# Теория формальных языков

Тух Игорь и Ютман Михаил
по лекциям Михаила Эдуардовича Дворкина

9 апреля 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 $\varepsilon$-переходами** **5**
3.1 Недетерминированные конечные автоматы с $\varepsilon$-переходами . . . 5
3.2 Детерминизация $\varepsilon$-НКА . . . 6
3.3 Произведение конечных автоматов . . . 6

**4. Регулярные выражения** **7**
4.1 Академические регулярные выражения . . . 7
4.2 Лемма о разрастании . . . 8
4.3 Динамическое программирование по ДКА . . . 9

**5. Формальные грамматики** **10**
5.1 Контекстно-свободные грамматики . . . 10
5.2 Дерево разбора . . . 11
5.3 Преобразования КС-грамматик . . . 11
5.4 Алгоритм Кока-Янгера-Касами (CYK) . . . 13

# 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) $\Sigma$ и скажем, что слово $w_1 \dots w_n$ – это конечная последовательность символов из алфавита. Язык $L$ – это множество слов (над алфавитом $\Sigma$).

*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.** $\Sigma^*$ – это множество всех слов над $\Sigma$. $\varepsilon$ – 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)

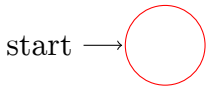
## 2.1. Детерминированные Конечные Автоматы

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

1. $\Sigma$ – 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.***

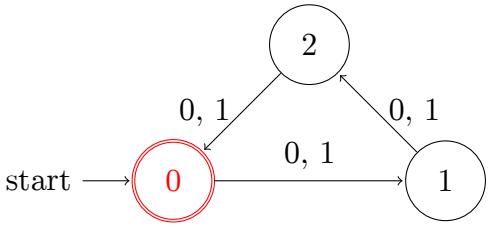
Далее в конспекте так будет обозначаться стартовые состояния:




А вот так терминальные:

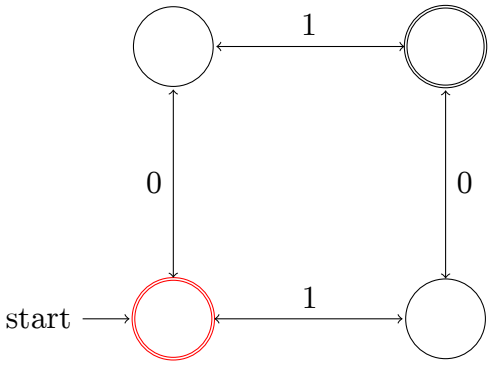
**Примеры.**

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

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

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

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

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

**Доказательство**. Рефлексивность, симметричность и транзитивность очевидны. □

**Алгоритм** (Минимизации ДКА)**.**

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

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

## 2.3. Правые контексты

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

Аналогично можно определить левые контексты.

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

*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 $\varepsilon$-переходами

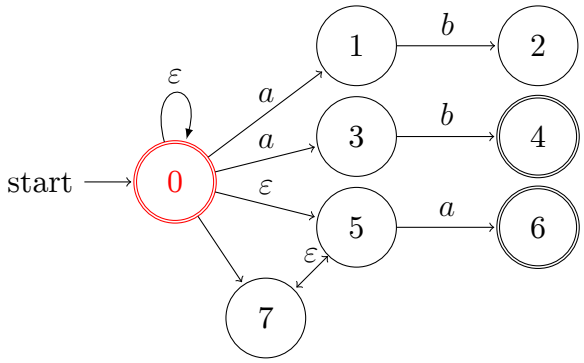
## 3.1. Недетерминированные конечные автоматы с $\varepsilon$-переходами

***Определение* 3.1.** Недетерминированный конечный автомат (НКА) – это пятёрка ($\Sigma$, $Q$, $q_0$, $T$, $\delta$), где $Q$ – конечное, $q_0 \in Q, T \subseteq Q, \delta : Q \times (\{\varepsilon\} \cup \Sigma) \to 2^Q$, которая принимает строку $w$ и выдаёт "Да", если есть хотя бы один путь по строке $w$ из начальной вершины в терминальную.

***Определение* 3.2.** $\varepsilon$-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 $\varepsilon$-переходу из 0 в 5, потом походить миллион раз туда-обратно по $\varepsilon$-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, $L(A) = \{w \in \Sigma^* : \exists$ путь из $q_0$ в $q \in T$, на ребрах которого написано слово $w\}$.

**Пример.** **TODO:** картинка

***Определение* 3.4.** $\varepsilon$-замыкание вершины $q$ edge это множество состояний, достижимых из $q$ по $\varepsilon$-переходам (можно закодить dfs-ом).

Каждый раз добавляем вместо вершины её $\varepsilon$-замыкание. Берём $\varepsilon$-замыкания и их битовые маски or-им. Мы договорились, что у нас размер алфавита константа, поэтому для каждого состояния мы могли предподсчитать битовую маску. Поэтому, если мы всё предподсчитали, то 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).

Если мы посмотрим на все возмножные замыкание, получится, что есть $2^{|}Q|$ различных. Таким образом, если программа хранит всегда конечное число информации и читает слово слева направо, то она является детерминированным конечным автоматом. Получлось, что программа, которая осуществляет переходы по недетерминированному конечному автомату является детерминированным конечным автоматом.

## 3.2. Детерминизация $\varepsilon$-НКА

Задача сделать из $\varepsilon$-НКА $A_1$ ДКА $A_2$, такой что $L(A_2) = L(A_1)$.

$A_2 = (\Sigma, 2^{Q_{A_1}}, \varepsilon\text{-closure}(q_{0A_1}), s : s \cap T \neq \varnothing, \delta(S, a) = \bigcup\limits_{p \in S, p \to q} \varepsilon\text{-closure}(q))$.

**Пример.** $L_1 = \{w : w_n = 0\}$

**TODO:** картинка

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

**TODO:** картинка

В ДКА для того же языка $|Q| \geq 2^n$ (оцените разницу по сравнению с $n+1$ цифрой в $\varepsilon$-НКА).

В НКА каждый переход в худшем случае за $|Q|^2$, а в ДКА – это просто переход в прямоугольном массиве.

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

Пусть есть два автомата: $A$ и $B$, тогда $A \times B = (\Sigma, Q_A \times Q_B, (q_{0_A}, q_{0_B}), \ldots, \delta((p, q), a) = (\delta_A(p, a), \delta_B(q, a)))$.

Если $T = T_A \times T_B$, то это $L_1 \cap L_2$, если же $T = T_A \times Q_B \cup Q_A \times T_B$, то это $L_1 \cup L_2$.

# 4. Регулярные выражения

## 4.1. Академические регулярные выражения

**TODO:** рефакторниг лекции

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

**TODO:** table

Введем рекурсивное определение.
Регулярное выражение $R$ – $L(R)$
$\varnothing$ – $\varnothing$
$\epsilon$ – $\{\epsilon\}$
$\forall a \in \Sigma$ – $\{a\}$
Далее, операции в порядке возрастания приоритета.
$R_1|R_2$ – $L(R_1) \cup L(R_2)$
$R_1R_2$ – $\{w \in \Sigma^* \mid w = xy, x \in L(R_1), y \in L(R_2)\}$ конкатенация языков
$R^*$ – $\{w \in \Sigma^* \mid w = x_1x_2 \dots x_n, n \in \mathbb{Z}_{0+}, \forall x_i \in L(R)\}$ замыкание Клини
$(R)$ – $L(R)$

**Примеры.**

1. $(0|1)^*$ – язык, состоящий из всех слов над алфавитом $\{0, 1\}$
2. $(0|1)^*001^*$ – 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. $RR(\epsilon|R)(\epsilon|R) = R\{2-4\}$
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.
   $\forall R \to \epsilon$–НКА $A: \; L(A) = L(R)$.
   Далее, аналог индукции:
   **TODO:** навести красоту

   **Условия**:

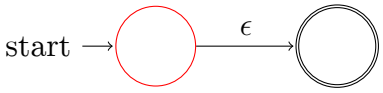
   (a) ровно одно терминальное состояние

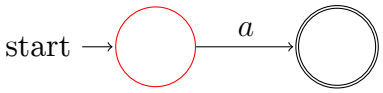
   (b) в начальное состоянии нет переходов

   (c) из терминального состояния нет переходов

(d) начальное и терминальное состояние не совпадают

Пустой язык: 


Пустое слово: 


Одна буква $a$: 


**TODO:** красивенькие простые картинки для $R_1|R_2, R_1R_2, R^*$

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) $i = j$ $R_{i,i,0} = \epsilon|a_1|a_2|\dots|a_k$, 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**
$R = R_{q_0,t_1,n}|R_{q_0,t_2,n}|\dots|R_{q_0,t_{|T|},n}$ или $\varnothing$, если терминальных состояний нет.
Тогда $L(R) = L(A)$.
Оценим время работы: $\mathcal{O}^*(4^n)$. Размер ответа такой же.

□

регулярные = автоматные

## 4.2. Лемма о разрастании

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

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

**TODO:** картинка автомата

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

□

**Пример.** $L =$ ПСП. Докажем, что он нерегулярный

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

## 4.3. Динамическое программирование по ДКА

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

**TODO:** картинка графа

**Пример.** Количество слов длины $l$ в $L$.

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

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

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

Ответ – это сумма элементов столбца, соответствующих терминальным вершинам.

# 5. Формальные грамматики

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

## 5.1. Контекстно-свободные грамматики

**Пример.**

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

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

$T$ – терминальные символы (алфавит)

$N$ – нетерминальные символы (синтаксические категории)

$S \in N$ – стартовый символ

$P$ – множество продукций

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

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

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

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

***Обозначение.***

стартовый символ – $S$

нетерминалы – заглавные латинские буквы

терминалы – строчные или цифры

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

**Пример.**

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

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

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

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

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

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

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

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

## 5.2. Дерево разбора

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

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

**Теорема 5.1.** Регулярные $\subsetneq$ КС-языки

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

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

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

□

## 5.3. Преобразования КС-грамматик

- **Меняем стартовый символ**

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

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

- **Избавление от $\varepsilon$-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 $\varepsilon$. 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 – $\varepsilon$-порождающие терминалы и эта подпоследовательность не пустая.

**Пример.**

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

$\to$

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

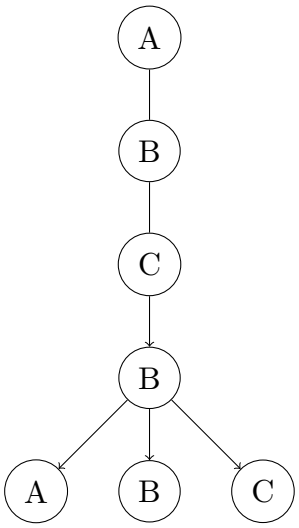
**Время работы** $\mathcal{O}(|G| \cdot 2^{maxrightparrtlen})$

- **Удаление цепных продукций**

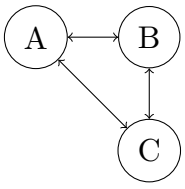
$A \to B; A, B \in N$
Построим транзитивное замыкание отношения "есть продукция $A \to B$"
Для любого ребра $A \Rightarrow^* B$ и для любой продукции $B \to \gamma$ получим продукцию $A \to \gamma$

**Пример.**

Рассмотрим следующие правила.




Преобразуем в




**TODO:** петли во втором рисунке

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

- **Удаление бесполезных нетерминалов**

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

2) Оставляем только достижимые из $S$ нетерминалы.

**Пример.**

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

**Время работы** $\mathcal{O}(|G|)$

- **Удаление длинных правых частей**

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

- **Удаление терминалов в правых частях**

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

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

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

## 5.4. Алгоритм Кока-Янгера-Касами (CYK)

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

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

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

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

Время работы $\mathcal{O}(|w|^3 \cdot |G|)$. Можно еще применить четырех русских **TODO:** (?)