РАДОСЛАВ ПАВЛОВ

# МАТЕМАТИЧЕСКА ЛИНГВИСТИКА

НАРОДНА ПРОСВЕТА

РАДОСЛАВ ПАВЛОВ

# МАТЕМАТИЧЕСКА ЛИНГВИСТИКА

## Увод в теорията на формалните езици и граматики



ДЪРЖАВНО ИЗДАТЕЛСТВО
„НАРОДНА ПРОСВЕТА"
СОФИЯ, 1982 год.

Основната задача на предлаганата книга е да охарактеризира обекта и методите на математическата лингвистика, нейното отношение към изучаването на езика и мястото ѝ в математиката.

Книгата включва изложение на основните понятия на математическата лингвистика: разглежда най-важната и най-разработената нейна част — теорията на формалните граматики и езици, връзките ѝ с информатиката и нейните математически основи, с теорията на автоматите, с теорията на езиците за програмиране. Разглеждат се и някои приложения в изучаването на естествените езици.

Книгата е насочена както към читатели—ученици и учители, с по-специални интереси към съвременната математика и към точните методи в лингвистиката, така и към широк кръг читатели, за които тя може да служи като въведение в една нова и развиваща се математическа дисциплина.

51





## ПРЕДГОВОР

Компютрите стремително навлизат във всички области на човешката дейност. С изключителни темпове нараства тяхната мощ, многостранност и разнообразие на видовете. Както сочи едно станало вече класическо сравнение, ако автомобилите се развиваха с темповете на усъвършенстване на компютрите, то един автомобил „Ролс-Ройс“ би трябвало сега да струва 20 долара, моторът му да е голям колкото глава на кибритена клечка, скоростта му да е 100 000 km/h и да изразходва 3 литра бензин на 1 000 000 km.

Какво обаче определя възможностите на компютрите, къде са границите на тези възможности, какво е общото в тяхното многообразие по видове и предназначение?

Отговор на тези и редица още аналогични въпроси можем да получим, изследвайки две фундаментални математически абстракции, свързани с компютрите: абстрактните информационни машини, наречени автомати, и абстрактните, формални езици.

Формален език се нарича всяко множество от думи (или още — низове, изкречения), построени по дадени правила. Думите от своя страна представляват крайни редици от символи, взети от дадено множество символи, наречено азбука или речник на езика. Теорията на формалните езици изучава задаването, разпознаването и преобразуването на формалните езици. Тя е тясно свързана с лингвистиката и с теорията на езиците за програмиране. Една от основните ѝ задачи е да разработва и изучава математически модели на свойствата на естествените езици, на езиците за общуване с компютрите и на различни други изкуствени езици.

Автоматите представляват устройства, които преработват дадени входни данни в изходни данни чрез управляващо устройство с крайна памет. Някои видове автомати могат да имат и външна памет. Автоматите могат да се разглеждат като математически модели не само на реалните компютри, но дори и на нервната система на човека.

Тази книга може да се разглежда като увод в математическата лингвистика, в който се изучават основните видове автомати и формални езици, начините им на задаване, техните свойства, връзките помежду им, както и отделни техни приложения в лингвистиката и в теорията на езиците за програмиране.

В глава I се въвежда понятието формален език. Разглеждат се двата основни метода за „крайно описание“ на формалните езици, които по-нататък ще бъдат обект на внимание: задаването на формалните езици чрез пораждащи устройства — граматики, и разпознаването им чрез автомати — разпознаватели. Въвеж-

да се йерархия на формалните езици в съответствие с вида на пораждащите ги граматики, известна като йерархия на Н. Чомски.

В следващите четири глави се изучават основните видове пораждащи граматики и формални езици: автоматни, безконтекстни, контекстни и от общ вид. Успоредно с това се въвеждат съответните им разпознаватели — автомати: крайни автомати, магазинни автомати, линейно ограничени автомати и машини на Тюринг. Показани са приложенията на автоматните езици и крайните автомати в лексическия анализ на езиците за програмиране; на безконтекстните граматики и на магазинните автомати при задаването на синтаксиса на езиците за програмиране, в синтактичния им анализ и в процеса на тяхната транслация.

Значително внимание е отделено на алгоритмичните проблеми, свързани с автоматните, безконтекстните, контекстните и рекурсивно-номерируемите езици, които в значителна степен определят границите на тяхната приложимост.

В последната глава са скицирани някои приложения на теорията на формалните езици и граматики в изучаването на естествените езици Тъй като безконтекстните граматики приписват на всяка дума или изречение от езика съответна синтактична структура и пораждат рекурсивни езици, те представляват удобен прост модел на синтаксиса на подмножества на естествените езици, който може с успех да се прилага в компютърния им анализ. Въведен е и още един математически модел на някои свойства на естествените езици — така наречените трансформационни граматики.

В края на всеки параграф има упражнения, които са предназначени за самопроверка, но често пъти съдържат и допълнителна информация.

При четенето на книгата не се изискват специални знания по математика и лингвистика. Всички необходими понятия се определят и разясняват в изложението. Въпреки това, доброто усвояване на включения материал предполага склонност към математически начин на мислене и известна математическа и лингвистична грамотност.

*Януари, 1982 г.* *Авторът*


*„Истината е, че преди да измислят сегашната азбука, която днешните наши безценни съкровища така лесно научават (когато станат достатъчно големи), човеците са имали най-различни начини на писане, като например Йероглифен, Демотически, Нилотски, Криптически, Куфически, Рунически, Дорически, Йонически и много още „ически-драскулически“.“*

Ръдиърд Киплинг — *„Приказки“*

# Глава 1

# ФОРМАЛНИ ЕЗИЦИ

## 1.1. АЗБУКИ. ДУМИ ВЪРХУ АЗБУКИ

Откъде да започнем изучаването на формалните езици? Може би от известната на всички нас дума „език“? Освен българския език, често употребяваме и други езици — английски, руски, френски, немски и т. н., а в кръга на нашите специални интереси използуваме и различни изкуствени езици: езика, регулиращ уличното движение, езика на правото, на математическата логика и много други. Това нарастващо многообразие от езици често пъти даже ни затруднява, когато искаме да определим дали някакъв графичен образ представлява картина или текст от изкуствен научен език. Изясняването на структурата на някои текстове, на тяхната правилност и смисъл изискват понякога задълбочени познания и значителен опит.

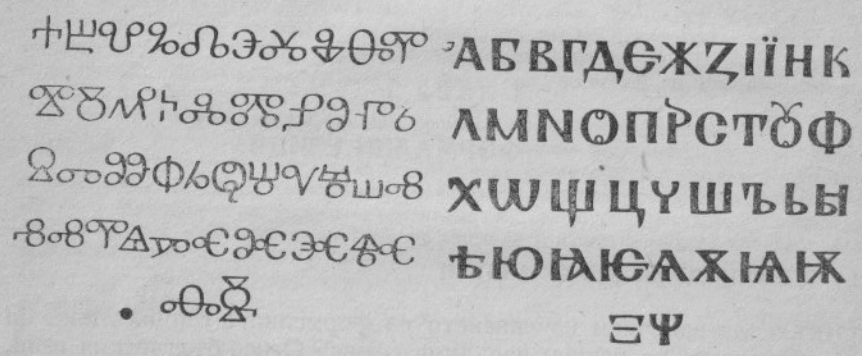
И все пак във всички езици, като правило, се забелязва една обща конструкция, която особено ясно личи при изкуствено създаваните езици. Всеки език използува някаква съвкупност от символи, които образуват в едни случаи — неговата азбука, а в други — неговия речник. От тези основни единици по определени правила (граматиката на езика) се строят елементите на самия език — думи, изречения и пр.

Изучаването на формалните езици ще започнем с изясняването на природата и ролята на тези три съставящи — азбуки, думи, граматики.

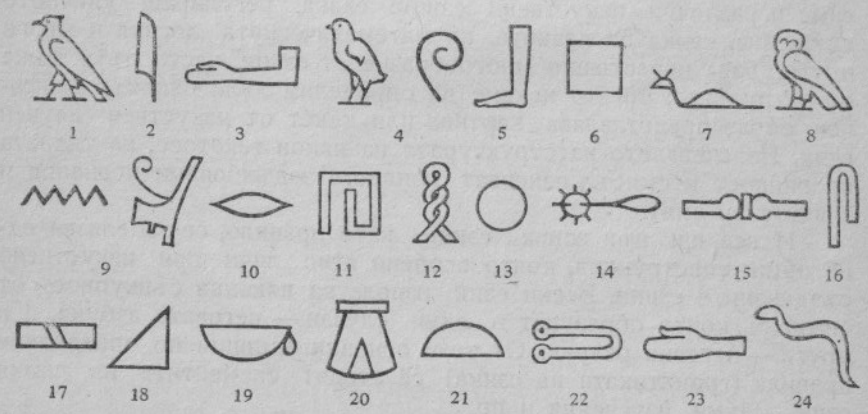
Преди всичко ще започнем с понятието азбука и с понятието дума, записана чрез буквите на дадена азбука.

Всеки от нас би могъл да даде примери за различни азбуки, да напише техните букви, да състави думи от тях и т. н. Като пример за азбука можем веднага да посочим глаголицата и кирилицата, съставени от Константин Кирил философ и брат му Методий през IX в. за нуждите на славянските езици (фиг. 1):

Друг пример представляват египетските йероглифи, които са били азбуката на древните египтяни (фиг. 2):

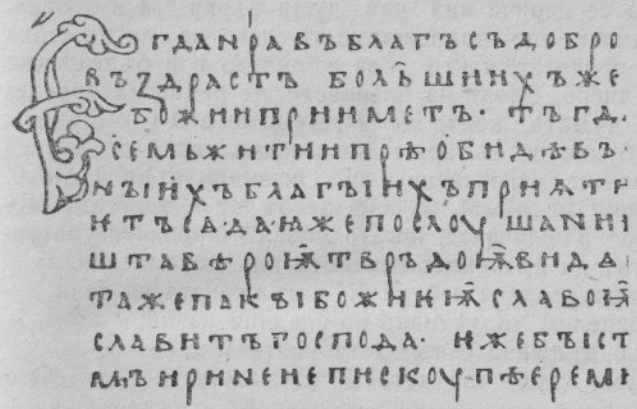


Фиг. 1 а) Глаголица б) Кирилица




Фиг. 2. Египетски фонетични йероглифи

От буквите на азбуките могат да се съставят различни думи, а от тях — текстове с една или друга големина (фиг. 3):



Фиг. 3. Из „Супрасълски сборник"

Азбуки има много, съставящите ги букви са най-различни, чрез тях са написани милиони и милиони думи. На нас обаче ще ни е необходимо едно общо понятие за азбука, едно общо понятие за буква, едно общо понятие за дума. При това тези понятия трябва да бъдат математически.

И така, какво е азбука? Всяко крайно множество от символи ще наричаме **азбука**. От гледна точка на еднозначното им прочитане е необходимо символите от дадена азбука да бъдат лесно различавани един от друг както като отделни символи, така и когато са написани един до друг. Върху тяхната природа не се налагат никакви ограничения. Символите могат да бъдат графични обекти, степента на намагнетизираност на магнитната лента, наличието или отсъствието на някакво състояние и други. Символите от дадена азбука често пъти ще наричаме още нейни **букви.**

Примери:

а) $V_1 = \{A, Б, B, \ldots, Ю, Я, а, б, в, \ldots ю, я, ., ,, :, ?, !, \ldots, —, ;, ,", „, (,)\}$;

б) $V_2 = \{0, 1\}$;

в) $V_3 = \{$**гонят, някои, всички, котки, кучета**$\}$.

Към азбуките можем да прилагаме различни операции, определени върху множества, например обединение, сечение, разлика

и др. В този смисъл можем да говорим за обединението на две азбуки, за тяхното сечение или разлика.

Нека е дадена произволна азбука. Всяка крайна редица от букви на азбуката се нарича **низ** или **дума** върху тази азбука. Членовете на редицата ще подреждаме от ляво на дясно, така че думите ще се прочитат, както това е прието и в българския език, от ляво на дясно. Броят на членовете на редицата определя **дължината на думата**, която ще означаваме с $\partial(\ )$. Думата, чиято редица от букви не съдържа нито един член, се нарича **празна**. Тази дума ще означаваме с $\Lambda$. Очевидно е, че $\partial(\Lambda)=0$.

Термините „низ“ и „дума“ за нас ще бъдат синоними. Ще предпочитаме „низ“ в случаите, когато поради широката употреба на „дума“ могат да възникнат неясноти или двусмислия.

Дума (низ) върху дадена азбука $W$ и дължината на тази дума можем да определим индуктивно по следния начин:

1) $\Lambda$ е дума с дължина 0 върху азбуката $W$;

2) ако $\alpha$ е дума върху азбуката $W$ с дължина $n$ и $a$ е буква от $W$, тогава $\alpha a$ е дума върху азбуката $W$ с дължина $n+1$;

3) $\alpha$ е дума върху азбуката $W$, само когато е получена чрез прилагане на 1) и 2) краен брой пъти.

Ще отбележим, че празната дума $\Lambda$ е дума върху всяка азбука.

Ето няколко примери на думи върху дадените по-горе азбуки $V_1$, $V_2$, $V_3$:

а) *аАААа*, *адалиетака*, *око* са думи с дължина съответно 5, 10, 3 върху азбуката $V_1$;

б) 1111111, 0, 10 са думи с дължина съответно 7, 1 и 2 върху азбуката $V_2$;

в) **някои котки гонят кучета** е дума с дължина 4 върху азбуката $V_3$.

Думите обикновено ще отбелязваме с малки гръцки букви $\alpha$, $\beta$, $\omega$, ... Често пъти ще използуваме и различни индекси.

Две думи са **равни**, ако имат една и съща дължина и еднакви първи букви, еднакви втори букви и т. н., еднакви последни букви. Например думите 0101 и 101 са различни, макар че са двоичен запис на едно и също число.

Могат да се определят различни операции върху думи.

**Обръщане** на една дума ще наричаме прочитането на дадената дума от дясно на ляво. Да означим тази операция с $O(\ )$ и да вземем думата $\alpha=a_1a_2\ldots a_n$. Тогава

$$O(\alpha)=a_na_{n-1}\ldots a_2a_1.$$



Очевидно е, че ако $\alpha$ е дума върху азбуката $W$, то $O(\alpha)$ също ще бъде дума върху $W$ и $\partial(\alpha)=\partial(O(\alpha))$.

**Конкатенация (съединяване)** на две дадени думи $\alpha$ и $\beta$ ще наричаме непосредственото записване на думата $\beta$ след думата $\alpha$ и ще означаваме с $\alpha\beta$. Ако

$$\alpha=a_1a_2\ldots a_n \text{ и } \beta=b_1b_2\ldots b_k,$$

тогава

$$\alpha\beta=a_1a_2\ldots a_nb_1b_2\ldots b_k.$$

Чрез конкатенация на думите *аз* и *бука* ще получим думата *азбука*. Ясно е, че ако $\alpha$ е дума върху азбуката $W$, а $\beta$ е дума върху азбуката $V$, $\alpha\beta$ ще бъде дума върху обединението на $W$ и $V$. Освен това

$$\partial(\alpha\beta)=\partial(\alpha)+\partial(\beta).$$

Нека са дадени три произволни думи $\alpha$, $\beta$, $\gamma$. При конкатенацията на $\alpha$ с $\beta\gamma$ се получава същата дума, както при конкатенацията на $\alpha\beta$ и $\gamma$. Това означава, че конкатенацията е асоциативна операция. Тя обаче не е комутативна операция, тъй като не е трудно да се намерят примери, когато $\alpha\beta$ не съвпада с $\beta\alpha$. Освен това лесно може да се забележи, че каквато и дума $\alpha$ да вземем, $\Lambda\alpha=\alpha\Lambda=\alpha$, т. е. за операцията конкатенация празната дума играе ролята на единица.

Операцията **степенуване** също може да се определи върху думи. Ако $\alpha$ е произволна дума то определяме $\alpha^0=\Lambda$. $\alpha^1=\alpha$, а за $n=2, 3, \ldots$ $\alpha^n=\alpha\alpha^{n-1}$ (конкатенация на $\alpha$ и $\alpha^{n-1}$).

Степента на една дума, е дума върху същата азбука.

Ще казваме, че думата $\alpha$ е **начало (префикс)** на думата $\beta$, ако съществува дума $\gamma$ такава, че $\beta=\alpha\gamma$.

Ако $\gamma\neq\Lambda$, $\alpha$ се нарича същинско начало на $\beta$.

Аналогично, думата $\alpha$ ще наричаме **край (суфикс)** на думата $\beta$, ако съществува дума $\gamma$, такава, че $\beta=\gamma\alpha$.

Ако $\gamma\neq\Lambda$, $\alpha$ е същински край на $\beta$.

И накрая, думата $\alpha$ ще наричаме **поддума** на думата $\beta$, ако съществуват думи $\gamma$ и $\delta$ такива, че $\beta=\gamma\alpha\delta$.

Ако $\gamma\delta\neq\Lambda$, $\alpha$ се нарича същинска поддума на $\beta$.

Например на думата *око* начало са думите $\Lambda$, *о*, *ок*, *око*, като първите три са същинско начало на *око*; а думите $\Lambda$, *о*, *ко*, *око* са край на същата дума.

Да вземем произволна непразна азбука $W$. Да означим с $W^*$ множеството на всички думи върху $W$. Очевидно, $W^*$ е безкрайно множество, тъй като можем да записваме думи с произволно

голяма дължина. Ще покажем, че $W^*$ е изброимо множество и ще посочим начин за изброяване на думите от $W^*$.

Да предположим, че в азбуката $W$ има $k$ символа. Думите от $W^*$ ще подреждаме така. На първо място поставяме празната дума. След това подреждаме думите по нарастването на тяхната дължина. За думите с еднаква дължина приемаме, че те представляват запис на числа в $k$-ична бройна система и тях подреждаме по големината на числата, които представят. По този начин всяка дума получава единствен номер (естествено число), различен от номерата на другите думи, а на всеки номер съответствува една единствена дума.

Да вземем азбуката $V_2=\{0, 1\}$ и да номерираме по този начин думите от $V_2^*$. Получаваме:

| | | |
|---|---|---|
| 1) | $\Lambda$ | дължина 0 |
| 2) | 0 | дължина 1 |
| 3) | 1 | |
| 4) | 00 | дължина 2 |
| 5) | 01 | |
| 6) | 10 | |
| 7) | 11 | дължина 3 |
| 8) | 000 | |
| 9) | 001 | |
| . . . | . . . | |
| . . . | . . . . . | |

## Упражнения

1. Дадена е азбуката $V_2=\{0, 1\}$ и множеството от думи $V_2^*=\{\Lambda, 0, 1, 00, 01, 11, 000, 001, \ldots\}$.

Да образуваме таблицата:

| | $\Lambda$ | 0 | 1 | 00 | 01 | 10 | 11 | 000 | . . . |
|---|---|---|---|---|---|---|---|---|---|
| $\Lambda$ | $\Lambda$ | 0 | 1 | 00 | 01 | 10 | 11 | 000 | . . . |
| 0 | 0 | 00 | 01 | 000 | 001 | 010 | 011 | 0000 | . . . |
| 1 | 1 | 10 | 11 | 100 | 101 | 110 | 111 | 1000 | . . . |
| 00 | 00 | 000 | 001 | 0000 | 0001 | 0010 | 0011 | 00000 | . . . |
| 01 | 01 | 010 | 011 | 0100 | 0101 | 0110 | 0111 | 01000 | . . . |
| 10 | 10 | 100 | 101 | 1000 | 1001 | 1010 | 1011 | 10000 | . . . |
| 11 | 11 | 110 | 111 | 1100 | 1101 | 1110 | 1111 | 11000 | . . . |
| 000 | 000 | 0000 | 0001 | 00000 | 00001 | 00010 | 00011 | 000000 | . . . |
| . | . . . . . | | | | | | | | |
| . | | | | | | | | | |
| . | | | | | | | | | |



Втази таблица на позицията, съответствуваща на $i$-тия ред и $j$-тия стълб стои думата, която се получава чрез конкатенация на $i$-тата и $j$-тата дума от $V_2^*$.

а) Намерете начин да определите колко пъти произволна дума $\alpha$ от $V_2^*$ ще се срещне в таблицата.

б) Номерирайте с числата 1, 2, 3, . . . думите от таблицата така, че думи на различни места да получат различни номера.

2. Дадена е азбуката $V=\{(, )\}$, съставена от лява и дясна скоба.

Някои думи от $V^*$ ще наричаме правилно построени.

Правилно построените думи от $V^*$ определяме по следния начин:

а) $\Lambda$ е правилно построена; б) ако $\alpha$ е правилно построена дума, то $(\alpha)$ е също правилно построена; в) ако $\alpha$ и $\beta$ са правилно построени думи, то $\alpha\beta$ е също правилно построена; г) няма други правилно построени думи освен получаваните по а), б) и в).

Докажете, че една дума $\alpha$ е правилно построена тогава и само тогава, когато са изпълнени следните две условия:

а) $\alpha$ има равен брой леви и десни скоби; б) всяко начало на $\alpha$ има поне толкова леви скоби „(“, колкото и десни скоби „)“.

3. **Палиндроми** ще наричаме следните думи:

а) $\Lambda$ е палиндром; б) ако $a$ е буква, то $a$ е палиндром; в) ако $a$ е буква, и $\alpha$ е палиндром, $a\alpha a$ е палиндром; г) палиндромите могат да се получават само по а), б) и в).

Докажете, че една дума $\alpha$ е палиндром тогава и само тогава, когато $\alpha = O(\alpha)$ (т. е., когато не се изменя при обръщане).

### 1.2. ФОРМАЛНИ ЕЗИЦИ

Към определянето на понятието формален език ще приложим същите изисквания, както при определянето на азбука и дума: ще искаме това понятие да обхваща всички съществуващи или възможни езици и да бъде математическо.

Бихме могли да изходим от така наречените **естествени езици**, като направим опит да ги определим математически. Под естествени езици обикновено се разбират езиците, които сега се говорят или някога са били говорени, от Човека. Такива например са българският, английският, руският, латинският и др. езици. Техният брой значително надхвърля числото 1000, а известна част от тях са така развити, че пълното им изучаване едва ли е по силите на отделния индивид. Освен това повечето естествени езици представляват динамично изменящо се явление и поради това в известна своя част са неясно определени и неточни. В този смисъл точното определяне на понятието естествен език и построяването върху тази база на понятието „формален език“ изглежда нецелесъобразно като подход. От друга страна, за самите естествени езици поради тяхното многообразие, развитост, богатство и изменяемост са нужни такива математически средства за изследване, каквито биха могли да бъдат например формалните езици.

Друг вид езици, които се появиха през последните тридесет години, са **езиците за програмиране**, предназначени да облекчат тежкия труд, свързан с програмирането на машинния език на компютрите. Такива езици са например ФОРТРАН, АЛГОЛ, PL/1 и други. Всеки компютър представлява физическа реализация на логически и математически елементи, а операциите, които извършва, се основават на съждителното смятане и двоичната аритметика. Следователно всяка програма, която трябва да бъде изпълнена от компютъра, както и резултатът, който той дава, се свеждат до верностни стойности и двоични числа. За намаляване на разстоянието между обичайната ни представа за езиково общуване и низовете от двоични числа и верностни стойности от машинния език на компютъра и за повишаване на ефективността на компютъра са построени и активно се създават хиляди езици за програмиране. Самите езици се задават, като се определя множеството от символи, които се използуват при записването на синтактически правилни програми, множеството от синтактически правилни програми, както и смисъла на всяка такава програма. Езиците за програмиране се състоят от различни части, които могат да бъдат също разглеждани като някакви, макар и твърде бедни, едностранни езици. Подобни езици се срещат отдавна в математиката, където можем да говорим например за езика на двоичната аритметика или за езика на алгебрата на множествата и т. н.

И така, можем да дадем следното общо определение за език: всяко множество от думи върху дадена азбука ще наричаме **език (формален език).**

Ако използуваме определеното вече от нас множество на всички думи върху дадена азбука, можем да кажем, че **едно множество $L$ е език тогава и само тогава, когато съществува азбука $V_L$ такава, че $L$ е подмножество на $V_L^*$.**

Даденото определение е достатъчно широко, за да обхване както естествените езици и езиците за програмиране, така и различните рекурсивни и рекурсивно-номерируеми множества, изучавани от теорията на автоматите. А това ни дава надеждата, че в теорията на формалните езици ще могат да бъдат построявани и изучавани математически модели за различни свойства на естествените езици, на езиците за програмиране и на множествата от низове, с които работят автоматите.

Действително, да вземем азбуката $V_1$, която съдържа всички букви от българската азбука и всички препинателни знаци, включително и шпацията. Тогава множеството на всички думи от българския език, множеството на всички изречения в българ-



ския език, множеството от всички съчинения на Иван Вазов и т. н. са формални езици. Ако вземем азбуката от всички основни символи на АЛГОЛ или ФОРТРАН, тогава множествата на синтактически правилните програми на АЛГОЛ или ФОРТРАН са формални езици.

Празното множество $\varnothing$ също е формален език. Формален език е и множеството $\{\Lambda\}$, състоящо се само от празната дума. Ще отбележим, че тези два формални езика са различни: $\varnothing \neq \{\Lambda\}$.

Да вземем азбуката $V_2=\{0, 1\}$. Множествата $L_1=\{0, 00, 010, 0110, 01110\}$; $L_2=\{0_n1^n;\ n\geq 0\}$; $L_3=\{1_p;\ p$ — просто число$\}$ са също формални езици върху $V_2$.

Ако $L$ е крайно множество, тогава $L$ се нарича **краен език**, в противен случай $L$ се нарича **безкраен език**. В горните примери езикът $L_1$ е краен, а $L_2$ и $L_3$ са безкрайни.

Върху формалните езици могат да се въведат различни операции. Тъй като формалните езици са множества, за тях операциите обединение, сечение, разлика могат да се въведат в обичайния теоретико-множествен смисъл.

Нека $L_1$ е език върху азбуката $V_1$, а $L_2$ е език върху азбуката $V_2$. Върху азбуката $V_1 \cup V_2$ определяме езика, съставен от всички конкатенации на думи от $L_1$ с думи от $L_2$. Този език наричаме **произведение на езиците** $L_1$ и $L_2$ и означаваме с $L_1L_2$. И така,

$$L_1L_2=\{\alpha\beta;\ \alpha\in L_1 \text{ и } \beta\in L_2\}.$$

Лесно може да се забележи, че произведението е асоциативна операция, но не е комутативна, т. е. за произволни езици $L_1$, $L_2$, $L_3$ $L_1(L_2L_3)=(L_1L_2)L_3$, но $L_1L_2\neq L_2L_1$. За тази операция езикът $\{\Lambda\}$ играе ролята на единица, а празното множество $\varnothing$ — ролята на нула, тъй като за произволен език $L$:

$$L\{\Lambda\}=\{\Lambda\}L=L;\ L\varnothing=\varnothing L=\varnothing.$$

За произволен език $L$ можем да определим и неговите **степени** по следния начин:

$$L^0=\{\Lambda\},\ L^1=L,\ L_n=LL^{n-1} \text{ за } n=2, 3. \ldots$$

Езикът $L^n$ се състои от всевъзможните конкатенации от $n$ на брой думи от езика $L$.

За да определим езика, който се състои от всевъзможните конкатенации на произволен брой думи от даден език $L$, ще въведем операцията итерация. **Итерация** $L^*$ на езика $L$ ще наричаме езика

$$L^*=\bigcup_{n\geq 0} L^n=L^0\cup L\cup L^2\cup L^3\cup \ldots$$

Нека е дадена произволна азбука $V$. Да приемем, че $V$ означава и езика, състоящ се от всички еднобуквени думи върху азбуката $V$. Тогава итерацията $V^*$ на езика $V$ ще се състои от всички думи върху азбуката $V$. А това показва, че можем да означаваме с една и съща звездичка итерацията $L^*$ на даден език $L$ и множеството $V^*$ на всички думи върху дадена азбука $V$.

Ще определим още една операция за произволен език $L$ върху азбуката $V$. **Допълнение на езика** ще наричаме множеството на всички думи върху $V$, които не принадлежат на $L$. Допълнението ще означаваме с $\overline{L}$. И така,

$$\overline{L}=V^*-L=\{\alpha\in V^*;\ \alpha\notin L\}.$$

Очевидно $\overline{V^*}=\varnothing$, а $\overline{\varnothing}=V^*$.

Как да описваме различните формални езици? Без общи начини за описване на езиците въведеното от нас понятие „формален език“ ще бъде безполезно. Ако езикът е краен, бихме могли да го опишем, като съставим списък на думите му. Този метод обаче практически е приложим само към твърде ограничен брой езици и също не дава никакви преимущества при тяхното изследване. При това, което е особено важно, бихме желали описанието както на крайните, така и на безкрайните езици да бъде крайно, т. е. да бъде с краен обем. Лесно може да се докаже, че не за всеки формален език върху дадена азбука ще има такова крайно описание. Действително, вече показахме, че **множеството на всички думи върху дадена непразна азбука е безкрайно изброимо множество**. Известно е обаче че множеството на всички подмножества на безкрайно изброимо множество не е изброимо, т. е. множеството на различните езици върху дадена азбука не е изброимо. Въпреки че не сме определили какво точно разбираме под „крайно описание“, ние интуитивно сме убедени, че не може да има повече от изброимо много крайни описания, тъй като всяко крайно описание трябва да бъде записано с краен брой думи. Следователно, двете множества — множеството на всички крайни описания и множеството на всички езици, имат различна мощност, т. е. **има формални езици, за които не съществуват крайни описания**.

Въпреки това, от гледна точка на реалните явления, които искахме да изучаваме чрез формалните езици, се налага да търсим методи за крайно описване на езиците.

В математическата лингвистика и по-точно в теорията на формалните езици се изследват именно онези езици и класове от езици, за които съществуват крайни описания.



Ще се спрем накратко върху два основни метода за описване на езиците, удовлетворяващи това изискване.

Първият метод се състои в това да зададем частичен алгоритъм, който проверява за произволна дума дали тя принадлежи на езика. В общия случай **частичният алгоритъм** ще спира и ще дава отговор „ДА“, ако думата, която проверява, е от езика, и ще спира с отговор „НЕ“ или ще работи неограничено дълго, ако думата не е от езика. За такъв частичен алгоритъм ще казваме, че **разпознава** езика, а устройството, което осъществява частичния алгоритъм, ще наричаме **разпознавател**.

Ще покажем как при наличието на разпознавател за езика $L$ върху азбуката $V$ могат да се изброяват думите от $L$. Вече разгледахме един начин за изброяване на думите от $V^*$. Не можем обаче подред да проверяваме чрез разпознавателя думите от $V^*$, тъй като за някои думи, които ние предварително не знаем, разпознавателят може да работи неопределено дълго. Ще считаме, че разпознавателят работи на тактове, така че можем да говорим, например, за $i$-тия му такт. Освен това ще използуваме един известен факт: множеството от наредени двойки $(m, n)$ от естествени числа може да се номерира с естествения ред на числата. Например номерът на наредената двойка $(m, n)$ се получава по формулата:

$$\mathrm{Nom}\ ((m,n))=\frac{(m+n-1)\ (m+n-2)}{2}+n.$$

Получаваме следното подреждане: (1, 1), (2, 1), (1, 2), (3, 1), (2, 2), (1, 3), . . . . Сега вече можем да преминем към номериране на думите от $L$. Номерираме последователно наредените двойки от естествени числа. Когато номерираме двойката $(m, n)$, пораждаме $m$-тата дума от $V^*$ (съгласно методът за номериране на думите от $V^*$) и върху тази дума пускаме разпознавателя да извърши първите $n$ такта. Ако той определи, че $m$-тата дума принадлежи на $L$, вписваме тази дума в редицата от думи, разпознати до този момент. След това преминаваме към следващата наредена двойка. Тъй като всяка дума, която принадлежи на $L$, се разпознава за краен брой тактове чрез разглеждания метод, разпознавателят действително номерира думите от $L$ и само тях. Поради това езиците, за които съществува разпознавател, ще наричаме **рекурсивно номерируеми.** Езици, за които разпознавателят винаги спира след краен брой стъпки с отговор „ДА“ или „НЕ“, ще наричаме **рекурсивни**.

Вторият метод за описване на формалните езици се състои в задаването на пораждаща система, наречена **формална**

**граматика,** която строи думата от езика по строго определени правила. По този начин формалната граматика поражда всички думи на езика и само тях. Редицата от действия, пораждаща дадена дума, приписва на тази дума определена структура, което прави този метод особено полезен от лингвистична гледна точка.

В следващите глави ще се занимаваме основно с изучаването на различни множества от думи — формални езици, с техните описания, структура и свойства и с прилагането на получените резултати при изследване на естествените езици, езиците за програмиране и т. н.

### Упражнения

1. Докажете, че операцията умножение е дистрибутивна относно обединението, т. е. че за произволни три езика $L_1$, $L_2$, $L_3$: $L_1(L_2 \cup L_3) = L_1L_2 \cup L_1L_3$.

2. Посочете примери на езици $L_1$, $L_2$, $L_3$, за които обединението не е дистрибутивно относно умножението, умножението не е дистрибутивно относно сечението и сечението не е дистрибутивно относно умножението, т. е.

$L_1 \cup (L_2L_3) \neq (L_1 \cup L_2)(L_1 \cup L_3)$; $L_1(L_2 \cap L_3) \neq L_1L_2 \cap L_1L_3$; $L_1 \cap (L_2L_3) \neq (L_1 \cap L_2)\ (L_1 \cap L_3)$.

3. Докажете, че обединението и сечението на два рекурсивно номерируеми езика са рекурсивно номерируеми езици.

4. Докажете, че ако един език и неговото допълнение са рекурсивно номерируеми, то този език е рекурсивен.

5. Докажете, че всеки краен език е рекурсивен.

## 1.3. ПОРАЖДАЩИ ГРАМАТИКИ

Формални граматики нарекохме такива описани с крайни средства математически системи, които пораждат думите на един краен или безкраен език. Както вече отбелязахме, чрез формални граматики не могат да се породят всички формални езици, но за езиците, представляващи интерес за лингвистиката и информатиката, формални граматики съществуват.

Ще въведем един сравнително прост вид формални граматики, наречени **пораждащи граматики.** (По-нататък, често пъти вместо пораждаща граматика на езика $L$ ще казваме граматика на езика $L$.) Пораждащите граматики се използуват изключително широко не само в лингвистиката, но преди всичко в информатиката — от езиците за програмиране и техния автоматичен превод до моделирането на изчислителни процеси. Понятието „пораждаща граматика“ ще бъде достатъчно за нашите цели, тъй като може фактически да се покаже, че за всеки език, който се описва с крайни средства, има и пораждаща го граматика.



Една пораждаща граматика се състои от четири части:

— **азбука на терминалните символи,** която представлява азбуката на пораждания език;

— **азбука на нетерминалните символи,** които участвуват в процеса на пораждане, но в думите на езика не се съдържат;

— **начален символ,** който представлява фиксиран за граматиката нетерминален символ;

— **правила на граматиката,** които описват процеса на пораждане и представляват наредени двойки от думи, съставени от терминални и нетерминални символи; в първата компонента на всяка двойка има поне един нетерминален символ. Правилата на граматиката обикновено се записват не като наредени двойки $(\alpha, \beta)$, а във вида $\alpha \to \beta$, което се чете „думата $\alpha$ се замества с думата $\beta$“. Считаме, че знакът $\to$ не е нито терминален, нито нетерминален символ.

И така, **пораждаща граматика** $\Gamma$ наричаме наредената четворка $\Gamma = (V, W, S, P)$, в която:

1) $V$ е азбука на терминалните символи;

2) $W$ е непразна азбука на нетерминалните символи;

(При това двете азбуки $V$ и $W$ нямат общи символи (т. е. $V \cap W = \varnothing$.)

3) $S \in W$ и се нарича начален символ;

4) $P$ е крайно множество от правила от вида $\alpha \to \beta$, в които $\alpha$, $\beta$ са от множеството на думите $(V \cup W)^*$, знакът $\to$ не е от азбуката $V \cup W$, а в $\alpha$ има поне един символ от азбуката $W$.

Граматиката поражда езика, като извежда от началния символ думи, съставени от терминални символи. Казваме, че **една дума $\rho$ се извежда непосредствено от думата $\omega$ в граматиката** $\Gamma = (V, W, S, P)$, ако съществуват думи $\gamma$ и $\delta$ от множеството $(V \cup W)^*$ и правило $\alpha \to \beta$ от $P$ такива, че

$$\omega = \gamma\alpha\delta;\ \rho = \gamma\beta\delta.$$

Непосредственото извеждане на думата $\rho$ от думата $\omega$ в граматиката $\Gamma$ записваме така: $\omega \overset{\Gamma}{\vdash} \rho$.

Ще казваме, че **думата $\rho$ се извежда от думата $\omega$ в граматиката** $\Gamma$, ако има думи $\omega_1, \omega_2, \ldots, \omega_n$; $n \geq 1$, всяка от които непосредствено се извежда от предишната в граматиката $\Gamma$, $\omega_1 = \omega$, а $\rho$ непосредствено се извежда от $\omega_n$ в граматиката $\Gamma$, т. е.

$$\omega = \omega_1;\ \omega_1 \overset{\Gamma}{\vdash} \omega_2 \ldots \omega_n \overset{\Gamma}{\vdash} \rho.$$

Когато думата $\rho$ се извежда от $\omega$ в граматиката $\Gamma$, това се записва така: $\omega \overset{\Gamma}{\vdash} \rho$.[1]

Казваме, че една дума се **поражда** от дадена граматика, ако тази дума е съставена само от терминални символи и се извежда в граматиката от началния символ.

Множеството на всички породени от дадена граматика думи представлява езика $L(\Gamma)$, който граматиката поражда, т. е.

$$L(\Gamma)=\{\omega;\ \omega\in V^* \text{ и } S \overset{\Gamma}{\models} \omega\}.$$

Да разгледаме няколко примера.

I) В граматиката

$\Gamma_1=(\{0,1\},\ \{S\},\ S,\ \{S\to SS,\ S\to 0S1,\ S\to 1S0,\ S\to 01,\ S\to 10\})$

терминални символи са 0 и 1, има само един нетерминален символ $S$, който е и начален символ, а правилата на граматиката са: 1) $S\to SS$, 2) $S\to 0S1$, 3) $S\to 1S0$, 4) $S\to 01$, 5) $S\to 10$. В тази граматика можем да изведем например думите 01010110 и 00111001 по следния начин:

$$S\vdash SS\vdash 0S1S\vdash 0S1SS\vdash 0S101S\vdash 0S10110\vdash 01010110;$$
$$S\vdash 0S1\vdash 0SS1\vdash 001S1\vdash 0011S01\vdash 00111001.$$

В първия извод прилагаме последователно правилата 1), 2), 1), 4), 5), 5), а във втория извод — правилата 2), 1), 4), 3), 5).

Лесно може да се забележи, че граматиката $\Gamma_1$ поражда само непразни думи, съставени от равен брой нули и единици. Действително правилата на граматиката са такива, че всяко правило или не прибавя, или прибавя равен брой нули и единици към непосредствено извежданата чрез това правило дума. И тъй като в началната дума на извода $S$ няма нули и единици, то всяка дума от нули и единици, която граматиката $\Gamma_1$ поражда, ще има равен брой нули и единици. Очевидно всички пораждани от $\Gamma_1$ думи са непразни, тъй като дума от терминални символи не може да се получи без прилагането на правила 4) и 5).

Чрез индукция по дължината на думата ще докажем и обратното — всяка непразна дума, съставена от равен брой нули и единици, се поражда от граматиката $\Gamma_1$.

[1] Знакът $\overset{\Gamma}{\vdash}$ използуваме, когато е извършена точно една стъпка от извода, а със знака $\overset{\Gamma}{\models}$ означаваме възможността за повече от една стъпка. Когато граматиката е фиксирана, ще пишем само $\vdash$ или $\models$ без буквата $\Gamma$.



При $n=2$ има две думи с равен брой 0 и 1. Това са 01 и 10. И двете думи се пораждат от граматиката $\Gamma_1$ с еднократно прилагане на правилата 4) и 5).

Да допуснем, че всички думи с дължина не по-голяма от $2n$, съставени от равен брой 0 и 1, се пораждат от граматиката $\Gamma_1$. Ще докажем твърдението за такава дума с дължина $2n+2$. Ще разгледаме два случая: 1) когато думата започва и завършва с различни букви и 2) когато думата започва и завършва с една и съща буква. Нека дадената дума $\omega$ с дължина $2n+2$ започва и завършва с различни букви. Тогава $\omega$ може да се представи като $0\alpha 1$ или като $1\alpha 0$. Но $\alpha$ е с дължина $2n$ и в $\alpha$ трябва също да има равен брой 0 и 1. Тогава според индуктивната хипотеза $S \overset{\Gamma_1}{\models} \alpha$. В такъв случай за $\omega$ получаваме извода $S \overset{\Gamma_1}{\vdash} 0S1 \overset{\Gamma_1}{\models} 0\alpha 1$ или извода $S \overset{\Gamma_1}{\vdash} 1S0 \overset{\Gamma_1}{\models} 1\alpha 0$.

Нека дадената дума $\omega$ с дължина $2n+2$ започва и завършва с една и съща буква, например $\omega=0\alpha 0$. Тогава в думата $0\alpha$ единиците са с една повече от нулите, така че, започвайки от началната буква 0, преди да достигнем до последната буква на $0\alpha$, ще намерим същинско начало $\beta$ на $0\alpha$ с равен брой 0 и 1. В такъв случай $\omega=\beta\gamma$, където $\beta$ и $\gamma$ са думи с равен брой 0 и 1 и чиято дължина е по-голяма от 0 и по-малка от $2n$. Съгласно индуктивната хипотеза $S \overset{\Gamma_1}{\models} \beta$ и $S \overset{\Gamma_1}{\models} \gamma$. Но тогава за $\omega$ получаваме:

$$S \overset{\Gamma_1}{\models} SS \overset{\Gamma_1}{\models} \beta S \overset{\Gamma_1}{\models} \beta\gamma=\omega,$$

т. е. в този случай $\omega$ се поражда от граматиката $\Gamma_1$. Когато $\omega=1\alpha 1$, разсъждаваме аналогично.

И така, вече можем да опишем точно езика $L(\Gamma_1)$, който зададената граматика $\Gamma_1$ поражда:

$L(\Gamma_1)=\{\omega;\ \partial(\omega)\geqq 2$ и в $\omega$ има равен брой 0 и 1$\}$.

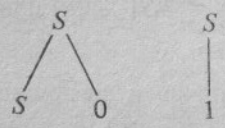
II) Да разгледаме граматиката

$\Gamma_2=(\{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\},\ \{S\},\ S,\ \{S\to S0,\ S\to S1,\ S\to S2,\ S\to S3,\ S\to S4,\ S\to S5,\ S\to S6,\ S\to S7,\ S\to S8, S\to S9,\ S\to 1,\ S\to 2,\ S\to 3,\ S\to 4,\ S\to 5, S\to 6,\ S\to 7,\ S\to 8,\ S\to 9\})$.

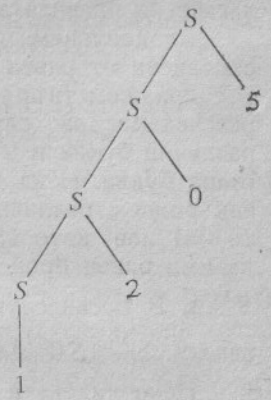
Тази граматика поражда десетичните записи на естествените числа, т. е. 0, 1, 2, . . . . . , 9, 10, 11, . . . . Например изводът на 1205 от $S$ изглежда така $S\vdash S5\vdash S05\vdash S205\vdash 1205$.

Ако всяко от правилата изобразим например по начина, показан на фиг. 4,

Фиг. 4




Фиг. 5




за извода на 1205 ще получим графичното представяне от фиг. 5.

III) Граматиката

$\Gamma_3 = (\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}, \{S, A\}, S, \{S \to SA, S \to A, A \to 0, A \to 1, A \to 2, A \to 3, A \to 4, A \to 5, A \to 6, A \to 7, A \to 8, A \to 9\})$

също поражда десетичните записи на естествените числа, но, възможно, с някакъв блок от нули пред първата значеща цифра:

0001205, 009, 195 и т. н.

Можем да интерпретираме нетерминалните символи на тази граматика като някакви синтактични категории за думите, които съставят пораждания език, а правилата на граматиките като връзките, които съществуват между тях. В разглеждания език такива синтактични категории са „цифра“ и „цяло неотрицателно число“ и в такъв случай $A$ по естествен начин можем да разглеждаме като означение на „цифра“, а $S$—като означение на „цяло неотрицателно число“. Правилата на граматиката показват, че 0,1,. . .,9 са цифри, цифрите означават цели неотрицателни числа, и ако към едно цяло неотрицателно число добавим открая цифра, отново получаваме цяло неотрицателно число. Освен това тези правила позволяват от основната синтактична категория на този език — цяло неотрицателно число, да пораждаме думите на езика, т. е. самите цели неотрицателни числа. Ако въведем вместо знака → друг знак :: =, който ще четем „по определение е“, а синтактичните категории заградим с ъглови скоби, ще получим следните правила:

⟨цяло неотрицателно число⟩ :: = ⟨цяло неотрицателно число⟩ ⟨цифра⟩;

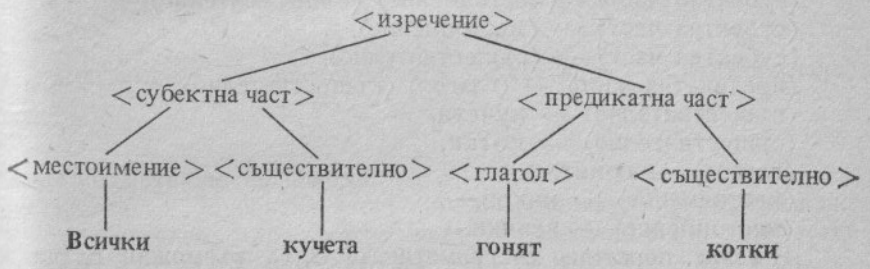


⟨цяло неотрицателно число⟩ :: = ⟨цифра⟩
⟨цифра⟩ :: = 0
⟨цифра⟩ :: = 1
. . . . . .
⟨цифра⟩ :: = 9.

Както ще видим по-нататък, по подобен начин се определя синтаксисът на езиците за програмиране.

Още по-добре интерпретацията на нетерминалните символи като синтактични категории или променливи си личи върху примери от естествените езици.

IV) Да вземем изречението „**Всички кучета гонят котки**“ и да изобразим графично синтактичния анализ, който ще направим на това изречение:




Фиг. 6

И така, това изречение се състои от субектна част и предикатна част; субектната му част е съставена от местоимение и съществително, а предикатната част — от глагол и съществително. Местоимението е „**всички**“, съществителните са „**кучета**“ и „**котки**“, а глаголът е „**гонят**“. Чертите определят връзките, които съществуват между различните синтактични категории или между тях и елементите на езика. Това графично описание можем да разглеждаме и като някакъв процес на пораждане на граматично правилни изречения: синтактичната категория ⟨изречение⟩ може да се замени със синтактичните категории ⟨субектна част⟩ ⟨предикатна част⟩, записани една след друга. Категорията ⟨субектна част⟩ може да се замени с последователно написаните категории ⟨местоимение⟩ ⟨съществително⟩, а предикатната част — с ⟨глагол⟩ ⟨съществително⟩. От своя страна, синтактичните категории ⟨местоимение⟩, ⟨съществително⟩, ⟨глагол⟩ могат да се заменят съответно с думи от речника на езика такива, като: „**всички**“, „**кучета**“, „**котки**“, „**гонят**“. По този начин получаваме пораждаща граматика,

която в частност поражда и разглежданото от нас изречение. Ще разширим малко тази граматика, като добавим в нея възможността субектната част да се състои само от местоимение или само от съществително. Ще добавим и още едно местоимение — „**някои**".

Получаваме следната пораждаща граматика **G**, в която азбуката на терминалните символи е въведената от нас вече азбука

$V_3$ = **(гонят, някои, всички, кучета, котки).**

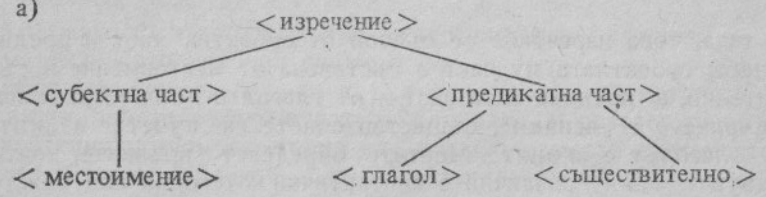
Азбуката на нетерминалните символи е

$W$ = {⟨изречение⟩, ⟨субектна част⟩, ⟨предикатна част⟩, ⟨местоимение⟩, ⟨съществително⟩, ⟨глагол⟩}, а начален символ, както може да се очаква, е ⟨изречение⟩. Граматиката **G** има следните правила:

⟨изречение⟩ → ⟨субектна част⟩ ⟨предикатна част⟩,
⟨субектна част⟩ → ⟨местоимение⟩ ⟨съществително⟩,
⟨субектна част⟩ → ⟨местоимение⟩,
⟨субектна част⟩ → ⟨съществително⟩,
⟨предикатна част⟩ → ⟨глагол⟩ ⟨съществително⟩,
⟨съществително⟩ → **кучета,**
⟨съществително⟩ → **котки,**
⟨глагол⟩ → **гонят,**
⟨местоимение⟩ → **някои,**
⟨местоимение⟩ → **всички.**

В езика, пораждан от граматиката **G**, са възможни следните три структури:

а)

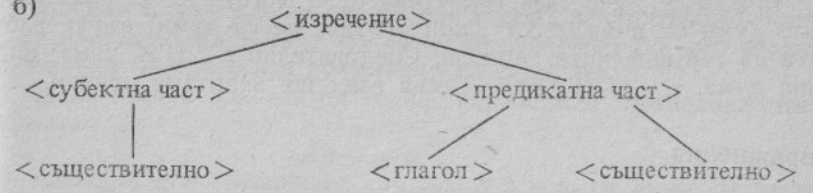



Фиг. 7

съответствуваща на извода:
⟨изречение⟩ ⊢ ⟨субектна част⟩ ⟨предикатна част⟩ ⊢ ⟨местоимение⟩ ⟨предикатна част⟩ ⊢ ⟨местоимение⟩ ⟨глагол⟩ ⟨съществително⟩;
съответствуваща на извода:
⟨изречение⟩ ⊢ ⟨субектна част⟩ ⟨предикатна част⟩ ⊢ ⟨съществително⟩ ⟨предикатна част⟩ ⊢ ⟨съществително⟩ ⟨глагол⟩ ⟨съществително⟩;

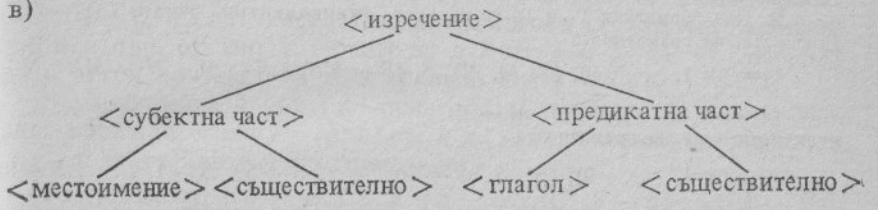


б)




Фиг. 8

в)




Фиг. 9

съответствуваща на извода:
⟨изречение⟩ ⊢ ⟨субектна част⟩ ⟨предикатна част⟩ ⊢ ⟨местоимение⟩ ⟨съществително⟩ ⟨предикатна част⟩ ⊢ ⟨местоимение⟩ ⟨съществително⟩ ⟨глагол⟩ ⟨съществително⟩.

Оттук лесно можем да получим езика $L$ (**G**), пораждан от граматиката **G**, като извършим възможните замени с терминални символи. Този език се състои от следните изречения:

$L$ (**G**) = {**всички гонят котки, всички гонят кучета, някои гонят котки, някои гонят кучета, котки гонят кучета, кучета гонят котки, котки гонят котки, кучета гонят кучета, всички котки гонят котки, всички котки гонят кучета, всички кучета гонят котки, всички кучета гонят кучета, някои котки гонят котки, някои кучета гонят котки, някои котки гонят кучета, някои кучета гонят кучета**}.

В избрания от нас пример всички низове или по-точно изречения от езика $L$(**G**) са осмислени. Това обаче в повечето случаи не е така, тъй като пораждащата граматика дава синтактически правилно построени низове или изречения, но няма никакво отношение към техния смисъл.

Накрая да разгледаме граматиката

$$\Gamma_5 = (\{0, 1\}, \{S\}, S, \{S \to 0S1\}).$$

Вижда се, че в тази граматика от $S$ могат да се извеждат само думи от вида $0^n S 1^n$, които обаче не са думи върху азбуката на терминалните символи. Следователно в $L(\Gamma_5)$ няма нито една дума, т. е. $L(\Gamma_5)=\varnothing$. Такъв език ще наричаме **празен.**

## Упражнения

**1.** Дадена е граматиката
$\Gamma=(\{a, b\}\ \{S, A\}, S, \{S\to aAb, aA\to aaAB, A\to\Lambda\})$.
а) Намерете няколко думи с различна дължина, породени от Г; б) Намерете езика $L(\Gamma)$.

**2.** Две граматики $\Gamma_1$ и $\Gamma_2$ се наричат **еквивалентни**, когато $L(\Gamma_1)=L(\Gamma_2)$. Докажете, че граматиката

$\Gamma_1=(\{0, 1, \ldots, 9\}, \{S, M, N\}, S, \{S\to MN,\quad S\to M,\quad N\to MN\quad N\to 0$

$N\to M,\ M\to 1, \ldots, M\to 9\})$
е еквивалентна на граматиката

$\Gamma_2=(\{0, 1, \ldots, 9\}, \{S\}, S, \{S\to S0, \ldots, S\to S9, S\to 1, \ldots, S\to 9\})$.

**3.** Трябва ли еквивалентните граматики да имат еднакви:
а) азбуки на терминалните символи; б) азбуки на нетерминалните символи?

**4.** Докажете, че граматиката

$\Gamma=(\{a, b, c\}, \{S, A, B\}, S, \{S\to a\ b\ c, S\to a\ A\ B\ c, Ab\to bA, AC\to B\ b\ c\ c$

$BC\to Cb, aC\to aaA, aC\to aa\})$

поражда езика $L=\{a^n b^n c^n;\ n\geq 1\}$.

**5.** Намерете езика, породен от граматиката
$\Gamma=(\{x, +, *, (, )\}, \{S, A, B\}, S, \{S\to A,\ S\to S+A,\ A\to A* B,\ A\to B$

$B\to x,\ B\to(S)\})$.

**6.** Намерете пораждаща граматика за езика:

а) $L=\{\omega\omega;\ \omega\in\{0, 1\}^*\}$; б) $L=\{\omega;\ \omega\in\{0, 1\}^*\}$; в) $L=\{a^{2^n};\ n\geq 0\}$; г) $L=\{a^{n^2};\ n\geq 1\}$.

### 1.4. КЛАСОВЕ ОТ ПОРАЖДАЩИ ГРАМАТИКИ И ЕЗИЦИ

Пораждащите граматики се строят за едни или други цели и към тях се прилагат различни специфични за целта изисквания. Обикновено тези изисквания се постигат, като се налагат ограничения върху вида на правилата на граматиката и с това се изменя тяхната пораждаща сила. За пръв път американският лингвист Н. Чомски в края на 50-те години предлага класификация на пораждащите граматики според вида на правилата им, която днес е общоприета и известна като **йерархия на Чомски** за пораждащите граматики.



Прието е пораждащи граматики, върху правилата на които не са наложени някакви допълнителни условия, да се наричат **граматики от тип 0** или **граматики от общ вид.**

Нека всички правила на една пораждаща граматика имат формата

$$\alpha A\beta\to\alpha\omega\beta,$$

при което $A$ е нетерминален символ, $\alpha$, $\beta$ и $\omega$ са думи от терминални и нетерминални символи, $\alpha$ и $\beta$ могат да бъдат и празни думи, а $\omega$ е непразна дума. При това правило нетерминалният символ $A$ може да се замества с думата $\omega$ само в даден „контекст“ $\alpha-\beta$, поради което такива граматики се наричат **контекстни** или още от **тип 1.** Прието е, думите $\alpha$ и $\beta$ да се наричат ляв и десен контекст на $A$ в правило $\alpha A\beta\to\alpha\omega\beta$.

В упражнение 4 от 1.3 приведохме пример на граматика, която поражда езика $L=\{a^n b^n c^n;\ n\geq 1\}$. Този език се поражда и от следната контекстна граматика:

$\Gamma=(\{a, b, c\}, \{S, A, B, C\}, S, \{S\to a\,SAC,\ S\to abC,\ CA\to BA,$
$BA\to BC,\ BC\to AC,\ bA\to bb,\ C\to c\})$.

В третото правило на Г буквата $C$ има десен контекст $A$, в четвъртото правило $A$ има ляв контекст $B$, в петото правило $B$ има десен контекст $C$, а в шестото правило $A$ има ляв контекст $b$. В тази граматика думата $a^2 b^2 c^2$, например, се поражда чрез следния извод:

$$S\vdash aSAC\vdash aabCAC\vdash aabBAC\vdash aabBCC\vdash aab\,ACC$$
$$\vdash aabbCC\vdash aabbcC\vdash aabbcc.$$

Така определени контекстните граматики не могат да пораждат празната дума, тъй като дясната част на всяко правило не е празна. Допълнително ще разширим нашето определение така, че контекстните граматики да могат да пораждат и празната дума.

Ще разрешим в контекстните граматики да има правило $S\to\Lambda$, където $S$ е началният символ, а $\Lambda$ — празната дума при условие, че началният символ $S$ не се среща в десните части на правилата на граматиката. Тогава правилото $S\to\Lambda$ ще може да се прилага еднократно, ще поражда само празната дума и няма да влияе върху пораждането на останалите думи. Обаче съгласно даденото определение за контекстна граматика началният символ $S$ може да се среща в десни части на правила от граматиката. Така е и в дадения по-горе пример. За такава контекстна граматика може да се намери еквивалентна на нея също контекстна граматика, за която условието — началният символ да не се среща в десните части на правилата, е изпълнено. За целта към нетерминалните символи на дадена контекстна граматика

добавяме нов нетерминален символ $S_1$, който определяме за начален символ на новата граматика. Азбуката на терминалните символи е същата, а правилата на новата граматика се състоят от всички правила на дадената граматика, към които за всяко правило от вида $S \to \alpha$ от дадената граматика е добавено ново правило $S_1 \to \alpha$. В такъв случай $S_1$ не се среща отдясно на никое правило, а новата граматика е контекстна, тъй като не сме променяли вида на правилата. Доказателството на това, че двете граматики са еквивалентни, предоставяме за упражнение.

И така, **контекстни** ще наричаме и граматики, в които има едно правило $S \to \Lambda$, където $S$ е началният символ, $\Lambda$ — празната дума, при условие, че $S$ не се среща в десните части на правилата на граматиката, а останалите правила са от вида $\alpha A \beta \to \alpha \omega \beta$, където $\alpha$, $\beta$, $\omega$ са думи от терминални и нетерминални символи, $\omega \neq \Lambda$, $A$ е нетерминален символ.

Например за езика $\{a^n b^n c^n;\ n \geqq 1\} \cup \{\Lambda\}$, т. е. за $\{a^n b^n c^n;\ n \geqq 0\}$, получаваме следната контекстна граматика:

$$\Gamma = (\{a, b, c\}, \{S_1, S, A, B, C\}, S_1, \{S_1 \to aSAC,\ S_1 \to abC,$$
$$S \to aSAC,\ S \to abC,\ CA \to BA,\ BA \to BC,\ BC \to AC,\ bA \to bb,$$
$$C \to c,\ S_1 \to \Lambda\}).$$

Следващият вид граматики са **безконтекстните граматики**, или още — **граматиките от тип 2**. Една пораждаща граматика е безконтекстна, ако всяко нейно правило е от вида $A \to \omega$, където $A$ е нетерминален символ, а $\omega$ непразна дума от терминални и нетерминални символи. Тези граматики се наричат безконтекстни, тъй като при тях левият и десният контекст на правилата от контекстните граматики са празни думи, т. е. нетерминалният символ $A$ може да се замества с $\omega$ независимо от контекста.

Така определени, безконтекстните граматики също не пораждат празната дума. Ще разширим нашето определение по същия начин, както това направихме при контекстните граматики: **безконтекстни** ще наричаме и всички граматики, в които има едно правило $S \to \Lambda$ ($S$ е началният символ) при условие, че $S$ не се среща в десните части на правилата на граматиката, а останалите правила са от вида $A \to \omega$. ($A$ — нетерминален символ, а $\omega$ — непразна дума от терминални и нетерминални символи).

Фактът, че за всяка безконтекстна граматика може да се намери еквивалентна на нея безконтекстна граматика, в която началният символ не се среща в дясната част на което и да е граматично правило, се доказва по същия начин, както при контекстните граматики.



Вижда се, че всяка безконтекстна граматика може да се разглежда и като контекстна, т. е. **класът на контекстните граматики обхваща класа на безконтекстните.**

Безконтекстна е например граматиката
$\Gamma_1 = (\{0, 1\}, \{S\}, S, \{S \to SS,\ S \to 0S1,\ S \to 1S0,\ S \to 01,\ S \to 10\})$,
за която доказахме, че поражда езика $L$ на думите, съставени от равен, но различен от нула, брой 0 и 1. Ако искаме към този език да прибавим и празната дума $\Lambda$, т. е. той да се състои от всички думи с равен брой 0 и 1, тогава езикът $L \cup \{\Lambda\}$ ще се поражда от следната безконтекстна граматика:

$\Gamma_1' = (\{0, 1\}, \{S_1, S\}, S_1, \{S_1 \to SS,\ S_1 \to 0S1,\ S_1 \to 1S0,\ S_1 \to 01,$
$S_1 \to 10,\ S \to SS,\ S \to 0S1,\ S \to 1S0,\ S \to 01,\ S \to 10,\ S_1 \to \Lambda\})$.

Безконтекстна е и граматиката:
$\Gamma_2 = (\{a, b\}, \{S, A\}, S, \{S \to aAb,\ A \to aAb,\ S \to ab,\ A \to ab,\ S \to \Lambda\})$,
която поражда езика $L(\Gamma_2) = \{a^n b^n;\ n \geqq 0\}$.

И накрая, последният вид граматики, които ще определим, са **автоматните граматики** (или още **граматики от тип 3; регулярни граматики**).

Една пораждаща граматика се нарича **автоматна**, ако всяко нейно правило е в една от формите: $A \to aB$; $A \to a$, където $A$ и $B$ са нетерминални символи, а $a$ е терминален символ. И тук ще разширим това понятие, като наречем **автоматни** и всички граматики, в които има правило $S \to \Lambda$ ($S$ — начален символ) при условие, че $S$ не се среща отдясно на никое правило на граматиката, а всички останали правила са от вида $A \to aB$ или $A \to a$.

Ако началният символ се среща отдясно на някои правила, можем да построим еквивалентна автоматна граматика, в която началният символ да се среща само отляво. Това може да стане по начина, който приложихме към контекстните граматики.

Граматиката $\Gamma_2$, която описахме в предишния параграф, е автоматна.

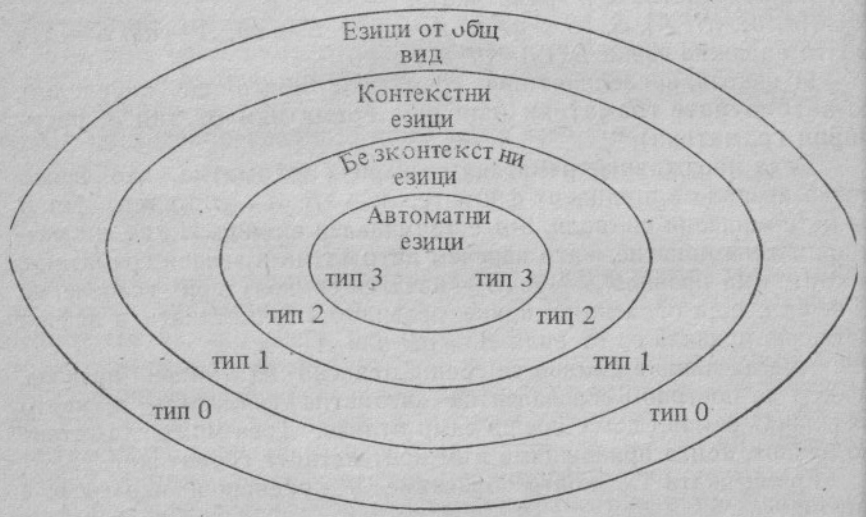
Друг пример за автоматна граматика е
$\Gamma = (\{0, 1\}, \{S, A\}, S, \{S \to 0S,\ S \to 1S,\ S \to 1A,\ A \to 1\})$,
която поражда всички думи от нули и единици, които завършват с две единици.

От дадените определения следва, че **класът на автоматните граматики се включва в класа на безконтекстните граматики.**

**Един формален език** ще наричаме **безконтекстен, автоматен, контекстен и от общ вид**, ако има граматика от съответния тип, която да го поражда.

Автоматните, безконтекстните, контекстните и от общ вид пораждащи граматики (езици) образуват **йерархията на Чомски за пораждащите граматики (езици).**

Както вече видяхме, класът на автоматните езици се включва в класа на безконтекстните езици, класът на безконтекстните езици се включва в класа на контекстните, а той от своя страна — в класа на езиците от общ вид. Възниква обаче въпросът дали това включване е същинско, т. е. дали например за всеки безконтекстен език не може да се намери пораждаща го автоматна граматика, така че класът на автоматните езици да съвпадне с класа на безконтекстните? По-нататък ще покажем, че има безконтекстен език, който не е автоматен, контекстен език, който не е безконтекстен и т. н., така че в йерархията на Чомски класовете строго се включват един в друг (фиг. 10):




Фиг. 10

Тази класификация на пораждащите граматики и съответно на формалните езици е интересна от гледна точка на естествените езици с това, че контекстните, а следователно безконтекстните и автоматните езици, притежават едно основно качество на естествените езици — тяхната рекурсивност. И тъй като езиците от общ тип не са рекурсивни в общия случай, а само рекурсивно-номерируеми, тази класификация дава възможност основните приложения на формалните езици да се търсят сред контекстните езици. От лингвистична гледна точка е естествено да се предположи, че носителят на езика винаги може да констатира правилността на някаква фраза от езика, независимо от това дали я чува за първи път, или не, да определи дали тази фраза е от езика, или не. С други думи у носителя на езика има разпознаващ алгоритъм, чрез който той определя за произволен низ дали този низ е от езика, или не. Както вече изтъкнахме, езици, за които съществува алгоритъм, определящ дали произволна дума е от езика, или не, се наричат рекурсивни.



**Контекстните езици също са рекурсивни.**

За да покажем това, ще построим алгоритъм, който да разпознава принадлежността на произволна дума към езика. Нека е дадена произволна контекстна граматика

$$\Gamma = (V, W, S, P).$$

Празната дума принадлежи на $L(\Gamma)$, само ако в $P$ има правило $S \to \Lambda$, така че принадлежността на $\Lambda$ към $L(\Gamma)$ се проверява с наличието или отсъствието на такова правило. Да премахнем правилото $S \to \Lambda$. Получаваме нова контекстна граматика

$$\Gamma' = (V, W, S, P'),$$

която поражда езика $L(\Gamma) - \{\Lambda\}$. Ще отбележим, че в граматиката $\Gamma'$, съгласно даденото определение за контекстна граматика правилата са такива, че дясната им част не може да бъде по-къса от лявата. А това означава, че при всеки извод в граматиката, $\Gamma'$ дължината на извежданите думи в сравнение с началната дума не може да намалява. Това съображение играе важна роля в нашите разсъждения.

Нека $\omega$ е произволна дума с дължина $p$, породена от граматиката $\Gamma'$. Това означава, че съществува извод в $\Gamma'$ с начална дума $S$ и крайна дума $\omega$. Можем да считаме, че в този извод нито една дума не се повтаря, тъй като, ако има повторение, можем да изхвърлим този „цикъл“ и пак ще получим извод на $\omega$. Но дължината на думите в извода, както вече знаем, не може да намалява, а думите не се повтарят. Следователно дължината на този извод не може да бъде по-голяма от броя на всички възможни думи с дължина $1, 2, \ldots p$, които могат да се построят върху обединението на терминалната и нетерминалната азбуки. Ако в това обединение има например $k$ букви, за броя на различните думи с дължина $1, 2, \ldots p$ получаваме

$$k + k^2 + \ldots + k^p \leqq (k+1)^{p+1}.$$

Това означава, че ако думата $\omega$ с дължина $p$ се извежда в граматиката $\Gamma$, то тя се извежда за не повече от $(k+1)^{p+1}$ стъпки.

Въз основа на тези разсъждения получаваме следния прост алгоритъм за определяне на това дали произволна дума се поражда от дадена контекстна граматика или не.

Нека $\omega$ е произволна дума, а $\Gamma$—произволна контекстна граматика. (Считаме, че $\omega \neq \Lambda$, а в $\Gamma$ няма правило от вида $S \to \Lambda$.)

Последователно строим възможните изводи в $\Gamma$ с брой на стъпките 1, 2, . . . , $(k+1)^{p+1}$, където $p = \partial(\omega)$, а $k$ е броят на символите в обединението на терминалната и нетерминалната азбуки. Това правим така: прилагаме еднократно към началния символ на $\Gamma$ всяко приложимо към него правило — получаваме думите, които се извеждат от началния символ с една стъпка. Към получените думи прилагаме по всички възможни начини еднократно правилата на граматиката и получаваме думите, които се извеждат от началния символ с 2 стъпки и т. н. На всеки етап проверяваме дали сред получените думи не е думата $\omega$. Ако на $m$-тия етап получим, че $\omega$ е сред изведените с $m$ стъпки думи, това означава, че $\omega$ се поражда от $\Gamma$. Ако до $(k+1)^{p+1}$-ия етап не сме намерили $\omega$ сред породените думи, това означава, че $\omega$ не се поражда от граматиката $\Gamma$. Това е така, тъй като, ако думата се поражда от $\Gamma$, тя се поражда за не повече от $(k+1)^{p+1}$ стъпки.

И така, за всяка контекстна граматика $\Gamma$ $L(\Gamma)$ е рекурсивно множество, тъй като посоченият алгоритъм след краен брой стъпки дава отговор на въпроса принадлежи ли произволна дума към езика $L(\Gamma)$ или не.

### Упражнения

1. Постройте автоматна граматика, пораждаща всички думи от нули и единици, които:

а) започват с три единици; б) не започват с три единици; в) имат нечетен брой нули; г) имат нечетен брой нули и нечетен брой единици.

2. Постройте безконтекстна граматика, която:

а) да поражда всички палиндроми върху азбуката $\{a, b\}$;

б) да поражда всички думи от вида $a^n b^m, n \geq 0, m \geq 0, n \neq m$.

3. Постройте контекстна граматика за езика $L = \{1^{3^n}; n \geq 0\}$.

4. Проверете дали думите *babab* и *bbaaa* се пораждат от граматиката $\Gamma = \{a, b\}, \{S\}, S, \{S \to bSS, S \to a\}$).

5. Постройте граматика за езика $L = \{a^n ; n \geq 0\}$.

6. Намерете автоматна граматика за езика $L = \{\Lambda, 001, 100\}$.

7. Докажете, че една граматика е автоматна тогава и само тогава, когато може да се построи еквивалентна на нея граматика, правилата на която да са от вида $A \to Ba$, $A \to a$.




*„А сега ще ти кажа една тайна: аз мога да чета думи от по една буква! Не е ли чудесно това? Но не се отчайвай. И ти ще се научиш да четеш някой ден."*

Луис Карол— *„Алиса в огледалния свят."*

# Глава 2

# АВТОМАТНИ ЕЗИЦИ И КРАЙНИ АВТОМАТИ

## 2.1. СВОЙСТВА НА АВТОМАТНИТЕ ЕЗИЦИ

В първа глава определихме няколко обичайни операции върху формални езици — операциите обединение, сечение, допълнение, произведение, итерация и др. Сега ще изследваме как тези операции действуват върху автоматните езици, т. е. дали обединението, сечението или произведението на два автоматни езика представлява автоматен език и дали допълнението и итерацията на един автоматен език е също автоматен език. На част от тези въпроси ще отговорим веднага. По-късно ще дадем отговор и на останалите въпроси.

И така, какво можем да кажем за обединението на два автоматни езика? Ще покажем, че по граматиките на дадените автоматни езици може да се построи автоматна граматика, която да поражда обединението на двата езика или, с други думи — че **обединението на автоматни езици е също автоматен език.**

Нека $L_1$ и $L_2$ са два дадени автоматни езика. Това означава, че съществуват автоматни граматики $\Gamma_1$ и $\Gamma_2$, които пораждат $L_1$ и $L_2$: $L_1 = L(\Gamma_1)$ и $L_2 = L(L_2)$.

Нека $\Gamma_1 = (V', W', S', P')$, а $\Gamma_2 = (V'', W'', S'', P'')$.

Ще приемем, че сечението на азбуките $W'$ и $W''$ е празно и никоя от тях не съдържа символа $S$. Това винаги може да се получи с подходящо означаване на нетерминалните символи от $W'$ и $W''$, които не участвуват, както е известно, в думите от $L_1$ и $L_2$.

Образуваме следната граматика:

$$\mathbf{G}=(V'\cup V'',\ W'\cup W''\cup\{S\},\ S,\ P),$$

където $P$ се състои от:

— всички правила от $P'$ и $P''$ с изключение на правилата $S'\to\Lambda$ и $S''\to\Lambda$, ако такива има;

— нови правила $S\to\omega$, добавени за всяко правило от вида $S'\to\omega$ от $P'$ или $S''\to\omega$ от $P''$ $(\omega\neq\Lambda)$;

— правило $S\to\Lambda$, ако в $P'$ или $P''$ има подобно правило.

Очевидно **G** е автоматна граматика, тъй като не сме променяли вида на правилата от $P'$ и $P''$, а $S$ не се среща отдясно на никое правило, така че можем да добавяме правилото $S\to\Lambda$.

Ще покажем, че $L(\mathbf{G})=L(\Gamma_1)\cup L(\Gamma_2)$.

Нека $\alpha$ е произволна дума от $L(\mathbf{G})$. Ако $\alpha$ е празната дума, в **G** има правило $S\to\Lambda$, а такова правило сме добавили към **G** само в случая, когато в $\Gamma_1$ или $\Gamma_2$ има правило $S'\to\Lambda$, или $S''\to\Lambda$, т. е. когато $\Lambda$ е дума от $L(\Gamma_1)$ или $L(\Gamma_2)$. Но $\Lambda$ ще е дума и от $L(\Gamma_1)\cup L(\Gamma_2)$. Ако $\alpha$ не е празната дума, тогава в **G** има извод

$$S\vdash\omega\vdash\ldots\vdash\alpha.$$

Първото правило от извода е $S\to\omega$, а това означава, че или в $\Gamma_1$, или в $\Gamma_2$ има правило $S'\to\omega$ или съответно, $S''\to\omega$. Тогава $\omega$ е дума или от $V'\cup W'$, или от $V''\cup W''$ и към нея се прилагат само правила съответно или от $P'$, или от $P''$ (азбуките $W'$ и $W''$ имат празно сечение). Следователно или $\omega\overset{\Gamma_1}{\models}\alpha$, или $\omega\overset{\Gamma_2}{\models}\alpha$. Но тогава, пред вид $S'\overset{\Gamma_1}{\vdash}\omega$ или $S''\overset{\Gamma_2}{\vdash}\omega$, получаваме или $S'\overset{\Gamma_1}{\models}\alpha$, или $S''\overset{\Gamma_2}{\models}\alpha$, т. е. $\alpha$ е дума или от езика $L(\Gamma_1)$, или от езика $L(\Gamma_2)$. В такъв случай тя е дума и от тяхното обединение.

Обратно, нека $\alpha$ е произволна дума от $L(\Gamma_1)$ или $L(\Gamma_2)$, например от $L(\Gamma_1)$. Ако $\alpha$ е празната дума, в $\Gamma_1$ ще има правило $S'\to\Lambda$, но тогава и в **G** ще има правило $S\to\Lambda$, т. е. празната дума ще се поражда от **G**. Ако $\alpha$ не е празната дума, в $\Gamma_1$ има извод:

$$S'\vdash\omega\vdash\ldots\vdash\alpha.$$



Но тогава в граматиката **G** има правило $S\to\omega$, а $\omega\overset{\mathbf{G}}{\models}\alpha$, тъй като всички правила от граматиката $\Gamma_1$ (без $S'\to\Lambda$) са правила и в **G**. Получаваме, че $S\overset{\mathbf{G}}{\models}\alpha$, т. е. думата $\alpha$ се поражда и от граматиката **G**.

**И така, $L(\mathbf{G})=L(\Gamma_1)\cup L(\Gamma_2)$.**

Да вземем, например автоматните езици $L_1=\{1^{2n};\ n\geqq 0\}$ и $L_2=\{1^{2n+1};\ n\geqq 0\}$. Тяхното обединение е езикът $L=\{1^n;\ n\geqq 0\}=\{\Lambda,\ 1,11,\ldots\}$. $L_1$ можем да зададем с граматиката

$$\Gamma_1=(\{1\},\ S',\ A',\ B'\},\ S',\ \{S\to 1A',\ A'\to 1B',\ B'\to 1A',\ A'\to 1,S'\to\Lambda\}).$$

Езикът $L_2$ можем да зададем с граматиката

$$\Gamma_2=(\{1\},\{S'',A'',B'',\},\ S',\ \{S'\to 1A'',\ S''\to 1,\ A''\to 1B'',B''\to 1A'',\ B''\to 1\}).$$

Построяваме автоматната граматика **G** за езика $L=L_1\cup L_2$. Съгласно изложения в доказателството общ метод получаваме.

$$\mathbf{G}=(\{1\},\{S,\ S',\ S'',\ A',\ A'',\ B',\ B''\},\ S,\ \{S'\to 1A',\ A'\to 1B',$$
$$B'\to 1A',\ A'\to 1,\ S''\to 1A'',\ S''\to 1,\ A''\to 1B'',\ B''\to 1A'',\ B''\to 1,$$
$$S\to 1A',\ S\to 1A'',\ S\to 1,\ S\to\Lambda\}.$$

Разбира се, в дадения случай за езика $L$ бихме могли да построим направо значително по-проста автоматна граматика.

Автоматните езици могат да бъдат както крайни, така и безкрайни. Например безкрайни са автоматните езици $L_1$, $L_2$ и $L$ от предишния пример, а краен е автоматният език $\{\Lambda,\ a,\ a^2\}$ с граматика $(\{a\},\ \{S,\ A\},\ S,\ \{S\to a,\ S\to aA,\ A\to a,\ S\to\Lambda\})$. Може ли обаче да има крайни езици, които да не са автоматни? Отговорът на този въпрос е отрицателен — **за всеки краен език може да се построи автоматна граматика, която да го поражда.** Празният език например се поражда от автоматната граматика $(\{a\},\ \{S\},\ S,\ \{S\to aS\})_1$ а езикът, състоящ се само от празната дума — от граматиката $(\{a\},\ \{S\},\ S,\ \{S\to\Lambda\})$.

Да разгледаме отначало езиците, съдържащи точно една непразна дума. Нека езикът $L$ се състои от думата $a_1a_2\ldots a_k$ върху азбуката $\{a_1,\ a_2,\ldots a_k\}$, т. е. $L=\{a_1a_2\ldots a_k\}$. Тогава автоматната граматика $\Gamma=(\{a_1,a_2,..,a_k\},\ \{S,A_1,\ldots A_{k-1}\},\ S,\{S\to a_1A_1,\ A_1\to a_2A_2,\ldots,\ A_{k-2}\to a_{k-1}A_{k-1},\ A_{k-1}\to a_k\})$ поражда $L$, тъй като единствената дума, която може да се породи от $\Gamma$, е $a_1a_2\ldots a_k$:

$$S\vdash a_1A_1\vdash a_1a_2A_2\vdash\ldots\vdash a_1a_2\ldots a_{k-1}A_{k-1}\to a_1a_2\ldots a_{k-1}a_k.$$

Действително, към $S$ може да се приложи само първото прави-

ло, към получената дума може да се приложи само второто правило и т. н.

Всеки краен език можем да разглеждаме като обединение на краен брой езици, съдържащи по една дума. Но тогава за всеки от тях можем да намерим автоматна граматика, а по тези граматики можем да построим автоматната граматика на обединението на автоматните езици.

**Произведението на два автоматни езика е също автоматен език.** Това ще покажем, като по граматиките на двата дадени езика построим автоматната граматика на произведението им.

Нека отначало $L_1$ и $L_2$ са два езика, несъдържащи празната дума, пораждани съответно от автоматните граматики

$$\Gamma'=(V'.\ W',\ S',\ P') \text{ и } \Gamma''=(V'',\ W'',\ S'',\ P'').$$

Както и преди, ще считаме, че сечението на нетерминалните азбуки $W'$ и $W''$ е празно. Да построим граматиката $\Gamma=(V'\cup V'',\ W'\cup W'',\ S',\ P)$, като $P$ се състои от следните правила:

— всички правила от $P'$ и $P''$ с изключение на правилата от $P'$, които имат вида $A'\to a$ $(A'\in W',\ a\in V')$;

— нови правила $A'\to aS''$, за всяко правило от $P'$, което има вида $A'\to a$ $(A'\in W',\ a\in V')$.

Граматиката Г е автоматна, тъй като всички нови правила имат необходимия вид. Ако граматиката Г′ поражда дума $\omega$, то в новата граматика Г от началния символ ще се извежда думата $\omega S'$, а следователно и всяка конкатенация на думата $\omega$ с дума, пораждана от Г″.

Следователно Г поражда всички думи от вида $\alpha\beta$, $\alpha\in L(\Gamma')$ $\beta\in L(\Gamma'')$, т. е.

$$L(\Gamma)=L(\Gamma')L(\Gamma'')=L_1L_2.$$

Ако само един от двата езика съдържа празната дума, например $L_1$, образуваме автоматната граматика $\overline{\Gamma'}$, пораждаща $L_1-\{\Lambda\}$ (чрез премахване на правилото $S\to\Lambda$). Тогава произведението $L_1L_2$ ще получим, като обединим два автоматни езика: произведението $L(\overline{\Gamma'})L_2$ и $L_2$. Това е така, защото към $L(\Gamma')L_2$ трябва да добавим конкатенациите на празната дума $\Lambda$ с думите от $L_2$, т. е. езика $L_2$.

Ако и двата езика съдържат празната дума, тогава образуваме граматиките $\overline{\Gamma'}$ и $\overline{\Gamma''}$ за езиците $L_1-\{\Lambda\}$ и $L_2-\{\Lambda\}$.

Произведението на $L_1$ и $L_2$ ще се състои от обединението на четири автоматни езика: $L(\overline{\Gamma})'$ $L(\overline{\Gamma''})$, $L_1$, $L_2$ и $\{\Lambda\}$.



Да образуваме произведението на езиците $L_1=\{0^n;\ n\geq 1\}$ и $L_2=\{1^m;\ m\geq 1\}$. Получаваме $L_1L_2=\{0^n1^m;\ n\geq 1,\ m\geq 1\}$. Езиците $L_1$ и $L_2$ се пораждат например от автоматните граматики:

$\Gamma'=(\{0\},\{S'\},\ S',\ \{S'\to 0S',\ S'\to 0\})$;
$\Gamma''=(\{1\},\ \{S''\},\ S'',\ \{S''\to 1S'',\ S''\to 1\})$.

Тогава автоматната граматика $\Gamma=(\{0,1\},\ \{S',\ S''\},\ S',\ \{S'\to 0S',\ S'\to 0S'',\ S''\to 1S'',\ S''\to 1\})$ поражда произведението $L_1L_2$.

По този начин лесно можем да докажем, че и **итерацията на един автоматен език е също автоматен език.** Действително, нека езикът $L$ се поражда от автоматната граматика $\Gamma=(V,W,\ S,\ P)$. От правилата на Г премахваме правилото $S\to\Lambda$ (ако такова има), а за всяко правило от вида $A\to a$ (без $S\to\Lambda$) добавяме ново правило $A\to aS$. Това позволява всеки път, когато дадената граматика приключва извеждането на терминалната дума, прилагайки правило от вида $A\to a$, новата автоматна граматика или също да приключи извеждането на думата чрез същото правило, или да приложи правило $A\to aS$ и по този начин да постави начало на извеждане на нова дума от Г, долепена отдясно на изведената. Като повтаря тази процедура произволен брой пъти, новата граматика може да поражда конкатенации на произволен брой непразни думи от $L$. А това означава, че тя поражда езика $L^*-\{\Lambda\}$. Но тогава итерацията $L^*$ на езика $L$ е обединение на два автоматни езика $L^*-\{\Lambda\}$ и $\{\Lambda\}$, т. е. също е автоматен език.

Да намерим по горния начин автоматната граматика за езика $L=\{(101)^n,\ n\geq 1\}$. Този език представлява итерацията на езика $L_1=\{101\}$ без празната дума, т. е. $L=L_1^*-\{\Lambda\}$. За $L_1$ лесно можем да намерим пораждащата го автоматна граматика — например $(\{0,1\},\ \{S,\ A,\ B\},\ S,\ \{S\to 1A,\ A\to 0B,\ B\to 1\}$ Тогава езикът $L$ ще се поражда от автоматната граматика

$$(\{0,1\},\{S,\ A,\ B\},S\{S\to 1A,\ A\to 0B,\ B\to 1,\ B\to 1S\}).$$

Дотук доказахме, че празният език $\varnothing$, езикът $\{\Lambda\}$, съставен от празната дума, и крайните езици са автоматни, а техните обединения, произведения и итерации също представляват автоматни езици. След като въведем крайните автомати, ще докажем и обратното: всеки автоматен език може да се получи от $\varnothing$, $\{\Lambda\}$ и крайните езици с помощта на операциите обединение, произведение и итерация.

## Упражнения

1. Намерете автоматна граматика, която да поражда всички думи от 0 и 1, които имат четен брой 0 или нечетен брой 1.

2. Намерете автоматна граматика, която да поражда целите неотрицателни степени на думите *aba*, *abba* и *abbba*.

3. Постройте автоматна граматика за езика

$$L=\{a^k b^l c^m,\ k\geq 0,\ l\geq 0,\ m\geq 0\}.$$

4. Постройте автоматна граматика за езика

$$L=\{(101)^n (1001)^m, n\geq 0,\ m\geq 1\}.$$

## 2.2. КРАЙНИ АВТОМАТИ

Автоматните езици определихме чрез вида на техните граматики т. е. от гледна точка на процеса на пораждането им. От това представяне получихме някои прости техни свойства. Редица други свойства на автоматните езици е удобно да изследваме от гледна точка на разпознаването им, т. е., като ги зададем чрез съответните им разпознаватели. Прието е разпознавателите обикновено да се наричат **автомати,** така че теорията на автоматите и теорията на формалните езици представляват две математически теории за изследване на един и същ обект — формалните езици. В този смисъл всяко достатъчно пълно изложение на теорията на формалните езици и математическата лингвистика изисква паралелно изследване на класовете езици чрез формални граматики и чрез разпознаватели — автомати.

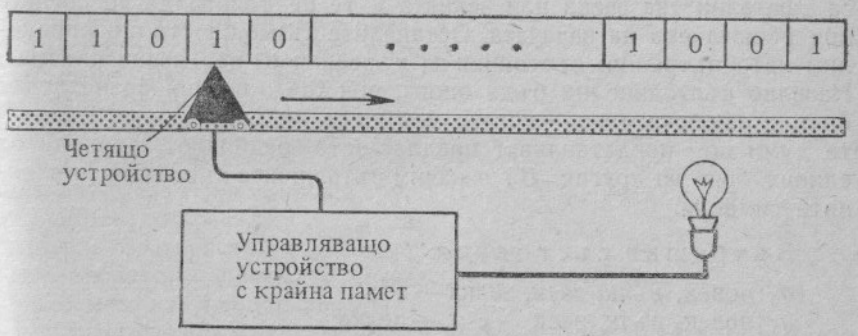
Тук ще въведем два нови разпознавателя — **детерминирани крайни автомати** и **недетерминирани крайни автомати.** За тях ще докажем, че разпознават точно класа на автоматните езици. (С това автоматните езици ще оправдаят името си, макар че по-точно би било в такъв случай да ги наричаме крайно-автоматни.) С помощта на крайните автомати ще получим редица основни свойства и характеристики на автоматните езици.

Крайните автомати представляват сравнително прости математически модели на устройства за разпознаване (или превод) на формалните езици. Обикновено един **краен автомат** се състои от **управляващо устройство с крайна памет**, което има краен брой **вътрешни състояния** и от **входна лента**, на която е записана крайна редица от **входни символи**, четени от ляво на дясно, т. е. дума от входни символи. Допустимите входни символи образуват **входната азбука на автомата.**



Крайният автомат работи в дискретни моменти от време — тактове. Във всеки такъв момент управляващото устройство се намира точно в едно от вътрешните си състояния и прочита един от символите, записани на входната лента. В началния момент от време управляващото устройство се намира в едно фиксирано за автомата вътрешно състояние, наречено **начално** и прочита най-левия символ, записан върху входната лента. При всеки такт, в зависимост от вътрешното си състояние и прочетения входен символ, управляващото устройство преминава в едно от вътрешните си състояния и се придвижва към следващия отдясно входен символ. Крайният автомат спира да работи, когато изчерпи всички входни символи, записани върху лентата. Ако крайният автомат спре преди това, то той не е определен за зададената дума. Резултат от работата на крайния автомат е вътрешното състояние, в което той се намира след изчитане на цялата входна дума. Ако това състояние е едно от фиксираните за автомата вътрешни състояния, наречени **заключителни,** зададената редица от входни символи — входната дума — се **разпознава** (или още **допуска**) от автомата. В противен случай (включително и когато не е определен) крайният автомат не разпознава зададената входна дума.

Схематично крайният автомат можем да представим по следния начин (фиг. 11):




Фиг. 11

И така, крайният автомат е определен, ако са зададени крайно множество от вътрешни състояния, крайно множество от входни символи, начално вътрешно състояние, множество на заключителните състояния и **функция на преходите (програма на автомата),** която по вътрешното състояние и прочетения входен

символ за дадения момент определя вътрешното състояние за следващия момент.

Преди да дадем формално определение за краен автомат, ще разгледаме следния занимателен пример, даден в книгата J. Hopcroft, J. Ullman„Introduction to Automata Theory, Languages and Computation".

Човек, вълк, заек и зелка се намират на левия бряг на една река. Човекът разполага с лодка, с която може да пренася на другия бряг себе си и единствено още или вълка, или заека, или зелката. Човекът желае да премине на десния бряг заедно с вълка, заека и зелката. Ако обаче той остави сами вълка и заека — вълкът ще изяде заека. Ако остави заека сам със зелката, заекът също ще я изяде. Възможно ли е човекът да премине на другия бряг с вълка, заека и зелката?

Ще построим краен автомат, който да разпознава дали предлаганото решение на задачата е правилно или не.

Има 16 възможности за разпределяне на човека, вълка, заека и зелката върху двата бряга на реката. Шест от тези възможности

| | | | |
|---|---|---|---|
| вълк, заек, зелка | — човек | човек, зелка | — вълк, заек |
| човек | — вълк, заек, зелка | заек, зелка | — човек, вълк |
| човек, вълк | — заек, зелка | вълк, заек | — човек, зелка |

са „фатални" за заека или зелката и те не трябва да се срещат при решаването на задачата. Останалите възможности ще определим като вътрешни състояния на търсения от нас краен автомат. Начално състояние ще бъде онова, при което всички са на левия бряг, а заключително — когато всички са на десния бряг. Входните думи ще представляват предлаганата редица от преходи от единия бряг на другия. Да означим вътрешните състояния и входните символи:

Вътрешни състояния:

$q_0$: човек, вълк, заек, зелка — $\varnothing$;
$q_1$: човек, вълк, заек — зелка;
$q_2$: човек, вълк, зелка — заек;
$q_3$: човек, заек, зелка — вълк;
$q_4$: човек, заек — вълк, зелка;
$q_5$: вълк, зелка — човек, заек;
$q_6$: зелка — човек, вълк, заек;
$q_7$: заек — човек, вълк, зелка;
$q_8$: вълк — човек, заек, зелка;
$q_9$: $\varnothing$ — човек, вълк, заек, зелка.



Входни символи:

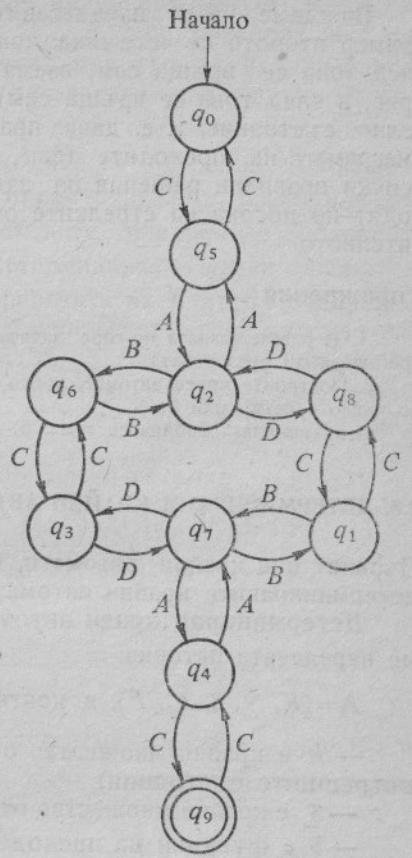
$A$: човекът минава реката сам;

$B$: човекът минава реката с вълка;

$C$: човекът минава реката със заека;

$D$: човекът минава реката със зелката.

Автомата ще изобразим графично по следния начин: вътрешните състояния ще означаваме с кръгчета, в които е означено състоянието. Заключителното състояние ще означаваме с двойно кръгче, а началното — с входна стрелка. Със стрелки ще означаваме възможните преходи от едно състояние в друго, като на всяка стрелка отбелязваме чрез кой входен символ се извършва този преход. Получаваме диаграмата, дадена на фиг. 12, която ще наричаме по-нататък **диаграма на преходите**. Чрез тази диаграма крайният автомат ще проверява правилността на предлаганото решение.



Фиг. 12

От едно вътрешно състояние автоматът преминава в друго, ако в диаграмата му има стрелка от първото състояние към второто, означена с четения в момента входен символ.

Да проверим как този краен автомат действува върху дадени входни думи (т. е. предлагани решения на задачата). За думите:

*CABBD, CADA, CABCDAC*

получаваме следните редици от състояния:

$q_0, q_4, q_2, q_6, q_2, q_8$;

$q_0, q_4, q_3,$ — неопределено

$q_0, q_4, q_2, q_6, q_3, q_7, q_4, q_9$.

Виждаме, че от предлаганите за проверка три решения (например второто се чете така: човекът преминава реката със заека, след това се връща сам, взема зелката и я занася на другия бряг, а след това се връща сам) само третото води до заключително състояние, т. е. дава правилно решение на задачата. От диаграмата на преходите (фиг. 12) лесно могат да се намерят всички правилни решения на задачата: това са тези думи, които водят по посока на стрелките от началното състояние до заключителното.

### Упражнения

1. В разглежданата по-горе задача намерете всички решения с най-малък брой преходи през реката.
2. Постройте краен автомат, който да разпознава, дали произволно естествено число е четно или не:
а) в двоичната бройна система; б) в десетичната бройна система.

## 2.3. ДЕТЕРМИНИРАНИ КРАЙНИ АВТОМАТИ

Първият вид крайни автомати, които ще въведем и обсъдим, са детерминираните крайни автомати.

**Детерминиран краен автомат** върху азбуката $\Sigma$ ще наричаме наредената петорка:

$\mathbf{A}=(K, \Sigma, \delta, q_0, F)$, в която

— $K$ е крайно множество от вътрешни състояния **(азбука на вътрешните състояния)**;

— $\Sigma$ е крайно множество от входни символи **(входна азбука)**;

— $\delta$ е **функция на преходите**, която на наредените двойки от вида („вътрешно състояние", „входен символ") съпоставя „вътрешно състояние";

— $q_0 \in K$ е **начално състояние**;

— $F \subset K$ е **множество на заключителните състояния**.

Ако функцията на преходите $\delta$ е определена за всички значения от дефиниционната област, т. е. за всички възможни двойки („вътрешно състояние", „входен символ"), автоматът **A** се нарича **напълно определен**.

Възможно е детерминираният краен автомат да не е определен за някои значения от дефиниционната област на функцията $\delta$, т. е. за някои двойки („вътрешно състояние", „входен символ") да няма определено следващо вътрешно състояние.



**Работата на детерминирания краен автомат A** върху зададена входна дума $a_{i_1} a_{i_2} \ldots a_{i_{k+1}} \in \Sigma^*$ се определя по следния начин: По началното състояние $q_0$ и първия входен символ $a_{i_1}$ чрез функцията на преходите се определя следващото състояние $\delta(q_0, a_{i_1})=p_1$. По състоянието $p_1$ и следващия входен символ $a_{i_2}$ чрез функцията на преходите се определя следващото състояние $\delta(p_1, a_{i_2})=p_2$ и т. н. По състоянието $p_k$ и входния символ $a_{i_{k+1}}$ чрез функцията на преходите се определя последното състояние $\delta(p_k, a_{i_{k+1}})=p_{k+1}$ на детерминирания краен автомат **A**, което представлява резултат от работата на **A** върху зададената входна дума. Ако $p_{k+1} \in F$, детерминираният краен автомат разпознава входната дума $a_{i_1} a_{i_2} \ldots a_{i_{k+1}}$. Ако $p_{k+1} \notin F$ или за някоя двойка $(p_i, a_{i+1})$ от вътрешно състояние $p_i$ и входен символ $a_{i+1}$ функцията на преходите не е определена, **A** не разпознава входната дума.

Да разгледаме детерминирания краен автомат

$$\mathbf{A}_1=(\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_0, q_1\}).$$

Функцията на преходите на $\mathbf{A}_1$ е следната:

| | |
|---|---|
| $\delta(q_0, 0)=q_1$ | $\delta(q_1,1)=q_0$ |
| $\delta(q_0, 1)=q_0$ | $\delta(q_2, 0)=q_2$ |
| $\delta(q_1, 0)=q_2$ | $\delta(q^2, 1)=q_2$. |

За входната дума 100110 получаваме следната поредица от състояния на $\mathbf{A}_1$:

$$\begin{array}{cccccccc} & 1 & 0 & 0 & 1 & 1 & 0 & \\ q_0 & q_0 & q_1 & q_2 & q_2 & q_2 & q_3, \end{array}$$

а за входната дума 0101101:

$$\begin{array}{cccccccc} & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ q_0 & q_1 & q_0 & q_1 & q_0 & q_0 & q_1 & q_0. \end{array}$$

Следователно, детерминираният краен автомат **A** разпознава думата 0101101, тъй като $q_0$ е заключително състояние, но не разпознава думата 100110, понеже $q_2$ не е заключително състояние.

Друг пример на детерминиран краен автомат е автоматът

$\mathbf{A}_2=(\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$ с функция на преходите

$\delta(q_0, a)=q_1$, $\delta(q_1, a)=q_1$, $\delta(q_1, b)=q_2$, $\delta(q_2, a)=q_1$ $\delta(q_2, b)=q_2$.

Лесно можем да се убедим, че $A_2$ разпознава например думите *ab*, *aabb*, но не разпознава *baa* или *abab*. Ще отбележим, че крайният детерминиран автомат $A_2$ не е напълно определен.

Крайното състояние на автомата $A_1$ за входните думи 100110 и 0101101 може да се запише чрез функцията на преходите по следния начин:

$$\delta(\delta(\delta(\delta(\delta(\delta(q_0, 1), 0), 0), 1), 1), 0)$$

и

$$\delta(\delta(\delta(\delta(\delta(\delta(q_0, 0), 1), 0), 1), 1), 0), 1),$$

което е трудно обозримо. Затова често пъти вместо функцията на преходите $\delta$, която на дадено състояние и входен символ съпоставя състояние, е по-удобно да се използува функцията $\delta'$, която на дадено състояние и входна дума съпоставя състояние.

За произволен детерминиран краен автомат $A=(K, \Sigma, \delta, q_0, F)$ функцията $\delta'$ се определя индуктивно по следния начин:

$\delta'(q, \Lambda)=q$ за всяко $q \in K$; ($\Lambda$ е празната дума);

$\delta'(q, \omega a)=\delta(\delta'(q, \omega), a)$ за всяка дума $\omega \in \Sigma^*$ и за всяка буква $a \in \Sigma$.

Първото правило показва, че **A** не може да промени състоянието си без прочитане на входен символ, а второто правило определя $\delta'$ чрез $\delta$.

В такъв случай $\delta'(q, \alpha)=p$ означава, че детерминираният краен автомат **A** от състояние $q$ преминава в състояние $p$ след прочитане на входната дума $\alpha$.

За функцията $\delta'$, приложена към отделен входен символ, получаваме

$$\delta'(q, a)=\delta(\delta'(q, \Lambda), a)=\delta(q, a),$$

което показва, че функциите $\delta'$ и $\delta$ съвпадат за онези значения на аргументите, за които и двете едновременно са определени. Това позволява по-нататък с един и същ знак $\delta$ да означаваме както функцията $\delta$, така и функцията $\delta'$.

За автомата $A_1$, приложен към думите 100110 и 0101101, получаваме следните записи чрез новата функция $\delta$:

$$\delta(q_0, 100110)=q_2, \text{ а } \delta(q_0, 0101101)=q_0.$$

Както вече казахме, детерминираният краен автомат $A=(K, \Sigma, \delta, q_0, F)$ разпознава думата $\alpha$, $\alpha \in \Sigma^*$, ако $\delta(q_0, \alpha)$ е заключително състояние, т. е. ако $\delta(q_0, \alpha) \in F$.



Множеството $T(A)$ на всички думи от $\Sigma^*$, разпознавани от детерминирания краен автомат **A**, се нарича **език, разпознаван от A**. И така,

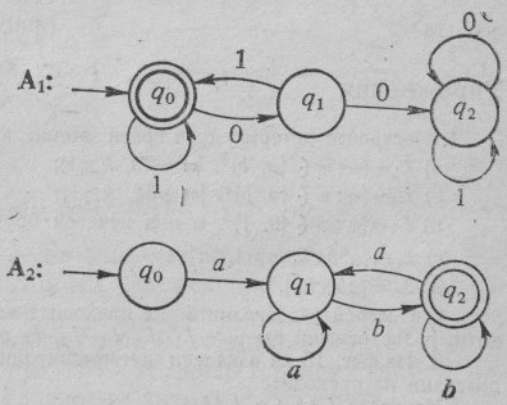
$$T(A)=\{\alpha;\ \alpha \in \Sigma^*,\ \delta(q_0, \alpha) \in F\}.$$

Два детерминирани крайни автомати $A_1$ и $A_2$ са **еквивалентни**, ако $T(A_1)=T(A_2)$, т. е. ако разпознават един и същ език.

На всеки детерминиран краен автомат **A** може да се съпостави ориентиран граф, наречен **диаграма на преходите**, по начина, който приложихме в задачата за човека, вълка, заека и зелката.

Диаграмата на преходите за **A** се получава, като на всяко състояние се съпостави връх, означен с това състояние, а на всяко равенство $\delta(q, a)=p$, $a \in \Sigma$, $p, q \in K$ от функцията на преходите се съпостави ориентирано ребро от върха $q$ към върха $p$, означено с $a$. Върхът, съответствуващ на началното състояние, се отбелязва със стрелка, насочена към този връх, а върховете, съответствуващи на заключителни състояния, са отбелязани с някакъв знак, например могат да бъдат двойно заградени. Думата $\alpha$ се разпознава от автомата **A**, ако в диаграмата на преходите има път от ребра, отбелязани последователно със символите на $\alpha$, който започва от върха на началното състояние и завършва с връх, съответствуващ на заключително състояние. Естествено пътят е по посока на стрелките. Понякога няколко ребра от един връх към друг могат да се отбелязват само с едно ребро, на което са означени входните символи на всяко от ребрата.

Фиг. 13



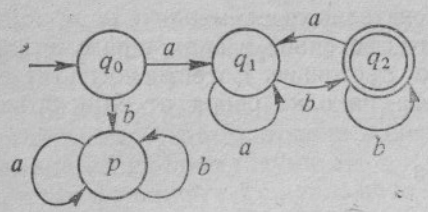

За автоматите $A_1$ и $A_2$ получаваме следните диаграми на преходите (фиг. 13):

От диаграмата на преходите лесно може да се забележи, че автоматът $A_1$ разпознава всички думи от 0 и 1, в които няма две последователни нули: $T(A_1)=\{\omega : \omega \in \{0,1\}^*$ и в $\omega$ няма две последователни нули$\}$.

А автоматът $\mathbf{A}_2$ разпознава всички непразни думи, започващи с $a$ и завършващи с $b$:

$$T(\mathbf{A}_2)=\{\omega : \omega \in \{a, b\}^* \text{ и } \omega = a\alpha b\}.$$

Детерминираният краен автомат $\mathbf{A}_1$ е напълно определен, а $\mathbf{A}_2$ — не е напълно определен, например състоянието $\delta(q_0, b)$ не е определено. Лесно може да се докаже обаче, че за всеки неопределен напълно детерминиран краен автомат $\mathbf{B}_1$ може да се построи еквивалентен на него, напълно определен детерминиран краен автомат $\mathbf{B}_2$. За целта към вътрешните състояния на $\mathbf{B}_1$ се добавя ново незаключително вътрешно състояние $p$, а към функцията на преходите му — равенство $\delta(q, a)=p$ за всяко неопределено значение на функцията на преходите на $\mathbf{B}_1$. Освен това за всеки входен символ $a$ определяме $\delta(p, a)=p$. Ясно е, че новият детерминиран краен автомат $\mathbf{B}_2$ е вече напълно определен. $\mathbf{B}_2$ ще бъде еквивалентен на $\mathbf{B}_1$, защото върху всички входни думи, за които автоматът $\mathbf{B}_1$ е определен, автоматът $\mathbf{B}_2$ ще действува по същия начин както $\mathbf{B}_1$, а за думите, за които $\mathbf{B}_1$ не е определен, $\mathbf{B}_2$ ще попада в състоянието $p$, в което ще остава до края на думата. И тъй като $p$ не е заключително състояние, множеството на разпознаваните от автомата $\mathbf{B}_2$ думи ще бъде същото както множеството на разпознаваните от автомата $\mathbf{B}_1$ думи.




фиг. 14

За детерминирания краен автомат $\mathbf{A}_2$ получаваме следния, напълно определен и еквивалентен на него краен автомат (фиг. 14):

## Упражнения

**1.** Постройте детерминиран краен автомат, който да разпознава следния език:

а) $L_1=\{\omega : \omega \in \{a, b\}^*, |\omega|=5k, k\geq 1\}$;

б) $L_2=\{\omega : \omega \in \{a, b\}^*, |\omega|\neq 5k, k\geq 1\}$;

в) $L_3=\{\omega : \omega \in \{0, 1\}^*$, $\omega$ има нечетен брой 0 и нечетен брой 1$\}$;

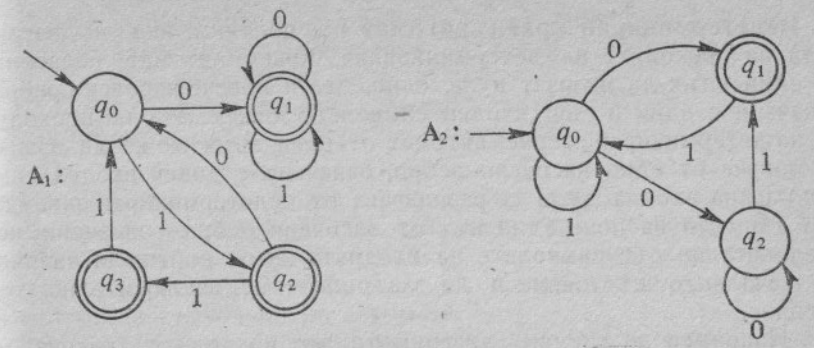
г) $L_4=\{a^n b a^m, n\geq 1, m\geq 1\}$;

д) $L_5=\{abccba\}$.

**2.** Постройте диаграмите на преходите на детерминираните крайни автомати, разпознаващи езиците $L_1$, $L_2$, $L_3$, $L_4$, $L_5$ от предишната задача.

**3.** На фиг. 15 са зададени детерминираните крайни автомати $\mathbf{A}_1$ и $\mathbf{A}_2$ с диаграми на преходите.

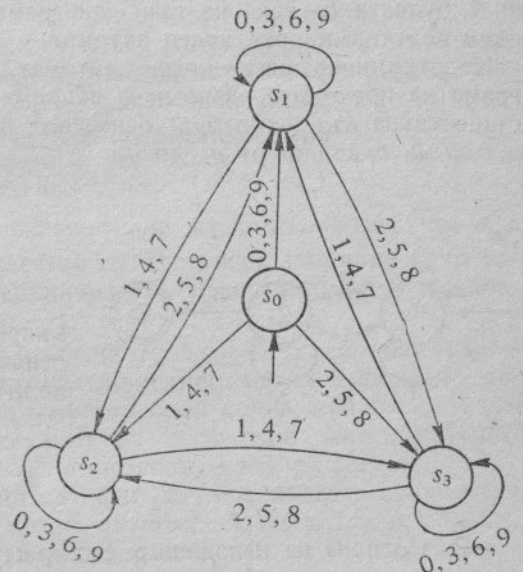
Намерете $L(\mathbf{A}_1)$ и $L(\mathbf{A}_2)$.




Фиг. 15

**4.** Нека $\delta$ е функция на преходите на произволен детерминиран краен автомат. Докажете, че за произволни входни думи $\alpha$ и $\beta$ $\delta(q, \alpha\beta)=\delta(\delta(q, \alpha), \beta)$.

**5.** Докажете, че детерминираният краен автомат, определен с диаграма на преходите от фиг. 16, разпознава десетичните записи на естествените числа, които се делят на 3.




Фиг. 16

## 2.4. НЕДЕТЕРМИНИРАНИ КРАЙНИ АВТОМАТИ

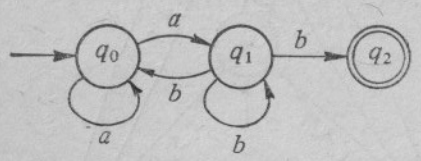
Друг малко по-различен, но не по-общ вид крайни автомати са недетерминираните крайни автомати. Въпреки че детерминираните и недетерминираните крайни автомати разпознават един и същ клас формални езици, въвеждането на недетерминираните крайни автомати е целесъобразно, тъй като те представляват удобен математически апарат за изследване на автоматните езици.

**Недетерминиран краен автомат** ще получим, ако в диаграмата на преходите на детерминирания краен автомат позволим от един връх да излизат нула, едно, две и повече насочени ребра, означени с един и същ входен символ. (В диаграмата на преходите на детерминирания краен автомат от един връх може да излиза не повече от едно насочено ребро, означено с даден входен символ.) Една входна дума се разпознава от недетерминиран автомат, ако съществува поне един път от насочени ребра, означени последователно със символите на входната дума, който да започва от началното състояние и да завършва със заключително състояние.

Например да вземем диаграмата на преходите, дадена на фиг. 17. От върха $q_0$ излизат две насочени ребра, означени с една и съща буква $a$, от върха $q_1$ — три насочени ребра, означени с буквата $b$, така че тази диаграма на преходите представя един недетерминиран краен автомат.

Недетерминираният краен автомат, определен от тази диаграма на преходите, разпознава например думата *aabbb*, тъй като съществува път от ребра, означени последователно с буквите $a, a, b, b, b$, водещ от $q_0$ до $q_2$:

$$\begin{matrix} & a & a & b & b & b & \\ q_0 & q_0 & q_1 & q_1 & q_1 & q_2, \end{matrix}$$




Фиг. 17

въпреки че съществува и път, означен с тези букви, който не води от $q_0$ до $q_2$:

$$\begin{matrix} & a & a & b & b & b & \\ q_0 & q_0 & q_1 & q_1 & q_1 & q_1. \end{matrix}$$

Въз основа на изложените съдържателни съображения можем да дадем следното определение:

**Недетерминиран краен автомат** $\mathbf{A}$ върху азбуката $\Sigma$ ще наричаме наредената петорка

$$\mathbf{A}=(K, \Sigma, \delta, q_0, F),$$

в която:

— $K$ е азбука на вътрешните състояния;
— $\Sigma$ е азбука на входните символи;



— $\delta$ е функция на преходите с дефиниционна област и област на значенията — множеството $P(K)$ на всички подмножества на $K$;
— $q_0 \in K$ е начално състояние;
— $F \subset K$ е множество на заключителните състояния.

Недетерминираният краен автомат се различава от детерминирания само по функцията на преходите; докато при детерминирания краен автомат $\delta(q, a)$ е едно вътрешно състояние, то при недетерминирания краен автомат $\delta(q, a)$ е крайно множество от състояния, т. е. $\delta(q,a)=\{p_1, p_2, \ldots, p_n\}$, където $p_1, p_2, \ldots, p_n$ са вътрешни състояния на автомата.

В съответствие с това работата на недетерминирания краен автомат върху входната дума $a_{i_1} \ldots a_{i_k}$ изглежда така: По началното състояние $q_0$ и входния символ $a_{i_1}$ функцията на преходите определя множеството на следващите състояния $\delta(q_0, a_{i_1}) = \{p_1, p_2, \ldots, p_n\}$. За всяко от тези състояния $p_j$ и за следващия входен символ $a_{i_2}$ се определя множеството на следващите състояния $\delta(p_j, a_{i_2})=\{r_{j_1}, \ldots, r_{j_s}\}$, като множеството на тези следващи състояния на автомата $\mathbf{A}$ е обединението им:

$$\delta(p_1, a_{i_2}) \cup \delta(p_2, a_{i_2}) \cup \ldots \cup \delta(p_n, a_{i_2}).$$

Този процес продължава включително до последната буква $a_{i_k}$. Ако след полученото множество от състояния има поне едно заключително, тогава недетерминираният краен автомат $\mathbf{A}$ разпознава думата $a_{i_1} a_{i_2} \ldots a_{i_k}$.

Тази множественост на следващите вътрешни състояния можем да си представим като многократно самовъзпроизвеждане на автомата при всеки такт на съответен брой копия, които са идентични помежду си и работят нататък независимо. Автоматът разпознава една дума, ако някое от копията му я разпознае.

За дадената диаграма на фиг. 17 преходите получаваме следния недетерминиран краен автомат $\mathbf{A}_1=(\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$, в който $\delta$ е следната функция:

| | |
|---|---|
| $\delta(q_0, a)=\{q_0, q_1\}$ | $\delta(q_1,b)=\{q_0, q_1, q_2\}$ |
| $\delta(q_0, b)=\varnothing$ | $\delta(q_2, a)=\varnothing$ |
| $\delta(q_1, a)=\varnothing$ | $\delta(q_2,b)=\varnothing$. |

Графично можем да изобразим работата на автомата $\mathbf{A}_1$, върху думата *aabbb* като разклоняващ се процес (фиг. 18).

След последната буква $b$ на думата *aabbb* автоматът $\mathbf{A}_1$ дава множеството от състояния $\{q_0, q_1, q_2\}$ и тъй като сред елементите му има заключително състояние — $q_0$, този автомат разпознава думата *aabbb*.

Функцията на преходите $\delta$ можем да доопределим за наредените двойки („състояние“, „дума върху $\Sigma$“) по следния начин:

$\delta(q, \Lambda)=\{q\}$

$\delta(q, \alpha a)=\bigcup_{p\in\delta(q,\alpha)} \delta(p, a)$ за всяка дума $\alpha\in\Sigma^*$ и за всяка буква $a\in\Sigma$.

(Ако $\delta(q, a)=\{p_1, p_2, \ldots, p_n\}$, то

$$\bigcup_{p\in\delta(q,a)}\delta(p, a)=\delta(p_1, a)\cup\delta(p_2, a)\cup\ldots\cup\delta(p_n, a).$$

В такъв случай недетерминираният краен автомат $\mathbf{A}=(K, \Sigma, \delta, q_0, F)$ разпознава думата $\alpha\in\Sigma^*$, ако $\delta(q_0,\alpha)\cap F=\varnothing$. Езикът $T(\mathbf{A})$, разпознаван от $\mathbf{A}$, се определя от всички думи на $\Sigma^*$, разпознавани от $\mathbf{A}$, т. е.

$T(\mathbf{A})=\{\alpha;\ \alpha\in\Sigma^*$ и $\delta(q_0, \alpha)\cap F\neq\varnothing\}$.

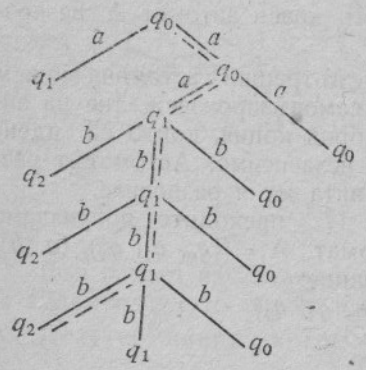
Да разгледаме следния недетерминиран краен автомат:

$\mathbf{A}_2=(\{q_0, q_1, q_2, q_3\}, \{a, b, c\}, \delta, q_0, \{q_2\})$

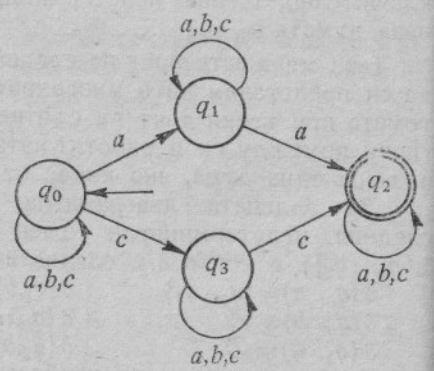
с функция на преходите

$\delta(q_0, a)=\{q_0, q_1\}$
$\delta(q_0, b)=\{q_0\}$
$\delta(q_0, c)=\{q_0, q_3\}$
$\delta(q_1, a)=\{q_1, q_2\}$
$\delta(q_1, b)=\{q_1\}$
$\delta(q_1, c=\{q_1\}$

$\delta(q_2, a)=\{q_2\}$
$\delta(q_2, b)=\{q_2\}$
$\delta(q_2, c)=\{q_2\}$
$\delta(q_3, a)=\{q_3\}$
$\delta(q_3, b)=\{q_3\}$
$\delta(q_3, c)=\{q_2, q_3\}$.




Фиг. 18




Фиг. 19

По диаграмата на преходите на $\mathbf{A}_2$ (фиг. 19) лесно се вижда, че езикът $T(\mathbf{A}_2)$, разпознаван от автомата $\mathbf{A}_2$, се състои от всички думи от вида $\alpha a\beta a\gamma$ или $\alpha c\beta c\gamma$, където $\alpha, \beta, \gamma\in\{a, b, c\}^*$.



Всеки детерминиран краен автомат очевидно може да се разглежда и като недетерминиран, така че всеки език, който се разпознава от детерминиран краен автомат, се разпознава и от недетерминиран. Сега ще покажем обратното: **ако един език се разпознава от недетерминиран краен автомат, той се разпознава и от някакъв детерминиран краен автомат.**

Да вземем произволен недетерминиран краен автомат $\mathbf{A}=(K, \Sigma, \delta, q_0, F)$. Ще построим следния детерминиран краен автомат

$$\mathbf{B}=(S, \Sigma, \delta', s_0, F'),$$

както следва:

— $S$ съдържа по едно вътрешно състояние за всяко подмножество на $K$; ако $\{p_1, \ldots, p_n\}$ е подмножество на $K$, съответното вътрешно състояние от $S$ ще означаваме с $s_{[p_1,p_2,\ldots p_n]}$. Броят на вътрешните състояния от $S$ е равен на $2^m$, където $m$ е броят на състоянията от $K$;

— $s_0=s_{[q_0]}$;

— $F'$ се състои от всички състояния $s_{[p_1, \ldots, p_n]}$, за които множеството $\{p_1, \ldots, p_n\}$ има непразно сечение с $F$ от $\mathbf{A}$;

— функцията на преходите $\delta'$ определяме за всяка буква $a$ от $\Sigma$ и всяко вътрешно състояние $s_{[p_1, \ldots, p_n]}$ от $S$ по следния начин:

$\delta'(s_{[p_1, \ldots, p_n]}, a)=s_{[q_1 \ldots, q_l]}$ тогава и само тогава, когато $\{q_1, \ldots, q_l\}=\delta(p_1, a)\cup\delta(p_2 a)\cup\ldots\cup\delta(p_n, a)$ ($\delta(p_1, a), \ldots, \delta(p_n, a$ са подмножества на $K$).

С индукция по дължината на думата $\alpha$ ще покажем, че $\delta'(s_{[q_0]}, \alpha)=s_{[p_1, \ldots, p_i]}$ тогава и само тогава, когато $\delta(q_0, \alpha)=\{p_1, \ldots, p_i\}$.

Действително, когато $\alpha=\Lambda$, $\delta(q_0, \Lambda)=q_0$ и тогава $\delta'(s_{[q_0]}, \Lambda)=s_{[q_0]}$ Обратно, ако $\delta'(s_{[q_0]}, \Lambda)=s_{[q_0]}$, тогава $\{q_0\}=\delta(q_0, \Lambda)$.

Да предположим, че за всички думи $\alpha\in\Sigma^*$ с дължина не по-голяма от $n$ $\delta'(s_{[q_0]}, \alpha)=s_{[p_1, \ldots, p_i]}$ тогава и само тогава, когато $\delta(q_0, \alpha)=\{p_1, \ldots, p_i\}$ и нека $\alpha a$ е произволна дума с дължина $n+1$. В такъв случай

$\delta'(s_{[q_0]}, \alpha a)=\delta'(\delta'(s_{[q_0]}, \alpha), a)=\delta'(s_{[p_1, \ldots, p_i]}, a)$ тогава и само тогава, когато $\delta(q_0, a)=\{p_1, \ldots, p_i\}$, а по определение $\delta'(s_{[p_1, \ldots, p_i]}, a)=s_{[r_1, \ldots, r_j]}$ тогава и само тогава, когато $\{r_1, \ldots, r_j\}=\delta(p_1, a)\cup\ldots\cup\delta(p_i, a)$.

Следователно

$\delta'(s_{[q_0]}, \alpha a)=s_{[r_1, \ldots, r_j]}$ тогава и само тогава, когато $\{r_1, \ldots, r_j\}=\delta(p_1, a)\cup\ldots\cup\delta(p_i, a)$, а $\{p_1, \ldots, p_i\}=\delta(q_0,\alpha)$, т. е. когато $\{r_1, \ldots, r_j\}=\{r_1, \ldots, r_j\}=\cup\delta(p_k, \alpha)p_k\in\delta(q_0, \alpha)$, а това не е нищо друго освен $\delta(q_0, \alpha a)$.

Езикът, разпознаван от **B**, е:
$T(\mathbf{B})=\{\omega : \omega \in \Sigma^* \text{ и } \delta'(s_{[q_0]}, \omega) \in F'\}$.

Съгласно доказаното от нас твърдение $\delta'(s_{[q_0]}, \omega)=s_{[p_1, \ldots, p_i]}$ тогава и само тогава, когато $\delta(q_0, \omega)=\{p_1, \ldots, p_i\}$. Състоянието $s_{[p_1, \ldots, p_i]}$ е заключително, когато сред $p_1, \ldots, p_i$ има поне едно заключително състояние от $K$, но тогава **A** и **B** едновременно разпознават думата $\omega$.

Получихме, че автоматите са еквивалентни, т. е. $T(\mathbf{A})=T(\mathbf{B})$.

И така, от гледна точка на разпознаваните от тях езици детерминираните и недетерминираните крайни автомати са еквивалентни: **един език се разпознава от недетерминиран краен автомат тогава и само тогава, когато се разпознава и от детерминиран краен автомат.**

Поради това, често пъти ще пропускаме думите детерминирани и недетерминирани и ще говорим само за крайни автомати.

### Упражнения

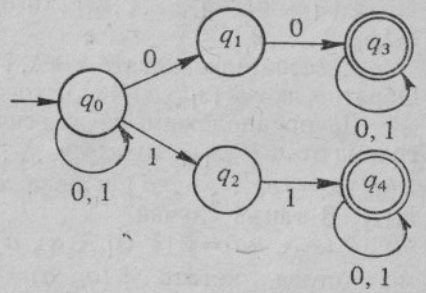
**1.** Постройте недетерминиран краен автомат, разпознаващ всички думи от вида:

а) $0\alpha 0$, $\alpha \in \{0, 1\}^*$; б) $aa\alpha bb$, $\alpha \in \{a, b\}^*$;

**2.** Постройте детерминирани крайни автомати, които да разпознават същите езици, които се разпознават от посочените като примери недетерминирани крайни автомати.

**3.** Нека недетерминираният краен автомат **A** е зададен със следната диаграма на преходите (фиг. 20):

Определете езика $T(\mathbf{A})$.




Фиг. 20

## 2.5 ЕКВИВАЛЕНТНОСТ НА КРАЙНИТЕ АВТОМАТИ И АВТОМАТНИТЕ ГРАМАТИКИ

В тази част ще изследваме какъв вид граматики пораждат езиците, разпознавани от крайни автомати. Ще покажем, че класът на автоматните езици съвпада с класа на езиците, разпознавани



от крайните автомати, т. е., че **всеки автоматен език се разпознава от краен автомат и всеки език, разпознаван от краен автомат, е автоматен.**

Действително, нека $L$ е произволен автоматен език (т. е. от тип 3). Това означава, че съществува автоматна граматика

$$\Gamma=(V, W, S, P),$$

която поражда думите на езика $L$: $L=L(\Gamma)$. По граматиката $\Gamma$ ще построим недетерминиран краен автомат **N**, който ще разпознава точно думите от $L$. Определяме автомата $\mathbf{N}=(K, \Sigma, \delta, q_0, F)$ по следния начин:

— вътрешни състояния на **N** са всички нетерминални символи от $W$ и един допълнителен символ $Q$, който не принадлежи на $V \cup W$, т. е. $K=W \cup \{Q\}$;

— входната азбука на **N** се състои от всички терминални символи, т. е. $\Sigma=V$;

— вътрешното състояние $Q$ принадлежи на множеството $\delta(A, a)$, само ако в $P$ има правило $A \to a$. На $\delta(A, a)$ принадлежат и всички състояния $B$, за които в $P$ има правило $A \to aB$. Освен това $\delta(Q, a)=\varnothing$ за всяка буква $a$;

— начално състояние на **N** е $S$;

— заключително състояние на **N** е $Q$, ако в $P$ няма правило $S \to \Lambda$. Ако в $P$ има правило $S \to \Lambda$, заключителни състояния са $Q$ и $S$. (Чрез добавянето на началното състояние $S$ към заключителните състояния осигуряваме наличието на празната дума $\Lambda$ в езика, разпознаван от **N**, когато $\Lambda$ се поражда от $\Gamma$).

Нека сега непразната дума $\omega=a_{i_1} a_{i_2} \ldots a_{i_k}$ се поражда от така определената граматика $\Gamma$. Това означава, че в $\Gamma$ има извод:

$$S \vdash a_{i_1} A_1 \vdash a_{i_1} a_{i_2} A_2 \vdash \ldots \vdash a_{i_1} a_{i_2} \ldots a_{i_{k-1}} A_{k-1} \vdash a_{i_1} \ldots a_{i_k},$$

където $A_1, \ldots A_{k-1}$ са някакви нетерминални символи. Но тогава съгласно определението на функцията на преходите $\delta(S, a_{i_1})$ съдържа $A_1$, $\delta(A_1, a_{i_2})$ съдържа $A_2$ и т. н., $\delta(A_{k-1}, a_{i_k})$ съдържа заключителното състояние $Q$. Получаваме, че $\omega$ се разпознава от крайния автомат **N**.

Нека сега непразната дума $\omega=a_{i_1} \ldots a_{i_k}$ се разпознава от автомата **N**. Това означава, че съществува поредица от състояния $S, A_1, A_2, \ldots, A_{k-1}, Q$ такива, че $\delta(S, a_{i_1})$ съдържа $A_1$, $\delta(A_1, a_{i_2})$ съдържа $A_2$ и т. н., $\delta(A_{k-1}, a_{i_k})$ съдържа $Q$. Но тогава съгласно определението на функцията на преходите в $P$ трябва да има правила

$$S \to a_{i_1} A_1,\ A_1 \to a_{i_2} A_2, \ldots, A_{k-1} \to a_k.$$

Тези правила, приложени последователно, пораждат думата $\omega$ в граматиката $\Gamma$.

И така, за всеки автоматен език $L$ съществува недетерминиран краен автомат N, който разпознава езика $L$, т. е. $T(\mathrm{N})=L$.

Обратно, нека $D$ е произволен детерминиран краен автомат:

$$D=(K, \Sigma, \delta, q_0, F).$$

Ще построим автоматна граматика $\Gamma$, която да поражда точно думите, разпознавани от $D$. Определяме следната автоматна граматика $\Gamma_1$:

$$\Gamma_1=(\Sigma, K, q_0, P),$$

като в множеството $P$ за всяко равенство $\delta(q, a)=p$ поставяме правило $q\to ap$, а ако $p$ е заключително състояние, поставяме и правило $q\to a$.

За тази автоматна граматика $\Gamma_1$ по същия начин се доказва, че тя поражда всички думи, които $D$ разпознава, без празната дума. В случая, когато $D$ разпознава празната дума, т. е, когато началното състояние е и заключително, намираме автоматната граматика, еквилалентна на $\Gamma_1$, в която началният символ не се среща отдясно на правилата. Към тази граматика добавяме правило $S\to\Lambda$ и получаваме търсената автоматна граматика $\Gamma$, която поражда езика $T(D)$.

И така, намерихме следните зависимости: езиците, разпознавани от детерминираните крайни автомати, се пораждат от автоматни граматики, автоматните езици се разпознават от недетерминирани крайни автомати, а езиците, разпознавани от недетерминираните крайни автомати, се разпознават и от детерминираните крайни автомати. Това може да изобразим с диаграмата на фиг. 21:

Фиг. 21


Разгледаните зависимости означават, че **класът на езиците, които се пораждат от автоматни граматики, съвпада с класа на езиците, които се разпознават от крайни автомати (детерминирани или не).**

Например за автоматната граматика

$$\Gamma=(\{0, 1\}, \{S, A,\} S, \{S\to 0S, S\to 1S, S\to 1A, A\to 1\}),$$

която вече разглеждахме, получаваме следния недетерминиран краен автомат $A$, разпознаващ езика $L(\Gamma)$:

$A=(\{S, A, Q\}\ \{0, 1\}, \delta, S, \{Q\})$ с функция на преходите:

| | |
|---|---|
| $\delta(S, 0)=\{S\}$ | $\delta(A, 0)=\varnothing$ |
| $\delta(S, 1)=\{S, A\}$ | $\delta(Q, 1)=\varnothing$ |
| $\delta(A, 1)=\{Q\}$ | $\delta(Q, 0)=\varnothing$. |

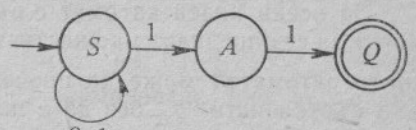
От диаграмата на преходите на автомата A (фиг. 22) става очевидно, че $L(\Gamma)$ се състои от всички думи, съставени от 0 до 1, които завършват с две последователни 1.

Като използуваме диаграмата на преходите на детерминирания краен автомат, разпознаващ правилните решения на задачата за човека, вълка, заека и зелката, можем да построим следната автоматна граматика, която поражда всички правилни решения на тази задача и само тях:

$$\Gamma=(\{A, B, C, D\}, \{q_0, q_1, \ldots, q_9\}, q_0, P),$$

като $P$ се състои от следните правила:

| | | |
|---|---|---|
| $q_0\to Cq_4$ | $q_3\to Cq_6$ | $q_6\to Bq_2$ |
| $q_1\to Bq_7$ | $q_3\to Dq_7$ | $q_6\to Cq_3$ |
| $q_1\to Cq_8$ | $q_4\to Aq_7$ | $q_7\to Bq_1$ |
| $q_2\to Aq_5$ | $q_4\to Cq_9$ | $q_7\to Aq_4$ |
| $q_2\to Bq_6$ | $q_5\to Cq_0$ | $q_7\to Dq_3$ |
| $q_2\to Dq_8$ | $q_5\to Aq_2$ | $q_8\to Cq_1$ |
| $q_9\to Cq_4$ | $q_9\to C$ | $q_8\to Dq_2$ |




Фиг. 22

## Упражнения

1. Дадена е автоматната граматика

$\Gamma=(\{a, b\}, \{S, A, B,\} S, \{S\to aA, A\to bB, A\to b, B\to aB, B\to Bb, B\to a, B\to b, S\to\Lambda\})$.

а) Постройте краен автомат, който да разпознава $L$ (Г).

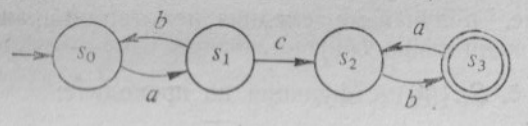
б) По диаграмата на преходите определете $L$ (Г).

в) Допълнете диаграмата на преходите така, че крайният автомат да бъде навсякъде определен и да разпознава същия език.

**2.** На фиг. 23 е дадена диаграма на преходите на недетерминирания краен автомат **А**.

а) Намерете езика $T$ (**А**).

б) Постройте автоматна граматика, която да поражда езика $T$ (**А**).




Фиг. 23

**3.** Постройте детерминиран краен автомат, който да разпознава думи от 0 до 1, в които има четен брой 0 и четен брой 1. За този автомат постройте автоматна граматика, която да поражда същия език.

## 2.6. ЕДНА ХАРАКТЕРИСТИКА НА КЛАСА НА АВТОМАТНИТЕ ЕЗИЦИ

Когато разглеждахме свойствата на автоматните езици, показахме, че крайните множества представляват автоматни езици и всички езици, които се получават от тях чрез операциите обединение, произведение и итерация, са също автоматни езици.

Интересно е, че е вярно и обратното: **всеки автоматен език може да се получи от крайни множества чрез операциите обединение, произведение и итерация.** С това автоматните езици могат напълно да се характеризират чрез крайните множества и операциите обединение, произведение и итерация.

За всеки краен автомат с вътрешни състояния $q_1, q_2, \ldots, q_n$ могат да се определят множествата $R^k_{ij}$, съставени от всички думи, които автоматът може да прочете, преминавайки от състоянието $q_i$ в състоянието $q_j$, без да е преминал през състояния с номер, по-голям от $k$. Ако вземем диаграмата на състоянията, $R^k_{ij}$ ще се състои от всички пътища, които водят от върха $q_i$ до върха $q_j$, без да преминават през върховете $q_{k+1}, \ldots, q_n$.

Например за построения вече от нас краен автомат (фиг. 24), който разпознава всички думи, в които няма две последователни единици, получаваме:

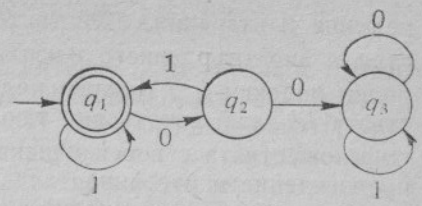
$$R^0_{12}=\{0\}\ ;\ R^1_{21}=\{1^n\ ;\ n\leqq 1\}\ ;\ R^1_{12}=\{1^n\,0\ ;\ n\geqq 0\}\ ;\ R^1_{13}=\varnothing.$$



Множествата $R^k_{ij}$ могат да се определят чрез индукция по горния индекс $k$ по следния начин:

$$R^0_{ij}=\{a\ ;\ a\in\Sigma \text{ и } \delta(q_i,a)=q_j\};$$
$$R^k_{ij}=R^{k-1}_{ij}\cup(R^{k-1}_{ik}.(R^{k-1}_{kk})^{*}R^{k-1}_{kj})$$

за всяко $i$ и $j$.




Фиг. 24

Ще поясним второто равенство. Нека $\alpha\in R^k_{ij}$ Тогава $\delta(q_i,\alpha)=q_j$, а в редицата последователни преходи от едно вътрешно състояние към следващото, започваща от $q_i$ и завършваща с $q_j$, не могат да се срещат вътрешни състояния с индекс, по-голям от $k$. Състоянието $q_k$ може обаче да се среща няколко пъти. Да означим всички негови появявания при работата на крайния автомат върху входната дума $\alpha$:

$$q_i, \ldots, q_k, \ldots, q_k, \ldots, q_k, \ldots, q_k, \ldots, q_j.$$

Тогава думата $\alpha$ може да се раздели на следните части: частта от $q_i$ до първото появяване на $q_k$, частта от първото появяване на $q_k$ до второто появяване на $q_k$, и т. н. и накрая частта на последното появяване на $q_k$ до края на $\alpha$. Но в такъв случай първата част ще бъде от множеството $R^{k-1}_{ik}$, всяка от следващите части ще бъде от множеството $R^{k-1}_{kk}$, а последната — от $R^{k-1}_{kj}$.

Ако състоянието $q_k$ не се среща в редицата от състояния, водещи от $q_i$ до $q_j$ чрез думата $\alpha$, тогава $\alpha$ е от $R^{k-1}_{ij}$.

Това означава, че всяка дума от $R^k_{ij}$ е или дума от $R^{k-1}_{ij}$, или дума от $R^{k-1}_{ik}(R^{k-1}_{kk})^{*}R^{k-1}_{kj}$.

Обратно, всяка дума от $R^{k-1}_{ij}$ и от $R^{k-1}_{ik}(R^{k-1}_{kk})^{*}R^{k-1}_{kj}$ също осъществява преход от $q_i$ до $q_j$ без преминаване през състояния с индекс по-голям от $k$.

Получаваме, че двете множества $R^k_{ij}$ и $R^{k-1}_{ij}\cup(R^{k-1}_{ik}(R^{k-1}_{kk})^{*}R^{k}_{kj})$ съвпадат.

В разглежданията от нас пример

$$R^1_{21}=R^0_{21}\cup(R^0_{21}.(R^0_{11})^{*})=\{1\}\cup(\{1\}.\{1\}^{*})=\{1\}\cup\{1^n\ ;\ n\geqq 1\}$$
$$=\{1^n;n\geqq 1\}.$$

От това представяне на множествата $R^k_{ij}$ следва, използувайки индукция по $k$, че за произволни $i$ и $j$ множествата $R^k_{ij}$ се получават от крайни множества посредством операциите обединение, произ-

ведение и итерация. Действително всички $R_{ij}^0$ са крайни множества, а ако твърдението е изпълнено за множества $R_{ij}^{k-1}$ с произволни индекси $i$, $j$ и горен индекс $k$—1, то е вярно и за множествата $R_{ij}^k$, тъй като всяко такова множество може да се получи от множествата с горен индекс $k$—1 чрез операциите обединение, произведение и итерация.

За езика $T(\mathbf{A})$, разпознаван от произволен краен автомат **A** с общо $n$ състояния, начално състояние $q_1$ и заключителни състояния $q_{p_1}, \ldots, q_{p_m}$ получаваме:

$$T(\mathbf{A})=R_{1p_1}^n \cup R_{1p_2}^n \cup \ldots \cup R_{1p_m}^n.$$

Но в такъв случай всеки език, разпознаван от краен автомат (т. е. всеки автоматен език), може да се получи от крайни множества чрез операциите обединение, произведение и итерация.

Получаваме следната основна характеристика за класа на автоматните езици:

**Един език е автоматен тогава и само тогава, когато може да се получи от крайни множества чрез операциите обединение, произведение и итерация.**

### Упражнения

1. Докажете, че за всеки автоматен език $L$ върху азбуката $V$ допълнението $\overline{L}$ на езика $L$ до множеството $V^*$ е също автоматен език.

Указание: В напълно определения детерминиран краен автомат $(K, \Sigma, \delta, q_0, F)$, разпознаващ езика $L$, заменете $F$ с $K-F$.

2. Докажете, че множеството $V^*$ е автоматен език за всяка азбука $V$.

3. Докажете, че сечението на два автоматни езика е също автоматен език.

Указание: Използувайте равенството $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ и задача 1.

### 2.7 СЪЩЕСТВУВАНЕ НА НЕАВТОМАТНИ ФОРМАЛНИ ЕЗИЦИ

Да вземем произволен детерминиран напълно определен краен автомат $\mathbf{A}=(K, \Sigma, \delta, q_0, F)$ и да разгледаме действието на **A** върху различни входни думи. Тъй като резултат от работата на **A** върху дадена дума е последното вътрешно състояние, можем да класифицираме думите от $\Sigma^*$ по това състояние. С други думи, върху $\Sigma^*$ автоматът **A** определя следната релация $R_A$: две думи $\alpha$ и $\beta$ са свързани с релацията $R_A$ точно когато автоматът **A**, започвайки работа в начално състояние, дава един и същ резултат за думите $\alpha$ и $\beta$:



$$R_A=\{(\alpha, \beta);\ \alpha, \beta \in \Sigma^*,\ \delta(q_0, \alpha)=\delta(q_0, \beta)\}.$$

Очевидно, тази релация е рефлексивна, симетрична и транзитивна и следователно $R_A$ **е релация на еквивалентност.** Известно е, че всяка релация на еквивалентност разделя множеството, върху което е определена, на непресичащи се класове на еквивалентност. Автоматът **A** е напълно определен, поради това всяка дума от $\Sigma^*$ попада в някакъв клас на еквивалентност. Ще отбележим също така, че броят на класовете на еквивалентност, на които $R_A$ разделя $\Sigma^*$, не е по-голям от броя на вътрешните състояния на **A**, тъй като в един клас на еквивалентност попадат думите, върху които работата на **A** завършва с едно и също състояние.

Броят на класовете на еквивалентност се нарича индекс на релацията на еквивалентност. В такъв случай може да се каже, че **релацията $R_A$ има винаги краен индекс,** тъй като всеки краен автомат има винаги краен брой вътрешни състояния.

Релацията $R_A$ притежава още следното свойство: Ако две думи са свързани с релацията $R_A$, то десните им конкатенации с произволна дума от $\Sigma^*$ са също свързани с тази релация: т. е. от $(\alpha, \beta) \in R_A$ следва, че $(\alpha\omega, \beta\omega) \in R_A$ за произволна дума $\omega \in \Sigma^*$. Това свойство ще наричаме **дясна инвариантност на релацията $R_A$.**

Дясната инвариантност на $R_A$ следва от това, че две думи $\alpha$ и $\beta$ са свързани с $R_A$, когато привеждат автомата **A** от $q_0$ в едно и също състояние, например $q_j$. Но тъй като **A** е детерминиран и напълно определен, каквато и дума $\omega \in \Sigma^*$ да вземем, автоматът $A$, продължавайки работа върху думата $\omega$ в състояние $q_j$, ще даде и в двата случая един и същ резултат, т. е.

$$\delta(q_0, \alpha\omega)=\delta(q_0, \beta\omega).$$

Ще определим върху множеството на всички думи $\Sigma^*$ още една релация. Нека е даден произволен език $L \subseteq \Sigma^*$. Този език определя следната релация $R_L$: две думи $\alpha$ и $\beta$ от $\Sigma^*$ са свързани с $R_L$ тогава и само тогава, когато конкатенациите на $\alpha$ и $\beta$ отдясно, с коя да е дума $\gamma$ от $\Sigma^*$ едновременно принадлежат или не на езика $L$:

$R_L=\{(\alpha, \beta);\ \alpha, \beta \in \Sigma^*$ и за всяко $\gamma \in \Sigma^*$ $\alpha\gamma \in L$ тогава и само тогава, когато $\beta\gamma \in L\}$.

**Релацията $R_L$ е също релация на еквивалентност върху $\Sigma^*$.** Наистина рефлексивността $((\alpha, \alpha) \in R_L)$ и симетричността (ако

$(\alpha, \beta) \in R_L$, то $(\beta, \alpha) \in R_L$) са очевидни. А транзитивността следва от това, че ако за всяко $\gamma \in \Sigma^*$: $\alpha\gamma \in L$ тогава и само тогава, когато $\beta\gamma \in L$, а $\beta\gamma \in L$ тогава и само тогава, когато $\omega\gamma \in L$, то за всяко $\gamma$: $\alpha\gamma \in L$ тогава и само тогава, когато $\omega\gamma \in L$. В такъв случай и релацията $R_L$ разделя множеството $\Sigma^*$ на непресичащи се класове на еквивалентност. **$R_L$ също е дясно инвариантна релация.** Действително, нека $(\alpha, \beta) \in R_L$. Тогава за всякакви думи $\sigma$: $\alpha\gamma\sigma \in L$ тогава и само тогава, когато $\beta\gamma\sigma \in L$. А това означава, че $(\alpha\gamma, \beta\gamma) \in R_L$ тъй като за всякакви думи $\sigma$: $\alpha\gamma\sigma \in L$ тогава и само тогава, когато $\beta\gamma\sigma \in L$.

Релациите $R_A$ и $R_L$ са необходими за доказателството на следната важна и с интересни приложения теорема.

**Нека $L \subseteq \Sigma^*$ е произволен език върху азбуката $\Sigma$. $L$ е автоматен език тогава и само тогава, когато релацията $R_L$ има краен индекс.**

Да вземем произволен автоматен език $L$. За този език съществува детерминиран напълно определен краен автомат $\mathbf{A} = (K, \Sigma, \delta, q_0, F)$, който го разпознава. Автоматът **A** определя релацията $R_A$ върху множеството $\Sigma^*$. Тъй като езикът $L$ се състои от думите, които довеждат автомата **A** от състояние $q_0$ в заключително състояние, $L$ е обединение на онези класове на еквивалентност на $R_A$, които съответстват на заключителните състояния на **A**. Да вземем две произволни думи $\alpha$ и $\beta$, свързани с $R_A$: $(\alpha, \beta) \in R_A$.

Тъй като $R_A$ е дясно инвариантна релация, за всяка дума $\gamma$ от $\Sigma^*$ $\alpha\gamma$ и $\beta\gamma$ са също свързани с релацията $R_A$, т. е. $\alpha\gamma$ и $\beta\gamma$ довеждат автомата **A** от състояние $q_0$ до едно и също крайно състояние. Ако това състояние е заключително, думите $\alpha\gamma$ и $\beta\gamma$ едновременно принадлежат на $L$. Ако това състояние не е заключително — двете думи $\alpha\gamma$ и $\beta\gamma$ едновременно не принадлежат на $L$. И така, ако $(\alpha, \beta) \in R_A$, за всяка дума $\gamma \in \Sigma^*$ $\alpha\gamma \in L$ тогава и само тогава, когато $\beta\gamma \in L$, т. е. $\alpha$ и $\beta$ са свързани с релацията $R_L$. В такъв случай, ако две думи принадлежат на един и същ клас на еквивалентност на релацията $R_A$, те са в един и същ клас на еквивалентност и на релацията $R_L$. Това означава, че всеки клас на еквивалентност на $R_A$ е подмножество на клас на еквивалентност на $R_L$ или, с други думи, всеки клас на еквивалентност на $R_L$ е обединение на класове на еквивалентност на $R_A$.

Но **A** е краен автомат, а тогава релацията $R_A$ има краен индекс. В такъв случай и релацията $R_L$ има краен индекс.

Обратно, нека за произволен формален език $L \subseteq \Sigma^*$ релацията $R_L$ има краен индекс. Ще докажем, че тогава $L$ е автоматен език.



За целта ще конструираме детерминиран краен автомат **M**, който да разпознава езика $L$.

Класовете на еквивалентност на релацията $R_L$ ще означаваме чрез някой техен представител: $[\alpha]$ ще означава класа на еквивалентност, в който е думата $\alpha$.

Определяме автомата **M** по следния начин:

— вътрешните състояния на **M** ще бъдат класовете на еквивалентност на релациите $R_L$. Те са краен брой, тъй като $R_L$ има краен индекс;

— входна азбука за **M** ще бъде азбуката $\Sigma$;

— за всяка буква $a \in \Sigma$ и за всяко вътрешно състояние $[\alpha]$ $\delta([\alpha], a) = [\alpha a]$. Това определение е коректно, тъй като $R_L$ е дясно инвариантна релация и следователно десните конкатенации на думите от един клас на еквивалентност с произволна буква $a$ ще бъдат отново в един и същ клас на еквивалентност. Ако представител на първия клас е думата $\alpha$, тогава за представител на втория клас можем да вземем думата $\alpha a$;

— начално състояние на **M** ще бъде класът на еквивалентност $[\Lambda]$, съдържащ празната дума $\Lambda \in \Sigma^*$;

— заключителни състояния на **M** ще бъдат всички класове на еквивалентност, съставени от думи на $L$. Ще отбележим, че ако в един клас на еквивалентност има поне една дума от $L$, тогава всички думи от този клас ще бъдат от $L$. Това следва от дясната инвариантност на релацията $R_L$: две думи са в един клас на еквивалентност тогава и само тогава, когато конкатенациите им с коя и да е дума от $\Sigma^*$ (в случая да вземем празната дума) едновременно принадлежат или не принадлежат на $L$.

Лесно се вижда, че $T(\mathbf{M}) = L$. Ако $\omega$ е дума от $L$, тогава $\delta([\Lambda], \omega) = [\Lambda\omega] = [\omega]$, а това е заключително състояние по определение. Ако $\omega$ се разпознава от **M**, тогава $\delta([\Lambda], \omega) = [\omega]$ е заключително състояние, а в такъв случай $\omega \in L$.

С това теоремата е доказана.

Като пряко следствие от тази теорема получаваме:

**За всеки автоматен език $L \subseteq \Sigma^*$ съществува детерминиран напълно определен краен автомат, разпознаващ $L$, който има минимален брой вътрешни състояния в сравнение с всеки друг детерминиран напълно определен краен автомат, разпознаващ $L$.**

Действително всеки детерминиран напълно определен краен автомат **A** определя върху множеството $\Sigma^*$ релацията на еквивалентност $R_A$. Всеки клас на еквивалентност на $R_L$ е обединение

на класовете на еквивалентност на $R_A^*$, така че броят на класовете на $R_A$ е винаги по-голям или равен на броя на класовете на $R_L$. Но класовете на $R_L$ са вътрешни състояния на построения в теоремата автомат **M**, а броят на класовете на еквивалентност на $R_A$ не е по-голям от броя на вътрешните състояния на **A**. Следователно броят на състоянията на произволен детерминиран напълно определен краен автомат **A** е винаги по-голям или равен на броя на състоянията на автомата **M**, а автоматите **A** и **M** разпознават един и същ език $L$. Обикновено **M** се нарича **минимален краен автомат за езика *L***.

Йерархията на Чомски за формалните езици определихме чрез йерархията на пораждащите ги граматики: езикът е автоматен, ако съществува пораждаща го автоматна граматика, езикът е безконтекстен, ако го поражда безконтекстна граматика, и т. н. Тази йерархия ще има смисъл, ако всеки клас има свое собствено съдържание, различно от съдържанието на останалите класове, т. е. ако има безконтекстен език, който не може да бъде автоматен, контекстен език, който не може да бъде безконтекстен и език от общ вид, който не може да бъде контекстен.

От теоремата, която доказахме, получаваме следното важно следствие:

**Съществуват безконтекстни езици, които не могат да бъдат автоматни, т. е. не могат да се породят от автоматна граматика.**

Да вземем например безконтекстния език $L=\{a^n b^n;\ n\geqq 0\}$, който, както знаем, се поражда от безконтекстната граматика Г:

$$\Gamma=(\{a,\ b\},\{S,\ A\},S,\{S\to aAb,A\to aAb,\ S\to ab,\ A\to ab,S\to\Lambda\}).$$

Да допуснем, че има автоматна граматика, която също да поражда езика $L$. Тогава $L$ е автоматен език и съгласно теоремата релацията $R_L$, определена върху $\{a, b\}^*$, има краен индекс. Това означава, че всяка дума от $\{a, b\}^*$ попада в някой от краен брой, например $m$, класове. Да вземем следната редица от думи от множеството $\{a, b\}^*$:

$$a,\ a^2, a^3,\ \ldots\ ,\ a^{m+1}.$$

Тъй като броят на различните класове е $m$, то поне две думи от тази редица $a^i$ и $a^j$ $(i\neq j)$ попадат в един и същ клас, т. е. $(a^i, a^j)\in R_L$. Съгласно определението на релацията $R_L$ десните конкатенации на $a^i$ и $a^j$ с произволна дума от $\{a, b\}^*$ едновременно принадлежат или не принадлежат на $L$. Нека тази дума да е $b^i$.



Тогава $a^i b^i$ и $a^j b^i$ $(i\neq j)$ едновременно принадлежат или не на $L$. Думата $a^i b^i$ е от езика $L$, следователно $a^j b^i$ $(i\neq j)$ също трябва да е от езика $L$. Това обаче противоречи на определението на езика $L$: в $L$ са думите от вида $a^n b^n$, $n\geqq 0$. В такъв случай допускането, че $L$ е автоматен език е невярно. Следователно, $L$ не е автоматен език.

Перифразирайки това доказателство, можем да покажем, че българският език не е автоматен език. Разбира се, излаганите при това аргументи ще имат математически, а не лингвистичен характер и ще се отнасят по-скоро до формалния модел на езика, отколкото до самия език.

Нека $\Sigma$ е един достатъчно пълен речник на българския език (към който са добавени различни препинателни знаци). Тогава $\Sigma^*$ ще се състои от различни низове от български думи (т. е. от дадения речник), а множеството от правилните изречения на българския език $L_{\text{БЕ}}$ ще бъде подмножество на $\Sigma^*$.

Нека $A$ и $B$ са произволни разказвателни изречения. От тях чрез съюза *нито ... , нито ...* можем да образуваме изречението

*Нито A, нито B.*

Да заместим $A$ със същото изречение. Получаваме

*Нито нито A, нито B, нито B.*

Този процес може да продължава неограничено, макар че все по-трудно ще вникваме в смисъла на получаваните изречения. Ако приложим $n$ пъти тази процедура, ще получим изречението

$$(\textit{нито})^n\ A\ (,\textit{нито}\ B)^n.$$

Сега да допуснем, че $L_{\text{БЕ}}$ е автоматен език. В такъв случай релацията $R_{L_{\text{БЕ}}}$ има краен индекс и следователно разделя множеството $\Sigma^*$ на краен брой, например $m$, непресичащи се класове на еквивалентност.

Да разгледаме редицата от низове от

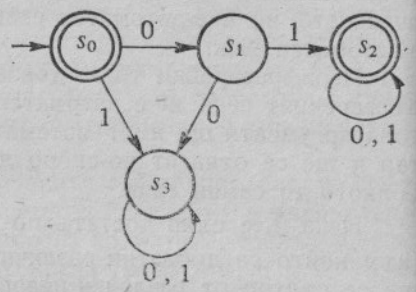
*нито*, $(\textit{нито})^2$, $(\textit{нито})^3$, ..... , $(\textit{нито})^{m+1}$.

Тъй като класовете на еквивалентност са $m$ на брой, поне два низа от тази редица, например $(\textit{нито})^k$ и $(\textit{нито})^l$ $(k\neq l)$, попадат в един и същ клас на еквивалентност, т. е. $((\textit{нито})^k, (\textit{нито})^l)\in R_{L_{\text{БЕ}}}$. Както знаем, от определението на $R_{L_{\text{БЕ}}}$ следва, че десните конкатенации на $(\textit{нито})^k$ и $(\textit{нито})^l$ е произволен низ от $\Sigma^*$ и едновременно принадлежат или не принадлежат на езика $L_{\text{БЕ}}$. Да разгледаме десните им конкатенации с низа $A$ $(,\textit{нито}\ B)^k$. Тъй като $(\textit{нито})^k A$ $(,\textit{нито}\ B)^k$ принадлежи на $L_{\text{БЕ}}$, тогава на $L_{\text{БЕ}}$ би трябвало да при-

надлежи и (*нито*)$^l$ $A$ (,нито $B$)$^k$, $k=l$. Това обаче е в противоречие с нашето разбиране на езика $L_{БЕ}$, тъй като правилно построени са само изреченията, в които $l=k$. Следователно, $L_{БЕ}$ не е автоматен език.

### Упражнения

1. Даден е напълно определения детерминиран краен автомат A (фиг.25). Намерете класовете на еквивалентност, на които релацията $R_A$ разделя множеството $\{0,1\}^*$.




Фиг. 25

2. Даден е езикът $L$, състоящ се от всички думи от 0,1, в които има две (може и непоследователни) единици. Опишете класовете на еквивалентност, на които $R_L$ разделя $\{0,1\}^*$.

3. Докажете, че следните езици не са автоматни:

а) $L = \{\alpha O(\alpha);\ \alpha \in \{a, b\},\ O(\alpha)$ е обръщане на $\alpha\}$;

б) $L = \{0^n 1^m,\ n \geq 1,\ m \leq n\}$;

в) $L = \{a^{n^2}, n \geq 1\}$;

г) $L = \{a^{2^n}\ n \geq 0\}$;

д) $L = \{\alpha\,\alpha;\ \alpha \in \{a, b\}^*\}$;

е) $L = \{b^n ab^n;\ n \geq 1\}$;

ж) $L = \{\alpha;\ \alpha \in \{0,1\}$ и в $\alpha$ има равен брой 0 и 1$\}$.

## 2.8. АЛГОРИТМИЧНИ ПРОБЛЕМИ, СВЪРЗАНИ С АВТОМАТНИТЕ ЕЗИЦИ

Формалните езици задаваме чрез различни крайни описания. Например автоматните езици можем да задаваме чрез автоматните граматики или чрез крайни автомати. Във връзка с това възникват редица въпроси за това какви са езиците, задавани с някакво крайно описание, дали са празни или не, дали са крайни или не, дали са еквивалентни или не и др. Ако съществува алгоритъм, който за произволни крайни описания на някакъв клас езици да определя дали задаваният език притежава дадено свойство, казваме, че **проблемът за разпознаване на това свойство за разглеждания клас езици е алгоритмично разрешим.**

Тук ще покажем, че **проблемите за разпознаване на това, дали произволен атоматен език е празен, краен или безкраен са алгоритмично разрешими.** Ще покажем и **алгоритмичната разрешимост на проблема за разпознаване на еквивалентността на два произволни автоматни езика.**

По-нататък ще считаме, че автоматните езици са зададени чрез автоматни граматики. Това няма да представлява никакво ограничение, тъй като, ако те са зададени чрез крайни автомати, за всеки краен автомат можем да намерим съответна автоматна граматика, която да поражда същия автоматен език.

Да започнем с проблема за това дали един език, зададен с автоматна граматика, е празен или не. Очевидно, ако пораждаме думите с дължина 1, след това с дължина 2 и т. н. и проверяваме последователно дали сред тях има терминална дума, няма да можем винаги да отговаряме на въпроса, дали езикът е празен или не, тъй като в случая, когато езикът е празен, процесът ще продължава неограничено дълго. Ако обаче съумеем да поставим някаква граница и да проверяваме само думите, чиято дължина е под тази граница, тогава винаги ще можем след краен брой стъпки да получаваме отговор „да“ или „не“.

Да вземем произволна автоматна граматика с $n$ нетерминални символа. Всяко пораждане на терминална дума $\omega$ в тази граматика графично е дадено на фиг. 26, където $S$ е начален символ, $A_1, A_2, \ldots A_m$ са произволни нетерминални символи, а $a_1$, $a_2, \ldots, a_{m+1}$ са произволни терминални символи.

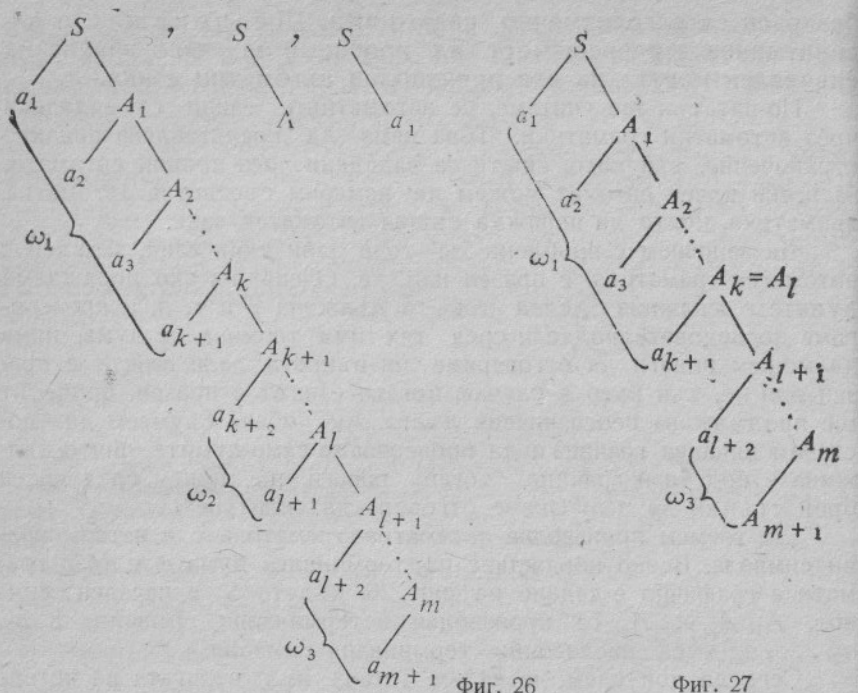
Сега да допуснем, че $m > n$. Тогава във веригата на нетерминалните символи $S, A_1, \ldots, A_m$ ще има поне два еднакви — например $A_k$ и $A_l$, като при това думата $\omega_2 = a_{k+2} \ldots a_{l+1}$ не е празна.

Можем да пропуснем частта от извода между нетерминалните символи $A_{k+1}$ и $A_l$ и да получим нов извод в дадената граматика (фиг. 27):

Както се вижда, думата $\omega_1\omega_3$ също се поражда от дадената граматика, а тя има по-малка дължина от думата $\omega = \omega_1\omega_2\omega_3$. Продължавайки този процес, след краен брой стъпки ще достигнем до дума, която се поражда от дадената граматика и чиято дължина не е по-голяма от $n$, т. е. от броя на нетерминалните символи в граматиката.

Налице е следният резултат: ако една автоматна граматика с $n$ нетерминални символи поражда терминална дума, тя поражда и терминална дума с дължина, по-малка или равна на $n$.

Оттук получаваме следния алгоритъм за проверка на това, дали произволна автоматна граматика поражда празен език: последователно построяваме всички възможни изводи с дължина 1, след това — с дължина 2 и т. н., и накрая — с дължина $n$ ($n$ е



Фиг. 26



Фиг. 27

броят на нетерминалните символи на граматиката). Ако на някой от етапите сме получили терминална дума, тогава езикът, пораждан от граматиката, не е празен. Ако на никой от етапите не сме получили терминална дума, тогава езикът, пораждан от тази граматика е празен.

Сега да преминем към проблема за това, дали произволен език, зададен с автоматна граматика е краен или безкраен.

Да се върнем отново към графичното представяне на процеса на пораждане на дума, чиято дължина е по-голяма от броя на нетерминалните символи на граматиката. Тъй като $A_k = A_l$, бихме могли вместо да съкратим частта между $A_k$ и $A_{l+1}$, да я повторим произволен брой пъти, например $i$ пъти. Получаваме, че за всяко $i \geqq 0$ в граматиката се извежда и думата $\omega_1 \omega_2^i \omega_3$, $|\omega_2| \geqq 1$. С други думи, ако една автоматна граматика поражда дума с дължина по-голяма от $n$, тази граматика поражда безкрайно много думи.

Обратно, нека автоматната граматика поражда безкраен език. Тогава граматиката ще поражда и думи с дължина по-голяма от



$n$, т. е. от броя на нетерминалните ѝ символи. Ще покажем, че поне една от тях е с дължина по-малка от $2n+1$. Действително, щом езикът е безкраен, в него има безкрайно много думи с дължина по-голяма от $n$, а сред тях трябва да има и дума $\omega$ с най-малка дължина. Ако $\partial(\omega) \leqq 2n$, твърдението е изпълнено. Нека $(\partial)\omega > 2n$. Тогава, както и преди, $\omega = \omega_1\omega_2\omega_3$, $1 \leqq \partial(\omega_2) \leqq n$, а думата $\omega_1\omega_3$ също се поражда от дадената граматика. Но $n < \partial(\omega_1\omega_3) < \omega$, което противоречи на това, че $\omega$ е най-късата дума от езика с дължина, по-голяма от $n$.

И така, езикът, породен от произволна автоматна граматика, е безкраен тогава и само тогава, когато в него има дума с дължина по-голяма от $n$ и по-малка от $2n+1$ ($n$ е броят на нетерминалните символи на граматиката).

Оттук получаваме следния алгоритъм: проверяваме дали сред думите с дължина между $n$ и $2n+1$, които се извеждат от $S$ в дадената граматика, има поне една терминална дума. Ако такава дума има — езикът е безкраен. Ако такава дума няма — езикът е краен.

Накрая да разгледаме и проблема за това дали две автоматни граматики пораждат един и същ език, т. е. дали са еквивалентни. Знаем, че обединението, сечението и допълнението на автоматни езици е също автоматен език (вж. например упражнения 2 и 3 от **2.6**. Нека $L_1$ и $L_2$ са произволни автоматни езици. Тогава автоматен ще бъде и езикът $(L_1 \cap \bar{L}_2) \cup (\bar{L}_1 \cap L_2)$. Веднага се вижда, че този език е празен тогава и само тогава, когато $L_1 = L_2$. Следователно чрез алгоритъма за проверка на това дали един автоматен език е празен или не, приложен към езика $(L_1 \cap \bar{L}_2) \cap (\bar{L}_1 \cup L_2)$, може да се провери, дали автоматните езици $L_1$ и $L_2$ са еквивалентни или не.

## Упражнения

1. Нека $L$ е автоматен език. Докажете, че съществува константа $n$ такава че за произволни думи $\alpha$, $\beta$ и $\gamma$ от $L$, при $\partial(\beta) = n$ думата $\beta$ може да се запише така $\beta = \omega_1\omega_2\omega_3$, където $\partial(\omega_2) \geqq 1$, и за всяко $i \geqq 0$ думите $\alpha\omega_1\omega_2^i\omega_3\gamma$ са също от езика $L$.

2. Докажете алгоритмичната разрешимост на проблема за разпознаване на това, дали един автоматен език $L_1$ се включва в друг автоматен език $L_2$, т. е. дали $L_1 \subseteq L_2$.

## 2.9 КРАЙНИТЕ АВТОМАТИ КАТО СИСТЕМИ ЗА ПРЕВОД

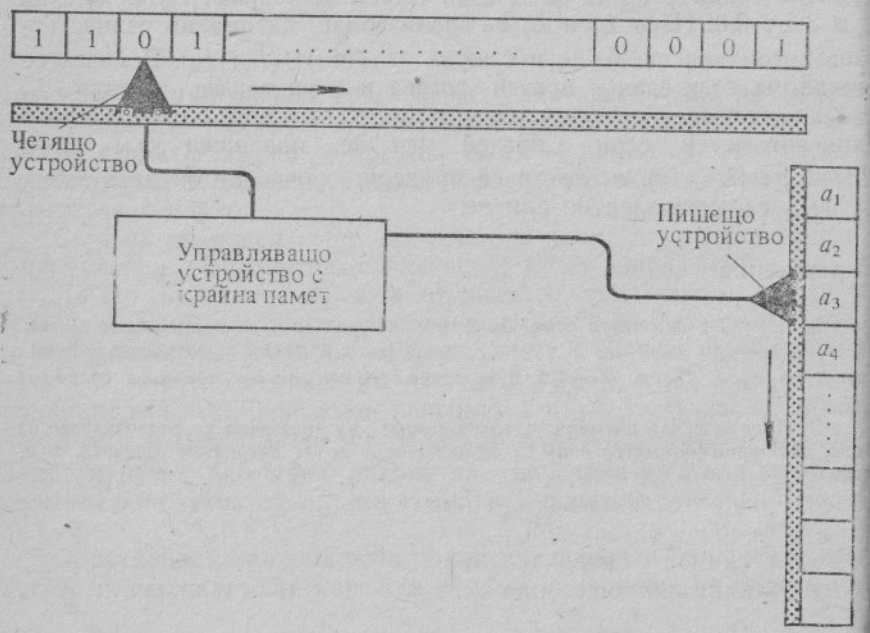
Крайните автомати въведохме като устройства за разпознаване на входни думи. Резултат от работата на крайния автомат беше последното вътрешно състояние на автомата след прочитане на думата, като при това ние се интересуваме само от това, дали състоянието е заключително или не (което съответствува на отговорите „да", „не", „принадлежи" — „не принадлежи").

Можем да модифицираме малко дадената схема за краен автомат и да позволим допълнително автоматът да печата на отделна изходна лента символи от някаква (изходна) азбука при всеки такт в зависимост от прочетения символ и вътрешното си състояние. Схематично това е дадено на фиг. 28.

Резултат от работата на такъв краен автомат ще бъде думата, напечатана на изходната лента, четена по посока на движението на печатащото устройство. Такъв автомат, както се вижда, извършва превод от един формален език на друг.

Ще определим следния прост математически модел на устройство за превод на формални езици — **автомат на Мили.**

Фиг. 28




**Автомат на Мили** с входна азбука $\Sigma$ и изходна азбука $\Delta$ ще наричаме наредена шесторка

$$\mathbf{M} = (K, \Sigma, \Delta, \delta, \lambda, q_0),$$ в която

— $K$ е крайно множество от вътрешни състояния, наречено азбука на вътрешните състояния;

— $\Sigma$ е входна азбука: елементите на $\Sigma$ се наричат входни символи;

— $\Delta$ е изходна азбука; елементите на която се наричат изходни символи;

— $\delta$ е функция на преходите, която на всяка наредена двойка („вътрешно състояние", „входен символ") съпоставя „вътрешно състояние";

— $\lambda$ е функция, която на всяка наредена двойка („вътрешно състояние", „входен символ") поставя в съответствие „изходен символ";

— $q_0$ е начално състояние, $q_0 \in K$.

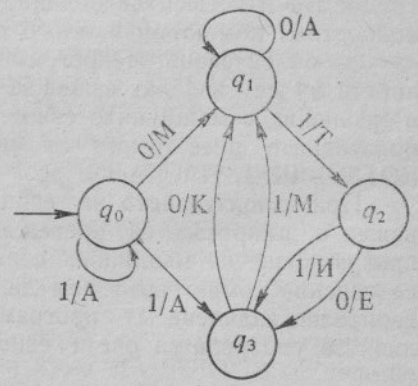
**Работата на автомата на Мили** върху дадена входна дума $b_1 b_2 \ldots b_k$ се определя така: по началното състояние $q_0$ и първия входен символ $b_1$ функцията $\delta$ определя следващото състояние $p_1$, а функцията $\lambda$ определя първата буква на изходната дума $\lambda(q_0, b_1)$. По състоянието $p_1$ и входната буква $b_2$ функцията $\delta$ определя следващото състояние $p_2$, а функцията $\lambda$ — следващия изходен символ $\lambda(p_1, b_2)$ и т. н. Накрая по състоянието $p_k$ и входната буква $b_k$, функцията $\delta$ определя последното състояние $p_k$, а функцията $\lambda$ — последния изходен символ $\lambda(p_{k-1}, b_k)$. Резултатът от работата на автомата на Мили е думата

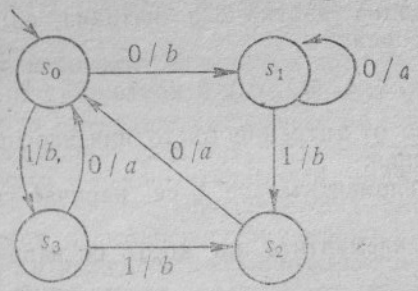
$$\lambda(q_0, b_1)\lambda(p_1, b_2) \ldots \lambda(p_{k-1}, b_k).$$

Ако на входа е подадена празната дума $\Lambda$, автоматът на Мили дава на изхода само празната дума.

Както при крайните автомати, на всеки автомат на Мили можем да съпоставим съответна диаграма на преходите с тази разлика, че реброто, свързващо върховете $p$ и $q$ ще означаваме с два символа: символ от входната азбука, който осъществява прехода

Фиг.29






Фиг. 30

от състоянието $q$ в състоянието $p$, т. е., за който $\delta(q, b)=p$, и символ $a$ от изходната азбука, за който $\lambda(q, b)=a$. Това ще отбелязваме така $b|a$. Превода на входната дума ще получим, като се движим по стрелките, определяни от нея и четем последователно изходните символи.

Диаграмата на фиг. 29 определя автомат на Мили. Този автомат преобразува входната дума 0010101100 в изходната дума МАТЕМАТИКА, а входната дума 11010 в изходната **дума** ИМАТЕ.

Като следващ пример можем да разгледаме автомата на Мили от фиг. 30. Автоматът **M** преобразува думата 000111 в думата *baabaa*, а думата 1111 в *bbaa*.

### 2.10. ЛЕКСИЧЕСКИ АНАЛИЗ

Всеки компютър „разбира" един определен специфичен за него език, наречен машинен език, в който се използуват само символите 0 и 1. В края на 40-те години, програмистите са пишели своите програми само на машинен език и тези програми са представлявали дълги редици от нули и единици. Машинният език е твърде далеч от езиците, които се използуват за общуване между хората, поради това написването на програма на машинен език и проверката на нейната вярност представляват извънредно трудоемка и неефективна дейност, изискваща добро познаване на конкретния компютър и много време.

За облекчаване на програмирането и повишаване на ефективността му се създават езици за програмиране от по-високо ниво в сравнение с машинните езици. Такива са например широко използуваните днес езици за програмиране ФОРТРАН, АЛГОЛ, КОБОЛ, PL/1, ПАСКАЛ и др.

При използуването на езиците за програмиране извънредно важен е въпросът за превеждането на програмите от езика за програмиране на машинния език на компютъра. Естествено е да се изисква компютърът сам да извършва тази дейност по предварително зададена му програма за превод. Поради това приемливи за употреба са онези езицц за програмиране, които са максимално удобни за хората и позволяват разработването на сравнително прости „автоматични преводи" на машинен език. Обикновено програмите -преводачи, се наричат транслатори.



И така, **транслаторът** е програма, която превежда програми, написани на определен език за програмиране, в програми, написани на разбираем за компютъра език. За да бъде извършен този превод, необходимо е да се направи анализ на зададената програма, като преди всичко трябва да се разграничат в текста на входната програма различните градивни единици на дадения език на програмиране — така наречените **символи на езика за програмиране.** Такива символи са **служебните думи** на езика, **идентификаторите,** числата, различните **знаци за операции и разделителите.**

Символите на езика за програмиране образуват неговата лексика и поради това разпознаването и представянето в определен вид на символите от дадена програма се нарича **лексически анализ,** а програмата, която извършва тези дейности — **лексически анализатор.**

След лексическия анализ транслаторът провежда т. нар. **синтактичен анализ,** чрез който определя дали превежданата програма е съставена от синтактически правилни конструкции на дадения език за програмиране.

На практика често пъти лексическите анализатори в транслаторите се реализират чрез крайни автомати, което показва, че символите на разглеждания език за програмиране могат да се породят от елементите на азбуката на езика с помощта на автоматна граматика.

Ще покажем как може да бъде реализиран лексически анализатор за един примерен език. Ще се ограничим в разглеждането на неголям и извънредно прост език. Ще се интересуваме само от символите на езика, защото лексическият анализ има за цел именно разпознаването на символите на дадения език за програмиране. Да разгледаме един примерен език със следните символи:

— разделители и знаци за операции: ␣, /, +, —, *, (,), // и **;

— служебни думи — НАЧАЛО, АКО, КРАЙ;

— идентификатори: произволни крайни редици от букви и цифри, започващи с буква, с изключенне на служебните думи;

— цели числа.

Примерният език удовлетворява следните условия:

— поне една шпация трябва да разделя един от друг различните символи на езика — идентификаторите, целите числа и служебните думи;

— в символите на езика не могат да се появяват шпации — например не се допуска служебната дума НАЧАЛО да се запише НАЧ␣АЛО;

— идентификаторите в езика са крайни редици от букви и цифри, започващи непременно с буква;

— целите числа в езика са крайни редици от цифри;

— допустимо е използването на **коментари** в програмите, написани на примерния език. Всеки коментар започва със символите /* (в примерния език това са два символа — символът / и символът *). Всеки коментар завършва със символите */.

Лексическият анализатор трябва да може да разпознава и елиминира коментарите от програмата. По-долу е даден пример за един коментар:

/* ␣ ЩЕ ␣ МЕ ␣ РАЗПОЗНАВА ␣ ЕДИН ␣ КРАЕН ␣ АВТОМАТ ␣ * /

Лексическият анализатор разпознава входните символи на езика и строи вътрешно представяне за всеки символ. В повечето случаи това е цяло число с фиксирана дължина. Този подход за вътрешно представяне е избран поради факта, че в другите части на транслатора е много по-ефективно да се обработват тези цели числа, а не низове с променлива дължина, каквито са всъщност символите на езика.

Във вътрешното представяне на езика някое число ще означава „идентификатор", друго число ще означава „цяло число" и т. н. По този начин във вътрешното представяне всички идентификатори ще се означават с едно и също число. Това е естествено, понеже „идентификатор" е терминален символ за синтактичния анализатор и поради това е без значение какъв конкретен идентификатор се среща във всеки конкретен случай. При изпълнението на програмата обаче се налага да се работи с конкретния идентификатор. Поради това лексическият анализатор трябва да подаде на изхода си всъщност две величини — едната е вътрешното представяне на даден символ, а втората — е фактическият символ или препратка към него (можем да си мислим, че фактическите символи от дадена програма се записват от лексическия анализатор в таблици и той подава на изхода число, посочващо мястото на символа в определена таблица).

Сега ще покажем как може да се построи краен автомат, който да представлява лексически анализатор на разглеждания по-горе примерен език.

Нека разгледаме автомата $\mathbf{M}=(K, \Sigma, \delta, q_0, F)$. Азбуката на автомата е следната:

$\Sigma=\{A, B, C, \ldots, Z, Б, Г, Д, \ldots, Ю, Я, 0, 1, \ldots, 9, /, +, -, *, (,)\}$. Автоматът има следните вътрешни състояния:

$$K=\{q_0, q_1, \ldots, q_8, q_E\},$$



като състоянията $q_7$ и $q_E$ са заключителни. Функцията $\delta$ се дефинира чрез диаграмата от фиг. 31.

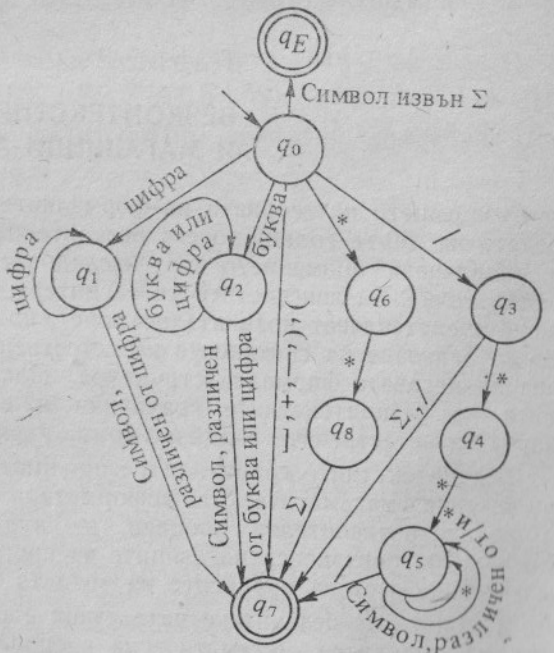
Символите /, //, /* , */, * , ** трябва да бъдат обработвани по различен начин и по тази причина в автомата са „отделени" състояния за тяхната обработка.

Автоматът **M** започва да работи над началото на символа, който ще бъде анализиран, и попада в заключително състояние, когато е анализиран и разпознат целият символ. Счита се, че един символ е разпознат, когато на входа се появи елемент на азбуката $\Sigma$, който не може да принадлежи на разглеждания символ.

Чрез състоянието $q_1$ се разпознават числата, чрез $q_2$ — идентификаторите, чрез $q_6$ — символа * и т. н.

Освен такъв краен автомат в лексическия анализатор би трябвало да се включват множество допълнителни програми за четене, за записване на изхода резултата от анализа с краен автомат и т. н.

Фиг. 31




Даденатa диаграма на състоянията (фиг. 31) показва схематично как може да се прави анализ на символите на езика.

Примерният автомат **M** има един сериозен недостатък: не различава служебните думи. Този автомат може да се „подобри", като след разпознаването на някакъв идентификатор се прави проверка, дали това не е някоя от служебните думи или като се конструира друг автомат (по-сложен), който разпознава служебните думи отделно от идентификаторите.

*„. . . онова, което въобще може да бъде казано, може да се каже ясно, за което е невъзможно да се говори, за него трябва да се мълчи“.*

Л. Витгенщайн — *„Логико-философски трактат“.*

# Глава 3

## БЕЗКОНТЕКСТНИ ЕЗИЦИ И МАГАЗИННИ АВТОМАТИ

От създаването на теорията на формалните езици и граматики в средата на 50-те години досега безконтекстните езици най-силно са привличали вниманието и от изследователска, и от приложна гледна точка. За лингвистите те са интересни преди всичко с това, че представляват сравнително прост, но достатъчно силен метод за задаване на синтаксиса на естествените езици и за изследване на неговата формална структура. Плодотворни са приложенията на безконтекстните граматики и езици в компютърния анализ на естествените езици и в компютърната лингвистика.

Безконтекстните граматики и езици имат многобройни и важни приложения в математическата информатика: и като формален модел на редица изчислителни процеси, и като основно средство за описване на синтаксиса на езиците за програмиране, и като важна съставна част на процесите на тяхната транслация.

Теорията на безконтекстните езици е силно разклонена и богата на резултати математическа дисциплина. В тази глава ще опишем само някои основни свойства и характеристики на безконтекстните езици, зададени чрез пораждащи системи — безконтекстните граматики и чрез разпознаватели — магазинни автомати. Накрая ще скицираме отделни техни приложения.

### 3.1. ДЪРВО НА ИЗВОДА В БЕЗКОНТЕКСТНИТЕ ГРАМАТИКИ

Според даденото от нас определение една пораждаща граматика е безконтекстна, ако всяко нейно правило е от вида

$$A \longrightarrow \omega,$$

където $A$ е нетерминален символ, а $\omega$ е непразна дума от нетерминални и терминални символи. $\omega$ може да бъде празна дума само в случая, когато $A$ е началният символ и този символ не се среща отдясно на никое правило.

Наличието на един единствен символ отляво на всяко правило дава възможност за едно естествено, еднозначно определено, графично представяне на всеки извод в безконтекстните граматики. Основната идея се състои в това, че на всяко правило съпоставяме графично изображение, а при всеки извод на мястото на заместваната буква поставяме графичното изображение на прилаганото правило.

Да вземем например безконтекстната граматика

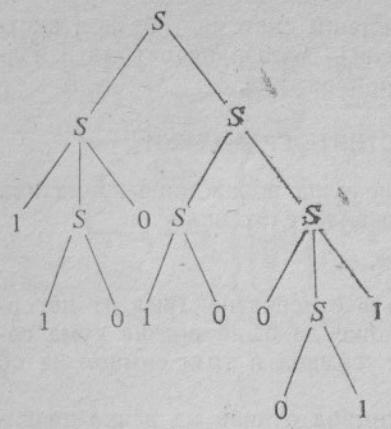
$$\Gamma_1 = (\{0,1\}, \{S\}, S, \{S \to SS, S \to 0S1, S \to 1S0, S \to 01, S \to 10\}).$$

Правилата можем да изобразим графично по следния начин (фиг. 32):


Фиг. 32

За извода $S \vdash SS \vdash 1S0S \vdash 1S0SS \vdash 1100SS \vdash 110010S \vdash 1100100S1 \vdash 1100100011$ получаваме изображението, на фиг. 33.

Очевидно е, че този начин на графично изобразяване отразява специфични свойства на безконтекстните граматики и в общия случай не е приложим към контекстните граматики и граматиките от общ вид.




Фиг. 33

Структурите, които се появяват при описаното графично представяне, в теорията на графите се наричат дървета, поради това нататък ще говорим за дърво на даден извод в дадена граматика. Тъй като дърветата ще играят съществена роля в математическите ни разсъждения, необходимо ни е точно определение на това понятие.

**Дърво** ще наричаме всяко крайно множество от върхове, свързани с ориентирани ребра (единият връх е начало, а другият — край на реброто), което отговаря на следните условия:

а) съществува точно един връх, наречен **корен** на дървото, който не е край на ребро;

б) всеки връх, освен корена, е край точно на едно ребро;

в) от корена до всеки връх има път от ориентирани ребра.

Лесно се проверява, че даденият пример на графично изобразяване на извод в граматиката $\Gamma_1$ удовлетворява тези условия, ако схващаме ориентацията на ребрата от горе на долу.

Оттук нататък ориентацията на ребрата специално няма да отбелязваме, а ще рисуваме дървото, като поставяме корена отгоре, а дъгите ориентираме от горе на долу.

**Преки потомци** на даден връх ще наричаме всички върхове, които са край на ориентирано ребро с начало дадения връх. Един връх е **потомък** на даден връх, ако съществува редица от върхове, всеки член на която е пряк потомък на предишния, започваща от дадения връх и завършваща в този връх.

Ще считаме, че преките потомци на един връх са винаги линейно наредени от ляво на дясно, а ако един връх е по-наляво от друг, то и всички негови потомци са по-наляво от потомците на другия връх.

Ясно е, че за всеки два върха на едно дърво или единият е потомък на другия, или единият се намира по-наляво от другия.

**Листа** на дървото ще наричаме върховете, които не са начало на ребро. Тъй като никой лист не може да бъде потомък на друг лист, листата на всяко дърво могат да се подредят от ляво на дясно. Така подредени листата на едно дърво образуват неговата **корона.**



И накрая **клон** на едно дърво, започващ от даден връх с потомци, ще наричаме този връх и всички негови потомци заедно със свързващите ги ребра. За короните на два клона можем да кажем, че или едната корона включва другата, или едната е по-наляво от другата.

Както вече се убедихме, в разгледания пример на графично изобразен извод получаваме дърво. В това дърво обаче има още нещо: върховете му са означени по определен начин с терминални и нетерминални символи на граматиката $\Gamma_1$. Това ни дава основание да въведем следното определение: едно дърво ще наричаме **дърво на извод в дадена граматика** $\Gamma=(V, W, S, P)$, ако са изпълнени следните условия:

а) коренът на дървото е означен с $S$;

б) всеки връх, който не е лист, е означен с нетерминален символ от $W$;

в) листата са означени с терминални и нетерминални символи, т. е. с букви от азбуката $V \cup W$;

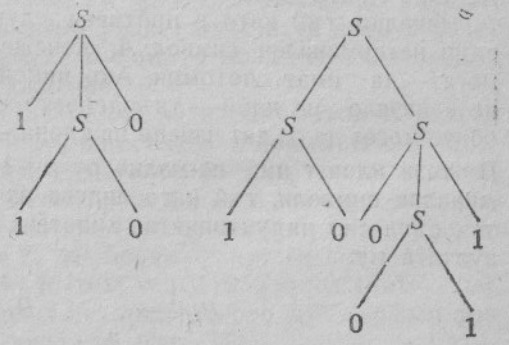
г) ако преките потомци на произволен връх, означен с $A$, изброени от ляво на дясно, са означени с $B_1, \ldots, B_n$, тогава в граматиката $\Gamma$ има правило $A \to B_1 \ldots B_n$.

**Резултат от дадено дърво на извод** ще наричаме думата, която се получава, като прочетем последователно символите на короната му. По същия начин се определя и резултатът от произволен клон.

Примерът, който вече разгледахме, според даденото от нас определение, е дърво на извод в граматиката $\Gamma_1$. Два негови клона са дадени на фиг. 34.

В първия клон двете единици са по-наляво от двете нули, горната единица е по-наляво от долната. Короната на цялото дърво се образува от листата означени последователно с 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, а на двата изобразени клона короните се състоят съответно от 1, 1, 0, 0, и 1, 0, 0, 0, 1, 1. Резултат от даденото дърво на извод, както лесно може да се забележи, е думата, 1 1 0 0 1 0 0 0 1 1.

Фиг. 34




Да разгледаме сега каква е връзката между възможните дървета на

извод в една безконтекстна граматика и възможните изводи в тази граматика.

Нека е дадена безконтекстната граматика $\Gamma=(V, W, S, P)$. Ще докажем, че **на всяко дърво на извод в дадената граматика Г с резултат — произволна дума $\omega$, съответствува извод $S\overset{\Gamma}{\models}\omega$ в тази граматика, където $S$ е началният символ на граматиката.**

За целта ще покажем, че е в сила по-общо твърдение — на всеки клон на дърво на извод в граматиката Г (клонът, за разлика от дървото на извод, може да започва с произволен нетерминален символ), започващ от върха $A$ и с резултат — думата $\omega$, съответствува извод $A\overset{\Gamma}{\models}\omega$ в граматиката Г.

Това твърдение ще докажем чрез индукция по броя на върховете в клона, означени с нетерминални символи.

Ако в клона има само един нетерминален символ $A$, той трябва да е в началото на клона, а останалите върхове $B_1, \ldots, B_k$ ще бъдат негови преки потомци, тъй като те са означени с терминални символи и следователно никой от тях не може да има потомци. Но тогава според пункт 2 на определението в граматиката Г има правило $A\to B_1 \ldots B_k$ (листата са изброени от ляво на дясно), т. е. получаваме извода: $A\overset{\Gamma}{\models}B_1 \ldots B_k$.

Да допуснем, че твърдението е вярно за всички клони на дърветата на извод в Г, които имат не повече от $n\geqq 1$ върхове, означени с нетерминални символи. Да разгледаме произволен клон на дърво на извод в Г с $n+1$ върхове, означени с нетерминални символи. Нека този клон да започва от върха, означен с нетерминалния символ $A$ и неговите преки потомци да бъдат последователно означени с $B_1, \ldots, B_k$. Тогава в граматиката има правило $A\to B_1 \ldots B_k$. Сред символите $B_1, \ldots, B_k$ непременно има нетерминални, тъй като в противен случай в клона ще има само един нетерминален символ $A$, понеже терминалните символи не могат да имат потомци. Ако никой от върховете $B_1, \ldots, B_k$ не е начало на клон — твърдението е доказано. Някои от тях обаче могат да бъдат начало на клони — нека това са $B_{i_1}, \ldots, B_{i_m}$. В тези клони има по-малко от $n+1$ върхове, означени с нетерминални символи, тъй като липсва началният връх $A$. За всеки от тях съгласно индуктивната хипотеза съществува извод на резултата му:

$$B_{i_1}\overset{\Gamma}{\models}\alpha_1, \ldots, B_{i_m}\overset{\Gamma}{\models}\alpha_m.$$



Но тогава получаваме следния извод на резултата на даденото дърво:

$$A\vdash B_1 B_2 \ldots B_k=\omega_1 B_{i1}\omega_2 B_{i_2}\omega_3 \ldots \omega_{m-1} B_{im}\omega_m \overset{\Gamma}{\models} \omega_1\alpha_1\omega_2 Bi_2$$
$$\omega_3 \ldots \omega_{m-1} B_{i_m}\omega_m \overset{\Gamma}{\models}\omega_1\alpha_1\omega_2\alpha_2\omega_3 \ldots\omega_{m-1}B_{im}\omega_m \overset{\Gamma}{\models}\ldots\overset{\Gamma}{\models}$$
$$\omega_1\alpha_1\omega_2\alpha_2\omega_3 \ldots\omega_{m-1}\alpha_m\omega_m.$$

В приведеното доказателство се забелязва следната особеност на извода, съответствуващ на дадено дърво: в индуктивния преход беше без значение в какъв ред ще заместваме нетерминалните символи $B_{i_1}, \ldots, B_{i_m}$ със съответните им изводи. Ние приехме да ги заместваме от ляво на дясно. Ако това правим на всяка стъпка при индуктивното построяване на извода, ще получим извод, в който винаги ще се замества най-лявата нетерминална буква. Такъв извод ще наричаме **ляв извод в граматиката** Г. Аналогично може да се построи **десен извод в граматиката** Г.

Допълнително получихме, че **за всяко дърво на извод в Г с резултат — думата $\omega$ съществува ляв (десен) извод $S\models\omega$ в граматиката** Г.

Обратно, **по всеки извод в дадена безконтекстна граматика може да се построи дърво на извода в тази граматика, на което резултат да бъде извежданата непразна дума.** Това става по същия начин, както в разглеждания вече пример. Отначало построяваме дървото, съответствуващо на първото правило, прилагано в извода, след това към върха, който отговаря на заместваната дума, добавяме клона за второто правило и т. н.

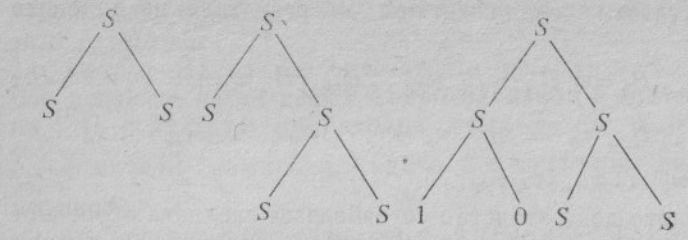
Доказателството на това, че на всеки извод $S\models\omega\,(\omega\neq\Lambda)$ в дадена безконтекстна граматика Г може да се съпостави дърво на извод в тази граматика с резултат — думата $\omega$, се извършва аналогично на предишното доказателство. И тук трябва да се вземе произволен извод $A\models\omega\,(\omega\neq\Lambda)$ и да се докаже чрез индукция по дължината му, че за този извод съществува клон с начален връх $A$ и резултат — думата $\omega$. Самото доказателство предоставяме на читателя за упражнение.

Оттук в частност получаваме, че **за всеки извод $S\models\omega$ в една безконтекстна граматика съществува и ляв (десен) извод на $\omega$ от $S$ в тази граматика.** Действително този извод можем да намерим, като построим дървото на дадения извод, а по него след това построим съответния ляв (десен) извод.

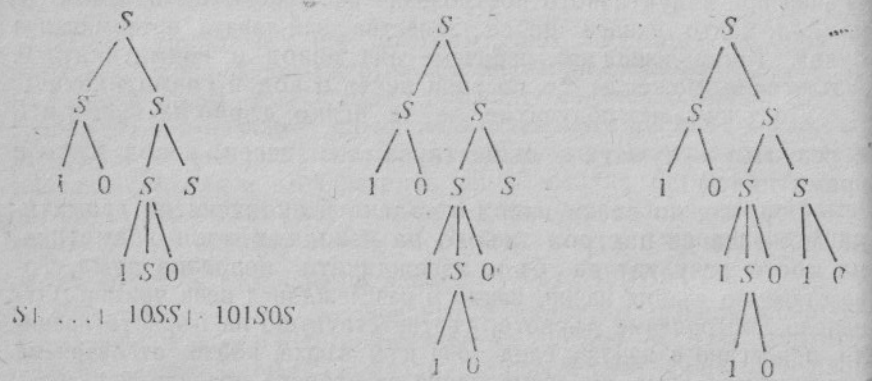
Например в граматиката $\Gamma_1$ за извода

$$S\vdash SS\vdash SSS\vdash 10SS\vdash 101S0S\vdash 101100S\vdash 10110010$$

можем да построим последователно съответното му дърво с резултат 10110010 по начина, даден на (фиг. 35).



$S \vdash SS$ , $S \vdash SS \vdash SSS$, $S \vdash SS \vdash SSS \vdash 10SS$ ...



$S \vdash \ldots \vdash 10SS \vdash 101S0S$

$S \vdash \ldots \vdash 101S0S \vdash 101100S$

$S \vdash \ldots \vdash 101100S \vdash 10110010$

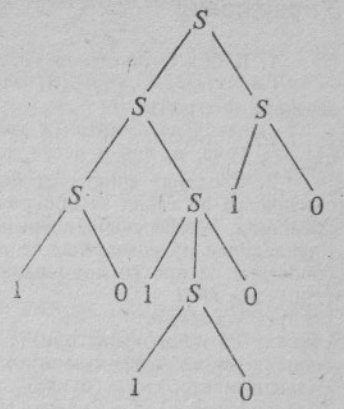
Фиг. 35

Може ли да се случи в една безконтекстна граматика да има няколко различни дървета с един и същ резултат?

Има граматики, в които това не е възможно, но лесно могат да се намерят и такива, в които този факт е налице. Наличието на две или повече дървета на извод в дадена граматика с един и същ резултат означава, че в тази граматика има дума, която се извежда с два или повече различни леви (десни) изводи. Такива граматики, както и породените от тях езици, се наричат **неоднозначни.** Така определена неоднозначността е свойство на граматиката, а



не на езика, породен от нея. Всеки език може да се зададе с различни граматики, някои от тях могат да бъдат неоднозначни, а други да бъдат еднозначни. Има обаче езици, за които има само неоднозначни граматики. Ако един неоднозначен език не може да се зададе с еднозначна граматика, той се нарича **съществено неоднозначен**. Такъв например е безконтекстният език $L=\{a^i b^j c^k;\ i=j$ или $j=k,\ i,\ j,\ k$ — естествени числа $\}$.



Фиг. 36

Граматиката $\Gamma_1$, която вече неколкократно разглеждахме, е неоднозначна, тъй като освен построеното от нас дърво на извод с резултат 1 0 1 1 0 0 1 0, може да се построи и друго дърво на извод със същия резултат фиг. 36.

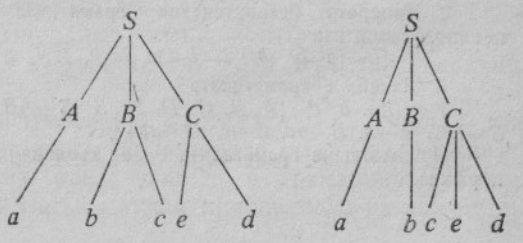
В лингвистиката на неоднозначните граматики съответствува понятието **синтактична неоднозначност.** В естествените езици този тип неоднозначност се среща твърде често. Достатъчно е да се вземе изречение с дълга поредица от съществителни с предлози, за да възникне лесно синтактична неоднозначност. Следният прост пример „Аристотел прочете диалозите на Платон“ може да се тълкува с два синтактично неоднозначни смисъла: „Диалозите на Платон са прочетени от Аристотел“ и „Аристотел е чел на Платон някакви диалози“.

Изречение от типа „Аристотел прочете диалозите на Платон“ се поражда например, от следната безконтекстна граматика

$$(\{a, b, c, d, e\},\ \{S, A, B, C\},\ S,\ \{S \to ABC,\ A \to a,\ B \to bc,\ C \to ed,\ C \to ced\}),$$

в която за думата $abced$ има две различни дървета на извод (фиг. 37):

Ако заменим $a$ с „Аристотел“, $b$ — с „прочете“ $c$—с „диалозите“, $e$—с „на“ и $d$ — с „Платон“, получаваме горното изречение.



Фиг. 37

## Упражнения

**1.** В безконтекстната граматика

$\Gamma=(\{x, +, *, (, )\}, \{S, A, B\}, S, \{S\to A, S\to S+A, A\to A * B, A\to B, B\to x, B\to(S)\})$ постройте:

а) извод за думата $(x+x) * (x * x+x)$; б) дърво на извод в $\Gamma$ с резултат същата дума; в) ляв и десен извод за построения извод.

**2.** Следният списък от безконтекстни правила са взети от описанието на синтаксиса на езика за програмиране АЛГОЛ (нетерминалните символи са заградени с ъглови скоби, например, — ⟨идентификатор⟩, а знакът : : = се чете „по определение е“ и може да се разглежда като →):

⟨описание на прости променливи⟩ : : = ⟨тип⟩⟨списък на идентификатори⟩
⟨тип⟩ : : = ***real***
⟨тип⟩ : : = ***integer***
⟨списък на идентификатори⟩ : : = ⟨идентификатор⟩
⟨списък на идентификатори⟩ : : = ⟨списък на идентификатори⟩, ⟨идентификатор⟩
⟨идентификатор⟩ : : = ⟨буква⟩
⟨идентификатор⟩ : : = ⟨идентификатор⟩ ⟨буква⟩
⟨идентификатор⟩ : : = ⟨идентификатор⟩ ⟨цифра⟩
⟨буква⟩ : : = *a*
. . . . . .
⟨буква⟩ : : = *z*
⟨цифра⟩ : : = 0
. . . . . .
⟨цифра⟩ : : = 9

Намерете извод на терминалната дума ***real***$p3192,q1273$ от нетерминалния символ ⟨описание на прости променливи⟩ и постройте дърво на този извод.

**3.** Дадена е безконтекстната граматика $\Gamma=(\{a\}, \{S\}, S, \{S\to SS, S\to a\})$

Покажете, че тази граматика е неednозначна и за всяка дума от езика ѝ намерете броя на различните леви изводи на тази дума.

**4.** Покажете, че безконтекстната граматика

$\Gamma=(\{a, b\}, \{S, A, B\}, S, \{S\to bA, S\to ab, A\to a, B\to b, A\to aS, B\to bS, A\to bAA, B\to aBB\})$ е неednозначна.

**5.** Дайте пример на синтактично неednозначно изречение от българския език и постройте съответните различни дървета на синтактичния му анализ.

**6.** Дадена е граматиката

$\Gamma=(\{$***if, then, else, for, do***$, a, b, c\}, \{S, A\}, S, \{S\to a, S\to$ ***if*** $b$ ***then*** $A, S\to$ ***if*** $b$ ***then*** $A$ ***else*** $S, A\to$ ***for*** $c$ ***do*** $S, A\to a\})$.

а) Покажете, че тази граматика е неednозначна, като намерите различни леви изводи на терминалната дума

***if*** $b$ ***then for*** $c$ ***do if*** $b$ ***then*** $a$ ***else*** $a$.

б) Намерете еднозначна граматика, която също да поражда езика $L(\Gamma)$.

**7.** Намерете безконтекстна граматика, която да поражда съществено неednозначния език

$L=\{a^i b^j c^k : i=j$ или $j=k$, $i, j, k$ са естествени числа$\}$.

**8.** Дадена е граматиката

$\Gamma=(\{a, b, c\}, \{A, B, C, D, S\}, S, \{S\to AB, A\to aA\ B\to bBc, C\to cC, D\to aDb, S\to DC, A\to a, B\to bc, C\to c, D\to ab\})$.

Покажете, че граматиката $\Gamma$ е неednозначна и опишете езика, който тя поражда.





За да можем да твърдим, че някакъв език не е безконтекстен, трябва да докажем, че не може да има безконтекстна граматика, която да го поражда. Това обаче не е никак просто, тъй като безконтекстните граматики са безкрайно много и не е възможно да проверяваме за всяка от тях дали поражда дадения език или не. За това ще използуваме следния метод: ще покажем, че всички безкрайни безконтекстни езици притежават определено свойство и ще намерим безкраен контекстен език, който няма това свойство. Такъв език не може да бъде безконтекстен.

Този резултат, който позволява да дадем единна характеристика на безкрайните безконтекстни езици, е твърде полезен в теорията на безконтекстните езици. От него например ще изведем не само съществуването на контекстен език, който не е безконтекстен, но и разрешимостта на редица алгоритмични проблеми за безконтекстните езици. За това ще го формулираме отделно чрез следната теорема.

**Нека $L$ е произволен безконтекстен език. Съществува естествено число $n$, зависещо от $L$, такова че, ако в $L$ има дума $\omega$ с дължина по-голяма от $n$, тази дума може да се представи по следния начин:**

$$\omega=\alpha\beta\gamma\mu\eta,$$

**при което думата $\gamma$ не е празна, а думите $\beta$ и $\mu$ не са едновременно празни. Освен това на езика $L$ ще принадлежат и всички думи от вида $\alpha\beta^k\gamma\mu^k\eta$ при $k\geq 0$.**

Нека $L$ е произволен безконтекстен език. Тогава има безконтекстна граматика $\Gamma$, която поражда езика $L$. Ще считаме, че в $\Gamma$ няма правила от вида $A\to B$ ($A$, $B$ са нетерминални символи). По-късно ще покажем, че за всяка безконтекстна граматика може да се построи еквивалентна на нея безконтекстна граматика, която да удовлетворява това условие.

Да означим броя на нетерминалните символи в граматиката $\Gamma$ с $l$.

Във всяко дърво от корена до кой и да е лист води единствен път. Сред тези пътища има такъв, който минава през най-много върхове. Броят на върховете в този път ще наричаме **височина на дървото.**

С нарастването на височината в дърветата на извод в граматиката $\Gamma$ нараства и ширината на короната им, т. е. дължината на техния резултат. Това е така, защото в граматиката $\Gamma$ няма правила от вида $A\to B$. В такъв случай прибавянето на нов нетер-

минален връх в пътя между корена и някой лист, означава добавянето и на нов клон, чийто листа ще увеличават дължината на резултата.

Обратно, не може думите в езика Г да растат неограничено по дължина, а височината на техните дървета на извод да бъде ограничена. Действително да допуснем, че височините на всички дървета на извод в граматиката Г са ограничени например от числото $m$. Всяко правило от граматиката Г има в дясната си част краен брой символи, например — не повече от $k$ на брой. Тогава листата на дървото с височина $m$ не могат да бъдат повече от $k^m$. А това е в противоречие с неограниченото нарастване на дължините на думите от езика $L$.

Получаваме следния резултат: за всяка дадена безконтекстна граматика Г може да се намери естествено число, такова че, ако дължината на коя и да е дума, пораждана от Г, е по-голяма от това число, тогава височината на дървото й за извод ще бъде по-голяма от $l+2$. ($l$ е броят на нетерминалните символи в граматиката Г.).

Нека $n$ от условието на теоремата е това естествено число.

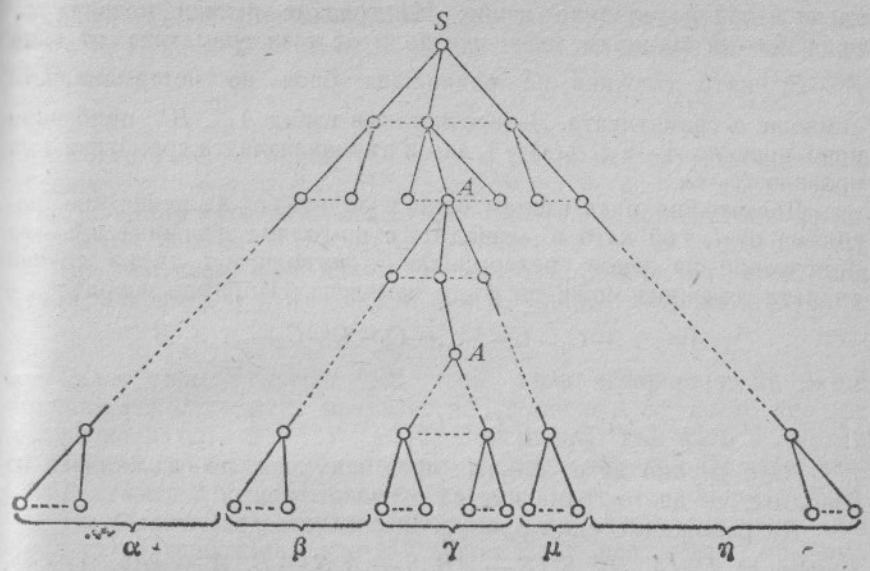
Да вземем произволна дума $\omega$ от езика $L$ с дължина, по-голяма от $n$. Тогава височината на нейното дърво на извод в граматиката Г ще бъде по-голяма от $l+2$. Получаваме, че в дървото на извод на думата $\omega$ в граматиката Г има път с не по-малко от $l+2$ върхове. В този път само един връх — листото, е означен с терминален символ, а останалите поне $l+1$ върхове са отбелязани с не повече от $l$ нетерминални символа. Това означава, че поне два върха от този път са означени с един и същ нетерминален символ — например $A$, тъй като в граматиката Г има $l$ нетерминални символа. Дървото на извод на думата $\omega$ може да се представи схематично, както на фиг. 38.

Клонът, започващ от горния връх $A$, има резултат $\beta\gamma\mu$, а клонът, започващ от долния връх $A$, е с резултат $\gamma$. Думите $\beta$ и $\mu$ не могат да бъдат едновременно празни. Това е така, защото в граматиката няма правила от вида $A \to B$, а правило от вида $A \to a$ не е приложено към горния връх $A$, понеже този връх има непреки потомци. Тогава от горния връх $A$ излиза поне още едно ребро освен реброто, водещо към долния връх $A$. Очевидно е, че думата $\gamma$ не е празна, тъй като потомък на долния връх $A$ ще бъде поне един лист.

Получихме, че ако дължината на думата $\omega$ е по-голяма от определено за Г число $n$, тази дума може да се представи така:

$$\omega = \alpha\beta\gamma\mu\eta,$$






Фиг. 38

при което $\gamma$ не е празната дума, а думите $\beta$ и $\mu$ не са едновременно празни.

От разгледаното дърво на извод в граматиката Г можем да получим ново дърво на извод в тази граматика, ако на мястото на клона с начало долният връх $A$ поставим клон, който е еднакъв с клона на горния връх $A$.

Резултат на новото дърво на извод ще бъде думата $\alpha\beta^2\gamma\mu^2\eta$.

Ако повтаряме тази процедура, ще получаваме дървета на извод в Г с резултати от вида: $\alpha\beta^k\gamma\mu^k\eta$, $k \geqq 1$.

Обратно, ако в първоначалното дърво на мястото на клона, започващ от горния връх $A$, поставим клон, еднакъв с клона на долния връх $A$, ще получим дърво на извод в Г с резултат думата $\alpha\gamma\eta$, т. е. $\alpha\beta^0\gamma\mu^0\eta$.

С това теоремата е доказана за безконтекстните граматики, в които няма правила от вида $A \to B$.

За всяка безконтекстна граматика с правила от вида $A \to B$ може да се построи еквивалентна безконтекстна граматика без такива правила.

Еквивалентната граматика без правила от вида $A \to B$ получаваме, като на дадената граматика изменим единствено правила-

та на извод по следния начин: Изключваме всички правила от вида $A \to B$. Намираме всички изводи от тази граматика от вида $A \overset{\Gamma}{=} B$, чиято дължина не надвишава броя на нетерминалните символи в граматиката. За всеки такъв извод $A \overset{\Gamma}{=} B$ прибавяме ново правило $A \to \alpha$ с $\partial(\alpha) > 1$, ако в първоначалната граматика има правило $B \to \alpha$.

Достатъчно е да вземем само изводите с дължина, не по-голяма от $l$, тъй като в изводите с по-голяма дължина ще има повторение на някои нетерминални символи, и в такъв случай тяхната дължина може да бъде намалена. Например изводът

$$A \vdash . \; . \; . \vdash C_1 \vdash C \vdash . \; . \; . \vdash C_2 \vdash C \vdash C_3 \vdash . \; . \; . \vdash B$$

може да се съкрати така

$$A \vdash . \; . \; . \vdash C_1 \vdash C \vdash C_3 \vdash . \; . \; . \vdash B.$$

Не е трудно да се докаже чрез индукция по дължината на изводите, че двете граматики са еквивалентни.

Да разгледаме един пример. В следната граматика

$\Gamma = (\{a, b\}, \{S, A, B\}, S, \{S \to AB, S \to A, S \to B, A \to aAb, A \to ab, B \to Bb, B \to b, S \to \Lambda\})$

можем да изключим по описания начин всички правила, заменящи един нетерминален символ с друг нетерминален символ. Възможните изводи на един нетерминален символ от друг нетерминален символ са два

$$S \overset{}{=} A \quad \text{и} \quad S \overset{}{=} B.$$

Еквивалентната на Г граматика от интересуващия ни вид има следните правила

| | | | |
|---|---|---|---|
| $S \to AB$ | $B \to Bb$ | $S \to aAb$ | $S \to Bb$ |
| $A \to aAb$ | $B \to b$ | $S \to ab$ | $S \to b$ |

Сега да разгледаме едно важно следствие от доказаната теорема.

Както вече посочихме, този резултат ни позволява да покажем, че **съществува контекстен език, който не е безконтекстен.**

За езика $L = \{a^m b^m c^m, m \geq 1\}$ вече посочихме контекстна граматика, която да го поражда. Сега ще покажем, че не може да съществува безконтекстна граматика, която също да поражда $L$.

Да допуснем, че $L$ е безконтекстен език. Както се вижда, $L$ е безкраен език и в него ще има думи, чиято дължина е по-голяма от кое да е фиксирано естествено число. Това ни дава основание да приложим доказаната теорема към думи от езика $L$



с дължина по-голяма от фиксираното за граматиката на езика число $n$. Нека $a^p b^p c^p$ е една такава дума. Тогава тя може да се представи така

$$\alpha\beta\gamma\mu\eta,$$

при което $\gamma$ не е празната дума, а $\beta$ и $\mu$ не са едновременно празни.

Да видим сега от какви букви могат да се състоят думите $\beta$ и $\mu$.

Да допуснем, че в някоя от тях има повече от един вид букви, например в $\beta$ има букви $a$ и $b$. Според теоремата думите

$$\alpha\beta^k \gamma\mu^k \eta, \quad k \geq 0$$

ще бъдат също от езика $L$. В тези думи обаче при $k > 1$ ще получим неколкократно редуване на букви $a$ и $b$, което ще се дължи на думата $\beta^k$. Но това е невъзможно, тъй като в думите от езика $L$ буквите $a$, $b$ и $c$ стоят в определен ред. И така, всяка от думите $\beta$ и $\mu$ може да се състои само от един вид букви.

Но съгласно теоремата думата $\alpha\gamma\eta$, която получаваме при $k=0$, също трябва да бъде от езика $L$. В нея обаче ще има повече от този вид букви, които не участвуват в думите $\beta$ и $\mu$. Това е невъзможно, понеже в думите от езика $L$ винаги има равен брой букви $a$, $b$ и $c$.

Остава $\beta$ и $\mu$ да са едновременно празни, което пък противоречи на теоремата.

В такъв случай нашето допускане, че $L$ е безконтекстен език, не е вярно.

И така, $L = \{a^m b^m c^m, m \geq 1\}$ не може да бъде безконтекстен език.

## Упражнения

**1.** За безконтекстната граматика

$\Gamma = (\{x, +, *, (, )\}, \{S, A, B\}, S, \{S \to A, S \to S+A, A \to A * B, A \to B, B \to x, B \to (S)\})$

постройте еквивалентна безконтекстна граматика без правила от вида $X \to Y$ ($X$, $Y$ са нетерминални символи).

**2.** Докажете, че следните езици не са безконтекстни:

а) $L = \{0^m 1^m 0^m ; \; m \geq 1\}$; б) $L = \{a^{2^n} ; \; n \geq 0\}$;

в) $L = \{\omega ; \; \omega \in \{a, b, c\}^*$ и в $\omega$ има равен брой букви $a$, $b$, $c\}$;

г) $L = \{a^p ; \; p$ — просто число$\}$.

**3.** Докажете, че езикът

$L=\{a^i b^j c^k$ ; $i=j$ или $j=k$, $i, j, k$ са естествени числа$\}$ е съществено неедноз̀начен.

Указание: Използувайте доказаната теорема.

### 3.3. НЯКОИ СВОЙСТВА НА БЕЗКОНТЕКСТНИТЕ ЕЗИЦИ

Когато изследвахме автоматните езици, установихме, че операциите обединение, произведение, итерация, сечение и допълнение, приложени към автоматни езици, дават отново автоматни езици. Тук ще изследваме как тези операции действуват върху безконтекстните езици. Както ще видим, някои от тях, приложени към безконтекстни езици, дават отново безконтекстни езици, за други обаче това няма да е вярно.

Да вземем два произволни безконтекстни езика: $L'$ с граматика $\Gamma'=(V', W'', S', P')$ и $L''$ с граматика $\Gamma''=(V'', W'', S'', P'')$. Както и преди ще считаме, че нетерминалните азбуки на двете граматики нямат общи символи — в противен случай ще преименуваме нетерминалните символи на едната граматика така, че това условие да е изпълнено. Ясно е, че тази процедура няма да се отрази на пораждания език.

Да започнем с обединението на езиците $L'$ и $L''$. Ще докажем, че тяхното обединение е безконтекстен език, като построим безконтекстна граматика, която да го поражда. За целта да вземем следната граматика:

$$\Gamma=(V' \cup V'', W' \cup W'' \cup \{S\}, S, P),$$

в която $S$ е нов нетерминален символ, различен от символите в азбуките $W'$ и $W''$. Множеството $P$ на правилата на граматиката $\Gamma$ включва в себе си всички правила от множествата $P'$ и $P''$ с изключение на правилата $S' \to \Lambda$ и $S'' \to \Lambda$ (ако такива има), новите правила $S \to S'$ и $S \to S''$ и накрая — правило $S \to \Lambda$, ако поне в една от граматиките $\Gamma'$ и $\Gamma''$ има подобно правило.

Ясно е, че граматиката $\Gamma$ е безконтекстна, тъй като както старите, така и новодобавените правила имат нужния вид, а правилото $S \to \Lambda$ е допустимо, понеже $S$ не се среща в дясната част на никое от посочените правила.

Граматиката $\Gamma$ поражда всички думи от езиците $L'$ и $L''$. За празната дума това е изпълнено — ако в една от граматиките $\Gamma'$ и $\Gamma''$ има правило $S' \to \Lambda$ или $S'' \to \Lambda$, подобно правило има и в граматиката $\Gamma$. Да вземем произволна непразна дума $\omega$, която да принадлежи на един от езиците $L'$ и $L''$ — например на $L'$. Това означава, че в граматиката $\Gamma'$ има извод $S' \models \omega$. От този извод и правилото $S \to S'$ получаваме извод в граматиката $\Gamma$: $S \vdash S' \models \omega$, тъй като всички правила от граматиката $\Gamma'$, използувани в извода, са правила и на граматиката $\Gamma$.



Граматиката $\Gamma$ поражда само думи от езиците $L'$ или $L''$. Действително, нека $S \models \omega$ е произволен извод, пораждащ някаква непразна дума $\omega$ от езика $L(\Gamma)$. Към $S$ можем да приложим само едно от правилата $S \to S'$ и $S \to S''$. Да приемем, че това е правилото $S \to S''$. Граматиката $\Gamma$ е безконтекстна, т. е. лявата част на всяко правило се състои от един нетерминален символ. В такъв случай получаваме, че към $S'$ можем да приложим само някое от правилата на граматиката $\Gamma''$. По този начин извеждаме нова дума, в която отново всички нетерминални символи ще бъдат от $W''$. Към всеки от тях можем да приложим само правило от граматиката $\Gamma''$ и т. н. В такъв случай изводът $S \models \omega$ изглежда така

$$S \vdash S' \models \omega,$$

като частта $S' \models \omega$ е изцяло в граматиката $\Gamma''$. Но тогава $\omega$ е от езика $L''$. Аналогично разсъждаваме, когато началното правило на извода $S \models \omega$ е $S \to S'$.

Празната дума принадлежи на $L(\Gamma)$, само ако тя принадлежи на някой от езиците $L'$ или $L''$.

И така, $L(\Gamma)=L' \cup L''$, т. е. **обединението на два безконтекстни езици е винаги безконтекстен език.**

Сега да разгледаме как стои въпросът с произведението на езиците $L'$ и $L''$.

Да приемем отначало, че езиците $L'$ и $L''$ не съдържат празната дума. Да построим следната безконтекстна граматика:

$$\Gamma=(V' \cup V'', W' \cup W'' \cup \{S\}, S, P).$$

И тук $S$ е нов нетерминален символ. Множеството $P$ се състои от всички правила от $P'$ и $P''$ и от едно ново правило $S \to S'S''$. Като се използуват разсъжденията, които проведохме в предишния случай, лесно може да се покаже, че граматиката $\Gamma$ поражда всички думи от произведението на езиците $L'$ и $L''$ и само тях.

Нека сега само един от езиците $L'$ и $L''$ съдържа празната дума, например $L'$. Тяхното произведение може да се изрази по следния начин

$$L'.L''=((L'-\{\Lambda\}).L'') \cup L''.$$

$L''$ и $L'-\{\Lambda\}$ са два безконтекстни езика, несъдържащи празната дума, тяхното произведение ще бъде безконтекстен език, а обединението на двата безконтекстни езика $(L'-\{\Lambda\}).L''$ и $L''$, както вече показахме, е също безконтекстен език.

В случая, когато и двата езика $L'$ и $L''$ съдържат празната дума, можем отново да изразим тяхното произведение чрез произведение на безконтекстни езици, които не съдържат празната дума, и операцията обединение:

$$L' . L'' = ((L' - \{\Lambda\}) . (L'' - \{\Lambda\})) \cup L' \cup L'' \cup \{\Lambda\}.$$

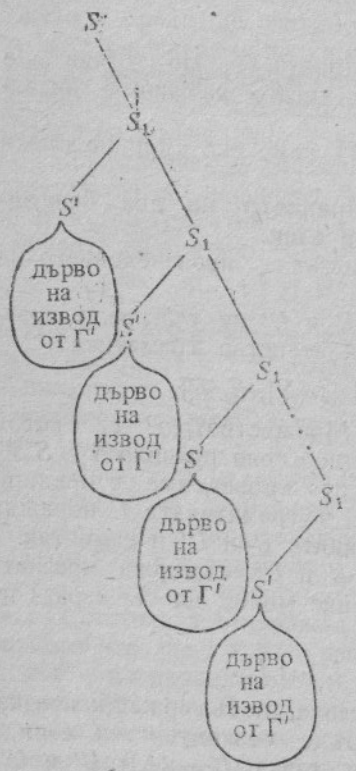
И така, получаваме, че **произведението на два безконтекстни езика е винаги безконтекстен език.**

Следната безконтекстна граматика поражда итерацията на произволно взетия безконтекстен език $L'$:

$$\Gamma = (V', W' \cup \{S, S_1\}, S, P).$$

$S$ и $S_1$ са нови нетерминални символи, а множеството $P$ се състои от всички правила от $P'$ с изключение на правилото $S' \to \Lambda$, ако такова има, и от новите правила $S \to S_1$, $S_1 \to S' S_1$, $S_1 \to S'$, $S \to \Lambda$.

Фиг. 39




Граматиката $\Gamma$ поражда само думи от вида $\omega_1\ \omega_2 \ldots \omega_k$, където $\omega_1, \ldots, \omega_k$ са думи от $L'$. Лесно се вижда, че всяко дърво на извод в $\Gamma$ изглежда, както на фиг. 39.

Обратно, всяка дума от вида $\omega_1 \ldots \omega_k$, $\omega_1, \ldots, \omega_k$ от $L'$ може да се породи от $\Gamma$. Тъй като $\omega_1, \ldots, \omega_k$ са от $L'$, за тях има изводи:

$$S' \models \omega_1, \ldots, S' \models \omega_k.$$

От тези изводи можем да построим следния извод в граматиката $\Gamma$:

$$S \vdash S_1 \vdash S' S_1 \models \omega_1, S_1 \vdash \omega_1\ S'\ S_1 \models \omega_1\ \omega_2\ S_1 \models \ldots \models \omega_1 \ldots \omega_k$$

Празната дума очевидно също се поражда в граматиката $\Gamma$. Следователно $L(\Gamma) = (L')^*$, т. е. **итерацията на безконтекстен е винаги безконтекстен език.**

Съществена разлика между автоматните езици и безконтекстните се появява при операциите сечение и допълнение. Сега ще покажем, че **сечението на безконтекстните езици не винаги е безконтекстен език.**



Да вземем безконтекстния език $L_1 = \{a^n\ b^n\ c^m;\ m \geqq 1,\ n \geqq 1\}$ с пораждаща граматика

$$\Gamma_1 = (\{a, b, c\}, \{S, A, B\}, S, \{S \to AB,\ A \to aAb,\ A \to ab,\ B \to cB,\ B \to c\})$$

и безконтекстния език $L_2 = \{a^m\ b^n\ c^n;\ m \geqq 1,\ n \geqq 1\}$ с пораждаща граматика

$\Gamma_2 = (\{a, b, c\}, \{S, A, B\}, S, \{S \to AB,\ B \to bBc,\ B \to bc,\ A \to Aa,\ A \to a\})$ и да образуваме тяхното сечение. Тъй като в езика $L_1$ буквите $a$ и $b$ трябва да бъдат от една и съща степен, а в езика $L_2$ буквите $b$ и $c$ трябва да бъдат от една и съща степен, то в сечението и трите букви ще имат еднакви степени, т. е.

$$L_1 \cap L_2 = \{a^n\ b^n\ c^n;\ n \geqq 1\}.$$

За този език обаче вече доказахме, че не е безконтекстен език.

Сега лесно можем да покажем, че **допълнението на безконтекстен език не винаги е безконтекстен език.** Да допуснем обратното. Тогава на това основание допълненията $\overline{L_1}$ и $\overline{L_2}$ на езиците $L_1$ и $L_2$ ще бъдат безконтекстни езици, а оттук и тяхното обединение $\overline{L_1} \cup \overline{L_2}$. Но тогава безконтекстен език ще бъде и допълнението му $\overline{\overline{L_1} \cup \overline{L_2}}$.

Обаче $\overline{\overline{L_1} \cup \overline{L_2}} = L_1 \cap L_2$ а за това сечение вече показахме, че не е безконтекстен език. Така получихме противоречие, т. е. допускането не е вярно.

### Упражнения

1. Нека $L$ е произволен безконтекстен език. Докажете, че езика $L' = \{O(\omega);\ \omega \notin L\}$ е винаги безконтекстен език.

2. Докажете или опровергайте твърдението, че разликата на два безконтекстни езика е винаги безконтекстен език.

3. Дайте примери за безконтекстни езици, чийто сечения не са безконтекстни езици.

## 3.4. АЛГОРИТМИЧНИ ПРОБЛЕМИ, СВЪРЗАНИ С БЕЗКОНТЕКСТНИТЕ ЕЗИЦИ

Безконтекстните езици обикновено задаваме чрез техните крайни описания — безконтекстните граматики. Това поставя редица въпроси за свойствата на пораждените от тях езици: кога езикът, зададен чрез някаква граматика е празен, краен или безкраен, кога празната дума му принадлежи, дали произволна дума е от този език или не, кога две граматики пораждат един и същ език и т. н. Всеки от тези въпроси можем да поставим и по-общо:

съществува ли алгоритъм, който по безконтекстната граматика да определя дали пораждания от нея език е празен, краен или безкраен, съдържа ли или не празната дума и т. н. Такива алгоритмични проблеми могат да бъдат поставени за всяко свойство, което притежават едни или други безконтекстни езици.

Вече разгледахме въпроса за съществуването на алгоритъм, който да определя дали произволна дума се поражда от дадена безконтекстна граматика, когато показахме, че контекстните езици са рекурсивни. Тъй като всеки безконтекстен език може да се разглежда и като контекстен, алгоритъмът, който определя дали произволна дума е от даден контекстен език или не, ще действува и за безконтекстните езици. Има, разбира се, значително по-прости и по-ефикасни алгоритми, които да решават същия този въпрос специално за безконтекстните езици.

Друг алгоритмичен проблем, който тривиално се решава за класа на безконтекстните езици, е проблемът за принадлежността на празната дума към езика. Според даденото от нас определение една безконтекстна граматика поражда празната дума, само ако в нея има правило $S \to \Lambda$, разбира се, при съответните условия. Така че наличието или отсъствието на такова правило в граматиката решава въпроса за принадлежността на празната дума към пораждания език.

Да разгледаме сега въпроса дали езикът, пораждан от произволна безконтекстна граматика е празен или не, т. е. алгоритмичния проблем за разпознаване на празнотата на безконтекстните езици.

Да вземем произволна безконтекстна граматика Г без правила от вида $A \to B$ ($A$, $B$ са нетерминални символи). Както знаем, за всяка безконтекстна граматика може да се намери еквивалентна безконтекстна граматика, която да удовлетворява това условие. Да означим с $l$ броя на нетерминалните символи в граматиката Г.

Да допуснем, че граматиката Г поражда някаква дума $\omega$. Да разгледаме нейното дърво на извод в $\Gamma$. Ако височината му е по-голяма от $l+1$, тогава в най-дългия път ще има поне два върха, означени с един и същ нетерминален символ, които са начало на вписани един в друг клони. Можем да заместим по-големия клон с по-малкия и ще получим ново дърво на извод в Г с резултат някаква дума от терминални символи. Действувайки по този начин, можем да намерим дърво на извод в Г, чиято височина не надвишава $l+1$, а резултатът му е терминална дума. Получихме следния резултат: ако граматиката Г поражда някаква дума, тя поражда и дума, чиято височина на дървото на извод не надвишава $l+1$.



Оттук можем да изведем следния алгоритъм за проверка на това дали произволна безконтекстна граматика Г поражда празен език или не. Строим всички дървета на извод в Г с височина 2 и сред тях търсим с резултат терминална дума. Ако такава дума има — езикът $L(\Gamma)$ не е празен. Ако такава дума няма, строим всички дървета на извод в Г с височина 3. Проверяваме дали има дърво на извод с резултат терминална дума и продължаваме нататък, ако такава дума няма. Накрая построяваме всички дървета на извод в Г с височина $l+1$. Ако и сред тях не намерим дърво с резултат терминална дума, това означава, че езикът $L(\Gamma)$ е празен.

С това доказахме, че **съществува алгоритъм, който да проверява дали езикът, породен от произволна безконтекстна граматика е празен, или не.**

Да преминем сега към алгоритмичния проблем за разпознаване на това, дали езикът, задаван от произволна безконтекстна граматика Г е краен или безкраен. За целта ще използуваме доказаната от нас теорема за безкрайните безконтекстни езици.

Ако граматиката Г поражда дума $\omega$, чието дърво на извод има височина по-голяма от $l+1$, тогава дължината на $\omega$ ще бъде по-голяма от фиксираното за Г число $n$. Прилагаме доказаната теорема и получаваме за думата $\omega$ представянето $\omega = \alpha\beta\gamma\mu\eta$, $\gamma \neq \Lambda$, $\beta\mu \neq \Lambda$, при което всички думи от вида

$$\alpha\beta^k\gamma\mu^k\eta, \quad k \geq 0,$$

са също от езика $L(\Gamma)$. Това означава, че $L(\Gamma)$ е безкраен език.

Обратно, нека $L(\Gamma)$ е безкраен език. Тогава в него ще има думи с произволно голяма дължина, а следователно и с произволно голяма височина на дърветата им на извод в Г, например по-голяма от $l+1$. Ще покажем, че в такъв случай в $L(\Gamma)$ ще има думи с височина на дърветата им на извод по-голяма от $l+1$ и по-малка или равна на $2l+1$. Както вече казахме, в $L(\Gamma)$ има думи с височина на дърветата им на извод по-голяма от $l+1$. Ако някое от тях е с височина по-голяма или равна на $2l+1$, твърдението е доказано. Да допуснем обаче, че всяко от тези дървета на извод е с височина по-голяма от $2l+1$. Да изберем от тях онова дърво, което има най-малка височина. В пътя от корена му до някой лист с дължина, равна на височината му, има поне два върха, означени с еднакви нетерминални символи. Да вземем тези два върха, които са най-близко един до друг. Клона на по-високия от тях можем да заменим с клона на по-ниския и ще получим път, чиято дължина е по-скъсена най-много с $l$ върха. Това е така, понеже в пътя от единия връх до другия това са единстве-

ните еднакво означени върхове. Ако при тази процедура височината не се намали, действуваме по същия начин и със следващия път, с дължина равна на височината на дървото. Пътищата от корена до листата на даденото дърво са ограничен брой и след краен брой стъпки ще стигнем до дърво на извод в Г с по-малка височина, която обаче не сме намалили повече от $l$. Получаваме противоречие с допускането, че сред дърветата на извод в Г, по-високи от $l+1$, няма дърво с височина по-малка или равна на $2l+1$.

И така, ако езикът $L(\Gamma)$ е безкраен, в него ще има дума с дърво на извод в граматиката Г, височината на което е по-голяма от $l+1$ и по-малка или равна на $2l+1$.

Оттук получаваме следния алгоритъм за проверка на това дали произволна безконтекстна граматика Г поражда краен или безкраен език. Строим дърветата на извод в Г с височина по-голяма от $l+1$ и по-малка или равна на $2l+1$ ($l$ е броят на нетерминалните символи в граматиката Г). Ако сред тях има дърво с резултат терминална дума — езикът $L(\Gamma)$ е безкраен. Ако няма такова дърво — езикът $L(\Gamma)$ е краен.

С това доказахме, че **проблемът за разпознаване на крайността или безкрайността на безконтекстните езици е алгоритмично разрешим.**

Докато проблемът за разпознаване на еквивалентността на автоматните граматики, както вече показахме, е разрешим, **проблемът за разпознаване на еквивалентността на безконтекстните граматики обаче е алгоритмично неразрешим.** Това означава, че не съществува алгоритъм, който по две произволни безконтекстни граматики да определя дали те са еквивалентни, или не.

**Алгоритмично неразрешими са още редица проблеми за безконтекстни езици: дали произволна безконтекстна граматика е неоднозначна, дали поражда автоматен език, дали поражда даден фиксиран език, дали поражда множеството от всички терминални думи и т. н.**

### Упражнения

**1.** Като използувате дадения алгоритъм, определете дали безконтекстната граматика

$$\Gamma=(\{a, b\}, \{S, A, B\}, S, \{S\to AB,\ B\to ABa,\ A\to a\})$$

поражда празен език.

**2.** Докажете алгоритмичната разрешимост на проблема за разпознаване на това дали нетерминален символ от произволна безконтекстна граматика участвува в пораждането на някоя дума.

**3.** Докажете, че всеки безконтекстен език върху азбука, съставена от една буква, е автоматен език.



## 3.5. МАГАЗИННИ АВТОМАТИ

Безконтекстните езици също имат свой тип разпознаватели — това са така наречените **магазинни автомати.** Магазинните автомати представляват недетерминирани крайни автомати, към които е добавена потенциално безкрайна външна памет във формата на **магазин,** или още — **стек.** Това съществено изменя разпознаващата сила на автоматите и всъщност организацията на външната памет определя възможността за разпознаване на един или друг език.

В магазина елементите на запомнена информация са подредени последователно един след друг и за автомата е достъпна само информацията от първия елемент. Този елемент може да се изтрива от паметта и на негово място, евентуално, да се добавя нова крайна редица от елементи на информация. Достъпен ще бъде само първият елемент от новодобавената редица. Ако не е добавена нова информация, тогава на мястото на първия елемент застава следващият след него. В този смисъл магазинът прилича на пълнител с патрони или на добре наредена опашка.

По-точно, за да бъде един автомат магазинен, той трябва да притежава следните неща. Преди всичко трябва да има **управляващо устройство с крайна памет,** което може да се намира в краен брой вътрешни състояния, и **входна лента,** на която е записана дума от входни символи, които също са краен брой. Управляващото устройство е свързано с **магазин (стек),** който представлява записана върху лента дума от символи на някаква азбука — **магазинна азбука.** Тъй като думите четем от ляво на дясно, най-левият символ на магазинната дума ще наричаме първи. Управляващото устройство може да разпознава или, с други думи, да прочита първия символ на магазинната дума и на негово място отпред на магазинната дума да записва произволни думи от магазинната азбука. Поради това считаме, че отпред лентата на магазина може произволно да се продължава.

Магазинният автомат също като крайните автомати работи в дискретни моменти от време — тактове. В началния момент от време управляващото устройство се намира в едно фиксирано вътрешно състояние — **начално състояние,** а в магазина има един фиксиран символ — **начален символ на магазина,** който бележи края или още — дъното на магазина. Във всеки следващ момент управляващото устройство може да извършва два вида действия. Първо, то може да прочете входния символ, върху който се намира, и първия магазинен символ от магазина. В зависимост от прочетения входен символ, вътрешното си състояние

и първия магазинен символ получава краен брой възможности за избор на следващо състояние и магазинна дума (може и празна), с която да замести първия магазинен символ. След като премине в ново вътрешно състояние и постави отпред на магазина нова дума, управляващото устройство преминава към следващия отдясно входен символ.

Второ, управляващото устройство може да не чете текущия входен символ и да не се придвижва към следващия отдясно входен символ, а в зависимост от вътрешното си състояние и първия магазинен символ да получава краен брой възможности за избор на ново вътрешно състояние и на магазинна дума, с която да замести първия магазинен символ. При това действие управляващото устройство се занимава само с магазина си и не чете входни символи. Ще отбележим, че този вид действие може да се извършва и след прочитане на последния символ от входната дума.

Магазинният автомат не може да премине към следващия такт, когато няма указание за следващи действия или когато магазинът му е празен.

В случай че всички следващи тактове са възможни, магазинният автомат спира, след като прочете последния входен символ и извърши след това избраните от него възможни действия.

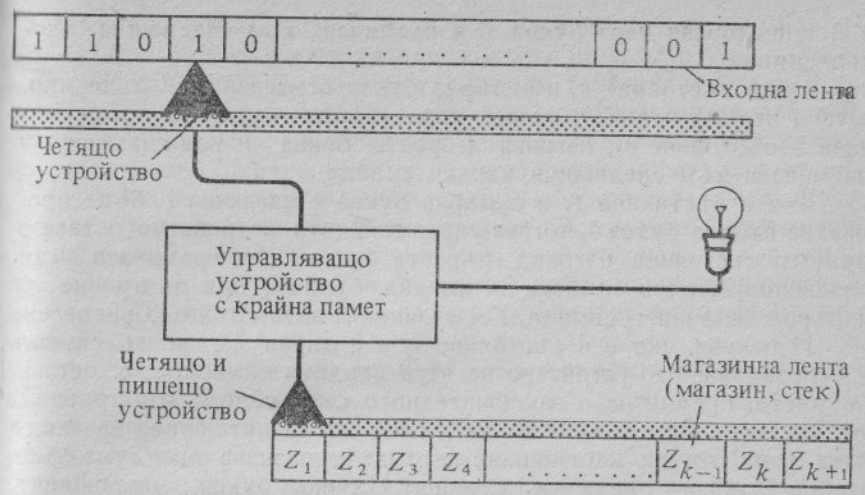
**Резултат от работата** на магазинния автомат могат да бъдат две неща. Едното е последното вътрешно състояние на управляващото устройство след прочитане на цялата входна дума — дали то е от определен вид вътрешни състояния, наречени заключителни. Второто е състоянието на магазина след прочитане на цялата входна дума — дали той е празен или не.

В съответствие с това има два начина да определим кога един магазинен автомат разпознава дадена входна дума — чрез заключително състояние и чрез празен магазин.

Схематично магазинният автомат можем да си представим по начина, показан на фиг. 40.

Ще опишем един магазинен автомат, който да разпознава чрез заключително състояние думите от вида $a^n b^n$, $n \geqq 0$. При наличието на външна памет — магазин, лесно можем да си представим работата на автомата. Магазинният автомат ще може да работи, само ако зададената входна дума започва с буква $a$. Преминавайки през тези букви $a$, управляващото устройство ще запомни техния брой, като за всяка буква $a$, през която е минало, ще поставя някакъв знак в магазина. Щом изчерпи буквите $a$, при всяко преминаване през буква $b$ ще изтрива по един от добавените в магазина знаци. В случай че знаците свършват точно






Фиг. 40

с прочитането на цялата дума, тогава управляващото устройство преминава в заключително състояние. В противен случай или магазинният автомат не е определен, или ще бъде накрая в незаключително състояние.

Входната азбука вече е определена — $\{a, b\}$. Необходими са три вътрешни състояния $s_0$, $s_1$, $s_2$ — начално състояние, състояние, в което ще бъде автоматът, преминавайки през букви $a$, и състояние, в което ще бъде автоматът, когато преминава през букви $b$. Заключително състояние ще бъде началното — $s_0$, тъй като магазинният автомат трябва да може да разпознава и празната дума. За магазина са необходими два символа — начален символ $Z$ и символ $Y$, който ще добавяме и изтриваме. В съответствие с тези означения и изложения принцип на действие получаваме следните правила за работа на построения от нас магазинен автомат:

Магазинният автомат започва работа в състояние $s_0$ и с единствен символ $Z$ в магазина.

Ако входната буква е $a$, управляващото устройство преминава в състояние $s_1$, на мястото на $Z$ в магазина поставя $YZ$, т. е. добавя $Y$, и се придвижва върху следващия входен символ. За входна буква $b$ магазинният автомат не е определен.

Ако в състояние $s_1$ и с първа буква в магазина $Y$ бъде прочетена входна буква $a$, управляващото устройство остава в същото състояние, на мястото на първия символ в магазина поставя

$YY$, т. е. добавя нова буква $Y$, и преминава към следващия входен символ.

Ако в състояние $s_1$ и с първа буква в магазина $Y$ бъде прочетена входна буква $b$, тогава управляващото устройство преминава в състояние $s_2$, изтрива първата буква $Y$ от магазина и преминава към следващия входен символ.

Ако в състояние $s_2$ и с първа буква в магазина $Y$ бъде прочетена входна буква $b$, тогава управляващото устройство остава в същото състояние, изтрива първата буква $Y$ и преминава към следващия входен символ. За входна буква $a$ при състояние $s_2$ и първи магазинен символ $Y$ магазинният автомат не е определен.

И накрая, ако е в състояние $s_2$ и с първи магазинен символ $Z$, управляващото устойство не чете входния символ, т. е. остава на място, преминава в заключителното състояние $s_0$ и изтрива $Z$ от магазина. Ако това е станало след последната буква на входната дума, тогава магазинният автомат разпознава тази дума. Ако това е станало преди да свършат всички букви, магазинният автомат ще бъде неопределен за входната дума, тъй като при празен магазин не може да продължи работа.

Не е трудно да се покаже, че така определеният магазинен автомат ще разпознава точно думите от вида $a^n b^n$, за които $n \geqq 0$.

### Упражнения

**1.** Опишете магазинен автомат, който да разпознава думите от вида $a^n c b^n$, $n \geqq 0$.

**2.** Опишете магазинен автомат, който да разпознава думите от вида $0^n 1^{2n} 0^n$, за които $n \geqq 0$.

**3.** Опишете магазинен автомат, който да разпознава думите, съставени от нули и единици, в които броят на нулите е равен на броя на единиците.

## 3.6. ОПРЕДЕЛЕНИЕ И СВОЙСТВА НА МАГАЗИННИТЕ АВТОМАТИ

След като разгледахме един модел на магазинен автомат, можем да дадем точно определение на това понятие. Магазинният автомат, който ще разглеждаме тук, е недетерминиран. От гледна точка на разпознаваните езици недетерминираният и детерминираният магазинни автомати не са еквивалентни. Докато чрез детерминирани магазинни автомати се разпознават само част от безконтекстните езици, то чрез недетерминираните магазинни автомати, както ще покажем по нататък, се разпознават всички безконтекстни езици и само тези езици.

**Магазинен недетерминиран автомат M** ще наричаме наредената седморка



$M = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, в която

— $K$ е **азбука на вътрешните състояния;**

— $\Sigma$ е **азбука на входните символи (входна азбука);**

— $\Gamma$ е **азбука на магазинните символи (магазинна азбука);**

— $\delta$ е функция, която на наредени тройки от вида („вътрешно състояние", „входен символ", „магазинен символ") или от вида („вътрешно състояние", „$\Lambda$", „магазинен символ") съпоставя крайни множества от наредени двойки („вътрешно състояние", „дума върху магазинната азбука");

— $q_0$ е от азбуката $K$ и се нарича **начално състояние;**

— $Z_0$ е от азбуката $\Gamma$ и се нарича **начален магазинен символ;**

— $F$ е подмножество на $K$ и се нарича **множество на заключителните състояния.**

Функцията $\delta$ определя два вида действия:

$\delta(q,a,Z)=\{(p_1\omega_1), \ldots, (p_k,\omega_k)\}$, където $q, p_1, \ldots, p_k \in K$, $a, \in \Sigma$, $Z \in \Gamma$, $\omega_1, \ldots, \omega_k \in \Gamma^*$ означава, че магазинният автомат в състояние $q$ с първи магазинен символ $Z$ прочита входния символ $a$ и чрез функцията $\delta$ определя възможните варианти за ново състояние и магазинна дума, която да замени $Z$. След това магазинният автомат преминава в едно от възможните състояния $p_i$, заменя $Z$ с думата $\omega_i$ и се придвижва надясно към следващия входен символ. Когато $\omega_i$ е празната дума, това означава, че се отстранява буквата $Z$ и първа буква на магазина става следващата буква.

$\delta(q, \Lambda, Z)=\{(p_1, \omega_1), \ldots, (p_k, \omega_k)\}$, където $q, p_1, \ldots, p_k \in K$, $Z \in \Gamma$, $\omega_1, \ldots, \omega_k \in \Gamma^*$, $\Lambda$ е празната дума, означава, че магазинният автомат не прочита входния символ и не се придвижва след това надясно, а в зависимост от вътрешното си състояние и първия магазинен символ чрез функцията $\delta$ определя възможните варианти за ново състояние $p_i$ и магазинна дума $\omega_i$, с която да замени $Z$. След това магазинният автомат преминава в един от възможните варианти. Този вид действия ще наричаме $\Lambda$-действия или $\Lambda$-тактове.

Функцията $\delta$ не е определена при празен магазин.

**Работа на магазинния автомат** върху дадена входна дума представлява поредица от действия, определени от функцията $\delta$ и символите на входната дума.

Ако след прочитане на последния символ на входната дума и извършване на възможните $\Lambda$-действия, магазинният автомат M спре в заключително състояние в поне един от различните възможни варианти на преходи, ще казваме, че M разпознава дума-

та ω чрез заключително състояние. В противен случай — когато не е определен за дадената входна дума или не е спрял накрая в заключително състояние в никой от възможните варианти на преходи, магазинният автомат не разпознава входната дума чрез заключително състояние.

Като заменим думите „заключително състояние“ с „празен магазин“, ще определим кога един магазинен автомат разпознава дадена дума чрез празен магазин.

Множеството от всички думи, които се разпознават от магазинния автомат **M** чрез заключителното състояние, ще означаваме с $T(\mathbf{M})$. Множеството от всички думи, които **M** разпознава чрез празен магазин, ще означаваме с $N(\mathbf{M})$.

Магазинните автомати можем също да описваме чрез тяхната диаграма на преходите. За да получим съответната на даден магазинен автомат **диаграма на преходите**, върхът на вътрешното състояние $q$ ще свързваме с върха на състоянието $p$ чрез насочено ребро, което ще означаваме с $a$, $Z/\omega$, в случай че

$$(p, \omega) \in \delta(q, a, Z).$$

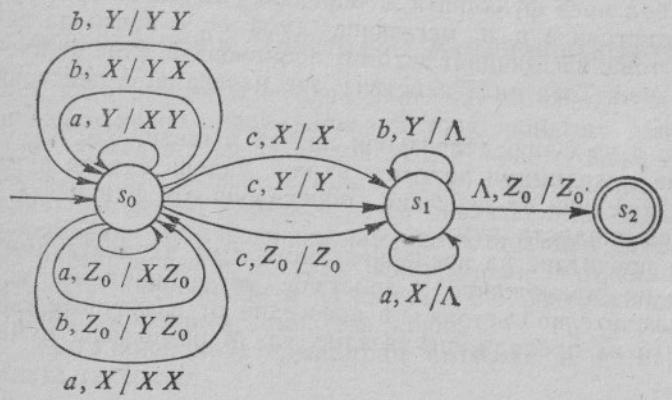
Да вземем например следния магазинен автомат:

$$\mathbf{M}=(\{s_0, s_1, s_2\}, \{a, b, c\}, \{Z, X, Y\}, \delta, s_0, Z, \{s_2\}),$$

като функцията $\delta$ е определена по следния начин:

$\delta(s_0, a, Z)=\{(s_0, XZ)\}$, $\quad \delta(s_1, \Lambda, Z)=\{(s_2, Z)\}$,
$\delta(s_0, b, Z)=\{(s_0, YZ)\}$, $\quad \delta(s_0, a, Y)=\{(s_0, XY)\}$,
$\delta(s_0, c, Z)=\{(s_1, Z)\}$, $\quad \delta(s_0, b, X)=\{(s_0, YX)\}$,
$\delta(s_0, c, Y)=\{(s_1, Y)\}$, $\quad \delta(s_0, c, X)=\{(s_1, X)\}$,
$\delta(s_1, a, X)=\{(s_1, \Lambda)\}$, $\quad \delta(s_0, a, X)=\{(s_0, XX)\}$,
$\delta(s_1, b, Y)=\{(s_1, \Lambda)\}$, $\quad \delta(s_0, b, Y)=\{(s_0, YY)\}$.

Диаграмата на преходите за магазинния автомат е дадена на фиг. 41:




Фиг. 41



Да зададем на магазинния автомат **M** входната дума *aabcbaa*. Осъществяваните от **M** преходи ще записваме във вид на тройки („вътрешно състояние“, „непрочетена част от думата“, „магазинна дума“). Тези тройки определят текущото състояние на автомата и обикновено се наричат **конфигурации**. Преходът от една конфигурация към друга ще означаваме със знака $\vdash$. За думата *aabcbaa* получаваме

$$(s_0, aabcbaa, Z)\vdash(s_0, abcbaa, XZ)\vdash(s_0, bcbaa, XXZ)\vdash(s_0, cbaa, YXXZ)\vdash(s_1, baa, YXXZ)\vdash(s_1, aa, XXZ)\vdash(s_1, a, XZ)\vdash(s_1, \Lambda, Z)\vdash(s_2, Z).$$

Състоянието $s_2$ е заключително и следователно магазинният автомат **M** разпознава думата *aabcbaa* чрез заключително състояние.

Вижда се, че **M** копира във външната си памет — в магазина, частта от думата до първата срещната буква $c$, а след това проверява чрез последователно изтриване дали думата след буквата $c$ представлява обръщане на думата преди буквата $c$. Ако това е така, преминава чрез $\Lambda$-такт в заключително състояние. В противен случай или магазинният автомат **M** не е определен, или **M** завършва работата си в незаключително състояние.

Да разгледаме сега следния магазинен автомат **R**, който се различава от **M** по това, че в неговата функция на преходите, равенството $\delta(s_1, \Lambda, Z)=\{(s_2, Z)\}$ е заменено с $\delta(s_1, \Lambda, Z)=\{(s_2, \Lambda)\}$, а множеството на заключителните състояния е празно. Новият магазинен автомат ще разпознава същите думи, които разпознава **M**, но вече чрез празен магазин.

Нещо повече, за всеки магазинен автомат **M**, разпознаващ чрез заключителни състояния, може да се намери магазинен автомат **R**, разпознаващ чрез празен магазин точно същите думи, които разпознава **M**, т. е. за който $T(\mathbf{M})=N(\mathbf{R})$.

И така, нека е даден произволен магазинен автомат

$$\mathbf{M}=(K, \Sigma, \Gamma, \delta, q_0, Z_0, F),$$

разпознаващ чрез заключителни състояния. Да построим следния магазинен автомат **R**:

$$\mathbf{R}=(K\cup\{s_0, s\}, \Sigma, \Gamma\cup\{X\}, \delta', s_0\ X, \varnothing),$$

в който $s_0$ и $s$ са нови вътрешни състояния, $X$ е нов магазинен символ, а функцията $\delta'$ се определя по следния начин:

— за всяко $q$ от $K$, $a$ от $\Sigma\cup\{\Lambda\}$, $Z$ от $\Gamma$, ако $\delta(q, a, Z)$ съдържа двойката $(p, \omega)$, то и $\delta'(q, a, Z)$ също съдържа двойката $(p, \omega)$;

— $\delta'(s_0, \Lambda, X)=\{(q_0, Z_0X)\}$;

— за всяко $q$ от $F$ и $Z$ от $\Gamma\cup\{X\}$ множеството $\delta'(q, \Lambda, Z)$ съдържа $(s, \Lambda)$;

— за всяко $Z$ от $\Gamma\cup\{X\}$ $\delta'(s, \Lambda, Z)=\{(s, \Lambda)\}$.

Магазинният автомат $R$ действува по следния начин: с начален $\Lambda$-такт преминава в началното състояние на магазинния автомат **M** и поставя отпред на магазина символа $Z_0$, като запазва на дъното на магазина символа $X$. След това $R$ моделира действията на **M**, като всеки път, когато автоматът **M** попада в заключително състояние, автоматът $R$ има възможност да избере, дали да продължи моделирането на действията на **M** или да премине чрез $\Lambda$-такт в състоянието $s$, което изтрива всички символи от магазина. Символът $X$ на дъното на магазина е необходим, за да не може при моделирането на действията на **M** да се изпразни целият магазин без автоматът **M** да е в заключително състояние. Вижда се, че ако автоматът **M** разпознава някоя дума чрез заключително състояние, автоматът $R$ ще я разпознае чрез празен магазин.

От друга страна, магазинният автомат $R$ може да разпознае някоя дума чрез празен магазин, само ако е попаднал в състояние $s$, а в това състояние той може да попадне чрез $\Lambda$-такове, ако преди това е бил в заключително състояние на магазинния автомат **M**, т. е. когато **M** разпознава същата дума чрез заключително състояние.

Може да се покаже, че е вярно и обратното твърдение: за всеки магазинен автомат $R$, разпознаващ чрез празен магазин, може да се намери магазинен автомат **M**, разпознаващ чрез заключителни състояния точно същите думи, които разпознава автоматът $R$.

Нека $\mathbf{R}=(K, \Sigma, \Gamma, \delta, q_0, Z_0, \varnothing)$ е произволен магазинен автомат, разпознаващ чрез празен магазин. Да построим магазинния автомат **M**:

$$\mathbf{M}=(K\cup\{s_0, s\}, \Sigma, \Gamma\cup\{X\}, \delta', s_0, X, \{s\}),$$

като в **M** $s_0$ и $s$ са нови вътрешни състояния, $X$ е нов магазинен символ, а функцията $\delta'$ е определена така:

— за всяко $q$ от $K$, $a$ от $\Sigma\cup\{\Lambda\}$, $Z$ от $\Gamma$ $\delta'(q, a, Z)=\delta(q, a, Z)$;

— $\delta'(s_0, \Lambda, X)=\{(q_0, Z_0X)\}$;

— за всяко $q$ от $K$ множеството $\delta'(q, \Lambda, X)$ съдържа $(s, \Lambda)$.

Магазинният автомат **M** с начален $\Lambda$-такт преминава в началното състояние на магазинния автомат **R** и поставя отпред на магазина началния магазинен символ на **R**, като при това запазва на дъното на магазина символа $X$. След това автоматът **M** моделира действията на автомата **R**. Ако при това автоматът **R** изпразни магазина си, автоматът **M** чрез $\Lambda$-такт преминава в заключителното състояние $s$. И обратно — в заключителното състояние $s$ магазинният автомат **M** може да премине, само ако автоматът **R** е изпразнил за дадената входна дума магазина си. С други думи, магазинните автомати **M** и **R** разпознават едни и същи множества от думи.



## Упражнения

1. Постройте магазинен автомат, който да разпознава всички думи с четна дължина, съставени от 0 и 1, които се четат еднакво от ляво на дясно и от дясно на ляво: а) чрез заключителни състояния; б) чрез празен магазин.

2. Постройте магазинен автомат, който да разпознава чрез заключителни състояния всички думи от вида $a^n b^m a^n, n\geq 1, m\geq 1$.

3. Постройте магазинен автомат, който да разпознава чрез празен магазин всички думи от вида $0^m 1^m 0^n 1^n$, $m\geq 1$, $n\geq 1$.

4. Постройте магазинен автомат, който да разпознава чрез заключителни състояния допълнението на езика: а) $L_1=\{\omega\omega;\ \omega\in\{0, 1\}^*\}$, б) $L_2=\{\omega O(\omega);\ \omega\in\{a, b\}^*\}$

## 3.7. ЕКВИВАЛЕНТНОСТ НА МАГАЗИННИТЕ АВТОМАТИ И БЕЗКОНТЕКСТНИТЕ ГРАМАТИКИ

Недетерминираните магазинни автомати са точно тези разпознаватели, които разпознават безконтекстните езици: за всеки безконтекстен език може да се построи магазинен автомат, който да го разпознава, и обратно — езиците, които се определят от магазинните автомати, са винаги безконтекстни. Това съответствие е от съществено значение за построяването на различни синтактични анализатори както на езиците за програмиране, така и на естествените езици.

Да вземем произволна безконтекстна граматика $\Gamma=(V, W, S, P)$. Ще построим магазинен автомат **R**, който да разпознава чрез празен магазин езика $L(\Gamma)$, породен от граматиката $\Gamma$.

Да приемем отначало, че $L(\Gamma)$ не съдържа празната дума.

Магазинният автомат **R** определяме по следния начин:

$\mathbf{R}=(\{s\}, V, V\cup W, \delta, s, S, \varnothing)$, като функцията $\delta$ изглежда така:

— $\delta(s, \Lambda, A)$ съдържа $(s, \alpha)$, ако в $P$ има правило $A\to\alpha$;

— за всяко $a$ от $V$ $\delta(s, a, a)=\{(s, \Lambda)\}$.

С работата си магазинният автомат **R** моделира в магазина левите изводи на думите, пораждани от $\Gamma$, а както знаем, за всяка дума от безконтекстния език $L(\Gamma)$ има ляв извод в граматиката $\Gamma$. Автоматът **R** извършва следните действия върху дадена входна

дума ω, различна от празната Λ: с Λ-такт заменя началния магазинен символ $S$ (той с и начален символ на Г) с думата $\alpha$, ако в Г има правило $S \to \alpha$. Ако има няколко такива правила, всички те са възможни варианти за действие. След това автоматът **R** изтрива в магазина си частта от думата $\alpha$ до първия нетерминален символ, в случай че тя съвпада с начална част от входната дума ω. При това магазинният автомат се придвижва през тази начална част на думата ω. Ако тези части не съвпадат, магазинният автомат е неопределен и не разпознава думата ω. По-нататък с Λ-такт автоматът **R** заменя първия нетерминален символ $A$ от магазинната дума (той ще бъде най-отпред в магазина) с думата $\alpha$, ако в Г има правило $A \to \alpha$. Ако **R** не може да направи това, той е неопределен. След това отново изтрива началната терминална част от магазинната дума, при условие че тя съвпада със следваща част от думата ω и т. н. Ако с преминаването на думата ω автоматът **R** изпразни магазина си, **R** разпознава тази дума, а работата на **R** показва, че думата се поражда чрез ляв извод в граматиката Г.

Чрез индукция по дължината на извода лесно може да се докаже, че ако $A \overset{\Gamma}{=} \omega$, то

$$(s, \omega, A) \vdash \ldots \vdash (s, \Lambda, \Lambda),$$

а оттук следва, че $(s, \omega, S) \vdash \ldots \vdash (s, \Lambda, \Lambda)$, когато $S \overset{\Gamma}{=} \omega$.

Обратното твърдение за $\omega \neq \Lambda$, че ако $(s, \omega, A) \vdash \ldots \vdash (s, \Lambda, \Lambda)$, то $A \overset{\Gamma}{=} \omega$, се доказва чрез индукция по броя на тактовете. В такъв случай при $A = S$ получаваме $S \overset{\Gamma}{=} \omega$, когато $(s, \omega, S) \vdash \ldots \vdash (s, \Lambda, \Lambda)$.

И така, магазинният автомат **R** разпознава точно онези думи, които безконтекстната граматика Г поражда.

В случай че езикът $L(\Gamma)$ съдържа празната дума, построяваме магазинния автомат **R** за граматиката на езика $L(\Gamma) - \{\Lambda\}$, а след това към множеството $\delta(s, \Lambda, S)$ добавяме $(s, \Lambda)$. С това осигуряваме възможността автоматът **R** да разпознава и празната дума.

Да разгледаме например граматиката $\Gamma = (\{0, 1\}, \{S\}, S, \{S \to SS, S \to 0S1, S \to 1S0, S \to 01, S \to 10\})$. За нея получаваме следния магазинен автомат **R**, който разпознава думите от езика $L(\Gamma)$:

$\mathbf{R} = (\{s\}, \{0, 1\}, \{0, 1, S\}, \delta, s, S, \varnothing)$ с функция $\delta$:

$\delta(s, \Lambda, S) = \{(s, SS), (s, 0S1), ((s, 01), (s, 10)\}$,

$\delta(s, 0, 0) = \{(s, \Lambda)\}$, $\delta(s, 1, 1) = \{(s, \Lambda)\}$.

Върху входната дума 100110 автоматът $R$ действува по следния начин (в един от възможните варианти):



$(s, 100110, S) \vdash (s, 100110, 1S1) \vdash (s, 00110, S0) \vdash (s, 00110, 0S10) \vdash (s, 0110, S10) \vdash (s, 0110, 0110) \vdash (s, 110, 110) \vdash (s, 10, 10) \vdash (s, 0, 0) \vdash (s, \Lambda, \Lambda)$.

Както се вижда, в работата си върху магазинните думи автоматът **R** всъщност моделира извода $S \vdash 1S0 \vdash 10S10 \vdash 100110$ и го сравнява последователно с входната дума.

Нека сега е даден произволен магазинен автомат **M**, разпознаващ думите чрез празен магазин. Ще покажем, че може да се намери безконтекстна граматика Г, която да поражда точно думите, които **M** разпознава, т. е., за която $L(\Gamma) = N(M)$.

Да вземем произволен магазинен автомат

$$\mathbf{M} = (K, \Sigma, \Gamma, \delta, q_0, Z_0, \varnothing).$$

По магазинния автомат $M$ ще построим граматика $\Gamma = (\Sigma, W, S, P)$, в която нетерминалната азбука се състои от новия символ $S$ и от символи $S_{p, Z, q}$ за всяко $p$ и $q$ от $K$ и $Z$ от Г, а множеството $P$ се състои от следните правила:

— за всяко $p$ от $K$ поставяме в $P$ правило $S \to S_{q_0, Z_0, p}$;

— ако за $p$, $q$ от $K$, $a$ от $\Sigma \cup \{\Lambda\}$ и $Z_1, Z_2, \ldots, Z_k$, от Г множеството $\delta(q, a, Z)$ съдържа двойката $(p, Z_1 Z_2 \ldots Z_k)$, в $P$ поставяме за всяка редица $p, p_1, \ldots, p_k$, $k \geq 1$, $p_1, \ldots, p_k$ от $K$ правилата

$$S_{q, Z, p} \to a S_{p, Z_1, p_1} S_{p_1, Z_2, p_2} \ldots S_{p_{k-1} Z_k p_k};$$

— ако $\delta(q, a, Z)$ съдържа двойката $(p, \Lambda)$, поставяме правилото $S_{q, Z, p} \to a$.

Граматиката Г, както лесно може да се забележи, е безконтекстна. Тя действува, като чрез ляв извод извежда последователно думи, които се състоят от прочетената част на входната дума за магазинния автомат и от превод на магазинната дума за момента. Тази граматика ще изведе една терминална дума ω, само когато автоматът **M** я прочете цялата и я разпознае чрез празен магазин. Може да се докаже чрез индукция, че $S_{p, Z, q} \overset{\Gamma}{=} \omega$ тогава и само тогава, когато $(p, \omega, Z) \vdash \ldots \vdash (q, \Lambda, \Lambda)$.

Получихме следния основен резултат:

**Класът на безконтекстните езици съвпада с класа на езиците, разпознавани от недетерминирани магазинни автомати.**

## Упражнения

1. Постройте магазинен автомат, който да разпознава чрез празен магазин езика, породен от безконтекстната граматика

$\Gamma = (\{a, b\}, \{S, A, B\}, S, \{S \to AB, S \to A, S \to B, A \to aAb, A \to ab, B \to Bb, B \to b, S \to \Lambda\})$.

2. Намерете безконтекстна граматика, която да поражда езика, разпознаван чрез заключителни състояния от следния магазинен автомат

$\mathbf{M}=(\{s_0, s_1, s_2\}, \{0, 1\}, \{Z_0, X\}, \delta, s_0, Z_0, \{s_2\}$ с функция:

$\delta(s_0, 0, Z_0)=\{(s_1, XZ_0)\}$ $\quad \delta(s_1, \Lambda, X)=\{(s_2, X)\}$

$\delta(s_0, 0, X)=\{(s_1, XX)\}$ $\quad \delta(s_2, 1, X)=\{(s_2, \Lambda)\}$.

$\delta(s_1, 0, X)=\{(s_0, XX)\}$

3. Постройте магазинен автомат, който да разпознава езика, породен от граматиката

$\Gamma=(\{+, *, (, ), 0, 1, \ldots, 9\}, \{A, B, C, D\}, A, \{A\to B, B\to C, C\to D, A\to A+B, B\to B*C, C\to(A), D\to 0, \ldots, D\to 9\})$.

4. Постройте магазинен автомат, който да разпознава описанията на прости променливи в езика АЛГОЛ (вж. задача 2 от 3.1).

## 8. СИНТАКТИЧЕН АНАЛИЗ

Задаването на формалните езици чрез формални граматики позволява на всяка дума от пораждания език да се припише вътрешна структура, съответстваща на извода на тази дума в дадената граматика. По аналогия с граматиката на естествените езици тази структура се нарича граматична или синтактична. Важен проблем в теорията на формалните езици е намирането на синтактичните структури на думите, образуващи езика в съответствие с правилата на пораждащата го граматика.

Процесът, който установява дали някаква дума от символи на езика удовлетворява правилата на дадена граматика (т. е. дали може да се породи чрез правилата на граматиката) и намира съответната й синтактична структура, се нарича **синтактичен анализ**, или още — **граматичен разбор**.

Разбира се, когато говорим за естествените езици, под „дума" можем да разбираме изречение или някакви други основни единици на езика.

Синтактичният анализ е важна съставна част на автоматичния превод на естествените или изкуствените езици (например — езиците за програмиране), тъй като този превод изисква на определен етап установяването на съответствия между синтактични структури.

Синтактичният анализ лесно може да се илюстрира върху безконтекстните граматики, тъй като при тях синтактичната структура на думите представлява сравнително прост обект — дърво.

Нека $\Gamma$ е безконтекстна граматика и $\omega$ е произволна дума. Нашата цел е да конструираме такъв **синтактичен анализатор**, който да определя дали думата $\omega$ е от езика $L(\Gamma)$ и в случай че $\omega$ е от $L(\Gamma)$, да дава някои (по възможност всички) дървета на извод в граматиката $\Gamma$ с корона думата $\omega$. Синтактичният анализатор може да бъде или някакъв автомат, или програма за компютър.



Да приведем един пример за това как може да се търси дървото на извод на някаква дума в дадена безконтекстна граматика. Да вземем граматиката

$\Gamma=(\{a, b\}, \{S, A\}, S, \{S\to a, S\to bSA, A\to b, S\to bA, A\to aS\})$.

Всички правила на тази граматика имат вида $X\to zY_1\ldots Y_k$, където $X, Y_1, \ldots, Y_k$ са нетерминални символи, а $z$ е терминален символ.

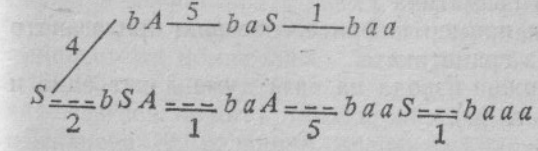
Доказано е, че за всяка безконтекстна граматика може да се намери еквивалентна на нея безконтекстна граматика, всичките правила на която имат горния вид. Това означава, че без загуба на общност можем да разглеждаме синтактичния анализ само на граматики от този вид.

Дървото на извод за произволна дума $\omega=d_1\ldots d_n$ се намира по следния начин. Синтактичният анализатор търси сред правилата на граматиката правило от вида $S\to d_1X_1\ldots X_l$ ($S$ е началният символ; $d_1$ е първият терминален символ от думата $\omega$; $X_1, \ldots, X_l$ са произволни нетерминални символи). Ако няма такова правило, думата $\omega$ е неизводима в $\Gamma$. В случай че в граматиката има такова правило, синтактичният анализатор избира първото правило от този вид и търси правило от вида $X_1\to d_2Y_1\ldots Y_m$. Процесът продължава по аналогичен начин нататък. Ако на някоя стъпка процесът не може да продължи, синтактичният анализатор се връща към последното място, на което е имал възможност за избор и търси да приложи друго правило вместо взетото преди това.

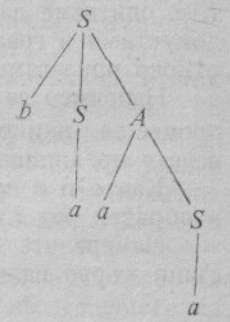
В случай че се получи извод на думата $\omega$, синтактичният анализатор запомня този извод и се връща отново към мястото, където за последен път е имал възможност да избира правило, което да прилага. Ясно е, че след краен брой стъпки ще се установи дали думата $\omega$ е изводима в граматиката $\Gamma$ и ще се намерят всички възможни дървета на извод на разглежданата дума.

Например за думата *baaa* получаваме следния процес на намиране на дървото й на извод, който графично е изобразен на фиг. 42:

Фиг. 42




Фиг. 43



Подчертаният път единствено дава извод на думата *baaa* в граматиката Г. За удобство върху всеки преход е отбелязан номерът на прилаганото правило.

Следователно, думата *baaa* има само едно дърво на извод в граматиката Г, което получаваме чрез графично изобразяване на прилаганите в процеса на извод правила (фиг. 43).

На този принцип са построени няколко системи за автоматичен синтактичен анализ на естествени езици. Такъв е например **предиктивният анализатор,** построен от Куно и Етингър през 1963 г. Анализаторът е построен на базата на безконтекстна граматика, описваща синтаксиса на фрагмент от английския език. Граматиката съдържа около 3500 правила, а анализът се извършва с помощта на речник, съдържащ около 25000 словоформи. За всяка зададена му фраза на английски език предиктивният анализатор намира всички възможни варианти на синтактичен анализ, които могат да бъдат направени в съответствие със зададените правила на граматиката. Предиктивният анализатор е използуван успешно за откриване на синтактични неоднозначности, например в американската конституция.

В разглеждания пример извличането на синтактичното дърво за дадена дума започва от началния символ, т. е. от корена и върви към короната на дървото. Такъв синтактичен анализ се нарича **анализ от горе на долу.**

Можем да извършваме синтактичния анализ и по обратния начин: като тръгваме от короната на дървото и се стараем да възстановим самото дърво, както това правим, когато анализираме синтактичната структура на изречение от естествен език.

Този метод се нарича **синтактичен анализ от долу на горе**

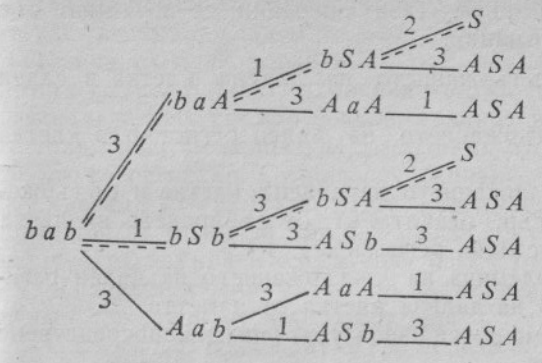
Най-общо синтактичният анализ от долу на горе изглежда така. Представяме си, че буквите на дадената дума са лист на някакво дърво. Търсим такива поддуми, които да представляват дясна част на граматично правило и ги заменяме с лявата част, т. е. опитваме да надстрояваме листата на дървото с различни допустими в граматиката клони. След това към получената дума отново прилагаме същия процес и т. н.

Например за думата *bab* получаваме показания на фиг. 44 процес за намиране на дървото й на извод в разглежданата вече в предишния пример граматика Г.

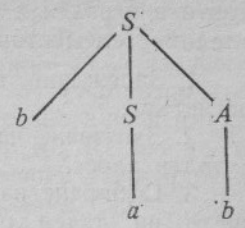
Както и в предишния пример цифрите означават прилаганото в обратен ред правило на граматиката.

Намерените два възможни извода на тази дума дават едно и също дърво на извод (фиг. 45).






Фиг. 44




Фиг. 45

## Упражнения

1. Дадена е граматиката

Г=({*a*, *b*, *c*, +, ∗ }, {*S*, *A*, *B*}, *S*, {*S*→*A*, *S*→*A*+*S*. *A*→*B* ∗ *A*, *A*→*B*, *B*→*a*, *B*→*b*, *B*→*c*}).

Намерете дърветата на извод на думата $a+b * c$ чрез: а) синтактичен анализ от горе на долу; б) синтактичен анализ от долу на горе.

2. Намерете алгоритъм за синтактичен анализ от долу на горе, който да намира само левите изводи на дадена дума в безконтекстна граматика.

3. Намерете алгоритъм за синтактичен анализ в произволна автоматна граматика.

### 3.9 ЕЗИЦИ ЗА ПРОГРАМИРАНЕ

Да вземем един прост компютър и да разгледаме възможностите за общуване с него, т. е. какво може той да върши, на какъв език трябва да бъде казано това, което трябва да свърши, къде да намерим и как да разберем получения отговор и т. н. Следвайки общата схема, нашият компютър трябва да има централен процесор, входно-изходни устройства (за четене и печат) и оперативна памет, в която да се съхраняват данните и програмите, да се записват междинните резултати и др.

Основните единици на паметта ще наричаме клетки и в тях могат да се записват думи от нули и единици с дължина 16 символа. Всяка клетка има свой адрес—някакво двоично число. Освен в оперативната памет, информация може да се съхранява и в две работни полета от паметта, намиращи се в централния процесор на компютъра — регистри, които ще наричаме с имената **R1** и **R0.**

Разглежданият компютър може чрез централния си процесор да изпълнява 12 различни операции, начинът на изпълнението на

които е вграден в компютъра. Тези операции се изразяват със следните елементарни команди:

1. Зареждане на съдържанието на дадена клетка в даден регистър.

2. Запазване на съдържанието на даден регистър в клетка с даден адрес.

3. Събиране на съдържанието на дадена клетка и съдържанието на даден регистър; резултатът от събирането на двете стойности се запазва в същия регистър.

4. Сравняване по големина на съдържанието на даден регистър със съдържанието на дадена клетка от паметта.

Следващите три команди се използуват винаги непосредствено след четвъртата команда:

5. Преход—ако при изпълнението на четвъртата команда е установено, че съдържанието на регистъра е по-голямо от съдържанието на клетката, управлението се предава на команда, записана в клетка с даден адрес.

6. Преход—ако е установено, че съдържанието на регистъра е по-малко от съдържанието на клетката, управлението се предава на команда, записана в клетка с даден адрес.

7. Преход—ако е установено, че съдържанието на регистъра е равно на съдържанието на клетката, управлението се предава на команда, записана в клетка с даден адрес.

Въведени са още и следните пет команди:

8. Преход—управлението се предава на команда, записана в клетка с даден адрес.

9. Вход—задават се началните адреси на група клетки, където да се запише входната информация.

10. Изход—задават се адресите на клетки, в които да се запише изходната информация.

11. Проверка за готовност на входното и изходното устройства.

12. Стоп—преустановяване на работата на компютъра.

Ще считаме, че резултатът от изпълнението на четвърта команда се запчсва в отделен регистър, а при изпълнението на пета, шеста, седма команди се проверява съдържанието на този регистър.

Както е известно, задачите, които компютрите могат да решават, имат алгоритмичен характер. Програмата, която компютърът трябва да изпълни, представлява алгоритъм, записан на „разбираем“ за него език. Без „преводач“ компютърът „разбира“ само собствения си машинен език.



Всяка **програма, записана на машинния език** на разглеждания от нас компютър, представлява списък от думи с дължина 16, съставени от нули и единици. Във всяка такава дума първите четири символа дават двоичния код на една от елементарните команди, петият символ е нула, единица или празно в зависимост от това, дали се използува работният регистър **R0**, **R1** или не се използува регистър. Останалите единадесет символа дават адреса на клетка от паметта. При тази организация на езика компютърът може да има най-много 16 елементарни команди, 2 работни регистъра и оперативна памет от $2048 = 2^{11}$ клетки.

Да предположим, че искаме да реализираме следния алгоритъм на разглеждания от нас компютър:

Стъпка 1: Въвеждат се три цели положителни числа във входното устройство и се преминава към стъпка 2.

Стъпка 2: Ако първото прочетено число е по-голямо от второто, на изходното устройство се предава второто число и се преминава към стъпка 5. В противен случай се преминава към стъпка 3.

Стъпка 3: Ако първото прочетено число е по-малко от третото, на изходното устройство се предава третото число и се преминава към стъпка 5. В противен случай се преминава към стъпка 4.

Стъпка 4: На изходното устройство се предава първото прочетено число и се преминава към стъпка 5.

Стъпка 5: Стоп.

Да приемем, че адресите на клетките в паметта са номерирани с числата от 0 до 2047 в двоичен запис; елементарните команди са номерирани последователно с числата 0001, 0010, ..., 1100; символът „—“ означава празно или запис, който е без значение, а входът представлява лента, състояща се от шест позиции, на които са записани три двуцифрени числа. Тогава горният алгоритъм можем да запишем на машинен език по следния начин:

**0.** 1001—11111 010000
**1.** 1011000000 000011
**2.** 1000—00000000001
**3.** 0001111111110 10000
**4.** 0100111111110 10001
**5.** 0110—000000001010
**6.** 0111—000000001010
**7.** 0001111111110 10001
**8.** 0010111111110 10010
**9.** 1000—000000001111
**10.** 0001 1 11111101000 0
**11.** 01001 1111101001 0
**12.** 0110—00000001 11 1
**13.** 0001 1 11111101000 0
**14.** 0010 1 11111101001 0
**15.** 1010— 1111101001 0
**16.** 10111 0000001001 0
**17.** 1000— 0000001000 0
**18.** 1100— ——————————

Както се вижда, машинният език е твърде далеч от езиците, на които човек е свикнал да общува. В програмите, написани на такъв език, трудно се проследява логиката им, основните структурни единици на езика не са ясно отделени, поради което не е лесно да се проверяват смисълът и правилността на употребата им. Освен това програмистът трябва да се грижи за разпределението на паметта, да помни адресите на командите, къде са записани междинните резултати и т. н.

Първата решителна стъпка за приближаването на средствата за диалог с компютъра към естествените езици е създаването на така наречените **асемблерни езици** и **асемблери.**

В **асемблерните езици** командите не се записват с двоичен код, а имат име, обикновено съответстващо на смисъла на командата. Например елементарните команди 1.—12. последователно можем да означаваме вече със следните имена: ЗР (зареждане), ЗП (запазване), С (събиране), СР (сравняване), ПРГ (преход при по-голям), ПРМ (преход при по-малък), ПРР (преход при равно), ПР (преход), В (вход), И (изход), Т (тест, проверка), К (край, стоп).

Друга една съществена разлика в сравнение с машинните езици се проявява в това, че програмистът не се грижи за разпределението на паметта така непосредствено—той вече не работи с конкретните адреси на клетки от паметта, а със зададени от него техни имена **(етикети)**. Освен това в асемблерния език може да има и команди, които предизвикват изпълнението на редица от елементарни команди.

Тези нови възможности на езика се осигуряват от програма преводач, която превежда на машинен език програмите, написани на асемблерен език. Тази програма може да извърши и някои допълнителни действия, за които получава съответни инструкции чрез специални команди, записани от програмиста в превежданата програма.

Програмите преводачи от асемблерен език на машинен език се наричат обикновено **асемблери**.

Разгледаният от нас алгоритъм можем да запишем на асемблерен език по следния начин:

| Етикет | Команда | Адрес |
|---|---|---|
| *J* | РЕЗ | 1 |
| *K* | РЕЗ | 1 |
| *L* | РЕЗ | 1 |
| НАЧАЛО | В | *J* |
| | ВХОД | НАЧАЛО |
| Е1 | Т, 0 | Е2 |
| | ПР | Е1 |
| Е2 | ЗР, R1 | *J* |
| | СР, R1 | *K* |
| | ПРМ | Е3 |
| | ПРР | Е3 |
| | ЗР, R1 | *K* |
| | ЗП, R1 | *L* |
| | ПР | Е4 |
| Е3 | ЗР, R1 | *J* |
| | СР, R1 | *L* |
| | ПРМ | Е4 |
| | ЗР, R1 | *J* |
| | ЗП, R1 | *L* |
| Е4 | И | *L* |
| Е5 | Т, 1 | Е6 |
| | ПР | Е5 |
| Е6 | К | |
| | КРАЙ | НАЧАЛО |

В тази програма на асемблерен език първите три реда представляват инструкция към асемблера да резервира (РЕЗ) три клетки от паметта за трите входни числа. В програмата те се цитират с имената *J*, *K*, *L*. Командата с име ВХОД означава за асемблера, че написаната програмна част може да бъде използувана от други програмни части от командата с етикет НАЧАЛО. Командата с име КРАЙ показва на асемблера, че редицата от команди за превеждане е свършила. Последният ред означава, че след успешен превод на дадената програма изпълнението на генерираната на машинен език програма трябва да започне от командата с етикет НАЧАЛО. Останалите команди са пряк превод на командите от машинния език.

В разгледания от нас пример могат да се видят отбелязаните по-горе предимства на асемблерните езици. Все пак асемблерните езици стоят твърде близко до машинните езици, поради което са тясно свързани с използувания компютър.

Машинните и асемблерните езици образуват съответно **езиците за програмиране от нулево и първо ниво.** Първият, независещ от компютрите език за програмиране, е създаден през 1956 година. Това е езикът ФОРТРАН. Малко по-късно през 1958—1960 г. г. се разработва още един език за програмиране—езикът АЛГОЛ. Тези два езика и до днес се използуват по целия свят. Езиците за програмиране подобни на ФОРТРАН и АЛГОЛ обра-

зуват **второто ниво на езиците за програмиране**. Следващото **трето ниво** се определя от т. нар. **проблемно-ориентирани езици**. Често пъти езиците от второ и трето ниво се наричат **езици от високо ниво** или само—**езици за програмиране**.

Какви са основните предимства на езиците за програмиране от високо ниво?

Преди всичко те са значително по-близко до естествените езици, отколкото машинните и асемблерните езици. Поради това езиците за програмиране от високо ниво се усвояват и използуват сравнително лесно. Основните им съставящи единици не са формулирани в термините на хардуера на компютъра, а се определят преди всичко от класа задачи, за чиито решаване са предназначени тези езици. Поради това програмите, написани на език за програмиране от високо ниво, лесно се поддават на разбиране, модифициране и коригиране, не са зависими от конкретен компютър и могат да бъдат изпълнявани на всеки компютър, за който съществува подходяща програма преводач, наречена **транслатор**.

Наличието на езици за програмиране на високо ниво поставя въпроса за тяхното използуване, т. е. за възможността програмите, написани на такъв език, да бъдат превеждани на машинния език на компютъра.

Въпреки че процесът на превод, или още—**транслация**, от език за програмиране от високо ниво на машинен език може да се извършва и непосредствено от човека, неговото посредничество е нежелателно по редица причини. За целта се създават програми преводачи, наречени **транслатори** или **компилатори**, които превеждат всяка програма, написана на даден език за програмиране от високо ниво, в програма, написана на асемблерен или машинен език.

Наличието на посредник—транслатор, между езиците за програмиране от високо ниво и машинния език на компютъра поставя изключително важния въпрос за точното, формално определяне на езиците за програмиране. Тъй като езиците за програмиране са предназначени за описване на класове от алгоритми под формата на програми, то езикът за програмиране може да се определи чрез съвкупността от програми. Това превръща всеки език за програмиране във формален език, в който от определени символи се строят различните низове—програми, влизащи в езика. Както знаем, един от основните начини за задаване на формалните езици са формалните граматики. Този подход е избран при задаването на преобладаващата част от езици за програмиране от високо ниво: определят се множеството от допустими символи (лексиката на езика)



и граматичните правила (синтаксисът на езика), които пораждат правилно построените програми.

За описване на **синтаксиса на езиците за програмиране** от високо ниво се използуват различни формални граматики. Например безконтекстните граматики представляват просто и удобно средство за задаване на голяма част от синтаксиса на такива езици за програмиране, като АЛГОЛ, а съответните им разпознаватели — магазинните автомати се използуват широко при синтактично ориентираната транслация на тези езици.

При описанието на синтаксиса на подобни езици за програмиране е прието да се използуват четири металингвистични символа :: =, | ,⟨,⟩, чрез които правилата на синтаксиса се записват в така наречената **нормална форма на Бекус**. В тази форма например е определен синтаксисът на езика АЛГОЛ.

Ние вече използувахме тези металингвистични символи и както знаем :: = означава „по определение е“ и отделя определяемото понятие от неговото определение; | означава „или“ и отделя алтернативните определения, а ъгловите скоби заграждат синтактичните категории (нетерминалните символи). Начален символ на граматиката е синтактичната категория ⟨програма⟩, тъй като желаем низовете, които граматиката поражда, да представляват програми. По-долу е описан синтаксисът на един примерен език за програмиране от високо ниво:

⟨програма⟩ :: = ⟨блок⟩
⟨блок⟩ :: = ***BEGIN*** ⟨тяло на блок⟩
⟨тяло на блок⟩ :: = ⟨декларация⟩ ⟨тяло на блок⟩ | ⟨списък от оператори⟩ ***END***
⟨декларация⟩ :: = ***LABEL*** ⟨списък от идентификатори⟩ | ***INTEGER*** ⟨списък от идентификатори⟩
⟨списък от идентификатори⟩ :: = ⟨идентификатор⟩ | ⟨остатък от списък от идентификатори⟩
⟨остатък от списък от идентификатори⟩ :: =; | ,⟨списък от идентификатори⟩
⟨списък от оператори⟩ :: = ⟨етикетна част по избор⟩ ⟨оператор по избор⟩ ⟨остатък от списък⟩
⟨етикетна част по избор⟩ :: = ⟨празно⟩ | ⟨идентификатор⟩:
⟨оператор по избор⟩ :: = ⟨празно⟩ | ⟨оператор⟩
⟨остатък от списък⟩ :: = ⟨празно⟩ | ;⟨списък от оператори⟩
⟨оператор⟩ :: = ⟨оператор за присвояване⟩ | ⟨оператор за преход⟩ | ⟨условен оператор⟩ | ⟨оператор за изход⟩ | ⟨блок⟩
⟨оператор за присвояване⟩ :: = ⟨израз⟩ ⇒ ⟨идентификатор⟩ ⟨остатък от списък за присвояване⟩
⟨остатък от списък за присвояване⟩ :: = ⟨празно⟩ | ⇒⟨идентификатор⟩ ⟨остатък от списък за присвояване⟩

⟨оператор за преход⟩ :: = ***GOTO*** ⟨идентификатор⟩
⟨условен оператор⟩ :: = ***IF***⟨израз⟩ ***THEN*** ⟨списък от оператори⟩ ⟨или по избор⟩ ***FI***
⟨или по избор⟩ :: = ⟨празно⟩ | ***ELSE*** ⟨списък от оператори⟩
⟨оператор за изход⟩ :: = ***OUTPUT*** (⟨описание на изход⟩)
⟨описание на изход⟩ : =⟨израз⟩ ⟨остатък от описание на изход⟩
⟨остатък от описание на изход⟩ :: = ⟨празно⟩ | , ⟨израз⟩ ⟨остатък от описание на изход⟩
⟨израз⟩ :: = ⟨изр. 1⟩ ⟨остатък от израз⟩
⟨остатък от израз⟩ :: = ⟨празно⟩ | ⟨операция за сравняване⟩ ⟨изр. 1⟩
⟨изр. 1⟩ :: = ⟨изр. 2⟩ ⟨остатък от изр. 1⟩
⟨остатък от изр. 1⟩ :: =⟨празно⟩ | ⟨операция за събиране⟩ ⟨изр. 2⟩ ⟨остатък от изр. 1⟩
⟨изр. 2⟩ :: = ⟨изр. 3⟩ ⟨оста-тък от изр. 2⟩
⟨остатък от изр. 2⟩ :: =⟨празно⟩ | ⟨операция за умножение⟩ ⟨изр. 3⟩ ⟨остатък от изр. 2⟩
⟨изр. 3⟩ :: = ⟨положителен изр. 3⟩ | —⟨положителен изр. 3⟩
⟨положителен изр. 3⟩ :: = ***INPUT*** | ⟨идентификатор⟩ | ⟨константа⟩ | (⟨израз⟩)
⟨операция за сравняване⟩ :: = < | > | =
⟨операция за събиране⟩ :: =+ | —
⟨операция за умножение⟩ :: = × | /
⟨празно⟩ :: =
⟨идентификатор⟩ :: = ⟨буква⟩ | ⟨идентификатор⟩ ⟨буква⟩ | ⟨идентификатор⟩ ⟨цифра⟩
⟨константа⟩ :: = ⟨цифра⟩ | ⟨константа⟩ ⟨цифра⟩
⟨буква⟩ :: = А | Б | В | Г | .... | Я
⟨цифра⟩ :: = 0 | 1 | ..... | 9

Следната програма, написана според синтаксиса на определения тук език за програмиране, дава описание на разгледания вече алгоритъм:

```
BEGIN INTEGER J, K, L;
      INPUT ⇒ J ⇒ K ⇒ L;
      IF J>K THEN K⇒L ELSE IF J>L+1 THEN
      J⇒L FI FI;
      OUTPUT (L);
END
```

**Упражнения**

1. Като използувате определения в тази част език за програмиране от високо ниво, напишете програма, която по зададени стойности на $n$ цели числа да пресмята сумата от абсолютните им стойности.

2. Намерете дървото на извод за програмата, написана на определения в тази част език за програмиране от високо ниво.

3. Разгледайте как елементарните команди на определения в тази част компютър се изразяват в езика за програмиране от високо ниво.


*„La machine arithmétique fait des effets, qui approchent plus de la pensée que tout ce que font des animaux“*[1]

Bl. Pascal

# Глава 4

# ФОРМАЛНИ ЕЗИЦИ ОТ ОБЩ ВИД И МАШИНИ НА ТЮРИНГ

## 4.1. ОПРЕДЕЛЕНИЕ И СВОЙСТВА НА МАШИНИТЕ НА ТЮРИНГ

През 1936 год. английският математик Алан Тюринг публикува работата си „On computable numbers with an application to the entscheidungsproblem“, в която представя един вид автомати (или още — машини) — математически модели на всеки изчислителен процес. Там той определя и машина, която може да имитира работата на всяка машина от определения вид, а следователно, която може да прави всичко, което могат да вършат съвременните компютри и да изчислява всяка ефективно изчислима функция. Тези автомати днес са известни като машини на Тюринг, а автоматът, който моделира работата на всяка машина на Тюринг е наречен универсална машина на Тюринг.

Тук ще представим накратко как изглеждат машините на Тюринг и какво могат те да правят. Ще покажем, че езиците, които машините на Тюринг разпознават, са точно езиците от типа 0 (или от общ вид) от йерархията на Чомски.

---

[1] Аритметичната машина прави неща по-близки до мисълта, отколкото всичко, което правят животните.

*Блез Паскал*

Дотук разгледахме различни разрешими алгоритмични проблеми. Като използуваме универсалната машина на Тюринг, ще получим, че съществуват и алгоритмично неразрешими проблеми, засягащи формалните езици.

Накрая ще покажем, че има формални езици от тип 0, които не могат да бъдат контекстни.

В съответствие с общото изложение машините на Тюринг ще определим като разпознаватели.

**Машините на Тюринг** се състоят от управляващото устройство с крайна памет, което може да се намира в едно от краен брой вътрешни състояния, и от лента, разделена на клетки, която може да бъде неограничено продължавана в двете направления Всяка клетка съдържа един символ от азбуката на лентата. Част от лентовите символи образуват входната азбука, с която се записват върху лентата входните думи. Един от лентовите символи (но не от входната азбука) е фиксиран, и когато той е записан в дадена клетка, казваме, че тази клетка е празна. Клетките, които се добавят с продължаването на лентата, са винаги празни.

Управляващото устройство е свързано с устройство, което може да прочита символа в дадена клетка и да написва евентуално нов символ на неговото място. Това устройство може да се придвижва с една клетка наляво или надясно или да остава на мястото си.

От вътрешните състояния на управляващото устройство определен брой са фиксирани и образуват множество на заключителните състояния. Едно от вътрешните състояния е също фиксирано и се нарича начално състояние.

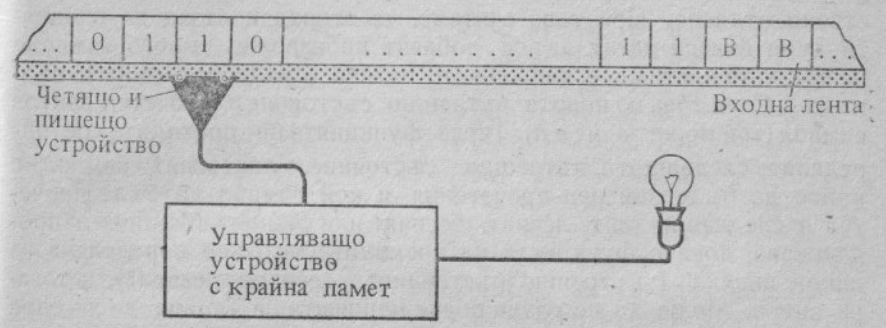
Машината на Тюринг също като останалите автомати работи в дискретни моменти от време — тактове. Тя започва работа в начално състояние и с четящо устройство върху първия символ на зададената входна дума. Във всеки следващ такт управляващото устройство определя в зависимост от вътрешното си състояние и прочетения лентов символ в какво ново състояние да премине, кой символ да запише на мястото на прочетния и дали четящото устройство да остане на мястото си, или да се придвижи с една клетка наляво или надясно. Машината спира работа, ако в управляващото устройство няма инструкция как тя да продължи. Резултат от работата е последното вътрешно състояние. Ще обърнем внимание на това, че машината на Тюринг може и никога да не спре. Машината на Тюринг разпознава дадената входна дума, ако състоянието, в което е спряла, е заключително.

Съществената разлика от магазинните автомати е в това, че безкрайната външна памет на машините на Тюринг — лентата, е



организирана по нов начин — всяка клетка може да бъде прочетена (а не само най-лявата, както е при магазинния автомат) и в нея да бъде записан някакъв друг символ. Това определя новите възможности на машините на Тюринг.

Схематично една машина на Тюринг може да се представи по следния начин (фиг. 46):




Фиг. 46

Формално **машина на Тюринг** наричаме наредената седморка $\mathbf{M}=(K, \Sigma, \Gamma, \delta, s_0, B, F)$, в която:

— $K$ е **азбука на вътрешните състояния**;

— $\Sigma$ е **азбука на входните символи**;

— $\Gamma$ е **азбука на лентовите символи**, което включва в себе си азбуката на входните символи и поне още един лентов символ $B$ (**празната клетка**);

— $\delta$ е **функция на преходите**, която на наредени двойки от вида („вътрешно състояние“, „лентов символ“) съпоставя наредени тройки от вида („вътрешно състояние“, „лентов символ“, „един от символите $L$, $R$, $N$“,) (символите $L$, $R$, $N$ не принадлежат на никоя от азбуките на машината);

— $s_0$ е от $K$ и се нарича **начално състояние**;

— $B$ е от $\Gamma$ и се нарича **празна клетка**. Символът $B$ не принадлежи на $\Sigma$ — азбуката на входните символи;

— $F$ е подмножество на $K$ и се нарича **множество на заключителните състояния**.

Машината на Тюринг ще бъде **недетерминирана**, ако функцията $\delta$ на наредени двойки („вътрешно състояние“, „лентов символ“) съпоставя крайни множества от наредени тройки от вида (вътрешно състояние, „лентов символ“, „един от символите $L$, $R$, $N$“).

**Работата на машината на Тюринг** върху дадена входна дума $a_{i_1}, a_{i_2}, \ldots a_{i_k}$ от $\Sigma^*$ изглежда така: По началното вътрешно състояние и първия входен символ $a_{i_1}$ чрез функцията на преходите се определят следващото вътрешно състояние, лентовият символ, който да замени $a_{i_1}$, и кой символ следва да бъде прочетен: при $N$ това е символът, който току-що е написан, при $L$ — символът непосредствено отляво, при $R$ — символът непосредствено отдясно. При това считаме, че отпред и отзад на входната дума винаги могат да се добавят произволно много символи на празната клетка, т. е. движението наляво и надясно е винаги възможно. След това по новото вътрешно състояние и прочетен лентов символ (той може да не е $a_{i_2}$) чрез функцията на преходите се определят следващото вътрешно състояние, лентовият символ, с който да бъде заменен прочетения, и кой символ да бъде прочетен в следващия такт: левият, десният или същият. Машината продължава, докато функцията на преходите не бъде определена за някоя двойка („вътрешно състояние", „лентов символ"), и тогава спира. Може да се случи обаче машината на Тюринг да не спре никога. **Резултат от работата** на машината е последното вътрешно състояние — дали то принадлежи на $F$ или не. **Входната дума се разпознава** от машината на Тюринг **M**, ако това състояние принадлежи на $F$, т. е. когато е заключително.

Множеството от всички думи, които една машина на Тюринг разпознава, образува езика $T(\mathbf{M})$, който тя разпознава.

Ако машината на Тюринг е недетерминирана, работата се определя както при останалите недетерминирани автомати: недетерминиран краен автомат и недетерминиран магазинен автомат. При всеки такт функцията на преходите определя краен брой възможности за избор на следващия преход: ново вътрешно състояние, символ за замяна на прочетения вече и вид движение. Недетерминираната машина на Тюринг разпознава дадена входна дума, ако в поне една от възможните поредици от преходи тя спре в заключително състояние. Ще добавим, че подобно на крайните автомати, **недетерминираните и детерминираните машини на Тюринг разпознават един и същ клас от езици**. Това се доказва, като за всяка недетерминирана машина на Тюринг се строи детерминирана машина на Тюринг, която систематично и детерминирано извършва всички възможни преходи на недетерминираната машина, докато тя спре. Без да привеждаме доказателството на този факт, ще считаме, че двете версии — детерминираната и недетерминираната, са еквивалентни по отношение на разпознаваните от **тях езици**.



**Диаграмата на преходите за машини на Тюринг** се определя по стандартния начин: два върха, съответстващи на състоянията $p$ и $q$ свързваме с насочено ребро от $p$ към $q$, означено с $a/b, D$, ако $\delta(p, a) = (q, b, D)$. Тук $a$ и $b$ са от Г, а $D$ е един от символите $L, R, N$.

Да определим следната машина на Тюринг **M**:

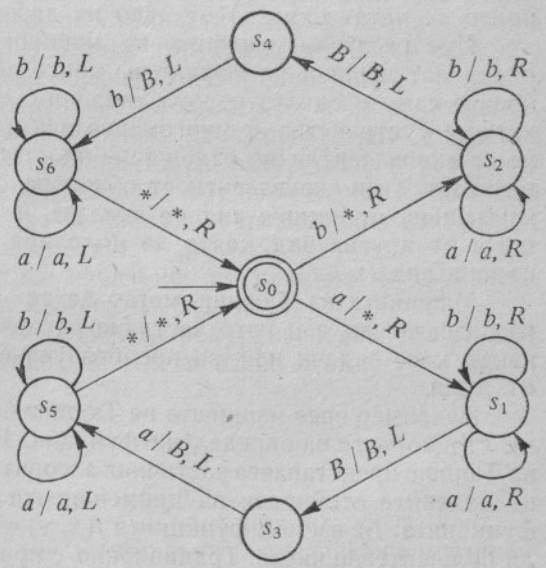
$\mathbf{M} = (\{s_0, s_1, \ldots, s_6\}, \{a, b\}, \{a, b, *, B\}, \delta, s_0, B, \{s_0\})$ с функция на преходите $\delta$:

$\delta(s_0, a) = (s_1, *, R)$,
$\delta(s_0, b) = (s_2, *, R)$,
$\delta(s_1, a) = (s_1, a, R)$,
$\delta(s_1, b) = (s_1, b, R)$,
$\delta(s_1, B) = (s_3, B, L)$,
$\delta(s_2, a) = (s_2, a, R)$,
$\delta(s_2, b) = (s_2, b, R)$,
$\delta(s_2, B) = (s_4, B, L)$,
$\delta(s_3, a) = (s_5, B, L)$,
$\delta(s_4, b) = (s_6, B, L)$,
$\delta(s_5, a) = (s_5, a, L)$,
$\delta(s_5, b) = (s_5, b, L)$,
$\delta(s_5, *) = (s_0, *, R)$,
$\delta(s_6, a) = (s_6, a, L)$,
$\delta(s_6, b) = (s_6, b, L)$,
$\delta(s_6, *) = (s_0, *, R)$.

Диаграмата на преходите е дадена на фиг. 47.

За да записваме по-лесно преходите, които една машина на Тюринг прави, когато работи върху зададена входна дума, ще въведем, както и преди, понятието **конфигурация**. Конфигурацията представлява моментна снимка на машината: тя показва какво е записано на лентата, в какво състояние е управляващото устройство и къде се намира четящото устройство. Естествено записаното на лентата ще представяме като дума от лентови символи, при което символи за празни клетки добавяме, само когато това е необходимо, за да не излезем извън записаната дума. Мястото на четящото устройство ще означаваме, като пред символа, който то чете,

Фиг. 47



поставяме символа на вътрешното състояние. Ако например на лентата е записана думата *aabacaa*, управляващото устройство е в състояние $p$, а четящото устройство се намира върху буквата $c$, това ще записваме така: *aabapcaa*. Тактовете на машината могат да се представят като преход от една конфигурация към друга и този преход ще означаваме със знака $\vdash$.

Работата на зададената машина на Тюринг **M** върху входните думи *aba*, *abb*, *baab* можем да представим по следния начин:
за думата *aba* : $s_0aba \vdash_* s_1ba \vdash_* bs_1a \vdash_* bas_1B \vdash_* bs_3a \vdash_* s_5bB \vdash s_5{*}\ bB \vdash_* s_0bB \vdash_* {*}\ s_2B \vdash_* s_4{*}$ ;

за думата *abb* : $s_0abb \vdash_* s_1bb \vdash_* bs_1b \vdash_* bbs_1B \vdash_* bs_3b$

за думата *baab* : $s_0baab \vdash_* s_2aab \vdash_* as_2ab \vdash_* aas_2b \vdash_* aabs_2B \vdash_* aas_4bB \vdash_* as_6aB \vdash_* s_6aaB \vdash_* s_0aaB \vdash_* s_0aaB \vdash_* {*}\ s_1aB \vdash_* {*}\ as_1B \vdash_* {*}\ s_3aB \vdash_* s_5{*}\ B \vdash_* {*}\ s_0B$.

Виждаме, че машината на Тюринг, която определихме, разпознава думата *baab*, понеже $s_0$ е заключително състояние, и не разпознава думите *aba* и *abb*, тъй като $s_4$ и $s_3$ не са заключителни състояния.

Не е трудно да се покаже, изхождайки от начина на действие на **M**, демонстриран върху думите *aba*, *abb* и *baab*, че машината на Тюринг **M** разпознава точно думите с четна дължина, които се четат еднакво от ляво на дясно и от дясно на ляво.

Има различни варианти на машини на Тюринг. Могат да се определят машини на Тюринг, на които лентата е потенциално безкрайна само в едната посока, машини с няколко ленти, с няколко четящи устройства, с многомерна лента и т. н. Всички те обаче са еквивалентни по отношение на класа от езиците, които разпознават. Тази еквивалентност обикновено се доказва, като за всяка машина от единия вид се показва, че може да се построи машина от другия вид, която да моделира работата на машината от първия вид.

Машините на Тюринг могат да се разглеждат не само като разпознаватели, а и като частични алгоритми за решаване на някакъв клас задачи или за преобразуване на някакво множество от думи.

Например чрез машината на Тюринг бихме могли да изчисляваме стойностите на определени функции. В такъв случай машината на Тюринг представлява частичния алгоритъм или компютъра, който по дадените стойности на променливите определя стойността на функцията. Да вземем функцията $f(x, y) = x + y$ и нека $m$ и $n$ са цели положитеени числа. Традиционно е прието входната дума за машината да представлява унарния запис на числата $m$ и $n$, отделени с някакъв друг символ, например 0: $\underbrace{11\ldots10}_{m}\ \underbrace{11\ldots1}_{n}$



Резултатът от работата на съответната за $f(x, y)$ машина на Тюринг ще представлява записаната върху лентата дума от единици, когато машината спре.

Следната машина на Тюринг **M** изчислява стойностите на функцията $f(x, y)$ за цели положителни числа:
$\mathbf{M} = (\{s_0, s_1, s_2\}, \{0,1\}, \{0,1, B\}, \delta, s_0, B, \{s_2\})$
с функция на преходите $\delta$:

$\delta(s_0, 1) = (s_0, 1, R)$ $\quad$ $\delta(s_1, B) = (s_2, B, L)$
$\delta(s_0, 0) = (s_1, 1, R)$ $\quad$ $\delta(s_2, 1) = (s_2, B, N)$
$\delta(s_1, 1) = (s_1, 1, R)$

Машината **M** действува по следния прост начин: движи се надясно през първия блок от единици, докато открие нулата, превръща я в единица и продължава надясно, докато открие края на следващия блок от единици. След това изтрива последната единица и спира. Ясно е, че на лентата ще бъде написан блок от $m+n$ единици.

Функциите, стойностите на които могат да се изчисляват с машини на Тюринг, се наричат **частично рекурсивни.** Ако машината на Тюринг, изчисляваща стойностите на функцията, спира след краен брой стъпки за всички допустими стойности на аргументите, тогава функцията се нарича **общорекурсивна.**

Това определение може да се разпространи и върху езиците, разпознавани от машини на Тюринг. В съответствие с нашите предварителни бележки върху понятията рекурсивен и рекурсивно номерируем език, изхождайки от предположението, че всеки частичен алгоритъм може да се представи чрез машина на Тюринг, получаваме следните определения.

Един формален език ще наричаме **рекурсивно номерируем,** ако той се разпознава от машина на Тюринг.

Един формален език ще наричаме **рекурсивен,** ако той се разпознава от машина на Тюринг, която за всяка входна дума спира след краен брой стъпки.

## Упражнения

1. Постройте машина на Тюринг, която да разпознава езика: а) $L = \{\omega\omega;\ \omega \in \{0, 1\}^*\}$; б) $L = \{a^n b^n c^n\ ;\ n \geq 1\}$.
2. Постройте машина на Тюринг, която да пресмята стойностите на функцията: а) $f(x) = 2x$; б) $f(x, y) = xy$; в) $f(x, y) = |x - y|$ за цели положителни $x$ и $y$.
3. Постройте машина на Тюринг, която да удвоява дадена входна дума.

## 4.2. ЕКВИВАЛЕНТНОСТ НА МАШИНИТЕ НА ТЮРИНГ И ПОРАЖДАЩИТЕ ГРАМАТИКИ ОТ ТИП 0

Сега ще покажем, че **за всяка граматика от тип 0 може да се построи машина на Тюринг, която да разпознава езика, породен от тази граматика.** И обратно — **за всяка машина на Тюринг може да се намери граматика от тип 0, която да поражда езика, разпознаван от тази машина.**

Ще изложим нашите аргументи доста схематично и без да навлизаме в детайли, тъй като точното доказателство изисква построяването на редица спомагателни машини на Тюринг. Можем да си представим например машина на Тюринг, която придвижва даден блок от лентови символи с определен брой клетки наляво или надясно, която заменя една лентова дума с друга и т. н. При добро желание акуратният читател би могъл точно да опише такива машини.

И така, нека $\Gamma=(V, W, S, P)$ е произволна граматика от тип 0. Ще покажем как може да се построи недетерминирана машина на Тюринг, която да разпознава всички думи, които граматиката Г поражда, и само тях.

Да вземем произволно правило от граматиката Г:

$$X_1X_2 \ldots X_k \rightarrow Y_1Y_2 \ldots Y_l.$$

Може да се построи машина на Тюринг, която върху дадена входна дума $\omega$ действува така: ако първият символ на $\omega$ е $X_1$, замества $X_1$ с $Y_1$. В случай че първият символ на $\omega$ не е $X_1$, машината е неопределена. След като е заменила $X_1$ с $Y_1$, машината заменя втория символ на $\omega$, ако той е $X_2$, със символа $Y_2$. Ако обаче вторият символ на $\omega$ не е $X_2$, машината е неопределена и т. н.

Когато $k=l$, машината ще спре в заключително състояние върху първия символ на $\omega$, след като замени всички символи $X$ със съответните символи $Y$.

Когато $k<l$, машината, след като е заменила $X_k$ с $Y_k$, придвижва останалите символи от $\omega$ с $l-k$ клетки надясно, на опразненото място поставя символите $Y_{k+1}, \ldots, Y_l$, придвижва се върху първия символ на $\omega$ и там спира в заключително състояние.

Когато $k>l$, машината изтрива незаменените символи $X_{l+1}, \ldots, X_k$, придвижва с $k-l$ клетки наляво останалата част от думата $\omega$, придвижва се върху първия символ на $\omega$ и там спира в заключително състояние.

Да построим по една такава машина на Тюринг $\mathbf{M}_i$ за всяко от правилата на граматиката Г. Без ограничения на общността можем да считаме, че машините $\mathbf{M}_i$ имат непресичащи се азбуки



на вътрешните състояния и завършват работа в заключително състояние, което не се използува никъде другаде в работата им.

Сега можем да опишем недетерминирана машина на Тюринг **M**, която ще разпознава точно думите, които граматиката Г поражда.

Нека на лентата на **M** е записана входната дума $\omega \varepsilon V^*$. Машината маркира двете страни на думата $\omega$ със специален лентов символ $*$, напечатва отдясно началния символ $S$ на граматиката Г и след това отново маркира отдясно $S$ и се връща върху $S$ в някакво състояние $q$. Получава се конфигурацията $* \omega * qS *$.

Машината **M** е недетерминирана и множеството $\delta(q,S)$ съдържа тройки $(s_0^i, S, N)$ за всяко начално състояние $s_0^i$ на машините $\mathbf{M}_i$. Това означава, че **M** има възможност да включи по избор всяка една от машините $M_i$, като с това замени символа $S$ с дясната част $Y_1 \ldots Y_l$ на правило $S \rightarrow Y_1 \ldots Y_l$ от граматиката Г. След това машината **M** осигурява възможност управляващото й устройство да премине в ново състояние $p$.

Всяко множество $\delta(p, X)$ за произволен символ $X$ от $V \cup W$ съдържа тройките $(s_0^i, X, N)$ за всяко начално състояние $s_0^i$ на машините $\mathbf{M}_i$ и освен това съдържа и тройките $(p, X, L)$ и $(p, X, R)$. Това дава възможност на машината **M** да направи избор дали да започне да заменя лявата част на правило с дясната му част, или да премине към следващия или предишния лентов символ.

Накрая $\delta(p, *)$ съдържа тройка $(r, *, N)$, където $r$ е началното състояние на машина, която сравнява думата между първия и втория символ $*$ с думата между втория и третия символ $*$.

Машината **M** спира в заключително състояние, само ако двете думи съвпадат.

Така описана недетерминираната машина на Тюринг **M** ще спира в заключително състояние в някоя от възможните поредици от преходи тогава и само тогава, когато в граматиката Г има извод $S \models \omega$.

Сега ще покажем как по произволна машина на Тюринг **M** може да се построи граматика Г от тип 0, която да поражда езика $T(\mathbf{M})$ на думите, разпознавани от **M**.

Ще преобразуваме последователно дадената машина **M**, без да изменяме езика, който тя разпознава.

Първо, може да се построи нова машина на Тюринг $\mathbf{M}_1$, еквивалентна на **M**, която да не печата празни клетки и да не спира върху празни клетки. Това може например да се постигне по последния начин: добавяме нов лентов символ $*$ и по две нови

състояния $p_r$ и $q_r$ за всяко вътрешно състояние $r$ на **M**. Във функцията на преходите заменяме навсякъде символа за празна клетка $B$ с новия символ $*$ и добавяме следните нови преходи за всяко вътрешно състояние $r$ на **M** и за всеки лентов символ $X$ на **M** (с изключение на $B$):

$$\delta(r, B)=\{(p_r, *, R), (q_r, *, L)\},\ \delta(p_r, X)=(r, X, N),$$
$$\delta(q_r, X)=(r, X, N).$$

Тогава $\mathbf{M}_1$ ще работи точно както **M**, като при необходимост ще си добавя от двете страни клетки със символ $*$, играещи ролята на празните клетки при **M**.

Второ, може да се построи машина $\mathbf{M}_2$, еквивалентна на $\mathbf{M}_1$, която да работи като $\mathbf{M}_1$, но когато $\mathbf{M}_1$ завърши работа в заключително състояние, машината $\mathbf{M}_2$ да продължи, като изтрие всички непразни клетки от лентата (а те са в компактен блок), да запише нов лентов символ $S'$ и да спре в ново заключително състояние $p_f$ върху $S'$.

Трето, не е трудно да се построи машина на Тюринг $\mathbf{M}_3$, еквивалентна на $\mathbf{M}_2$, в която началното състояние $p_0$ да се използува само веднъж. Това например може да се получи, като се добавят ново начално състояние $p_0$ и преходите $\delta(p_0, a)=(q_0, a, N)$ за всеки входен символ $a$. Тук $q_0$ е началното състояние на $\mathbf{M}_2$.

По машината на Тюринг $\mathbf{M}_3$ ще построим следната граматика Г от тип 0. Терминални символи на Г ще бъдат входните символи на $\mathbf{M}_3$. Ще забележим, че **M** и $\mathbf{M}_3$ имат едни и същи входни символи. Нетерминални символи са един нов символ $S$, всички лентови символи на $\mathbf{M}_3$, които не са входни, както и всички вътрешни състояния на $\mathbf{M}_3$. Начален символ на граматиката ще бъде $S$, а правилата са следните:

1) $S \to p_f S$;

2) $p_0 a \to a$ за всяка терминална буква $a$;

3) за всеки преход $\delta(q, X)=(p, Y, R)$ се записва правило $Yp \to qX$;

4) за всеки преход $\delta(q, X)=(p, Y, N)$ се записва правило $pY \to qX$;

5) за всеки преход $\delta(q, X)=(p, Y, L)$ се записват правилата $pZY \to ZqX$ за всеки лентов символ $Z$ на $\mathbf{M}_3$.

В недетерминирания случай се записват правила за всеки член на $\delta(q, X)$.

Да вземем произволна дума $\omega$, която машината на Тюринг $\mathbf{M}_3$ разпознава. Това означава, че съществува следната редица от конфигурации:

$$p_0\omega \vdash \ldots \vdash X_1X_2 \ldots qX_i \ldots X_k \vdash \ldots \vdash p_f S'.$$



Тъй като правилата на граматиката Г моделират различните преходи, но в обратен ред, можем да запишем следния извод в Г:

$$S \underset{\Gamma}{|} p_f S' \underset{\Gamma}{|} \ldots \underset{\Gamma}{|} X_1X_2 \ldots q\ X_i \ldots X_k \underset{\Gamma}{|} \ldots \underset{\Gamma}{|} p_0\omega \underset{\Gamma}{|} \omega.$$

Обратно, всеки извод $S \underset{\Gamma}{|}= \omega$ в граматиката Г определя в обратен ред редица от конфигурации, започваща с началната конфигурация на думата $\omega$ и завършваща със заключителна конфигурация $p_f S'$.

Резюмирайки полученото дотук, можем да изкажем следното твърдение:

**Един език е от тип 0 тогава и само тогава, когато е рекурсивно номерируем, т. е. когато се разпознава от машина на Тюринг.**

### Упражнения

1. Постройте машина на Тюринг, която премества с една клетка надясно даден блок от 0 и 1.

2. Постройте машина на Тюринг, която на мястото на дадена входна дума записва друга дадена дума със същата дължина.

3. Дадена е граматиката $\Gamma=(\{a, b\}, \{S\}, S, \{S \to aSb, S \to bSa, S \to ab, S \to ba\})$.

Постройте машина на Тюринг M, за която $T(\mathrm{M})=L(\Gamma)$

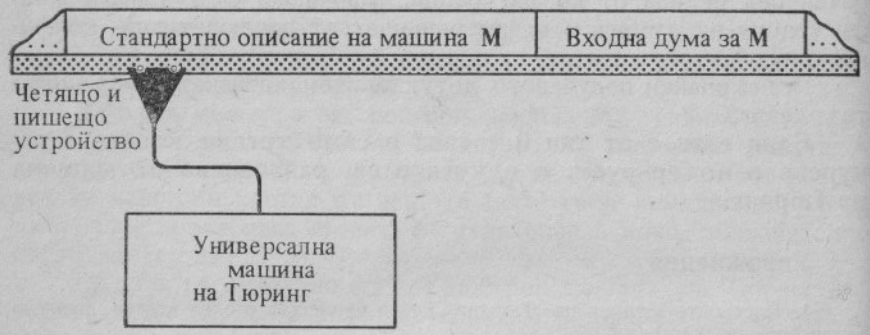
## 4.3 АЛГОРИТМИЧНО НЕРАЗРЕШИМИ ПРОБЛЕМИ ЗА ЕЗИЦИТЕ ОТ ТИП 0

В работата си от 1936 год. Алан Тюринг показа как може да се построи **универсална машина,** която да моделира работата на всяка отделна машина от определения от него клас. Универсалната машина на Тюринг получава входна дума, състояща се от стандартно описание на функцията на преходите на коя да е машина **M** и от още някаква дума $\omega$. След това тя имитира действията, които би извършила машината **M** върху входната дума $\omega$ и спира в заключително състояние тогава и само тогава, когато и **M** би спряла в заключително състояние.

Универсалната машина на Тюринг може да се разглежда като предшественик и идеален математически модел на всеки универсален компютър. Действително на компютъра обикновено се задава програма, написана на някакъв език за програмиране и изходни данни. Компютърът преработва програмата на машинен език и след това започва по тази програма да обработва изходните данни. Разбира се, универсалната машина на Тюринг като идеален

математически модел може да имитира работата на всяка машина на Тюринг, докато всеки реален универсален компютър има ограничени възможности за реализация на една или друга програма при едни или други изходни данни.

Получаваме следната приблизителна схема за универсалната машина на Тюринг (фиг. 48):




Фиг. 48

Да изясним какво означават думите **„стандартно описание на машина M“.**

Лентовите символи на всяка машина на Тюринг можем да вземаме от един безкраен списък на символи $S_1, S_2, \ldots, S_n, \ldots$ На всяка дума $S_{i_1} S_{i_2} \ldots S_{i_k}$ от тези символи съпоставяме числото $2^{i_1} 3^{i_2} \ldots p_k^{i_k}$, където 2, 3, . . . , $p_k$, . . . е редицата на простите числа, $p_k$ е $k$-тото просто число. Двоичният запис на числото $2^{i_1} 3^{i_2} \ldots p_k^{i_k}$ ще наричаме **код** на думата $S_{i_1} S_{i_2} \ldots S_{i_k}$. Например код на думата $S_3 S_1$ ще бъде двоичният запис 11000 на числото $2^3 . 3^1 = 24$.

Аналогично, вътрешните състояния на всяка машина на Тюринг можем да вземаме от безкрайния списък $q_1, q_2, q_3, \ldots, q_m, \ldots$ От тези символи няма да образуваме думи, така че е достатъчно на всеки от тях да съпоставяме някакво двоично число — например двоичния запис на индекса му: на $q_1$ съпоставяме 1, на $q_2$—10, на $q_3$—11 и т. н. Можем да приемем, че машините на Тюринг, които по-нататък ще разглеждаме, работят с лентова азбука, съставена от 0, 1 и $B$ ($B$ е празната клетка). Това е така, защото за всяка машина на Тюринг може да се построи друга машина на Тюринг с лентови символи 0, 1, $B$, която ще разпознава една дума от $\{0, 1\}^*$ тогава и само тогава, когато тази дума е код на



дума, разпознавана от първата машина. Можем също така да приемем, че началното състояние на всяка машина на Тюринг ще бъде означавано с $q_1$, а заключително ще бъде само състоянието $q_2$.

Следвайки работата на Алан Тюринг, ще приемем функцията на преходите на всяка машина да представяме във вид на таблица с редове $q_1$, $q_2$, . . . и стълбове 0, 1 и $B$ (този ред е фиксиран); в клетката, съответстваща на реда $q_i$ и например на стълба 1 ще пишем тройката, която ни дава като стойност функцията на преходите $\delta(q_i, 1)$.

Ще въведем освен 0, 1 и $B$ един допълнителен символ $*$.

Да вземем за пример следната машина на Тюринг:

$\mathbf{M}_1 = (\{q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q_2\})$ с функция $\delta$:

$\delta(q_1, 0) = (q_3, 1, R)$

$\delta(q_3, 0) = (q_4, 1, R)$

$\delta(q_4, 1) = (q_2, 1, L)$.

Тази функция има следната таблица:

| | 0 | 1 | $B$ |
|---|---|---|---|
| $q_1$ | $q_3, R, 1$ | — | — |
| $q_2$ | — | — | — |
| $q_3$ | $q_4, R, 1$ | — | — |
| $q_4$ | — | $q_2, 1, 1$ | — |

Сега ще дадем стандартното описание на машината $\mathbf{M}_1$. То се състои от групи: група от тройки на първото състояние, група от тройки на второто състояние и т. н. Групите ще отделяме една от друга с две звездички $*\ *$. Всяка група се състои от тройките, съответстващи на реда на състоянието, взети отляво на дясно. Местата, където има чертичка, ще означаваме само с 0, а символите от останалите тройки ще записваме в посочения в таблицата ред, като вместо състоянията пишем съответния им двоичен запис. Тройките от една група отделяме с по една звездичка $*$ . Началото и края на стандартното описание отбелязваме с три звездички $*\ *\ *$. Стандартното описание на машината $\mathbf{M}_1$, която взехме като пример, е следното:

$*\ *\ *\ 11\,R\,1 * 0\ *0 * *\ 0\ *0\ *0\ *\ *100\,R\,1 * 0 * \ 0 * *\ \ 0 * 10\,L\,1\ *0\ *\ *\ *$.

Стандартното описание на една машина $\mathbf{M}$ ще означаваме с $\langle \mathbf{M} \rangle$.

Вижда се, че функцията на преходите на всяка машина на Тюринг може да я зададе във вид на стандартно описание, а по дадено стандартно описание машината може да се възстанови еднозначно.

И така, на всяка машина на Тюринг поставихме в съответствие дума върху азбуката

$$\{0, 1, B, R, L, N, *\}$$

Ще номерираме думите върху тази азбука по начина, който вече използувахме в началото на книгата. Думите ще подреждаме по тяхната дължина. Думите с равна дължина ще нареждаме помежду им в съответствие с това, какво число представя тази дума в числовата система с основа 7 (в която 0, 1, *B*, *R*, *L*, *N*, * играят ролята на цифри).

Аналогично могат да се подредят и думите върху азбуката

$\{0, 1\}$: Λ, 0, 1, 00, 01, 10, 11, 000, . . .

Ще считаме, че на всяка дума от азбуката $\{0, 1, B, R, L, N, *\}$ съответствува машина на Тюринг. Ако по думата не може да се възстанови никаква функция на преходите, на тази дума ще отговаря машина, която нищо не прави върху коя дя е входна дума, т. е. чийто език е празен.

В такъв случай **на всяко естествено число *n* може да се съпостави машина на Тюринг — това е машината, която съответствува на *n*-тата дума в номерацията на думите върху** $\{0, 1, B, R, L, N, *\}$.

Обратното, както вече видяхме, също е вярно. **На всяка машина на Тюринг отговаря определено стандартно описание, чийто пореден номер в редицата на думите върху** $\{0, 1, B, R, L, N, *\}$ **можем да съпоставим на тази машина.**

С други думи, можем да направим списък на машините на Тюринг: $M_0$, $M_1$, $M_2$, $M_3$, . . . , $M_n$ . . . , в който ще се среща всяка машина.

Аналогично, имаме и списък на думите върху азбуката $\{0, 1\}$: $\omega_0$, $\omega_1$, $\omega_2$, $\omega_3$, . . . , $\omega_n$, . . .

Сега да се върнем към универсалната машина на Тюринг. Може да се докаже, че съществува **универсална машина на Тюринг U** (и това го е направил за първи път самият Тюринг), която разпознава думата $\alpha$ тогава и само тогава, когато $\alpha = \langle M \rangle \omega$ за някаква машина на Тюринг M, която разпознава ω.

Да разгледаме езика, който универсалната машина U разпознава:

$T(U) = \{\langle M \rangle \omega$; M разпознава $\omega\}$.

Очевидно е, че $T(U)$ е рекурсивно номерируем език (тъй като се разпознава от машина на Тюринг).

Да допуснем, че $T(U)$ е рекурсивен език. Това означава, че съществува машина на Тюринг M, не обезателно U, която разпознава $T(U)$ и спира след краен брой тактове за всяка входна дума.

Тогава можем да построим и следната машина на Тюринг N. По дадена входна дума $\omega \in \{0,1\}^*$ машината N намира номера *i* на думата ω в списъка на думите от $\{0,1\}^*$. Това тя би могла да направи, като сравнява ω с първата дума, след това с втората и т. н. По номера *i* тя намира *i*-та дума $\langle M_i \rangle$ от списъка на думите върху азбуката $\{0, 1, B, R, L, N, *\}$ и образува думата $\langle M_i \rangle \omega_i$. След това включва хипотетичната машина M (която разпознава $T(U)$ и спира винаги) върху тази входна дума. След краен брой стъпки машината M спира. Ако машината M е спряла в незаключително състояние, тогава машината N включва преход към своето заключително състояние. Ако M е спряла в заключително състояние, тогава машината N включва машина, която работи неограничено дълго — например осъществява непрекъснат преход надясно върху лентата си.



От тази конструкция се вижда, че машината на Тюринг N ще разпознава думата $\omega_i$ тогава и само тогава, когато машината $M_i$ не я разпознава.

Както всяка машина на Тюринг, машината N също ще се среща в списъка на машините на Тюринг и ще има някакъв пореден номер в този списък — например *l*.

Но тогава да приложим машината N, т. е. $M_l$, към думата $\omega_l$. Получаваме, че $M_l$ разпознава $\omega_l$ тогава и само тогава, когато $M_l$ не я разпознава. Или с други думи: $\langle M_l, \omega_l \rangle \in T(U)$ тогава и само тогава, когато $\langle M_l, \omega_l \rangle \notin T(U)$. Получаваме противоречие. В такъв случай нашето допускане, че $T(U)$ е рекурсивен език не е вярно. Следователно, $T(U)$ не е рекурсивен език.

С това скицирахме начина на доказателство на следния основен за теорията на формалните езици резултат:

**Съществува рекурсивно номерируем език, който не е рекурсивен.**

Такъв език е езикът, разпознаван от универсалната машина на Тюринг.

Оттук можем да получим алгоритмичната неразрешимост на проблема за разпознаването на всички съдържателни свойства на езиците от тип 0 (или още — на рекурсивно номерируемите езици). Ще припомним, че проблемът за разпознаване на някакво свойство на формалните езици наричаме алгоритмично разрешим, ако съществува алгоритъм, който да определя дали произволен формален език притежава това свойство или не. Проблемът е алгоритмично неразрешим, ако такъв алгоритъм не съществува.

Рекурсивно номерируемите езици могат да притежават или да не притежават различни свойства. Например могат да бъдат или да не бъдат празни, крайни или безкрайни, съществено неоднозначни, автоматни, безконтекстни, контекстни и т. н.

Ако едно свойство на рекурсивно номерируемите езици се притежава или не се притежава едновременно от всички рекурсивно номерируеми езици, такова свойство ще наричаме **тривиално.** Тривиално свойство на рекурсивно номерируемите езици е да бъдат рекурсивно-номерируеми. Нетривиални са например всички по-горе изброени свойства, тъй като има рекурсивно номерируеми езици, които ги притежават, и такива, които не ги притежават.

Лесно се вижда, че ако проблемът за разпознаване на някакво свойство е алгоритмично разрешим, тогава алгоритмично разрешим е и проблемът за разпознаване на отрицанието на това свойство. Действително, ако има алгоритъм, който по произволен рекурсивно номерируем език да определя, дали той притежава дадено свойство или не, този алгоритъм ще определя и кога произволен език не притежава това свойство. С други думи, **проблемите за разпознаването на едно свойство и на неговото отрицание са едновременно алгоритмично разрешими или неразрешими,** и е безразлично за кой от двата проблема ще доказваме алгоритмичната му разрешимост или неразрешимост.

Нека $\Omega$ е произволно нетривиално свойство на рекурсивно номерируемите езици. Без ограничения на общността можем да считаме, че празното множество $\varnothing$ не притежава това свойство, тъй като в противен случай ще вземем отрицанието на свойството $\Omega$.

От нетривиалността на свойството $\Omega$ следва, че има и рекурсивно номерируем език $L$, който притежава $\Omega$. Нека $L$ се разпознава от машината на Тюринг $\mathbf{M}_L$.

Да вземем произволна машина на Тюринг $\mathbf{M}$ и произволна дума $\omega$. По тях и по $\mathbf{M}_L$ можем да построим следната машина на Тюринг $\mathbf{M}'$. Нека на $\mathbf{M}'$ е зададена входна дума $\alpha$. Машината $\mathbf{M}'$ включва отначало машината $\mathbf{M}$ върху входната дума $\omega$. Ако $\mathbf{M}$ не е определена, тогава $\mathbf{M}'$ е също неопределена. Ако машината $\mathbf{M}$ спре в незаключително състояние, машината $\mathbf{M}$ също спира в незаключително състояние. И накрая, ако $\mathbf{M}$ спре в заключително състояние, $\mathbf{M}'$ включва машината $\mathbf{M}_L$ върху входната дума $\alpha$.

Вижда се, че езикът $T(\mathbf{M}')$ е или празен, или е езикът $L$ в зависимост от това дали машината $\mathbf{M}$ разпознава думата $\omega$. Но $\varnothing$ не притежава $\Omega$, а езикът $L$ има свойството $\Omega$. Получаваме, че $T(\mathbf{M}')$ притежава свойството $\Omega$ тогава и само тогава, когато $\mathbf{M}$ разпознава $\omega$, т. е. когато $\langle \mathbf{M} \rangle \omega \in T(\mathbf{U})$.

Сега да допуснем, че проблемът за разпознаване на свойството $\Omega$ е алгоритмично разрешим. Този алгоритъм ще определя и за езиците $T(\mathbf{M}')$ дали притежават, или не свойството $\Omega$. Тогава той ще определя и дали $\langle \mathbf{M} \rangle \omega \in T(\mathbf{U})$, или не. По алгоритъма



бихме могли да построим машина на Тюринг, която за краен брой стъпки да разпознава дали $\langle \mathbf{M} \rangle \omega \in T(\mathbf{U})$. Както обаче доказахме, такава машина не може да има, тъй като $T(\mathbf{U})$ не е рекурсивен език. Получаваме противоречие.

Следователно допускането, че проблемът за разпознаване на нетривиалното свойство $\Omega$ е алгоритмично разрешим, не е вярно, така че този проблем е алгоритмично неразрешим.

С това получихме следния интересен резултат, известен като **теорема на Райс:**

**Проблемът за разпознаване на кое да е нетривиално свойство на рекурсивно номерируемите езици е алгоритмично неразрешим.**

## Упражнения

1. Докажете, че допълнението на езика $T(\mathbf{U})$, разпознаван от универсалната машина на Тюринг $\mathbf{U}$, не е рекурсивен език.
2. Докажете, без да използувате теоремата на Райс, че проблемът за разпознаване на това дали произволен рекурсивно номерируем език е рекурсивен, е алгоритмично неразрешим.
3. Посочете различни нетривиални свойства на рекурсивно номерируемите езици и докажете тяхната нетривиалност.

*„Все по-близо е върхът; още малко усилие и край на Сизифовия труд! Но камъкът се изплъзва от ръцете му и с трясък се търкулва надолу, като вдига облаци прах. И Сизиф отново се залавя за работа.*

*Така вечно търкаля Сизиф камъка и никога не може да стигне до целта — до върха на планината.“*

Н. Кун *„Старогръцки легенди и митове“*

## Глава 5

## КОНТЕКСТНИ ЕЗИЦИ И ЛИНЕЙНО ОГРАНИЧЕНИ АВТОМАТИ

Както вече изтъкнахме, разпознавателната сила на един автомат до голяма степен се определя от организацията на външната му памет. Машините на Тюринг могат да четат и пишат на всяко място на неограничената си лента. Магазинните автомати могат да прочитат само най-левия (или най-горен) символ на магазинната лента и да записват нова дума само на неговото място. Магазинната памет на всеки магазинен автомат е линейна функция на дължината на входната дума, така че тя е ограничена.

Междинно място между машините на Тюринг и магазинните автомати заемат **линейно ограничените автомати:** те могат да четат и да пишат на всяко място на входната си лента, но дължината на тяхната входна лента е определена от мястото, което е необходимо за входната дума.

И така, линейно ограниченият автомат прилича на недетерминирана машина на Тюринг — може да чете и пише, да се движи наляво и надясно, но не може да излиза извън клетките, в които е била написана входната дума. Обикновено това се постига, като на двете страни на входната дума се поставят ограничителни знаци — ляв и десен ограничителни маркери. Четящото устройство не може да премине през тях, а автоматът не може да ги придвижи.



Формално, **линейно ограничен автомат** наричаме наредената шесторка $A = (K, \Sigma, \Gamma, \delta, q_0, F)$, като компонентите на шесторката имат традиционното си вече съдържание: $K$ е азбука на вътрешните състояния, $\Sigma$ е входна азбука и съдържаща специални символи — **маркери** $*$ и $\bigcirc$, $\Gamma$ е лентова азбука, която включва входната азбука, $\delta$ е функция на преходите, която на наредени двойки („вътрешно състояние“, „лентов символ“) поставя в съответствие множества от наредени тройки от вида („вътрешно състояние,“ „лентов символ,“ „един от символите $L$, $R$, $N$“), $q_0$ е начално състояние, а $F$ е подмножество на $K$, съставено от заключителните състояния. Входната дума се задава във вида $*\omega\bigcirc$, $\omega\in\{\Sigma - \{*, \bigcirc\}\}^*$.

Работата на линейно ограничения автомат се определя както при недетерминираната машина на Тюринг, като функцията на преходите не позволява преминаване през символите $*$ и $\bigcirc$, тяхното преместване или изтриване. Конфигурацията на линейно ограничения автомат при всеки такт изглежда така $*\,\omega_1\, q\, \omega_2\, \bigcirc$, а дължината на думата $*\omega_1 q \omega_2 \bigcirc$ е равна на дължината на входната дума $*\omega\bigcirc$.

Една входна дума се разпознава от линейно ограничения автомат, ако след краен брой тактове автоматът спре в заключително състояние.

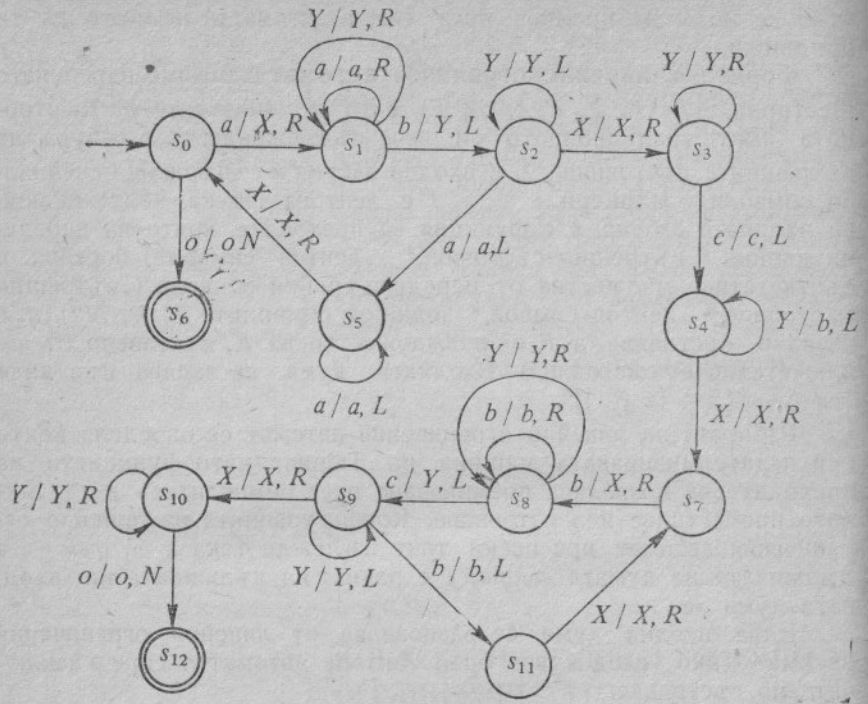
Езикът, разпознаван от даден линейно ограничен автомат, се образува от всички входни думи, които автоматът разпознава.

Диаграмата на преходите за линейно ограничени автомати се определя, както при машините на Тюринг.

Например линейно ограниченият автомат от фиг. 49, зададен с диаграмата на преходите си, разпознава езика $L=\{a^n b^n c^n;\ n\geq 0\}$.

Линейно ограничените автомати представляват разпознавателите на контекстните езици. Може да се докаже, че **един език е контекстен тогава и само тогава, когато се разпознава от линейно ограничен автомат.**

Доказателството използува същия метод, който приложихме, когато показвахме еквивалентността на машините на Тюринг и граматиките от тип 0, с допълнителни съображения, които изхождат от това, че контекстните граматики са монотонни и следователно никоя междинна дума в кой и да е извод не надвишава по дължина извежданата дума. Това в обратен ред съответствува на факта, че при линейно ограничените автомати допустимата лента се определя от входната дума.




Фиг. 49

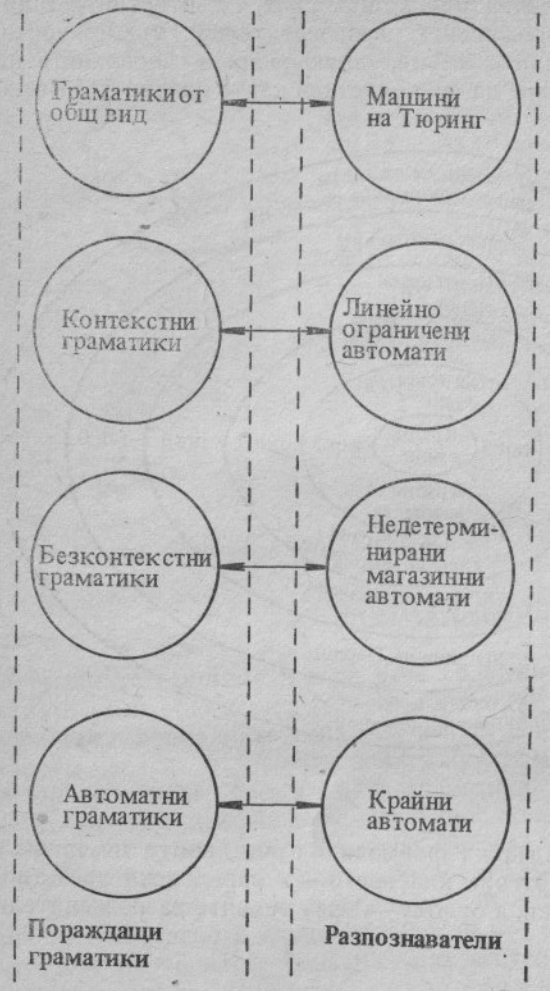
Съществуват ли езици от тип 0, които да не са контекстни? Не е никак лесно да се намерят съответни примери. Но ние всъщност имаме вече един пример: това е езикът $T$ (U), разпознаван от универсалната машина U. Действително, както установихме, $T$ (U) е рекурсивно номерируем, но не е рекурсивен език. Знаем обаче, че контекстните езици са рекурсивни — това изведохме в началото на книгата. Следователно $T$ (U) е език от тип 0 поради еквивалентността на машините на Тюринг с граматиките от тип 0, но не е контекстен, заради рекурсивността на контекстните езици.

И така, **съществува формален език от тип 0, който не е контекстен.**

* * *

Формалните езици в съответствие с начина им на задаване разделихме на няколко класа. Преди всичко отделихме онези формални езици, които имат крайни описания — пораждащи граматики и разпознаватели. Изследвайки тези две крайни описа-

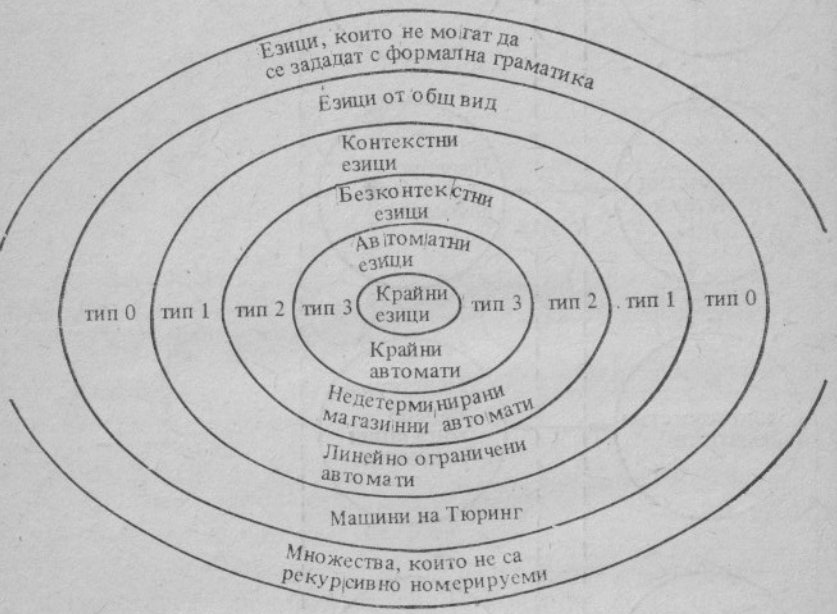





Фиг. 50

ния, установихме еквивалентността на въведените от нас видове, пораждащи граматики със съответни видове разпознаватели по отношение на описваните от тях езици (фиг. 50).

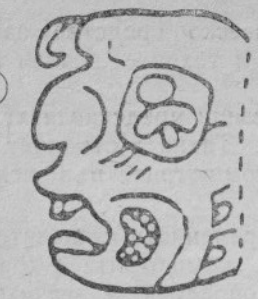
Същевременно установихме, че различните видове пораждащи граматики имат различна пораждаща сила, а съответните типове разпознаватели — различни разпознавателни възможности. Йерархията на формалните езици, включваща и йерархията на Чомски, схематично може да се представи както нафиг. 51. Всеки




Фиг. 51

следващ клас езици съдържа формални езици, които не принадлежат на предишния. Отгоре класовете са определени чрез типа пораждащи ги граматики, а отдолу — чрез техните разпознаватели.






*„А по цялата земя се употребяваше един език и един говор .... И рече Господ: Ето, едни люде са, и всички говорят един език; и това е що са почнали да направят; и не ще може вече да им се възбрани каквото да е било нещо що биха намислили да направят. Елате, да слезем, и там да разбъркаме езика им, тъй щото един други да не разбират езика си. Така Господ ги разпръсна от там по лицето на цялата земя; а те престанаха да градят града. За това той се именува Вавилон . . .“*

*Библия, „Битие“ гл. 11.*

## Глава 6

## ФОРМАЛНИ И ЕСТЕСТВЕНИ ЕЗИЦИ

Формалните езици за нас са вече сравнително добре познати математически обекти, които знаем как да задаваме и как да изследваме. Формалните езици са изкуствени езици, тяхната форма и обем зависят от нашата воля и нужди и поради това можем да ги опишем точно чрез тяхната граматика. От математическа гледна точка това положение е напълно приемливо: едно изречение (или дума) принадлежи на езика тогава и само тогава, когато удовлетворява граматичните критерии.

Съвсем друго е положението при естествените езици. Както знаем, естествените езици са обективна реалност. Те могат да се изучават и описват с една или друга степен на формализация, но не могат да се формират според желанието на изследователя. Естествените езици се изменят в процеса на тяхната еволюция. Дългото историческо развитие на повечето естествени езици ги е направило твърде сложно, но ефикасно средство за общуване и познание. Формалната граматика на всеки естествен език е само добро или лошо приближение към неговата реална граматика. Поради това формалните езици и техните граматики представля-

ват само едно, макар и основно, математическо средство за изучаване и описване на естествените езици и техните реални граматики.

Формалните аспекти на естествените езици представляват интерес не само за изследователя-лингвист. Тяхната стойност все повече нараства с бурното навлизане на компютрите във всички сфери на обществения живот.

Характерен белег в развитието на съвременната изчислителна техника е непосредственото свързване на компютрите с хората. Способността за обработка на символи определя в значителна степен комуникационните отношения между хората. Възможността на съвременните компютри да обработват данни в символна форма е основната предпоставка да се говори за изкуствения интелект като свойство на компютрите. Благодарение на тази възможност те могат да се включват в комуникационните отношения между хората, да анализират и синтезират текстове и реч, да разпознават образи и сцени, да резгират на тях. Във връзка с това все по-важно място заема диалога между човека и компютъра. Налага се да се разработват нови начини за общуване с компютрите, съответстващи на тяхната роля в обществото.

Разработването на подходящи езикови средства, позволяващи общуване с компютъра в рамките на фрагменти от естествените езици (с някои ограничения върху граматичната структура на изреченията и ограничения върху предметната област), значително разширява възможностите за ефективно и масово използуване на компютрите в огромен брой области на приложение.

Това извежда на преден план проблема за изучаване на връзките между естествените и изкуствените, формални, езици в тяхното единство и взаимодействие.

При общуване с компютър на език, близък до естествения, един от основните въпроси е как да се опише поне онази част от естествения език, която играе най-съществена роля в общуването между хората.

Има два тривиални подхода. Първият възможен подход е да се направи списък на всички интересуващи ни изречения от най-съществената част на езика. Естественият език обаче представлява отворена система и един такъв списък трябва да бъде допълван непрекъснато. Освен това понятията „всички интересуващи ни изречения“ и „най-съществена част на езика“ имат твърде субективен характер и не могат да бъдат определящи при задаването на естествения език. Този подход, както се вижда, е твърде ограничен и неефективен.



Вторият възможен подход е да се състави ръководство, чрез което да се определя кои изречения попадат в интересуващата ни част и кои са извън нея. Този подход се използува обикновено при обучаването на различни изкуствени езици. Такова едно ръководство превръща използувания фрагмент от езика в изкуствен, формален език, тъй като то регламентира кое изречение е от езика и кое не е. Ако желаем да запазим основните, характерни отлики на естествения език, не можем да използуваме и този подход.

Счита се, че има поне три основни аспекта на лингвистичното описание: **синтактика, семантика и прагматика.** Докато в изкуствените езици тези аспекти могат да се отделят един от друг, в естествените езици те са твърде тясно, органически преплетени и са свойствени на всеки езиков фрагмент.

**Синтактиката** е онзи аспект на езика, който определя правилността на построяването на разглеждания езиков фрагмент. Към нея спада изучаването на вътрешната структура на текста и на отношенията между символите на езика. Елементарните езикови символи могат да се съчетават само по определени правила, които образуват граматиката на езика. Синтактичният анализ определя граматическата правилност или неправилност на езиковия текст.

**Семантиката** отразява смисловото значение на езиковите символи или на групи от езикови символи. Тя изучава отношенията между знаците и означаваните с тях неща, между смисъла на отделните символи и смисъла на съставения от тях текст.

Накрая, **прагматиката** е онзи аспект на езика, който отразява възприемането на знаците, произхода, историята и използуването им в разглеждания контекст.

Всяка крайна система — граматика, целяща да постигне подобро описание на естествения език, трябва да отчита по възможност повече аспекти на езика. По-нататък ще разгледаме различни подходи към формалното описание на естествените езици.

### 6.1. СТРУКТУРНИ ОПИСАНИЯ

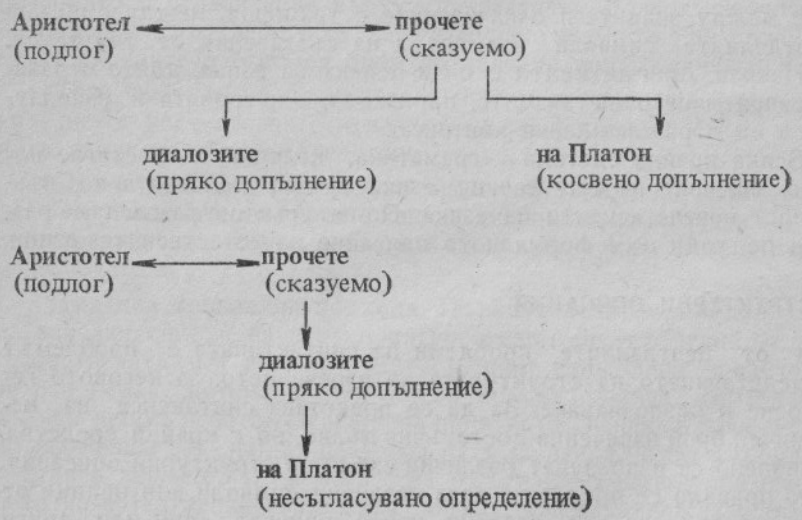
Един от централните проблеми на синтактиката е проблемът за представянето на структурата на изречението, за неговото генериране и разпознаване. За да се представи синтаксиса на неограничен брой изречения достатъчно пълно, но с крайни средства, обикновено се използуват различни схеми на структурни описания. Всяко правило се представя чрез различни символи или редици от символи, свързани със стрелки, които описват едни или други

структурни връзки между тях. В първа глава вече приведохме пример на подобно структурно описание на изречения от българския език и построихме сравнително проста структурна граматика, която да описва не само синтактичната структура на едно изречение, но и на един твърде ограничен фрагмент от езика.

Създадени са редица лингвистични теории, които по различен начин описват структурата на изречението в съответствие с това какви основни синтактични единици се определят и какви връзки се установяват между тях.

Най-популярна от тези теории е „училищната граматика", която оперира с така наречените **части на изречението**. Части на изречението са такива функционални единици като подлог, сказуемо, допълнение, определение и т. н., които могат да се състоят от една или повече думи и общо изпълняват определена роля в изречението. Има два типа връзки между функционалните единици: взаимноподчинителни и подчинителни. Взаимноподчинителна е връзката между подлога и сказуемото, тъй като те имат един и същ най-висок ранг в синтактичната йерархия и никоя от тези единици не доминира над другата. Подчинителни са връзките на подлога и сказуемото с останалите части на изречението, както и връзките между тези части на изречението, например между сказуемото и допълнението, между допълнението и неговото определение и т. н. Подчинителните връзки определят и коя част на изречението с коя част трябва да съгласува своята форма.

Фиг. 52






Да вземем например изречението **„Аристотел прочете диалозите на Платон"**, което вече разгледахме във връзка с неговата синтактична неоднозначност. Неоднозначно е и представянето на структурата на частите на това изречение. Получаваме следните двете структурни описания от фиг. 52.

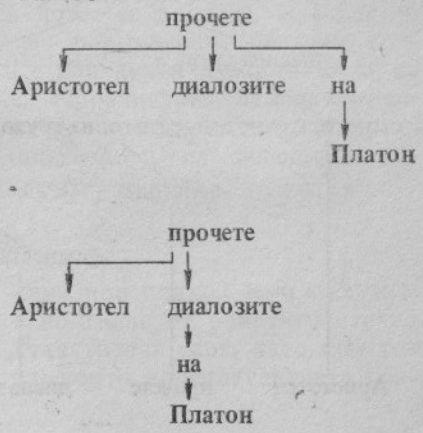
Изборът на това, кое от двете структурни описания отразява структурата на изречението, може да се извърши само на базата на вложения в изречението смисъл, т. е. на неговата семантика.

Друга теория, която структурно описва изреченията, е **граматиката на зависимостите**. Тя приема за единици отделните думи, съставящи изречението. Всички връзки между отделните единици са подчинителни, като глаголът — сказуемото, не може да е подчинен на никоя друга единица. Единиците се характеризират например като съществително, предлог, глагол, спомагателен глагол и т. н. Структурното описание на изреченията съгласно граматиката на зависимостите представлява дърво, в корена на което е поставен глаголът. За разглеждания от нас пример получаваме следните две описания съобразно граматиката на зависимостите, дадени на фиг. 53.

И граматиката на зависимостите, и граматиката на частите на изречението недостатъчно пълно отразяват реда на думите в изречението, тъй като съответните структурни описания не определят еднозначно линейната структура на изречението.

По-подробно ще се спрем на така наречената **конституентна граматика (граматика на непосредствено съставящите, НС—граматика)**, в която структурното описание се представя чрез определен вид пораждащи граматики. Единиците на тази граматика се наричат непосредствено съставящи и се получават, като изречението и вече определените непосредствено съставящи се разделят на не повече от две максимални по обем конструкции, всяка от които отново образува непосредствено съставяща. Когато непосредствено съставящата представлява дума (или морфема)тя по-нататък не се дели на непосредстве-

Фиг. 53

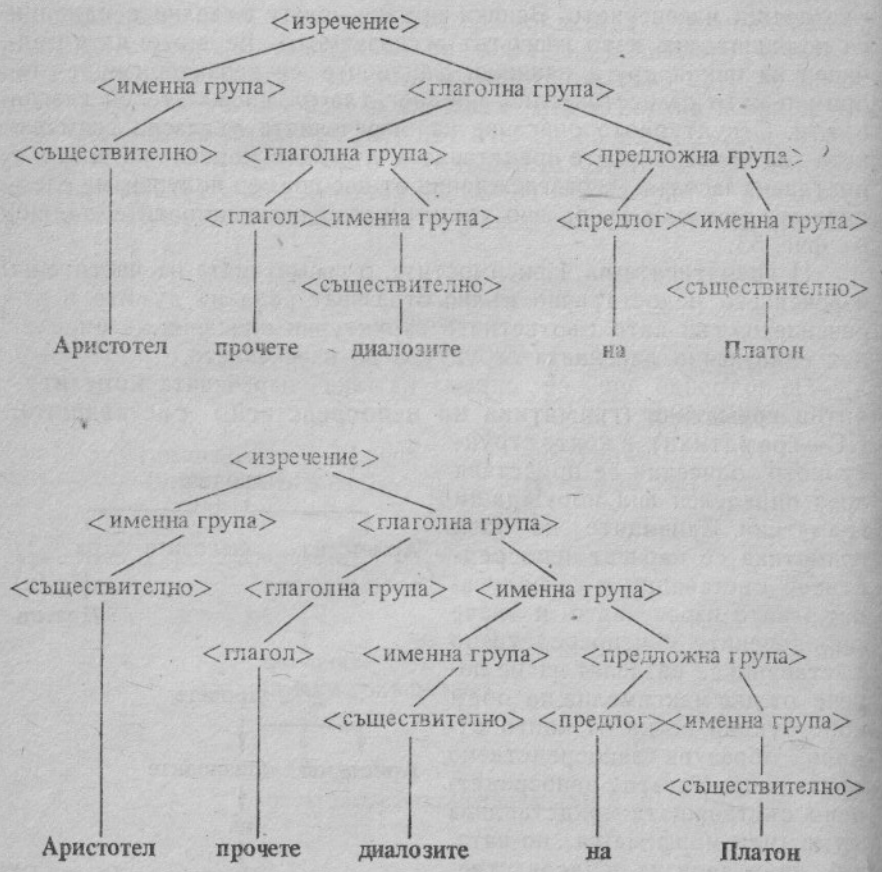


но съставящи. Непосредствено съставящите са или граматични категории такива като ⟨изречение⟩, ⟨именна група⟩, ⟨предложна група⟩, ⟨глаголна група⟩, ⟨съществително⟩, ⟨глагол⟩, ⟨предлог⟩ и т. н., или думи от езика. Връзките между непосредствено съставящите се получават в резултат на последователното линейно разчленяване на изречението.

Графически конституентната граматика се представя чрез дървета, които се образуват както дърветата на извод за безконтекстните граматики.

За разглеждания от нас пример получаваме следните две различни графични описания (фиг. 54):

Фиг. 54




В горните две структурни описания на изречението **„Аристотел прочете диалозите на Платон“** са използувани следните граматични правила:

⟨изречение⟩ → ⟨именна група⟩ ⟨глаголна група⟩,
⟨глаголна група⟩ → ⟨глаголна група⟩ ⟨именна група⟩,
⟨глаголна група⟩ → ⟨глаголна група⟩ ⟨предложна група⟩,
⟨именна група⟩ → ⟨именна група⟩ ⟨предложна група⟩,
⟨предложна група⟩ → ⟨предлог⟩ ⟨именна група⟩,
⟨именна група⟩ → ⟨съществително⟩,
⟨глаголна група⟩ → ⟨глагол⟩.
⟨съществително⟩ → **Аристотел** |**Платон**| **диалозите,**
⟨глагол⟩ → **прочете,**
⟨предлог⟩ → **на.**

Тези правила имат следния граматичен смисъл: изречението се състои от именна група и глаголна група, глаголната група представлява глаголна група, последвана от именна група или от предложна група и т. н.

Можем да се опитаме, както това е прието в теорията на формалните езици и граматики, да използуваме тази граматика за пораждане на съответния й език. В него несъмнено ще попадне разглежданият от нас пример, както и още редица други изречения. Някои от тях обаче въпреки правилната им структура трудно могат да бъдат причислени към българския език. Освен това породеният език съдържа твърде малко изречения на българския език. За да приближим пораждания език до разглеждания естествен език — българският, трябва да изменим някои правила така, че неприемливите изречения да бъдат изхвърлени, и да добавим нови правила за разнообразяване на граматичните структури на пораждените изречения.

Този процес в крайна сметка довежда до това, че едва ли не за всяка дума от езика трябва да формулираме правила, които да определят по какъв начин и с кои други думи тя може да се съчетава в осмислени изречения. Поради това, въпреки редица важни преимущества на конституентните граматики, те представляват едно полезно, но не единствено и изчерпващо средство за структурно описване на естествения език.

### 6.2. ТРАНСФОРМАЦИОННА ГРАМАТИКА

През 1957 г. Н. Чомски предлага един нов подход към създаването на формални граматики, които по-пълно от конституентната граматика описват граматиката на естествения език, като при това запазват основните преимущества, които конституентната гра-

матика притежава. Прието е тези граматики да се наричат **трансформационни (трансформационно пораждащи)** граматики.

Построена в началото върху чисто синтактични основи, трансформационната граматика в своето развитие демонстрира определен стремеж да обхване и смисъла като компонента на езиковото описание.

Подобно на конституентната граматика трансформационната граматика представлява генератор на възможните правилно построени изречения от подмножества на естествения език, но за разлика от нея предполага различни нива в описанието на езика. Нивото, което определя семантичната интерпретация на изречението представлява неговата **дълбинна структура**; нивото, определящо фонетичната форма, образува неговата **повърхностна структура.**

В съответствие с това трансформационната граматика се състои от две компоненти: **базова компонента и трансформационна компонента.** Базовата компонента представлява конституентна граматика, която поражда различни изречения, заедно с техните структурни описания — дървета. Тези изречения обаче не описват изреченията от разглеждания езиков фрагмент, а са носители на смисъла им. Трансформационната компонента представлява наредено множество от **трансформационни правила (трансформации)**, които преработват дълбинните структури в повърхностни, т. е. — в изреченията от езиковия фрагмент.

Породените от базовата компонента дървета се подават на първото трансформационно правило, което ги преработва в други дървета, те от своя страна се преработват от второто трансформационно правило и т. н. Ако някое трансформационно правило е неприложимо към дадено дърво, това дърво преминава без изменения към следващото трансформационно правило.

Трансформационните правила се състоят от две части: **структурно условие** и **структурна промяна.** Структурното условие представлява крайна редица от нетерминални символи (граматически категории) и променливи. Тази редица посочва какви формални свойства трябва да притежава преработваното дърво, за да се приложи към него трансформационното правило, и как да се раздели дървото на онези групи, които влизат в структурната промяна. Структурната промяна представлява крайна поредица от елементарни трансформации, които се състоят в изтриване, заместване и присъединяване на части от дървото, описани в структурното условие.

Да разгледаме един пример. Нека е дадено отново изречението **„Аристотел прочете диалозите на Платон“**, което можем



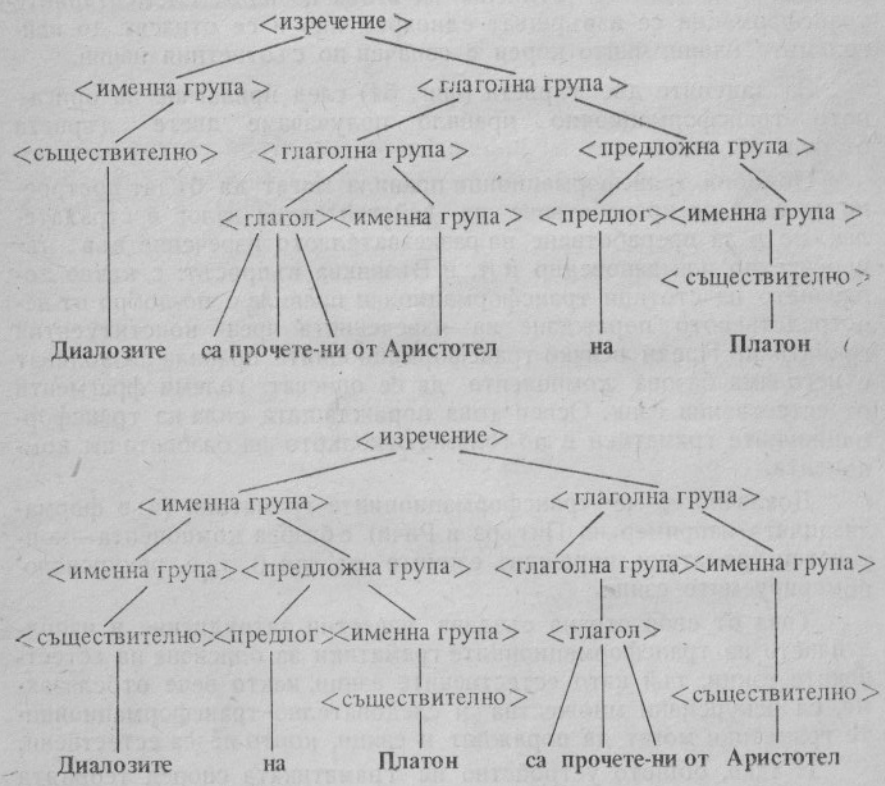
да породим чрез конституентна граматика по двата описани от нас начина. Да се опитаме да конструираме трансформационно правило, което да преработва действителния залог в страдателен. Ще опростим максимално трансформационното правило, за да не навлизаме в излишни за случая подробности.

Структурното условие на описваното трансформационно правило може да се представи така:

(⟨именна група⟩, ⟨глагол,⟩ ⟨именна група⟩, $X$)

1 2 3 4

Фиг. 55



Тук $X$ е променлива, чиито стойности са различни клони на дървета, а ⟨именна група⟩ и ⟨глагол⟩ са нетерминални символи — непосредствено съставящи в съответната конституентна граматика. Трансформационното условие означава, че трансформационното правило е приложимо към дървета, чиято корона може да бъде последователно разделена на четири части, първата и третата от които представляват корони на клони с начало ⟨именна част⟩, втората представлява корона на клон с начало ⟨глагол⟩, а четвъртата — корона на произволен клон.

Структурното условие на описваното трансформационно правило се състои от следните елементарни трансформации: заместване на първа позиция с трета позиция, заместване на трета позиция с първа позиция, присъединяване на „са“ отляво на втора позиция и на „-ни от“ отдясно на втора позиция. Елементарните трансформации се извършват едновременно и се отнасят до най-големите клони, чийто корен е означен по съответния начин.

За дадените две дървета (фиг. 54) след прилагане на описаното трансформационно правило получаваме двете дървета от фиг. 55.

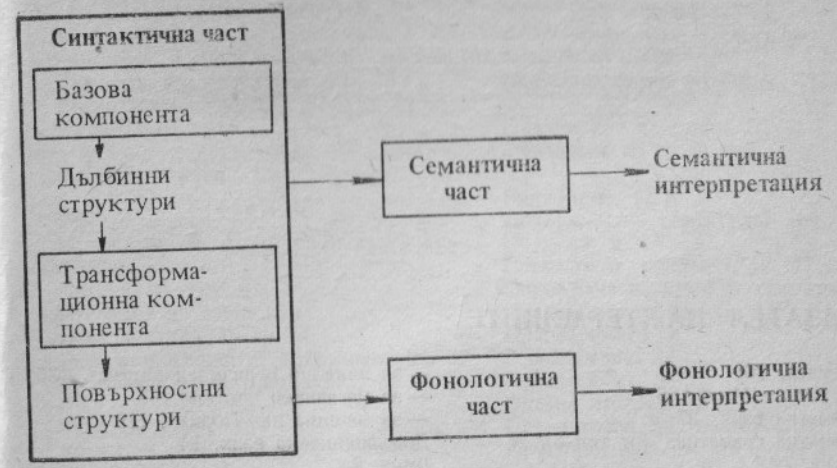
Подобни трансформационни правила могат да бъдат построени не само за преработване на действителния залог в страдателен, но и за преработване на разказвателното изречение във въпросително или заповедно и т. н. Възниква въпросът: с какво добавянето на стотици трансформационни правила е по-добро от непосредственото пораждане на изреченията чрез конституентни граматики? Преди всичко трансформационните правила позволяват с неголяма базова компонента да се описват големи фрагменти от естествения език. Освен това пораждащата сила на трансформационните граматики е по-голяма, отколкото на базовата им компонента.

Доказано е, че трансформационните граматики (във формализацията например на Питърз и Ричи) с базова компонента — контекстни граматики, пораждат езиците от тип 0, т. е. рекурсивно номерируемите езици.

Това от своя страна създава известни затруднения в използването на трансформационните граматики за описване на естествените езици, тъй като естествените езици, както вече отбелязахме, са рекурсивни множества и следователно трансформационните граматики могат да пораждат и езици, които не са естествени.

И така, общото устройство на граматиката според теорията на трансформационните граматики изглежда приблизително по следния начин (фиг. 56):






Фиг. 56

Съществуват обаче и модификации на тази схема, в които се допуска възможността повърхностната структура да влияе на семантичната интерпретация.

## УКАЗАТЕЛ НА ТЕРМИНИТЕ

## ЕРАТУРА

ладкий, А. В., И. А. Мельчук. Элементы математической лингвистики
., 1969.
хо, А., Дж. Ульман. Теория синтаксического анализа, перевода и ком.
иляции, т. I, т. II. М., 1978.
афаэл, Б., Думающий компьютер. М., 1979.
аккиман, У., Дж. Хорнинг, Д. Уортман. Генератор компиляторов.
., 1980.
асевич, В. Б., Элементы общей лингвистики. М., 1977.
rainerd, B. Introduction to the Mathematics of Language Study. N. Y., 1971.
ardenas, A., L. Presser, M. Marin. Computer Science. Wiley-Inter-
cience, 1972.
eckman, F. Mathematical Foundations of Programming. Addison-Wesley
Publ. Company, 1980.
Vall, R. Introduction to Mathematilcal Linguistics. Prentice-Hall.
Hopcroft, J., J. Ullman. Introduction to Automata Theory, Languages
and Computation. Addison Wesley Publ. Company, 1979.
ackhouse, R. Syntax of Programming Languages — Theory and Practice.
Prentice-Hall International, 1979.
Aho, A. (ed.). Currents in the Theory of Computing. Prentice-Hall, 1973.
00 задач по языковедению и математике. Издательство Московского универ-
ситета, Москва, 1972.
Іончев, В. Шрифтът през вековете. София, 1964.
Denning J., J. Dennis, J. Qualitz. Machines, Languages and
Computation. Prentice-Hall, Inc. Englewood Cliffs, N., J., 1978.

## СЪДЪРЖАНИЕ

*Радослав Димов Павлов*

МАТЕМАТИЧЕСКА ЛИНГВИСТИКА

Рецензенти *Йордан Пенчев Пенчев, Йордан Дечев Денев*
Зав. редакция *Копринка Михайлова*
Редактор *Недялка Лазарова*
Художник на корицата *Мария Димитрова*
Художник-редактор *Красимира Коцева*
Технически редактор *Лила Костова*
Коректор *Николина Иванова*
Калиграфи *Зорка Петрова* и *Силвия Терзиянова*

Код: 01 $\frac{9532221222}{2220-37-82}$

Българска. Издание I. Дадена за набор на 26. IV. 1982 г. Подписана за печат на 12. XI. 1982 г. Излязла от печат на 8. XII. 1982 г. Формат 60×84/16. Печ. коли 9.50. Изд. коли 8,86. УИК 9,20. Тираж 3500+105. Поръчка № 980. Цена 1,26 лв.

Държавно издателство „*Народна просвета*" — София
Държавна печатница „*Ат. Стратиев*" — Хасково


Цена 126 лева