и $\forall 0 \le i \le n-1$ верно, что $w_i \Rightarrow_{rm} w_{i+1}$.

Для ПСП существует однозначная грамматика: $G = S \to (S)\,S|\varepsilon$

6.2. Дерево разбора

Дерево разбора описывает способ вывода слова в данной грамматике.

Пример. Дерево разбора однозначной граматики для ПСП, строка (())()



Чтобы получить строку по дереву разбора, нужно обойти листья в порядке слева направо и выписать все символы оттуда (ε при выписывания не меняет строку).

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 6.1. Регулярные $\subsetneq$ КС-языки

Доказательство.

$\neq$ 0^n1^n распознаётся КС-грамматикой и не распознаётся ДКА

$\subset$ 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 $q \to ap$, 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 $S \to \varepsilon | S'$, 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) $A \to \varepsilon$ удаляем
2) $A \to A_1 \dots A_n$, $A_i \in N \cup T$. 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 $A_1 \dots A_n$, только если все удаленные символы – ε-порождающие терминалы и эта подпоследовательность не пустая.

Пример.

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

$$\to$$

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

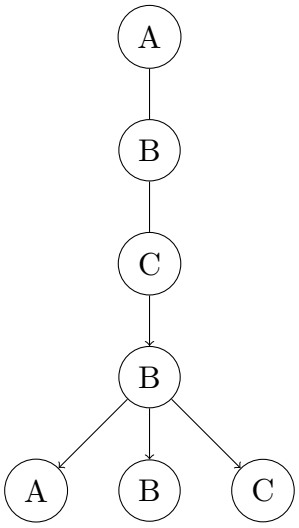
Время работы $\mathcal{O}(|G| \cdot 2^{max\ right\ part\ len})$

- **Удаляем цепные продукции**

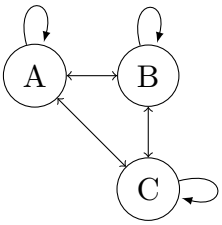
$A \to B; A, B \in N$
Построим транзитивное замыкание отношения "есть продукция $A \to B$"
Для любого ребра $A \Rightarrow^* B$ и для любой продукции $B \to \gamma$ получим продукцию $A \to \gamma$

Пример.

Рассмотрим следующие правила.



Преобразуем в



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

- **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.

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

Время работы $\mathcal{O}(|G|)$

- **Удаляем длинные правые частей**

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

- **Удаляем терминалы 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* 6.12.** G записанная в этом виде – К.С. грамматика в нормальной форме Хомского.

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

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

Замечание. Можно применить идею четырех русских.

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

1. Q – конечное множество состояний автомата
2. Σ – конечный алфавит принимаемых слов
3. Γ – конечный стековый алфавит
4. $q_0 \in Q$ – начальное состояние автомата
5. $z_0 \in \Gamma$ – символ, который в самом начале лежит на стеке
6. $T \subseteq Q$ – множество терминальных состояний
7. $\delta : Q \times \Sigma \cup \{\varepsilon\} \times \Gamma \to 2^{Q \times \Gamma^*}$ такое, что все возможные значения δ являются **конечными** множествами – функция перехода.

Замечание. По факту, функция перехода снимает один символ со стека, либо считывает одну букву, либо нет (ε-переход) и дописывает на верхушку стека строку над стековым алфавитом (возможно пустую). Для того, чтобы МП-автомат работал, нужен начальный символ на стеке aka дно, потому что если стек пустой, то функция перехода не определена.

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

***Определение* 7.3.** Из моментального описания $(xw, g\alpha, q)$ можно перейти в моментальное описание $(w, \gamma\alpha, p)$ (обозн. $(xw, g\alpha, q) \vdash (w, \gamma\alpha, p)$), если $(p, \gamma) \in \delta(q, x, g)$.

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

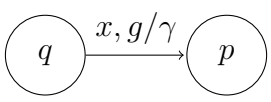
***Определение* 7.4.** Из одного моментального описания можно перейти в другое ($\vdash^*$, определяется аналогично $\Rightarrow^*$).

***Определение* 7.5.** Принципы принятия слова:

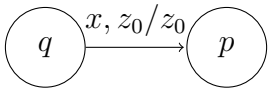
1. По терминальным состояниям, $w \in L(A) \Leftrightarrow \exists \gamma \in \Gamma^*, p \in T : (w, z_0, q_0) \vdash^* (\varepsilon, \gamma, p)$, то есть мы смогли из начального состояния перейти в терминальное состояние, считав всё слово.
2. По пустому стеку, $w \in L(A) \Leftrightarrow \exists p \in Q : (w, z_0, q_0) \vdash^* (\varepsilon, \varepsilon, p)$, то есть мы смогли из начального состояния перейти в состояние с пустым стеком, считав всё слово.

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

Обозначение. следующая картинка обозначает, что $(p, \gamma) \in \delta(q, x, g)$



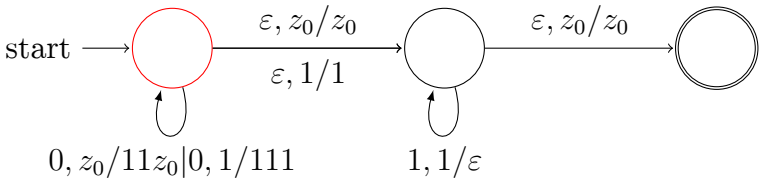
Пример. Покажем, как из НКА A сделать МП-автомат B. Для этого надо просто заменить переход по символу x на



$L(A) = L(B)$, если принимать по терминальным состояниям.

Замечание. Таким образом, регулярные языки $\subseteq$ МП-автоматные.

Пример. $L = \{0^n 1^{2n}\}$

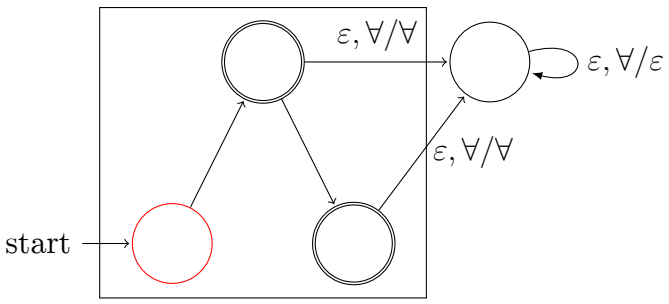


Пока считываем нолики, кладём по две единички на стек. Когда начинаем считывать единички, снимаем по одной единичке со стека.

Тем временем, язык $\{0^n 1^n 2^n\}$ не МП-автоматный. Строго доказательства здесь не будет, нестрогое такое: чтобы проверить, что единичек столько же, сколько ноликов, надо, когда мы читаем нолики, класть что-то на стек, а когда читаем единички, это со стека снимать, но проверить потом, что двоечек столько же, уже не представляется возможным, так как для них мы должны были на стек положить n символов, отличных от тех, которые были для единичек.

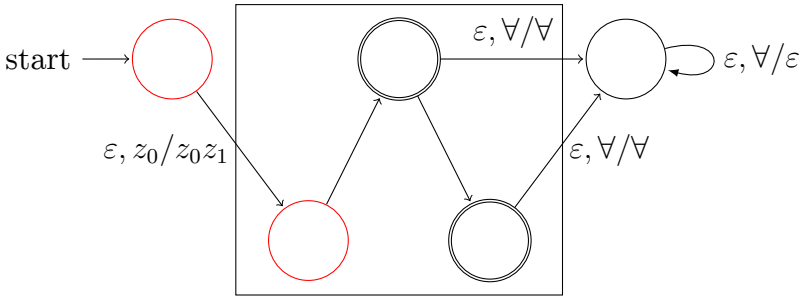
Теорема 7.1. Множество слов, принимаемое МП-автоматом по терм. состоянию совпадает с множеством, языков принимаемых МП-автоматом по пустому стеку.

Доказательство. Переделаем МПА, принимающий по терминальным состояниям в МПА, принимающий по пустому стеку. Для этого надо для терминальных состояний добавить дополнительный ε-переход в состояние, в котором мы всё со стека снимаем.

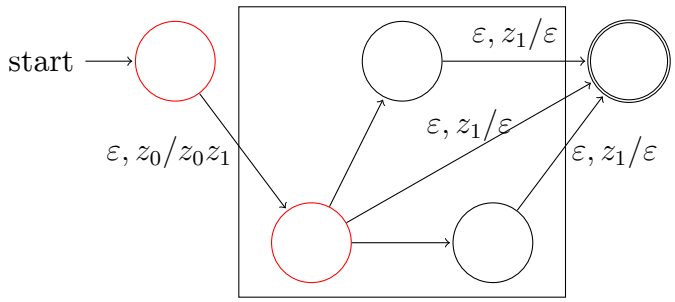


Здесь под $\forall$ понимается любой символ.

Чтобы мы случайно не приняли ничего лишнего, если стек неожиданно опустеет, добавим новую стартовую вершину, из которой будет переход в старую, добавляющий на стек **под** z_0 специальный символ z_1, как бы, второе дно. Изначально по этому символу не было переходов, значит МПА не примет ничего лишнего, и если надо перейдёт в состояние, где чистится стек.



Теперь переделаем МП, принимающий по пустому стеку, в МП, принимающий по терминальным состояниям. Делать это будем таким же образом, с помощью добавления второго дна. Также сделаем новый старт, а также из всех вершин добавим переход по z_1 в новую терминальную вершину, потому что если у нас на верхушке стека лежит z_1, то это в старом представлении означает, что наш стек опустел.



□

Замечание. Множество терминальных состояний нам теперь не нужно, мы научились обходиться без него.

Теорема 7.2. МП-автоматы и КС-грамматики распознают одинаковый класс языков

Доказательство.

МП-автомат A ← КС-грамматика G

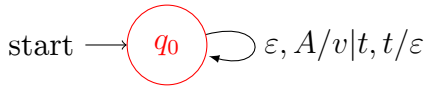
Будем рассматривать левосторонний вывод слова.

$S \Rightarrow_{lm} \cdots \Rightarrow_{lm} \underbrace{\alpha N \dots}_{} \Rightarrow_{lm} \cdots \Rightarrow_{lm} w$, где α – строка из терминалом, а N – самый левый нетерминал на очередном шаге левостороннего вывода. Назовём α словом, а $N\dots$ – стеком, который нам надо обработать.

Построим по данной КС-грамматике МПА, который принимает слова по пустому стеку. Для этого опишем его составные части

1. $\Sigma =$ терминалы
2. $\Gamma = N \cup \Sigma$
3. $z_0 = S$
4. q_0 – единственное и неповторимое состояние
5. $Q = \{q_0\}$, вообще состояние всегда может быть одно, потому что если нам нужно хранить состояние, то его можно хранить на стеке, взяв в качестве стекового алфавита $\Gamma \times Q$.
6. δ будет основываться на следующих двух правилах
   (a) Если у нас было правило $A \to v$, где A – нетерминал, а v какая-то строка из $(\Sigma \cup \Gamma)^*$, то нам надо добавить в МПА переход из q_0 в q_0, который снимает со стека символ A и кладёт туда строку v.
   (b) Также для каждого терминала t надо добавит переход из q_0 в q_0, который считывает t и снимает его со стека.

Таким образом, автомат будет выглядеть так:

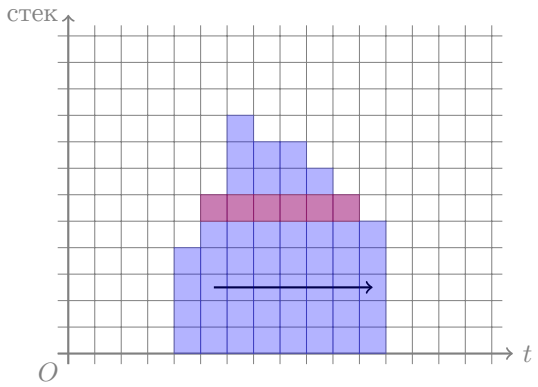


МП-автомат $A \to$ КС-грамматика G

Автомат берём по пустому стеку.

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

У каждого символа, который мы клали на стек есть первый момент, когда мы этот символ со стека сняли, то есть когда размер стека стал меньше (причем ровно на один, так как мы каждый раз снимаем по одному символу), чем был тогда, когда этот символ на стеке появился (имеется в виду такое понимание вещей).



На картинке можно наблюдать жизненный цикл символа на стеке с тех пор, как он был верхним на стеке, до момента его снятия.

$[pgq]$ – это синтаксическая категория, которая описывает все слова, которые могут снять со стека символ g, переведя автомат из состояния p в состояние q, то есть производит действие "стрелочка", если смотреть на картинку.

Возьмём любое ребро, которое считывает x и переводит g в некоторое γ. Если γ пустое, то мы добились необходимого эффекта. В противном случае, нам нужно последовательно таким же образом снять со стека все символы строки γ. Понятно, что если мы сняли γ_1, то на вершине сейчас находится γ_2, если такой есть, и т.д.

Таким образом, слова из указанной нами синтаксической категории имеют следующий вид $[pgr_k] \to x[q\gamma_1 r_1][r_1\gamma_2 r_2] \dots [r_{k-1}\gamma_k r_k]\ \forall r_1, r_2, \dots, r_k \in Q^k$, по указанным выше соображениям.

Сделаем стартовый символ S и добавим для него правило $S \to [q_0 z_0 q_0] | [q_0 z_0 q_1] | \dots | [q_0 z_0 q_{last}]$, что будет означать, что мы начинаем из состояния q_0, на стеке сейчас лежит символ z_0, и мы хотим снять его со стека, перейдя не важно в какое состояние, что идентично приёму слова по пустому стеку (z_0 сняли $\Leftrightarrow$ стек пустой).

□

Замечание. Тут нет никаких противоречий, потому что чтобы грамматика была содержательной (непустая), необходимо, чтобы была хотя бы одна продукция, не содержащая нетерминалов, а для этого у нас должны быть переходы к пустому γ, то есть те, которые ничего не пишут на стек.

Замечание. Форма всех продукций выглядит так: $A \to tN_1N_2 \dots N_k$, где N_k – нетерминалы, а $t \in \Sigma \cup \varepsilon$ (ослабленная нормальная форма Грейбах_ки). Если бы было $\leqslant 2$ нетерминалов, то это называлось бы нормальной формой Грейбах_ки.

7.1. Детерминированные конечные автоматы с магазинной памятью

***Определение* 7.6.** Все тоже самое (т.е. $(\Sigma, \Gamma, Q, T, \delta, z_o, q_o)$), но
$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to Q \times \Gamma^* \cup \{\perp\}$, причём, если определено $\delta(q, \epsilon, g)$, то ни для какого $a \in \Sigma$

не определено $\delta(q, a, g)$, и обратно, если хотя бы для какого-то $a \in \Sigma$ определено $\delta(q, a, g)$, то $\delta(q, \epsilon, g)$.

То есть из одной тройки у нас не может одновременно быть перехода, который читает символ и перехода, который его не читает.

***Определение* 7.7.** 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 $\Leftrightarrow$

$\Leftrightarrow \begin{cases} L \text{ принимается ДМП по терминальному состоянию} \\ L \text{ – префиксный} \end{cases}$

Доказательство.

$\Rightarrow$ Покажем префиксность. Если у нас есть два слова u и v из языка, распознаваемого ДМП по пустому стеку, такие, что u префикс v, то так как автомат детерминированный, то две строки, которые начинаются одинаково переходят всегда по одним и тем же состояниям с одними и теми же записями на стеке. Значит, в момент, когда мы приняли u, стек уже опустел, и значит, если v длиннее, чем u, то такое слово мы уже никак принять не можем.

Теперь покажем, что он принимается ДМП по терминальному состоянию. Сделаем то же самое, что и для недетерминированного МПА, а именно, добавим второе дно, а потом из всех состояний добавим переход, удаляющий второе дно в специальную терминальную вершину.

$\Leftarrow$ Заметим, что так как язык префиксный, то если мы пришли в терминальную вершину, то дальше идти нет смысла, поэтому удалим все переходы из терминальных вершин, а потом добавим из всех терминальных ε-переход в специальную вершину, где мы снимаем всё со стека.

Чтобы не принимать по пустому стеку лишние слова, опять же, как и в случае с недетерминированным МПА добавим второе дно, которое гарантированно нигде в этом автомате не удаляется. □

Теорема 7.4. Регулярные языки $\subsetneq$ ДМП-языки по терминальному состоянию $\subsetneq$ КС-языки

Доказательство.

1) Регулярные языки $\subsetneq$ ДМП-языки по терминальному состоянию

   $\subset$ Чтобы из ДКА сделать ДМП-автомат, просто на каждом переходе нужно доставать дно и класть его обратно.

   $\neq$ Язык $\{0^n 1^{2n}\}$ не является регулярным по лемме о накачке, однако ранее мы строили ДМПА, его распознающий.

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

   $\subset$ Мы ранее доказали, что недетерминированные МП-автоматы распознают те же языки, что и КС-грамматики. При этом каждый ДМП-автомат ялвяется при этом ещё и недетерминированным МП-автоматом, значит они распознают не больший класс языков.

   $\neq$ Рассмотрим язык $L = \{0^n 1^n\} \cup \{0^n 1^{2n}\}$. Он принимается МП-автоматом, так как мы можем построить МП-автомат для $\{0^n 1^n\}$ и для $\{0^n 1^{2n}\}$, а потом собрать из них автомат для их объединения, просто сделав новую стартовую вершину и добавив из неё два ε-перехода, не меняющих стек, в стартовые вершины исходных двух автоматов.

Теперь докажем от противного, что этот язык не принимается ДМП-автоматом. Пусть этот язык принимается ДМПА A: $L(A) = L$. Делаем полную копию A. В этой копии все единички поменяем на двоечки (буквы языка). Теперь из каждого терминального состояния старого автомата проведем переход по ϵ, не меняющий стек, в соответствующее ему состояние автомата-копии. Получился МП-автомат B.

Покажем, что $L(B) = \{0^n1^n\} \cup \{0^n1^{2n}\} \cup \{0^n1^n2^n\}$. Если мы не перешли в копию, и приняли слово, то это значит, что мы приняли 0^n1^n или 0^n1^2n. Если же мы перешли в копию, то ε-переход, который мы сделали, чтобы туда перейти мы сделали из терминальной вершины. Это означает, что перед переходом мы уже распознали префикс слова 0^n1^n или 0^n1^{2n}.

Когда мы перешли в копию, мы там походили-походили и пришли в терминальное состояние, распознав какое-то слово. Заметим, что если бы мы не переходили в копию, и по модулю этого проделали ровно такой же путь в исходном автомате, то мы бы тоже распознали какое-то слово. Это значит, что мы бы, в итоге, приняли либо слово 0^n1^n, либо 0^n1^{2n}. Но перед тем, как проделать этот остаток пути, мы уже распознали, либо 0^n1^n, либо 0^n1^{2n}, значит за остаток пути мы либо ничего не считали, либо считали n единичек (только в том случае, если до этого распознали 0^n1^n). Это означает, что если мы перешли в копию, то мы там могли либо ничего не считать, либо считать n двоечек (опять же только в том случае, если до этого распознали 0^n1^n). Значит такой автомат действительно распознаёт язык $\{0^n1^n\} \cup \{0^n1^{2n}\} \cup \{0^n1^n2^n\}$.

Позже мы докажем, что этот язык не КС-свободный. Мы знаем, что МП-автоматы эквиваленты КС-грамматикам, значит мы получили противоречие, и исходный язык был не ДМП-автоматный.

□

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

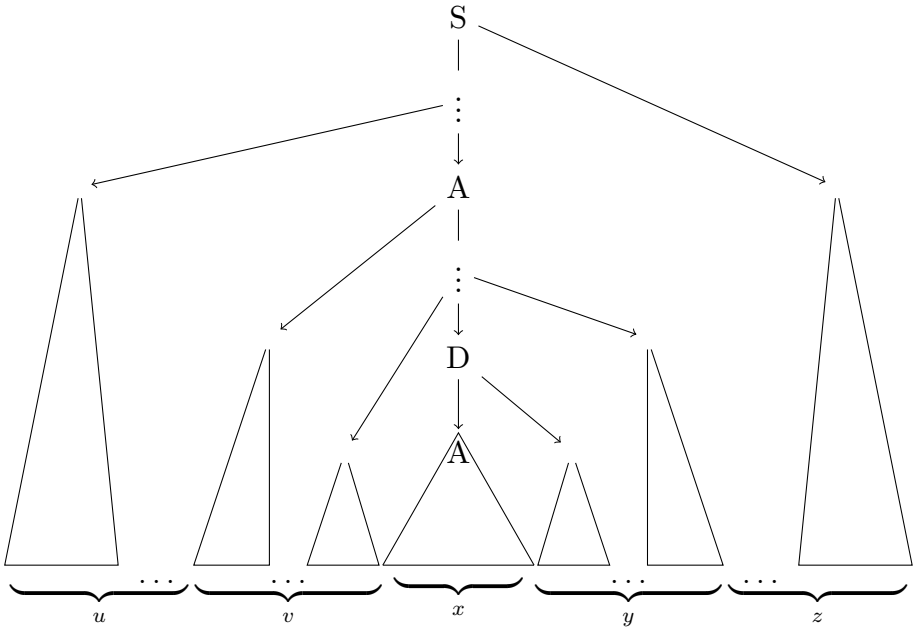
$$\begin{cases} w = uvxyz \\ vy \neq \epsilon \\ |vxy| \leq n \\ \forall i \in \mathbb{N}_0 uv^ixy^iz \in L \end{cases}$$

Доказательство.

Перейдем в нормальную форму Хомского. Положим $n = 2^{|N|}$. Пусть $|w| \geq n$. Возьмем самый длинный путь в дереве разбора этого слова. Его длинна будет хотя бы $|N| + 1$. По принципу Дирихле на этом пути есть хотя бы два одинаковых нетерминала. Возьмем самую нижнюю такую пару.

Так как мы взяли самый длинный путь, и длина этого пути от верхнего из двух одинаковых нетерминалов до листа не превосходит $|N|$ (потому что на этом пути повторяется всего один нетерминал, ведь мы взяли самый нижний повтор), то верхний из этих двух терминалов породит строку длиной не более, чем $2^{|N|}$ (дерево разбора двоичное, а $|N|$ – глубина поддерева, поэтому количество листьев в поддереве не больше, чем в $2^{|N|}$).

На картинке обозначенно, что мы берём, соответственно, за u, v, x, y и z. Так как у нас нормальная форма Хомского, то либо v, либо y будет непустой. Почему $|vxy| \leqslant n$ объяснено выше. Последовательность продукций, которая используется между A и A, может быть проигнорирована (если сразу вставить на место этого A сразу нижнее поддерево), а может быть повторена сколько угодно раз, это будет соответствовать тому, что мы либо убрали u и y, либо повторили их одинаковое количество раз. Таким образом, получили то, что и было обещано в условии.



□

Пример. $\{0^n1^n\} \cup \{0^n1^{2n}\} \cup \{0^n1^n2^n\}$ не является КС.

От противного. Пусть он КС, тогда применим лемму о накачке. По Лемме существует n с указанными свойствам, тогда возьмём это n и слово $0^{2n}1^{2n}2^{2n}$. Это слово лежит в языке, и имеет достаточную длину, поэтому его можно накачать по лемме. Поймём, что vy не содержит по крайней мере одну из трёх цифр 0, 1 и 2. Значит при накачке мы получим слово, в котором есть все три цифры, но при этом количество какой-то не совпадает с количеством какой-то другой, а значит мы получим слово не из языка. Противоречие.

7.2. Иерархия Хомского

0) Произвольные грамматики

1) Контекстно-зависимые

2) Контекстно-свободные

3) Регулярные

***Определение* 7.8.**

Праволинейной грамматикой назовем такую грамматику, что $A \to a$ и $A \to aB$.

Леволинейной грамматикой назовем такую грамматику, что $A \to a$ и $A \to Ba$.

Тогда регулярные грамматики это леволинейные $\cap$ праволинейные.

Замечание. Есть прямое соответствие между ДКА и линейными грамматиками. Т.е. то, что задается леволинейными, задается и ДКА, то, что задается ДКА, задатеся и праволинейными и наоборот.

***Определение* 7.9.**

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* 7.10.**

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

Теорема 7.5. Неукорачивающие $\Leftrightarrow$ контекстно-зависимые

Доказательство.

$\Leftarrow$ Из определения контекстно-зависимых следует, что они неукорачивающие.

$\Rightarrow$ Будем по неукорачивающей строить эквивалентную ей контекстно-зависимую. Для каждого терминала a заведём уникальный нетерминал A и продукцию $A \to a$.

Теперь для каждой продукции $a_1 a_2 \dots a_n \to b_1 b_2 \dots b_m$ ($m \geqslant n$, $a_i, b_j \in (\Sigma \cup N)$, среди a_i есть хотя бы один нетерминал, и если a_i нетерминал, то мы его ни на что не меняем, иначе меняем на соответствующий ему A_i, то же самое для b_j), которая была в неукорачивающей грамматике, заведём уникальные нетерминалы (специально для этой продукции) $Z_1, Z_2, \dots, Z_n$. И добавим следующие продукции.

$$\begin{cases} A_1 A_2 \dots A_n \to Z_1 A_2 \dots A_n \\ Z_1 A_2 \dots A_n \to Z_1 Z_2 \dots A_n \\ \dots \\ Z_1 Z_2 \dots Z_{n-1} A_n \to Z_1 Z_2 \dots Z_{n-1} Z_n \\ Z_1 Z_2 \dots Z_n \to B_1 Z_2 \dots Z_n \\ \dots \\ B_1 B_2 \dots B_{n-1} Z_n \to B_1 B_2 \dots B_{n-1} B_n B_{n+1} \dots B_m \end{cases}$$

После применения первой продукции, единственный способ избавиться от всех Z_i в силу его уникальности – это применить все остальные продукции, то есть перевести $A_1 \dots A_n$ в $B_1 \dots B_m$. □

Теорема 7.6.

Любую контекстно-зависимую (неукорачивающую) грамматику можно привести к Н.Ф. Куроды:
$AB \to CD$
$A \to BC$
$A \to a$

Доказательство.

1. Исключим из правых частей терминалы. Для каждого терминала a заведём уникальный нетерминал A и продукцию $A \to a$.

2. Остались только продукции вида $\alpha \to \beta$, где $\alpha, \beta \in N^+$ и $|\alpha| \leqslant |\beta|$.

   (a) $|\beta| \leqslant 2$, тогда это уже нормальная форма Куроды

   (b) $|\beta| \geqslant 3, |\alpha| = 1$, тогда продукция имеет вид $A \to B_1 \dots B_n$. Создадим для этой продукции уникальные нетерминалы A_i, и такие продукции:

$$\begin{cases} A \to B_1 A_1 \\ A_1 \to B_2 A_2 \\ \dots \\ A_{n-2} \to B_{n-1} B_n \end{cases}$$

   Тогда в силу уникальности A_i, избавиться от них можно только применив все продукции, чтобы получить $A \to B_1 \dots B_n$

(c) $|\beta| \geqslant 3, |\alpha| \geqslant 2$, тогда продукция имеет вид $A_1 \dots A_n \to B_1 \dots B_m$.

Сделаем уникальный нетерминал C и две такие продукции:

$$\begin{cases} A_1 A_2 \to B_1 C \\ C A_3 \dots A_n \to B_2 \dots B_m \end{cases}$$

Со второй продукцией разберемся с помощью индукции по размеру продукции с помощью одного из трёх правил, в зависимости от длины левой правой части. Можем себе позволить, так как длина и левой и правой части уменьшилась на 1, это хорошо, потому что и размер продукции уменьшился и неравенство $|\alpha| \leqslant |\beta|$.

□

***Определение* 7.11.**

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 7.7.

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.

□