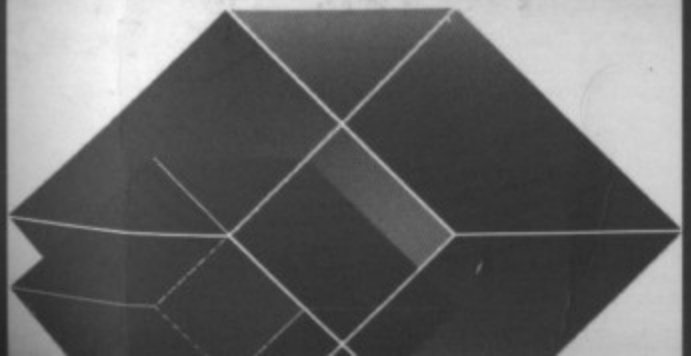
УВОД В ДИСКРЕТНАТА МАТЕМАТИКА

КРАСИМИР МАНЕВ

# УВОД В ДИСКРЕТНАТА МАТЕМАТИКА



лекции по информатик

ЛЕКЦИИ ПО ИНФОРМАТИКА

Красимир Манев

# УВОД В ДИСКРЕТНАТА МАТЕМАТИКА

Четвърто издание

КЛМН, София, 2005

СЕРИЯ "ЛЕКЦИИ ПО ИНФОРМАТИКА"

Красимир Манев
**Увод в дискретната математика**, Четвърто издание
1. Дискретна математика. 2. Теоретична информатика


**ISBN 954-535-136-5**

Печат и подвързия "Абагар" АД, Велико Търново

## Предговор към четвъртото издание

Уважаеми читателю,

Изминаха 10 години откакто започнах работа над тази книга. В ръцете ти е нейното Четвърто издание, което много прилича на предишните две по това, че основните изменения са опитите да се отстранят грешките, допуснати в Първото издание, които все още се съдържат в текста. Безкрайно съм признателен на хората, които няма да се успокоят, докато не поправя всички грешки. Надявам се, че и бъдещите читатели ще бъдат толкова критични и ще помогнат за достигането на тази висока цел, както и с конструктивна критика за цялостното подобряване на учебника. Продължавам да се надявам, че при внасяне на поправките не допускам много нови грешки и така, рано или късно, текстът ще бъде изчистен окончателно.

В резултат на поредното форматиране на текста, разполагането му на страници е различно от това на предишните издания. С това искам да обърна внимание на читателите, които по различни поводи биха цитирали книгата със споменаване на страници, че се налага да указват и поредният номер на изданието.

Всички забележки ще бъдат приети с благодарност. За да изпратите препоръките си или за директен контакт може да използвате адреса: `manev@fmi.uni-sofia.bg`.

*Автора*

София, Декември 2005г.

## Съдържание

## Въведение

Понятието *дискретен* произлиза от латинското *discernere* (разделям) и в повечето езици, в които е пренесено, е натоварено с двоен смисъл. В българския, руския и френския език се използва една и съща дума за двете значения (фр. *discret*). В английския език се използват еднакво звучащите, но с различен правопис думи *discreet* и *discrete*. Основното речниково значение е свързано със способността на човек ясно да разграничава нещата, за които може да се говори открито, от тези за които е по-добре да се премълчи, или пък действията, които могат да се извършват публично, от тези, които е по-добре да останат за по-тесен, по-интимен кръг. Второто значение се използва предимно в математиката за характеризиране на обекти, които са съставени от отделни, ясно различими части. В този смисъл понятието дискретен е противоположно на *непрекъснат*.

Дискретни математически теории са тези направления на математиката, в които е невъзможно или няма смисъл въвеждането на непрекъснатост. Тези теории са лишени от възможността да използват мощния апарат, свързан с непрекъснатостта – сходимост, производни, интеграли и т.н. В замяна на това, дискретните математически теории притежават изключително важното свойство *ефективност*. Доказателствата на твърдения за съществуване в тези теории почти винаги съдържат точни и ясни процедури, позволяващи конструирането на съществуващия обект от обектите, предпоставени в условието на съответната теорема. Такива процедури наричаме *алгоритми*. Затова за дискретните математически теории се казва, че са *конструктивни* или *алгоритмични*.

Тук трябва да отбележим, че в дискретната математика алгоритмите не са просто следствие от теоремите. Често алгоритъмът е неразделна част от доказателството на теоремата, нейна същност. Освен това, дискретната математика се занимава с формализацията на неформалното понятие алгоритъм и тези формализации са едни от основните обекти за изследване на дискретната математика. Тя изучава *алгоритмичните свойства* на дискретните математически обекти, за да отговори на въп-

росите за *съществуване* на алгоритми за определени класове задачи, за *коректността* на алгоритмите, за тяхната *сложност* и т.н.

Дискретната математика е толкова стара, колкото и математиката като цяло. Дискретна математическа теория е теорията на числата, занимаваща се с естествените числа – с представянето им в различни бройни системи, с извършването на аритметични операции с естествени числа, с въпросите за делимост и простота на естествени числа и т.н. Един от най-старите алгоритми, свързван с името на Евклид, е алгоритъмът за намиране на най-голям общ делител на две естествени числа.

Бурното развитие след XVII век на непрекъснатата математика, предизвикано от нуждите на техническата революция и обусловено от резултатите на Нютон и Лайбниц, оставя на по-заден план дискретната математика. Обществото не поставя реални задачи пред нея и затова тя се развива сравнително бавно, движена само от присъщата на всяка математическа теория вътрешна необходимост.

В началото на нашия век ситуацията се изменя съществено. Аналитичните методи на непрекъснатата математика не винаги успяват да се справят с неимоверно усложнилите се практически задачи на механиката и математическата физика. Налага се да се разглеждат дискретни приближения на тези задачи, изследванията на които стават с алгоритми, изискващи огромен обем пресмятания. Ръчното изпълнение на такива алгоритми е изключено. Освен това, лавинообразно нараства обемът на натрупаната научна и техническа информация, обработката на която с наличните технически средства става практически невъзможна. Човечеството се приближава към изключително важна крачка в своето развитие – създаването на универсалния компютър.

Изследванията, които се правят в тази насока, недвусмислено показват, че математическа основа на изграждането и експлоатацията на изчислителни машини ще бъде дискретната математика. Така интересът към дискретните математически теории значително се повишава. Между тях и зараждащата се компютърна наука се установяват дълбоки двустранни връзки. От една страна, построяването на сложни изчислителни устройства поставя задачи, решаването на които става предимно със средствата на дискретната математика. От друга страна, компютърните методи стават неразделна част от методологията на дискретната математика, наравно с методите на класическата математическа логика. Самата математическа логика също получава сериозен импулс за развитие, следствие на усилията да разреши проблемите от логическо естество, поставени от бурното развитие на компютърната наука и свързаните с нея дискретни математически теории. Затова границите между



компютърната наука, дискретната математика и съответните дялове на математическата логика днес са трудно установими.

Дискретната математика е доста голяма, и не съвсем ясно очертана, съвкупност от математически теории. Например, редица дискретни проблеми са намерили естественото си място в алгебрата и се развиват в рамките на този дял на математиката. Обстойното изложение на всяка от тези теории трябва да бъде обект на отделна книга (или няколко книги). Затова целта на настоящото издание е да запознае читателя с най-важните понятия и резултати, необходими за навлизането в областта и изграждане на навици за самостоятелно изучаване на многобройните ѝ специфични направления. Специално внимание се отделя на техниката на доказателство, базирана на конструиране на обектите и показване наличието на исканите свойства. Настоятелно са търсени и подчертавани аналогиите и взаимните връзки между отделните понятия.

Книгата е предназначена да обезпечи един или няколко базови курса по Дискретна математика за университетските специалности Информатика (Компютърни науки, Софтуерни технологии, Информационни системи), Приложна информатика, Математика, Математика и информатика и т.н. Тя може да бъде използвана и като помагало при четене на курсове по отделни направления, както и за самостоятелна подготовка на специалисти от други области, интересуващи се от методите на дискретната математика.

Изложението съдържа дефиниции (или обяснения) на почти всички използвани термини, като се предполагат минимален брой предварителни понятия, изучавани в средното училище. В началото например, за илюстрация на някои дефиниции, се използват реални числа, които не са обект на изложението. Предполага се още, че читателят, без да е изучавал специално математическа логика, може да си служи с нея при провеждане на математически разсъждения. За да му помогнем да се справи по-лесно, ще си позволим да резюмираме накратко някои от използваните техники на доказателство.

Всяка математическа теория е съвкупност от верни твърдения за добре дефинирани понятия – абстракции на реално съществуващи обекти и явления или пък умозрителни техни аналогии и обобщения. Изграждането на теорията започва с избор на *първични понятия* и *основни твърдения* (*аксиоми*) за тях. Първичните понятия не се дефинират (дефинициите са невъзможни за тях, тъй като няма дефинирани други понятия, на които да се опрат). Смисълът и същността на първичните понятия, както и техните свойства се изясняват от основните твърдения – аксиомите, верността на които не се подлага на съмнение. Всяко следващо

понятие на теорията се дефинира само с помощта на първичните и/или вече дефинирани понятия. Верността на всяко ново твърдение трябва да бъде доказана само с помощта на аксиомите и/или вече доказани твърдения.

С помощта на правилата на математическата логика извеждаме верността на съставни твърдения от верността на съставящите ги. Например, ако $p$ и $q$ са твърдения, тогава твърдението „$p$ и $q$“ е вярно, когато и двете съставящи го твърдения са верни. Твърдението „$p$ или $q$“ е вярно, когато поне едно от двете съставящи го твърдения е вярно. С „$\neg p$“ (четем „не е вярно, че $p$“) означаваме *обратното твърдение* на $p$, което е вярно само ако $p$ не е вярно. Твърдението „$p \Rightarrow q$“ (четем „ако $p$, то $q$“ или „от $p$ следва $q$“) не е вярно, само ако $p$ е вярно, а $q$ – не е. Твърдението „$p \Leftrightarrow q$“ е по-кратък запис на „$p \Rightarrow q$ и $q \Rightarrow p$“ (четем „$p$ е вярно тогава и само тогава, когато $q$ е вярно“ и за по кратко пишем „$p$ т.с.т.к. $q$“).

С използването на посочените по-горе правила, се извеждат типични техники на доказателство, които използваме в текста. Например, техниката известна като *доказателство с допускане на противното*: Нека $p$ е вярно твърдение, а $q$ – твърдение, верността на което искаме да установим. Нека допуснем, че $q$ не е вярно и с логически разсъждения да достигнем до неверността на $p$, т.е. установили сме верността на твърдението $\neg q \Rightarrow \neg p$. Тъй като $\neg p$ не е вярно, не е възможно $\neg q$ да е вярно. Следователно допускането е неправилно и $q$ е вярно твърдение.

В общия случай твърдението $p(x)$ може да зависи от един или няколко параметъра, които условно сме означили с $x$. Твърдения от вида „за всяка стойност на $x$ е вярно $p(x)$“ (за по-кратко пишем $\forall x(p(x))$, $\forall x, p(x)$ или $p(x), \forall x$) доказваме, като проверим верността на твърдението $p(a)$, за всяка възможна стойност $a$ на параметрите $x$. Твърдения от вида „съществува стойност на $x$, за което е вярно $p(x)$“ (за по-кратко пишем $\exists x(p(x))$ или $\exists x, p(x)$) доказваме, като посочим поне една стойност $a$ на параметрите $x$, за която $p(a)$ е вярно. Обратното твърдение на „за всяко $x$ е вярно $p(x)$“ е „не за всяко $x$ е вярно $p(x)$“ или „съществува $x$, за което не е вярно $p(x)$“. Обратното твърдение на „съществува $x$, за което е вярно $p(x)$“ е „не съществува $x$, за което е вярно $p(x)$“ или „за всяко $x$ не е вярно $p(x)$“.

Елементарна техника на доказателство е *дедукцията*, при която от верността на твърдението $\forall x(p(x))$ правим извод за верността на твърдението $p(a)$, където $a$ е произволна стойност на параметрите $x$. В текста са разгледани и други често използвани техники за доказателство на верността на твърдения.



Материалът в тази книга се основава на лекциите по Дискретна математика, четени от автора във Факултетите по математика и информатика на Софийския Университет и Велико Търновския Университет. Той бе допълнен с избрани раздели на алгебрата, комбинаториката и теорията на вероятностите, за да покрие и курса по Дискретни математически структури, четен от автора в Департамента по компютърни науки на Новия Български Университет, София. Добавени бяха и някои раздели за допълнително четене.

Материалът в книгата е разделен в седем глави. Глава първа въвежда читателя в основните понятия, използвани не само в дискретната математика, но и в почти всички математически теории – множества, релации, функции, множества с операции (алгебри). Основно внимание е отделено на дискретните (крайни или изброими безкрайни) варианти на тези понятия.

Глава втора е посветена на основите на комбинаториката. Посочени са най-важните начини за намиране броя на различни комбинаторни конфигурации. Разглеждат се някои по-сложни комбинаторни конфигурации – независими фамилии, матроиди, блок-дизайни и крайни геометрии. Въведено е понятието дискретно вероятностно пространство, имащо пряка връзка с комбинаториката.

На твърде общия и широко използван клас от комбинаторни конфигурации, наричани графи, е посветена глава трета. Специално се спираме на понятието дърво, заради ролята му в компютърната наука. Представени са важни и характерни алгоритми в графи.

Глава четвърта е въведение в теорията на формалните езици и абстрактните математически машини, използвани за разпознаването на формални езици.

Дискретните и по-специално двоичните (булевите) функции са предмет на глава пета. Разгледани са въпросите за представяне на дискретни функции с формули и схеми от функционални елементи, за пълнотата на множество от булеви функции и намирането на минимални формули за тях.

Глава шеста е въведение в актуалните теории за кодиране на информацията – компресията, шумозащитното кодиране и криптографията.

В последната, седма, глава се разглеждат неформалните понятия масова задача и алгоритъм. На базата на предхождащия материал са разгледани възможностите за формализация на тези понятия и за изследване с математически средства на важни техни свойства.

Обемът на книгата надхвърля рамките на двусеместриален университетски курс (2 часа седмично). Задачите, дадени в края на всяка глава

пък не са достатъчни, за да покрият упражненията на двусеместриален курс. Тяхната цел е по-скоро да очертаят основните приложения на разгледанията теоретичен материал или да въведат някои понятия, които не са основни за изложението.

Авторът си дава ясна сметка за това, че не е възможно да посочи в пълния обем всички издания, които са оказали влияние върху възгледите му за дискретната математика и начина на поднасяне на материала. Затова са цитирани само заглавията, имащи пряко влияние върху изграждането на текста. Ако съответният източник е бил разпространяван в България в превод, при цитирането е предпочетено преводното издание.

Приятно задължение на автора е да изкаже своята дълбока благодарност: на Йордан Денев, който го насочи към преподавателска и изследователска работа в областта на дискретната математика; на Йордан Денев, Радослав Павлов и Янош Деметрович – за възможността дълго време да използва в преподавателската си работа тяхната книга „Дискретна математика“ (изд. Наука и изкуство, София, 1984), от която настоящото издание е силно повлияно; на Васил Василев – за настойчивостта и осигурената финансова и технологична подкрепа, без които тази книга не би била реалност; на своето семейство, което не само стоически понесе работата върху книгата, но и помогна много за подготовката на текста и илюстрациите.

Авторът с удоволствие и благодарност ще приеме всякакви мнения и критични бележки, целящи подобряването на изложението при бъдещи издания на книгата.

*София, февруари 1996г.*

# Глава 1

# Основни понятия

## 1.1 Множества

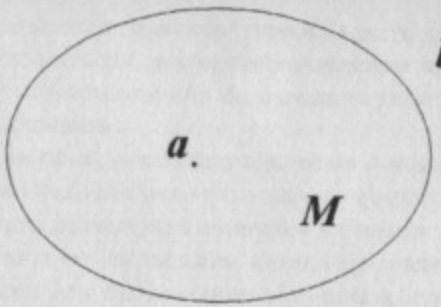
Понятието *множество* е основополагащо за дискретната математика, както за повечето математически теории, и за него не даваме дефиниция. Ще обясним същността му и (без да се впускаме във формализация) ще се спрем на по-важните аксиоми на Теорията на множествата. За целта ще наречем *протоелементи* всички реални или въображаеми обекти, с помощта на които ще изграждаме множества, като никое множество не е протоелемент.

Казваме, че е зададено множеството $M$, ако са определени обектите, от които то е съставено и които наричаме негови *елементи*. Допустими обекти са протоелементите и самите множества. За да се избягнат някои парадокси, ще използваме следния **конструктивен** (постъпков) подход: при образуване на ново множество, за негови елементи могат да се посочват или протоелементи, или вече образувани на някои предходни стъпки множества. Например, в началото може да се образува произволно множество $M$, съставено само от протоелементи. Ако множеството $M$ е вече конструирано, то може да бъде включено като елемент на произволно друго множество. Така не се допуска възможността множество да бъде елемент на самото себе си, а това изключва споменатите вече парадокси.

За благозвучие вместо множество от множества, понякога ще казваме *фамилия* от множества, като считаме понятието фамилия синоним на множество.

Удобен начин за визуализация на множествата са *диаграмите на Вен*, чрез които всяко множество се представя като частта от равнината, заградена от някаква несамопресичаща се и затворена крива (вж. Фиг. 1.1).

За означаване на множествата ще използваме големите букви на ла-




Фигура 1.1: Множество.

тинската азбука, а за елементите – малките латински букви. Фактът, че $a$ е елемент на множеството $M$ означаваме с $a \in M$ и четем „$a$ принадлежи на $M$“, или пък $M \ni a$ и четем „$M$ съдържа $a$“. Непринадлежността на елемента $b$ към множеството $M$ означаваме с $b \notin M$ и четем „$b$ не принадлежи на $M$“, или пък $M \not\ni b$ и четем „$M$ не съдържа $b$“. Фактът, че множествата $M$ и $M'$ са съставени от едни и същи елементи отбелязваме с $M = M'$ и казваме, че $M$ и $M'$ съвпадат. Несъвпадението на $M$ и $M'$ пък отбелязваме с $M \neq M'$.

Първата аксиома на Теорията на множествата отразява факта, че всяко множество се определя напълно и еднозначно от принадлежността на някакви елементи към него. Формално можем да я запишем така:

***Аксиома за обема.*** $\forall a(a \in A \Leftrightarrow a \in B) \Rightarrow A = B$.

Аксиомата ни позволява да задаваме множества, като изписваме всичките им елементи, разделени със запетаи, между знаците { и }. Например, за множеството $J_2$ с елементи 0 и 1 пишем $J_2 = \{0, 1\}$.

Естествено, при конструирането на множество можем да не включим в него нито един елемент. Получаваме множество, което наричаме *празно* и означаваме с $\{\}$ или $\emptyset$. Съгласно по-горе изложения принцип за конструиране, можем да построим множеството $M_\emptyset = \{\emptyset\}$. Забележете, че $M_\emptyset \neq \emptyset$, тъй като $M_\emptyset$ има един елемент, а $\emptyset$ няма елементи. Продължавайки конструирането в същия дух, можем да получим важна последователност от множества – $\emptyset$, $M_\emptyset$, $M_{\emptyset,\emptyset} = \{\emptyset, M_\emptyset\}$, $M_{\emptyset,\emptyset,\emptyset} = \{\emptyset, M_\emptyset, M_{\emptyset,\emptyset}\}$ и т.н. без използване на протоелементи.

От аксиомата за обема следва, например, че $\{a, a, b\} = \{a, b\}$, т.е. в множествата **няма повтаряне** на елементи и всъщност всеки елемент участва еднократно в множеството, както и че $\{a, b\} = \{b, a\}$, т.е. в множествата **няма наредба** на елементите и редът, по който са изписани те, е без значение. Освен това, аксиомата ни дава начин за доказателство на често срещаните в математиката твърдения за съвпадение на две множества. Тези доказателства ще извършваме по следната схема:



ТЕОРЕМА. „Дадени“ са множествата $A$ и $B$. Твърдим, че $A = B$.

ДОКАЗАТЕЛСТВО:

1) За всеки елемент $a \in A$ показваме, че $a \in B$.

2) За всеки елемент $b \in B$ показваме, че $b \in A$.

3) Съгласно аксиомата за обема, от (1) и (2) следва $A = B$. □

Втората аксиома на теорията на множествата ни дава възможност от елементите на зададено множество $M$ да образуваме ново множество $M'$ с помощта на свойството $\pi(x)$ такова, че $\forall x \in M$ можем да проверим дали е в сила това свойство. Ще означаваме с $\{x | x \in M, \pi(x)\}$ съвкупността от всички елементи $x \in M$, за които $\pi(x)$ е в сила. Всъщност получаваме по една аксиома за всеки конкретен избор на $M$ и $\pi(x)$, която можем да формулираме така:

***Аксиома за отделянето.*** *$M' = \{x | x \in M, \pi(x)\}$ е множество.*

Множеството $M'$ получено от $M$ чрез аксиомата за отделянето наричаме *подмножество* на $M$ и означаваме факта, че $M'$ е подмножество на $M$ с $M' \subseteq M$ или $M \supseteq M'$. Казваме също, че $M$ е *надмножество* на $M'$. Интересни частни случаи са свойствата с постоянни вярностни значения $t(x) = true$ (истина) и $f(x) = false$ (неистина), $\forall x \in M$. От първото получаваме, че $\forall M(M \subseteq M)$, а от второто, че $\forall M(\emptyset \subseteq M)$. Ако $M' \subseteq M$ и $M' \neq M$, казваме, че $M'$ е *свщинско подмножество* на $M$ и означаваме този факт с $M' \subset M$ или $M \supset M'$. Нека $\sigma(X)$ е свойство, верността на което е проверима $\forall X \subseteq M$. Казваме, че множеството $M_1 \subseteq M$ е *минимално (по включване)* със свойството $\sigma$, ако $\sigma(M_1) = true$ и $\forall M_2 \subset M_1, \sigma(M_2) = false$. Казваме, че множеството $M_1 \subseteq M$ е *максимално (по включване)* със свойството $\sigma$, ако $\sigma(M_1) = true$ и $\forall M_2 \supset M_1, M \supseteq M_2, \sigma(M_2) = false$.

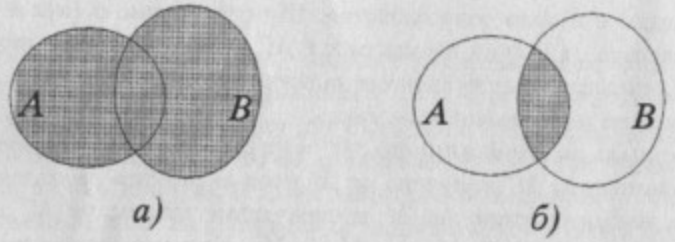
Третата аксиома, която ще използваме, е следната:

***Аксиома за степента.*** *Съвкупността от всички подмножества на множеството $M$ е множество.*

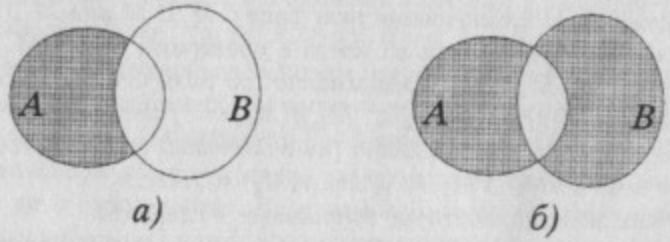
Действително, след като множеството $M$ е конструирано, всяко негово подмножество също може да бъде конструирано и в теоретичен план можем да считаме построено множеството $\{M' | M' \subseteq M\}$ от всички подмножества на $M$. Означаваме го с $2^M$ и го наричаме *степен* или *булеан* на $M$. Например, булеанът на множеството $J_2 = \{0, 1\}$ е $2^{J_2} = \{\emptyset, \{0\}, \{1\}, \{0, 1\}\}$. Не е възможно множеството $M$ и булеанът му да се представят с една и съща диаграма на Вен.

Понякога е полезно да разглеждаме някакво множество $U$ като надмножество на всички интересуващи ни множества. В такъв случай наричаме множеството $U$ *универсално*.

Нека $U$ е множество, което сме избрали за универсално. Конструктивният принцип ни позволява по зададени подмножества $A$ и $B$ на $U$, да построим ново подмножество на $U$ – $A \cup B = \{x | x \in A$ или $x \in B\}$, състоящо се от елементите на $A$ и $B$. Действително, всеки елемент на множеството $A$ или $B$ е протоелемент или вече конструирано множество, затова можем да го използуваме при конструкцията на новото множество $A \cup B$, което наричаме *обединение* на $A$ и $B$. С помощта на диаграма на Вен операцията обединение е представена на Фиг. 1.2.а.




Фигура 1.2: Обединение (а) и сечение (б) на множества.



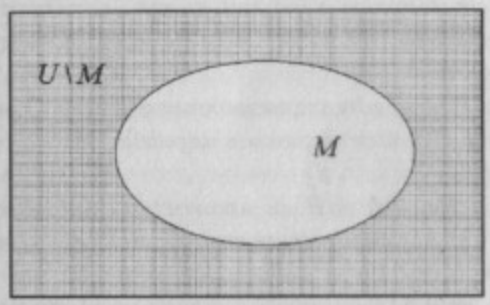

Фигура 1.3: Разлика (а) и симетрична разлика (б) на множества.

Аналогично можем да дефинираме и следната операция между множествата $A$ и $B$: $A \cap B = \{x | x \in A$ и $x \in B\}$ Резултатът от операцията е множество, което наричаме *сечение* на множествата $A$ и $B$. С помощта на диаграмите на Вен операцията сечение се изобразява така, както е показано на Фиг. 1.2.б.



Други подобни операции с множества са *разликата* на $A$ и $B$: $A \setminus B = \{x | x \in A$ и $x \notin B\}$ и *симетричната разлика* на $A$ и $B$: $A \triangle B = \{x | (x \in A$ и $x \notin B)$ или $(x \notin A$ и $x \in B)\}$. Те са представени с диаграми на Вен на Фиг. 1.3.

За всяко подмножество $M$ на $U$ означаваме с $\overline{M}^U$ (или просто с $\overline{M}$, когато множеството $U$ се подразбира) разликата $U \setminus M$ и наричаме полученото множество *допълнение* (на $M$ до $U$). На Фиг. 1.4 е показано множество $M$ и допълнението му до универсалното множество $U$ (защриховано на фигурата).




Фигура 1.4: Допълнение на множество.

Оставяме на читателя да докаже (с директно прилагане на дефинициите) или да покаже с диаграми на Вен верността на следните свойства на основните операции с множества, $\forall A, B, C \in U$:

1. Идемпотентност
а) $A \cup A = A$; б) $A \cap A = A$.
2. Комутативност
а) $A \cup B = B \cup A$; б) $A \cap B = B \cap A$; в) $A \triangle B = B \triangle A$.
3. Асоциативност
а) $(A \cup B) \cup C = A \cup (B \cup C)$;
б) $(A \cap B) \cap C = A \cap (B \cap C)$;
в) $(A \triangle B) \triangle C = A \triangle (B \triangle C)$.
4. Дистрибутивност
а) $(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$;
б) $(A \cap B) \cup C = (A \cup C) \cap (B \cup C)$.
5. Свойства на празното и универсалното множество
а) $A \cup \emptyset = A$; б) $A \cap \emptyset = \emptyset$; в) $A \cup U = U$; г) $A \cap U = A$.
6. Свойства на допълнението
а) $A \cup \overline{A} = U$; б) $A \cap \overline{A} = \emptyset$; в) $\overline{\overline{A}} = A$.

7. Закони на Де Морган
а) $\overline{A \cup B} = \overline{A} \cap \overline{B}$; б) $\overline{A \cap B} = \overline{A} \cup \overline{B}$.
8. Други свойства
а) $A \subseteq A \cup B$; б) $(A \cap B) \subseteq A$ и $(A \cap B) \subseteq B$;
в) $A \cup (A \cap B) = A$; г) $A \cap (A \cup B) = A$;
д) $(A \cap B) \cup (A \cap \overline{B}) = A$; е) $(A \cup B) \cap (A \cup \overline{B}) = A$.

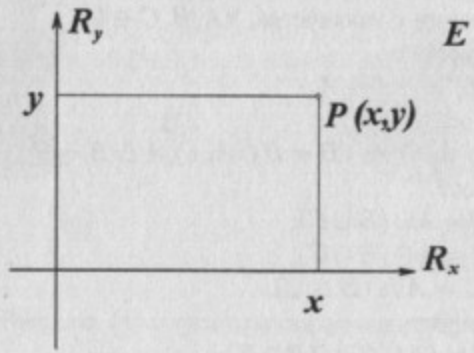
Асоциативността на операциите обединение и сечение ни позволява да избегнем употребата на скоби при многократното им прилагане. Например, вместо $(A \cup B) \cup C$ или $A \cup (B \cup C)$ пишем $A \cup B \cup C$.

Друг способ да получим ново множество, елементите на което не са измежду елементите на множествата от които е получено, ни дават следните дефиниции:

***Дефиниция.*** Нека $a$ и $b$ са произволни елементи. Означаваме с $(a, b)$ множеството $\{a, \{a, b\}\}$ и го наричаме *наредена двойка* от елементите $a$ и $b$.

***Дефиниция.*** Нека $A$ и $B$ са множества. Множеството $A \times B = \{(a, b) | a \in A, b \in B\}$ наричаме *декартово произведение* на множествата $A$ и $B$.

Дефинициите са в съответствие с възприетия конструктивен принцип, защото след като $A$ и $B$ са вече построени можем да образуваме наредените двойки $(a, b), \forall a \in A, \forall b \in B$ и да конструираме множество от всички такива двойки.




Фигура 1.5: Евклидовата равнина е декартово произведение.

Класически пример за декартово произведение (дължим го на Рьоне Декарт, откъдето и името на операцията) е множеството от точките на евклидовата равнина $E$, в която е въведена правоъгълна координатна



система посредством две перпендикулярни реални прави $R_x$ и $R_y$ (вж. Фиг. 1.5). Ако означим с $x$ ортогоналната проекция на точката $P$ върху правата $R_x$, а с $y$ – ортогоналната ѝ проекция върху правата $R_y$, то точката $P$ еднозначно се определя от наредената двойка $(x, y)$. Така за цялата равнина $E$ получаваме, че $E = R_x \times R_y$. Декартовото произведение $M \times M = M^2$ наричаме *декартов квадрат* на $M$. За $J_2 = \{0, 1\}$ ще имаме $J_2 \times J_2 = J_2^2 = \{(0,0), (0,1), (1,0), (1,1)\}$. Всъщност, ако приемем, че $R_x = R_y = R$, евклидовата равнина съвпада с декартовия квадрат $R^2$ на множеството $R$ на реалните числа.

Операцията декартово произведение не е асоциативна по принцип. Например, наредената двойка $((a, b), c)$ се различава съществено от наредената двойка $(a, (b, c))$. За редица приложения, обаче, е интересно да се разглежда асоциативен вариант на операцията, при която $(A \times B) \times C = A \times (B \times C) = A \times B \times C$, като в този случай елементите на тройното декартово произведение означаваме с $(a, b, c)$ и ги наричаме *наредени тройки*. Аналогично дефинираме наредени четворки, петорки и т.н.

И при операцията декартово произведение, както при булеана на едно множество, не е възможно да представим резултата с диаграма на Вен, тъй като елементите му имат съвършенно друга природа в сравнение с елементите на множествата операнди.

Със следващата аксиома ще узаконим един важен способ за построяване на множества – *индуктивното дефиниране* на множество $M$. За целта са необходими непразно множество $M_0$ и множество от операции $\mathcal{F}$. Да разгледаме първо основните моменти на индуктивната дефиниция на множеството $M$, състояща се от поредица конструктивни стъпки:

а) БАЗА. На тази стъпка включваме в $M$ елементите на $M_0$, които ще наричаме базови елементи на $M$.

б) ИНДУКЦИОННО ПРЕДПОЛОЖЕНИЕ. Нека в $M$ вече са включени някакви елементи (в началото поне тези от базовото множество $M_0$).

в) ИНДУКЦИОННА СТЪПКА. Включваме в $M$ всеки елемент (протоелемент или множество), който е резултат на някоя от операциите от $\mathcal{F}$, изпълнена върху елементи, вече включени в $M$.

г) ЗАКЛЮЧЕНИЕ. В $M$ няма други елементи освен базовите и включените чрез индукция.

Означаваме получената съвкупност с $< M_0, \mathcal{F} >$. Четвъртата аксиома е следната:

***Аксиома за индукцията.*** *$M =< M_0, \mathcal{F} >$ е множество.*

За пример да разгледаме индуктивно дефинираната фамилия $N$, с базов елемент празното множество и операция, която построява нов еле-

мент от обединението на всички елементи, включени вече в $N$. Очевидно $N = \{\emptyset, M_\emptyset, M_{\emptyset,\emptyset}, M_{\emptyset,\emptyset,\emptyset} \ldots\}$. Това е добре познатото множество на естествените числа, като по традиция обозначаваме елементите му с $0, 1, 2, \ldots$ и пишем $N = \{0, 1, 2, \ldots\}$. Друг начин да дефинираме индуктивно $N$ е като подмножество на множеството на реалните числа $R$:

а) $0 \in N$;

б) нека $i \in N$;

в) тогава $i + 1 \in N$;

г) няма други естествени числа.

Ще считаме, че за естествените $i$ и $j$ е в сила свойството $i \leq j$, ако при индуктивното построяване на $N$, $i$ е влязло в $N$ не по късно от $j$. С помощта на аксиомата за отделянето можем да дефинираме $\forall n \in N, n \neq 0$, множеството $I_n = \{x | x \in N, 1 \leq x \leq n\}$, което означаваме с $I_n = \{1, 2, \ldots, n\}$, и множеството $J_n = \{x | x \in N, 0 \leq x \leq n-1\}$, което означаваме с $J_n = \{0, 1, \ldots, n-1\}$.

От аксиомата за индукцията получаваме важен метод за доказване на твърдения за индуктивно дефинирани множества, наричан *доказателство по индукция*. Ето схемата на едно такова доказателство:

ТЕОРЕМА. За всеки елемент $x$ на индуктивно дефинираното множество $M$ е в сила $\pi(x)$.

ДОКАЗАТЕЛСТВО:

а) БАЗА. За всеки базов елемент $x_0$ от $M_0$ проверяваме верността на $\pi(x_0)$.

б) ИНДУКЦИОННО ПРЕДПОЛОЖЕНИЕ. Допускаме, че $\pi(x)$ е в сила за всеки елемент $x$, включен до определен момент в множеството $M$.

в) ИНДУКЦИОННА СТЪПКА. Показваме, че при направеното предположение, за всеки построен с помощта на операциите от $\mathcal{F}$ елемент $y$ на $M$, също е в сила $\pi(y)$.

г) ЗАКЛЮЧЕНИЕ. $\pi$ е в сила за всеки елемент на множеството $M$.

Наистина, съгласно заключението на всяка индуктивна дефиниция елементите на индуктивно дефинираното множество са или базови, или получени чрез индукционни стъпки от вече включени в множеството елементи. Верността на твърдението за базовите елементи проверяваме в базата на доказателството, а за включените чрез индукционни стъпки елементи верността на твърдението е следствие от индукционното предположение и индукционната стъпка. Отначало операциите се прилагат само върху базови елементи, за които твърдението е в сила и затова то е в сила и за резултата. След това като аргументи се използват както базови, така и получени вече като резултат на операциите елементи, но



и за едните, и за другите е в сила твърдението. Следователно то винаги е в сила и за новополучения резултат.

За да си представим по-ясно механизма на доказателствата по индукция, ще използваме следната индуктивна игра: нека произволно непразно множество плочки за домино са поставени върху най-малката си стена в редица, на такова разстояние една от друга, че падането на всяка плочка предизвиква падането на съседната ѝ (в посоката на падането) плочка. Сега, ако съборим първата в редицата плочка по посока на следващата я, то всяка плочка в редицата, рано или късно, ще падне.

Най-често срещани са доказателства с индукция по елементите на множеството на естествените числа, но в дискретната математика не са рядкост и доказателствата с индукция по елементите на други индуктивно дефинирани множества.

Често ще използваме елементите на едно множество $I$ за обозначаване на различните елементи на някое друго множество $A$. Първото множество наричаме *индексно множество*, а второто – *индексирано множество*. В такъв случай всеки елемент на $A$ представяме с буква, към която долу вдясно е приписан някой елемент на индексното множество, наричан *индекс* – $A = \{a_i | i \in I\}$. При това различните елементи на $A$ трябва да имат различни индекси и всеки елемент на $I$ трябва да е индекс на един елемент от $A$. За пример, с помощта на множествата $N$ и $I_n$, сме индексирали $A = \{a_0, a_1, a_2, \ldots\}$ и $B = \{b_1, b_2, \ldots, b_n\}$, съответно. На формалното уточняване на това обозначение ще се спрем малко по-късно.

Заради асоциативния характер на операциите обединение, сечение и декартово произведение не е трудно да дефинираме многократни обединение, сечение и декартово произведение, които (аксиоматично) считаме за множества. Нека $I$ е индексно множество на фамилията $\{A_i | i \in I\}$. Означаваме с $\bigcup_{i \in I} A_i = \{x | \exists j \in I (x \in A_j)\}$ *многократното обединение*, а с $\bigcap_{i \in I} A_i = \{x | \forall j \in I (x \in A_j)\}$ *многократното сечение* на всички множества от $\{A_i | i \in I\}$.

За операцията *многократно декартово произведение* ще разгледаме само два частни, но достатъчни за целите ни, случая

а) $I = I_n$:

$\times_{i \in I_n} A_i = A_1 \times A_2 \times \cdots \times A_n = \{(a_1, a_2, \ldots, a_n) | \forall i \in I_n (a_i \in A_i)\}$.

В този случай елементите на построеното декартово произведение наричаме *наредени n-торки*. Когато $A_1 = A_2 = \cdots = A_n = A$ вместо $\times A_i$ пишем $A^n$, полученото декартово произведение наричаме *декартова n-та степен* на $A$, а елементите му *n-мерни вектори* над $A$ или просто *вектори*.

б) $I = N$:

$\times_{i \in N} A_i = A_0 \times A_1 \times A_2 \times \cdots = \{(a_0, a_1, a_2, \ldots) | \forall i \in N (a_i \in A_i)\}$

В този случай елементите на построеното декартово произведение наричаме *редици*.

Понякога вместо $i \in I_n$ пишем „$i = 1, 2, \ldots, n$“ или поставяме „$i = 1$“ под знака за многократна операция и „$n$“ над него. Аналогично вместо $i \in N$ пишем „$i = 0, 1, 2, \ldots$“ или поставяме „$i = 0$“ под знака за многократна операция и „$\infty$“ над него.

***Дефиниция.*** Нека $A$ е произволно множество. Казваме, че фамилията $\mathcal{R} = \{S_i | S_i \subseteq A, \forall i \in I\}$, където $I$ е подходящо индексно множество, е *разбиване* на $A$, ако са в сила:

1) $\forall i \in I (S_i \neq \emptyset)$;

2) $\forall i, j \in I (i \neq j \Rightarrow S_i \cap S_j = \emptyset)$;

3) $\bigcup_{i \in I} S_i = A$.

За пример да разгледаме множествата: $N_E = \{x | x \in N, x = 2i, i \in N\}$ на четните и $N_O = \{x | x \in N, x = 2i + 1, i \in N\}$ на нечетните естествени числа. Очевидно е, че фамилията $\mathcal{N} = \{N_E, N_O\}$ образува разбиване на множеството $N$ на естествените числа.

## 1.2 Релации

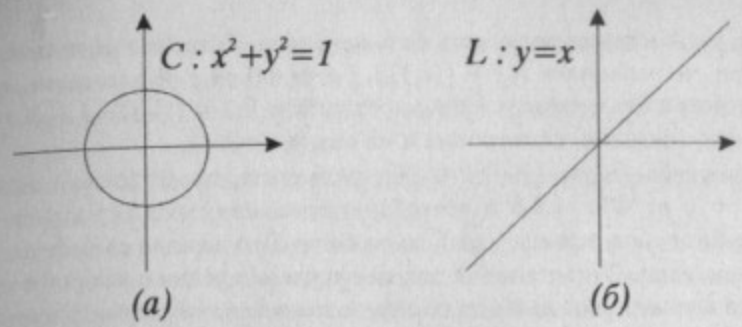
Понятието релация е основно за математиката, както в теоретичен, така и в приложен аспект. От теоретична гледна точка, то лежи в основата на много други математически понятия, а от практическа гледна точка е математически модел на най-разнообразни проблеми от реалния свят, за които се налага да се търси математическо или алгоритмично решение.

***Дефиниция.*** Нека $n \in N$ и $A_1, A_2, \ldots, A_n$ е фамилия от множества, индексирана с множеството $I_n$. Всяко $R \subseteq A_1 \times A_2 \times \cdots \times A_n$ наричаме *$n$-местна релация* над декартовото произведение $A_1 \times A_2 \times \cdots \times A_n$ или просто $n$-местна релация. Множествата $A_1, A_2, \ldots, A_n$ наричаме, съответно, първи *домен*, втори домен,..., $n$-ти домен на тази релация. В много често срещания случай $n = 2$ релацията наричаме *двуместна*, или просто *релация*.

На Фиг. 1.6.а и 1.6.б са показани две подмножества от точки на евклидовата равнина – окръжност $C$ с център $(0, 0)$ и радиус 1 и правата $L : y = x$, които в съответствие с дефиницията са релации над декартовия квадрат $R^2$ на множеството $R$ на реалните числа.

Фактът, че най-често се занимаваме с двуместни релации не е съществено ограничение, тъй като можем да си мислим, че двата домена $A$ и $B$ на релацията $R \subseteq A \times B$ също са декартови произведения






Фигура 1.6: Релации.

$A = A_1 \times A_2 \times \cdots \times A_i$ и $B = A_{i+1} \times A_{i+2} \times \cdots \times A_n$ и всъщност

$$R \subseteq A \times B = (A_1 \times A_2 \times \cdots \times A_i) \times (A_{i+1} \times A_{i+2} \times \cdots \times A_n) = A_1 \times A_2 \times \cdots \times A_n.$$

При двуместните релации често използваме обозначение различно от това в дефиницията: вместо $(a, b) \in R_\star$ пишем $aR_\star b$ или $a \star b$ и четем „$a$ е в релация $R_\star$ с $b$“. Тук $\star$ е някакъв специален знак за по-кратко обозначаване на релацията. Записът $a \star b$ наричаме *инфиксен*.

Типични примери за двуместни релации между числа са „по-малко от“, „по-малко от или равно на“, „не равно на“ и „равно на“, които считаме добре известни и ще използваме за примери.

Очевидно понятието релация е толкова общо, че трудно могат да се очакват някакви много интересни свойства. Затова ще се спрем по-подробно на *релациите над декартови квадрати*. Ще дадем следната:

***Дефиниция.*** Нека $R \subseteq A \times A$. Казваме, че релацията $R$ е:

а) *рефлексивна*, ако $\forall a \in A, (a, a) \in R$;

б) *симетрична*, ако $\forall a, b \in A, (a, b) \in R \Rightarrow (b, a) \in R$;

в) *транзитивна*, ако $\forall a, b, c \in A, (a, b) \in R, (b, c) \in R \Rightarrow (a, c) \in R$;

г) *антирефлексивна*, ако $\forall a \in A, (a, a) \notin R$;

д) *антисиметрична*, ако $\forall a, b \in A, a \neq b, (a, b) \in R \Rightarrow (b, a) \notin R$ (или $\forall a, b \in A, (a, b) \in R, (b, a) \in R \Rightarrow a = b$);

е) *силно антисиметрична*, ако $\forall a, b \in A, a \neq b$ точно едно от $(a, b) \in R$ или $(b, a) \in R$ е в сила.

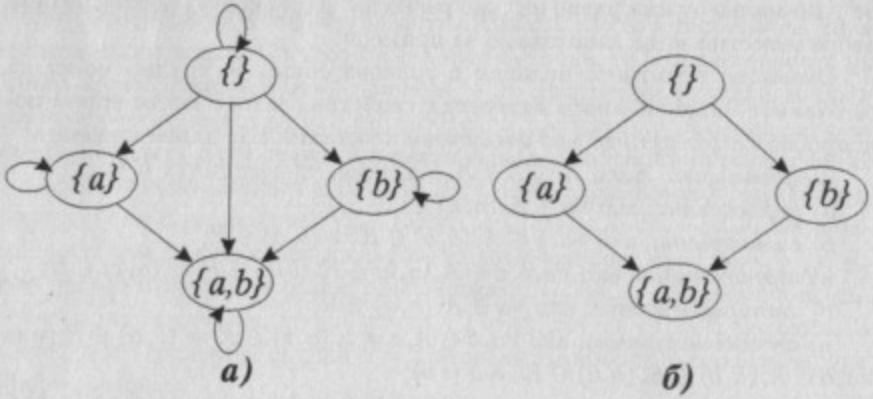
Релацията $R_< = \{(i, j) | i, j \in N, i < j\}$ не е рефлексивна, защото $i < i$ не е в сила за никое $i \in N$. Това означава също, че тя е антирефлексивна. Очевидно ако $i < j$, то не е вярно $j < i$ и следователно релацията е антисиметрична. Тъй като при $i \neq j$ точно едно от двете $i < j$ или $j < i$ е в сила, релацията е силно антисиметрична. От $i < j$ и $j < k$

следва $i < k$ и значи релацията е транзитивна. Оставяме на читателя да провери, че релацията $R_{\leq} = \{(i,j)|i,j \in N, i \leq j\}$ е рефлексивна, силно антисиметрична и транзитивна, а релацията $R_{\neq} = \{(i,j)|i,j \in N, i \neq j\}$ е антирефлексивна, симетрична и не е транзитивна.

Релацията $R_{=} = \{(i,i)|i \in N\}$, релацията, представена с правата $L : x = y$ от Фиг. 1.6.б и всяка друга релация, която съдържа само наредени двойки от вида $(x,x)$, са особени. Те очевидно са рефлексивни и транзитивни. Отсъствието в тях на наредени двойки с различен първи и втори елемент позволява да ги определим както като симетрични, така и като антисиметрични. Релацията представена с окръжността $C$ от Фиг. 1.6.а е симетрична, но не е рефлексивна и транзитивна.

Като още един пример да разгледаме релацията $R_{\subseteq A} = \{(S_i, S_j)|S_i, S_j \subseteq A, S_i \subseteq S_j\}$, дефинирана над декартовия квадрат $2^A \times 2^A$, за произволно множество $A$. Оставяме на читателя да покаже, че тя е рефлексивна, антисиметрична и транзитивна, но не е силно антисиметрична.

Релациите над декартови квадрати се представят удобно в равнината, като на всеки елемент $a$ на образуващото множество се съпоставя точка, а елементите $a$ и $b$ се свързват със стрелка, насочена от $a$ към $b$, тогава и само тогава, когато са в релация. Полученото представяне наричаме *диаграма на релацията*. На Фиг. 1.7.а е представена диаграмата на релацията $R_{\subseteq A}$ за множеството $A = \{a,b\}$.




Фигура 1.7: Диаграма на релация (а) и образуващата я (б).

***Дефиниция.*** Нека $R \subseteq A \times A$. Казваме, че $R' \subseteq A \times A, R \subseteq R'$ е:

а) *рефлексивно затваряне* на $R$, ако $R'$ е рефлексивна и минимална по включване с исканите свойства;



б) *симетрично затваряне* на $R$, ако $R'$ е симетрична и минимална по включване с исканите свойства;

в) *транзитивно затваряне* на $R$, ако $R'$ е транзитивна и минимална по включване с исканите свойства.

Релацията $R_<$ съвпада с транзитивното си затваряне, рефлексивното ѝ затваряне е $R_{\leq}$, а симетричното – $R_{\neq}$.

Така дефинираните операции ни позволяват да представяме една релация чрез по-проста *образуваща релация*, посредством рефлексивно, симетрично или транзитивно затваряне на образуващата, или чрез комбинации от тези три операции. На Фиг. 1.7.б е представена релация $R'$ такава, че релацията $R_{\subseteq\{a,b\}}$ от Фиг. 1.7.а е нейно рефлексивно и транзитивно затваряне.

***Дефиниция.*** Релацията $R \subseteq A \times A$ наричаме *релация на еквивалентност*, ако е рефлексивна, симетрична и транзитивна.

Релацията $R_{=} \subseteq N \times N$ и релацията от Фиг. 1.6.б са еквивалентности. По-нататък ще посочим и други релации на еквивалентност. Тук ще докажем, че всяка релация на еквивалентност над декартовия квадрат $A \times A$ по естествен начин задава разбиване на множеството A.

**Теорема 1.2.1** *Нека $R \subseteq A \times A$ е релация на еквивалентност, $a \in A$ и $[a] = \{b|b \in A, aRb\}$. Нека фамилията $\mathcal{R} = \{[a_i]|i \in I\}$ се състои от всички различни множества $[a_i]$, където $I$ е подходящо индексно множество. Тогава $\mathcal{R}$ е разбиване на $A$.*

***Доказателство.*** Очевидно е, че $[a_i] \subseteq A, \forall i \in I$. Остава да покажем, че са в сила трите изисквания от дефиницията за разбиване:

1) $\forall i \in I([a_i] \neq \emptyset)$, защото $R$ е симетрична, т.е. $a_i R a_i$ и следователно $a_i \in [a_i]$;

2) $[a_i] \neq [a_j] \Rightarrow [a_i] \cap [a_j] = \emptyset$. Действително, да допуснем противното, т.е. $[a_i] \neq [a_j]$, но $[a_i] \cap [a_j] \neq \emptyset$. Следователно $\exists b \in [a_i] \cap [a_j]$, т.е. $b \in [a_i]$ и $b \in [a_j]$. От дефиницията на $[a_i]$ и $[a_j]$ получаваме $a_i R b$ и $a_j R b$. От симетричността на релацията $R$ следва $b R a_j$, а от транзитивността ѝ – $a_i R a_j$. Нека $b' \in [a_j]$, т.е. $a_j R b'$. От $a_i R a_j$, $a_j R b'$ и транзитивността получаваме $a_i R b'$, следователно $b' \in [a_i]$. Последното е в сила за всеки елемент на $[a_j]$, следователно $[a_j] \subseteq [a_i]$. Аналогично (като разменим местата на $[a_i]$ и $[a_j]$) показваме, че $[a_i] \subseteq [a_j]$. Следователно $[a_i] = [a_j]$, което противоречи на нашето допускане. И така допускането е невярно, т.е. $[a_i] \neq [a_j] \Rightarrow [a_i] \cap [a_j] = \emptyset$;

3) Ще покажем, че $\bigcup_{i \in I} [a_i] = A$. Действително, за кой да е елемент $a$ на $A$ имаме $a \in [a] = [a_j]$, за някое $j \in I$. Следователно $a \in \bigcup_{i \in I} [a_i]$ и

следователно $A \subseteq \bigcup_{i \in I}[a_i]$. Но $[a_j] \subseteq A, \forall j \in I$, следователно $\bigcup_{i \in I}[a_i] \subseteq A$ и $\bigcup_{i \in I}[a_i] = A$. □

Определените в Теорема 1.2.1 различни множества $[a_i]$ наричаме *класове на еквивалентност* на $R$. Ако $a_i R a_j$, казваме, че $a_i$ е *еквивалентен* на $a_j$ или че $a_i$ и $a_j$ са *еквивалентни* (по отношение на $R$).

В разгледаните досега релации на еквивалентност всеки клас на еквивалентност се състои от един единствен елемент. По-нататък ще дефинираме релации на еквивалентност, в които класовете на еквивалентност могат да имат и повече елементи.

***Дефиниция.*** Релацията $R \subseteq A \times A$ наричаме *частична наредба*, ако е рефлексивна, транзитивна и антисиметрична.

От разгледаните вече релации частични наредби са $R_{\leq}$ и $R_{\subseteq A}$. Релацията $R_<$, например, не е частична наредба, защото не е рефлексивна.

Като пример нека въведем частична наредба в множеството $J_2^n = \{0,1\}^n$ на $n$-мерните *двоични вектори*. Дефинираме в $J_2^n$ релацията $R_{\preceq}$ по следния начин: $\alpha(a_1, a_2, \ldots, a_n) \preceq \beta(b_1, b_2, \ldots, b_n)$, ако $a_i \leq b_i, i = 1, 2, \ldots, n$. Очевидно е, че тази релация е частична наредба, защото $\leq$ е частична наредба в $\{0,1\}$.

***Дефиниция.*** Частичната наредба $R \subseteq A \times A$ наричаме *пълна* (или *линейна*) наредба, ако е силно антисиметрична.

От разгледаните вече релации пълна наредба е $R_{\leq}$, а $R_{\subseteq A}$ и $R_{\preceq}$ не са. При $A = \{a, b\}$, например, нито $\{a\} \subseteq \{b\}$, нито $\{b\} \subseteq \{a\}$. Същото е в сила и за двумерните двоични вектори $(0,1)$ и $(1,0)$.

***Дефиниция.*** Нека $R \subseteq A \times A$. Редица $a_{i_0}, a_{i_1}, \ldots, a_{i_l}, l \geq 1$, в която $a_{i_j} R a_{i_{j+1}}$ и $a_{i_j} \neq a_{i_{j+1}}, \forall j \in J_l$, наричаме *верига* на релацията $R$. Верига, при която $l \geq 2$ и $a_{i_l} R a_{i_0}$ наричаме *контур* на релацията $R$. Множеството $\{a_{i_0}, a_{i_1}, \ldots, a_{i_l}\} \subseteq A$ такова, че $\forall j \neq k$ нито $(a_{i_j}, a_{i_k}) \in R$, нито $(a_{i_k}, a_{i_j}) \in R$ наричаме *антиверига* на релацията $R$.

**Теорема 1.2.2** *Нека релацията $R \subseteq A \times A$ е рефлексивна и транзитивна. Необходимо и достатъчно условие $R$ да е частична наредба е да не съдържа контур.*

***Доказателство.*** 1) *Необходимост.* Нека $R$ е частична наредба. Следователно $R$ е антисиметрична. Да допуснем, че в $R$ съществува контур $a_{i_0}, a_{i_1}, \ldots, a_{i_l} = a_{i_0}$. От транзитивността на релацията и $a_{i_j} R a_{i_{j+1}}$, $\forall j \in J_l$ получаваме, че $a_{i_0} R a_{i_{l-1}}$. Но $a_{i_{l-1}} R a_{i_0} = a_{i_l}$, а това противоречи на антисиметричността на релацията. Следователно, допускането не е вярно и $R$ няма контури.

2) *Достатъчност.* Нека $R$ не съдържа контури. Да допуснем, че $R$ не е антисиметрична. Следователно $\exists a_i, a_j \in A, a_i \neq a_j, a_i R a_j, a_j R a_i$. Но тогава $a_i, a_j, a_i$) е контур, което противоречи на условието. Следователно допускането не е вярно и $R$ е антисиметрична. Но тогава $R$ е частична наредба, защото е рефлексивна и транзитивна. □



***Дефиниция.*** Релация без контури наричаме *преднаредба*.

***Дефиниция.*** Частичната наредба $R \subseteq A \times A$ наричаме *вложена* в пълната $R' \subseteq A \times A$, ако $R \subseteq R'$.

***Дефиниция.*** Нека $R \subseteq A \times A$. Елемента $a \in A$ наричаме *минимален* в $R$, ако не съществува $b \in A, b \neq a$ и такъв, че $bRa$. Елемента $a \in A$ наричаме *максимален* в $R$, ако не съществува $b \in A, b \neq a$ и такъв, че $aRb$.

## 1.3 Функции

***Дефиниция.*** Релацията $f \subseteq X \times Y$ наричаме *частична функция*, ако $\forall a \in X \; \exists$ не повече от едно $b \in Y$ такова, че $(a, b) \in f$. Множеството $X$ наричаме *дефиниционна област* на функцията, а множеството $Y$ – *кообласт* или *област на изменение*. В случай, че $\forall a \in X \; \exists$ точно едно $b \in Y$ такова, че $(a, b) \in f$, казваме, че функцията $f$ е *тотална*. Когато функцията е тотална и това се подразбира от контекста, тогава я наричаме просто *функция*.

По-често вместо $f \subseteq X \times Y$ пишем $f : X \to Y$ и казваме, че е зададена функцията $f$ от $X$ в $Y$ (или изобразяваща $X$ в $Y$). За да покажем значението на функцията върху произволен елемент $x \in X$, използваме означението $f(x)$, а $x$ наричаме *независима променлива*. Така, ако $(a, b) \in f$, пишем $f(a) = b$. Ако дефиниционната област $X$ е декартово произведение $X = X_1 \times X_2 \times \cdots \times X_n$, тогава използваме означението $f(x_1, x_2, \ldots, x_n)$, с независими променливи $x_1, x_2, \ldots, x_n$ и вместо $((a_1, a_2, \ldots, a_n), b) \in f$ пишем $f(a_1, a_2, \ldots, a_n) = b$.

***Дефиниция.*** Ако $f(x_1) \neq f(x_2), \forall x_1 \neq x_2 \in X$, тогава функцията $f$ наричаме *еднозначна* или *инекция*. В случай, че $\forall b \in Y \exists a \in X$, $f(a) = b$, казваме че $f$ е *сюрекция* на $X$ върху $Y$ или, че изобразява $X$ върху $Y$.

Релацията $C$ от Фиг. 1.6.а не е функция, тъй като $\forall a$ в интервала $(-1, 1)$ съществува повече от едно $b$, такова че $C(a) = b$. Релацията $L$ от Фиг. 1.6.б е функция.

Нека $X = J_2^2 = \{0,1\}^2 = \{(0,0), (0,1), (1,0), (1,1)\}$. Функция $f : J_2^2 \to J_2$ можем да дефинираме като $\forall (a_1, a_2) \in J_2^2$ посочим $b \in J_2$ такова, че $f(a_1, a_2) = b$. Например $f(0,0) = 0, f(0,1) = 1, f(1,0) = 1, f(1,1) = 1$.

Като трети пример да разгледаме подпрограмата:

```
int sqr(int n)
{
   int m=0;
   while(n!=0)
   {n=n-(2*m+1); m=m+1;}
   return(m);
}
```

Тя получава като входен параметър цяло число $n$ и връща като резултат квадратния корен на $n$, ако $n$ е точен квадрат. В противен случай програмата никога не завършва своята работа и не връща никакво значение. Имаме типичен пример на частична функция $f : Z \to N$ (с $Z$ сме означили множеството на целите числа),

$$f(n) = \begin{cases} \sqrt{n} & \text{ако } n \text{ е точен квадрат} \\ \text{неопределено} & \text{в противен случай.} \end{cases}$$

Ако в горната програма добавим непосредствено преди оператора за цикъл *while* реда

```
if (n<0) return(-1);
```

а условието на цикъла променим на

```
while(n>0)
```

ще получим програмата

```
int sqr(int n)
{
   int m=0;
   if (n<0) return(-1);
   while(n>0)
   {n=n-(2*m+1); m=m+1;}
   return(m);
}
```

която задава тоталната функция $f' : Z \to N \cup \{-1\}$,

$$f'(n) = \begin{cases} \lfloor \sqrt{n} \rfloor & n \geq 0 \\ -1 & n < 0 \end{cases}$$



където с $\lfloor x \rfloor$ означаваме най-голямото цяло по-малко или равно на $x$ – *долна цяла част* или просто *цяла част* на $x$. Аналогично с $\lceil x \rceil$ означаваме най-малкото цяло по-голямо или равно на $x$ – *горна цяла част* на $x$.

Този пример до голяма степен показва ролята на частичните функции в дискретната математика. Те се получават по естествен начин от изчислителни (алгоритмични) процедури, които при определени входни данни никога не завършват работата си (зациклят). Нещо повече, в много случаи не е възможно да се определи предварително максималното по включване подмножество $X' \subseteq X$, в което функцията е тотална.

***Дефиниция.*** Нека $f : A \to B$, $A' \subset A$ и $f' : A' \to B$ са такива, че $f'(a) = f(a)$, $\forall a \in A'$. Тогава $f'$ наричаме *рестрикция* на функцията $f$ върху $A'$.

Особен интерес представляват функциите, определени от следната

***Дефиниция.*** Ако тоталната функция $f : X \to Y$ е инекция и сюрекция едновременно, казваме, че $f$ е *взаимно-еднозначна* или *биекция*. Ако $f : X \to Y$ е биекция, тогава еднозначно е определена биекцията $f^{-1} : Y \to X$ такава, че $\forall b' \in Y$ е в сила $f^{-1}(b') = a'$, такова че $f(a') = b'$. Наричаме я *обратна функция* на $f$.

От всички функции, които разгледахме като примери, само функцията от Фиг. 1.6.б е биекция.

***Дефиниция.*** Множеството $A$ е крайно, ако $A = \emptyset$ или $\exists n \in N, n \geq 1$ и биекция $f : A \to I_n$. Цялото $| A | = 0$, ако $A = \emptyset$ и $| A | = n$, в противен случай, наричаме *брой на елементите (кардиналност)* на $A$.

***Дефиниция.*** Казваме, че множеството $A$ е *изброимо безкрайно*, ако $\exists$ биекция $f : A \to N$. Казваме, че множеството $A$ е *изброимо*, ако е крайно или изброимо безкрайно.

Всъщност индексирането на елементите на едно множество $A$ с елементите на друго множество $I$ е знак за съществуващата между тях биекция $f : A \to I$, при което всеки елемент $e \in A$ се записва като $a_{f(e)}$, където $a$ е някакво общо обозначение за елементите на $A$. Разбира се елементите на $\emptyset$ не индексираме, защото празното множество няма елементи. Елементите на непразно крайно множество с $n$ елемента обикновено индексираме с $I_n$ (или $J_n$) за съответното $n \in N$, а на безкрайно изброимо – с $N$.

***Дефиниция.*** Нека $R \subseteq A \times A, R' \subseteq A' \times A'$ и $\gamma : A \to A'$ е такава, че $(a, b) \in R \Leftrightarrow (\gamma(a), \gamma(b)) \in R'$. Тогава $\gamma$ наричаме *хомоморфизъм* на $R$ в $R'$. Ако $\gamma$ е биекция, тогава $\gamma$ е *изоморфизъм* на $R$ и $R'$.

Нека $A$ е множество, $A = \{x_1, x_2, \ldots, x_n\}$. Дефинираме биекцията $\varphi : 2^A \to J_2^n$, съпоставяща на всяко подмножество $S \subseteq A$ вектор

$\alpha_S = (a_1, a_2, \ldots, a_n)$ от $J_2^n$ по следния начин: $a_i = 1 \Leftrightarrow x_i \in S$. Вектора $\alpha_S$ наричаме *характеристичен вектор* на подмножеството $S$. Очевидно, ако $S_1, S_2 \subseteq A, S_1 \subseteq S_2$, то $\alpha_{S_1} \preceq \alpha_{S_2}$ и обратно. Следователно $\varphi$ е изоморфизъм на частичната наредба $R_{\subseteq A}$ на подмножествата на $A, |A| = n$, в частичната наредба на съответните им характеристични вектори от $J_2^n$.

Следната теорема до голяма степен показва обхвата на понятието изброимо множество:

**Теорема 1.3.1** *Обединението на елементите на изброимо безкрайна фамилия изброимо безкрайни множества е изброимо безкрайно множество.*

***Доказателство.*** Нека множеството $A = \{A_0, A_1, A_2, \ldots\}$ е изброимо безкрайно и всеки негов елемент $A_i$ е също изброимо безкрайно множество, т.е. $A_i = \{a_{i0}, a_{i1}, a_{i2}, \ldots\}$. Ще покажем, че $\tilde{A} = \bigcup_{i \in N} A_i = \bigcup_{i \in N} \{a_{ij} | j \in N\}$ е изброимо безкрайно. За целта е достатъчно да посочим биекция $f : \tilde{A} \to N$, или (което е еквивалентно) биекция $f : N \times N \to N$, съпоставяща на всяка двойка от индекси $(i, j)$ (т.е. на всеки $a_{ij}$) единствен елемент от $N$. В таблицата по-долу са изобразени елементите на $\tilde{A}$, като елементите на $A_i$ са дадени в един ред:

$$
\begin{array}{llllll}
a_{00}, & a_{01}, & a_{02}, & \underline{a_{03}}, & \ldots, & a_{0n}, \ \ldots \\
a_{10}, & a_{11}, & \underline{a_{12}}, & a_{13}, & \ldots, & a_{1n}, \ \ldots \\
a_{20}, & \underline{a_{21}}, & a_{22}, & a_{23}, & \ldots, & a_{2n}, \ \ldots \\
\underline{a_{30}}, & a_{31}, & a_{32}, & a_{33}, & \ldots, & a_{3n}, \ \ldots \\
\ldots & & & & &
\end{array}
$$

$k$-ти слой на $\tilde{A}$ наричаме $\{a_{k0}, a_{(k-1)1}, a_{(k-2)2}, \ldots, a_{0k}\}$. В таблицата са подчертани елементите на третия слой. Исканата биекция построяваме, като последователно номерираме с естествените числа елементите на 0-вия слой, 1-вия слой,..., $k$-тия слой,... и.т.н. до безкрайност. В рамките на един слой елементите номерираме последователно по нарастващия втори индекс. Така получаваме

$$
f(i,j) = \begin{cases} 0 & i = 0, j = 0 \\ 1 + 2 + \ldots + (i+j) + j & \text{в противен случай.} \end{cases}
$$

Не е трудно да се покаже, че $f(i, j)$ е биекция. По-интересно е да се намери обратната функция $f^{-1} : N \to N \times N$, която по зададен номер определя двата индекса на съответния елемент. По-долу привеждаме процедура, пресмятаща $f^{-1}$.



**Алгоритъм.**
**Дадено**: Цяло $n \in N$.
**Резултат**: $i, j \in N, (i, j) = f^{-1}(n)$.
**Процедура**:
1. Ако $n$=0, тогава $i = 0, j = 0$. Край.
2. $k = 1$. Докато $n > k$
$\{n = n - k; k = k + 1;\}$
3. $j = n; i = k - j$; Край. □

От следващото твърдение може да се усети съществената разлика между безкрайните изброими множества и безкрайните неизброими – тези, за които не съществува биекция с естествените числа.

**Теорема 1.3.2** *Нека $A$ е изброимо безкрайно множество. Тогава $2^A$ не е изброимо.*

***Доказателство.*** Нека $A = \{a_0, a_1, a_2, \ldots\}$. Очевидно е, че $2^A$ не е крайно. Ще покажем, че не е изброимо безкрайно. Допускаме противното, т.е. че $2^A = \{A_0, A_1, A_2, \ldots, A_n, \ldots\}$. За всяко подмножество $A_n$ определяме *характеристична редица* $(b_{n0}, b_{n1}, b_{n2}, \ldots)$, в която $b_{ni} \in \{0, 1\}$ и $b_{ni} = 1$ т.с.т.к. $a_i \in A_n$. Очевидно функцията съпоставяща на всяко подмножество $A_n$ характеристична редица $(b_{n0}, b_{n1}, b_{n2}, \ldots)$ е биекция, т.е. всяка характеристична редица еднозначно определя подмножество на $A$. Образуваме таблицата от характеристичните редици на всички подмножества на $A$:

$$
\begin{array}{lllllll}
A_0: & b_{00}, & b_{01}, & b_{02}, & \ldots, & b_{0n}, & \ldots \\
A_1: & b_{10}, & b_{11}, & b_{12}, & \ldots, & b_{1n}, & \ldots \\
A_2: & b_{20}, & b_{21}, & b_{22}, & \ldots, & b_{2n}, & \ldots \\
 & \ldots & & & & & \\
A_n: & b_{n0}, & b_{n1}, & b_{n2}, & \ldots, & b_{nn}, & \ldots \\
 & \ldots & & & & &
\end{array}
$$

Нека с $\bar{A}$ означим подмножеството на $A$ определено от редицата $(\bar{b}_{00}, \bar{b}_{11}, \bar{b}_{22}, \ldots)$, където $\bar{b}_{jj} = 1$ т.с.т.к. $b_{jj} = 0$, т.е. $\forall j \in N(\bar{b}_{jj} \neq b_{jj})$. Според допускането всички подмножества на $A$ са в редицата $(A_0, A_1, A_2, \ldots)$, следователно $\exists n \in N(\bar{A} = A_n)$. Това е противоречие, защото редиците на $\bar{A}$ и $A_n$ се различават в $n$-та позиция (там $\bar{A}$ има $\bar{b}_{nn}$, а $A_n$ – $b_{nn}$). Следователно $2^A$ не е изброимо. □

Доказателството на горната теорема е типичен пример на техника, която дължим на Г. Кантор, известна като *диагонален метод на Кантор*.

Без доказателство ще приведем следното твърдение за крайни множества, популярно в математиката като

***Принцип на Дирихле.*** *Нека $X$ и $Y$ са крайни множества, $|X| > |Y|$. Тогава за всяка тотална функция $f : X \to Y$ съществуват $a_1 \neq a_2 \in X$ такива, че $f(a_1) = f(a_2)$.*

В по-неформален вид Принципът на Дирихле изглежда така:

***Принцип на чекмеджетата.*** *Ако вземем $n$ предмета и ги поставим по произволен начин в $m$ чекмеджета и $n > m$, то поне в едно чекмедже ще има поне два предмета.*

А ето и едно важно свойство на крайните частични наредби. По традиция използваме означенията $a \leq b$ или $a \geq b$, за да покажем че $a$ е, съответно, пред или след $b$ в зададена частична наредба. Аналогично, ще пишем $a < \cdot b$ или $a \cdot > b$, за да покажем, че $a$ е непосредствено пред или след $b$ в наредбата, т.е. между $a$ и $b$ няма други елементи. Едно важно свойство на крайните частични наредби дава следната

**Лема 1.3.1** *Всяка крайна частична наредба има минимален и максимален елемент.*

***Доказателство.*** Нека $R \subseteq A \times A$ е крайна частична наредба. Да допуснем, че в нея не съществува минимален елемент. Нека $a_0$ е произволен елемент на $A$, тогава $\exists a_1 \in A$, $a_1 \neq a_0$ такъв, че $a_0 \geq a_1$ (в противен случай ще се окаже, че $a_0$ е минимален, противно на допускането). Аналогично намираме $a_2 \in A, a_2 \neq a_1$ такъв, че $a_1 \geq a_2$. По този начин, разсъждавайки индуктивно, можем да построим колкото искаме дълга верига $a_1, a_2, \ldots, a_p$, където $p > |A|$. Съгласно Принципа на Дирихле в тази верига ще има поне една двойка повтарящи се елементи $a_i = a_j, i < j \leq p$, следователно в нея ще има контур. Това, обаче, е противоречие, тъй като от Теорема 1.2.2 знаем, че частичните наредби не съдържат контури.

Доказателството за съществуване на максимален елемент е аналогично. □

**Теорема 1.3.3** *Всяка крайна частична наредба се влага в пълна.*

***Доказателство.*** Предлагаме следната процедура за влагане на крайна частична наредба в пълна:

**Алгоритъм. Топологическо сортиране.**

**Дадени**: Множество $A = \{a_1, a_2, \ldots, a_n\}$ и частична наредба $R \subseteq A \times A$.



**Резултат**: Наредена $n$-торка $(a_{i_1}, a_{i_2}, \ldots, a_{i_n})$ от различните елементи на $A$, задаваща пълна наредба $R' \subseteq A \times A, R \subseteq R'$ по следния начин: $\forall m < k, m \in I_n, k \in I_n, (a_{i_m}, a_{i_k}) \in R'$.

**Процедура**:

1) $j = 1; T = R; M = A$;

2) Намираме $a_{i_j}$ – минимален в $M$ за релацията $T$;

3) $T = T \setminus \{(a_{i_j}, a_l) | (a_{i_j}, a_l) \in T\}; M = M \setminus \{a_{i_j}\}; j = j + 1$;

4) Ако $M \neq \emptyset$ преминаваме към (2), а в противен случай Край.

Тъй като всеки път избираме минимален измежду останалите елементи, изискването за влагане ще бъде очевидно изпълнено. За завършване на доказателството на теоремата остава да покажем съществуването в текущата релация $T$ на минимален елемент. В началото това е така поради твърдението на Лема 1.3.1. За следващите стъпки е достатъчно да покажем, че след отстраняването на $a_{i_j}$ и всички елементи $(a_{i_j}, a_l)$ получената релация, която е рефлексивна и антисиметрична, ще бъде без контури. Но това е очевидно, тъй като с премахване на елементи на релация без контури не е възможно да се получи релация с контур. Следователно на всяка стъпка на алгоритъма релацията $T$ е частична наредба и има минимален елемент. □

Последните две твърдения са в сила и за всяка преднаредба (т.е. релация без контури). Следващите дефиниции ще използваме по-нататък в изложението.

***Дефиниция.*** Нека $\{[a_i] | i \in I\}$ са класовете на релация на еквивалентност $R$. Ако $I = I_n, n \in N$, броя $n$ на класовете на еквивалентност наричаме *индекс на релацията на еквивалентност* $R$ и го означаваме с $IX(R)$. Казваме още, че релацията има краен индекс. Ако $I$ е безкрайно, запазваме означението $IX(R)$, но казваме, че индекса на релацията не е краен.

***Дефиниция.*** Частичната наредба $R \subseteq A \times A$ се нарича *локално-крайна*, ако всяка верига между два елемента на $A$ е крайна. Казваме, че локално-крайната частична наредба $R \subseteq A \times A$ удовлетворява *условието на Жордан-Дедекинд*, ако всички максимални вериги между два елемента на $A$ са с еднакава дължина.

## 1.4 Множества с операции

***Дефиниция.*** Нека $A$ е произволно множество и $n \in N$. Функцията $\varphi : A^n \to A$ наричаме *$n$-местна (алгебрична) операция* в $A$.

Най-често използваме едноместните и двуместни операции от вида $\varphi_1 : A \to A$ и $\varphi_2 : A^2 \to A$. В такива случаи, обикновено, вместо $\varphi_1(a)$

пишем $\varphi_1 a$ (*префиксен запис* на знака на операцията), а вместо $\varphi_2(a,b)$ пишем $a\varphi_2 b$ (*инфиксен запис* на знака на операцията).

***Дефиниция.*** Нека $\varphi_1, \varphi_2, \ldots, \varphi_m$ са операции в множеството $A$, като операцията $\varphi_i$ е $n_i$-местна, $i = 1, 2, \ldots, n$ . Тогава $\mathcal{A} = (A; \varphi_1, \varphi_2, \ldots, \varphi_m)$ наричаме *алгебра* в множеството $A$.

***Дефиниция.*** Множеството $C \subseteq A$ наричаме *затворено* относно $n$-местната операция $\varphi$, ако $\forall (a_1, a_2, \ldots, a_n) \in C^n$ е изпълнено $\varphi(a_1, a_2, \ldots, a_n) \in C$. Ако $\mathcal{A} = (A; \varphi_1, \varphi_2, \ldots, \varphi_m)$ е алгебра, а множеството $C \subseteq A$ е затворено относно операциите $\varphi_1, \varphi_2, \ldots, \varphi_m$, то $\mathcal{A}' = (C; \varphi_1', \varphi_2', \ldots, \varphi_m')$ е *подалгебра* на алгебрата $\mathcal{A}$, където $\varphi_1', \varphi_2', \ldots, \varphi_m'$ са рестрикциите на $\varphi_1, \varphi_2, \ldots, \varphi_m$ в $C$.

Като най-популярни примери ще посочим алгебрата $(R; +, .)$ на реалните числа $R$ с двуместните операции събиране $(+)$ и умножение $(.)$ и нейните подалгебри $(Q; +, .)$ и $(Z; +, .)$, породени съответно от рационалните и целите числа.

***Дефиниция.*** Нека с $\star$ сме означили двуместната операция $\varphi_\star : A^2 \to A$. Казваме, че $\varphi_\star$ е

а) *асоциативна*, ако $\forall a, b, c \in A (a \star (b \star c) = (a \star b) \star c)$;

б) *комутативна*, ако $\forall a, b \in A (a \star b = b \star a)$;

в) *идемпотентна*, ако $\forall a \in A (a \star a = a)$.

Нека с $\diamond$ сме означили друга двуместна операция $\varphi_\diamond : A^2 \to A$. Казваме, че $\varphi_\star$ е

г) *ляво-дистрибутивна* по отношение на $\varphi_\diamond$, ако $\forall a, b, c \in A (a \star (b \diamond c) = (a \star b) \diamond (a \star c))$;

д) *дясно-дистрибутивна* по отношение на $\varphi_\diamond$, ако $\forall a, b, c \in A ((a \diamond b) \star c = (a \star c) \diamond (b \star c))$;

е) *дистрибутивна* по отношение на $\varphi_\diamond$, ако е ляво-дистрибутивна и дясно-дистрибутивна по отношение на $\varphi_\diamond$.

***Дефиниция.*** Алгебрата $(A; \varphi)$ наричаме *асоциативна*, ако $\varphi$ е асоциативна и *комутативна*, ако $\varphi$ е комутативна.

По-нататък ще разглеждаме само асоциативни алгебри, даже и да не го споменаваме специално.

Операцията събиране от алгебрата на реалните числа е асоциативна и комутативна, но не е идемпотентна. Умножението е дистрибутивно по отношение на събирането, а събирането не е дистрибутивно по отношение на умножението. Всички тези свойства се пренасят естествено в подалгебрите на рационалните и целите числа.

Като друг пример да разгледаме алгебрата $(2^A; \cup, \cap)$ на подмножествата на дадено множество $A$ с операциите обединение и сечение. Освен, че са асоциативни и комутативни, операциите на тази алгебра са идемпотентни. Нещо повече, както вече видяхме, обединението и сечението са взаимно дистрибутивни по отношение една на друга.



***Дефиниция.*** Елемента $e \in A$ наричаме *ляв неутрален* за двуместната операция $\varphi_\star : A^2 \to A$, ако $\forall a \in A (e \star a = a)$ и *десен неутрален* за операцията, ако $\forall a \in A (a \star e = a)$. В случай, че $e$ е едновременно ляв и десен неутрален, казваме че той е *неутрален елемент* на операцията.

Неутралният елемент на една операция е единствен, защото ако допуснем съществуването на два неутрални елемента $e_1 \neq e_2$, ще получим видимо противоречие: $e_1 = e_1 \star e_2 = e_2$.

***Дефиниция.*** Нека $\varphi_\star : A^2 \to A$ е двуместна асоциативна операция с неутрален елемент $e$ и $a \in A$. Казваме, че елементът $b \in A$ е *ляв обратен* на $a$ (по отношение на операцията $\varphi_\star$), ако $b \star a = e$, и че е *десен обратен* на $a$ (по отношение на същата операция), ако $a \star b = e$. Ако $b$ е едновременно ляв и десен обратен на $a$, казваме, че $b$ е *обратен* на $a$ (по отношение на операцията $\varphi_\star$).

Ако $a$ има обратен елемент по отношение на операцията $\varphi_\star$, то този елемент е единствен, защото ако допуснем съществуването на два обратни елемента $b_1 \neq b_2$, $a \star b_1 = a \star b_2 = e$, ще получим, че $b_1 \star (a \star b_1) = b_1 \star (a \star b_2)$. От асоциативността на операцията следва, че $(b_1 \star a) \star b_1 = (b_1 \star a) \star b_2$ или $e \star b_1 = e \star b_2$, т.е. $b_1 = b_2$, което е противоречие. Единственият обратен елемент на $a$ по отношение на $\varphi_\star$ означаваме с $a_\star^{-1}$ или просто с $a^{-1}$, когато операцията се подразбира.

***Дефиниция.*** Нека $\mathcal{A} = (A; \varphi_1, \varphi_2, \ldots, \varphi_m)$ и $\mathcal{B} = (B; \psi_1, \psi_2, \ldots, \psi_m)$ са алгебри такива, че $\varphi_i$ и $\psi_i$ са $n_i$-местни $\forall i \in I_m$. Функцията $\gamma : A \to B$ наричаме *хомоморфизъм* на алгебрата $\mathcal{A}$ в алгебрата $\mathcal{B}$, ако

$$\gamma(\varphi_i(a_{j_1}, a_{j_2}, \ldots, a_{j_{n_i}})) = \psi_i(\gamma(a_{j_1}), \gamma(a_{j_2}), \ldots, \gamma(a_{j_{n_i}})), \forall i \in I_m.$$

Ако хомоморфизмът $\gamma$ е биекция, ще го наричаме *изоморфизъм*. Изоморфизма $\gamma : A \to A$ наричаме *автоморфизъм* на алгебрата $\mathcal{A}$.

***Дефиниция.*** Множеството $O \subseteq M$ наричаме *множество образуващи* на $(M; \star)$, ако $\forall a \in M$, различно от неутралния елемент, $a = a_{i_1} \star a_{i_2} \star \cdots \star a_{i_k}, a_{i_j} \in O, j = 1, 2, \ldots, k$ (това представяне може да не е единствено). Нека $O$ е множество образуващи на алгебрата $(M; \star)$, такова, че $\forall a \in M$, различно от неутралния елемент, $\exists$ единствено представяне $a = a_{i_1} \star a_{i_2} \star \cdots \star a_{i_k}, a_{i_j} \in O, j = 1, 2, \ldots, k$. Казваме, че $(M; \star)$ е *свободна алгебра* над $O$.

Множествата с операции са основен обект на изследване на алгебрата. Затова тук ще разгледаме само някои от тях, които представляват интерес за дискретната математика.

### 1.4.1 Полугрупи, групи и полета

***Дефиниция.*** Алгебрата $\mathcal{A} = (A; \star)$ с двуместна операция $\varphi_\star$ наричаме *полугрупа*, ако операцията е асоциативна. Ако операцията има неутрален елемент, тогава $(A; \star)$ наричаме *полугрупа с неутрален елемент*. Всяка подалгебра $\mathcal{A}'$ на $\mathcal{A}$, която е полугрупа, наричаме *подполугрупа* на $\mathcal{A}$.

За математическата теория и особено за тези дискретни математически теории, които имат приложение в информатиката, интерес представлява следната

***Дефиниция.*** Нека $X = \{a_1, a_2, \ldots, a_n\}$ е крайно множество, което наричаме *азбука*, а елементите му *букви*. Нека $\varepsilon \notin X$.

а) $\varepsilon$ е *дума* над азбуката $X$ – *празната дума*. *Дължината* на празната дума е $d(\varepsilon) = 0$;

б) нека $\alpha$ е дума над азбуката $X$ с дължина $d(\alpha)$;

в) тогава $\alpha a_i$ също е дума над азбуката $X$, $\forall i \in I_n$. Дължината ѝ е $d(\alpha a_i) = d(\alpha) + 1$;

г) няма други думи над $X$.

Да означим с $X^*$ множеството от всички думи над азбуката $X$. Нека $X^0 = \{\varepsilon\}, X^1 = X, X^n = \{\alpha | \alpha \in X^*, d(\alpha) = n\}$. Тогава $X^* = X^0 \cup X^1 \cup \cdots \cup X^n \cup \cdots$. Да означим с $X^+$ множеството от всички непразни думи над $X$, т.е. $X^+ = X^* \setminus \{\varepsilon\} = X^1 \cup X^2 \cup \cdots \cup X^n \cup \cdots$.

***Дефиниция.*** Нека $\alpha = a_{i_1} a_{i_2} \ldots a_{i_k}$ и $\beta = a_{j_1} a_{j_2} \ldots a_{j_l}$ са думи над азбуката $X$. Двуместната операция $\kappa : X^* \times X^* \to X^*$ такава, че $\kappa(\alpha, \beta) = a_{i_1} \ldots a_{i_k} a_{j_1} \ldots a_{j_l}$ наричаме *конкатенация* на $\alpha$ и $\beta$ и я записваме за по-кратко $\kappa(\alpha, \beta) = \alpha\beta$. При това означение $\varepsilon\alpha = \alpha\varepsilon = \alpha$ и $\varepsilon\varepsilon = \varepsilon$.

Нека $X = \{a, b\}$. Тогава думи над $X$ са $\varepsilon, \varepsilon a = a, \varepsilon b = b$, $aa, ab, ba, bb$, $aaa, aab, aba, abb, \ldots$.

Операцията конкатенация, очевидно, е асоциативна и $(X^*; \kappa)$ е полугрупа. В полугрупата $(X^*; \kappa)$ думата $\varepsilon$ е неутрален елемент и затова тя е полугрупа с неутрален елемент. Очевидно азбуката $X$ е множество образуващи на полугрупата $(X^*; \kappa)$, тъй като всяка дума различна от $\varepsilon$ е конкатенация на букви от азбуката. Тя е свободна алгебра, защото всяка дума от $X^+$ се представя по единствен начин като конкатенация на букви от $X$. Наричаме я *свободна полугрупа* на $X$.

За следващите два примера ще използваме крайното множество $J_n = \{0, 1, \ldots, n-1\}, n \geq 2$. Операциите $\oplus_n : J_n \times J_n \to J_n$ и $\odot_n : J_n \times J_n \to J_n$ дефинираме така: $i \oplus_n j = i + j (\mathrm{mod}\ n)$, а $i \odot_n j = i.j (\mathrm{mod}\ n)$. Асоциативността на двете операции лесно се получава от асоциативността на събирането и умножението на естествени числа. Следователно $(J_n; \oplus_n)$ и $(J_n; \odot_n)$ са полугрупи. Неутрален елемент на $\oplus_n$ е елементът 0 (неутралният елемент на алгебри с операция, записвана като събиране, често наричаме *нула* на алгебрата). Неутрален елемент на $\odot_n$ е елементът 1 (неутралният елемент на алгебри с операция, записвана като умножение, често наричаме *единица* на алгебрата). Полугрупите $(J_n; \oplus_n)$ и $(J_n; \odot_n)$ очевидно са комутативни.



В Таблица 1.1 са дадени операциите $\oplus_3$ и $\odot_3$.

| $\oplus_3$ | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 1 | 2 |
| 1 | 1 | 2 | 0 |
| 2 | 2 | 0 | 1 |

| $\odot_3$ | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 |
| 2 | 0 | 2 | 1 |

Таблица 1.1: Събиране и умножение по модул 3.

Като последен пример за полугрупа да разгледаме множеството $\mathcal{F}_M = \{f | f : M \to M\}$. Операцията $\varphi_\circ : \mathcal{F}_M \times \mathcal{F}_M \to \mathcal{F}_M$ дефинираме с $f \circ g = h(x) = f(g(x)), h \in \mathcal{F}_M$ и наричаме *композиция* на $f$ и $g$.

Ще покажем, че композицията на функции е асоциативна операция. Нека $f, g, h \in \mathcal{F}_M$. Тогава $((f \circ g) \circ h)(x) = (f \circ g)(h(x)) = f(g(h(x))) = f((g \circ h)(x)) = (f \circ (g \circ h))(x)$.

Функцията $i(x) \in \mathcal{F}_M$ такава, че $i(x) = x, \forall x \in M$ наричаме *идентитет* в $\mathcal{F}_M$. Идентитетът е неутрален елемент за композицията на функции, защото $(f \circ i)(x) = f(i(x)) = f(x) = i(f(x)) = (i \circ f)(x)$. Следователно $(\mathcal{F}_M; \circ)$ е полугрупа с неутрален елемент. Тази полугрупа играе специална роля в множеството на полугрупите с неутрален елемент, както се вижда от следната

**Теорема 1.4.1** *Нека $(M; \star)$ е полугрупа с неутрален елемент $e$. Тогава $\exists M_\mathcal{F} \subseteq \mathcal{F}_M$ такова, че $(M_\mathcal{F}; \circ)$ е подполугрупа на $(\mathcal{F}_M; \circ)$ изоморфна на $(M; \star)$.*

***Доказателство.*** Нека $a \in M$. Да означим с $f_a : M \to M$ функцията $f_a(x) = a \star x$. Ще покажем, че множеството $M_\mathcal{F} = \{f_a(x) | a \in M\}$ е затворено относно композицията на функции, т.е. че $f_a \circ f_b \in M_\mathcal{F}, \forall a, b \in M$. Нека $c = a \star b$. В сила е $f_a \circ f_b = f_c$, защото $(f_a \circ f_b)(x) = f_a(f_b(x)) = f_a(b \star x) = a \star (b \star x)$, което, заради асоциативността на операцията в полугрупата $(M; \star)$, е равно на $(a \star b) \star x = c \star x = f_c(x)$. Следователно $(M_\mathcal{F}; \circ)$ е подполугрупа на $(\mathcal{F}_M; \circ)$ с неутрален елемент $i(x) = f_e(x)$. Ще покажем, че функцията $\gamma : M \to M_\mathcal{F}$ такава, че $\gamma(a) = f_a(x)$ е изоморфизъм на $(M; \star)$ в $(M_\mathcal{F}; \circ)$. Функцията $\gamma$ е биекция, защото е тотална, изобразява $M$ върху $M_\mathcal{F}$ и ако $a \neq b, a, b \in M$, то $f_a \neq f_b$, защото $f_a(e) = a \star e = a$, а

$f_b(e) = b \star e = b$. За да завършим доказателството, трябва да докажем, че $\gamma(a \star b) = \gamma(a) \circ \gamma(b)$, т.е. $f_{a\star b}(x) = (f_a \circ f_b)(x)$. Но това е равенството, с което доказахме по-горе затвореността на $M_{\mathcal{F}}$ относно композицията на функции. □

***Дефиниция.*** Полугрупата $(M;\star)$ с неутрален елемент $e$ и такава, че $\forall a \in M\ \exists a^{-1}$ наричаме *група*. Всяка подалгебра $(M';\star)$ на групата $(M;\star)$ наричаме *подгрупа* на $(M;\star)$.

От разгледаните по-горе полугрупи, групи са алгебрите $(J_n, \oplus_n), \forall n \geq 2$, защото $\forall i \in J_n$ е определен $i^{-1} = \ominus_n i = j \in J_n$ такъв, че $i + j = n$. Тези групи са комутативни, защото съответните им полугрупи са комутативни. Свободната полугрупа над крайна азбука не е група, защото, с изключение на празната дума, никоя дума няма обратен елемент. Не са групи и алгебрите $(J_n; \odot_n)$, защото елементът 0 няма обратен за операцията $\odot_n$.

От полугрупите $(\mathcal{F}_M; \circ)$ да разгледаме един интересен частен случай. Нека $M$ е крайно, без ограничение на общността $M = I_n$. Да означим с $\mathcal{P}_n$ биекциите от $I_n$ в $I_n$, наричани още *пермутации* на елементите на $I_n$. Множеството $\mathcal{P}_n$ очевидно е затворено относно композицията на функции, а идентитетът $i(x) = x \in \mathcal{P}_n$. Следователно $(\mathcal{P}_n; \circ)$ е полугрупа с неутрален елемент. За всяка $f \in \mathcal{P}_n$ съществува обратен за операцията композиция елемент и това е обратната функция $f^{-1}$. Действително $f^{-1}$ също е биективна, т.е. $f^{-1} \in \mathcal{P}_n$ и $f^{-1} \circ f = i$. Следователно $(\mathcal{P}_n; \circ)$ е група $\forall n \geq 1$. Наричаме я *симетрична група* на $n$ елемента и я означаваме с $\mathcal{S}_n$.

Да представим всяка функция $f \in \mathcal{P}_n$ с наредената $n$-торка от стойностите ѝ $(f(1), f(2), \ldots, f(n))$. При $n = 3$ имаме следните 6 пермутации $\alpha = (1,2,3)$ – идентитетът, $\beta = (1,3,2), \gamma = (2,1,3), \delta = (2,3,1), \epsilon = (3,1,2)$ и $\zeta = (3,2,1)$. Значенията на композицията в $\mathcal{S}_3$ са дадени в Таблица 1.2. Забележете, че групата $\mathcal{S}_3 = (\mathcal{P}_3; \circ)$ не е комутативна. $(\{\alpha\}; \circ)$, $(\{\alpha, \beta\}; \circ)$ и $(\{\alpha, \delta, \epsilon\}; \circ)$ са подгрупи на $\mathcal{S}_3$ с 1,2 и 3 елемента съответно. Множествата $\{\gamma, \delta\}$ и $\{\epsilon, \zeta\}$ са образуващи на групата. Например $\alpha = \gamma \circ \gamma, \beta = \gamma \circ \delta, \epsilon = \delta \circ \delta, \zeta = \delta \circ \gamma$.

**Теорема 1.4.2** *Нека $(M;\star)$ е група, $a, a_i, a_j \in M$ и $a_i \neq a_j$. Тогава $a \star a_i \neq a \star a_j$ и $a_i \star a \neq a_j \star a$.*

***Доказателство.*** Ще докажем само първото твърдение, тъй като второто се доказва аналогично. Ако допуснем, че $a \star a_i = a \star a_j$, тогава $a^{-1} \star a \star a_i = a^{-1} \star a \star a_j$, т.е. $a_i = a_j$, което противоречи на условието. □

Казано с други думи ако $(M;\star)$ е група и $a \in M$, то множествата $a \star M = \{a \star a_i | a_i \in M\}$ и $M \star a = \{a_i \star a | a_i \in M\}$ съвпадат с М. Ако М е



| $\circ$ | $\alpha$ | $\beta$ | $\gamma$ | $\delta$ | $\epsilon$ | $\zeta$ |
|---|---|---|---|---|---|---|
| $\alpha$ | $\alpha$ | $\beta$ | $\gamma$ | $\delta$ | $\epsilon$ | $\zeta$ |
| $\beta$ | $\beta$ | $\alpha$ | $\epsilon$ | $\zeta$ | $\gamma$ | $\delta$ |
| $\gamma$ | $\gamma$ | $\delta$ | $\alpha$ | $\beta$ | $\zeta$ | $\epsilon$ |
| $\delta$ | $\delta$ | $\gamma$ | $\zeta$ | $\epsilon$ | $\alpha$ | $\beta$ |
| $\epsilon$ | $\epsilon$ | $\zeta$ | $\beta$ | $\alpha$ | $\delta$ | $\gamma$ |
| $\zeta$ | $\zeta$ | $\epsilon$ | $\delta$ | $\gamma$ | $\beta$ | $\alpha$ |

Таблица 1.2: Симетричната група $\mathcal{S}_3$.

крайно, тогава можем да кажем, че всеки ред и стълб на таблицата на груповата операция е пермутация на елементите на $M$.

**Теорема 1.4.3** *Всяка крайна група $(M;\star)$ е изоморфна на подгрупа на $\mathcal{S}_n$.*

***Доказателство.*** Нека $M = \{a_1, a_2, \ldots, a_n\}$ и нека $a_1$ е неутралният елемент на $(M;\star)$. Нека $\iota : M \to I_n$ е такава, че $\iota(a_i) = i$. Дефинираме операцията $\varphi_\star : I_n \times I_n \to I_n$ така, че $i \star j = \iota(a_i \star a_j)$. Тогава $\iota$ е естествен изоморфизъм на $(M;\star)$ в $(I_n;\star)$ и $(I_n;\star)$ е група с неутрален елемент 1. Затова ще докажем твърдението за групата $(I_n;\star)$. Тъй като $(I_n;\star)$ е полугрупа с неутрален елемент да приложим Теорема 1.4.1. Функцията $\phi : I_n \to \mathcal{F}_{I_n}$ такава, че $\phi(i) = f_i(x) = i \star x$ изобразява съгласно Теорема 1.4.1 $I_n$ в $P = \{f_1, f_2, \ldots, f_n\} \subseteq \mathcal{P}_n$ и следователно $(P;\circ)$ е полугрупа от пермутации с неутрален елемент $f_1(x) = 1 \star x = x$ – идентитетът на $\mathcal{P}_n$. Ако $i^{-1}$ е обратният на $i$ в $(I_n;\star)$, то $f_{i^{-1}}$ е обратен на $f_i$ в $(P;\circ)$, защото $f_{i^{-1}} \circ f_i = f_{i^{-1}\star i} = f_1$. Следователно $(P;\circ)$ е група, подгрупа на симетричната група $\mathcal{S}_n$. □

**Теорема 1.4.4** *Нека $(M;\star)$ е крайна група, а $(M';\star)$ е нейна подгрупа. Дефинираме $\forall a \in M$ множеството $[a] = \{x | x = a \star b, b \in M'\}$. Фамилията $\{[a_1], [a_2], \ldots, [a_k]\}$ от всички различни такива множества е разбиване на $M$.*

***Доказателство.*** Да разгледаме релацията $R \subseteq M \times M, R = \{(x,y) | \exists i \in I_k,\ x, y \in [a_i]\}$. Очевидно $R$ е рефлексивна и симетрична. Нека $(x,y) \in R$ и $(y,z) \in R$. Тогава $\exists i, j, 1 \leq i \leq k, 1 \leq j \leq k, x, y \in [a_i], y, z \in [a_j]$. Следователно $x = a_i \star b_1, y = a_i \star b_2, y = a_j \star b_3, z = a_j \star b_4$ за някои $b_1, b_2, b_3, b_4 \in M'$. От $a_i \star b_2 = a_j \star b_3 = y$ следва $a_j = a_i \star b_2 \star b_3^{-1}, b_3^{-1} \in M'$. Затова $z = a_i \star b_2 \star b_3^{-1} \star b_4$. Но $(M';\star)$ е подгрупа на $(M;\star)$ и

следователно $b_2 \star b_3^{-1} \star b_4 = b \in M'$. От тук $z = a_i \star b$, следователно $(x, z) \in R$ и $R$ е транзитивна. Следователно $R$ е релация на еквивалентност, $[a_1], [a_2], \ldots, [a_k]$ са класовете ѝ на еквивалентност и $\{[a_1], [a_2], \ldots, [a_k]\}$ е разбиване на $M$. □

От теоремата следва, че ако $(M; \star)$ е крайна група, а $(M'; \star)$ е нейна подгрупа, то $|M'|$ дели $|M|$.

***Дефиниция.*** Алгебрата $(M; +, .)$, с две двуместни операции, наричани събиране и умножение, е *поле*, ако:

а) $(M; +)$ е комутативна група, с неутрален елемент 0 – *нула на полето*;

б) $(M; .)$ е комутативна полугрупа, с неутрален елемент 1 – *единица на полето* а $(M \setminus \{0\}; .)$ е комутативна група;

в) умножението е дистрибутивно относно събирането.

Най-познато ни е полето на реалните числа с обичайните операции събиране и умножение. За дискретната математика по-голям интерес представляват *крайните полета*. Еварист Галуа е доказал, че за всяко просто цяло $p \geq 2$ и цяло $m \geq 1$ съществува единствено с точност до изоморфизъм крайно поле с $q = p^m$ елемента. В чест на откривателя им наричаме крайните полета *полета на Галуа* и означаваме с $GF(q)$ полето на Галуа с $q$ елемента. При $m = 1$ полето $GF(p) = (J_p; \oplus_p, \odot_p)$, като елементите му са остатъците при делене на простото $p$. При $m > 1$ полето се образува от всичките $p^m$ полинома от степен ненадминаваща $m - 1$ с коефициенти от $GF(p)$, като операциите са събиране и умножение на полиноми по модул някой неразложим полином от степен $m$ с коефициенти от $GF(p)$.

В Таблица 1.1 са дефинирани операциите на $GF(3)$. Най-често използваното в дискретната математика и приложенията ѝ крайно поле е $GF(2)$. Операциите му (за по-просто ще изпускаме индекса 2 при означенията им) са дадени в Таблица 1.3.

| $\oplus$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

| $\odot$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

Таблица 1.3: Поле на Галуа $GF(2)$.

Всички познати ни процедури от полето на реалните числа, например за решаване на уравнения, се пренасят безусловно в крайните полета, като операциите извършваме с използване на таблиците на полето. Да решим уравнението $x^2 + x + 2 = 0$ в полето $GF(3)$. Тъй като $4 \equiv 1 (\mathrm{mod}$



3) дискриминантата му ще бъде $1 \odot_3 1 \ominus_3 1 \odot_3 1 \odot_3 2 = 1 \ominus_3 2 = 1 \oplus_3 1 = 2$. Коренуването извършваме с помощта на таблицата. От нея се вижда, че никой елемент на полето не дава 2 при повдигане на втора степен и значи квадратен корен от 2 не е дефиниран (аналогия на опита да коренуваме отрицателно число в полето на реалните числа). Следователно, уравнението няма решения в полето $GF(3)$.

### 1.4.2 Решетки. Булеви алгебри

***Дефиниция.*** Алгебра $\mathcal{A} = (A; \vee, \wedge)$ с две двуместни операции, които са асоциативни, комутативни, идемпотентни и поглъщащи, т.е. $\forall a, b \in A$ са в сила $a \vee (a \wedge b) = a$ и $a \wedge (a \vee b) = a$, наричаме *решетка* (в някои по-стари текстове се използва терминът *структура*).

За да посочим пример за решетка да разгледаме частично наредено множество $A$, като отново без ограничение на общността използваме означението $a \leq b$ за да покажем, че $a$ предшества $b$ в наредбата. Нека $B \subseteq A$. Елементът $s \in A$ наричаме *горна граница* на множеството $B$ в частично-наредения $A$, ако $b \leq s, \forall b \in B$. Елементът $i \in A$ наричаме *долна граница* на множеството $B$ в частично-наредения $A$, ако $i \leq b, \forall b \in B$.

Горната граница $s'$ на множеството $B$, такава че $s' \leq s$ за всяка горна граница $s$ на $B$ наричаме *супремум* на $B$. Очевидно е, че ако $B$ има супремум, то той е единствен. Да допуснем противното, т.е. $\exists s' \neq s''$, супремуми на $B$. Тогава $s' \leq s''$, защото $s'$ е супремум, а $s''$ – горна граница на $B$. Но $s'' \leq s'$, защото $s''$ е супремум, а $s'$ – горна граница на $B$. От антисиметричността на частичната наредба получаваме $s' = s''$. Единственият супремум на $B$, ако съществува означаваме със $\sup B$. Долната граница $i'$ на множеството $B$, такава че $i \leq i'$ за всяка долна граница $i$ на $B$ наричаме *инфимум* на $B$. Аналогично на доказателството за супремум показваме, че ако съществува, инфимумът на $B$ е единствен и го означаваме с $\inf B$.

Лесно се вижда, че всяко едноелементно подмножество $\{a\}$ на частично-наредения $A$ има супремум и инфимум и $\sup\{a\} = \inf\{a\} = a$.

**Теорема 1.4.5** *Нека $A$ е частично-наредено множество, в което всяко двуелементно подмножество има супремум и инфимум. Тогава $(A; a \vee b = \sup\{a, b\}, a \wedge b = \inf\{a, b\})$ е решетка.*

***Доказателство.*** Ще покажем, че са в сила условията от дефиницията на решетка.

1) Комутативността следва от симетричността на понятията супремум и инфимум по отношение на двата аргумента.

2) Идемпотентността следва от $a \vee a = \sup\{a, a\} = \sup\{a\} = a$ и $a \wedge a = \inf\{a, a\} = \inf\{a\} = a$.

3) Асоциативност. Ще покажем, че $\forall a, b, c \in A$ е всила $(a \vee b) \vee c = a \vee (b \vee c) = \sup\{a, b, c\}$. Нека $d = a \vee b, e = d \vee c$ т.е. $a \leq d, b \leq d, d \leq e, c \leq e$. От транзитивността на частичната наредба следва, че $e$ е горна граница на $\{a, b, c\}$. Нека $f$ е горна граница на $\{a, b, c\}$. Тогава $f$ е горна граница и на $\{a, b\}$, и следователно $d \leq f$, защото $d = \sup\{a, b\}$. И така $f$ е горна граница на $c$ и $d$ и следователно $e \leq f$, защото $e = \sup\{c, d\}$. Следователно $(a \vee b) \vee c = e = \sup\{a, b, c\}$, защото $e$ предхожда всяка горна граница на $\{a, b, c\}$. Аналогично, заменяйки $\{a, b\}$ с $\{b, c\}$ и $c$ с $a$ доказваме, че $a \vee (b \vee c) = \sup\{a, b, c\}$. Доказателството, че $\forall a, b, c \in A$ е всила $(a \wedge b) \wedge c = a \wedge (b \wedge c) = \inf\{a, b, c\}$ е аналогично.

4) Поглъщане. Ще покажем, че $a \vee (a \wedge b) = a$ и $a \wedge (a \vee b) = a$. Очевидно $\inf\{a, b\} = a \wedge b \leq a \leq a \vee b = \sup\{a, b\}$. Следователно $a \vee (a \wedge b) = \sup\{a, a \wedge b\} = a$, а $a \wedge (a \vee b) = \inf\{a, a \vee b\} = a$.

От (1), (2), (3) и (4) следва, че $(A; a \vee b, a \wedge b)$ е решетка. □

**Теорема 1.4.6** *Нека $\mathcal{A} = (A; a \vee b, a \wedge b)$ е решетка. Тогава $\mathcal{A}$ задава частична наредба на $A$, в която всяко двуелементно подмножество на $A$ има супремум и инфимум.*

***Доказателство.*** В множеството $A$ въвеждаме релацията $R = \{a \leq b | a \vee b = b\}$. (Оставяме на читателя да докаже, че същата релация се получава от $\{a \leq b | a \wedge b = a\}$). Ще покажем, че $R$ е частична наредба. От идемпотентността $a \vee a = a$ следва, че релацията е рефлексивна. Нека $a \leq b$ и $b \leq a$, т.е. $a \vee b = b$ и $b \vee a = a$. От комутативността на операцията $\vee$ получаваме $b = a \vee b = b \vee a = a$, следователно релацията е антисиметрична. Нека $a \leq b, b \leq c$, т.е. $a \vee b = b$ и $b \vee c = c$. Тогава от асоциативността на операцията $\vee$ получаваме $c = b \vee c = (a \vee b) \vee c = a \vee (b \vee c) = a \vee c$, т.е. $a \leq c$ и релацията е транзитивна. Следователно $R$ е частична наредба. Ще покажем, че всяко двуелементно подмножество на $A$ има супремум и инфимум. Нека $c$ е горна граница на $\{a, b\}$, т.е. $a \vee c = b \vee c = c$. Следователно $c = (a \vee c) \vee (b \vee c) = (a \vee b) \vee c$ и $a \vee b \leq c$. Но $a \vee b$ е горна граница на $\{a, b\}$, защото $a \vee (a \vee b) = b \vee (a \vee b) = a \vee b$. Следователно $a \vee b = \sup\{a, b\}$. Аналогично (с помощта на алтернативната дефиниция на $R$) показваме че $\{a, b\}$ има инфимум $\inf\{a, b\} = a \wedge b$. □

Наблюдателният читател навярно е забелязал от горните доказателства, че редица свойства на решетките не се променят, ако заменим всяка



една от двете операции с другата. Това явление можем да обясним с факта, че всяка частична наредба $R_{\leq}$ определя еднозначно *двойствена релация* $R_{\geq} = \{b \geq a | a \leq b\}$. Оставяме на читателя да покаже, че $R_{\geq}$ също е частична наредба. Произволно доказателство в рамките на $R_{\leq}$ можем по аналогия да пренесем в $R_{\geq}$ като заменим $\leq$ с $\geq$. В частност, от единствеността на супремума и инфимума за всяко двуелементно множество при $R_{\leq}$ ще следва единствеността на инфимума и супремума при $R_{\geq}$ (забележете смяната на местата на инфимума и супремума при смяната на релацията!). Тези наблюдения ни позволяват да формулираме следния

***Принцип за двойственост:*** *Ако в твърдението $\pi$ за решетката $(A; \vee, \wedge)$ заменим навсякъде $\vee$ с $\wedge$ и обратно, ще получим двойственото твърдение $\pi^*$, верността на което съвпада с верността на $\pi$.*

Ще разгледаме няколко примера на решетки, естествено получаващи се от частични наредби.

Пример 1. За релацията $R_{\subseteq A} \subseteq 2^A \times 2^A$, $R_{\subseteq A} = \{(S_1, S_2) | S_1, S_2 \subseteq A, S_1 \subseteq S_2\}$ по-горе видяхме, че е частична наредба. Очевидно $\sup\{S_1, S_2\} = S_1 \cup S_2$, а $\inf\{S_1, S_2\} = S_1 \cap S_2$. Следователно $(2^A; \cup, \cap)$ е решетка.

Пример 2. За частичната наредба $R_{\preceq} \subseteq J_2^n \times J_2^n$, в която $\alpha(a_1, a_2, \ldots, a_n) \preceq \beta(b_1, b_2, \ldots, b_n)$ ако $a_i \leq b_i, i = 1, 2, \ldots, n$ показахме, че е изоморфна на $R_{\subseteq A}$, когато $|A| = n$. Нека $\varphi : 2^A \to J_2^n$ е изоморфизмът, съпоставящ на всяко подмножество $S \subseteq A$ характеристичния му вектор $\alpha_S$ от $J_2^n$. Тогава $\forall\, \alpha_{S_1}(a_1, a_2, \ldots, a_n), \alpha_{S_2}(b_1, b_2, \ldots, b_n) \in J_2^n$ получаваме

$$\begin{aligned} \sup\{\alpha_{S_1}, \alpha_{S_2}\} &= \varphi(\sup(S_1, S_2)) = \varphi(S_1 \cup S_2) = \\ &= (max(a_1, b_1), \ldots, max(a_n, b_n)) \end{aligned}$$

и

$$\begin{aligned} \inf\{\alpha_{S_1}, \alpha_{S_2}\} &= \varphi(\inf(S_1, S_2)) = \varphi(S_1 \cap S_2) = \\ &= (min(a_1, b_1), \ldots, min(a_n, b_n)). \end{aligned}$$

Означаваме $\sup\{\alpha, \beta\}$ с $\alpha \vee \beta$, а $\inf\{\alpha, \beta\}$ с $\alpha \wedge \beta$. И така $(J_2^n; \vee, \wedge)$ е решетка, изоморфна на $(2^A; \cup, \cap)$.

Пример 3. Като последен пример да разгледаме множеството $N^+$ на положителните естествени числа. Релацията $R_| \subseteq N^+ \times N^+$ дефинираме с $R_| = \{(i, j) | i, j \in N^+, j = ki, k = 1, 2, \ldots\}$. Ако $(i, j) \in R_|$ казваме, че $i$ *дели* $j$ и отбелязваме този факт с $i|j$. Лесно се вижда, че $R_|$ е частична наредба. Действително (а) $i|i$, защото $i = 1i$; (б) ако $i|j$ и $j|i$,

тогава $j = k_1 i, i = k_2 j$ и следователно $i = k_2 k_1 i, k_1 = k_2 = 1$ и $i = j$; (в) ако $i|j$ и $j|l$, тогава $j = k_1 i, l = k_2 j$ и следователно $l = k_1 k_2 i = ki$ за $k = k_2 k_1$ и $i|l$. Както добре знаем, при тази частична наредба всеки две положителни естествени числа $i$ и $j$ имат супремум – най-малкото общо кратно НМОК$(i,j)$ и инфимум – най-големият общ делител НГОД$(i,j)$ на тези числа. Следователно $(N^+;$НГОД, НМОК$)$ е решетка.

***Дефиниция.*** Нека $(M;\vee,\wedge)$ е решетка. Ако съществуват $\sup M$ и $\inf M$, тогава $\sup M$ наричаме *единица* на решетката, а $\inf M$ – *нула* на решетката и ги означаваме с $\tilde{1}$ и $\tilde{0}$ съответно. Очевидно, $a \vee \tilde{0} = a \wedge \tilde{1} = a$, $a \wedge \tilde{0} = \tilde{0}$ и $a \vee \tilde{1} = a\tilde{1}$.

***Дефиниция.*** Решетката $(M;\vee,\wedge)$ наричаме *дистрибутивна*, ако $\forall a, b, c \in M, a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$. От принципа за двойственост следва, че за дефиниция може да се вземе и двойственото равенство $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$, което също е в сила в дистрибутивната решетка.

***Дефиниция.*** Казваме, че $x$ е *допълнение* на $a \in M$ в решетката с $\tilde{0}$ и $\tilde{1}$ $(M;\vee,\wedge)$ , ако $x \wedge a = \tilde{0}$, а $x \vee a = \tilde{1}$.

**Теорема 1.4.7** *В дистрибутивна решетка $(M;\vee,\wedge)$ с $\tilde{0}$ и $\tilde{1}$ всеки елемент има не повече от едно допълнение.*

***Доказателство.*** Нека $b_1 \neq b_2$ са допълнения на $a$. Тогава $b_1 = b_1 \wedge \tilde{1} = b_1 \wedge (a \vee b_2) = (b_1 \wedge a) \vee (b_1 \wedge b_2) = \tilde{0} \vee (b_1 \wedge b_2) = b_1 \wedge b_2$. Аналогично (разменяйки $b_1$ с $b_2$ получаваме $b_2 = b_2 \wedge b_1$ и от комутативността следва, че $b_1 = b_2$. □

Допълнението на $a$ в дистрибутивна решетка $(M;\vee,\wedge)$ означаваме с $\bar{a}$.

***Дефиниция.*** Дистрибутивната решетка $(M;\vee,\wedge)$ с $\tilde{0}$ и $\tilde{1}$ такава, че всеки елемент има допълнение, се нарича *булева алгебра*. Означаваме я с $(M;\vee,\wedge,\bar{},\tilde{0},\tilde{1})$.

Пример 1. Нека $A$ е множество. Решетката $(2^A;\cup,\cap)$ е дистрибутивна. Празното множество е нула на решетката, защото е инфимум на $2^A$, а $A$ е единица на решетката, защото е супремум на $2^A$. Очевидно $\bar{S} = A \setminus S$ е допълнение на $S$ в решетката, тъй като $\bar{S} \cap S = \emptyset = \tilde{0}$ и $\bar{S} \cup S = A = \tilde{1}$. Следователно $(2^A;\cup,\cap,\bar{},\emptyset,A)$ е булева алгебра.

Пример 2. Нека $|A| = n$. Както видяхме по-горе решетката $(J_2^n;\vee,\wedge)$ е изоморфна на решетката $(2^A;\cup,\cap)$, която е булева алгебра. С използване на изоморфизма $\varphi : 2^A \to J_2^n$ намираме биективните съответствия на нулата и единицата, както и допълнението на всеки елемент. Получаваме



$\tilde{0} = \varphi(\emptyset) = \underbrace{(0,0,\ldots,0)}_{n}, \tilde{1} = \varphi(A) = \underbrace{(1,1,\ldots,1)}_{n}$ и $\forall \alpha(a_1,a_2,\ldots,a_n) = \varphi(S) \in J_2^n$ определяме допълнението му $\bar{\alpha} = \varphi(\bar{S}) = (\bar{a_1},\bar{a_2},\ldots,\bar{a_n})$.

Решетката $(N^+;$НГОД,НМОК$)$ дефинирана в Пример 3 е дистрибутивна (оставяме на читателя да провери това) и има нула – естественото число 1 (случва се и това !!!). Тя обаче няма единица и следователно не е булева алгебра.

***Дефиниция.*** Нека $\mathcal{R}$ е решетка с $\tilde{0}$, частичната наредба $R \subseteq A \times A$ на която удовлетворява условието на Жордан-Дедекинд. Тогава дължината на произволна максимална верига от $\tilde{0}$ до $a \in A$ наричаме *ранг* на $a$ и я означаваме с $r(a)$. Изображението $r : A \to N$ наричаме *рангова функция* на наредбата.

Лесно се вижда, че ако $r$ е рангова функция, то $r(\tilde{0}) = 0$, а $r(b) = r(a) + 1$, ако $a <\cdot b$.

### 1.4.3 Алгебри на Клини

***Дефиниция.*** Алгебрата $(M;+,.,*)$, в която '+' (събиране) и '.' (умножение) са двуместни операции, а '$^*$' (итерация) е едноместна операция наричаме *алгебра на Клини* ако:

а) '+' е комутативна, асоциативна, идемпотентна и има неутрален елемент, означен с $\tilde{0}$;

б) '.' е асоциативна и има неутрален елемент, означен с $\tilde{1}$;

в) $\tilde{0}.a = a.\tilde{0} = \tilde{0}$, $\forall a \in M$;

г) $a.(b+c) = (a.b) + (a.c)$ и $(b+c).a = (b.a) + (c.a)$;

д) $a^* = a^0 + a^1 + a^2 + \cdots$, където $a^0 = \tilde{1}, a^1 = a, a^{n+1} = a^n.a$.

Пример 1. Нека $X$ е крайна азбука, а $X^*$ множеството от всички думи над $X$. Всяко подмножество $L \subseteq X^*$ наричаме *език* над $X$. В множеството $2^{X^*}$ на всички езици над $X$ дефинираме *сума* и *произведение* на езици по следния начин:

$L_1 + L_2 = L_1 \cup L_2, \forall L_1, L_2 \in 2^{X^*}$;

$L_1.L_2 = \{\alpha\beta | \forall \alpha \in L_1, \beta \in L_2\}, \forall L_1, L_2 \in 2^{X^*}$.

Операцията събиране е асоциативна, комутативна и идемпотентна, защото съвпада с операцията обединение на множества. $\emptyset$ (тук *празният език*) е неутрален елемент за събирането, защото е неутрален елемент на обединението.

Операцията произведение е асоциативна, заради асоциативността на пораждащата я операция – конкатенацията на думи. Езикът $\{\varepsilon\}$ е неутрален елемент на умножението на езици, защото $\varepsilon$ е неутрален елемент на конкатенацията.

Свойствата (в) и (г) се проверяват лесно. За да е изпълнено и (д) дефинираме *итерация* на езика $L$ като $L^* = \{\varepsilon\} + L^1 + L^2 + \cdots$.

И така, множеството на езиците над крайна азбука, с въведените по-горе операции, е алгебра на Клини.

Пример 2. Нека $A$ е множество, а $\mathcal{R} = 2^{A\times A}$ – съвкупността от всички релации над $A \times A$. В този случай операциите събиране, умножение и итерация дефинираме така:

$R_1 + R_2 = R_1 \cup R_2, \forall R_1, R_2 \in 2^{A\times A}$;

$R_1.R_2 = \{(a,c)|\exists b \in A, (a,b) \in R_1, (b,c) \in R_2\}, \forall R_1, R_2 \in 2^{A\times A}$.

Неутрален елемент на събирането отново е $\emptyset$, а неутрален елемент на умножението е релацията $R_I = \{(a,a)|\forall a \in A\}$. Оставяме на читателя да провери свойствата (в) и (г). За да е изпълнено и (д) дефинираме итерацията на релацията $R$ като $R^* = R_I + R^1 + R^2 + \cdots$

И така съвкупността на релациите над декартовия квадрат на дадено множество с така дефинираните операции е алгебра на Клини.

Да разгледаме квадрата $R.R = R^2$ на произволна релация над $A \times A$. Той се състои от всички наредени двойки $(a,c)$, за които $\exists b \in A, (a,b) \in R$ и $(b,c) \in R$, т.е. $a$ е начало, а $c$ е край на верига на $R$ с дължина $\leq 2$ (забележете, че съгласно дефиницията на верига, изключваща примките, при $a = b$ или $b = c$ ще получим верига с дължина 1). Разсъждавайки индуктивно заключаваме, че $R^n$ се състои от всички наредени двойки, които са, съответно, начало и край на вериги на релацията $R$ с дължина $\leq n$, за $n = 1, 2, \ldots$. Тъй като $R_I = R^0$ се състои от всички възможни примки на релацията, очевидно е, че итерацията $R^*$ е точно рефлексивното и транзитивно затваряне на $R$.

В случай, че множеството $A$ е крайно, от принципа на Дирихле следва, че всяка верига от $a$ до $b$ с дължина $\geq n$ ще съдържа контур, но може да бъде заменена с верига от $a$ до $b$ без контури с дължина $< n$. Затова за всяка релация $R$ над крайно множество е в сила $R^{n-1} = R^n = R^{n+1} = \cdots$ и следователно рефлексивното и транзитивно затваряне на $R$ е $R^* = R_I + R + R^2 + \cdots + R^{n-1}$.

### 1.4.4 Множества с външни операции

Досега разглеждахме само операции от вида $\varphi : A^n \to A$, които нарекохме алгебрични. Нека $C$ и $A$ са множества. Функция от вида $\varphi : C \times A \to A$ наричаме *външна (за множеството A) операция*.

***Дефиниция.*** Нека $F$ е поле, а $V$ е множество с две операции, алгебрична $\varphi_1 : V \times V \to V$ – събиране (означаваме я с $+$) и външна



$\varphi_2 : F \times V \to V$ – умножение с елемент на полето (без знак за операцията), за които са в сила:

а) $(V;+)$ е комутативна група, неутралният елемент на която означаваме с $\tilde{0}$, а обратният на $v \in V$ с $-v$;

б) ако 1 е неутралният елемент на операцията умножение в полето $F$, то $1v = v, \forall v \in V$;

в) $a(bv) = (ab)v, \forall a, b \in F, \forall v \in V$;

г) $a(v_1 + v_2) = av_1 + av_2, \forall a \in F, \forall v_1, v_2 \in V$;

д) $(a_1 + a_2)v = a_1v + a_2v, \forall a_1, a_2 \in F, \forall v \in V$.

Тогава $L = (V, F)$ наричаме *линейно пространство*. Когато елементите на $V$ са вектори, наричаме $L$ *линейно векторно пространство*.

Пример. Нека $V = J_2^n$ и $F = GF(2)$. С $\tilde{0}$ означаваме $(0, 0, \ldots, 0) \in J_2^n$. Операцията *събиране на вектори* дефинираме с

$$(a_1, a_2, \ldots, a_n) + (b_1, b_2, \ldots, b_n) = (a_1 \oplus b_1, a_2 \oplus b_2, \ldots, a_n \oplus b_n),$$

а операцията *умножение на вектор с елемент на* $GF(2)$ с равенствата: $1\alpha = \alpha, 0\alpha = \tilde{0}$. Очевидно $(J_2^n; +)$ е комутативна група с неутрален елемент $\tilde{0}$, а обратният на $\alpha \in J_2^n$ е $-\alpha = \alpha$. Останалите изисквания от дефиницията на линейно пространство се проверяват лесно. И така $L_{2,n}$, определено от $J_2^n$ с дефинираните операции събиране на двоични вектори и умножение на двоичен вектор с елемент на полето $GF(2)$ е линейно векторно пространство.

Аналогично можем да дефинираме линейно векторно пространство $L_{q,n}$ за всяко крайно поле $GF(q)$ и множеството $J_q^n$ от $n$-мерните $q$-ични вектори. По-нататък в изложението под линейно пространство ще разбираме линейно векторно пространство.

Числото $n$ наричаме *размерност* на линейното пространство $L_{q,n} = (J_q^n, GF(q))$.

***Дефиниция.*** Нека $L = (V, F)$ е линейно пространство. Векторите $\alpha_1, \alpha_2, \ldots, \alpha_k \in V$ наричаме *линейно-зависими* , ако $\exists a_1, a_2, \ldots, a_k \in F$, не всички равни на 0 и такива, че $a_1\alpha_1 + a_2\alpha_2 + \ldots + a_k\alpha_k = \tilde{0}$. В противен случай, векторите са *линейно-независими*. Лявата част на предното равенство наричаме *линейна комбинация* на векторите $\alpha_1, \alpha_2, \ldots, \alpha_k$.

Без доказателство ще приведем следния важен резултат:

**Теорема 1.4.8** *В линейното векторно пространство $L_{q,n}$ максималният брой линейно-независими вектори е $n$.*

***Дефиниция.*** Произволни $n$ линейно-независими вектори наричаме *базис* на $L_{q,n}$.

***Дефиниция.*** *Скаларно произведение* на векторите $\alpha(a_1, a_2, \ldots, a_n)$ и $\beta(b_1, b_2, \ldots, b_n) \in L_{q,n}$ наричаме елементът $c \in GF(q), c = \alpha\beta = a_1b_1 \oplus_q a_2b_2 \oplus_q \cdots \oplus_q a_nb_n$.

***Дефиниция.*** Функцията $f : A \times A \to R$ наричаме *метрика (или разстояние)* в $A$, ако:

а) $f(a,b) \geq 0, \forall a, b \in A$;

б) $f(a,a) = 0, \forall a \in A$;

в) $f(a,b) = f(b,a), \forall a, b \in A$;

г) $f(a,b) + f(b,c) \leq f(a,c), \forall a, b, c \in A$ (неравенство на триъгълника).

***Дефиниция.*** Функцията $\rho : J_q^n \times J_q^n \to N$, такава че $\rho(\alpha, \beta)$ е броят на различаващите се компоненти на $\alpha$ и $\beta$ наричаме *разстояние на Хеминг*.

**Лема 1.4.1** *Разстоянието на Хеминг е метрика в* $J_q^n$.

***Доказателство.*** Свойствата а), б) и в) са очевидни. Ще докажем неравенството на триъгълника за разстоянието на Хеминг. Да образуваме матрица с три реда и $n$ стълба от произволни три вектора $\alpha, \beta$ и $\gamma \in J_q^n$ (Фиг. 1.8), като групираме вектор-стълбовете в пет групи: в първа – стълбовете с три различаващи се две по две компоненти; във втора, трета и четвърта – трите възможни типа стълбове, при които две от компонентите съвпадат, а третата е различна от тях. Стълбове от петата група, в които трите компоненти съвпадат, нямат значение за разстоянието на Хеминг. Да означим с $r_i, i = 1, 2, 3, 4$ броят на вектор-стълбовете от първите четири групи. Очевидно $\rho(\alpha, \beta) = r_1 + r_3 + r_4$, $\rho(\beta, \gamma) = r_1 + r_2 + r_3$ и $\rho(\alpha, \gamma) = r_1 + r_2 + r_4$. Сега $\rho(\alpha, \beta) + \rho(\beta, \gamma) = 2(r_1 + r_3) + r_2 + r_4 = r_1 + 2r_3 + \rho(\alpha, \gamma)$ и следователно $\rho(\alpha, \beta) + \rho(\beta, \gamma) \geq \rho(\alpha, \gamma)$, защото $r_1 + 2r_3 \geq 0$. □

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $\alpha$ | $X$ | $X$ | $X$ | $Y$ | $X$ |
| $\beta$ | $Y$ | $X$ | $Y$ | $X$ | $X$ |
| $\gamma$ | $Z$ | $Y$ | $X$ | $X$ | $X$ |

Фигура 1.8: Неравенство на триъгълника за разстоянието на Хеминг.

С помощта на разстоянието на Хеминг в $J_q^n$ можем да дефинираме *тегло* на $q$-ичния вектор $\alpha$ като $wt(\alpha) = \rho(\alpha, \tilde{0})$. Очевидно теглото на всеки вектор е равно на броя на ненулевите му компоненти.

***Дефиниция.*** Нека $L = (V, F)$ е линейно пространство, а $L' = (V', F')$, $V' \subseteq V$, $F' \subseteq F$ също е линейно пространство. Тогава наричаме



$L'$ *линейно подпространство* на $L$. Очевидно, неутралният елемент $\tilde{0}$ на групата $(V; +)$ принадлежи на всяко подпространство на $L$.

Ако размерността на $L$ е $n$ и $L'$ е линейно подпространство на $L$ с размерност $k$, то $0 \leq k \leq n$. Множеството $L'' = \{\beta | \beta\alpha = 0, \forall \alpha \in L'\}$ също е линейно подпространство на $L$, наричаме го *ортогонално подпространство* на $L'$. Размерността на $L''$ е $n - k$.

---

## Упражнения

### Упражнение 1.1

Обосновете правилността или неправилността на твърденията:

а) съвкупността от всички протоелементи е множество;

б) съвкупността от всички множества е множество.

### Упражнение 1.2

За всяко от следващите множества посочете броя на елементите и колко от тях са множества:

а) $\emptyset$; б) $\{a\}$; в)$\{\{a\}\}$;

г) $\{\emptyset\}$; д)$\{\emptyset, \{a, \{a\}\}\}$;

е) $\{\{a\}, b, \{\{\{c\}, \{d, e\}\}, a, \{b, c\}\}\}$.

### Упражнение 1.3

Опитайте се да определите по-точно всяко от следните множества:

а) $A = \{2, 4, 6, \ldots, 12\}$;

б) $B = \{$януари,февруари,...,декември$\}$;

в) $C = \{2, 3, 6, 7, 14, \ldots, 127\}$;

г) $D = \{0, 1, 1, 2, 3, 5, 8, \ldots, 34\}$;

д) $C' = \{2, 3, 6, 7, 14, \ldots, 127, \ldots\}$;

е) $D' = \{0, 1, 1, 2, 3, 5, 8, \ldots, 34, \ldots\}$.

### Упражнение 1.4

Проверете кои от следните твърдения са вярни:

а) $\emptyset \subseteq \emptyset$; б) $\emptyset \in \emptyset$; в) $\emptyset \in \{\emptyset\}$; г) $\emptyset \subseteq \{\emptyset\}$; д) $a \subseteq \{a\}$;

е) $\{a, b\} \in \{a, b, \{a, b\}\}$; ж) $\{a, b\} \subseteq \{a, b, \{a, b\}\}$;

з) $\{a,b\} \subseteq \{\{a,b\}\}$; и) $\{a,b\} \subseteq 2^{\{a,b,\{a,b\}\}}$;
к) $\{a,b\} \in 2^{\{a,b,\{a,b\}\}}$; л) $\{a,b\} = \{a,b,\{a,b\}\} \setminus \{a,b\}$.

### Упражнение 1.5

Представете в по-прост вид множествата:
а) $(\{1,3,5\} \setminus \{2,3,4\}) \cap (\{5,7,9\} \setminus \{3,7,5\})$;
б) $(\{1,2,5\} \setminus \{5,7,9\}) \cup (\{5,7,9\} \setminus \{1,2,5\})$;
в) $2^{\{7,8,9\}} \setminus 2^{\{7,9\}}$; г) $2^{\emptyset}$.

### Упражнение 1.6

Докажете, че $\forall n \in N \setminus \{0\}$:
а) $n^2 = 1+3+\cdots+(2n-1)$;
б) $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$;
в) $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$;
г) $\sum_{i=1}^{n} i^3 = (\sum_{i=1}^{n} i)^2$;
д) $1.2.3 + 2.3.4 + \cdots + n(n+1)(n+2) = \frac{n(n+1)(n+2)(n+3)}{4}$.

### Упражнение 1.7

Разгледайте следното „доказателство“ на абсурдното твърдение *Всички елементи на едно непразно множество са еднакви*: „а) За множествата с един елемент твърдението е очевидно; б) Да допуснем, че твърдението е в сила за всяко множество с $n$ елемента; в) Нека $S$ е $(n+1)$-елементно множество и $a \in S$. Образуваме подмножества на $S$ с $n$ елемента $S_1$ и $S_2$ такива, че $S_1 \cup S_2 = S$, $a \in S_1$, $a \in S_2$. От индукционното предположение следва, че всички елементи на $S_1$ съвпадат с $a$ и всички елементи на $S_2$ съвпадат с $a$. Следователно всички елементи на $S$ съвпадат помежду си.“ Къде е грешката в това доказателство?

### Упражнение 1.8

Намерете елементите на множеството:
а) $\{1,3\} \times \{1,2\} \times \{1,2,3\}$;
б) $\emptyset \times \{1,2\}$;
в) $2^{\{1,2\}} \times \{1,2\}$.



### Упражнение 1.9

Нека $S = \{a,b,c,d\}$.
а) Постройте разбиванията на $S$ с максимален и минимален брой елементи;
б) Постройте всички разбивания на $S$ с 2 елемента;
в) Намерете броя на различните разбивания на $S$.

### Упражнение 1.10

Проверете за рефлексивност, антирефлексивност, симетричност, антисиметричност и транзитивност следните релации над декартовия квадрат на множеството от всички хора:
а) $R = \{(a,b) | a$ е баща на $b\}$;
б) $R = \{(a,b) | a$ и $b$ имат общ родител$\}$;
в) $R = \{(a,b) | a$ е предшественик (родител или прародител) на $b\}$.

### Упражнение 1.11

Докажете, че следните релации са релации на еквивалентност:
а) $R = \{(a,b) | a$ и $b$ имат общ баща$\}$;
б) $R = \{(a,b) | a,b \in N, a \equiv b (\mathrm{mod}\, q)\}, \forall q \in N, q \geq 2$.

### Упражнение 1.12

За всяко от следните разбивания на множеството $\{a,b\}^{\leq 3}$, съставено от думите на азбуката $\{a,b\}$ с дължина не надминаваща 3, опишете (без да изброявате двойки) релация на еквивалентност, която го поражда :
а) $\{\{\varepsilon\}, \{a,b\}, \{aa,ab,ba,bb\}, \{aaa,aab,aba,abb,baa,bab,bba,bbb\}\}$;
б) $\{\{\varepsilon\}, \{a,aa,aaa,aba\}, \{ab,aab,abb\}, \{ba,baa,bba\}, \{b,bb,bab,bbb\}\}$;
в) $\{\{\varepsilon,a,b\}, \{aa,ba,aaa,aab,baa,bab\}, \{ab,bb,aba,abb,bba,bbb\}\}$.

### Упражнение 1.13

Проверете кои от следните релации са частични наредби:
а) $R = \{(a,b) | a,b \in N, (a = b$ или $b = a+1)\}$;
б) $R = \{((a,b),(c,d)) | a,b,c,d \in N, a \leq c, b \leq d\}$;
в) $R = \{((a,b),(c,d)) | a,b,c,d \in N, a \leq c$ или $b \leq d\}$.

**Упражнение 1.14**

Докажете, че ако $R_1$ и $R_2$ са частични наредби над $A \times A$, то $R_1 \cap R_2$ е частична наредба. Частична наредба ли е $R_1 \cup R_2$?

**Упражнение 1.15**

Постройте всички частични наредби на множеството $\{a, b, c, d\}$ такива, че:

а) $a$ е единствен минимален, а $d$ – единствен максимален елемент;
б) $a$ е едновременно минимален и максимален елемент;
в) $a$ и $b$ са несравними.

**Упражнение 1.16**

Постройте диаграма на релация с минимален брой елементи, рефлексивно и транзитивно затваряне на която е частичната наредба $R_{\subseteq\{1,2,3\}}$.

**Упражнение 1.17**

Намерете рефлексивното, симетричното и транзитивното затваряне на релацията $R \subseteq \{a, b, c, d, e\}^2$, $R = \{(a, b), (a, c), (a, d), (d, c), (d, e)\}$.

**Упражнение 1.18**

Нека $\pi$ е фамилията от всички разбивания на крайното множество $S$, а $R \subseteq \pi \times \pi$ е такава, че $(\pi_1, \pi_2) \in R$ т.с.т.к. $\forall S_1 \in \pi_1\ (\exists S_2 \in \pi_2, S_1 \subseteq S_2)$. Покажете, че $R$ е частична наредба. Постройте тази наредба при $S = \{1, 2, 3\}$.

**Упражнение 1.19**

Вложете релацията без контури $R \subset I_9 \times I_9$, $R = \{(1, 4), (1, 5), (2, 5), (2, 6), (3, 2), (3, 5), (3, 6), (5, 4), (6, 7), (6, 8), (6, 9), (7, 4), (8, 5), (8, 9), (9, 7)\}$ в пълна, като използвате топологическо сортиране.

**Упражнение 1.20**

Докажете, че фамилията от крайните подмножества на изброимо множество е изброима.



**Упражнение 1.21**

Посочете биекции между:

а) $N$ и нечетните в $N$;
б) $N$ и $Z$;
в) $N$ и $N^3$.

**Упражнение 1.22**

Докажете, че $(2^A; \cup)$ и $(2^A; \cap)$ са полугрупи с неутрален елемент.

**Упражнение 1.23**

Нека $(M; \star)$ е комутативна полугрупа и $\forall a \in M, a \star a = a$ (*идемпотентна полугрупа*). Докажете, че $(M; \star)$ е изоморфна на подполугрупа на $(2^M; \cap)$.

**Упражнение 1.24**

Постройте групата от автоморфизми на квадрата $ABCD$ в Евклидовата равнина.

**Упражнение 1.25**

Докажете, че $(2^A; \triangle)$ е група.

**Упражнение 1.26**

Нека $(M_1; \star)$ и $(M_2; \star)$ са групи. Докажете, че $(M_1 \times M_2; \circ)$, където $(a_1, b_1) \circ (a_2, b_2) = (a_1 \star a_2, b_1 \star b_2)$, е група.

**Упражнение 1.27**

Решете в $GF(3)$:

а) уравнението $x^3 + 2x_2 - 2 = 0$;
б) системата линейни уравнения:

$$\left|\begin{array}{rcrcrcl} x & + & 2y & + & z & = & 1 \\ x & + & y & - & z & = & 0 \\ x & - & y & + & z & = & 1 \end{array}\right.$$

**Упражнение 1.28**

Докажете, че ако $(M; \vee, \wedge)$ е дистрибутивна решетка с $\bar{0}$ и $\bar{1}$ и елементите $a$ и $b$ имат допълнения, то елементите $a \vee b$ и $a \wedge b$ също имат допълнения, като $\overline{a \vee b} = \bar{a} \wedge \bar{b}$ и $\overline{a \wedge b} = \bar{a} \vee \bar{b}$.

**Упражнение 1.29**

Нека $(M; \vee, \wedge, \bar{}, \bar{0}, \bar{1})$ е булева алгебра и $R_{\preceq}$ е дефинираната от нея частична наредба в $M$. Докажете, че:

а) ако $\alpha \preceq \beta$ то в $M' = \{\gamma | \alpha \preceq \gamma \preceq \beta\}$ може да се дефинира булева алгебра при подходящ избор на операциите и константите.

б) ако $\alpha \not\preceq \beta$ и $\beta \not\preceq \alpha$ то в $M' = \{\gamma | \inf\{\alpha, \beta\} \preceq \gamma \preceq \sup\{\alpha, \beta\}\}$ може да се дефинира булева алгебра при подходящ избор на операциите и константите.

**Упражнение 1.30**

Докажете, че подмножеството от двоичните $n$-мерни вектори с четно тегло е линейно подпространство на пространството от всички двоични вектори. Какво можете да кажете за подмножеството от $q$-ичните вектори с четни тегла?

# Глава 2

# Комбинаторика

Комбинаториката е измежду най-старите и силно развити математически теории, които причисляваме към дискретната математика. Обектите, с които се занимава комбинаториката наричаме *комбинаторни конфигурации*. Комбинаторните конфигурации строим от елементите на някое крайно множество, комбинирайки ги по зададени правила.

Най-често срещаните задачи на комбинаториката са по зададено множество и правила за комбиниране да се намери броят на получаващите се комбинаторни конфигурации като функция на броя на елементите в образуващото множество и евентуално, на някои други параметри, зададени в правилата. Предполага се, че правилата за комбиниране са такива, че получените конфигурации са краен брой. На този род задачи е посветена голяма част от изследванията в областта на комбинаториката, която ще наричаме *изброителна комбинаторика*. Трябва да отбележим, че изброяването на комбинаторните конфигурации, даже при много прости правила, не винаги е тривиална задача. Напротив, много задачи на изброителната комбинаторика все още не са решени.

От друга страна, важен теоретичен въпрос е, при зададени правила за комбиниране на елементите да се изследват свойствата (структурата) на получените комбинаторни конфигурации и, в частност, да се докаже съществуването на комбинаторни конфигурации с предварително зададени свойства (параметри). Тази част на комбинаториката ще наричаме *структурна комбинаторика*. Всъщност, редица комбинаторни конфигурации са толкова съществени за практиката и дискретната математика, че на изследването на структурата им са посветени самостоятелни раздели на комбинаториката - теорията на крайните графи и дървета, теорията на блок-дизайните, теорията на матроидите, теорията на крайните геометрии и т. н.

В тази глава ще се спрем накратко на основите на изброителната

комбинаторика и на структурните свойства на някои интересни комбинаторни конфигурации. Поради изключително голямата роля на крайните графи и дървета ще ги разгледаме подробно в трета глава.

## 2.1 Принципи на изброителната комбинаторика

Решаването на основната задача на изброителната комбинаторика не рядко е изключително труден процес, изискващ голяма математическа култура и изобретателност. Независимо от това съществуват известен брой прости правила, следствие от съответни твърдения, които са изключително полезни и познаването им е задължително за всеки, който започва изучаването на комбинаториката. Доказателството на съответните твърдения понякога е толкова очевидно, че го пропускаме, а самите твърдения често се наричат принципи.

Един от най-често използваните принципи на изброителната комбинаторика е *Принципът на Дирихле*, с който се запознахме в предната глава. Ще припомним тук това твърдение в неговата най-популярна форма

**Теорема 2.1.1 (Принцип на чекмеджетата)** *Нека $X$ е множество с $n$ елемента (предмета), а $Y$ е множество с $m$ елемента (чекмеджета) и $n > m$. Както и да поставим всички предмети в чекмеджетата, поне в едно чекмедже ще има поне два предмета.*

Ще приложим Принципа на Дирихле за доказателството на следващата

**Теорема 2.1.2 (Принцип на биекцията)** *Нека $X$ и $Y$ са крайни множества, $|X| = n$ и $|Y| = m$. Съществува биекция $f : X \to Y$ тогава и само тогава, когато $n = m$.*

***Доказателство.*** 1) Нека $n = m$. Тогава можем да напишем $X = \{a_1, a_2, \ldots, a_n\}$ и $Y = \{b_1, b_2, \ldots, b_n\}$. Сега дефинираме функцията $f : X \to Y$ по следния начин: $f(a_i) = b_i$, $\forall i \in I_n$, която очевидно е биекция.

2) Нека $f : X \to Y$ е биекция. Да допуснем, че $n > m$. Първо ще отбележим, че ако гледаме на елементите на $X$ като на предмети, а на елементите на $Y$ като на чекмеджета, то всяка функция $g : X \to Y$ задава един начин на поставяне на предметите от $X$ в чекмеджетата от $Y$ – предметът $a_i$ е поставен в чекмеджето $g(a_i)$. Съгласно Принципа на Дирихле $\forall g : X \to Y$, $\exists b \in Y, \exists a_i \neq a_j \in X$ такива, че $g(a_i) = g(a_j) = b$.



Същото е в сила за $f$, което противоречи на факта, че $f$ е биекция. Аналогично опровергаваме допускането, че $m > n$, като използваме биекцията $f^{-1}$. Следователно $n = m$. □

**Теорема 2.1.3** ***(Принцип на разбиването, Принцип на събирането)*** *Нека $A$ е крайно множество, а $\mathcal{R} = \{S_1, S_2, \ldots, S_k\}$ е разбиване на $A$. Тогава $|A| = \sum_{i=1}^{k} |S_i|$.*

Твърдението е очевидно. Ще си позволим обаче да го подкрепим с две не съвсем формални съображения на здравия разум, които много често помагат при решаването на изброителни задачи. За да бъде приета за правилна една формула за брой на елементите на някакво множество е необходимо и достатъчно *всеки елемент да е преброен* и *никой елемент да не е преброен повече от един път*. В горното твърдение двете съображения очевидно са налице, заради свойствата на разбиването. $A = \bigcup_{i=1}^{k} S_i$ означава, че всеки елемент участва в някое $S_i$ и следователно е преброен, а $S_i \cap S_j = \emptyset, i \neq j$ означава, че никой елемент не е преброен повече от 1 път.

Елементарно следствие от Принципа на разбиването при $k = 2$ е следната

**Теорема 2.1.4 (Принцип на изваждането)** *Нека $A$ е крайно множество, $A', A'' \subseteq A, A' = A \setminus A''$. Тогава $|A'| = |A| - |A''|$.*

Принципът на изваждането е много удобен за случаите, когато броят на елементите на $A$ е известен, а броят на елементите на $A''$ се пресмята много по-лесно от броя на елементите на $A'$. За илюстрация да изразим броя на елементите на множеството $B'$ на $n$-мерните двоични вектори които имат поне две единици, чрез броя на всички вектори. Да означим с $B''$ множеството от останалите вектори от $J_2^n = \{0,1\}^n$, т.е. векторите с по-малко от 2 единици. Тъй като имаме точно един вектор без нито една единица и точно $n$ вектора с точно една единица, то $|B''| = n + 1$ и получаваме, че $|B'| = |J_2^n| - n - 1$.

Ще приложим Принципа на разбиването за доказателство на следващата

**Теорема 2.1.5** ***(Принцип на декартовото произведение, Принцип на умножението)*** *Нека $X$ и $Y$ са крайни множества, $|X| = n, |Y| = m$. Тогава $|X \times Y| = |X|.|Y| = nm$.*

***Доказателство.*** 1) Нека $n = 0$, т.е. $X = \emptyset$. Тогава $X \times Y = \emptyset$ и $|X \times Y| = 0 = nm$. При $m = 0$ доказателството е аналогично.

2) Нека $n \neq 0, m \neq 0, X = \{a_1, a_2, \ldots, a_n\}, Y = \{b_1, b_2, \ldots, b_m\}$. Нека $\forall a_i \in X$ дефинираме множеството $S_{a_i} = \{(a_i, b) | b \in Y\}$ и да означим с $\mathcal{R} = \{S_{a_1}, S_{a_2}, \ldots, S_{a_n}\}$ фамилията от тези множества. От естествената биекция $f : X \to \mathcal{R}, f(a_i) = S_{a_i}$ следва, че $|\mathcal{R}| = |X| = n$. За всяко $S_{a_i}$ дефинираме биекцията $g_i : S_{a_i} \to Y$, $g_i(a_i, b) = b$. Следователно за всяко $S_{a_i}$ имаме $|S_{a_i}| = |Y| = m$. Но множеството $\mathcal{R}$ е разбиване на $X \times Y$, защото

а) $|S_{a_i}| = m \neq 0$ и следователно $S_{a_i} \neq \emptyset$;

б) $S_{a_i} \cap S_{a_j} = \emptyset, a_i \neq a_j$;

в) $\bigcup_{i \in I_n} S_{a_i} = X \times Y$.

От Принципа на разбиването получаваме $|X \times Y| = \sum_{i \in I_n} |S_{a_i}| = \sum_{i \in I_n} m = nm = |X|.|Y|$. □

**Следствие 2.1.1** *Нека* $|A_1| = m_1, |A_2| = m_2, \ldots, |A_n| = m_n$. *Тогава* $|A_1 \times A_2 \times \cdots \times A_n| = m_1.m_2.\cdots.m_n$.

Доказателството извършваме като приложим $n - 1$ пъти Принципа на декартовото произведение.

**Следствие 2.1.2** *Нека* $|A| = m$. *Тогава* $|A^n| = |A|^n = m^n$.

Доказателството получаваме от Следствие 2.1.1 при $A = A_1 = A_2 = \cdots = A_n$. В частния случай $A = J_2^n$ получаваме за броя на $n$-мерните двоични вектори $|J_2^n| = |J_2|^n = 2^n$.

**Следствие 2.1.3** *Нека* $|A| = n$. *Тогава* $|2^A| = 2^{|A|}$.

***Доказателство.*** Ще използваме биекцията между $2^A, |A| = n$ и $J_2^n$, съпоставяща на всяко подмножество на $A$ съответния му характеристичен вектор. От Принципа на биекцията следва, че $|2^A| = |J_2^n| = 2^n = 2^{|A|}$. □

Понякога е много по-лесно да разгледаме вместо множеството $A$, чиито елементи искаме да изброим, множеството $A'$ получено като за всеки елемент на $A$ в $A'$ участват $m \neq 0$ негови представители и всеки елемент на $A'$ е представител на точно един елемент от $A$. В сила е следната очевидна

**Теорема 2.1.6 (Принцип на делението)** *Нека $A$ е крайно множество и $A'$ е построено от $A$ с повтаряне на всеки елемент $m \neq 0$ пъти. Тогава* $|A| = |A'|/m$.



В следващия раздел ще дадем пример за типично използване на Принципа на делението. А сега ще докажем едно твърдение, което далеч не е толкова просто, както изредените по-горе. То в известен смисъл обобщава Принципа на разбиването за случая, в който фамилията подмножества на множеството $A$ не образува разбиване. Да напомним, че с $\overline{Y}^A$ означаваме разликата $X \setminus Y$(допълнението на $Y$ до $X$).

**Теорема 2.1.7** ***(Принцип на включването и изключването)*** *Нека $A$ е крайно множество и $A_1, A_2, \ldots, A_n \subseteq A$. Тогава*

$$|\overline{A_1}^A \cap \overline{A_2}^A \cap \cdots \cap \overline{A_n}^A| = |A| - \sum_{i \in I_n} |A_i| + \sum_{\substack{i,j \in I_n \\ i \neq j}} |A_i \cap A_j| -$$

$$- \sum_{\substack{i,j,k \in I_n \\ i \neq j, j \neq k, i \neq k}} |A_i \cap A_j \cap A_k| + \cdots + (-1)^n |A_1 \cap A_2 \cap \cdots \cap A_n|.$$

***Доказателство.*** Ще докажем твърдението с индукция по $n$. Нека $n = 1$. Твърдението се превръща в $|\overline{A_1}^A| = |A| - |A_1|$ или $|A \setminus A_1| = |A| - |A_1|$, но това е твърдението на Принципа на изваждането.

Да допуснем, че твърдението е в сила за $A$ и подмножествата $A_1, A_2, \ldots, A_{n-1}$, т. е.

$$|\overline{A_1}^A \cap \overline{A_2}^A \cap \cdots \cap \overline{A_{n-1}}^A| = |A| - \sum_{i \in I_{n-1}} |A_i| + \sum_{\substack{i,j \in I_{n-1} \\ i \neq j}} |A_i \cap A_j| -$$

$$- \sum_{\substack{i,j,k \in I_{n-1} \\ i \neq j, j \neq k, i \neq k}} |A_i \cap A_j \cap A_k| + \cdots + (-1)^{n-1} |A_1 \cap A_2 \cap \cdots \cap A_{n-1}|.$$

Ще докажем, че от тук следва твърдението за $A$ и $A_1, A_2, \ldots, A_n$. Множеството $\overline{A_1}^A \cap \overline{A_2}^A \cap \cdots \cap \overline{A_{n-1}}^A$ се разбива от множествата $\overline{A_1}^A \cap \cdots \cap \overline{A_{n-1}}^A \cap \overline{A_n}^A$ и $\overline{A_1}^A \cap \cdots \cap \overline{A_{n-1}}^A \cap A_n$, защото всеки елемент на първото множество е елемент на $A_n$ или не е елемент на $A_n$, т. е. е елемент на $\overline{A_n}^A$. Следователно от Принципа на разбиването

$$\begin{aligned} |\overline{A_1}^A \cap \overline{A_2}^A \cap \cdots \cap \overline{A_{n-1}}^A| &= |\overline{A_1}^A \cap \cdots \cap \overline{A_{n-1}}^A \cap \overline{A_n}^A| + \\ &+ |\overline{A_1}^A \cap \cdots \cap \overline{A_{n-1}}^A \cap A_n|, \end{aligned}$$

от където

$$\begin{aligned} |\overline{A_1}^A \cap \cdots \cap \overline{A_{n-1}}^A \cap \overline{A_n}^A| &= |\overline{A_1}^A \cap \overline{A_2}^A \cap \cdots \cap \overline{A_{n-1}}^A| - \\ &- |\overline{A_1}^A \cap \cdots \cap \overline{A_{n-1}}^A \cap A_n|. \end{aligned} \quad (2.1)$$

Множеството $\overline{A_1}^A \cap \cdots \cap \overline{A_{n-1}}^A \cap A_n$ можем да представим като $(\overline{A_1}^A \cap A_n) \cap \cdots \cap (\overline{A_{n-1}}^A \cap A_n)$ заради комутативността и идемпотентността на операцията сечение и заместваики в (2.1) да получим

$$|\overline{A_1}^A \cap \cdots \cap \overline{A_{n-1}}^A \cap \overline{A_n}^A| = |\overline{A_1}^A \cap \overline{A_2}^A \cap \cdots \cap \overline{A_{n-1}}^A| - |(\overline{A_1}^A \cap A_n) \cap \cdots \cap (\overline{A_{n-1}}^A \cap A_n)|. \quad (2.2)$$

Лесно се вижда верността на следното твърдение: $\overline{A_i}^A \cap A_n = A_n \setminus A_i = \overline{A_i \cap A_n}^{A_n}, \forall i \in I_{n-1}$ т.е. $(\overline{A_1}^A \cap A_n) \cap \cdots \cap (\overline{A_{n-1}}^A \cap A_n) = \overline{A_1 \cap A_n}^{A_n} \cap \cdots \cap \overline{A_{n-1} \cap A_n}^{A_n}$. От индукционното предположение (приложено за множеството $A_n$ и подмножествата му $\overline{A_1 \cap A_n}^{A_n}, \ldots, \overline{A_{n-1} \cap A_n}^{A_n}$ получаваме

$$|\overline{A_1 \cap A_n}^{A_n} \cap \ldots \cap \overline{A_{n-1} \cap A_n}^{A_n}| = \quad (2.3)$$

$$= |A_n| - \sum_{i \in I_{n-1}} |A_i \cap A_n| + \sum_{\substack{i,j \in I_{n-1} \\ i \neq j}} |A_i \cap A_j \cap A_n| -$$

$$- \sum_{\substack{i,j,k \in I_{n-1} \\ i \neq j, j \neq k, i \neq k}} |A_i \cap A_j \cap A_k \cap A_n| + \cdots + (-1)^{n-1} |A_1 \cap \cdots \cap A_{n-1} \cap A_n|.$$

Сега, заместваики (2.3) в (2.2) получаваме

$$|\overline{A_1}^A \cap \cdots \cap \overline{A_{n-1}}^A \cap \overline{A_n}^A| =$$

$$= |\overline{A_1}^A \cap \overline{A_2}^A \cap \cdots \cap \overline{A_{n-1}}^A| - |(\overline{A_1}^A \cap A_n) \cap \cdots \cap (\overline{A_{n-1}}^A \cap A_n)| =$$

$$= |\overline{A_1}^A \cap \overline{A_2}^A \cap \cdots \overline{A_{n-1}}^A| - |\overline{A_1 \cap A_n}^{A_n} \cap \ldots \cap \overline{A_{n-1} \cap A_n}^{A_n}| =$$

$$= |A| - \sum_{i \in I_{n-1}} |A_i| + \sum_{\substack{i,j \in I_{n-1} \\ i \neq j}} |A_i \cap A_j| -$$

$$- \sum_{\substack{i,j,k \in I_{n-1} \\ i \neq j, j \neq k, i \neq k}} |A_i \cap A_j \cap A_k| + \cdots + (-1)^{n-1} |A_1 \cap A_2 \cap \cdots \cap A_{n-1}| -$$

$$- |A_n| + \sum_{i \in I_{n-1}} |A_i \cap A_n| - \sum_{\substack{i,j \in I_{n-1} \\ i \neq j}} |A_i \cap A_j \cap A_n| +$$

$$+ \sum_{\substack{i,j,k \in I_{n-1} \\ i \neq j, j \neq k, i \neq k}} |A_i \cap A_j \cap A_k \cap A_n| + \cdots + (-1)^n |A_1 \cap \cdots \cap A_{n-1} \cap A_n| =$$

$$= |A| - \sum_{i \in I_n} |A_i| + + \sum_{\substack{i,j \in I_n \\ i \neq j}} |A_i \cap A_j| -$$

$$- \sum_{\substack{i,j,k \in I_n \\ i \neq j, j \neq k, i \neq k}} |A_i \cap A_j \cap A_k| + \cdots + (-1)^n |A_1 \cap A_2 \cap \cdots \cap A_n|. \square$$



Да приложим Принципа за включване и изключване и да намерим броя $P_{100}$ на простите числа $p, 1 < p \leq 100$. Едно число в този интервал е просто, ако не се дели на нито едно просто число не по-голямо от $\sqrt{100} = 10$. Затова да означим с $A$ множеството от разглежданите 99 числа, а с $A_{i_1,i_2,\ldots,i_r}$ множеството на непростите числа от интервала, кратни едновременно на $i_1, i_2, \ldots, i_r$, където $\{i_1, i_2, \ldots, i_r\} \subseteq \{2,3,5,7\}$. Не е трудно да се покаже, че $|A_{i_1,i_2,\ldots,i_r}| = \lfloor 100/(i_1.i_2.\cdots.i_r) \rfloor$ при $r > 1$ и $|A_{i_1}| = \lfloor 100/i_1 \rfloor - 1$.

От Принципа за включване и изключване, като предварително отстраним празните множества $A_{i_1,i_2,\ldots,i_r}$, за които $i_1 i_2 \cdots i_r > 100$, получаваме

$$P_{100} = |A| - |A_2| - |A_3| - |A_5| - |A_7| +$$
$$+ |A_{2,3}| + |A_{2,5}| + |A_{2,7}| + |A_{3,5}| + |A_{3,7}| + |A_{5,7}| -$$
$$- |A_{2,3,5}| - |A_{2,3,7}| - |A_{2,5,7}| =$$
$$= 99 - \lfloor 100/2 \rfloor + 1 - \lfloor 100/3 \rfloor + 1 - \lfloor 100/5 \rfloor + 1 - \lfloor 100/7 \rfloor + 1 +$$
$$+ \lfloor 100/6 \rfloor + \lfloor 100/10 \rfloor + \lfloor 100/14 \rfloor + \lfloor 100/15 \rfloor + \lfloor 100/21 \rfloor + \lfloor 100/35 \rfloor -$$
$$- \lfloor 100/30 \rfloor - \lfloor 100/42 \rfloor - \lfloor 100/70 \rfloor =$$
$$= 99 - 49 - 32 - 19 - 13 + 16 + 10 + 7 + 6 + 4 + 2 - 3 - 2 - 1 =$$
$$= 99 - 113 + 45 - 6 =$$
$$144 - 119 = 25.$$

## 2.2 Основни комбинаторни конфигурации

Измежду многото възможности за създаване на комбинаторни конфигурации, в този раздел ще се спрем на две много често използвани – *наредбата* и *повтарянето* на елементите на образуващото множество. В зависимост от това, използваме или не повтаряне на елементи при построяване на конфигурациите и от това, дали използваме или не наредба, получаваме 4 основни типа комбинаторни конфигурации. За образуващо използваме множеството $A$ с $n > 0$ елемента. За разлика от множествата, ненаредените конфигурации с повтаряне (наричаме ги още *мултимножества*) записваме като поставяме елементите на конфигурацията в правоъгълни скоби.

**Комбинаторни конфигурации с наредба и повтаряне.**

Нека $m$ е цяло число, $m > 0$. Да означим с $\mathcal{K}_{\text{н,п}}(n,m)$ множеството, в което всеки елемент е наредена $m$-торка (или $m$-мерен вектор), с елементи от образуващото множество $A$, като един елемент на $A$ може да участва произволен брой пъти в $m$-торката. Очевидно $\mathcal{K}_{\text{н,п}}(n,m)$ е добре познатото ни $A^m$ – $m$-тата декартова степен на $A$. Затова и

$|\mathcal{K}_{\text{н,п}}(n,m)| = |A^m| = n^m$.

**Комбинаторни конфигурации с наредба, без повтаряне.**

Нека $1 \leq m \leq n$. Да означим с $\mathcal{K}_{\text{н}}(n,m)$ множеството от наредените $m$-торки с елементи от образуващото множество $A$, в които всеки елемент участва точно 1 път. Нека $m \geq 2$. Тогава на първо място в $m$-торката може да се постави кой да е от елементите на $A$, които са $n$ на брой, а на второ място всеки от останалите $n-1$. Така че за първите 2 позиции в $m$-торката, съгласно Принципа на умножението, получаваме $n(n-1)$ възможности. За всяка от тях можем да си изберем трети елемент по $(n-2)$ възможни начина и съгласно Принципа на умножението за първите 3 позиции имаме $n(n-1)(n-2)$ възможности. Разсъждавайки индуктивно, получаваме за броя на конфигурациите с наредба и без повтаряне $|\mathcal{K}_{\text{н}}(n,m)| = n(n-1)\ldots(n-m+1)$. В случай, че $m = 1$ имаме точно $n$ възможности, и тъй като $n = \mathcal{K}_{\text{н}}(n,1)$, формулата е в сила и в този случай.

В математическата литература комбинаторните конфигурации от $m$ елемента, построени от множество с $n$ елемента, $1 \leq m \leq n$, с наредба и без повтаряне се наричат още *вариации* на $n$ елемента от $m$-ти клас и се означават с $V_n^m$.

Интересен частен случай получаваме при $m = n$. Вариациите на $n$ елемента от $n$-ти клас са точно векторите, които съпоставихме в 1.4 на биекциите $f : A \to A$, нарекохме пермутации на елементите на $A$ и обозначихме с $\mathcal{P}_n$. Без ограничение на общността, за образуващо множество считаме множеството $\{1,2,\ldots,n\}$. Съгласно доказаното вече $|\mathcal{P}_n| = |\mathcal{K}_{\text{н}}(n,n)| = n(n-1).....2.1 = n!$. Последният израз четем „$n$ факториел“. При $n = 3$ имаме $|\mathcal{P}_3| = 3! = 1.2.3 = 6$ и това беше броят на елементите на симетричната група от пермутации $\mathcal{S}_3$.

**Комбинаторни конфигурации без наредба, без повтаряне.**

Нека $n > 0$ и $0 \leq m \leq n$. Да означим с $\mathcal{K}(n,m)$ множеството от ненаредените $m$-торки, без повтаряне, от елементи на образуващото множество $A$ с $n$ елемента. Всъщност отсъствието на наредба и повтаряне означава, че разглежданите конфигурации са $m$-елементните подмножества на $A$. Наричаме ги още *комбинации на $n$ елемента от $m$-ти клас* и ги означаваме с $C_n^m$.

За да намерим броя на комбинациите на $n$ елемента от $m$-ти клас при $m \neq 0$, ще използваме Принципа на делението. Ако вместо ненаредените $m$-торки без повтаряне разгледаме наредените, множеството $\mathcal{K}(n,m)$ ще се разшири до $\mathcal{K}'(n,m) = \mathcal{K}_{\text{н}}(n,m)$, при това всяка ненаредена $m$-орка ще се среща в новото множество по толкова начина, по колкото можем



да подредим всичките ѝ елементи, т.е. по $m!$ начина. Следователно

$$|\mathcal{K}(n,m)| = \frac{\mathcal{K}_{\text{н}}(n,m)}{m!} = \frac{n(n-1).....(n-m+1)}{m!} = \frac{n!}{m!(n-m)!}.$$

Така получения израз означаваме с $\binom{n}{m}$, четем „$n$ над $m$“ и наричаме *биномен коефициент*.

При $m = 0$ очевидно имаме само 1 комбинация на $n$ елемента от нулев клас – празното множество. Затова е удобно да определим $0! = 1$. Тогава $\binom{n}{0} = \frac{n!}{0!n!} = 1$ и следователно $\forall m, 0 \leq m \leq n, \mathcal{K}(n,m) = C_n^m = \binom{n}{m}$.

Названието биномен коефициент произлиза от следната класическа

**Теорема 2.2.1 (И. Нютон, „Нютонов бином“)** *За всяко естествено число $n$ и реални $x$ и $y$ е в сила*

$$(x+y)^n = \sum_{m=0}^{n} \binom{n}{m} x^m y^{n-m}.$$

Доказателството на тази теорема може да бъде направено по чисто комбинаторен път. За целта е достатъчно да отбележим, че при повдигане бинома $x+y$ на $n$-та степен всеки едночлен от вида $x^m y^{n-m}$ ще се получи толкова пъти, по колкото начина можем да изберем $m$ множители измежду $n$-те от които вземаме $x$ (от останалите $n-m$ избираме $y$), а това е точно броят на ненаредените конфигурации без повторение $\binom{n}{m}$.

Дефиницията на биномен коефициент остава в сила, когато вместо цялото положително $n$ вземем произволно реално число $r$. Например $\binom{-5}{3} = \frac{(-5).(-6).(-7)}{1.2.3} = -35$ или $\binom{\frac{1}{2}}{2} = \frac{\frac{1}{2}.(-\frac{1}{2})}{1.2} = -\frac{1}{8}$. Всъщност в сила е следното обобщение на Теоремата на Нютон, доказателството на което е извън рамките на курса по дискретна математика:

**Теорема 2.2.2** *За всяко реално число $r$ и реални $x, y, |x/y| < 1$ е в сила*

$$(x+y)^r = \sum_{m \in N} \binom{r}{m} x^m y^{r-m}.$$

Ще използваме Теоремата на Нютон, за да покажем достатъчно общ начин за извличане на комбинаторни тъждества по формален математически път. Полагаме в Нютоновия бином $x = y = 1$ и получаваме

$$2^n = \sum_{m=0}^{n} \binom{n}{m}. \tag{2.4}$$

Както отбелязахме по-горе $\binom{n}{m}$ е точно броят на $m$-елементните подмножества на образуващото $n$-елементно множество. Така равенството (2.4) има следния очевиден комбинаторен смисъл: броят на всички подмножества на множество с $n$ елемента можем да изразим като сума на броя на $m$-елементните му подмножества за $m = 0, 1, \ldots, n$. В комбинаториката едно тъждество може да бъде доказано както по формален път (със заместявания, еквивалентни преобразувания и т.н.), така и по чисто комбинаторен път, чрез преброяване на елементите на подходящо избрана конфигурация по два различни начина. Тази техника е известна като *Принцип на двукратното броене*. За илюстрация ще докажем с двукратно броене някои свойства на биномните коефициенти.

**Теорема 2.2.3** *Нека $n, m \in N, n > m$. Тогава $\binom{n}{m} = \binom{n-1}{m} + \binom{n-1}{m-1}$.*

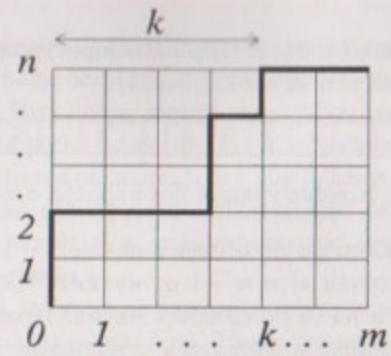
***Доказателство.*** Множеството комбинаторни конфигурации $C_m^n$ с образуващо $A, |A| = n$, можем да разбием на две подмножества: $K_1$ – конфигурациите не съдържащи елемента с номер $n$ и $K_2$ – съдържащите този елемент. Но в $K_1$ са $m$-елементните подмножества на множеството от първите $n-1$ елемента на $A$, т. е. $|K_1| = \binom{n-1}{m}$. От друга страна елементите на $K_2$ се получават като към всяко $(m-1)$-елементно подмножество на множеството от първите $n-1$ елемента на $A$, добавим $n$-тия елемент, т.е. $|K_2| = \binom{n-1}{m-1}$. От Принципа за разбиването $\binom{n}{m} = |\mathcal{K}(n,m)| = |K_1| + |K_2| = \binom{n-1}{m} + \binom{n-1}{m-1}$. □

**Теорема 2.2.4** *Нека $n, m \in N, n > m$. Тогава $\binom{n}{m} = \binom{n}{n-m}$.*

***Доказателство.*** Достатъчно е да забележим, че на всяко подмножество $A'$ на $A$, $|A| = n, |A'| = m$ еднозначно съответства подмножество $A \setminus A'$ с $(n-m)$ елемента и да приложим Принципа на биекцията. □

Друг начин за получаване на тъждества с биномни коефициенти е следният. На Фиг. 2.1 е показана мрежа от квадратчета с $n$ реда и $m$ стълба, $n > m$. Всяко придвижване от долния ляв до горния десен ъгъл на мрежата, като на всяка стъпка се придвижваме от долния ляв ъгъл на квадратчето, в което се намираме, в горния му ляв или долния му десен ъгъл, става с точно $n+m$ стъпки. Броят на различните придвижвания е точно $\binom{n+m}{m} = \binom{n+m}{n}$, тъй като всеки път еднозначно се определя от това кои $m$ от $(n+m)$-те стъпки ще направим надясно или пък кои $n$ от $(n+m)$-те стъпки ще направим нагоре. Сега следвайки Принципа на двукратното броене да намерим същия брой, като въведем параметър на придвижването $k, 0 \leq k \leq m$ – отдалечеността от левия край на






Фигура 2.1: Квадратна мрежа.

мрежата на точката, в която придвижването достига до горния ѝ край. Очевидно е, че за да се окажем в горния край на мрежата на разстояние $k$ от началото, е необходимо и достатъчно да направим стъпка нагоре от горния десен ъгъл на квадратчето с координати $k$ и $n-1$. Следователно броят на придвижванията с параметър $k$ е $\binom{n+k-1}{k}$. Когато $k$ се мени от 0 до $m$ ще получим всички възможни придвижвания. От Принципа на разбиването получаваме

**Теорема 2.2.5** *Нека $n, m \in N$. Тогава $\binom{n+m}{m} = \sum_{k=0}^{m} \binom{n+k-1}{k}$.*

**Комбинаторни конфигурации без наредба, с повтаряне.**

Да означим с $\mathcal{K}_{\Pi}(n,m)$ множеството от конфигурациите с повтаряне, съставени от $m, m \geq 0$ елемента на образуващо множество с $n, n > 0$ елемента (обикновено, без ограничение на общността използваме $I_n$). За да намерим броя на елементите на това множество ще построим биекция между него и множество от комбинаторни конфигурации, елементите на което лесно можем да преброим. Нека $m > 0$. Разглеждаме последователност от $m+n-1$ различими кутии и нека разполагаме с достатъчен брой екземпляри от елемент, който $\notin A$ (за примера по-долу сме избрали $\star$). Нека е дадена произволна ненаредена конфигурация от $m$ елемента с повторение. Започвайки отляво, оставяме празни толкова кутии, колкото пъти се среща елементът 1 (ако не се среща – 0 кутии) и в следващата поставяме $\star$. Пропускаме толкова празни кутии, колкото пъти се среща елементът 2 и в кутията след тях поставяме $\star$ и т.н., докато всички $n-1$ елемента $\star$ попаднат в кутиите. Следователно, на всяка конфигурация без наредба, с повтаряне съпоставихме по единствен начин разполагане на $n-1$ елемента $\star$ в някои $n-1$ от различните $m+n-1$ кутии. Фиг. 2.2

изобразява разполагането, съответно на конфигурацията $[1,3,3,4,4,5]$, като в случая $n=5, m=6, A=\{1,2,3,4,5\}$.

| 1 | ⋆ | ⋆ | 3 | 3 | ⋆ | 4 | 4 | ⋆ | 5 |
|---|---|---|---|---|---|---|---|---|---|

Фигура 2.2: Конфигурации без наредба с повтаряне.

Обратно, да поставим по произволен начин $n-1$ екземпляра на избрания елемент (в случая $\star$) в $n-1$ от кутиите. Останалите $m$ кутии запълваме с елементи на $A$ по следния начин. Празните кутии от началото до първия елемент $\star$ запълваме с елемента 1 (може изобщо да няма такива кутии). Празните кутии между първия и втория елемент $\star$ запълваме с 2, и т. н. Празните кутии от последния, $(n-1)$-ви елемент $\star$ до края запълваме с $n$. Приложена към последователност от 10 кутии, с елемента $\star$ във втората, третата, шестата и деветата кутия, гореописаната процедура дава конфигурацията $[1,3,3,4,4,5]$. Очевидно всяко различно поставяне на $n-1$ елемента $\star$ в $n+m-1$ кутии поражда точно една такава конфигурация. Установеното съответствие е биекция и съгласно Принципа на биекцията, броят на конфигурациите е равен на броя на различните начини, по които можем да изберем място на $n-1$ елемента $\star$ в $m+n-1$ различими кутии, т.е. на броя на $(n-1)$-елементните подмножества на множество с $m+n-1$ елемента. И така

$$|\mathcal{K}_{\Pi}(n,m)| = \binom{m+n-1}{n-1} = \binom{m+n-1}{m}.$$

Ако $m=0$ от формулата получаваме $|\mathcal{K}_{\Pi}(n,m)|=1$ (празното множество е единствената конфигурация) и следователно получената формула е вярна и в този случай.

В края на този раздел ще дадем и формула за броя на пермутациите $\mathcal{P}_n^{n_1,n_2,\ldots,n_k}$ на $n$ неразлични елемента от $k$ вида – $n_1$ от първия вид, $n_2$ от втория, и т.н., $n_k$ – от $k$-тия вид. Ясно е, че произволна пермутация на елементите от даден вид не променя конфигурацията. Затова, прилагайки $k$ пъти Принципа за деление получаваме:

$$|\mathcal{P}_n^{n_1,n_2,\ldots,n_k}| = \frac{n!}{n_1!n_2!\ldots n_k!}.$$

## 2.3 Производящи функции и рекурентни отношения

В този раздел ще разгледаме задачи на изброителната комбинаторика от вида: „дадена“ е безкрайна фамилия $\{\mathcal{K}(n)|n\in N\}$ от крайни множества от комбинаторни конфигурации. Да се намери $a_n=|\mathcal{K}(n)|, \forall n\in N$.



Много често при такива задачи се сблъскваме с трудности при опита да намерим достатъчно просто дефинирана функция $f: N\to N$, такава че $f(n)=a_n, \forall n\in N$. Има доста задачи от подобен род, за които все още не е известен начин за лесно намиране на значенията $a_n$. (Естествено, изключваме буквалното изброяване, което при големи $n$ не е възможно да се извърши за разумно време, дори с помощта на най-мощните съвременни компютри. Изключваме и „псевдоформули“ от вида $a_n=\sum_{i\in\mathcal{K}(n)}1$, повтарящи дефиницията.)

Ще разгледаме техника от анализа, която понякога успешно решава задачата за намиране на $f(n)$. Нека елементите на редицата $\tilde{a}=(a_0,a_1,a_2,\ldots,a_n,\ldots)$ са значенията на функцията $f: N\to Z$. (Разширяваме областта на изменение на $f$ до $Z$, за да можем по-свободно да оперираме със стойностите $a_i$.) Формалният степенен ред

$$\tilde{f}(z)=a_0+a_1z+a_2z^2+\cdots+a_nz^n+\cdots=\sum_{i\in N}a_iz^i$$

наричаме *производяща функция* на редицата $\tilde{a}$, където $z$ е реална променлива. Нека за подходяща околност на нулата редът е сходящ и $\tilde{f}(z)$ е добре дефинирана функция на $z$ в зададената околност. От този факт могат да се получат редица полезни свойства на $f(n)$. Например $a_0=f(0)=\tilde{f}(0)$, а ако с $\tilde{f}^{(r)}(z)$ означим $r$-тата производна на $\tilde{f}(z)$, то $a_n=f(n)=\tilde{f}^{(n)}(0)$.

Например, нека $f(n)=1, \forall i\in N$. Тогава редът $\tilde{f}(z)=1+z+z^2+\cdots+z^n+\cdots$ е сходящ за $|z|<1$, и лесно се проверява, че $\tilde{f}(z)=1/(1-z)$. Като втори пример да разгледаме функцията

$$f(n)=\begin{cases}1 & n=0,1,2\\ 0 & n>2\end{cases}.$$

Сега $\tilde{f}(z)=1+z+z^2$ е сходящ навсякъде и

$$\begin{aligned}\tilde{f}(z) &= 1+z+\cdots+z^n+\cdots-\\ &- z^3-z^4-\cdots-z^{n+3}-\cdots=\\ &= \textstyle\sum_{i\in N}z^i-z^3\sum_{i\in N}z^i=\\ &= (1-z^3)\textstyle\sum_{i\in N}z^i=\frac{1-z^3}{1-z}.\end{aligned}$$

Без да пренебрегваме сходимостта на степенния ред, в по-нататъшните разглеждания няма да се интересуваме от нея и ще гледаме на производящите функции като на формализъм за представяне на съответните редици от цели числа.

***Дефиниция.*** Нека $\tilde{f}(z) = \sum_{i\in N} a_i z^i$ и $\tilde{g}(z) = \sum_{i\in N} b_i z^i$. Тогава:

а) *сума* на $\tilde{f}(z)$ и $\tilde{g}(z)$ е редът $\tilde{h}(z) = \tilde{f}(z) + \tilde{g}(z) = \sum_{i\in N} c_i z^i$, където $c_i = a_i + b_i, i = 0, 1, 2, \ldots$;

б) *произведение* на $\tilde{f}(z)$ и $\tilde{g}(z)$ е редът $\tilde{h}(z) = \tilde{f}(z).\tilde{g}(z) = \sum_{i\in N} c_i z^i$, където $c_i = \sum_{j=0}^{i} a_j b_{i-j}, i = 0, 1, 2, \ldots$.

Получаваме алгебра с две операции, за които лесно се вижда, че са асоциативни и комутативни, а събирането е дистрибутивно по отношение на умножението. Неутрален елемент на събирането е производящата функция $\tilde{0} = 0 + 0z + 0z^2 + \cdots + 0z^n + \cdots$, а на умножението - $\tilde{1} = 1 + 0z + 0z^2 + \cdots + 0z^N + \cdots$. За всяка производяща функция $\tilde{f}(z) = \sum_{i\in N} a_i z^i$ обратен елемент за операцията събиране е $-\tilde{f}(z) = \sum_{i\in N} -a_i z^i$. За същата производяща функция, ако $a_0 \neq 0$, можем да определим обратен елемент за операцията умножение - функцията $\tilde{f}^{-1}(z) = \sum_{i\in N} d_i z^i$, като от дефиниционното равенство $\tilde{f}(z).\tilde{f}^{-1}(z) = 1$ лесно определяме $d_0 = 1/a_0$ и $d_i = -\frac{a_1 d_{i-1} + \cdots + a_i d_0}{a_0}, i = 1, 2, \ldots$. Без ограничение на общността можем да изключим от разглеждане производящите функции от вида $\tilde{f}(z) = 0 + 0z + \ldots + 0z^{k-1} + az^k + \ldots, a \neq 0 = z^k \tilde{g}(z)$, защото изследването им се свежда до изследването на съответната $\tilde{g}(z)$ с ненулев свободен член, т.е. имаща обратна спрямо умножението.

И така множеството $\mathcal{G}[z]$ на производящите функции, заедно с двете операции събиране и умножение е поле и можем да извършваме в него преобразованията, с които сме свикнали в полето на реалните числа.

**Теорема 2.3.1** *В сила са следните връзки между $f(n)$ и $\tilde{f}(z)$:*

$$
\begin{array}{lll}
1.f(n) = f_1(n) + f_2(n) & \Leftrightarrow & \tilde{f}(z) = \tilde{f}_1(z) + \tilde{f}_2(z); \\
2.f(n) = kf_1(n) & \Leftrightarrow & \tilde{f}(z) = k\tilde{f}_1(z); \\
3.f(n) = k^n f_1(n) & \Leftrightarrow & \tilde{f}(z) = \tilde{f}_1(kz); \\
4.f(n) = nf_1(n) & \Leftrightarrow & \tilde{f}(z) = z\tilde{f}_1^{(1)}(z); \\
5.f(n) = k^n & \Leftrightarrow & \tilde{f}(z) = 1/(1 - kz); \\
6.f(n) = \begin{cases} 0 & n < k \\ f_1(n-k) & n \geq k \end{cases} & \Leftrightarrow & \tilde{f}(z) = z^k \tilde{f}_1(z); \\
7.f(n) = f_1(n+k) & \Leftrightarrow & \tilde{f}(z) = \frac{\tilde{f}_1(z) - \sum_{i=0}^{k-1} f(i) z^i}{z^k}.
\end{array}
$$

Доказателствата се получават от дефиницията на производяща функция, затова ги оставяме на читателя за упражнение. Ще демонстрираме възможностите на техниката на производящите функции за да намерим функция $f(n) = a_n$ за редицата $a_0 = 0, a_1 = 1, a_{n+2} = a_{n+1} + a_n, n \geq 0$, наречена *редица на Фибоначи*.



Нека $\tilde{f}(z)$ е производящата функция на редицата на Фибоначи. Дефинираме функциите $f_1(n) = f(n+1)$, $f_2(n) = f(n+2)$ и нека $\tilde{f}_1(z)$ и $\tilde{f}_2(z)$ са съответните им производящи функции. От дефиницията на редицата $f(n+2) - f(n+1) - f(n) = 0$, следователно $f_2(n) - f_1(n) - f(n) = 0$ и от Теорема 2.3.1(1) получаваме $\tilde{f}_2(z) - \tilde{f}_1(z) - \tilde{f}(z) = 0$. От Теорема 2.3.1(7) имаме $\tilde{f}_2(z) = \frac{\tilde{f}(z) - f(0) - f(1)z}{z^2} = \frac{\tilde{f}(z) - z}{z^2}$ и $\tilde{f}_1(z) = \frac{\tilde{f}(z) - f(0)}{z} = \frac{\tilde{f}(z)}{z}$. Следователно $\frac{\tilde{f}(z) - z}{z^2} - \frac{\tilde{f}(z)}{z} - \tilde{f}(z) = 0$ откъдето $\tilde{f}(z)(1 - z - z^2) = z$ и $\tilde{f}(z) = \frac{z}{1 - z - z^2}$.

Сега да намерим $\alpha$ и $\beta$, такива че $1 - z - z^2 = (1 - \alpha z)(1 - \beta z) = 1 - (\alpha + \beta)z + \alpha\beta z^2$. Лесно се вижда, че съответната система от две уравнения (от втора степен) за неизвестните $\alpha$ и $\beta$ има единствено решение $\alpha = (1 + \sqrt{5})/2$ и $\beta = (1 - \sqrt{5})/2$.

По същия начин да намерим $a$ и $b$, такива че $\tilde{f}(z) = \frac{z}{(1-\alpha z)(1-\beta z)} = \frac{a}{1-\alpha z} + \frac{b}{1-\beta z}$. Тук съответната система за $a$ и $b$ е линейна и решението ѝ е $a = 1/\sqrt{5}, b = -1/\sqrt{5}$. Следователно $\tilde{f}(z) = \frac{1}{\sqrt{5}}\left(\frac{1}{1-\alpha z} - \frac{1}{1-\beta z}\right)$. Прилагайки Теорема 2.3.1(5) два пъти, а след това Теорема 2.3.1(1) получаваме $f(n) = \frac{1}{\sqrt{5}}(\alpha^n - \beta^n)$.

Ще обобщим демонстрираната по-горе техника.

***Дефиниция.*** Нека редицата $\tilde{a} = (a_0, a_1, a_2, \ldots)$ е такава, че

$$a_{i+r} = c_1 a_{i+r-1} + c_2 a_{i+r-2} + \cdots + c_r a_i, i = 0, 1, 2, \ldots$$

за някакви константи $c_j, j = 1, 2, \ldots, r$, където $r \in N, r > 0$ и $c_r \neq 0$. Равенството, определящо $a_{i+r}$ като линейна функция на предхождащите $r$ члена на редицата наричаме *линейно рекурентно отношение* от ред $r$.

За да е напълно определена една редица с линейно рекурентно отношение от ред $r$ е необходимо да са зададени първите $r$ члена на тази редица (вж. дефинираната по-горе с линейно рекурентно отношение от ред 2 редица на Фибоначи). Без доказателство ще приведем следната

**Теорема 2.3.2** *Нека редицата $\tilde{a} = (a_0, a_1, a_2, \ldots)$ е зададена с линейното рекурентно отношение*

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_r a_{n-r}, n \geq r$$

*и $\alpha_1, \alpha_2, \ldots, \alpha_s$ са различните комплексни корени на **характеристичното уравнение** $x^r - c_1 x^{r-1} - \cdots - c_{r-1}x - c_r = 0$, като $\alpha_i$ е с кратност $k_i, k_1 + k_2 + \cdots + k_s = r$. Тогава $a_n = \sum_{i=1}^{s} P_i(n)\alpha_i^n$, където $P_i(n)$ е полином на $n$ от степен $< k_i$. Полиномите $P_i$ имат общо $r$ коефициента, които се определят еднозначно от първите $r$ члена на редицата $\tilde{a}$.*

Рекурентните отношения, които разгледахме, се наричат *хомогенни*. Същата техника за разрешаване е приложима и към *нехомогенните линейни рекурентни отношения* от вида

$$a_n = c_1a_{n-1} + c_2a_{n-2} + \cdots + c_ra_{n-r} + b_1^nQ_1(n) + \cdots + b_m^nQ_m(n), n \geq r,$$

където $b_j$ са различни една от друга константи, а $Q_j(n)$ – полиноми на $n$ от степен $d_j - 1, j = 1, 2, \ldots, m$. Достатъчно е да означим $\alpha_{r+j} = b_j, j = 1, 2, \ldots, m$ и да гледаме на него като на $d_j$ кратен корен на характеристичното уравнение. Тогава $a_n = \sum_{i=1}^{r+m} P_i(n)\alpha_i^n$, като за намирането на коефициентите на полиномите $P_i(n)$, освен зададените $r$, ще трябва да бъдат изчислени с помощта на рекурентното отношение още $d_1 + d_2 + \ldots + d_m$ члена на редицата.

За пример да разгледаме редицата, зададена с $H_0 = 0, H_n = 2.H_{n-1} + 1, n \geq 1$ (с $H_n$ е означен броят на преместванията, които прави класическият рекурсивен алгоритъм за решаване на задачата за Ханойските кули, когато трябва да премести $n$ кръга). От хомогенната част $H_n - 2.H_{n-1} = 0$ получаваме характеристичното уравнение $x - 2 = 0$ с еднократен корен 2. Нехомогенната част представяме във вида $b^nq(n)$, където $b = 1$, а $q(n) = 1$ е полином на $n$ от степен 0. Това добавя един корен 1 към корена на характеристичното уравнение. Следователно общият член на редицата търсим във вида $H_n = x_1.2^n + x_2.1^n = x_1.2^n + x_2$, при което ще трябва да намерим двата неизвестни коефициента $x_1$ и $x_2$. За целта пресмятаме още един член на редицата: $H_1 = 2.H_0 + 1 = 2.0 + 1 = 1$. Така, заместващи $n$ с 0 и 1 в равенството за $H_n$, получаваме системата линейни уравнения

$$\left|\begin{array}{lcl} x_1 + x_2 & = & 0 \\ 2x_1 + x_2 & = & 1 \end{array}\right.$$

с корени $x_1 = 1, x_2 = -1$. Следователно $H_n = 2^n - 1$.

## 2.4 Комбинаторни конфигурации в наредени множества

Комбинаторните конфигурации в крайни наредени множества, които ще разгледаме, имат пряка връзка с важни оптимизационни задачи на дискретната математика, но представляват и самостоятелен интерес.

***Дефиниция.*** Нека $R \subseteq A \times A$ е частична наредба в крайното множество $A$. Антиверигата $S$ на $R$ наричаме *максимална*, ако $|S| \geq |S'|$ за всяка друга антиверига $S'$ на $R$. Множеството $\mathcal{C} = \{C_1, C_2, \ldots, C_r\}$ от вериги на $R$ наричаме *верижно разбиване* на $A$, ако всеки елемент



$a \in A$ се среща в точно една верига от $\mathcal{C}$. Верижното разбиване $\mathcal{C}$ на $A$ е *минимално*, ако $|\mathcal{C}| \leq |\mathcal{C}'|$ за всяко друго верижно разбиване $\mathcal{C}'$ на $A$.

Важна връзка между максималните антивериги и минималните верижни разбивания на крайна частична наредба се дава от следната

**Теорема 2.4.1 (Р. Дилуорт)** *Нека $R$ е частична наредба в крайното множество $A$, $\mathcal{C}$ е минимално верижно разбиване на $A$, а $S$ максимална антиверига на $R$. Тогава $|S| = |\mathcal{C}|$.*

***Доказателство.*** Нека $M = |S|$, а $m = |\mathcal{C}|$. Очевидно $M \leq m$, защото иначе ще се окаже, че някои два елемента на $S$ участват в една верига на $\mathcal{C}$, което е противоречие. С индукция по броя на елементите на $A$ ще покажем, че не е възможно $M < m$. Действително, ако $|A| = 1$, тогава $M = m = 1$ и твърдението е в сила. Затова да допуснем, че твърдението е вярно за всяко частично наредено множество $A$, $1 \leq |A| < n$. За случая $|A| = n$ имаме две възможности:

а) $R = \{(a_i, a_i) | a_i \in A\}$. В този частен случай всички елементи на $A$ са едновременно максимални и минимални. Сега $A = \{a_1, a_2, \ldots, a_n\}$ е максимална антиверига, $\mathcal{A} = \{\{a_1\}, \{a_2\}, \ldots, \{a_n\}\}$ е минимално верижно разбиване и $M = m = n$.

б) $R \neq \{(a_i, a_i) | a_i \in A\}$. За максималните антивериги на $R$ има две възможности:

– съществува максимална антиверига $S = \{a_{i_1}, a_{i_2}, \ldots, a_{i_M}\}$, която не съдържа нито всички минимални, нито всички максимални елементи на $R$. В такъв случай дефинираме множествата $A^+ = \{a | \exists a_{i_j} \in S, a \geq a_{i_j}\}$ и $A^- = \{a | \exists a_{i_j} \in S, a \leq a_{i_j}\}$ и означаваме с $R^+$ и $R^-$ рестрикциите на $R$ върху $A^+$ и $A^-$, които са частични наредби с по-малко елементи от $R$. Очевидно $S \subseteq A^+$ и $S \subseteq A^-$. Тъй като не всички минимални и максимални елементи са в $S$, то $A^+ \neq A$ и $A^- \neq A$. Да допуснем, че съществува $a \in A, a \notin A^+ \cup A^-$. Тогава стигаме до противоречието, че $S \cup \{a\}$ е антиверига с повече елементи от $S$. Следователно $A = A^+ \cup A^-$. Ако допуснем, че съществува $a \notin S, a \in A^+ \cap A^-$, тогава $\exists j, k \in I_M$ така, че $a_{i_j} \leq a \leq a_{i_k}$, т.е. $(a_{i_j}, a_{i_k})$ е верига и отново получаваме противоречие. Следователно $S = A^+ \cap A^-$, т.е. $R^+$ и $R^-$ имат максимална антиверига с $M$ елемента и съгласно индукционното предположение – и минимални верижни разбивания с $M$ елемента. Всяка верига в разбиването на $A^-$ завършва в елемент на $S$, а всяка верига в разбиването на $A^+$ започва в елемент на $S$. Съчленяваме всяка такава двойка вериги в общата им точка и получаваме, че $A$ има верижно разбиване с $M$ елемента, т.е. $m = M$.

= всяка максимална антиверига $S$ съдържа или всички максимални или всички минимални елементи на $R$. Тъй като $R$ не е рефлексивно затваряне на празната релация, съществува поне един минимален $a$ и поне един максимален $b$, такива, че $a < b$. Рестрикцията на $R$ върху $A \setminus \{a, b\}$ е частична наредба с по-малко елементи от $R$, има максимална антиверига с $M-1$ елемента и съгласно индукционното предположение притежава минимално разбиване $\mathcal{C}'$ с $M-1$ елемента. Но $(a, b)$ е верига, следователно $\mathcal{C}' \cup \{(a, b)\}$ е разбиване на $A$ с $M$ елемента и отново $m = M$. □

Аналогично можем да определим *максимална верига* и *минимално разбиване на антивериги* на частична наредба. Неочаквано просто се доказва „двойнственото" твърдение:

**Теорема 2.4.2** *Нека $R$ е частична наредба в крайното множество $A$, $S$ е минимално разбиване на $A$ в антивериги, а $C$ максимална верига на $R$. Тогава $|C| = |S|$.*

***Доказателство.*** И тук да означим $M = |C|$, а $m = |S|$. Очевидно $M \leq m$, иначе ще се окаже, че някои два елемента на $C$ участват в една антиверига на $S$, което е противоречие. Да означим с $l(x)$ броя на елементите на максималната верига с край в $x \in A$. Ако $l(a) = l(b), a \neq b$, то $a$ и $b$ са несравними. Действително, да допуснем, че $a < b$ и $(a_1, \ldots, a_{l(a)-1}, a)$ е максимална верига с край в $a$. Тогава $(a_1, \ldots, a_{l(a)-1}, a, \ldots, b)$ е верига с край в $b$ и дължина $l, l > l(a)$ и $l \leq l(b)$, което е противоречие. Следователно $A_i = \{a | l(a) = i\}, i \in I_M$, са антивериги, а $\{A_i | i \in I_M\}$ е разбиване на $A$ в $M$ антивериги. Т.е. $m \leq M$ и следователно $m = M$. □

Не е лесно да се намери броят на елементите на максималната антиверига за произволен клас частични наредби. Тук ще покажем как това може да стане за наредбата на подмножествата на дадено множество.

**Теорема 2.4.3 (Е. Шпернер)** *Броят на елементите на всяка антиверига на частичната наредба $R_{\subseteq A}, |A| = n$, е по-малък или равен на $\max\limits_k \binom{n}{k}$.*

***Доказателство.*** Нека $(a_{i_1}, a_{i_2}, \ldots, a_{i_n})$ е произволна пермутация на елементите на множеството $A$. Тя поражда максималната верига на $R_{\subseteq A}$

$$C = \{\emptyset, \{a_{i_1}\}, \{a_{i_1}, a_{i_2}\}, \ldots, \{a_{i_1}, a_{i_2}, \ldots, a_{i_n}\}\}.$$

Обратно, всяка максимална верига ще бъде последователност от подмножества, всяко следващо с един елемент повече и естествено ще задава



пермутация на елементите на $A$. Затова броят на различните максимални вериги на $R_{\subseteq A}$ е $n!$. Нека $S = \{A_1, A_2, \ldots, A_m\}$ е антиверига. Всяка максимална верига съдържа не повече от едно от множествата на $S$, например $A_i$. Но броят на различните максимални вериги, в които $A_i$ участва е точно $|A_i|!(n - |A_i|)!$ (вж. по-горе за максималния брой вериги). Следователно $\sum_{i=1}^{m} |A_i|!(n - |A_i|)! \leq n!$ или

$$\sum_{i=1}^{m} \frac{1}{\binom{n}{|A_i|}} \leq 1.$$

Да означим с $d_k$ броя на $k$-елементните множества в $S$. Тогава последното неравенство добива вида

$$\sum_{k=0}^{n} \frac{d_k}{\binom{n}{k}} \leq 1 \text{ или } \frac{m}{\max \binom{n}{k}} \leq 1.$$

Следователно $m \leq \max \binom{n}{k}$. □

Оставяме на читателя да съобрази, че максимумът от Теоремата на Шпернер се достига при $k = n/2$ за четно $n$. При нечетно $n$ имаме максимални антивериги при $k = \lfloor \frac{n}{2} \rfloor$ и $k = \lceil \frac{n}{2} \rceil$. И така, фамилиите от всички $k$-елементни подмножества на множество $A$ с $n$ елемента (при определените по-горе $k$) са максимални антивериги на релацията $R_{\subseteq A}$. Наричаме ги *Шпернерови фамилии* от подмножества.

Следващото понятие ще въведем само за частичната наредба на подмножествата на зададено крайно множество.

***Дефиниция.*** *Матроид* наричаме двойката $(E, \mathcal{M})$, в която $E$ е крайно множество, а фамилията $\mathcal{M} \subseteq 2^E$ е такава, че:

а) $\emptyset \in \mathcal{M}$;

б) ако $X \in \mathcal{M}$ и $Y \subseteq X$, то $Y \in \mathcal{M}$;

в) $\forall X, Y \in \mathcal{M}, |X| = |Y| + 1$ съществува $x \in X \setminus Y$, такова че $Y \cup \{x\} \in \mathcal{M}$.

Ако не е изпълнено условието (в) на дефиницията, двойката $(E, \mathcal{M})$ наричаме *независима фамилия подмножества*. За елементите на $\mathcal{M}$ казваме че са *независими*, а за елементите на $2^E \setminus \mathcal{M}$, че са *зависими* множества на матроида $(E, \mathcal{M})$. Множеството $E$ наричаме *носител* на матроида.

Класически пример на матроид получаваме, когато за носител вземем произволно множество $V$ от вектори на линейно векторно пространство и обявим за независимо всяко множество $V' \subseteq V$, векторите на което са линейно-независими. Оставяме на читателя (упражнение по-скоро по линейна алгебра, отколкото по комбинаторика) да покаже, че

посочената конструкция е матроид. Ето и един по-близък до нашите разглеждания пример. Да вземем за носител произволно множество $E$ и нека $\mathcal{M}_k = \{S | S \subseteq E, |S| \leq k\}$. Полученият матроид $E_{k,n} = (E, \mathcal{M}_k)$ наричаме *еднороден*. Очевидно са изпълнени условията (а), (б) и (в) на дефиницията.

***Дефиниция.*** Максималните (по включване) независими множества на матроида $(E, \mathcal{M})$ наричаме *бази*, а минималните (по включване) зависими – *цикли* на матроида.

**Теорема 2.4.4** *Независимата фамилия от подмножества $(E, \mathcal{M})$ е матроид т.с.т.к. всички максимални независими подмножества на $A \subseteq E$ са с равен брой елементи.*

***Доказателство.*** 1. Нека $(E, \mathcal{M})$ е матроид, т.е в сила е свойството (в) на дефиницията. Да допуснем, че съществуват две максимални (по включване) независими подмножества $X$ и $Y$ на $A \subseteq E$ и $|X| > |Y|$. Избираме $X' \subseteq X$ такова, че $|X'| = |Y| + 1$. Тогава от (в) следва, че $\exists x \in X', x \notin Y$ и $Y \cup \{x\}$ е независимо. Това противоречи на максималността на $Y$. Следователно всички максимални независими множества са с равен брой елементи.

2. Нека независимата фамилия $(E, \mathcal{M})$ удовлетворява допълнителното условие на теоремата. Нека независимите $X$ и $Y$ са такива, че $|X| = |Y| + 1$. Образуваме $A = X \cup Y$. $Y$ не е максимално (по включване) в $A$. Следователно съществува независимо $Y' = Y \cup \{x\} \subseteq A$, което го съдържа. Но $x \notin Y$, следователно $x \in X \setminus Y$ и значи е в сила свойството (в) на Дефиницията и $(E, \mathcal{M})$ е матроид. □

Теорема 2.4.4 дава алтернативна дефиниция на понятието матроид посредством максималните независими подмножества на дадено множество. В случая, когато $A = E$ получаваме, че всички бази на един матроид са с еднакъв брой елементи. Без доказателство ще посочим характеризация на матроида посредством циклите му, която също може да бъде използвана за алтернативна дефиниция.

**Теорема 2.4.5** *Фамилията $\mathcal{C} = \{C_1, C_2, \ldots, C_r\}$ от подмножества на множеството $E$ е фамилия от цикли на матроид т.с.т.к.*

*а) $\mathcal{C}$ е антиверига на релацията $R_{\subseteq E}$;*

*б) $\forall C_i, C_j \in \mathcal{C}, C_i \neq C_j$ и $\forall x \in C_i \cap C_j$, $\exists C_k \in \mathcal{C}$ такова, че $C_k \subseteq C_i \cup C_j \setminus \{x\}$.*



## 2.5 Крайни геометрии и блок-дизайни

В някои източници разгледаните в предния раздел матроиди са наречени *предгеометрии*. В настоящия раздел ще се запознаем с комбинаторни конфигурации, които носят името геометрии и в този смисъл са следваща стъпка в разглеждането на комбинаторни конфигурации. От друга страна крайните геометрии са естествени дискретни аналози на по-популярните непрекъснати геометрии (например, геометрията на Евклид, и получената от нея, чрез добавяне на идеални точки и идеална права, проективна геометрия).

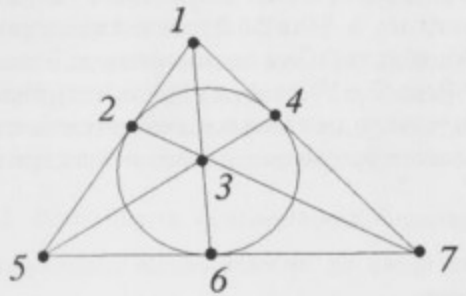
***Дефиниция.*** Двойката $(A, \mathcal{L})$, където $A$ е множество от *точки*, а $\mathcal{L} \subseteq 2^A$ – фамилия от *прави* наричаме *проективна равнина* или *проективна геометрия*, ако са в сила:

а) $\forall P_1, P_2 \in A$, $\exists$ единствена $l \in \mathcal{L}$ такава, че $P_1 \in l, P_2 \in l$;

б) $\forall l_1, l_2 \in \mathcal{L}$, $\exists P \in A$, такава че $P \in l_1, P \in l_2$;

в) съществуват четири точки, никои три от които не принадлежат на една и съща права (същински четириъгълник).

Множеството от точки $A$ наричаме *носител* на проективната равнина. Ако $P \in l$ казваме, че $P$ *лежи* на $l$, а $l$ *минава* през $P$. Тъй като всяка права се определя еднозначно от произволни свои две точки, ще означаваме с $P_1P_2$ правата, минаваща през точките $P_1$ и $P_2$.




Фигура 2.3: Геометрия на Фано

На Фиг. 2.3 е показана проективна равнина с $P = I_7$ и

$$\mathcal{L} = \{\{1,2,5\}, \{1,4,7\}, \{2,3,7\}, \{1,3,6\}, \{3,4,5\}, \{5,6,7\}, \{2,4,6\}\},$$

наричана *проективна равнина (геометрия) на Фано*.

Директно от дефиницията получаваме следните свойства:

**Лема 2.5.1** *Всеки две различни прави на проективната геометрия минават през точно една обща точка.*

***Доказателство.*** Наличието на поне една обща точка $P$ е следствие от условието (б) на дефиницията. Ако допуснем, че двете прави имат още една обща точка $Q$, ще се окаже, че две различни прави минават през точките $P$ и $Q$, което противоречи на условието (а) на дефиницията. □

**Лема 2.5.2** *В проективната равнина съществуват четири прави, никои три от които нямат обща точка.*

***Доказателство.*** Нека точките $P_1, P_2, P_3$ и $P_4$ образуват същински четириъгълник. Тогава правите $P_1P_2$, $P_2P_3$, $P_3P_4$ и $P_4P_1$ удовлетворяват условието на лемата. Наистина, ако допуснем (без ограничаване на общността), че първите три от тях имат обща точка $P$, ще стигнем до противоречието, че през точките $P$ и $P_2$ (както и през точките $P$ и $P_3$) минава повече от една права. □

И тук може да се забележи явление, подобно на срещаното при решетките, което ще формулираме като **Принцип за двойственост (при проективните геометрии)**: Ако в твърдение за проективна геометрия разменим точките с правите (както и релацията „лежи на“ с „минава през“) ще получим ново твърдение, *двойствено* на даденото, със същата вярност.

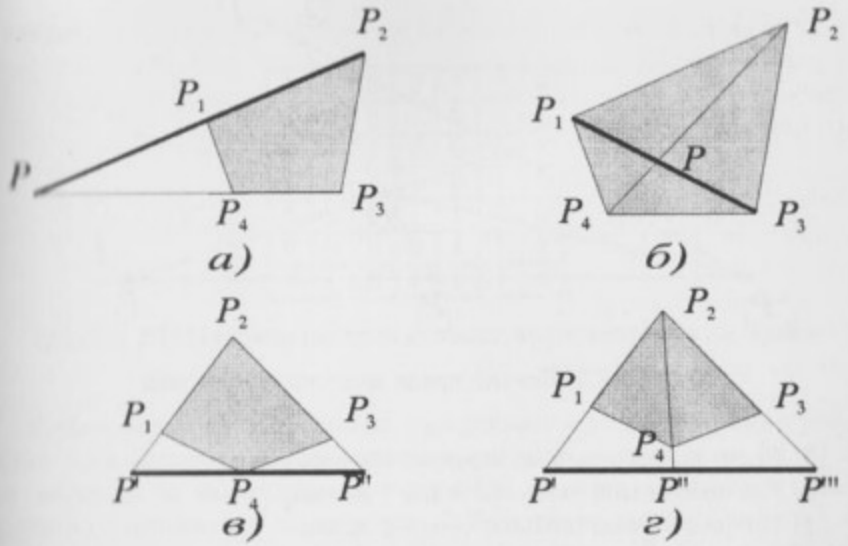
Доказаната по-горе Лема 2.5.1 е двойствено твърдение на изискването (а) на Дефиницията, а Лема 2.5.2 е двойствено твърдение на изискването (в) на Дефиницията. Сега лесно можем да обосновем Принципа за двойственост. Нека $\mathcal{T}$ и $\mathcal{T}'$ са двойствени твърдения. За доказване на $\mathcal{T}'$ ще бъде достатъчно да следваме доказателството на $\mathcal{T}$, като на всяка стъпка прилагаме аргумент – двойствен на приложения в доказателството на $\mathcal{T}'$.

**Лема 2.5.3** *Всяка права на проективната геометрия $(A, \mathcal{L})$ минава през поне три точки.*

***Доказателство.*** Нека $P_1P_2P_3P_4$ е същински четириъгълник. Заради свойствата на четириъгълника, произволна права може да има с него 0, 1 или 2 общи точки.

а) Нека правата $l$ и четириъгълникът $P_1P_2P_3P_4$ имат две общи точки. Ако те са съседни в четириъгълника, например $P_1$ и $P_2$, тогава пресечната точка $P$ на правите $P_1P_2$ и $P_3P_4$ е трета точка за правата $l$ (вж. Фиг. 2.4.а). Ако не са съседни, например $P_1$ и $P_3$, тогава пресечната точка $P$ на правите $P_1P_3$ и $P_2P_4$ е трета точка за правата $l$ (вж. Фиг. 2.4.б).






Фигура 2.4: Всяка права има поне три точки.

б) Нека правата $l$ и четириъгълникът $P_1P_2P_3P_4$ имат една обща точка $P_4$. Ако правите $P_1P_2$ и $P_2P_3$ пресичат $l$ в точки $P'$ и $P''$, това са втора и трета точка за правата $l$ (вж. Фиг. 2.4.в).

в) Нека правата $l$ и четириъгълникът $P_1P_2P_3P_4$ нямат общи точки. Тогава правите $P_2P_1$, $P_2P_4$ и $P_2P_3$ пресичат $l$ в точки $P'$, $P''$ и $P'''$, които са три различни точки за правата $l$ (вж. Фиг. 2.4.г). □

От Принципа за двойственост получаваме

**Следствие 2.5.1** *Всяка точка на проективна геометрия лежи на поне три прави.*

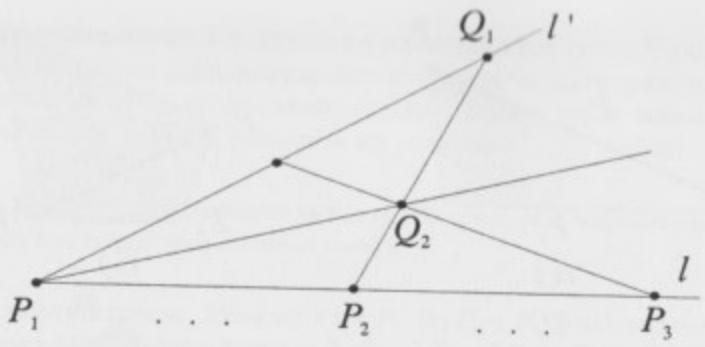
Всичко, което казахме до тук, се отнасяше до произволна проективна равнина. Сега към условията на Дефиницията да добавим:

г) съществува права с $n+1$ точки, където $n \geq 2$ е фиксирано естествено число.

Изискването $n \geq 2$ е задължително, защото правите в проективните равнини трябва да имат поне 3 точки.

**Теорема 2.5.1** *За проективна равнина, удовлетворяваща условието (г) са в сила:*

*1) Всяка права минава през точно $n+1$ точки;*




Фигура 2.5: Всички прави имат по $n+1$ точки

*2) Всяка точка лежи на точно $n+1$ прави;*
*3) Равнината има точно $n^2+n+1$ точки;*
*4) Равнината има точно $n^2+n+1$ прави.*

***Доказателство.*** Тъй като (2) е двойнствено на (1), а (4) на (3), ще докажем само (1) и (3).

1) Нека $l = \{P_1, P_2, \ldots, P_{n+1}\}$ е права с $n+1$ точки и нека $l'$ е произволна друга различна права. Нека (без ограничение на общността) $P_1$ е единствената обща точка на $l$ и $l'$ (вж. Фиг 2.5). Нека $l'$ минава през $Q_1 \neq P_1$ и нека $P_i \neq P_1$. Правата $P_iQ_1$ съдържа поне една точка $Q_2$, различна от $P_i$ и $Q_1$. Да означим с $\mathcal{L}'$ множеството от $n+1$ прави, свързващи $Q_2$ с точките на $l$. Не съществува права през $Q_2$, която да не е в $\mathcal{L}'$, защото в противен случай тя или няма обща точка с $l$, или ще бъде втора права, минаваща през $Q_2$ и някоя от точките на $l$. Следователно $l'$ съдържа точно $n+1$ точки – пресечните ѝ точки с правите от $\mathcal{L}'$.

3) Всяка точка на равнината лежи на права от $\mathcal{L}'$ и тъй като на всяка от тези прави ($n+1$ на брой) има точно $n$ различни от $Q_2$ точки, то всички точки на равнината са $(n+1)n+1 = n^2+n+1$. □

***Дефиниция.*** Проективна геометрия, всяка права на която съдържа точно $n+1$ точки наричаме *крайна проективна равнина от ред $n$* .

Геометрията на Фано е крайна проективна равнина от ред 2 – най-малката крайна проективна равнина. Съществуват процедури за конструиране на крайни проективни геометрии за безкраен брой различни редове. За съжаление, няма процедури за конструиране на крайна проективна геометрия от зададен ред. Нещо повече, известни са естествени числа $n$ (най-малкото от които е 6), за които не съществуват геометрии от ред $n$, а за други въпросът за съществуване на съответната геометрия



е открит.

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |

Таблица 2.1: Матрица на инцидентност на геометрията на Фано

Удобен начин за представяне на крайните проективни геометрии, както и на произволни комбинаторни конфигурации от вида $(A, \mathcal{L})$ е т.н. *матрица на инцидентност* – двумерна матрица с толкова стълба, колкото са елементите на носителя $A$ (точките на конфигурацията) и толкова реда, колкото са елементите на фамилията $\mathcal{L}$. Елементите на матрицата на инцидентност са от множеството $\{0,1\}$ и задават конфигурацията по следния начин: елементът в ред $i$ и стълб $j$ на матрицата на инцидентност е 1, ако множеството $l_i \in \mathcal{L}$ съдържа елемента $a_j \in A$ и 0 – в противен случай. В Таблица 2.1 е показана матрицата на инцидентност на крайната проективна равнина на Фано.

Ще завършим разглеждането на комбинаторни конфигурации със следното понятие:

***Дефиниция.*** Двойката $(A, \mathcal{B})$, където $A$ е крайно множество с $v$ елемента – *точки*, а фамилията $\mathcal{B} = \{B_1, B_2, \ldots, B_b\}$ от $k$-елементни подмножества на $A$ – *блокове* – е такава, че всеки $t$ елемента на $A$ се срещат едновременно в точно $\lambda$ блока от $\mathcal{B}$, наричаме $t$-$(v,k,\lambda)$ *дизайн*.

В горната дефиниция и по-долу в текста, когато казваме, че елементите на едно множество $X$ се срещат в $Y$, това означава, че $X \subseteq Y$.

За пример да разгледаме крайната проективна равнина от ред 2. За точки на дизайн да изберем точките на равнината, а за блокове – правите на равнината. Тъй като всеки две точки на равнината лежат точно на една права, то крайната проективна равнина от ред 2 е 2-(7,3,1) дизайн. Въобще, крайна проективна равнина от ред $n$ е 2-$(n^2+n+1, n+1, 1)$ дизайн. По същия начин както при крайните проективни равнини въвеждаме *матрица на инцидентност на дизайн*.

***Дефиниция.*** Дизайните с $t = 2$ наричаме $(v,k,\lambda)$ *блок-дизайни*, а дизайните с $\lambda = 1$ – *Щайнерови системи*. Щайнеровата система, която е $(v,k,1)$ *дизайн означаваме с* $S(t,k,v)$.

Крайната проективна равнина от ред $n$ е $(n^2+n+1, n+1, 1)$ блок-дизайн и Щайнерова система $S(2, n+1, n^2+n+1)$.

Ето две важни свойства на блок-дизайните.

**Теорема 2.5.2** *Ако $(A, \mathcal{B})$ е $(v, k, \lambda)$ блок-дизайн, то всяка точка от $A$ се среща в точно $r = \lambda(v-1)/(k-1)$ блока от $\mathcal{B}$.*

***Доказателство.*** Нека $a_i \in A$ и $r_i$ е броят на блоковете, в които $a_i$ участва. Да намерим по два различни начина броя на двуелементните подмножества на блокове, в които участва $a_i$. Във всеки от $r_i$-те блока $a_i$ образува по $k-1$ такива подмножества с останалите елементи на блока, т.е. броят им е $r_i(k-1)$. От друга страна $a_i$ се среща заедно с всеки от останалите $v-1$ елемента в точно $\lambda$ блока, т.е. за същия брой получаваме $\lambda(v-1)$. Следователно $r_i(k-1) = \lambda(v-1)$, т.е. $r_i = \lambda(v-1)/(k-1)$ не зависи от избора на $a_i$ и всеки елемент участва в точно $\lambda(v-1)/(k-1)$блока. □

**Теорема 2.5.3** *Ако $(A, \mathcal{B})$ е $(v, k, \lambda)$ блок-дизайн, броят на блоковете на дизайна е $b = vr/k$.*

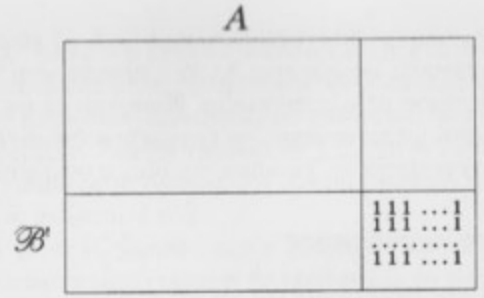
***Доказателство.*** Нека преброим по два различни начина общия брой участия на елементи на $A$ в блокове на $\mathcal{B}$. От предната Теорема всеки от $v$-те елемента участва в точно $r$ блока, т.е. $vr$ участия. От друга страна всеки от блоковете – $b$ на брой – съдържа по $k$ елемента. От тук $rv = kb$ или $b = rv/k$. □

В общия случай имаме следната

**Теорема 2.5.4** *За произволни $i$ точки на $t$-$(v, k, \lambda)$ дизайн, $0 \le i \le t$, броят на блоковете, в които тези точки се срещат едновременно, не зависи от избора на точките и се определя от $\lambda_i = \lambda\binom{v-i}{t-i}/\binom{k-i}{t-i}$. ($\lambda_0$ е броят $b$ на всички блокове на дизайна, тъй като празното множество от точки се среща във всеки блок.)*

***Доказателство.*** Твърдението ще докажем с индукция по $i$. При $i = t$ имаме по дефиниция, че всеки $t$ точки от $A$ се срещат точно в $\lambda$ блока от $\mathcal{B}$ на дизайна $(A, \mathcal{B})$. От формулата пресмятаме $\lambda_t = \lambda\binom{v-t}{0}/\binom{k-t}{0} = \lambda$ и твърдението е в сила. Да допуснем, че произволни $i+1$ точки, $0 \le i < n$, се срещат в $\lambda_{i+1} = \lambda\binom{v-i-1}{t-i-1}/\binom{k-i-1}{t-i-1}$ блока. Нека $p_1, p_2, \ldots, p_i$ са произволни $i$ точки на дизайна. Означаваме $A' = A \setminus \{p_1, p_2, \ldots, p_i\}$ и нека $\mathcal{B}'$ е подмножеството на $\mathcal{B}$ от тези блокове, които съдържат едновременно точките $p_1, p_2, \ldots, p_i$. Фиг. 2.6 отразява схематично разположението на $A'$ и $\mathcal{B}'$ в матрицата на инцидентност на дизайна.






Фигура 2.6: Теорема 2.5.4.

Да разгледаме долната лява част на матрицата. Да означим с $h_{B,q}$ елемента в реда на $B$ и стълба на $q$. Да намерим по два различни начина броя на единиците в тази част на матрицата. Получаваме

$$\sum_{q \in A'} \sum_{B \in \mathcal{B}'} h_{B,q} = \sum_{B \in \mathcal{B}'} \sum_{q \in A'} h_{B,q}.$$

Тъй като всеки блок на $\mathcal{B}'$ съдържа $p_1, p_2, \ldots, p_i$, то $\sum_{q \in A'} h_{B,q} = k - i$ и следователно $\sum_{B \in \mathcal{B}'} \sum_{q \in A'} h_{B,q} = |\mathcal{B}'|(k-i)$. Съгласно индукционното предположение всяка точка $q \in A'$ принадлежи на точно $\lambda_{i+1}$ блока заедно с $p_1, p_2, \ldots, p_i$, т.е. $\sum_{B \in \mathcal{B}'} h_{B,q} = \lambda_{i+1}$ и следователно $\sum_{q \in A'} \sum_{B \in \mathcal{B}'} h_{B,q} = |A'|\lambda_{i+1} = (v-i)\lambda_{i+1}$. И така $(v-i)\lambda_{i+1} = |\mathcal{B}'|(k-i)$, $|\mathcal{B}'| = \lambda_{i+1}\frac{v-i}{k-i}$ и $|\mathcal{B}'|$ не зависи от избора на точките. Означаваме го с $\lambda_i$. Сега

$$\begin{aligned} \lambda_i &= \frac{v-i}{k-i}\lambda_{i+1} = \lambda\frac{v-i}{k-i}\frac{\binom{v-i-1}{t-i-1}}{\binom{k-i-1}{t-i-1}} = \\ &= \lambda\frac{\frac{v-i}{t-i}\binom{v-i-1}{t-i-1}}{\frac{k-i}{t-i}\binom{k-i-1}{t-i-1}} = \lambda\frac{\binom{v-i}{t-i}}{\binom{k-i}{t-i}}. \square \end{aligned}$$

Горната теорема показва, че всеки $t$-$(v, k, \lambda)$ дизайн е също $i$-$(v, k, \lambda_i)$ дизайн, $i < t$. Освен това, изведохме важното свойство:

$$(v-i)\lambda_{i+1} = (k-i)\lambda_i.$$

от което можем да получим редица следствия. Като знаем, че $b = \lambda_0$, за броя на блоковете на дизайн с произволно $t$ получаваме $b = \lambda_0 = \lambda\binom{v}{t}/\binom{k}{t}$. Броят $r$ пък на блоковете, в които се среща коя да е точка е $r = \lambda_1 = k\lambda_0/v = kb/v$, което е твърдението на Теорема 2.5.3. С други думи $r$ не зависи от $t$.

Много важно е, че за съществуването на $t$-$(v,k,\lambda)$ дизайн е необходимо изразът от теоремата, определящ $\lambda_i$, да е целочислен $\forall i, 0 \leq i \leq t$. За съжаление това условие не е достатъчно. Известни са многобройни конструкции на дизайни, включително на безкрайни фамилии, но за някои параметри съществуването на дизайни все още е открит въпрос.

## 2.6 Дискретна вероятност

Съществуват процеси от реалния живот, които се състоят в многократно повтаряне на някакво действие, като след завършването му се получава някакъв резултат – елемент на крайно множество. Кой от възможните резултати ще се получи не е известно предварително, но когато се експериментира голям брой пъти могат да се забележат някои закономерности – например, всички резултати се случват приблизително равен брой пъти, или пък едни резултати се срещат много по-често от други.

Типични примери са така наречените „игри на щастието". При многократното хвърляне на един шестостенен зар се забелязва, че броят на появяванията на числата 1, 2, 3, 4, 5 и 6, изписани по стените му е приблизително еднакъв. Ако хвърляме два зара и като резултат вземем сумата на получените две числа, тогава забелязваме, че сумите 2 и 12 се срещат доста по-рядко от сумата 7.

Като втори пример да разгледаме произволен текст на български език и да пресметнем броя на появяванията на всяка буква в текста. Елементарно е да установим, че гласните *а, е, и, о* се срещат много по-често от някои съгласни – *ц, ч, ш, щ*, например.

***Дефиниция.*** Нека при $n$-кратно провеждане на експеримент с $k$ възможни изхода – елементите на $I_k = \{1, 2, \ldots, k\}$ – сме получили $n_i$ пъти резултата $i \in I_k$. Частното $n_i/n$ наричаме *честота* на резултата $i$ при този експеримент.

Тъй като $\sum_{i=1}^{k} n_i = n$, то $\sum_{i=1}^{k} n_i/n = 1$ и $\forall i \in I_k$, $0 \leq n_i/n \leq 1$. При хвърляне на един зар достатъчно много пъти, ще получим за всяко число от 1 до 6 честота близка до 1/6.

Естествено е, че при различните експерименти ще получаваме различни честоти за един и същи резултат. За математически цели е полезно да се абстрахираме от тези разлики и да заменим честотите на един резултат при различните експерименти с характеристика, не зависеща от експеримента. Разбира се, при определянето на тази нова характеристика трябва да се стремим да я дефинираме така, че да е колкото може по-близка до значенията на честотите за достатъчно голям брой експерименти.



***Дефиниция.*** Нека $S$ е крайно множество от възможните резултати на някакъв експеримент (без ограничение на общността $S = I_k$), елементите на което ще наричаме *елементарни събития*. Нека множеството от реални числа $\{p_1, p_2, \ldots, p_k\}$ е такова, че $\sum_{i=1}^{k} p_i = 1$, $0 < p_i < 1$. Тогава $(S; p_1, p_2, \ldots, p_k)$ наричаме *събитийно пространство*, а $p_i$ - *вероятност* на елементарното събитие $i \in S$.

Изключваме $p_i = 0$, защото ако в резултат на многобройни експерименти едно елементарно събитие се появява с честота 0, то изглежда не е елемент на нашето пространство и по-добре е да го изключим от $S$. От друга страна $p_i \neq 1$, защото противното означава $S$ да се състои само от един елемент, а такова събитийно пространство не представлява интерес.

От казаното по-горе за хвърлянето на един зар е ясно, че съпоставеното му събитийно пространство е

$$(\{1,2,3,4,5,6\}; 1/6, 1/6, 1/6, 1/6, 1/6, 1/6).$$

Разбира се, за дадено събитийно пространство, е интересно да се опитаме да съпоставим вероятности и на други събития, различни от елементарните. Например, при хвърлянето на един зар можем да се запитаме каква е вероятността да се падне число по-голямо от 3. Затова разширяваме понятието събитийно пространство по следния начин:

***Дефиниция.*** Нека $(S; p_1, p_2, \ldots, p_k)$ е събитийно пространство и $(2^S; \cup, \cap, \bar{}, \emptyset, S)$ е булевата алгебра на подмножествата на $S$. Всяко $S' \subseteq S$ наричаме *събитие*. Двойката $(S; Pr)$, $Pr : 2^S \to \{x | x \in R, 0 \leq x \leq 1\}$, наричаме *вероятностно пространство*, ако:

а) $Pr(i) = p_i$, за всяко елементарно събитие $i \in S$;

б) $Pr(\emptyset) = 0$ и $Pr(S) = 1$;

в) ако $S_1, S_2 \subseteq S$ и $S_1 \cap S_2 = \emptyset$, то $Pr(S_1 \cup S_2) = Pr(S_1) + Pr(S_2)$.

Пряко от дефиницията на вероятностно пространство $(S; Pr)$ получаваме верността на следните твърдения:

**Лема 2.6.1** *Нека $S_1, S_2 \subseteq S$. Тогава*

*а) $Pr(\overline{S_1}) = 1 - Pr(S_1)$;*

*б) $Pr(S_1 \cup S_2) = Pr(S_1) + Pr(S_2) - Pr(S_1 \cap S_2)$.*

**Лема 2.6.2** *Нека $|S| = k$ и $p_i = 1/k$ $\forall i \in S$. Тогава $Pr(S') = |S'|/k$, $\forall S' \subseteq S$.*

За събитието от примера по-горе, състоящо се от всички числа по-големи от 3, последната лема ни дава $Pr(\{4,5,6\}) = |\{4,5,6\}|/6 = 3/6 =$

1/2. Да пресметнем вероятността при хвърляне на един зар да се падне събитието $\sigma$ – четно число, по-голямо от 3. Получаваме

$$Pr(\sigma) = Pr(\{2,4,6\} \cap \{4,5,6\}) = Pr(\{4,6\}) = 1/3.$$

Когато елементарните събития са равновероятни, пресмятането на вероятността на някое събитие е комбинаторен проблем, тъй като се свежда до намиране на броя на елементите му. Макар че при произволно разпределение на вероятностите за елементарните събития нещата малко се усложняват, аналогията между вероятностните пространства и съответното комбинаторно пространство се запазва. Твърденията на Лема 2.6.1.а и Лема 2.6.2 са вероятностни аналози на комбинаторните Принципи на изваждането и делението. Изискването (в) на Дефиницията е частен случай на вероятностния аналог на комбинаторния Принцип на разбиването, който в общия случай гласи:

**Лема 2.6.3** *Нека $(S; Pr)$ е вероятностно пространство, а фамилията от събития $\{S_1, S_2, \ldots, S_r\}$ е разбиване на $S' \subseteq S$. Тогава $Pr(S') = \sum_{i=1}^{r} Pr(S_i)$.*

Твърдението на Лема 2.6.1.б пък се обобщава до следния вероятностен аналог на комбинаторния Принцип на включването и изключването:

**Лема 2.6.4** *Нека $S_1, S_2, \ldots, S_r$ са събития от вероятностното пространство $(S; Pr)$. Тогава*

$$\begin{aligned} Pr(\overline{S_1} \cap \overline{S_2} \cap \ldots \cap \overline{S_r}) &= Pr(S) - \textstyle\sum_{i=1}^{r} Pr(S_i) + \sum_{i \neq j} Pr(S_i \cap S_j) \\ &\quad - \ldots + (-1)^r Pr(S_1 \cap S_2 \cap \cdots \cap S_r). \end{aligned}$$

Оставяме на читателя да обмисли доказателството на последните две леми.

Сега ще покажем конструкция на вероятностно пространство, което е вероятностен аналог на комбинаторния Принцип на умножението. Нека два експеримента, с резултати от крайните множества $S_1$ и $S_2$ се провеждат едновременно, като резултатите им не зависят един от друг. Нека образуваме съответните вероятностни пространства $(S_1; Pr)$ и $(S_2; Pr)$. Да разгледаме декартовото произведение $\mathcal{S} = S_1 \times S_2$ и да образуваме вероятностно пространство $(\mathcal{S}; Pr)$, елементарните събития на което са наредените двойки $(s_1, s_2)$ от $\mathcal{S}$. За всяко такова елементарно събитие дефинираме вероятност $Pr(s_1, s_2) = Pr(s_1).Pr(s_2)$. Тъй като



$0 < Pr(s_1) < 1$ и $0 < Pr(s_2) < 1$, то и $0 < Pr(s_1, s_2) < 1$, а

$$\begin{aligned} \textstyle\sum_{s_1 \in S_1, s_2 \in S_2} Pr(s_1, s_2) &= \textstyle\sum_{s_1 \in S_1} \sum_{s_2 \in S_2} Pr(s_1) Pr(s_2) = \\ &= \textstyle\sum_{s_1 \in S_1} Pr(s_1) (\sum_{s_2 \in S_2} Pr(s_2)) = \\ &= \textstyle\sum_{s_1 \in S_1} Pr(s_1).1 = 1. \end{aligned}$$

Събитията $s_1$ и $s_2$ от тази конструкция наричаме *независими*. Вероятностите на неелементарните събития, съгласно дефиницията на вероятностно пространство, дефинираме като сума от вероятностите на съставящите ги елементарни.

За пример да разгледаме експеримент, при който хвърляме два различаващи се един от друг зара – бял и черен. Вероятността на кое да е елементарно събитие $(i, j)$ – $i$ се е паднало на белия, а $j$ – на черния зар е $Pr(i, j) = Pr(i).Pr(j) = 1/6.1/6 = 1/36$. Ако двата зара са неразличими, тогава образуваме пространство от ненаредените двойки $\{i, j\}$, като вероятността $Pr(\{i, j\}) = 1/36$, ако $i = j$ и $Pr(\{i, j\}) = Pr(i, j) + Pr(j, i) = 1/36 + 1/36 = 1/18$, ако $i \neq j$.

Да разгледаме ситуация, когато вероятността някое събитие да се случи в резултат на експеримента зависи от случването на друго събитие. Например, каква е вероятността при хвърляне на два различими зара – бял и черен, да се получи сума поне 8 при условие, че на белия зар се е паднало поне 5.

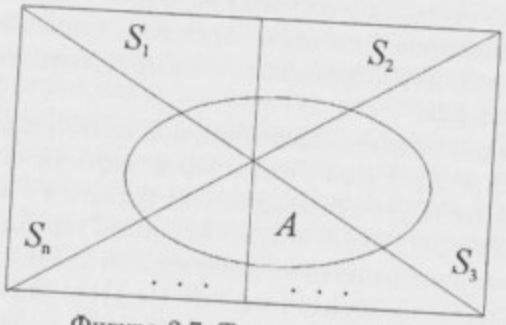
***Дефиниция.*** Нека $Pr(S_1) \neq 0$. Вероятността да се случи $S_2$ при условие, че се е случило $S_1$, наричаме *условна вероятност* за $S_2$ при условие $S_1$ и бележим с $Pr(S_2|S_1)$.

За да въведем правило за пресмятане на условната вероятност, ще забележим, че вероятността да се случи $S_2$ при условие че се е случило $S_1$ се дава от $Pr(S_1 \cap S_2)$ в рамките на пространството, съдържащо $S_1$ и $S_2$. Ние, обаче, искаме да намерим тази вероятност в рамките на пространството образувано от $S_1$, за което знаем с абсолютна сигурност, че се е случило (вероятност 1 за $S_1$). Комбинаторната аналогия ни подсказва да дефинираме $Pr(S_2|S_1) = Pr(S_1 \cap S_2)/Pr(S_1)$.

За условното събитие, което дадохме като пример получаваме $S_1 = \{(5, x), (6, x) | x = 1, 2, 3, 4, 5, 6\}$ и $Pr(S_1) = 1/3$, а $S_2 = \{(2,6), (3,5), (3,6), (4,4), (4,5), (4,6), (5,3), (5,4), (5,5), (5,6), (6,2), (6,3), (6,4), (6,5), (6,6)\}$, т.е. $S_1 \cap S_2 = \{(5,3), (5,4), (5,5), (5,6), (6,2), (6,3), (6,4), (6,5), (6,6)\}$ и $Pr(S_1 \cap S_2) = 9/36 = 1/4$. Така $Pr(S_2|S_1) = (1/4)/(1/3) = 3/4$. Разбира се $Pr(S_2|\overline{S_1}) = 1 - Pr(S_2|S_1)$ (докажете го!). Последният факт можем да обобщим в следната

**Теорема 2.6.1 (Бейз)** *Нека $\{S_1, S_2, \ldots, S_r\}$ е разбиване на вероятностното пространство $(S; Pr)$, а $A$ е събитие от това пространство (вж. Фиг. 2.7). Тогава*

$$Pr(S_i|A) = \frac{Pr(S_i).Pr(A|S_i)}{\sum_{j=1}^{r} Pr(S_j).Pr(A|S_j)}, \forall i \in I_r.$$




Фигура 2.7: Теорема на Бейз

***Доказателство.*** Очевидно $\{A \cap S_j | j \in I_r\}$ е разбиване на $A$ и следователно $Pr(A) = \sum_{j=1}^{r} Pr(A \cap S_j)$. Но според дефиницията $Pr(A \cap S_i) = Pr(S_i).Pr(A|S_i)$. Следователно

$$Pr(S_i|A) = \frac{Pr(S_i \cap A)}{Pr(A)} = \frac{Pr(S_i).Pr(A|S_i)}{\sum_{j=1}^{r} Pr(A \cap S_j)} = \frac{Pr(S_i).Pr(A|S_i)}{\sum_{j=1}^{r} Pr(S_j).Pr(A|S_j)}. \square$$

Понятието, което ще дефинираме сега е механизъм, за преобразуване на произволно вероятностно пространство в такова, елементарните събития на което са реални числа. Най-естествена е тази трансформация в случаите, когато самите елементарни събития са реални числа, но и в редица други случаи трансформацията ни дава пространства, много удобни за различни практически нужди.

***Дефиниция.*** Нека $(S; Pr)$ е вероятностно пространство, а $\mathcal{S} = \{S_1, S_2, \ldots, S_r\}$ е разбиване на $S$. Еднозначната функция $\chi : \mathcal{S} \to R$ наричаме *случайна величина* или *случайна функция*. Множеството от вероятности $\{Pr(S_1), Pr(S_2), \ldots, Pr(S_r)\}$ наричаме *разпределение* на случайната величина $\chi$.

Например, при хвърляне на един зар можем да дефинираме случайната величина $\chi_1 : I_6 \to R$, такава че $\chi_1(i) = i$. Разпределението на тази случайна величина е $\{1/6, 1/6, 1/6, 1/6, 1/6, 1/6\}$. При хвърляне на два различими зара можем да дефинираме, например, случайната величина $\chi_2 : I_6 \times I_6 \to R$, такава че $\chi_2(i, j) = i + j$. Значения на



тази случайна величина са целите числа от 2 до 12, а разпределението ѝ е $\{1/36, 2/36, 3/36, 4/36, 5/36, 6/36, 5/36, 4/36, 3/36, 2/36, 1/36\}$. Като последен пример да разгледаме вероятностното пространство, състоящо от всички граждани на дадена страна, като на всеки гражданин е съпоставена една и съща вероятност. Да дефинираме разбиване на пространството на събития $\mathcal{B} = \{B_m, B_{m+1}, \ldots, B_M\}$, където $m$ и $M$ са, съответно, минималният и максималният номер на обувки $(m < M)$, които се носят в страната. Всеки гражданин, носещ обувки с номер $i$, е поставен в събитието $B_i$. Така получаваме случайната величина $\chi_3 : B \to \{m, m+1, \ldots, M\}$. Разпределението ѝ може да се установи с достатъчна точност след съответни демографски изследвания. За целите на изложението, ще разглеждаме само случайни величини, които приемат крайно много стойности.

***Дефиниция.*** Нека $x_1, x_2, \ldots, x_n$ са стойностите на случайната величина $\chi$, а $p_1, p_2, \ldots, p_n$ е съответното ѝ разпределение. Реалното число $E(\chi) = \sum_{i=1}^{n} p_i x_i$ наричаме *математическо очакване* на случайната величина $\chi$.

Забележете, че математическото очакване може да се окаже число, различно от всички стойности на съответната случайна величина. Ролята му е да бъде проста обобщена мярка за най-вероятната стойност на случайната величина. Да пресметнем математическото очакване на случайните величини $\chi_1$ и $\chi_2$, дефинирани по горе. Получаваме $E(\chi_1) = \sum_{i=1}^{6} i/6 = 21/6 = 3.5$ и $E(\chi_2) = \sum_{i=1}^{5} \frac{14.i}{36} + \frac{6.7}{36} = \frac{210+42}{36} = 7$. За случайната величина $\chi_3$, не разполагаме със съответните данни, но от практически опит можем да твърдим, че ако се ограничим само до мъжката половина от населението, математическото очакване на $\chi_3$ ще бъде около 41-42.

Вижда се, че във втория и третия пример математическото очакване определя наистина важни характеристики – числото, на което най-често трябва да залагаме, ако играем хазартна игра с хвърляне на два зара или пък номерът обувки, от които трябва да се произведат най-големи количества, за да бъде най-добре задоволено пазарното търсене. За математическото очакване от първия пример не може да се каже такова нещо, защото при хвърляне на един зар всички значения са равновероятни. За да определим доколко математическото очакване е важно като характеристика, дефинираме следното понятие:

***Дефиниция.*** Нека $\chi$ е случайна величина със стойности $x_1, x_2, \ldots, x_n$, а $p_1, p_2, \ldots, p_n$ е съответното ѝ разпределение. Дефинираме случайната величина $|\chi - E(\chi)|$ със стойности $|x_1 - E(\chi)|, |x_2 - E(\chi)|, \ldots, |x_n - E(\chi)|$ и разпределение $p_1, p_2, \ldots, p_n$. Математическото очакване $E(|\chi - E(\chi)|) =$

$D(\chi)$ наричаме *отклонение* на случайната величина $\chi$.

Отклонението на случайната величина показва големината на най-вероятното отместване от математическото ѝ очакване. За първия пример стойността на тази характеристика е твърде голяма: $D(\chi_1) = (2.5 + 1.5 + 0.5 + 0.5 + 1.5 + 2.5)/6 = 1.5$ и показва, че с много голяма вероятност случайната величина ще се отклонява от математическото очакване до 2 вляво и 5 в дясно. За втория пример получаваме отклонение $D(\chi_2) = (10.1 + 8.2 + 6.3 + 4.4 + 2.5)/36 < 2$, което показва, че даже сумите 5 и 9 не са от твърде вероятните. От практически съображения можем да твърдим, че при номерата на обувките на мъжката половина отклонението едва ли надхвърля 3.

---

## Упражнения

### Упражнение 2.1

Докажете следните тъждества:

а) $\binom{n}{k} = \binom{n-1}{k} + \binom{n-2}{k-1} + \ldots + \binom{n-k-1}{0}$;

б) $\binom{n+m}{m} = \sum_{k=0}^{n} \binom{r+k}{r}\binom{m-r}{n-k}$;

в) $\sum_{k=0}^{n} k\binom{n}{k} = n2^{n-1}$;

г) $\sum_{k=0}^{n} \binom{n}{k}(m-1)^{n-k} = m^n$.

### Упражнение 2.2

Две сядания около кръгла маса не са различни, ако всеки от седналите има едни и същи съседи. По колко различни начина могат да седнат около кръгла маса:

а) $n (n > 2)$ човека;

б) $n$ мъже и $n$ жени $(n > 2)$, като две лица от един и същ пол не седят едно до друго?

### Упражнение 2.3

Колко решения в естествени числа имат следните уравнения:

а) $x_1 + x_2 + x_3 = 10$;

б) $x_1 + x_2 + x_3 = 10, x_1 \geq 1, x_2 \geq 3$;

в) $x_1 + x_2 + x_3 = 10, x_1 \geq 1, x_2 \geq 2, x_3 \leq 3$?



### Упражнение 2.4

Колко различни думи могат да се получат от разместването на буквите на думата:

а) *PERMUTATION*;

б) *ALABALA*;

в) *MISSISSIPPI*?

### Упражнение 2.5

От колода от 52 карти се избират 13. В колко различни извадки ще се срещнат:

а) точно 1 ас;

б) поне 2 аса;

в) между 4 и 7 пики;

г) точно 2 аса и точно 2 трефи;

д) точно 2 аса и не повече от 2 трефи;

е) поне 7 карти от един цвят?

### Упражнение 2.6

По колко различни начина могат да се поставят:

а) $n$ различими предмета в $m$ различими кутии;

б) $n$ неразличими предмета в $m$ различими кутии;

в) $n$ неразличими предмета в $m$ различими кутии така, че нито една кутия да не е празна;

г) $n$ неразличими предмета в $m$ неразличими кутии?

### Упражнение 2.7

Намерете броя на $n$-мерните двоични вектори:

а) които са на равно разстояние на Хеминг от два зададени вектора;

б) за които сумата от разстоянията на Хеминг до два зададени вектора е $k$ $(1 \leq k \leq n)$.

### Упражнение 2.8

Множеството от всички двоични вектори от $\{0,1\}^n$, които във фиксирани $n - k$ позиции имат еднакви значения, $0 \leq k \leq n$, наричаме *$k$-куб*.

а) Колко вектора има в един $k$-куб?

б) Колко различни $k$-куба има в $\{0,1\}^n$?

в) Колко различни $k$-куба съдържат даден вектор (т.е. 0-куб)?
г) Колко различни $k$-куба съдържат даден $m$-куб, $0 \leq m < k$?

**Упражнение 2.9**

На колко части разделят:

а) евклидовата равнина $n$ прави, ако всеки две се пресичат, но никои три не минават през една точка;

б) тримерното евклидово пространство $n$ равнини, минаващи през една точка, никои три от които нямат обща права?

**Упражнение 2.10**

Намерете броя на пермутациите на $n$ елемента, които:
а) оставят на мястото им точно $m, 0 \leq m \leq n$ елемента;
б) не оставят на мястото му нито един елемент.

**Упражнение 2.11**

Група от 15 депутати използва услугите на 5 секретарки. За всеки депутат е определена точно една секретарка и никоя секретарка не работи за повече от 4 депутати. Докажете, че поне 3 секретарки работят за 3 и повече депутати.

**Упражнение 2.12**

Намерете производяща функция $\tilde{f}(z)$ за:
а) $f(n) = kn, n = 0, 1, 2, \ldots$;
б) $f(n) = n^2, n = 0, 1, 2, \ldots$;
в) $f(n) = k^n n, n = 0, 1, 2, \ldots$.

**Упражнение 2.13**

Изразете производящата функция на $F(n)$ чрез производящата функция на $f(n)$, ако:
а) $F(n) = \sum_{i=0}^{n} f(i), n = 0, 1, 2, \ldots$;
б) $F(n) = f(n+1) - f(n), n = 0, 1, 2, \ldots$;
в) $F(n) = \sum_{r=0}^{n} f(r)g(n-r), n = 0, 1, 2, \ldots$.



**Упражнение 2.14**

Решете рекурентните уравнения:
а) $a_0 = 0; a_n = 2a_{n-1} + n, n \geq 1$;
б) $a_0 = 0; a_n = 2a_{[n/2]} + c, n \geq 1, n = 2^k$;
в) $a_0 = 0; a_n = 4a_{[n/2]} + n, n \geq 1, n = 2^k$;
г) $a_0 = 0; a_n = na^2_{[n/2]}, n \geq 1, n = 2^k$.

**Упражнение 2.15**

Каква е вероятността с 1 фиш от тото-играта „6 от 49“ да улучите:
а) $k (1 \leq k \leq 6)$ числа; б) поне $k (1 \leq k \leq 6)$ числа?

**Упражнение 2.16**

Нека град с квадратни квартали е построен във формата на „триъгълник“ така, че кръстовищата му са номерирани с двойките естествени числа $(i, j), i + j \leq n$. Кръстовищата с $i + j = l$ наричаме $l$-ти ред, а кръстовището $(i, j)$ – $i$-то кръстовище на $(i + j)$-тия ред. Тръгвайки от кръстовището $(0, 0)$, на всяко кръстовище $(i, j)$ избираме с вероятност $1/2$ посоката към кръстовище $(i+1, j)$ и със същата вероятност посоката към кръстовище $(i, j+1)$. Каква е вероятността след $n$ такива стъпки да попаднем на $i$-то кръстовище от $n$-я ред?

**Упражнение 2.17**

В страната Хикстия 45% гласуват за партията $A$, 30% за партията $B$, 5% за партията $C$, а 20% не гласуват за никоя партия. От привържениците на $A$ 25% подкрепят настоящото правителство, от привържениците на $B$ – 60%, от привържениците на $C$ – 90%, а от останалите – всеки втори. Гражданинът $P$ се изказва против правителството. Пресметнете вероятността за привързаността му към различните политически сили.

**Упражнение 2.18**

Вестник с тираж 100000 броя и цена 1 лев разиграва лотария чрез талон, напечатан в един брой на вестника, с награда – компютър на стойност 1000 лева. Какъв финансов резултат може да очаквате, ако се включите в лотарията заедно с всички читатели, изкупили тиража на този вестник?

**Упражнение 2.19**

Докажете, че съществува само една (с точност до изоморфизъм) проективна равнина от ред 2.

**Упражнение 2.20**

В пистово състезание с мотоциклети участват 20 състезатели. Във всеки кръг на състезанието участват 4 състезатели. Може ли да се състави програма на състезанието така, че всеки двама състезатели да се срещнат в точно един кръг?

**Упражнение 2.21**

Постройте 2-(13,4,1) дизайн.

**Упражнение 2.22**

Намерете групата от автоморфизми на 2-(7,3,1) дизайна.

**Упражнение 2.23**

Докажете, че непразното множество $\mathcal{B}$ от подмножества на множеството $E$ е множество от бази на матроид $(E, \mathcal{B})$, ако $\forall B_1, B_2 \in \mathcal{B}$ и $\forall x \in B_1 \setminus B_2$, $\exists y \in B_2 \setminus B_1$ такова, че $B_1 \cup \{y\} \setminus \{x\} \in \mathcal{B}$.

# Глава 3

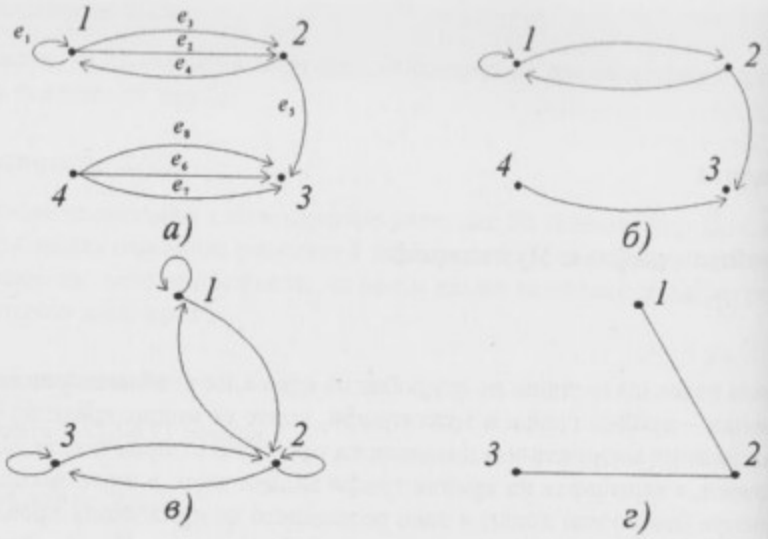
# Крайни графи и мултиграфи

В тази глава ще се спрем по-подробно на един клас комбинаторни конфигурации – крайни графи и мултиграфи, които са мощно средство за изграждане на математически модели на проблеми от практиката. За преведената в термините на крайни графи задача често е известно някакво решение (добро или лошо) и така решаването на приложния проблем се свежда до решаването на съответната графова задача. Затова крайните графи са едни от най-изследваните комбинаторни конфигурации.

## 3.1 Основни понятия

*Дефиниция.* Нека $V = \{v_1, v_2, \ldots, v_n\}$ е крайно множество, елементите на което наричаме *върхове*, а $E = \{e_1, e_2, \ldots, e_m\}$ е крайно множество, елементите на което наричаме *ребра*. Функцията $f_G : E \to V \times V$, съпоставяща на всяко ребро наредена двойка от върхове, наричаме **краен ориентиран мултиграф**. Казваме още, че е зададен краен ориентиран мултиграф $G$ с върхове $V$, ребра $E$ и *свързваща функция* $f_G$ и го означаваме с $G(V, E, f_G)$.

За представяне на крайни ориентирани мултиграфи ще използваме аналог на диаграмите (графичното представяне) на релациите от раздел 1.2. Всеки връх $v_i \in V$ изобразяваме с точка в равнината, а всяко ребро $e \in E$ такова, че $f_G(e) = (v_i, v_j)$ – с линия, започваща във $v_i$ и завършваща със стрелка във $v_j$ (за да се посочи кой е вторият елемент на наредената двойка). Ребрата $e$ такива, че $f_G(e) = (v_i, v_i)$, наричаме *примки*. Името на реброто записваме до линията. На Фиг. 3.1.а е изобразен крайният ориентиран мултиграф $G$ с върхове $V = \{1, 2, 3, 4\}$, ребра $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$ и функция $f_G$, дефинирана както следва: $f_G(e_1) = (1, 1)$, $f_G(e_2) = f_G(e_3) = (1, 2)$, $f_G(e_4) = (2, 1)$, $f_G(e_5) = (2, 3)$, $f_G(e_6) = f_G(e_7) = f_G(e_8) = (4, 3)$. Реброто $e_1$ е примка.




Фигура 3.1: Крайни мултиграфи и графи.

***Дефиниция.*** Нека $G(V, E, f_G)$ е краен ориентиран мултиграф и функцията $f_G$ е еднозначна, т.е. $f(e_i) \neq f(e_j), i \neq j$. Тогава $G(V, E, f_G)$ наричаме *краен ориентиран граф.*

Не е трудно да се забележи, че понятието краен ориентиран граф може да се дефинира и по друг начин. Тъй като функцията $f_G$ е еднозначна, всяка наредена двойка от $V \times V$ може да участва най-много един път в дефиницията на граф и затова множеството $E$ може да бъде определено като подмножество на $V \times V$ или, с други думи, крайният ориентиран граф е релация над декартовото произведение $V \times V$. В този смисъл не се нуждаем от имена на ребрата – всяко ребро се определя еднозначно от съответната двойка върхове. Не е необходима и свързващата функция. Затова обикновено задаваме крайния ориентиран граф $G$ с множеството му от върхове $V$ и множеството му от ребра $E \subseteq V \times V$, като го обозначаваме с $G(V, E)$. Казваме, че ориентираният граф е зададен с *множеството от ребрата си.* На Фиг. 3.1.б е представен крайният ориентиран граф $G(V, E)$ с $V = \{1, 2, 3, 4\}, E = \{(1, 1), (1, 2), (2, 1), (2, 3), (4, 3)\}$. Представянията на крайните ориентирани графи в равнината съвпадат с диаграмите на съответните релации.

Връзката между крайните ориентирани мултиграфи и крайните ориентирани графи (релациите над $V \times V$) е както между конфигурациите с наредба и повтаряне и конфигурациите с наредба без повтаряне, разгледани в 2.2.



***Дефиниция.*** Нека $G(V, E)$ е краен ориентиран граф, такъв че релацията $E \subseteq V \times V$ е рефлексивна и симетрична. В такъв случай $G(V, E)$ наричаме *краен неориентиран граф* или просто *граф.*

На Фиг. 3.1.в. графът $G(V, E)$ с $V = \{1, 2, 3\}$ и $E = \{(1, 1), (1, 2), (2, 1), (2, 2), (2, 3), (3, 2), (3, 3)\}$ е показан като краен ориентиран граф. За простота при представянето на графи в равнината няма да изобразяваме примките, а вместо двете линии, съответстващи на ребрата $(v_i, v_j)$ и $(v_j, v_i)$ ще изобразяваме една линия, свързваща двата върха $v_i$ и $v_j$ без посочен първи и втори край (*неориентирано ребро*). Ще запазим означението $(v_i, v_j)$ (или $(v_j, v_i)$) и в неориентирания случай, независимо от това, че всъщност става дума за неориентираното ребро $\{v_i, v_j\}$. На Фиг. 3.1.г. е дадено общоприетото представяне на крайния граф от фигура 3.1.в. Ориентираният граф, който представяме за по-просто като неориентиран е фактически рефлексивно и симетрично затваряне на релацията, получена от неориентирания с произволен избор на ориентация на всяко неориентирано ребро.

Крайният неориентиран граф $G(V, E)$ можем да превърнем в *краен неориентиран мултиграф*, ако позволим повече от едно неориентирано ребро да свързва два върха от $V$, т.е. ако вместо множество $E \subseteq V \times V$ вземем мултимножество от елементи на $V \times V$.

Връзката между крайните неориентирани графи и крайните неориентирани мултиграфи е както между конфигурациите без наредба и без повтаряне (множествата) и конфигурациите без наредба с повтаряне (мултимножествата), разгледани в 2.2.

***Дефиниция.*** Върховете $v_i, v_j \in V$ на графа $G(V, E)$ наричаме *съседни*, ако $(v_i, v_j) \in E$. Ако графът е ориентиран, казваме, че $v_i$ е *баща* на $v_j$, а $v_j$ е *син* на $v_i$. Казваме, че $v_i$ и $v_j$ са *краища* на реброто $(v_i, v_j)$. Броят $d(v_i)$ на ребрата в неориентирания граф $G(V, E)$, на които $v_i \in V$ е край, наричаме *степен* на $v_i$. Броят $d^-(v_i)$ на ребрата в ориентирания граф $G(V, E)$, които започват във $v_i \in V$, наричаме *полустепен на изхода* на $v_i$, а броят $d^+(v_i)$ на ребрата, които завършват във $v_i \in V$, наричаме *полустепен на входа* на $v_i$.

***Дефиниция.*** Краен неориентиран граф, всички върхове на който са с една и съща степен $d$ наричаме *регулярен* със степен $d$.

***Дефиниция.*** Нека $G(V, E, f_G)$ е краен ориентиран мултиграф, $V' \subseteq V$. Нека $E' \subseteq E$ се състои от тези ребра, краищата на които са във $V'$, а $f'_G(e) = f_G(e), \forall e \in E'$ ($f'_G$ е рестрикцията на $f_G$ върху $E'$). Тогава

крайният ориентиран мултиграф $G'(V', E', f'_G)$ наричаме *подмултиграф* на $G$. Ако $G$ е краен ориентиран или неориентиран граф, то $G'(V', E')$ наричаме *подграф* на $G$. И в двата случая казваме, че $G'$ е *индуциран* от $V'$ подграф на $G$.

***Дефиниция.*** Нека $G(V, E)$ и $G'(V', E')$ са крайни графи. Функцията $\varphi : V \to V'$ наричаме *хомоморфизъм* на $G$ в $G'$, ако $\forall (v_i, v_j) \in E$ е в сила $(\varphi(v_i), \varphi(v_j)) \in E'$, т.е. ако $\varphi$ е хомоморфизъм на съответните релации. Ако $\varphi$ е взаимно-еднозначна, казваме че е *изоморфизъм* на $G$ и $G'$, а ако $G = G'$, изоморфизмът $\varphi$ наричаме *автоморфизъм* на $G$.

$$M_{\text{а}} = \begin{Vmatrix} 1 & 2 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 \end{Vmatrix} \quad M_{\text{б}} = \begin{Vmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{Vmatrix} \quad M_{\text{г}} = \begin{Vmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{Vmatrix}$$

Фигура 3.2: Матрици на съседства.

***Дефиниция.*** Матрицата $M = ||a_{ij}||$ с размери $|V| \times |V|$ наричаме *матрица на съседства* на крайния ориентиран мултиграф $G(V, E, f_G)$, ако $\forall v_i, v_j \in V$ е в сила $a_{ij} = |\{e | e \in E, f_G(e) = (v_i, v_j)\}|$.

Когато представяме неориентирани графи с матрици на съседства не отчитаме примките. Така за трите примера от Фиг. 3.1.а, б и г получаваме матриците на съседства, показани на Фиг. 3.2.

Както се вижда, матриците на съседства на крайните ориентирани мултиграфи могат да имат за елементи произволни естествени числа, на крайните ориентирани графи само 0 и 1, а матриците на крайните неориентирани графи са симетрични и с нули по диагонала.

Представянето с матрица на съседства се използва често при компютърна реализация на алгоритми върху мултиграфи и графи, особено ако в алгоритъма често трябва да се отговаря на въпроса: „съседни ли са върховете $v_i$ и $v_j$“. Само една проверка е достатъчна, за да се отговори на такъв въпрос, ако мултиграфът или графът е представен с матрица на съседства. За други алгоритми много по-удобни могат да се окажат други представяния.

Например, в дефинициите използвахме представяне, наречено *списък на ребрата*. То е удобно за реализация на алгоритми, в които съществува фрагмент от вида: „за всяко ребро на графа ...“. Трето, често използвано представяне е *списъци на съседите*. При него за всеки връх е зададено множеството (мултимножеството) от неговите съседи. За трите примера



от Фиг. 3.1 имаме следните представяния със списъци на съседите:

а) 1:1,2,2; 2:1,3; 3:; 4:3,3,3.

б) 1:1,2; 2:1,3; 3:; 4:3.

в) 1:2; 2:1,3; 3:2.

Този тип представяне е удобен за реализация на алгоритми, в които основни са фрагментите от вида: „за всички съседи на върха $v_i$ ...“. Разбира се, допустими са и други представяния на мултиграфите и графите, които облекчават работата на конкретни алгоритми. Не е изключено да се използуват и хибриди на посочените представяния или по няколко представяния едновременно, ако алгоритъмът го изисква.

***Дефиниция.*** Последователността от върхове $v_{i_0}, v_{i_1}, \ldots, v_{i_l}$ на крайния ориентиран мултиграф наричаме *маршрут*, ако $\forall j = 0, 1, 2, \ldots, l-1$, $\exists e \in E$ такова, че $f_G(e) = (v_{i_j}, v_{i_{j+1}})$. Числото $l$ наричаме *дължина* на този маршрут. В случая, когато $v_{i_0} = v_{i_l}$ маршрута наричаме *контур*.

**Теорема 3.1.1** *Нека $G(V, E, f_G)$ е краен ориентиран мултиграф и нека $M = ||a_{ij}||$ е матрицата му на съседства. Нека $M^k = ||a_{ij}^{(k)}||$ е $k$-та степен на $M$ при целочисленото умножение на матрици. Тогава $a_{ij}^{(k)}$ е броят на маршрутите с дължина $k$ от $v_i$ до $v_j$ в крайния ориентиран мултиграф $G$.*

***Доказателство.*** Нека $|V| = n$. Ще докажем твърдението с индукция по $k$.

а) Нека $k = 1$. Твърдението е очевидно, тъй като $M^1 = M$ и $a_{ij}^{(1)}$ е точно броят на маршрутите с дължина 1 (ребрата) от $v_i$ до $v_j$ съгласно дефиницията на матрица на съседства.

б) Да допуснем, че за някое $k > 1$ броят на маршрутите с дължина $k-1$ от $v_i$ до $v_j$ е $a_{ij}^{(k-1)}, \forall i, j \in I_n$.

в) Разглеждаме $M^{(k)} = ||a_{ij}^{(k)}||$, $k > 1$. Съгласно правилото за умножение на матрици $a_{ij}^{(k)} = a_{i1}^{(k-1)} a_{1j}^{(1)} + \cdots + a_{il}^{(k-1)} a_{lj}^{(1)} + \cdots + a_{in}^{(k-1)} a_{nj}^{(1)}$. Да разбием множеството от всички маршрути $T_{ij}^{k}$ от $v_i$ до $v_j$ с дължина $k$ на подмножества $T_{ij}^{kl}$ с еднакъв предпоследен връх $v_l \in V, l \in I_n$. От Принципа на разбиването следва, че $|T_{ij}^{k}| = \sum_{l=1}^{n} |T_{ij}^{kl}|$. Всеки маршрут с дължина $k$ от $v_i$ до $v_j$ и предпоследен връх $v_l$ се получава от някой маршрут от $v_i$ до $v_l$ с дължина $k-1$ и някой маршрут с дължина 1 (ребро) от $v_l$ до $v_j$. Съгласно индукционното предположение броят на различните маршрути от $v_i$ до $v_l$ с дължина $k-1$ е $a_{il}^{(k-1)}$, а на ребрата от $v_l$ до $v_j$ по

дефиниция е $a_{ij}^{(1)}$. От Принципа на умножението имаме $|T_{ij}^{kl}| = a_{il}^{(k-1)} a_{lj}^{(1)}$ и следователно $|T_{ij}^{k}| = \sum_{l=1}^{n} |T_{ij}^{kl}| = \sum_{l=1}^{n} a_{il}^{(k-1)} a_{lj}^{(1)} = a_{ij}^{(k)}$. □

Тази теорема ни позволява да решим, например, следната задача: за дадени два различни върха $v_i$ и $v_j$ в мултиграф $G$ да определим дали съществува маршрут от $v_i$ до $v_j$ (но не и да го намерим!). Достатъчно е да повдигнем матрицата на съседства на $G$ последователно на степени $1, 2, \ldots, n-1$. (Оставяме на читателя да докаже, с прилагане на Принципа на Дирихле, че не е нужно да продължаваме след $(n-1)$-та степен.) Степента $k$, при която за първи път $a_{ij}^{(k)} \neq 0$ ще бъде дължината на най-късия маршрут, свързващ $v_i$ с $v_j$. Ако $a_{ij}^{(k)} = 0, \forall k \in I_{n-1}$, то очевидно не съществува маршрут от $v_i$ до $v_j$.

За случая на неориентирани графи ще дефинираме алтернатива на понятието маршрут по следния начин:

***Дефиниция.*** Последователността от върхове $v_{i_0}, v_{i_1}, \ldots, v_{i_l}$ на крайния неориентиран граф $G(V, E)$ наричаме *път* в $G$ от $v_{i_0}$ до $v_{i_l}$, ако $(v_{i_j}, v_{i_{j+1}}) \in E$, $\forall j \in J_l$ и $v_{i_{j-1}} \neq v_{i_{j+1}}$, $\forall j \in I_{l-1}$. Броят $l$ на ребрата наричаме *дължина* на пътя. Ще считаме, че съществува *тривиален път* с дължина 0 от всеки връх $v_i$ до $v_i$. (Така отчитаме наличието на примки в неориентирания граф, но им даваме тежест 0 при образуване дължината на пътя.) Път, за който $l \geq 3$ и $v_{i_0} = v_{i_l}$, наричаме *цикъл*.

За всеки краен неориентиран граф $G(V, E)$ дефинираме релацията $\mathcal{P}_G \subseteq V \times V$ така: $(v_i, v_j) \in \mathcal{P}_G$ тогава и само тогава, когато съществува път в $G$ от $v_i$ до $v_j$.

**Теорема 3.1.2** *Релацията $\mathcal{P}_G$ е релация на еквивалентност.*

***Доказателство.*** а) Релацията е рефлексивна, защото $\forall v_i \in V$ съществува път от $v_i$ до $v_i$ (с дължина 0) и следователно $(v_i, v_i) \in \mathcal{P}_G$.

б) Нека $(v_i, v_j) \in \mathcal{P}_G$, т.е. съществува път $v_i = v_{i_0}, v_{i_1}, \ldots, v_{i_l} = v_j$, тогава $v_j = v_{i_l}, v_{i_{l-1}}, \ldots, v_{i_0} = v_i$ е път от $v_j$ до $v_i$ и $(v_j, v_i) \in \mathcal{P}_G$. Следователно релацията е симетрична.

в) Нека $(v_i, v_j) \in \mathcal{P}_G$ и $(v_j, v_k) \in \mathcal{P}_G$, т. е. съществуват пътищата $v_i = v_{i_0}, v_{i_1}, \ldots, v_{i_t} = v_j$ и $v_j = v_{j_0}, v_{j_1}, \ldots, v_{j_m} = v_k$. Ако $v_i = v_k$, тогава има път от $v_i$ до $v_k$ (тривиалният път с дължина 0). Затова нека $v_i \neq v_k$. Двата пътя имат поне един общ връх $v_j$, затова нека $t \in \{0, 1, \ldots, l\}$ е най-малкото, такова че $v_{i_t} = v_{j_s}$ за някое $s \in \{0, 1, \ldots, m\}$. Ако $s = m$, тогава $v_{i_t} = v_{j_m} = v_k$ и $v_i = v_{i_0}, v_{i_1}, \ldots, v_{i_t} = v_k$ е път от $v_i$ до $v_k$. Нека $s < m$. Образуваме последователността от върхове $v_i = v_{i_0}, \ldots, v_{i_{t-1}}, v_{i_t} = v_{j_s}, v_{j_{s+1}}, \ldots, v_{j_m} = v_k$, която е път от $v_i$ до $v_k$, защото $v_{i_{t-1}} \neq v_{j_{s+1}}$.

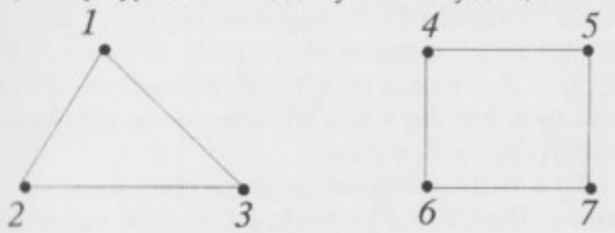


Следователно от $(v_i, v_j) \in \mathcal{P}_G$ и $(v_j, v_k) \in \mathcal{P}_G$ следва $(v_i, v_k) \in \mathcal{P}_G$ и релацията е транзитивна.

От а), б) и в) получаваме, че $\mathcal{P}_G$ е релация на еквивалентност. □

Тук за първи път се срещаме с нетривиална релация на еквивалентност, класовете на която могат да съдържат повече от един елемент. Върховете от всеки клас на еквивалентност $V' \subseteq V$ на релацията $\mathcal{P}_G$ индуцират в $G(V, E)$ по един подграф, който наричаме *свързана компонента* на $G$.

***Дефиниция.*** Графът $G(V, E)$ се нарича *свързан*, ако $\forall v_i, v_j \in V$, $\exists$ път в $G$ от $v_i$ до $v_j$. Дефиницията е приложима и за ориентирания случай (както и за случая на мултиграф), със замяна на понятието път с маршрут, но тъй като изискването е твърде силно, ще въведем алтернативно понятие. Ориентираният граф $G(V, E)$ ще наричаме *слабо свързан*, ако $\forall v_i, v_j \in V$, $\exists$ маршрут в $G$ от $v_i$ до $v_j$ или от $v_j$ до $v_i$.



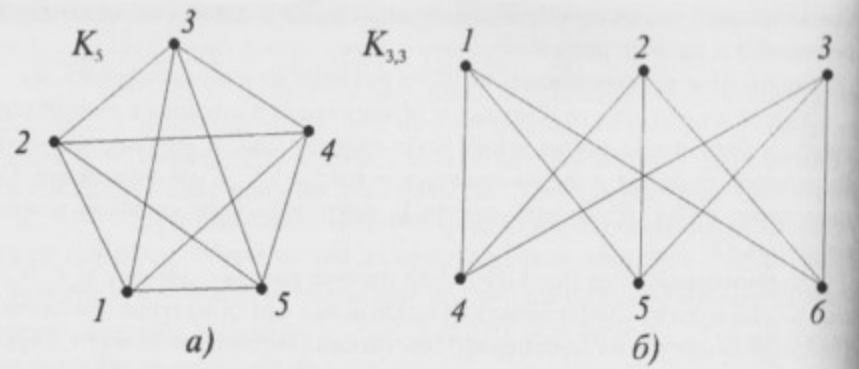

Фигура 3.3: Несвързан граф.

Всеки свързан граф има само една свързана компонента. Графът от Фиг. 3.1.г е свързан, а графът $G(\{1, 2, 3, 4, 5, 6, 7\}, E)$ от Фиг. 3.3 не е свързан. Той има 2 свързани компоненти. Ориентираният граф от Фиг. 3.1.б не е свързан и дори не е слабо свързан.

***Дефиниция.*** Графът $G(V, E)$ наричаме *пълен*, ако $\forall v_i, v_j \in V$, $(v_i, v_j) \in E$. Пълният граф с $n$ върха означаваме с $K_n$. Пълният подграф $G'(V', E')$ на графа $G(V, E)$, индуциран от $V' \subseteq V$, наричаме *клика* на $G$, или *$m$-клика* , ако $|V'| = m$. За всеки граф *кликово число* $\kappa(G)$ определяме, като максималното естествено $m$ такова, че $G$ има $m$-клика.

***Дефиниция.*** Графът $\overline{G}(V, (V \times V) \setminus E)$ наричаме *допълнение* на $G(V, E)$. Допълнението на пълния граф $K_n$ наричаме *празен граф*. Празният граф с $n$ върха означаваме с $\overline{K}_n$. Празният подграф $G'(V', E')$ на графа $G(V, E)$, индуциран от $V' \subseteq V$ наричаме *антиклика* на $G$, или *$m$-антиклика*, ако $|V'| = m$.

На Фиг. 3.4.а е показан пълният граф $K_5$. Максималната клика на $K_n$ съвпада с $K_n$. Затова $\kappa(K_n) = n$.




Фигура 3.4: Пълен (а) и пълен двуделен (б) графи.

***Дефиниция.*** Графът $G(V,E)$ наричаме *двуделен* , ако съществува разбиване $\{V_1, V_2\}$ на $V$, такова че $\forall(v_i,v_j) \in E$ е в сила $v_i \in V_1$, $v_j \in V_2$. Ако $\forall v_i \in V_1, v_j \in V_2$ е в сила $(v_i,v_j) \in E$, казваме, че $G(V_1 \cup V_2, E)$ е *пълен двуделен граф*. Ако $|V_1| = n$, а $|V_2| = m$, тогава означаваме пълния двуделен граф $G(V_1 \cup V_2, E)$ с $K_{n,m}$.

На Фиг. 3.4.б е показан пълният двуделен граф $K_{3,3}$.

***Дефиниция.*** Нека $G(V,E)$ е граф, а $C$ крайно множество от *цветове*. Функцията $f : V \to C$ наричаме *оцветяване на върховете* на $G$, ако $\forall(v_i,v_j) \in E$ е в сила $f(v_i) \neq f(v_j)$. Функцията $g : E \to C$ наричаме *оцветяване на ребрата* на $G$, ако $\forall(v_i,v_j)$ и $(v_i,v_k) \in E$, $v_j \neq v_k$ е в сила $f(v_i,v_j) \neq f(v_i,v_k)$. Минималното цяло $k$, такова че съществува оцветяване на върховете на $G$ с $k$ цвята, наричаме *върхово хроматично число* на $G$ и го означаваме с $\chi_V(G)$. Минималното цяло $k$, такова че съществува оцветяване на ребрата на $G$ с $k$ цвята, наричаме *ребрено хроматично число* на $G$ и го означаваме с $\chi_E(G)$.

Очевидно за всеки граф $G$ е в сила $\chi_V(G) \geq \kappa(G)$, тъй като за оцветяване на върховете на $k$-клика са необходими точно $k$ цвята. Аналогично $\chi_E(G) \geq d(G)$, където с $d(G)$ сме означили максималната степен на връх в $G$.

***Дефиниция.*** Графът $G(V,E)$ е *планарен*, ако съществува диаграма на $G$, в която никои две ребра не се пресичат (общият край на две ребра не е пресечна точка).

Пълният граф $K_5$ и пълният двуделен граф $K_{3,3}$ не са планарни. Не е планарен и всеки граф, на който $K_5$ или $K_{3,3}$ са подграфи. Графите от Фиг. 3.5 са планарни, макар че за графа от 3.5.а това не личи ясно от фигурата. Очевидно, върховете от степен 2 нямат значение за планарността на графа, затова можем да ги изключим, като заменим двете ребра на които те са краища с едно ребро. Полученият граф наричаме *свиване* на първоначалния. Без доказателство ще приведем следната

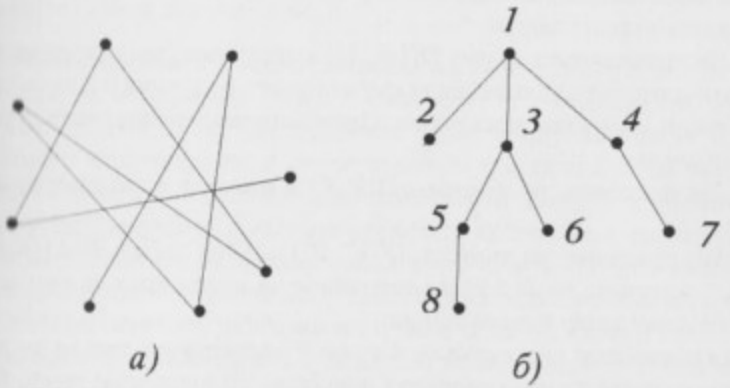


**Теорема 3.1.3 (К. Куратовски)** *Графът $G(V,E)$ е планарен т.с.т.к. не съдържа подграфи, които се свиват до $K_5$ и $K_{3,3}$.*

## 3.2 Дърветa

***Дефиниция.*** Свързан граф без цикли наричаме *дърво*, а несвързан граф без цикли – *гора*.

На Фиг. 3.5.а е показан граф, който е дърво. Отсъствието на цикли се забелязва трудно, свързаността още по-трудно. Затова ще дадем няколко алтернативни дефиниции на това изключително важно за информатиката понятие.




Фигура 3.5: Дърво (свързан граф без цикли) (а) и кореново дърво (б).

***Дефиниция.*** а) Графът $D(\{r\},\{\})$ с един връх $r$ и без ребра наричаме *дърво с корен* $r$ (*кореново дърво*). Единственият връх $r$ е единствен *лист* на това кореново дърво.

б) Нека $D(V,E)$ е дърво с корен $r \in V$ и листа $l_1, l_2, \ldots, l_r$. Нека $u \in V, w \notin V$. Тогава $D'(V',E') = D'(V \cup \{w\}, E \cup \{(u,w)\})$ е също дърво с корен $r$. Ако $u = l_i, 1 \leq i \leq r$, тогава листа на $D'$ са $l_1, \ldots, l_{i-1}, w, l_{i+1}, \ldots, l_r$. В противен случай листа на $D'$ са $l_1, l_2, \ldots, l_r, w$.

в) Няма други коренови дървета.

Операцията, която приложихме в индуктивната стъпка на горната дефиниция, наричаме *присъединяване на връх*. Кореновите дървета са

неориентирани графи, но от дефиницията получаваме *неявна ориентация* на всяко ребро – от върха, към който присъединяваме (*баща*) към присъединения връх (*син*). Затова, при записване на ребрата на кореново дърво, ще поставяме на първо място в наредената двойка бащата, а на второ – сина.

На Фиг. 3.5.б е показано кореновото дърво $D(V,E)$ с $V = \{1,2,3,4,5,6,7,8\}$, $E = \{(1,2),(1,3),(1,4),(3,5),(3,6),(4,7),(5,8)\}$. Корен е върхът 1. Върховете 2, 3 и 4 са присъединени към 1. Върховете 5 и 6 към 3, върхът 7 към 4, а върхът 8 към 5. Листа на това кореново дърво са върховете 2, 6, 7 и 8 – към никой от тях не е присъединен друг връх.

Понятието кореново дърво е много близко до понятието дърво. Това се вижда и от следната

**Теорема 3.2.1** *Всяко дърво с корен е дърво.*

***Доказателство.*** Ще покажем, че всяко дърво с корен е свързан граф и няма цикли. Доказателството ще направим с индукция по дефиницията на дърво с корен.

а) За тривиалното дърво $D(\{r\},\{\})$ е очевидно, че е свързан граф, защото съществува тривиалният път от $r$ до $r$ с дължина 0. Освен това $D$ няма цикли, тъй като няма ребра. Следователно, тривиалното кореново дърво е дърво.

б) Да допуснем, че дървото $D(V,E)$ с корен $r$ е свързан граф без цикли.

в) Ще докажем, че дървото $D'(V',E') = D'(V \cup \{w\}, E \cup \{(u,w)\})$ с корен $r$, получено от $D$ с присъединяване на новия връх $w$ към $u \in V$ е също свързан граф и няма цикли.

За произволни два върха $v_i$ и $v_j$ от $V$ съществува път от $v_i$ до $v_j$ в $D$, защото според индукционното допускане $D$ е свързан граф. Следователно, съществува такъв път и в $D'$. Път от произволен връх $v_i \in V$ до $w$ в $D'$ получаваме, като към съществуващия (съгласно индукционното допускане) в $D$ път от $v_i$ до $u$ добавим реброто $(u,w)$, с което $w$ е присъединен към $u$. Не е възможно $w$ да съвпада с върха, предхождащ $u$, защото $w \notin V$. Следователно в $D'$ съществува път между всеки два върха и $D'$ е свързан граф.

Съгласно индукционното допускане в $D$ няма цикли. За да има цикъл в $D'$, той непременно трябва да съдържа реброто $(u,w)$. Това обаче не е възможно, тъй като $w$ е край само на реброто $(u,w)$ и съгласно дефиницията за път не може да съществува продължение до цикъл за никой път, достигащ до $w$. Следователно допускането е погрешно и $D'$ също няма цикли. □



Обратното твърдение е безсмислено, защото графите, които определяме като дърветa нямат корен, но в следващия раздел ще видим, че всяко дърво можем да предефинираме като кореново, избирайки произволен от върховете му за корен.

Редица свойства на кореновите дърветa се доказват с елементарни индукционни разсъждения. Например

**Теорема 3.2.2** *Нека $D(V,E)$ е дърво с корен $r$. Тогава $|V| = |E| + 1$.*

***Доказателство.*** а) За тривиалното кореново дърво $D(V = \{r\}, E = \{\})$ имаме $|V| = 1, |E| = 0$ и следователно $|V| = |E| + 1$.

б) Нека $D(V,E)$ е кореново дърво и $|V| = |E| + 1$.

в) Присъединяваме $w$ към върха $u \in V$ и получаваме кореновото дърво $D'(V',E'), V' = V \cup \{w\}, E' = E \cup \{(u,w)\}$. Сега $|V'| = |V|+1, |E'| = |E| + 1$. Следователно $|V'| - |E'| = |V| - |E| = 1$ и $|V'| = |E'| + 1$. □

**Теорема 3.2.3** *Нека $D(V,E)$ е кореново дърво. Съществува единствен път между всеки два върха на $D$.*

***Доказателство.*** Съществуването на поне един път между всеки два върха на кореновото дърво е следствие от свързаността му. Ако допуснем наличието на два различни пътя между върховете $v_i$ и $v_j$, ще получим противоречие с отсъствието на цикли в $D$, тъй като тези пътища непременно образуват поне един цикъл. Наистина, вървейки от $v_i$ към $v_j$, да пропуснем началните върхове, в който двата пътя съвпадат, докато достигнем до последния такъв връх $v_l \neq v_j$ (това е възможно, защото двата пътя са различни). След това продължаваме по един от двата пътя до първия, различен от $v_l$ общ връх $v_m$ (такъв непременно има – например, общият край $v_j$ на двата пътя). Частите от двата пътя между $v_l$ и $v_m$ образуват цикъл. □

***Дефиниция.*** Дължината на пътя от корена $r$ до върха $v_i$ в кореновото дърво $D$ наричаме *височина* на $v_i$. Максималната височина на връх на кореновото дърво $D$ наричаме *височина* на това дърво.

Височината на дървото от Фиг. 3.5.б е 3 (дължината на пътя от 1 до 8).

***Дефиниция.*** Максималният брой синове на един връх от кореновото дърво $D$ наричаме *разклоненост* на това дърво.

Разклонеността на дървото от Фиг. 3.5.б е 3, тъй като има връх с 3 сина (например, коренът 1) и няма връх с повече синове.

**Теорема 3.2.4** *Нека $D(V,E)$ е кореново дърво с височина $h$ и разклоненост $m$. Тогава $D$ има не повече от $m^h$ листа.*

***Доказателство.*** Доказателството ще направим с индукция по височината на дървото.

а) Нека $h = 1$. Тогава всеки връх, различен от корена, е лист на дървото и син на корена. Тъй като разклонеността е $m$, коренът има точно $m$ сина, които са листа на дървото. Следователно, дървото има $m = m^1 = m^h$ листа.

б) Допускаме, че при височина $h$ твърдението е в сила, т.е. броят на листата в кореново дърво с височина $h$ и разклоненост $m$ не надхвърля $m^h$.

в) Нека кореновото дърво $D$ е с височина $h+1$. Да разгледаме неговия подграф $D'$, индуциран от върховете, които не са листа. Този подграф е дърво с височина $h$ и има съгласно индукционното допускане $\leq m^h$ листа. Всеки лист на $D'$ има $\leq m$ сина в $D$ и следователно броят на листата в дървото с височина $h+1$ не надхвърля $m^h.m = m^{h+1}$ (съгласно Принципа на умножението). □

За представяне на кореновите дървета можем да използваме всички представяния за графи. Кореновите дърветата обаче са специален вид графи. В тях за всеки връх $w$, с изключение на корена, има единствен връх $u$ към който е присъединен, затова върхът $u$ можем да наречем баща на $w$, а за $w$ казваме, че е син на $u$. Едно от най-ефективните представяния на дърво с корен е чрез *списък на бащите*. За корена, който няма баща, в списъка поставяме някой несъществуващ връх. Кореновото дърво от Фиг. 3.5.б е представено на Фиг. 3.6 със списък на бащите в едномерен масив $p$ с осем елемента. Като баща на корена 1 в $p[1]$ е посочен несъществуващият връх 0.

| $p[i]$ | 0 | 1 | 1 | 1 | 3 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|---|
| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Фигура 3.6: Списък на бащите в кореново дърво.

***Дефиниция.*** Кореново дърво с разклоненост 2 наричаме *двоично дърво*.

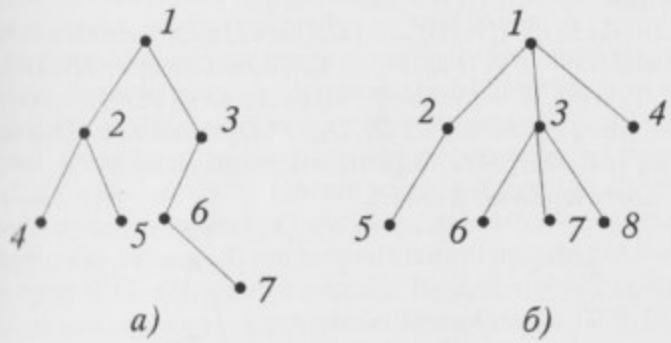
В двоичното дърво всеки връх, който не е лист, има не повече от два сина. Без ограничение на общността можем да наредим двата сина, като първия в наредбата наречем (условно, разбира се) *ляв син*, а втория – *десен син*. На Фиг. 3.7.а е показано двоично дърво със 7 върха. Върхът 2 е ляв, а 3 – десен син на 1. Върхът 3 има само ляв, а върхът 6 – само десен син. Двоичните дървета с наредба на синовете се представят по сравнително прост и хомогенен начин, който наричаме *списък на синовете*. Списъкът на синовете съдържа за всеки връх на кореновото дърво



наредената двойка от синовете му, като отсъствието на съответен син се означава с някакъв специален знак, например $\star$. Двоичното дърво от Фиг. 3.7.а представяме със следния списък на синовете:

1 : 2 3
2 : 4 5
3 : 6 $\star$
4 : $\star$ $\star$
5 : $\star$ $\star$
6 : $\star$ 7
7 : $\star$ $\star$

Аналогично можем да дефинираме *$m$-ично дърво*, като кореново дърво с разклоненост $m$. На Фиг. 3.7.б е показано троично дърво. $m$-ично дърво с наредба на синовете можем да представим чрез списък от наредените $m$-торки от синове на всеки връх. С нарастването на $m$ нараства и възможността много от синовете да липсват, затова при такива дървета посоченото по-горе представяне е неефективно.




Фигура 3.7: Двоично (а) и троично (б) дървета.

***Дефиниция.*** Нека в кореновото дърво $D$ е въведена наредба на синовете за всеки връх. Наредената съвкупност от всички синове на даден връх ще наричаме *братство*. Първият в братството син на всеки не лист наричаме негов *най-ляв син*. Съседът отдясно на даден връх от братството наричаме *десен брат*. Последният в братството връх няма десен брат.

За коренови дървета с по-голяма разклоненост и много липсващи синове често се използува хомогенното представяне, наречено *най-ляв*

*син – десен брат*. Това представяне е списък от наредени двойки, по една за всеки връх на кореновото дърво, като първи елемент на двойката е най-левият син на съответния връх, а втори – неговият десен брат. И в този случай отсъствието на съответния син или брат отбелязваме с $\star$. За дървото от Фиг. 3.7.б получаваме следното представяне:

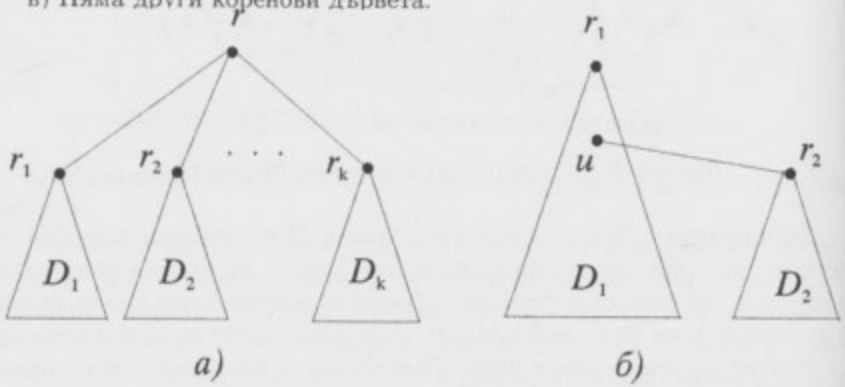
$$\begin{array}{ll} 1: & 2 \ \star \\ 2: & 5 \ 3 \\ 3: & 6 \ 4 \\ 4: & \star \ \star \\ 5: & \star \ \star \\ 6: & \star \ 7 \\ 7: & \star \ 8 \\ 8: & \star \ \star \end{array}$$

Дърветата с корен се използват често в програмистката практика, тъй като са естествени модели на различни информационни явления и процеси. Затова ще си позволим да дадем още някои еквивалентни дефиниции на понятието кореново дърво.

***Дефиниция.*** а) $D(\{r\}, \{\})$ е дърво с корен $r$.

б) Нека $D_1(V_1, E_1), D_2(V_2, E_2), \ldots, D_k(V_k, E_k)$ са дърветата с корени $r_1$, $r_2$, ..., $r_k$ и върхът $r \notin V_1 \cup V_2 \cup \cdots \cup V_k$. Тогава графът $D(V, E), V = V_1 \cup V_2 \cup \cdots \cup V_k \cup \{r\}$, $E = E_1 \cup E_2 \cup \cdots \cup E_k \cup \{(r, r_1), (r, r_2), \ldots, (r, r_k)\}$ е също дърво с корен $r$. Листата на $D_1, D_2, \ldots, D_k$ са листа на $D$, а самите $D_1, D_2, \ldots, D_k$ (вж. Фиг. 3.8.а) наричаме *поддърветата* на $D$.

в) Няма други коренови дървета.




Фигура 3.8: Коренови дървета с поддървета.

***Дефиниция.*** а) $D(\{r\}, \{\})$ е дърво с корен $r$.



б) Нека $D_1(V_1, E_1)$ и $D_2(V_2, E_2)$ са дърветата с корени $r_1$ и $r_2$ съответно и $u \in V_1$. Тогава графът $D(V_1 \cup V_2, E_1 \cup E_2 \cup \{(u, r_2)\})$ е дърво с корен $r_1$. Листа на $D$ са листата на $D_1$ и $D_2$ (без $u$, разбира се, ако е бил лист на $D_1$). Дървото $D_2$ (вж. Фиг. 3.8.б) наричаме поддърво на $D_1$.

в) Няма други коренови дървета.

Оставяме на читателя да докаже еквивалентността на последните две дефиниции с началната дефиниция на кореново дърво.

***Дефиниция.*** Нека $G(V, E)$ е граф, а $D(V, E'), E' \subseteq E$ е дърво. Тогава $D$ наричаме *покриващо дърво* на $G$.

Не всеки граф има покриващо дърво. Ще покажем това в следната

**Теорема 3.2.5** *Графът $G(V, E)$ има покриващо дърво тогава и само тогава, когато е свързан.*

***Доказателство.*** 1) Нека $G(V, E)$ има покриващо дърво $D(V, E')$. Това означава, че в $D$ има път между всеки два върха от $V$. Но $E' \subseteq E$ и всеки път в $D$ е път и в $G$. Следователно $G$ е свързан.

2) Нека $G(V, E)$ е свързан. Ще покажем, че $G$ има покриващо дърво. За целта да разгледаме следната процедура: Докато в разглеждания граф има цикъл, изхвърляме ребро от този цикъл.

Процедурата завършва след краен брой стъпки, защото на всяка стъпка от текущия граф се отстранява ребро, а ребрата са краен брой. Ще докажем, че след всяка стъпка полученият граф остава свързан. Да допуснем противното, т.е. че отстраненото ребро $(v_k, v_l)$ разрушава път, свързващ върховете $v_i$ и $v_j$. Но сега вместо реброто $(v_k, v_l)$ можем да използваме остатъка от цикъла (или подходяща негова част) и да получим друг път между $v_i$ и $v_j$. Следователно, когато процедурата завърши, ще имаме граф $G'(V, E')$, който е свързан. Но процедурата завършва, когато в $G'$ не е останал нито един цикъл. Следователно $G'(V, E')$ е дърво и това е покриващо дърво на $G$. □

Процедурата от доказателството на Теорема 3.2.5 е приложима и когато графът не е свързан. Тя води до построяването на граф без цикли, когото наричаме *гора от покриващи дървета*, като всяко дърво на тази гора покрива една свързана компонента на графа.

## 3.3 Обхождане на графи

В общия случай различните задачи върху графи изискват специфичен алгоритъм. Съществуват обаче техники, които водят до построяване на

сходни алгоритми за задачи, които на практика съществено се различават. Такива техники ще наричаме *алгоритмични схеми*. Под *обхождане на граф* ще разбираме процедура, която систематически, по определени правила, „посещава" (разглежда) всички върхове на графа. В този раздел се спираме на два вида обхождане, които могат да се използват като алгоритмични схеми. Ще разгледаме и обхождания, които нямат качествата на алгоритмична схема, а са специфични алгоритми.

**Обхождане в ширина.** Съществено за тази алгоритмична схема е понятието *ниво на обхождане*, което представлява подмножество на множеството от върховете на свързания граф $G(V,E)$. В резултат на обхождането в ширина, $V$ се разбива на нива $L_0, L_1, \ldots, L_k$ по следния начин (по-долу, под „обхождане" на връх, разбираме някакви специфични действия, зависещи от конкретната задача):

1) Избираме *начален връх* $r$ на обхождането (началният връх може и да е зададен предварително). Образуваме $L_0 = \{r\}$ и нека $l = 0$. Обявяваме върха $r$ за „обходен".

2) Ако $L_0 \cup L_1 \cup \ldots \cup L_l = V$, тогава обхождането е завършено. В противен случай преминаваме към 3.

3) Нека сме обходили върховете от нивата $L_0, L_1, \ldots, L_l$. Образуваме нивото $L_{l+1}$ от всички необходени върхове $v_i \notin L_0 \cup L_1 \cup \ldots \cup L_l$, които са съседи на върхове от $L_l$. Обявяваме върховете от $L_{l+1}$ за „обходени". Нека $l = l + 1$ и преминаваме към 2.

За илюстрация на схемата търсене в ширина, ще построим алгоритъм за намиране на покриващо дърво на свързан граф. В този случай под обхождане на връх ще разбираме присъединяването му към дървото, съставено от по-рано обходените върхове.

И така, нека $G(V,E)$ е свързан граф и без всякакви ограничения върхът $r$ е избран за корен на бъдещото покриващо дърво.

**Алгоритъм за построяване на покриващо дърво (в ширина).**

**Дадено:** Свързан граф $G(V,E)$.

**Резултат:** Покриващо дърво $D(V,E')$ на $G$.

**Процедура:** 1) Коренът $r$ на покриващото дърво ще изберем за начален връх на обхождането. Затова $L_0 = \{r\}$. Образуваме дървото $D_0(V_0, E_0), V_0 = L_0, E_0 = \emptyset$ и $l = 0$.

2) Ако $V_l = V$ – край, търсеното покриващо дърво е $D_l$. В противен случай преминаваме към 3.

3) Нека $D_l(V_l, E_l)$ е дървото, построено след $l$-тата стъпка и $L_l$ са върховете от $l$-то ниво. Образуваме следващото ниво $L_{l+1} = \{v | v \notin V_l, \exists w \in L_l, (w,v) \in E\}$. Образуваме дървото $D_{l+1}(V_{l+1}, E_{l+1})$, $V_{l+1} = V_l \cup L_{l+1}$,

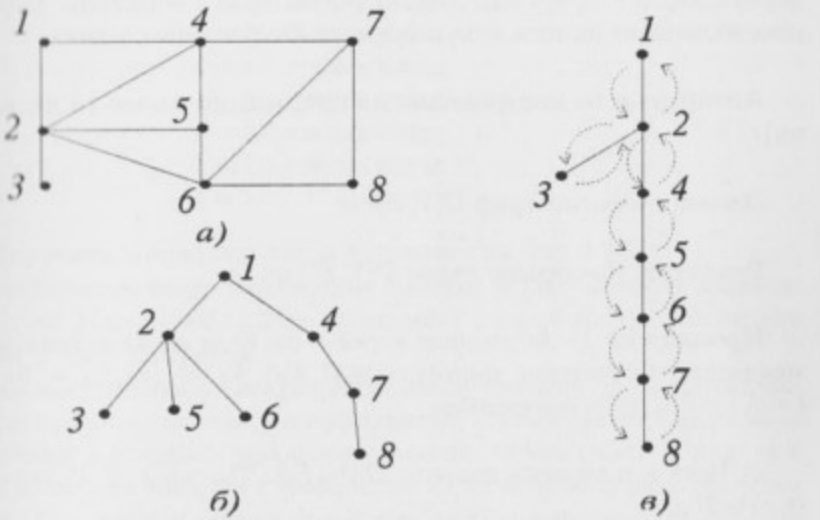


$E_{l+1} = E_l \cup \{(w,v) | w \in L_l, v \in L_{l+1}\}$ с присъединяване на всеки $v \in L_{l+1}$ към съответния $w \in L_l$. Нека $l = l + 1$ и преминаваме към 2. □

Ще приложим алгоритъма към графа $G(V,E)$ на Фиг. 3.9.а, за да построим покриващо дърво с корен 1. Последователно получаваме:

$l = 0$ $L_0 = \{1\}$ $E_0 = \{\}$
$l = 1$ $L_1 = \{2,4\}$ $E_1 = \{(1,2),(1,4)\}$
$l = 2$ $L_2 = \{3,5,6,7\}$ $E_2 = \{(2,3),(2,5),(2,6),(4,7)\}$
$l = 3$ $L_3 = \{8\}$ $E_3 = \{(7,8)\}$

На Фиг. 3.9.б е показано съответното покриващо дърво




Фигура 3.9: Покриващи дървета (б) в ширина и (в) в дълбочина на свързан граф (а).

**Обхождане в дълбочина.** При тази схема основни са понятията *текущ връх* $t$ и *баща* на върха $t$ – $p(t)$. Началният връх на обхождането $r$ и тук е зададен предварително или се определя произволно. Обхождането става по следния начин:

1) „Обхождаме" началния връх $r$. При това $t = r$, а $p(t)$ е неопределен.

2) Търсим необходен връх, който е съседен на текущия $t$:

а) ако има такъв връх $v$, тогава („стъпка напред"): $p(v) = t$, $t = v$ и обявяваме $v$ за „обходен". Преминаваме към 2.

б) ако няма такъв връх:

б.1) ако $t \neq r$, $p(t)$ е определен. Тогава $t = p(t)$ и преминаваме към 2 („стъпка назад");

б.2) ако $t = r$ обхождането е завършило.

С други думи схемата за обхождане в дълбочина ни предписва да опитваме стъпки напред (в дълбочина) докато това е възможно, а в случай на невъзможност, да направим стъпка назад и отново да опитаме стъпка напред. Затова тази алгоритмична техника често е наричана *обхождане с връщане*.

Да приложим схемата към задачата за построяване на покриващо дърво с корен $r$. И тук под „обхождане" на връх е естествено да разбираме включване на този връх в дървото. Получаваме следния

**Алгоритъм за построяване на покриващо дърво (в дълбочина).**

**Дадено:** Свързан граф $G(V, E)$.

**Резултат:** Покриващо дърво $D(V, E')$ на $G$.

**Процедура:** 1) Зададеният корен $r$ ще бъде начален връх на обхождането. Образуваме дървото $D_0(V_0, E_0)$, $V_0 = \{r\}, E_0 = \emptyset$. Нека $t = r, i = 0$ и $p(t)$ е неопределен.

2) Нека е построено дървото $D_i(V_i, E_i)$. Търсим $v \notin V_i$ такъв, че $(t, v) \in E$ и:

а) ако има такъв, построяваме дървото $D_{i+1}(V_{i+1}, E_{i+1})$, $V_{i+1} = V_i \cup \{v\}, E_{i+1} = E_i \cup \{(t, v)\}$. Сега $p(v) = t, t = v, i = i + 1$ и преминаваме към 2.

б) в противен случай:

б.1) ако $t \neq r$, тогава $t = p(t)$ и преминаваме към 2.

б.2) иначе – край. Търсеното покриващо дърво е $D_i$.

За графа от Фиг. 3.9.а да построим покриващо дърво с начален връх 1 чрез обхождане в дълбочина. Последователно получаваме следните върхове (когато имаме няколко възможности за $v$ избираме върха с най-



малък номер):

| | | |
|---|---|---|
| $t = 1$ | стъпка напред | $(1, 2)$ |
| $t = 2$ | стъпка напред | $(2, 3)$ |
| $t = 3$ | стъпка назад | |
| $t = 2$ | стъпка напред | $(2, 4)$ |
| $t = 4$ | стъпка напред | $(4, 5)$ |
| $t = 5$ | стъпка напред | $(5, 6)$ |
| $t = 6$ | стъпка напред | $(6, 7)$ |
| $t = 7$ | стъпка напред | $(7, 8)$ |
| $t = 8$ | стъпка назад | |
| $t = 7$ | стъпка назад | |
| $t = 6$ | стъпка назад | |
| $t = 5$ | стъпка назад | |
| $t = 4$ | стъпка назад | |
| $t = 2$ | стъпка назад | |
| $t = 1$ | край | |

Полученото покриващо дърво е показано на Фиг. 3.9.в.

Алгоритмите, които разгледахме по-горе, строят коренови дървета. Така всяко дърво може да бъде превърнато в кореново, чрез обхождане с някой от тези алгоритми.

Забележете принципните различия между двата подхода. обхождането в ширина дава по-ниски и по-разклонени, а обхождането в дълбочина – по-високи и по-малко разклонени дървета. Обхождането в ширина е неудобно поради това, че е необходимо да се помнят всички обходени върхове (или поне върховете от последното обходено ниво), за да може да се построи следващото ниво. За обхождането в дълбочина са достатъчни само върховете, разположени на пътя от началния до текущия връх. Затова при ограничена компютърна памет за предпочитане са алгоритмите, построени по схемата в дълбочина.

От друга страна, ако обхождането се прави с цел да се намери връх със зададени свойства и той се окаже на ниво с малък номер, тогава алгоритъмът в ширина ще го намери много бързо. Търсенето в дълбочина ще работи непредсказуемо дълго, защото може да се окаже, че върхът е измежду последните обходени (дори и да е на сравнително ниско ниво). При такъв род задачи за предпочитане е схемата в ширина.

Като компромис на двете схеми могат да се разработят *хибридни обхождания*, съчетаващи добрите страни на едната и другата схема, и неутрализиращи, доколкото е възможно, недостатъците им. Например, бързо спускане в дълбочина и след това обхождане в ширина на малкото

оставащи върхове от „дъното“. Предлагаме на читателя сам да уточни подробностите на тази или друга хибридна схема.

**Ойлерови обхождания.**

**Дефиниция.** *Ойлеров път* в свързания мултиграф $G(V, E)$ наричаме път, който минава еднократно през всяко ребро на мултиграфа. Ако Ойлеровият път има начало и край, които съвпадат, тогава той се нарича *Ойлеров цикъл*. Мултиграф, ребрата на който образуват Ойлеров цикъл, наричаме *Ойлеров*.

В сила е следната

**Теорема 3.3.1 (Л. Ойлер)** *Свързаният мултиграф $G(V, E)$ е Ойлеров тогава и само тогава, когато всеки връх на $G$ е с четна степен.*

***Доказателство.*** 1) Нека ребрата на $G(V, E)$ образуват Ойлеров цикъл. Тогава всеки връх има четна степен, защото при обхождане на мултиграфа по Ойлеровия цикъл, на всяко ребро, „влизащо“ във връх $v_i$ съответства ребро (различно от първото, според дефиницията за път), „излизащо“ от $v_i$, а цикълът съдържа всички ребра точно по един път.

2) Нека мултиграфът $G(V, E)$ е свързан и такъв, че всеки връх има четна степен. Ще опишем процедура, която построява Ойлеров цикъл от всички ребра на мултиграфа.

Започваме със следните стъпки:

а) Нека $r$ е произволен връх на мултиграфа. Обявяваме го за текущ $t$.

б) Докато е възможно, строим път по следното правило: от върха $t$, в който се намираме, преминаваме към някой съседен $v$ по необходено ребро. Използваното ребро обявяваме за обходено, а върхът $v$ за текущ.

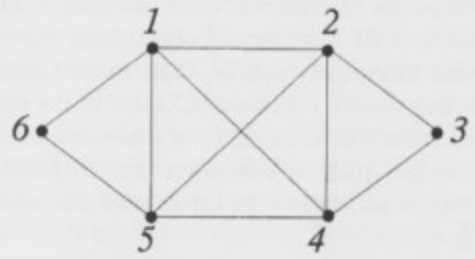
Да допуснем, че когато правилото от стъпка (б) не е вече приложимо, то $t \neq r$. Тогава ще се окаже, че степента на $t$ е нечетна, тъй като влизайки за последен път в $t$ сме отбелязали като обходено едно ребро, а при всяко предно минаване през този връх сме отбелязвали като обходени по две негови ребра. Това е противоречие. Следователно, когато правилото от стъпка (б) не е вече приложимо, $t = r$ и всички обходени на този етап ребра образуват цикъл $C$.

в) Ако цикълът $C$ съдържа всички ребра – край на обхождането. Построен е Ойлеров цикъл.

г) В противен случай премахваме от мултиграфа ребрата на намерения цикъл. Полученият мултиграф е такъв, че всеки връх е с четна степен. Търсим в построения цикъл връх $r'$, от който излиза поне едно необходено ребро. Ако допуснем, че такъв връх няма, при условието, че не всички ребра са обходени, ще получим противоречие със свързаността на мултиграфа. Повтаряме действията от стъпка (б) с начален връх $r = r'$ и построяваме нов цикъл. Сега разкъсваме двата цикъла в общия им връх $r'$ и с отъждествяването на получените краища два по два, получаваме нов цикъл $C$ и преминаваме към (в).

Алгоритъмът ще завърши работа, когато всички ребра се изчерпят и построява Ойлеров цикъл. Следователно мултиграфът е Ойлеров. □

За Ойлеровия граф на Фиг. 3.10 построяваме първо цикъла 1, 2, 3, 4, 5, 6, 1. В него 1 е първият връх, от който излиза необходено ребро, и алгоритъмът построява нов цикъл 1,5,2,4,1. Разкъсваме в 1 двата цикъла и отъждествявайки краищата им получаваме 1,5,2,4,1,2,3,4,5,6,1 което е търсеният Ойлеров цикъл.




Фигура 3.10: Ойлеров граф.

**Следствие 3.3.1** *Свързаният мултиграф $G(V, E)$ съдържа Ойлеров път тогава и само тогава, когато има точно 2 върха с нечетна степен.*

***Доказателство.*** 1) Нека $v_i$ и $v_j$ са върховете с нечетна степен. Добавяме в графа ребро $(v_i, v_j)$. Полученият мултиграф е Ойлеров и следователно можем да построим Ойлеров цикъл. Отстраняваме добавеното ребро и получаваме път от $v_i$ до $v_j$, който съдържа всички ребра на графа $G(V, E)$ точно по веднъж и следователно е Ойлеров път.

2) Нека ребрата на $G(V, E)$ образуват Ойлеров път от $v_i$ до $v_j$. Добавяме ребро $(v_i, v_j)$ и този път се превръща в Ойлеров цикъл за новополучения мултиграф и следователно в него всички върхове са с четна степен. Добавянето на реброто $(v_i, v_j)$ е увеличило с 1 само степените на $v_i$ и $v_j$. Следователно в $G$ всички върхове са с четна степен, с изключение на тези два върха. □

Дефинициите, които дадохме за Ойлерови мултиграфи, се пренасят лесно в ориентирани мултиграфи. За тях имаме аналогичните

**Теорема 3.3.2** *Крайният ориентиран мултиграф $G(V, E, f_G)$ е Ойлеров т.с.т.к. за всеки връх полустепените на входа и изхода съвпадат.*

и

**Следствие 3.3.2** *Крайният ориентиран мултиграф $G(V, E, f_G)$ съдържа Ойлеров път т.с.т.к само за два от върховете му полустепените на входа и изхода не съвпадат, като в единия връх полустепента на изхода е с единица по-голяма от полустепента на входа, а при другия обратно.*

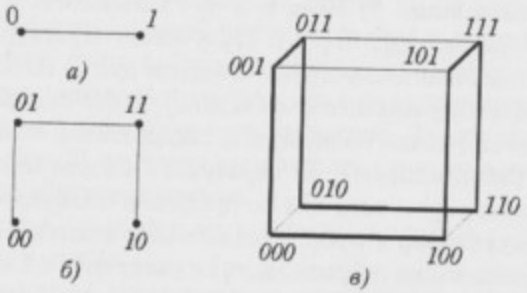
**Хамилтонови обхождания.**

**Дефиниция.** *Хамилтонов път* в свързания граф $G(V, E)$ наричаме път, минаващ еднократно през всеки връх на графа. Ако началото и краят на пътя съвпадат, той се нарича *Хамилтонов цикъл*. Граф, който съдържа Хамилтонов цикъл, се нарича *Хамилтонов граф*.

На пръв поглед понятията Ойлеров и Хамилтонов граф се различават малко. Всъщност това не е така. Не е известно необходимо и достатъчно условие за един граф да е Хамилтонов. В резултат на това не е известен бърз алгоритъм, който да проверява дали зададен свързан граф е Хамилтонов.

Тук ще разгледаме интересен клас графи, за които лесно се вижда, че са Хамилтонови.

***Дефиниция.*** Графът $\mathcal{B}_n(J_2^n, E_n)$ с върхове $n$-мерните двоични вектори и ребра $E_n = \{(\alpha_i, \alpha_j) | \rho(\alpha_i, \alpha_j) = 1\}$ наричаме *$n$-мерен двоичен куб*. На Фиг. 3.11 са показани 1-мерният, 2-мерният и 3-мерният двоични кубове. Очевидно $\mathcal{B}_n$ има $2^n$ върха. (Пресметнете броя на ребрата му!)




Фигура 3.11: 1-мерен (а), 2-мерен (б) и 3-мерен (в) двоични кубове.



**Теорема 3.3.3** *Графът $\mathcal{B}_n, n \geq 2$ е Хамилтонов.*

***Доказателство.*** Твърдението ще докажем с индукция по $n$.

а) При $n = 2$ то е очевидно. Пътят $00, 01, 11, 10, 00$ е Хамилтонов цикъл.

б) Допускаме верността на твърдението за $n$ и нека $\alpha_0, \alpha_1, \ldots, \alpha_{2^n-1}$, $\alpha_0$ е Хамилтонов цикъл в $\mathcal{B}_n$.

в) Построяваме следната последователност от върхове на $\mathcal{B}_{n+1}$: $0\alpha_0$, $0\alpha_1$, ..., $0\alpha_{2^n-1}$, $1\alpha_{2^n-1}$, $1\alpha_{2^n-2}$, ..., $1\alpha_0$, $0\alpha_0$. Очевидно $0\alpha_i$ и $0\alpha_{i+1}$, при $i = 0, 1, \ldots, 2^n - 2$ се различават в една позиция и следователно образуват ребра на $\mathcal{B}_n$. Аналогично, ребра са и $(1\alpha_i, 1\alpha_{i-1})$, при $i = 2^n - 1, 2^n - 2, \ldots, 1$, а така също $(0\alpha_{2^n-1}, 1\alpha_{2^n-1})$ и $(1\alpha_0, 0\alpha_0)$. Следователно посочената последователност е цикъл. Той минава точно по един път през всеки от върховете на $\mathcal{B}_n$, защото от индуктивното предположение имаме, че $\alpha_0, \alpha_1, \ldots, \alpha_{2^n-1}$ са всички различни върхове на $\mathcal{B}_n$ и следователно $0\alpha_0$, ..., $0\alpha_{2^n-1}$, $1\alpha_{2^n-1}$, ..., $1\alpha_0$ са всички различни върхове на $\mathcal{B}_{n+1}$. Следователно цикълът е Хамилтонов. □

Забележете, че $\mathcal{B}_1$ не съдържа Хамилтонов цикъл, защото има само едно ребро, но все пак съдържа Хамилтонов път.

***Дефиниция.*** Последователността от вектори на $J_2^n$ такава, че всеки два съседни вектора (включително първият и последният) се различават в една позиция се нарича *двоичен код на Грей*.

Конструкцията на Хамилтонов път от Теорема 3.3.3 е един от възможните начини за построяване на двоичния код на Грей – така наречения *двоично-отразен код на Грей*. На Фиг. 3.11.в с удебелени ребра е показан Хамилтоновият цикъл (двоично-отразеният код на Грей) за $n = 3$ : 000,001,011,010,110, 111,101,100,000.

## 3.4 Оптимизационни задачи в графи

Крайните графи пораждат многобройни задачи за намиране на максимуми или минимуми на функции, които наричаме *оптимизационни*. Измежду тях се намират много „трудни“ – такива, за които не са известни бързи алгоритми за решаването им. Тук ще разгледаме няколко задачи, за които съществуват такива алгоритми. Ще видим, че всеки качествен алгоритъм се опира на някакви полезни математически свойства на разглежданите обекти.

**Оптимално покриващо дърво.**

***Дефиниция.*** Нека $G(V, E)$ е свързан граф и $c : E \to R$ е функция с реални стойности, дефинирана по ребрата на графа. Стойността

$c(e), e \in E$, наричаме *цена* или *тегло* на реброто $e$. Нека $D(V, E')$ е покриващо дърво на $G(V, E)$. *Цена (тегло)* на дървото наричаме сумата $c(D) = \sum_{e \in E'} c(e)$. Покриващото дърво $D_0(V, E_0)$ на $G(V, E)$ наричаме *минимално (максимално)*, ако $c(D_0) \leq c(D)$ $(c(D_0) \geq c(D))$ за всяко друго покриващо дърво $D$ на $G$.

Понятията минимално и максимално покриващо дърво са аналогични, затова ще говорим за *оптимално покриващо дърво*, означавайки с това понятие кое да е от двете. Разглежданията ще правим само за минимално покриващо дърво. Аналогични резултати за случая на максимално покриващо дърво се получават, ако се замени min с max и се обърнат съответните неравенства. Получаваме следната задача: *Даден е свързан граф $G(V, E)$ с тегла на ребрата, определени от $c : E \to R$. Да се построи минимално покриващо дърво (МПД) на $G$.*

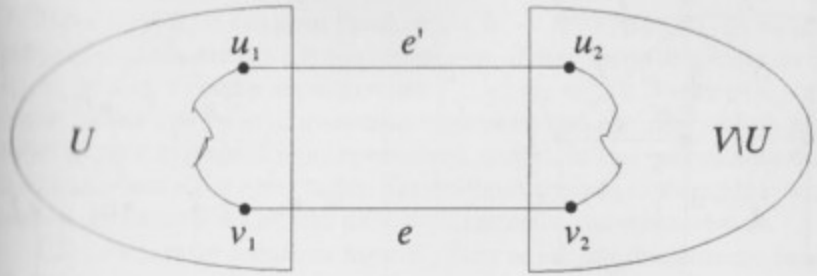
Преди да посочим алгоритми за решаване на тази задача, ще докажем едно важно нейно свойство.

**Теорема 3.4.1 (МПД-свойство)** *Нека $G(V, E)$ е свързан граф с ценова функция на ребрата $c : E \to R$ и $U \subset V, U \neq \emptyset$. Нека $e = (v_i, v_j) \in E$ е такова, че $v_i \in U, v_j \in V \setminus U$ и $e$ има минимално тегло измежду всички такива ребра. Тогава съществува МПД $D_0(V, E_0)$ на $G$, такова че $e \in E_0$.*

***Доказателство.*** Нека е фиксирано някакво $U, U \subset V, U \neq \emptyset$ и нека $e = (v_i, v_j)$ е реброто с минимално тегло, свързващо $U$ и $V \setminus U$. Да допуснем, че то не участва в нито едно МПД. Нека $D(V, E')$ е МПД на $G$. В графа $G'(V, E' \cup \{e\})$ ще има единствен цикъл, в който $e$ участва, защото в $D$ има път от $v_i$ до $v_j$. Този цикъл има поне едно друго ребро $e'(u_1, u_2)$, за което $u_1 \in U, u_2 \in V \setminus U$ (вж. Фиг. 3.12). Действително, ако всички останали ребра бяха с краища само в $U$ или само във $V \setminus U$, нямаше да могат да образуват цикъл. Построяваме дървото $D'(V, E' \cup \{e\} \setminus \{e'\})$, разкъсвайки цикъла, съдържащ $e'$. Но $c(D') = c(D) + c(e) - c(e')$ и от $c(e) \leq c(e')$ следва $c(D') \leq c(D)$. Ако $c(D') = c(D)$ (т.е. $c(e) = c(e')$), тогава $D'$ също е оптимално и съдържа $e$, противно на допускането. Ако $c(D') < c(D)$ (т.е. $c(e) < c(e')$), тогава получаваме противоречие с оптималността на $D$. И в двата случая допускането не е вярно и теоремата е доказана. □

От практическа гледна точка Теорема 3.4.1 ни показва, че при построяването на оптимално покриващо дърво, от всички ребра, свързващи две множества върхове, трябва да изберем това, което има минимална цена (иначе полученото покриващо дърво няма да е оптимално). Кое от






Фигура 3.12: МПД-свойство.

ребрата с най-малка цена ще изберем е без значение, защото независимо от избора ще получим минимално покриващо дърво.

Твърдението на Теорема 3.4.1 ни предоставя *МПД-свойството*, позволяващо да се създадат алгоритми, които бързо намират решение на задачата. Първият алгоритъм, който ще разгледаме, построява дърво със зададен корен $r$.

**Алгоритъм на Прим**

**Дадено:** Свързан граф $G(V, E)$ и функция $c : E \to R$, задаваща тегла на ребрата му.

**Резултат:** Минимално покриващо дърво $D(V, E')$ с корен – зададен връх $r \in V$ на $G$.

**Процедура:**

1) Построяваме дървото $D_0(V0, E_0), V_0 = \{r\}, E_0 = \emptyset$ и $k = 0$.

2) Нека сме построили $D_k(V_k, E_k)$. Търсим реброто $e = (v_i, v_j), v_i \in V_k, v_j \in V \setminus V_k$ с минимално тегло и построяваме $D_{k+1}(V_{k+1}, E_{k+1}), V_{k+1} = V_k \cup \{v_j\}$, $E_{k+1} = E_k \cup \{e\}$ и $k = k + 1$.

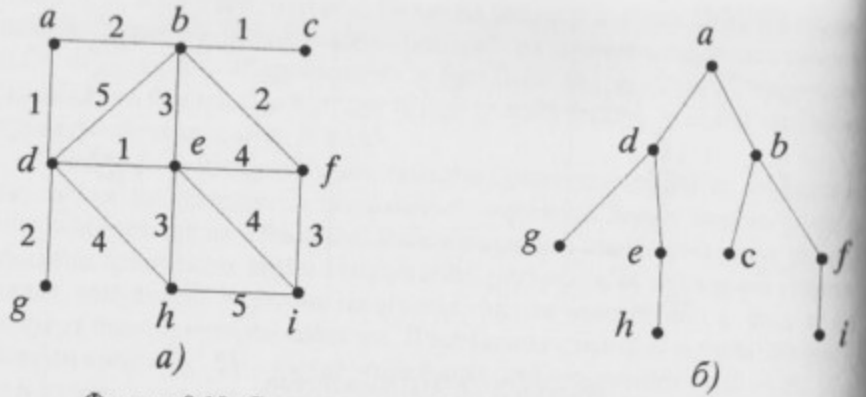
3) Ако $V_k = V$ – край, полученото дърво $D(V, E'), E' = E_k$ е оптималното. Иначе преминаваме към 2. □

За графа от Фиг. 3.13.а алгоритъмът на Прим строи оптималното покриващо дърво с корен $a$, показано на Фиг. 3.13.б, като последователно присъединява върховете $d, e, b, c, f, g, h$ и $i$.

Вторият алгоритъм, който ще разгледаме, построява оптимално покриващо дърво без корен.

**Алгоритъм на Крускал.**

**Дадено:** Свързан граф $G(V, E)$ и функция $c : E \to R$, задаваща тегла на ребрата му.




Фигура 3.13: Свързан граф (а) и едно негово МПД (б).

**Резултат:** Минимално покриващо дърво $D(V, E')$ на $G$.
**Процедура:**

1) Сортираме ребрата на $G$ в нарастващ ред на цената и нека този ред е $e_1, e_2, \ldots, e_m$.

2) От всеки връх $v$ на графа образуваме тривиално дърво $D_v(\{v\}, \emptyset)$.

3) За всяко ребро $e_i = (v_{i_1}, v_{i_2}), i \in I_m$, (по реда, определен от сортирането) правим следното: ако $v_{i_1}, v_{i_2}$ са в различни дървета, $D'(V', E')$ и $D''(V'', E'')$ съответно, съединяваме двете в дървото $D(V' \cup V'', E' \cup E'' \cup \{(v_{i_1}, v_{i_2})\})$. □

Изборът на най-леко ребро за всяка стъпка на алгоритъма на Крускал се гарантира от сортирането на ребрата по нарастващи тегла, т.е. ако едно ребро е по-леко от друго, то се избира по-рано за построяване на дървото. Разбира се, ако двата края на едно ребро са в едно и също поддърво, то не може да участва в минималното дърво (други по-леки ребра вече са осигурили свързването на двата края в бъдещото оптимално дърво).

За графа от Фиг. 3.13.а, при една от възможните наредби на ребрата, алгоритъмът на Крускал включва в оптималното покриващо дърво последователно ребрата $(a, d)$, $(b, c)$, $(d, e)$, $(a, b)$, $(b, f)$, $(d, g)$, $(e, h)$ и $(f, i)$. Получихме отново дървото от фиг. 3.13.б. В общия случай може да се получи и друго оптимално покриващо дърво. Последните стъпки на алгоритъма могат да бъдат спестени, ако се организира броене на използваните ребра и изпълнението на цикъла се прекрати, след като броят им стане равен на $|V| - 1$.



**Най-къс път в граф.**

Нека $G(V, E)$ е свързан граф, а $c : E \to R^+$ – теглова функция на ребрата с положителни реални стойности. *Претеглена дължина* на пътя $v_{i_0}, v_{i_1}, \ldots, v_{i_l}$ в графа ще наричаме $\sum_{j=0}^{l-1} c(v_{i_j}, v_{i_{j+1}})$. Пътят от $v_{i_0}$ до $v_{i_l}$ с най-малка претеглена дължина наричаме *най-къс път* от $v_{i_0}$ до $v_{i_l}$. Естествено е да дефинираме претеглена дължина 0 за тривиалния път от $v$ до $v$ за всеки връх $v$ на графа. Когато няма опасност от недоразумения, можем да казваме дължина вместо претеглена дължина на път.

Ще разгледаме следната задача: *Да се намерят дължините на най-късите пътища (и самите най-къси пътища) от зададен връх $v_0$ до всички останали върхове на свързания граф $G(V, E)$ с теглова функция на ребрата $c : E \to R^+$.*

В частния случай $c(e) = 1, \forall e \in E$, претеглената дължина е равна на дължината на пътя. За този случай решението на задачата се дава от следната

**Теорема 3.4.2** *Нека $G(V, E)$ е свързан граф с теглова функция $c(e) = 1, \forall e \in E$ и $D$ е покриващо дърво на $G$ с корен $v_0$, построено в ширина. Пътищата в $D$ от корена до останалите върхове на $G$ са най-къси пътища от върха $v_0$ до тези върхове.*

***Доказателство.*** С индукция по нивата $L_i, i = 0, 1, \ldots, l$ на построеното в ширина дърво ще докажем, че дължината на минималния път от корена $v_0$ до всеки връх от ниво $L_i$ е точно $i$. Тъй като всеки път от корена $v_0$ до връх от ниво $L_i$ е с дължина $i$, това доказва твърдението на Теоремата.

а) Дължината на най-късия път от $v_0$ до $v_0$ е 0 и тъй като той е единствен връх в $L_0$, твърдението е в сила за $L_0$.

б) Да допуснем верността за нивата $L_0, L_1, \ldots, L_i$.

в) Ще покажем, че най-късите пътища от $v_0$ до всеки връх от $L_{i+1}$ в покриващото дърво са с дължина $i + 1$. Да допуснем, че $\exists v \in L_{i+1}$, такъв че $v_0, \ldots, w, v$ е най-къс път от $v_0$ до $v$ с дължина $k < i + 1$. Тогава $v_0, \ldots, w$ е най-къс път от $v_0$ до $w$ и дължината му е $k - 1 < i$. Съгласно индукционното предположение $w \in L_{k-1}$ и алгоритъмът трябваше да постави $v$ в $L_k, k < i+1$. Но това е противоречие с $v \in L_{i+1}$. Следователно, твърдението е в сила и за $L_{i+1}$. □

По-долу е описан алгоритъм, който по зададен свързан граф $G(V, E)$ с ценова функция $c : E \to R^+$ и връх $v_0$ строи всички най-къси пътища от $v_0$ до останалите върхове на графа. (Оставяме на читателя да прецени какво ще се получи като резултат, ако графът е несвързан.) Пътищата

са представени в *дърво на най-късите пътища* с корен $v_0$ (аналог на покриващото дърво, построено с обхождане в ширина от Теорема 3.4.2), в което единственият път от $v_0$ до $v \in V$ е минимален за графа. Дървото на най-късите пътища представяме като списък от бащи (вж.раздел 3.2).

За опростяване работата на алгоритъма да додефинираме ценовата функция $c$ и върху двойките върхове $v_i, v_j$ такива, че $(v_i, v_j) \notin E$ :

$$c^*(v_i, v_j) = \begin{cases} c(v_i, v_j) & (v_i, v_j) \in E \\ \infty & (v_i, v_j) \notin E \end{cases}$$

Вместо $\infty$ за практически нужди можем да използваме числото $M = \sum_{e \in E} c(e) + \varepsilon, \varepsilon > 0$ – по-голямо от дължината на кой да е път (неповтарящ ребра) в графа. Такива тегла по ребрата ще доведат до това, че никое ребро не може да участва в най-къс път.

Нека $|V| = n+1$ и означим за простота началния връх с 0, а останалите с $1, 2, ..., n$. Ще използваме масивите *dist* и *part* с по $n+1$ елемента. Елементът $dist[i]$ ще съдържа дължината на временно най-късия път от 0 до $i$, а $part[i]$ – бащата (предшестващия връх) на $i$ по този път. На всяка стъпка алгоритъмът намира най-късия път до един от върховете, образувайки множество $U$ от върхове, за които най-късият път е намерен. В началото $U = \{0\}$, защото най-късият път от 0 до 0 е тривиалният път с нулева дължина от 0 до 0.

**Алгоритъм на Дейкстра.**

**Дадено:** Свързан граф $G(V, E)$ с теглова функция по ребрата $c : E \to R^+$ и начален връх $v_0 \in V$.

**Резултат:** Дърво на минималните пътища от $v_0$ до всички останали върхове в $G$.

**Процедура:**

1) Разширяваме $c : E \to R^+$ до $c^* : V \times V \to R^+$, както бе посочено по-горе.

2) Нека $dist[0] = 0, part[0] = -1$ и $U = \{0\}$, а $dist[i] = c^*(0, i)$ и $part[i] = 0$, за $i =\in I_n$.

3) Повтаряме $n-1$ пъти следните стъпки:

3.а) Избираме връх $j \notin U$, за който $dist[j]$ е минимално и $U = U \cup \{j\}$.

3.б) За всеки $k \notin U$ пресмятаме $dist[k] = \min(dist[k], dist[j] + c^*(j, k))$. Ако min е $dist[j] + c^*(j, k)$, тогава $part[k] = j$. □

Както се вижда, този алгоритъм избира върха $j \in V \setminus U$ с минимално временно разстояние $dist[j]$ до началния и обявява това разстояние за глобално минимално. Ще докажем, че алгоритъмът на Дейкстра прави това основателно.



***Дефиниция.*** Нека $G(V, E)$ е граф, $V = \{0, 1, 2, \ldots, n\}$. Нека $U \subset V, U \neq \emptyset$ и $0, i_1, \ldots, i_{k-1}, i_k$ е път в $G$ от 0 до $i_k$ такъв, че $0, i_1, \ldots, i_{k-1} \in U$. Такъв път наричаме *специален път* по отношение на $U$.

**Теорема 3.4.3** *Нека $U \subseteq V$ е получено след поредна стъпка на алгоритъма на Дейкстра. Тогава:*
*I) $\forall i \in U$ в $dist[i]$ е дължината на най-къс път от 0 до $i$ (за $i \neq 0$ в $part[i]$ е бащата на $i$ по този най-къс път).*
*II) $\forall k \notin U$ в $dist[k]$ е дължината на най-къс специален път (по отношение на $U$) от 0 до $k$ (в $part[k]$ е бащата на $k$ по този най-къс път).*

***Доказателство.*** Двете части на теоремата доказваме паралелно с индукция по построяването на $U$.

а) Нека $U = \{0\}$. I) Тривиалният път е най-къс път от 0 до 0, така че твърдението е вярно (0 е корен в бъдещото дърво на най-късите пътища и следователно няма баща).

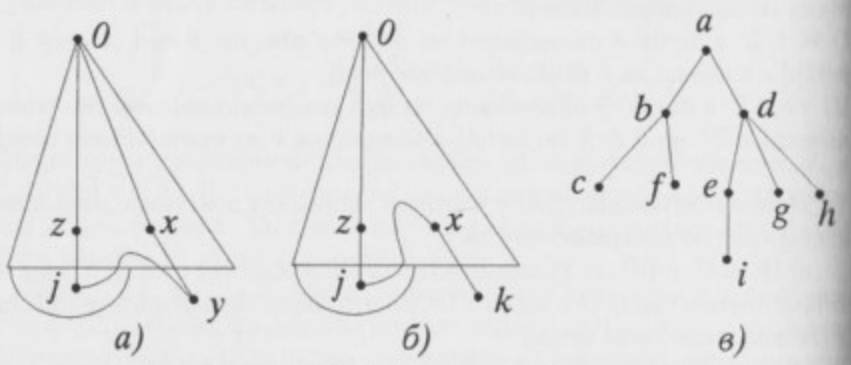
II) Специалният път до $k$ трябва да бъде реброто $(0, k)$, тъй като $U = \{0\}$. В действителност, в този момент $dist[k]$ съдържа дължината на съответното ребро ($\infty$ за върховете, несвързани с 0). Освен това, за всеки връх $k$ в $part[k]$ е посочен като баща 0, което съответства на така получените временно най-къси специални пътища.

б) Да допуснем, че твърденията (I) и (II) са в сила за някое $U$ и съгласно предписаните в 3.а и 3.б действия на алгоритъма на Дейкстра да намерим връх $j$ с исканото свойство, да го добавим към $U$ и да преизчислим $dist[k]$ и $part[k]$.

в) Ще докажем, че твърдението е в сила и при $U' = U \cup \{j\}$. I) За всеки връх $i \in U$ в $dist[i]$ е дължината на най-къс път, съгласно индукционното предположение (I). Да допуснем, че запомненото в $dist[j]$ число не е дължината на най-къс път от 0 до $j$ (съответно, запомненият в *part* път не е най-къс). Но съгласно индукционното предположение (II) това е най-късият специален път до $j$ по отношение на $U$. Следователно най-късият път до $j$ не е специален по отношение на $U$, т. е. поне веднъж напуска $U$ преди да стигне до $j$ (вж. Фиг. 3.14.а). Да означим с $y$ първия връх $\notin U$ по този път, а с $z$ предпоследния връх по запомнения в *part* път.

За дължините на разглежданите по-горе пътища получаваме $dist[j] = c^*(0, \ldots, z, j) > c^*(0, \ldots, x, y, \ldots, j) > c^*(0, \ldots, x, y)$. Първото неравенство следва от това, че специалният път $0, ..., z, j$ не е най-къс, а второто от това, че теглата на ребрата са положителни. Но пътят $0, ..., x, y$ е специален и очевидно трябва да е най-къс специален път до $y$, иначе ще

съществува още по-къс път през $y$ до $j$. Следователно $c^*(0,\ldots,x,y) = dist[y] < dist[j]$. Но това е противоречие с избора на $j$ за разширяване на $U$. Действително, в такъв случай алгоритъмът на Дейкстра щеше да избере $y$ за разширяване на $U$, а не $j$. И така допускането е невярно и специалният път до $j$ (който е запомнен в *part*) е глобално най-къс.




Фигура 3.14: Алгоритъм на Дейкстра.

II) Ще покажем, че преизчисляването на $dist[k]$ за всеки $k \notin U'$ води до намиране дължината на минималния специален път за $k$ (по отношение на $U'$) и този път е запомнен в *part*. Първо ще докажем едно важно монотонно свойство на алгоритъма на Дейкстра: върховете на графа влизат в $U$ в нарастващ ред на дължините на минималните им пътища до 0. В началото в $U$ влиза самия 0 с нулева дължина на минималния път, който е по-къс от всеки друг път. Да допуснем, че свойството е било изпълнено до включването на върха $j$ в $U$. Но $dist[j]$ е минимално, т.е. $dist[j] \leq dist[k], \forall k \notin U$, а преизчисляването или запазва $dist[k]$ или го заменя с $dist[j] + c^*(j,k) > dist[j]$ (заради $c^*(j,k) > 0$). Следователно, след всяка стъпка ще имаме $dist[i] \leq dist[k]$, $\forall i \in U, k \notin U$ и винаги избираният от алгоритъма връх ще има дължина на минималния път до 0 не по-малка от дължините на минималните пътища до 0 на всички върхове от $U$.

Съществуват три възможности за $k \notin U$, след разширяването на $U$ с $j$: запомненият специален път до $k$ да остане най-къс, най-къс да стане специалният път до $k$, за който предпоследен връх е върхът $j$ или най-къс да стане специален път до $k$, минаващ през $j$, но $j$ не е предпоследен връх. Възможността минимален да е специален път, неизползващ $j$, различен от запомнения, се изключва от индукционното предположение (II).



Първите две възможности се разглеждат от алгоритъма на Дейкстра. Да допуснем, че в сила е третата възможност (вж Фиг. 3.14.б) и нека $z \neq j$ е предпоследният връх от $U$ по този най-къс специален път до $k$. Но тогава $c^*(0,\ldots,z,\ldots,j) < c^*(0,\ldots,z,\ldots,j,\ldots,x)$, защото теглата на ребрата са положителни. Пътищата $0,\ldots,z,\ldots,j$ и $0,\ldots,z,\ldots,j,\ldots,x$ са най-къси за $j$ и $x$, защото в противен случай щеше да има по-къс специален път до $k$. Това е в противоречие с доказаното монотонно свойство, защото алгоритъмът на Дейкстра е избрал $x$ преди $j$. □

За пример да разгледаме графа от Фиг. 3.13.а и да намерим най-късите пътища от $a$ до всички останали върхове. В Таблицата 3.1 са дадени последователните значения на масивите *dist* и *part*, индексирани с върховете на графа $\neq a$. За върховете, влезли в $U$, сме престанали да попълваме съответните елементи на масивите.

*dist*:

| $b$ | $c$ | $d$ | $e$ | $f$ | $g$ | $h$ | $i$ |
|---|---|---|---|---|---|---|---|
| 2 | $\infty$ | 1 | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| 2 | $\infty$ | | 2 | $\infty$ | 3 | 5 | $\infty$ |
| | 3 | | 2 | 4 | 3 | 5 | $\infty$ |
| | 3 | | | 4 | 3 | 5 | 6 |
| | | | | 4 | 3 | 5 | 6 |
| | | | | 4 | | 5 | 6 |
| | | | | | | 5 | 6 |
| | | | | | | | 6 |

*part*:

| $b$ | $c$ | $d$ | $e$ | $f$ | $g$ | $h$ | $i$ |
|---|---|---|---|---|---|---|---|
| $a$ | $a$ | $a$ | $a$ | $a$ | $a$ | $a$ | $a$ |
| $a$ | $a$ | | $d$ | $a$ | $d$ | $d$ | $a$ |
| | $b$ | | $d$ | $b$ | $d$ | $d$ | $a$ |
| | $b$ | | | $b$ | $d$ | $d$ | $e$ |
| | | | | $b$ | $d$ | $d$ | $e$ |
| | | | | $b$ | | $d$ | $e$ |
| | | | | | | $d$ | $e$ |
| | | | | | | | $e$ |

Таблица 3.1: Намиране на минимални пътища с алг. на Дейкстра.

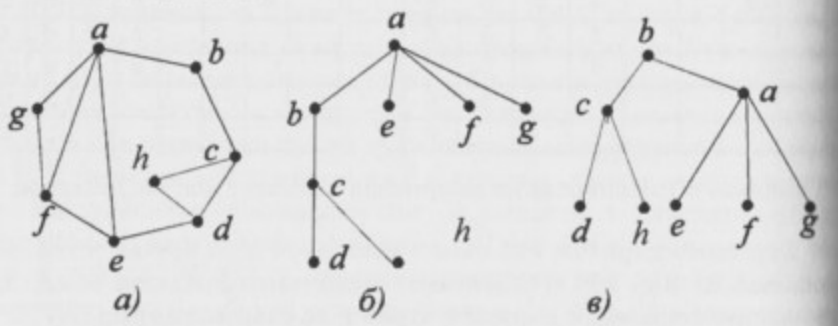
Търсеното дърво на най-късите пътища от $a$ до всички останали е показано на Фиг. 3.14.в. Забележете съществената разлика между минималните покриващи дървета с корен $r$ на свързания граф $G(V,E)$ и дървото на най-късите пътища от $r$ до всички останали върхове на този граф.

**Лакоми алгоритми и матроиди**

По-горе разгледахме три алгоритъма, изградени по една и съща схема за построяване на алгоритми, решаващи оптимизационни задачи. Тази схема образно е наречена „лакома“ (greedy, англ.). Тя е достатъчно проста, така че получените алгоритми работят много бързо. Разбира се, лакомата схема има и сериозен недостатък. Лакомите алгоритми използват само локални съображения, не отчитат всички подробности на задачата и в общия случай не намират оптимума. Но задачите, за които става дума са толкова сложни, че понякога алгоритъмът, построен

по лакомата схема е единствен начин да се намери някакво прилично (макар и не оптимално) решение за разумно време.

Същността на схемата, наречена лакома е следната: когато построяваме решение на оптимизационна задача (например в граф) естествено достигаме до момент, в който трябва да изберем пореден елемент на графа – връх, ребро или някоя по-сложна конфигурация. Тогава избираме този, който от гледна точка на ситуацията, в която се намираме, изглежда най-подходящ, най-привлекателен. От тук е ясно, че в множество ситуации лакомите алгоритми прибягват към прибързан избор, който ги лишава от възможността да намерят оптимално решение. Нека разгледаме следната оптимизационна задача: Даден е свързан граф. Да се намери най-ниското му покриващо дърво. На Фиг. 3.15.а е даден граф, за който искаме да построим такова оптимално по височина дърво. Един „лаком" алгоритъм би избрал да започне с върха, имащ най-много съседи, с надеждата, че ако постави повече върхове на първо ниво, шансовете да се намали височината се увеличават. На Фиг. 3.15.б е показано дървото, получено с такава лакома стратегия. На Фиг. 3.15.в е показано дърво, което има по-малка височина и демонстрира дефектите на лакомите стратегии.




Фигура 3.15: Неоптимално и оптимално по височина ПД на граф.

Интересно е да се знае кога схемата, наречена „лакома" построява алгоритми, водещи до намиране на оптимални решения. За да дадем отговор на този въпрос ще е необходимо да разгледаме достатъчно обща формализация на тази алгоритмична схема.

Нека $(E, \mathcal{I})$ е независима фамилия подмножества на $E$, а $c : E \to R^+$ положителна теглова функция, дефинирана върху елементите на $E$. Тегло на $S \subseteq E$ наричаме $c(S) = \sum_{e \in S} c(e)$. Да разгледаме задачата: Да се намери независимо подмножество $S_0 \in \mathcal{I}$ с максимално тегло.

***Дефиниция.*** *Лаком алгоритъм* наричаме следната процедура за



намиране на независимо подмножество с максимално тегло:

1. Нека $S = \emptyset$.
2. Образуваме $T = \{e | e \in E \setminus S, S \cup \{e\} \in \mathcal{I}\}$.
3. Ако $T = \emptyset$, край. В противен случай избираме $x \in T, c(x) = \max\{c(y) | y \in T\}$. Нека $S = S \cup \{x\}$ и преминаваме към 2.

Не е трудно да се забележи, че този алгоритъм е обобщение на алгоритъма на Крускал за намиране на максимално покриващо дърво на свързан граф. Действително, нека $E$ е множеството от ребра на краен ориентиран граф $G(V, E)$, с теглова функция $c : E \to R$. Образуваме независима фамилия множества от ребра на графа $(E, \mathcal{I})$, като наречем едно множество от ребра $D$ на $G$ независимо, ако не съдържа цикъл. Тогава максималните независими множества са точно множествата от ребра, образуващи покриващи дървета. Както доказахме по-горе, лакомият алгоритъм решава задачата за построяване на максимално (минимално) покриващо дърво. За съжаление не за всяка независима фамилия подмножества това е така.

Да разгледаме независимата фамилия подмножества $(\{1,2,3\}, \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}\})$ и нека $c$ е ценова функция със стойности $c(1) = 2$, $c(2) = 2$, $c(3) = 3$. На първата стъпка лакомият алгоритъм ще избере елемента 3, защото $\emptyset \cup \{3\}$ е независимо, $c(3) > c(1)$ и $c(3) > c(2)$. Но множеството $\{3\}$ е максимално независимо и затова алгоритъмът завършва с намерена максимална стойност 3. Всъщност независимото множество с най-висока цена 4 е $\{1,2\}$.

Разликата между двете задачи, от която се предопределят различните възможности за успех на лакомия алгоритъм се дава от следващите две теореми.

**Теорема 3.4.4** *Нека $(E, \mathcal{M})$ е матроид с ценова функция $c : E \to R^+$. Тогава лакомият алгоритъм намира независимо множество $S \in \mathcal{M}$ с максимална цена.*

**Доказателство** Нека лакомият алгоритъм намира като решение независимото множество $S \in \mathcal{M}$. Заради положителните стойности на ценовата функция е невъзможно независимото множество $S$, намерено от алгоритъма, да бъде разширено до независимо множество с по-голяма цена. Това означава, че $S$ е база на матроида. Да допуснем, че $S$ не е с максимална цена. Тогава съществува база $S'$ с максимална цена, т.е. $c(S') > c(S)$. Нека сме избрали $S'$ така, че $|S' \cap S|$ е с максимален брой елементи. Нека $x \in S \setminus S'$ е първият елемент на $S$, избран от алгоритъма, който не е в $S'$. Да разширим $(S \cap S') \cup \{x\}$ до базата

$(S' \setminus \{y\}) \cup \{x\}, y \in S', y \notin S$. Тъй като $S'$ е оптимално $c(y) \geq c(x)$. Не е възможно $c(y) > c(x)$, защото лакомият алгоритъм би избрал първо $y$ вместо $x$. Следователно $c(y) = c(x)$ и $S' \setminus \{y\} \cup \{x\}$ е оптимална база, с повече общи елементи с $S$, което е противоречие с избора на $S'$. □

В сила е и обратната

**Теорема 3.4.5** *Независимата фамилия от множества $(E, \mathcal{M})$ е матроид, ако при произволна ценова функция $c : E \to R^+$ лакомият алгоритъм намира независимо множество с максимална цена.*

***Доказателство.*** Тъй като $(E, \mathcal{M})$ е независима фамилия от множества, остава да покажем, че е в сила нарастващото свойство (в) от Дефиницията на матроид. Нека $X, Y \in \mathcal{M}$, $|X| = k, |Y| = k+1$. Дефинираме ценова функция $c : E \to R^+$ по следния начин:

$$c(e) = \begin{cases} (k+1)/(k+2) & \text{за } e \in X \\ k/(k+1) & \text{за } e \in Y \setminus X \\ \epsilon & \text{в противен случай,} \end{cases}$$

където $0 < \epsilon < k/(k+1)$. Лакомият алгоритъм, очевидно, ще избере първо всички елементи на $X$, заради високото им тегло. Но $c(Y) \geq k$, а $c(X) = k(k+1)/(k+2) < k$. Следователно $X$ не е с максимална цена и лакомият алгоритъм трябва да го разшири до независимо множество с по-висока цена. Тъй като $c(y) > c(z), \forall y \in Y \setminus X$, $\forall z \in E \setminus (X \cup Y)$, то трябва да има $y \in Y \setminus X$ такъв, че $X \cup \{y\} \in \mathcal{M}$. □

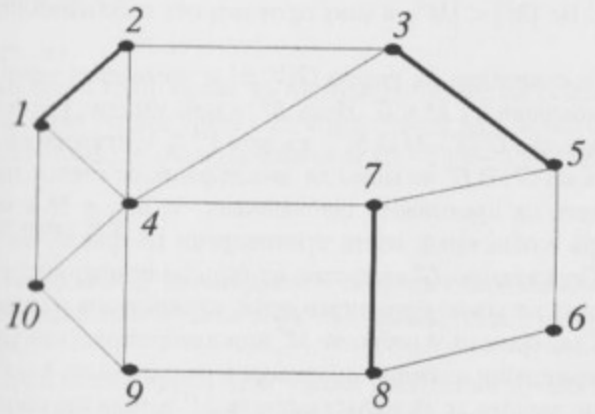
Лесно се вижда, че независимата фамилия $(E, \mathcal{I})$ от множества ребра на краен свързан граф, дефинирана по-горе, е матроид. Достатъчно е да забележим, че всички бази на тази фамилия, т.е. покриващите дървета на $G$, имат един и същ брой елементи $|E| - 1$. Така горните две теореми по нов начин доказват коректността на алгоритъма на Крускал. Обосновката в случая на минимално покриващо дърво може да бъде направена с подходяща трансформация на ценовата функция, която ще оставим за упражнение на читателя. По-различна е формулировката на задачата за намиране на най-късите пътища от зададен връх до всички останали върхове на графа. Но и тук успехът на алгоритъма на Дейкстра (лаком по същността си) може да бъде предвиден, защото се търси екстремална, в определен смисъл, база на матроида – покриващото дърво на минималните пътища.

**Максимално съчетание**

***Дефиниция.*** Нека $G(V, E)$ е краен граф. Множеството $M \subseteq E$ наричаме *съчетание,* ако всеки $v \in V$ е край на не повече от едно ребро от $M$. Съчетанието $M$ е *максимално,* ако за всяко друго съчетание $M'$ е в сила $|M'| \leq |M|$. Върховете, които са краища на ребра от съчетанието наричаме *покрити,* а тези, които не са краища на ребра от съчетанието – *свободни.*



Множеството ребра {(1,2),(3,5),(7,8)} е съчетание за графа от Фиг. 3.16. Върховете 1, 2, 3, 5, 7 и 8 са покритите от това съчетание върхове. Очевидно това съчетание не е максимално. Максималното съчетание не може да има повече от $\lfloor |V|/2 \rfloor$ елемента, но тази оценка може да се окаже твърде неточна.




Фигура 3.16: Съчетание в граф.

***Дефиниция.*** Нека $M$ е съчетание на крайния граф $G(V, E)$. Един път в $G$ е *алтерниращ относно* $M$, ако от всеки две последователни ребра в него, точно едно принадлежи на $M$. Алтерниращият относно съчетанието $M$ път наричаме *нарастващ,* ако началният и крайният му връх са свободни относно съчетанието $M$.

Пътят $(4, 1, 2, 3, 5, 6)$ е алтерниращ за графа от Фиг. 3.16, по отношение на съчетанието $\{(1,2), (3,5), (7,8)\}$. При това той е нарастващ път по отношение на $M$, защото върховете 4 и 6 са свободни спрямо $M$. Забележете, че в нарастващия път първото и последното ребро не са от $M$, т.е. всеки нарастващ път има нечетен брой ребра, като ребрата от $M$ са с 1 по-малко от ребрата от $E \setminus M$. Всички върхове на нарастващия път, без началния и крайния, са покрити от съчетанието.

В сила е следната

**Теорема 3.4.6** *Съчетанието $M$ е максимално в крайния граф $G$ тога-*

*ва и само тогава, когато не съществува нарастващ алтерниращ път относно $M$ в $G$.*

***Доказателство.*** 1) Нека M е максимално. Да допуснем, че съществува нарастващ алтерниращ път и нека $P \subseteq E$ са ребрата на този път, $P' = P \cap M$, а $P'' = P \cap (E \setminus M)$. Както отбелязахме по-горе $|P'| < |P''|$. Да образуваме множеството $M' = M \triangle P = (M \setminus P') \cup P''$. То е съчетание, защото смяната на ребрата от $P'$ с ребрата от $P''$ оставя вътрешните върхове на пътя покрити и покрива свободните до момента, начален и краен, върхове. Но $|M| < |M'|$, а това противоречи на оптималността на $M$.

2) Нека $M$ е съчетание на графа $G(V, E)$ и няма нито един нарастващ път по отношение на $M$ в $G$. Нека $M'$ е максимално съчетание. Да образуваме подграфа $G'(V', M \triangle M')$, където $V' \subseteq V$ естествено се определя от ребрата на $G'$. В $G'$ не може да има върхове от степен по-голяма от 2. Допускането на противното ще означава, че или в $M$ или в $M'$ е имало две ребра с общ край, което противоречи на факта, че $M$ и $M'$ са съчетания. Сега графът $G'$ се състои от (една или няколко) свързани компоненти, всяка от които е изолиран връх, алтерниращ път, започващ с ребро от $M$ и завършващ с ребро от $M'$ или алтерниращ цикъл с четна дължина. Алтерниращи пътища започващи и завършващи с ребро от $M$ не са възможни, защото те са нарастващи за $M'$ и това ще противоречи на максималността на $M'$. Алтерниращи пътища започващи и завършващи с ребро от $M'$ не са възможни, защото те са нарастващи за $M$ и това ще противоречи на условието на Теоремата. Циклите с нечетна дължина са невъзможни, защото тогава ще се окаже, че или в $M$ или в $M'$ е имало две ребра с общ край, което е недопустимо, защото $M$ и $M'$ са съчетания. Така във всяка от допустимите компоненти на $G'$ има по равен брой ребра от $M$ и $M'$. Но ребрата участващи в $G'$ са от $M \triangle M'$, следователно $|M| = |M'|$ и $M$ също е максимално. □

В примера разгледан по-горе нарастващият път $P_1 = \{(4,1), (1,2), (2,3), (3,5), (5,6)\}$ ни позволява да построим съчетанието $M_1 = M \triangle P_1 = \{(4,1), (2,3), (5,6), (7,8)\}$. Лесно се вижда, че пътят $P_2 = \{(9,4), (4,1), (1,10)\}$ е нарастващ по отношение на $M_1$. Това ни дава ново съчетание $\{(9,4), (1,10), (2,3), (5,6), (7,8)\}$, което е очевидно максимално, защото броят на участващите ребра е половината от броя на върховете.

Горната теорема е основа за построяване на алгоритми за намиране на максимално съчетание, повече или по-малко ефективни, в зависимост от възможностите за бързо построяване на нарастващи пътища. Доста добър резултат теоремата дава в двуделни графи, затова препоръчваме



на читателя сам да разработи алгоритъм за построяване на нарастващ път по отношение на дадено съчетание $M$ в двуделен граф.

---

## Упражнения

### Упражнение 3.1

Докажете, че:

а) във всеки граф броят на върховете с нечетна степен е четен;

б) всеки регулярен граф с нечетна степен има четен брой върхове;

в) всеки граф с $n$ върха и повече от $(n-1)(n-2)/2$ ребра е свързан.

### Упражнение 3.2

Познанството между двама души е симетрична релация. Докажете, че:

а) в произволна група от хора има двама души с равен брой познати в групата;

б) в произволна група от 6 човека има трима души, всеки от които познава останалите двама или трима души, които не се познават взаимно.

### Упражнение 3.3

Нека $G(V, E, f)$ е краен ориентиран мултиграф без примки. Матрица на инцидентност на $G$ наричаме матрицата $B_{|V| \times |E|} = ||b_{ij}||$, в която

$$b_{ij} = \begin{cases} 1 & \text{ако } e_j \text{ влиза във } v_i \\ -1 & \text{ако } e_j \text{ излиза от } v_i \\ 0 & \text{ако } e_j \text{ не е инцидентно на } v_i \end{cases}.$$

Какво можете да кажете за матрицата $B.B^T$?

### Упражнение 3.4

Докажете, че $\forall n > 0$, $n$-мерният двоичен куб е:

а) свързан граф;

б) двуделен граф.

**Упражнение 3.5**

*Пермутационен граф* $G_{\mathcal{P}}(n)$ наричаме графът с върхове пермутациите на $I_n$, като две пермутации са свързани с ребро, ако се получават една от друга с транспозиция (разместване) на два елемента на $I_n$.

а) Начертайте диаграмата на $G_{\mathcal{P}}(3)$;

б) Докажете, че $G_{\mathcal{P}}(n)$ е регулярен и намерете степента му;

в) Намерете броя на ребрата на $G_{\mathcal{P}}(n)$;

г) Докажете, че $G_{\mathcal{P}}(n)$ е свързан, двуделен, а при $n > 2$ – не планарен.

**Упражнение 3.6**

Намерете хомоморфизъм на $n$-мерния двоичен куб в $k$-мерния двоичен куб при $n > k > 0$.

**Упражнение 3.7**

Намерете групата от автоморфизми на графа $G(\{1,2,3,4,5\}, \{(1,3), (1,5), (2,3), (2,5), (4,3), (4,5), (1,2), (2,4), (3,5)\})$.

**Упражнение 3.8**

Постройте граф с 6 върха и единствен автоморфизъм – идентитетът.

**Упражнение 3.9**

Постройте всички неизоморфни свързани графи с 4 върха.

**Упражнение 3.10**

Оцветете с минимален брой цветове:

а) върховете;

б) ребрата;

на графа $G(I_{12}, \{(1,2), (2,8), (2,3), (2,6), (2,9), (3,4), (3,10), (4,5), (4,7), (5,12), (6,7), (7,11), (8,9), (9,10), (10,11), (11,12)\}$.

**Упражнение 3.11**

Докажете, че всеки планарен граф може да бъде оцветен в 5 цвята.



**Упражнение 3.12**

*Ребрен граф* на $G(V,E)$ наричаме графа $\tilde{G}(E,E')$ с върхове ребрата на $G$, като два върха на $\tilde{G}$ са свързани с ребро, ако имат общ край в $G$. Докажете, че ребреният граф на свързан граф е също свързан.

**Упражнение 3.13**

Пълен подграф с три върха на графа $G(V,E)$ наричаме *триъгълник* на $G$. Колко най-много ребра може да има граф с $n$ върха, в който няма триъгълници.

**Упражнение 3.14**

Намерете броя на ребрата в граф без цикли, който има $n$ върха и $k$ свързани компоненти.

**Упражнение 3.15**

Докажете, че в мултиграф с $2k > 0$ върха с нечетна степен множеството на ребрата се разбива на $k$ пътя, всеки от които започва и завършва във връх с нечетна степен.

**Упражнение 3.16**

Редицата от 0 и 1 $a_0, a_1, \ldots, a_{2^n-1}$ се нарича *редица на Де Брюйн*, ако $(a_i, a_{i+1 (\mathrm{mod}\, 2^n)}, \ldots, a_{i+n-1 (\mathrm{mod}\, 2^n)})$ за $i = 0, 1, \ldots, 2^n - 1$ са всички $n$-мерни двоични вектори. Постройте редица на Де Брюйн за $n = 3$.

**Упражнение 3.17**

Постройте алгоритъм за намиране на всички цикли на свързан граф.

**Упражнение 3.18**

Постройте алгоритъм по схемата „обхождане в дълбочина“ за търсене на хамилтонов път в граф.

**Упражнение 3.19**

Постройте лаком алгоритъм за минимално оцветяване на граф.

**Упражнение 3.20**

Постройте минимално покриващо дърво на графа $G$ с върхове $\{A, B, C, D, E, F, G, H, I, J, K\}$ и ценова функция $c$ на ребрата със значения $c(A,B) = 3$, $c(A,C) = 1$, $c(A,D) = 5$, $c(B,C) = 6$, $c(B,E) = 4$, $c(B,F) = 2$, $c(C,D) = 1$, $c(C,F) = 4$, $c(C,G) = 5$, $c(D,G) = 1$, $c(E,F) = 6$, $c(E,H) = 3$, $c(F,G) = 1$, $c(F,H) = 2$, $c(F,I) = 6$, $c(F,J) = 4$, $c(H,I) = 1$, $c(H,K) = 3$, $c(I,J) = 5$, $c(I,K) = 4$, $c(J,K) = 1$.

**Упражнение 3.21**

Постройте дърво на минималните пътища от върха $A$ до всички останали върхове в графа от предната задача.

**Упражнение 3.22**

Постройте алгоритъм за намиране на минималните пътища между всеки два върха на граф със зададена ценова функция на ребрата.

**Упражнение 3.23**

Посочете максимално съчетание в $n$-мерния двоичен куб.

# Глава 4

# Формални езици и абстрактни машини

Един от най-важните проблеми, свързани с използването на изчислителни машини е проблемът за езика, на който общуват потребител и компютър. За съжаление, компютрите все още не са в състояние да разбират достатъчно добре естествените човешки езици. Те се управляват от сложни последователности от команди, със строго фиксиран начин на построяване и строго фиксиран смисъл на всяка правилно построена управляваща последователност. Материалът в тази глава дава най-обща представа за начините, по които могат да се задават такива формализирани системи, както и за разпознаването правилността на една формална управляваща последователност.

Използваната в този дял на дискретната математика терминология е заимствана от теорията на естествените езици, но за правилното разбиране на същността ѝ е добре да не се търси директната аналогия, а да се асоциират понятията от Теорията на формалните езици със съответните понятия от Теорията на естествените езици в съответствие с Таблица 4.1.

| ФОРМАЛЕН ЕЗИК | ЕСТЕСТВЕН ЕЗИК |
|---|---|
| буква | дума |
| азбука | лексика, речник |
| дума | фраза, последователност от думи |
| език | правилно построени фрази, изречения |
| нетерминал | граматическа категория |
| терминал | обикновена дума |
| аксиома | граматическата категория "изречение" |

Таблица 4.1: Формални и естествени езици.

## 4.1 Формални езици и граматики

Да припомним някои понятия и означения, въведени в раздел 1.4. Нека $X = \{x_1, x_2, \ldots, x_n\}$ е крайна азбука и с $X^*$ да означим множеството от всички думи над тази азбука. С $d(\alpha)$ означаваме дължината на думата $\alpha \in X^*$, с $X^k$ – множеството от думите с дължина $k$, $X^1 = X$, а $X^0$ се състои само от празната дума $\varepsilon$.

Всяко от множествата $X^k, k = 0, 1, 2, \ldots$ е крайно, $|X^k| = |X|^k$ и тъй като $X^* = X^0 \cup X^1 \cup X^2 \cup \cdots$ е обединение на изброима фамилия изброими множества, можем да приложим Теорема 1.3.1 и да получим следното

**Следствие 4.1.1** *Множеството $X^*$ от думите над крайна азбука $X$ е изброимо.*

Не е трудно да се посочи и биекция $\phi : X^* \to N$. Оставяме на читателя да провери, че функцията

$$\phi(\alpha) = \begin{cases} 0 & \alpha = \varepsilon \\ i & \alpha = a_i, 1 \leq i \leq n \\ |X|.\phi(\alpha') + i & \alpha = \alpha' a_i \end{cases}$$

е такава биекция. Следната процедура намира $\forall i \in N$ думата $\phi^{-1}(i)(|X| = n)$:

**Дадено:** $i \in N$.
**Резултат:** $\alpha \in X^*, \phi(\alpha) = i$.
**Процедура:** (буквите на $\alpha$ се получават отзад напред):
1. Ако $i = 0$, думата е $\varepsilon$. Край.
2. Ако $i \leq n$, думата е $a_i$. Край.
3. Докато $i > n$
3.а. Намираме $j$ такова, че $i = kn + j, 1 \leq j \leq n$;
3.б. Поставяме $a_j$ в думата;
3.в. $i = k$. □

Забележете минималната разлика между горния алгоритъм и алгоритъма за превръщане на естествени числа от една позиционна бройна система с положителна цяла основа, в друга такава позиционна бройна система.

***Дефиниция.*** Нека $X$ е крайна азбука. Тогава всяко $L \subseteq X^*$ наричаме *език* над $X$.

Празното множество е подмножество на $X^*$ и следователно е език – *празният език*. Множеството $\{\varepsilon\} = X^0$ също е език. Забележете, че $\emptyset \neq \{\varepsilon\}$.



От Теорема 1.3.2 (за мощността на булеана на изброимо множество) получаваме следното

**Следствие 4.1.2** *Множеството от езиците над крайна азбука не е изброимо.*

Този факт поставя следния проблем: Как да задаваме езиците над крайна азбука $X$? За едно крайно подмножество на $X^*$ бихме могли да изредим всички елементи, но за безкрайните езици трябва да имаме някакво разумно крайно описание. Това описание, обаче, ще е дума над някаква (в общия случай различна от $X$) азбука. Тъй като тези думи са изброимо много, те не са достатъчни за описанието на всички езици. Затова на практика, каквато и крайна форма за описание на езиците да изберем, ще се наложи да се откажем от разглеждането на всички езици и да работим само с едно изброимо подмножество – ще ги наречем *формални езици*. За практическите нужди на теоретичната информатика това е напълно достатъчно. Интересен е въпросът дали когато сменяме формата на описание няма да се получи друго подмножество от формални езици. В изложението ще дадем достатъчно сериозни основания за това, че множеството от езици, които можем да представим с крайни описания, не зависи от формата на описване.

***Дефиниция.*** Елементите на релацията $R_{\to} \subseteq X^* \times X^*$, където $X$ е крайна азбука, наричаме *правила за извод* или просто *правила*. Означаваме ги с $\alpha \to \beta$. Думата $\alpha$ наричаме *лява част* , а $\beta$ – *дясна част* на правилото $\alpha \to \beta$ (четем "$\alpha$ отива в $\beta$").

***Дефиниция.*** С всяко множество от правила $R_{\to}$ свързваме релацията $R_{\vdash} \subseteq X^* \times X^*$, елементите на която се дефинират посредством $R_{\to}$ по следния начин: $\gamma \vdash \delta \in R_{\vdash}$, ако $\gamma = \gamma' \alpha \gamma''$, $\delta = \gamma' \beta \gamma''$ и $\alpha \to \beta \in R_{\to}$ (четем "от $\gamma$ непосредствено се извежда $\delta$"). Транзитивното и рефлексивно затваряне на $R_{\vdash}$ бележим с $R_{\vdash\!\!=}$. Елементите на $R_{\vdash\!\!=}$ означаваме с $\gamma \vdash\!\!= \delta$ и четем "от $\gamma$ се извежда $\delta$" (или "$\gamma$ извежда $\delta$").

***Дефиниция.*** *Формална граматика* наричаме четворката

$$\Gamma = < \mathcal{N}, \mathcal{T}, S, \mathcal{P} >,$$

в която $\mathcal{N}$ е крайна азбука, елементите на която наричаме *нетерминали*; $\mathcal{T}$ е крайна азбука, елементите на която наричаме *терминали*, $\mathcal{N} \cap \mathcal{T} = \emptyset$; $S \in \mathcal{N}$ – *аксиома (начален нетерминал)*; $\mathcal{P} \subseteq (\mathcal{N} \cup \mathcal{T})^* \times (\mathcal{N} \cup \mathcal{T})^*$ – множество от правила $\alpha \to \beta$, за които е изпълнено:
а) $\alpha = \alpha' A \alpha'', A \in \mathcal{N}, \alpha', \alpha'' \in (\mathcal{N} \cup \mathcal{T})^*$;
б) $\beta \in (\mathcal{N} \cup \mathcal{T})^*$;

в) $d(\alpha) \leq d(\beta)$ (*несъкращаващи* правила).

С помощта на правилата $\mathcal{P}$ на формалната граматика $\Gamma$ дефинираме релациите на непосредствен извод $\mathcal{P}_{\vdash}$ и на извод $\mathcal{P}_{\models}$, по начина посочен по-горе. Ще ги означаваме с $\overset{\Gamma}{\vdash}$ и $\overset{\Gamma}{\models}$, за да подчертаем, че съответните изводи са в граматиката $\Gamma$.

***Дефиниция.*** Множеството $L_\Gamma = \{\omega | S \overset{\Gamma}{\models} \omega, \omega \in \mathcal{T}^*\}$ наричаме *език породен от граматиката* $\Gamma$ или просто *език* над $\Gamma$. Последователността

$$S \overset{\Gamma}{\vdash} \alpha_1 \overset{\Gamma}{\vdash} \alpha_2 \overset{\Gamma}{\vdash} \cdots \overset{\Gamma}{\vdash} \alpha_m \overset{\Gamma}{\vdash} \omega$$

от която сме установили, че $S \overset{\Gamma}{\models} \omega$ наричаме *извод* на $\omega$ в граматиката $\Gamma$. В случаите, когато граматиката се подразбира, може и да не пишем името ѝ в означенията на релациите за непосредствен извод и извод.

За пример да разгледаме граматиката

$$\begin{aligned}\Gamma_1 \quad = \quad & < \{S, A, B\}, \{a, b\}, S, \\ & \{S \to aA, A \to aA, A \to b, S \to bB, B \to bB, B \to a\} > .\end{aligned}$$

Две са възможностите за започване на извод в тази граматика. Ако започнем с $S \to aA$, тогава в общия случай ще трябва да приложим $0, 1, \ldots$ пъти правилото $A \to aA$ и да завършим с $A \to b$. Получаваме извод от следния вид

$$S \vdash aA \vdash aaA \vdash \cdots \vdash a^{i+1}A \vdash a^{i+1}b, i = 0, 1, 2, \ldots .$$

Аналогично, ако започнем с $S \to bB$, ще получим извод от вида

$$S \vdash bB \vdash bbB \vdash \cdots \vdash b^{i+1}B \vdash b^{i+1}a, i = 0, 1, 2, \ldots .$$

Следователно $S \models a^{i+1}b, S \models b^{i+1}a, i = 0, 1, 2, \ldots$ или $L_{\Gamma_1} = \{a^{i+1}b, b^{i+1}a | i = 0, 1, 2, \ldots\}$.

Като втори пример да разгледаме граматиката

$$\Gamma_2 =< \{S\}, \{a, b\}, S, \{S \to ab, S \to aSb\} > .$$

И тук има две възможности за начало на извода. При първата той завършва след една стъпка: $S \vdash ab$, следователно $S \models ab$. При втората възможност, в общия случай, прилагаме $i - 1$ пъти второто правило $(i = 2, 3, 4, \ldots)$ и завършваме извода с прилагане на първото правило. Получаваме извод от вида

$$S \vdash aSb \vdash a^2Sb^2 \vdash \cdots \vdash a^{i-1}Sb^{i-1} \vdash a^ib^i, i = 1, 2, 3, \ldots .$$



т.е. $S \models a^ib^i, i = 1, 2, 3, \ldots$. Следователно $L_{\Gamma_2} = \{a^ib^i | i = 1, 2, 3, \ldots\}$.

Като последен пример да разгледаме граматиката

$$\begin{aligned}\Gamma_3 \quad = \quad & < \{S, A, B, C\}, \{a, b, c\}, S, \{S \to aSBC, S \to abC, \\ & CB \to AB, AB \to AC, AC \to BC, bB \to bb, C \to c\} > .\end{aligned}$$

В този случай не е толкова просто да се определят думите, извеждани от аксиомата на граматиката. Ще посочим единствената възможност за извеждане на думи, съставени само от терминали: Прилагаме $i - 1$ пъти първото правило, $i = 1, 2, 3, \ldots$. След това прилагаме един път второто и получаваме

$$S \vdash aSBC \vdash a^2S(BC)^2 \vdash \cdots \vdash a_{i-1}S(BC)^{i-1} \vdash a^ibC(BC)^{i-1}.$$

С последователното прилагане на трето, четвърто и пето правило се разменят местата на нетерминалите $C$ и $B$, когато са един до друг. Това позволява да извадим всички букви $B$ преди всички $C$ в поддумата $C(BC)^{i-1}$ и да получим $a^ibB^{i-1}C^i$. Сега $i - 1$ пъти прилагаме шестото правило, а $i$ пъти седмото и получаваме

$$a^ibB^{i-1}C^i \vdash a^ib^2B^{i-2}C^i \vdash \cdots \vdash a^ib^iC^i \vdash a^ib^icC^{i-1} \vdash \cdots \vdash a^ib^ic^i.$$

Или $S \models a^ib^ic^i, i = 1, 2, 3, \ldots$.

Оставяме на читателя да провери, че всяко отклонение от така описаната схема довежда до думи, съдържащи нетерминали, за които изводът не може да продължи и затова никои други думи, съставени само от терминали не могат да бъдат извеждани от $S$. Получаваме $L_{\Gamma_3} = \{a^ib^ic^i | i = 1, 2, \ldots\}$.

Забележете две особености на изводите в разгледаните граматики. Всяко правило има поне един нетерминал в лявата си част. Затова, след получаване на дума, съставена само от терминали, изводът не може да продължи и не може да се изведе и друга дума, съставена само от терминали. Втората особеност е резултат от това, че правилата са несъкращаващи. По време на извода дължините на извежданите думи могат само да нарастват или да остават същите. Тези две особености до голяма степен улесняват анализа на езиците, породени от разглежданите граматики.

От несъкращаемостта на извежданата дума следва, че не е възможно да бъде изведена празната дума, тъй като $d(S) = 1$, а $d(\varepsilon) = 0$. Ще разрешим този проблем, като първо покажем, че граматиките могат да бъдат освободени от аксиомата в десните части на правилата.

***Дефиниция.*** Граматиките $\Gamma_1$ и $\Gamma_2$ се наричат *еквивалентни* (отбелязваме го с $\Gamma_1 \cong \Gamma_2$), ако $L_{\Gamma_1} = L_{\Gamma_2}$.

**Теорема 4.1.1** *За всяка граматика* $\Gamma$ *съществува* $\Gamma' \cong \Gamma$, *в която аксиомата не се среща в дясна част на правило.*

***Доказателство.*** Нека $\Gamma =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} >$. Построяваме $\Gamma' =< \mathcal{N} \cup \{S'\}, \mathcal{T}, S', \mathcal{P}' >$, в която $S' \notin \mathcal{N} \cup \mathcal{T}$, а $\mathcal{P}' = \mathcal{P} \cup \{S' \to S\}$. Ще покажем, че $L_\Gamma = L_{\Gamma'}$.

а) Нека $\omega \in L_\Gamma$, т.е. $S \overset{\Gamma}{\models} \omega$. Но всяко правило на $\Gamma$ е в $\Gamma'$, следователно можем да напишем $S \overset{\Gamma'}{\models} \omega$. Добавяме към този извод първа стъпка от правилото $S' \to S$ и получаваме $S' \overset{\Gamma'}{\vdash} S \overset{\Gamma'}{\models} \omega$, следователно $S' \overset{\Gamma'}{\models} \omega$ и $\omega \in L_{\Gamma'}$.

б) Нека $\omega \in L_{\Gamma'}$. Следователно $S' \overset{\Gamma'}{\models} \omega$ или $S' \overset{\Gamma'}{\vdash} S \overset{\Gamma'}{\models} \omega$. Но $S'$ се среща само в правилото $S' \to S$ и не може да участва в извода $S \overset{\Gamma'}{\models} \omega$, т.е. този извод е направен само с правила на старата граматика и $S \overset{\Gamma}{\models} \omega$, т.е. $\omega \in L_\Gamma$.

Така $L_\Gamma = L_{\Gamma'}$ и тъй като $S' \to S$ е единственото правило на $\Gamma'$ съдържащо аксиомата $S'$, тя е граматика без аксиома в дясна част на правило. □

За граматиката $\Gamma_2$, използвайки конструкцията на Теорема 4.1.1 получаваме еквивалентната ѝ $\Gamma_2' =< \{S', S\}, \{a, b\}, S', \{S' \to S, S \to ab, S \to aSb\} >$.

***Дефиниция.*** Правила от вида $A \to B, A, B \in \mathcal{N}$, наричаме *преименуващи.*

В естествените езици такова правило съответства на даване на две различни имена на една и съща граматическа категория. Естествено е да очакваме, че преименуващите правила не могат да допринесат нищо за пораждания език. Ще покажем как можем да се освободим от тях.

***Дефиниция.*** Последователност от правила от вида

$$A_{i_1} \to A_{i_2}, A_{i_2} \to A_{i_3}, \ldots, A_{i_{k-1}} \to A_{i_k}$$

наричаме *верига от преименуващи правила* и за по-кратко я записваме

$$A_{i_1} \to A_{i_2} \to A_{i_3} \to \ldots \to A_{i_k}.$$

**Теорема 4.1.2** *За всяка формална граматика* $\Gamma$ *съществува* $\Gamma' \cong \Gamma$, *която няма преименуващи правила.*



***Доказателство.*** Нека $\Gamma =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} >$. Построяваме граматиката $\Gamma' =< \mathcal{N}, \mathcal{T}, S, \mathcal{P}' >$ с множество правила $\mathcal{P}' = (\mathcal{P} \setminus \{A_{j_1} \to A_{j_2}\}) \cup Q$, където

$$Q = \{A \to \alpha | \forall A \to A_{i_1} \to \ldots \to A_{i_k}, A_{i_k} \to \alpha, \alpha \notin \mathcal{N}, 1 \leq k < |\mathcal{N}|\},$$

в които няма преименуващи правила. Ще покажем, че $L_\Gamma = L_{\Gamma'}$.

А) Нека $\delta \overset{\Gamma}{\models} \omega$. С индукция по броя на веригите от преименуващи правила в този извод ще докажем, че $\delta \overset{\Gamma'}{\models} \omega$.

а) Нека в $\delta \overset{\Gamma}{\models} \omega$ няма вериги от преименуващи правила, т.е. използвани са само непреименуващи правила. Но всички такива правила са в $\mathcal{P}'$ и следователно $\delta \overset{\Gamma'}{\models} \omega$.

б) Допускаме, че ако $\delta' \overset{\Gamma}{\models} \omega'$ с не повече от $n$ вериги от преименуващи правила, то $\delta' \overset{\Gamma'}{\models} \omega'$.

в) Нека $\delta \overset{\Gamma}{\models} \omega$ е извод с $n+1$ вериги от преименуващи правила. Да отделим частта от извода до първата такава верига.

$$\delta \overset{\Gamma}{\models} \delta_1 A \delta_2 \overset{\Gamma}{\vdash} \delta_1 A_{i_1} \delta_2 \overset{\Gamma}{\vdash} \ldots \overset{\Gamma}{\vdash} \delta_1 A_{i_k} \delta_2 \overset{\Gamma}{\vdash} \delta_1 \alpha \delta_2 \overset{\Gamma}{\models} \omega.$$

В извода $\delta \overset{\Gamma}{\models} \delta_1 A \delta_2$ няма преименуващи правила и следователно $\delta \overset{\Gamma'}{\models} \delta_1 A \delta_2$. Ако $k < |\mathcal{N}|$, тогава в $\mathcal{P}'$ съществува $A \to \alpha$. В противен случай, от принципа на Дирихле следва, че в множеството $\{A, A_{i_1}, \ldots, A_{i_k}\}$ има двойка повтарящи се нетерминали $A_{i_j}, A_{i_{j+r}}, r \geq 1$ и можем да получим по-къса верига от преименуващи правила, премахвайки частта от $A_{i_{j+1}}$ до $A_{i_{j+r}}$. Ако получената верига е отново по-дълга от броя на нетерминалите, повтаряме горната стъпка дотогава, докато получим верига от преименуващи правила с дължина по-малка от $|\mathcal{N}|$. Така намираме съответно правило $A \to \alpha \in \mathcal{P}'$ и в този случай.

Останалата част на извода $\delta_1 \alpha \delta_2 \overset{\Gamma}{\models} \omega$ съдържа точно $n$ вериги от преименуващи правила и съгласно индукционното предположение $\delta_1 \alpha \delta_2 \overset{\Gamma'}{\models} \omega$. И така получихме

$$\delta \overset{\Gamma'}{\models} \delta_1 A \delta_2 \overset{\Gamma'}{\vdash} \delta_1 \alpha \delta_2 \overset{\Gamma'}{\models} \omega$$

и твърдението е доказано. В частния случай $\delta = S$ получаваме, че ако $S \overset{\Gamma}{\models} \omega$, то $S \overset{\Gamma'}{\models} \omega$, т.е. ако $\omega \in L_\Gamma$, то $\omega \in L_{\Gamma'}$.

Б) Нека $\delta \overset{\Gamma'}{\models} \omega$. С индукция по броя на новодобавените в $\Gamma'$ правила, които се срещат в този извод, ще докажем, че $\delta \overset{\Gamma}{\models} \omega$.

а) Нека в $\delta \overset{\Gamma'}{\models} \omega$ няма нито едно ново правило, т.е. всички използвани правила са от $\mathcal{P}$. Тогава $\delta \overset{\Gamma}{\models} \omega$.

б) Допускаме, че ако $\delta' \overset{\Gamma'}{\models} \omega'$ е извод с не повече от $n$ нови правила, то $\delta' \overset{\Gamma}{\models} \omega'$.

в) Нека $\delta \overset{\Gamma'}{\models} \omega$ е извод с $n+1$ нови правила. Отделяме частта до първото такова правило

$$\delta \overset{\Gamma'}{\models} \delta_1 A \delta_2 \overset{\Gamma'}{\vdash} \delta_1 \alpha \delta_2 \overset{\Gamma'}{\models} \omega.$$

В извода $\delta \overset{\Gamma'}{\models} \delta_1 A \delta_2$ не се срещат нови правила, следователно $\delta \overset{\Gamma}{\models} \delta_1 A \delta_2$. Съгласно конструкцията на $\mathcal{P}'$, правилото $A \to \alpha$ е включено в $\mathcal{P}'$, защото в $\Gamma$ съществува $A \to A_{i_1} \to A_{i_2} \to \ldots \to A_{i_k}$ и правило $A_{i_k} \to \alpha$. Следователно

$$\delta_1 A \delta_2 \overset{\Gamma}{\vdash} \delta_1 A_{i_1} \delta_2 \overset{\Gamma}{\vdash} \cdots \overset{\Gamma}{\vdash} \delta_1 A_{i_k} \delta_2 \overset{\Gamma}{\vdash} \delta_1 \alpha \delta_2$$

или $\delta_1 A \delta_2 \overset{\Gamma}{\models} \delta_1 \alpha \delta_2$. Частта от извода $\delta_1 \alpha \delta_2 \overset{\Gamma'}{\models} \omega$ съдържа точно $n$ нови правила и съгласно индукционното предположение $\delta_1 \alpha \delta_2 \overset{\Gamma}{\models} \omega$. Следователно

$$\delta \overset{\Gamma}{\models} \delta_1 A \delta_2 \overset{\Gamma}{\models} \delta_1 \alpha \delta_2 \overset{\Gamma}{\models} \omega$$

и твърдението е доказано.

В частния случай $\delta = S$ получаваме, че ако $S \overset{\Gamma'}{\models} \omega$, то $S \overset{\Gamma}{\models} \omega$, т.е. ако $\omega \in L_{\Gamma'}$, то $\omega \in L_\Gamma$.

От А) и Б) следва, че $L_\Gamma = L_{\Gamma'}$ и граматиката $\Gamma'$ е без преименуващи правила. □

Както видяхме, заради това, че правилата на разглежданите граматики са несъкращаващи, не можем в такива граматики да извеждаме празната дума. За да се освободим от това ограничение, въвеждаме следното ("съкращаващо") правило: $S \to \varepsilon$. В граматика $\Gamma$, съдържаща такова правило ще бъде възможен изводът $S \overset{\Gamma}{\models} \varepsilon$ и тъй като $\varepsilon \in \mathcal{T}^*$, то $\varepsilon \in L_\Gamma$.



**Теорема 4.1.3** *Нека $\Gamma =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} >$ е граматика без аксиома в дясна част на правило. Тогава, ако $\Gamma' =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} \cup \{S \to \varepsilon\} >$, то $L_{\Gamma'} = L_\Gamma \cup \{\varepsilon\}$.*

***Доказателство.*** След като $\Gamma$ е без аксиома в дясна част на правило, единственият извод в който правилото $S \to \varepsilon$ участва е $S \overset{\Gamma'}{\vdash} \varepsilon$, т.е. $S \overset{\Gamma'}{\models} \varepsilon$ и $\varepsilon \in L_{\Gamma'}$. В останалите си изводи $\Gamma'$ по нищо не се отличава от $\Gamma$ и това доказва твърдението. □

Забележете, че ако $S$ се среща в дясна част на правило, то $S$ ще участва във вътрешността на съществени изводи и включването на $S \to \varepsilon$, може да доведе до получаване в $L_{\Gamma'}$ на думи, които не са в $L_\Gamma$. За пример да разгледаме граматиката $\Gamma_4 =< \{S\}, \{a, b, x\}, S, \{S \to aSb, S \to x\}$, езикът на която е $L_{\Gamma_4} = \{a_i x b_i | i = 1, 2, \ldots\}$. Сега да добавим правилото $S \to \varepsilon$, без да премахваме $S$ от дясната част на правилата. Получаваме $\Gamma_4' =< \{S\}, \{a, b, x\}, S, \{S \to aSb, S \to x, S \to \varepsilon\} >$. Тогава $L_{\Gamma_4'} = \{\varepsilon\} \cup \{a_i x b_i | i = 1, 2, \ldots\} \cup \{a_i b_i | i = 1, 2, \ldots\}$.

Ще припомним, че за всеки два езика $L_1$ и $L_2$ над крайна азбука в раздел 1.4 дефинирахме сума $L_1 + L_2$ и произведение $L_1.L_2$, а за всеки език $L$ – итерация $L^*$.

Ще се спрем на една класификация на формалните езици, породени от формални граматики, в зависимост от вида на използваните правила.

**Дефиниция** (*Йерархия на Чомски*).

а) Формалната граматиката $\Gamma$ наричаме граматика от *общ тип* (от *тип 0*), ако всяко нейно правило е от вида

$$\gamma_1 A \gamma_2 \to \beta, \gamma_1, \gamma_2 \in (\mathcal{N} \cup \mathcal{T})^*, \beta \in (\mathcal{N} \cup \mathcal{T})^+, A \in \mathcal{N}.$$

Ако $\Gamma$ е без аксиома в дясна част на правило, допускаме в нея и правилото $S \to \varepsilon$. Езиците, породени от граматики от общ тип, наричаме *езици от общ тип* (от *тип 0*) и ги означаваме с $\mathcal{L}_0$.

б) Формалната граматиката $\Gamma$ наричаме *контекстна* граматика (*контекстно-зависима* или граматика от *тип 1*), ако всяко нейно правило е от вида

$$\gamma_1 A \gamma_2 \to \gamma_1 \beta \gamma_2, \gamma_1, \gamma_2 \in (\mathcal{N} \cup \mathcal{T})^*, \beta \in (\mathcal{N} \cup \mathcal{T})^+, A \in \mathcal{N}.$$

Думите $\gamma_1$ и $\gamma_2$ наричаме *ляв* и *десен контекст*. Ако $\Gamma$ е без аксиома в дясна част на правило, допускаме в нея и правилото $S \to \varepsilon$. Езиците, породени от контекстни граматики, наричаме *контекстни* езици (*контекстно-зависими* езици или езици от *тип 1*) и ги означаваме с $\mathcal{L}_1$.

в) Формалната граматиката Г наричаме *контекстно-свободна* граматика (*безконтекстна* граматика или граматика от *тип 2*), ако всяко нейно правило е от вида

$$A \to \alpha, \alpha \in (\mathcal{N} \cup \mathcal{T})^+, A \in \mathcal{N}.$$

Ако Г е без аксиома в дясна част на правило, допускаме в нея и правилото $S \to \varepsilon$. Езиците, породени от контекстно-свободни граматики, наричаме *контекстно-свободни* езици (*безконтекстни* езици или езици от *тип 2*) и ги означаваме с $\mathcal{L}_2$.

г) Формалната граматиката Г наричаме *автоматна* граматика (*ляволинейна* или от *тип 3*), ако правилата ѝ са от вида

$$A \to B, A \to aB, A \to a, a \in \mathcal{T}, A, B \in \mathcal{N}.$$

Ако Г е без аксиома в дясна част на правило, допускаме в нея и правилото $S \to \varepsilon$. Езиците, породени от автоматни граматики, наричаме *автоматни* езици (*ляво-линейни* езици или езици от *тип 3*) и ги означаваме с $\mathcal{L}_3$.

Граматиките $\Gamma_1, \Gamma_2, \Gamma_3$ и $\Gamma_4$, които разгледахме като примери по-горе са контекстно-зависими. Граматиките $\Gamma_1, \Gamma_2$ и $\Gamma_4$ са контекстно-свободни, а $\Gamma_3$ не е контекстно-свободна. Граматиката $\Gamma_1$ е автоматна, а граматиките $\Gamma_2$ и $\Gamma_4$ не са автоматни.

**Теорема 4.1.4** $\mathcal{L}_3 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_0$.

***Доказателство.*** Достатъчно е да се отбележи, че правилата на граматиките от тип $i, i = 3, 2, 1$ са частен случай на правилата на граматиките от тип $i - 1$. □

Тази теорема обяснява термина йерархия, с който наричаме класификацията на Чомски.

По-нататък ще докажем, че съществуват контекстно-свободни езици, които не са автоматни и контекстно-зависими езици, които не са контекстно-свободни. Забележете, че това не следва от факта, че има граматики, които са от тип $i$, но не са от тип $i+1$, защото не е изключено един и същ език да се поражда от граматика от тип $i$ и от граматика от тип $i + 1$.

От дефиницията на Йерархията на Чомски и Теорема 4.1.4 получаваме следното

**Следствие 4.1.3** *Нека $L$ е език от тип $i, i = 0, 1, 2, 3$. Тогава $L \cup \{\varepsilon\}$ и $L \setminus \{\varepsilon\}$ са също езици от тип $i$.*



## 4.2 Автоматни езици

Съгласно дефиницията на Чомски, автоматни са езиците, които се пораждат от граматики, в които правилата са от вида $A \to B, A \to aB$ или $A \to a$, където $A, B \in \mathcal{N}, a \in \mathcal{T}$. За преименуващите правила доказахме, че могат да бъдат премахнати от граматиката без да се измени породеният език. Затова ще разглеждаме автоматни граматики с преименуващи правила, само когато това е необходимо.

Правилата от вида $A \to a$ ще наричаме *къси*, а от вида $A \to aB$ – *дълги*. Лесно се забелязва, че всеки извод на дума $\neq \varepsilon$ в автоматна граматика има вида

$$S \vdash a_{i_1}A_{i_1} \vdash a_{i_1}a_{i_2}A_{i_2} \vdash \cdots \vdash a_{i_1}a_{i_2}\ldots a_{ik-1}A_{ik-1} \vdash a_{i_1}a_{i_2}\ldots a_{ik-1}a_{i_k},$$

т.е. последователно се прилагат дългите правила $S \to a_{i_1}A_{i_1}$, $A_{i_1} \to a_{i_2}A_{i_2}, \ldots, A_{i_{k-2}} \to a_{i_{k-1}}A_{i_{k-1}}$, при което броят на терминалите в извежданата дума се увеличава на всяка стъпка с 1 и винаги остава точно един нетерминал в края на думата. Когато за първи път се приложи късо правило, например $A_{i_{k-1}} \to a_{i_k}$, извежданата дума вече не съдържа нетерминали и изводът завършва.

Да представим всички изводи на автоматната граматика Г в дървовидна структура, в която върховете могат да бъдат безкрайно множество. Върховете на дървото са надписани с думите, извеждани от граматиката, като коренът е надписан с аксиомата. Два върха са съединени с ориентирано ребро ако думата, с която е надписан първият извежда непосредствено думата, с която е надписан вторият. Ако една и съща дума се получава с повече от един извод за нея се включва в безкрайното дърво по един връх за всеки извод, от който тя се получава.

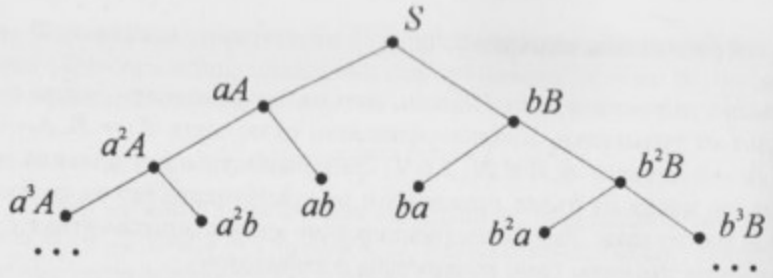
За граматиката $\Gamma_1$ от раздел 4.1 началото на безкрайното дърво е показано на Фиг. 4.1.

### 4.2.1 Свойства на автоматните езици

Ще покажем, че множеството на автоматните езици е затворено относно операциите събиране, умножение и итерация на езици.

**Теорема 4.2.1** *Ако $L'$ и $L''$ са автоматни езици, то $L' + L''$ е автоматен език.*

***Доказателство.*** Нека $L'$ се поражда от автоматната граматика $\Gamma' = < \mathcal{N}', \mathcal{T}', S', \mathcal{P}' >$, а езикът $L''$ от автоматната граматика $\Gamma'' =$




Фигура 4.1: Безкрайно дърво на думите на автоматен език.

$< \mathcal{N}'', \mathcal{T}'', S'', \mathcal{P}'' >$. Без ограничение на общността можем да поискаме $\mathcal{N}' \cap \mathcal{N}'' = \emptyset$, $\mathcal{N}' \cap \mathcal{T}'' = \emptyset$, $\mathcal{T}' \cap \mathcal{N}'' = \emptyset$ (ако това не е изпълнено, можем да преименуваме всеки нетерминал, повтарящ буква от друга азбука).

Построяваме граматиката $\Gamma =< \mathcal{N}' \cup \mathcal{N}'' \cup \{S\}, \mathcal{T}' \cup \mathcal{T}'', S, \mathcal{P} >$, където $S \notin \mathcal{N}' \cup \mathcal{N}'' \cup \mathcal{T}' \cup \mathcal{T}''$, с правила:

$$\begin{aligned} \mathcal{P} &= ((\mathcal{P}' \cup \mathcal{P}'') \setminus \{S' \to \varepsilon, S'' \to \varepsilon\}) \cup \{S \to S', S \to S''\} \cup \\ &\cup \{S \to \varepsilon | S' \to \varepsilon \text{ или } S'' \to \varepsilon\}. \end{aligned}$$

Очевидно получената граматика $\Gamma$ е автоматна, защото е образувана от правилата на автоматните граматики $\Gamma'$ и $\Gamma''$, правилата $S \to S', S \to S''$ (винаги можем да се освободим от тези преименуващи правила, ако е необходимо) и евентуално от правилото $S \to \varepsilon$ (аксиомата $S$ не се среща в дясна част и предварително сме се освободили от другите $\varepsilon$-правила).

Ще докажем, че $L_\Gamma = L_{\Gamma'} + L_{\Gamma''} = L' + L''$.

А) Нека $\omega \in L_{\Gamma'} + L_{\Gamma''}$. Без ограничение на общността можем да считаме, че $\omega \in L_{\Gamma'}$. Разсъжденията в случая, когато $\omega \in L_{\Gamma''}$ са аналогични.

а) ако $\omega = \varepsilon$, тогава $S' \to \varepsilon \in \mathcal{P}'$ и следователно $S \to \varepsilon \in \mathcal{P}$. Тогава $\omega \in L_\Gamma$.

б) Ако $\omega \neq \varepsilon$, тогава $S' \overset{\Gamma'}{\models} \omega$ (без използване на $S' \to \varepsilon$). Но всички останали правила на $\Gamma'$ са в $\Gamma$, следователно $S' \overset{\Gamma}{\models} \omega$. Но $S \to S' \in \mathcal{P}$ и следователно $S \overset{\Gamma}{\vdash} S' \overset{\Gamma}{\models} \omega$, т.е. $S \overset{\Gamma}{\models} \omega$ и $\omega \in L_\Gamma$.

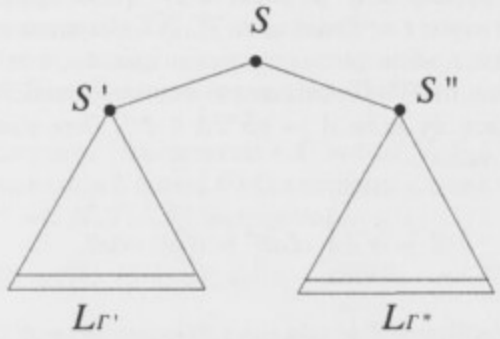
Б) Нека $\omega \in L_\Gamma$.

а) ако $\omega = \varepsilon$, тогава $S \to \varepsilon \in \mathcal{P}$ и следователно или $S' \to \varepsilon \in \mathcal{P}'$ или $S'' \to \varepsilon \in \mathcal{P}''$, т.е. $\omega \in L_{\Gamma'} + L_{\Gamma''}$.



б) ако $\omega \neq \varepsilon$, тогава $S \overset{\Gamma}{\models} \omega$ (без използване на $S \to \varepsilon$), т.е изводът започва или с $S \to S'$ или с $S \to S''$. Заради $N' \cap N'' = \emptyset$, ако изводът е $S \overset{\Gamma}{\vdash} S' \overset{\Gamma}{\models} \omega$ тогава $S' \overset{\Gamma'}{\models} \omega$, а ако $S \overset{\Gamma}{\vdash} S'' \overset{\Gamma}{\models} \omega$ тогава $S'' \overset{\Gamma''}{\models} \omega$. В първия случай $\omega \in L_{\Gamma'}$, а във втория $\omega \in L_{\Gamma''}$. Следователно $\omega \in L_{\Gamma'} + L_{\Gamma''}$ и в двата случая. □

На Фиг. 4.2 е изобразена конструкцията на безкрайното дърво на $\Gamma$ от безкрайните дървета на $\Gamma'$ и $\Gamma''$ (за случая $\varepsilon \notin L_{\Gamma'}$, $\varepsilon \notin L_{\Gamma''}$), която илюстрира същността на доказаната теорема.




Фигура 4.2: Сума на автоматни езици.

**Теорема 4.2.2** *Ако $L'$ и $L''$ са автоматни езици, то $L'.L''$ е автоматен език.*

***Доказателство.*** Нека $L'$ и $L''$ се пораждат от автоматните граматики $\Gamma' = < \mathcal{N}', \mathcal{T}', S', \mathcal{P}' >$ и $\Gamma'' =< \mathcal{N}'', \mathcal{T}'', S'', \mathcal{P}'' >$. Без ограничение на общността можем да поискаме $\mathcal{N}' \cap \mathcal{N}'' = \emptyset, \mathcal{N}' \cap \mathcal{T}'' = \emptyset, \mathcal{T}' \cap \mathcal{N}'' = \emptyset$ (в противен случай ще преименуваме всеки нетерминал, повтарящ буква от друга азбука).

А) Нека $\varepsilon \notin L_{\Gamma'}$ и $\varepsilon \notin L_{\Gamma''}$. Построяваме граматиката $\Gamma =< \mathcal{N}' \cup \mathcal{N}'', \mathcal{T}' \cup \mathcal{T}'', S', \mathcal{P} >$, където

$$\mathcal{P} = ((\mathcal{P}' \cup \mathcal{P}'') \setminus \{A \to a | A \to a \in \mathcal{P}'\}) \cup \{A \to aS'' | \forall A \to a \in \mathcal{P}'\}.$$

Очевидно $\Gamma$ е автоматна, защото е образувана от правилата на $\mathcal{P}'$, $\mathcal{P}''$ и новодобавените от вида $A \to aS''$, които също са автоматни. Ще покажем, че $L_\Gamma = L_{\Gamma'}.L_{\Gamma''} = L'.L''$.

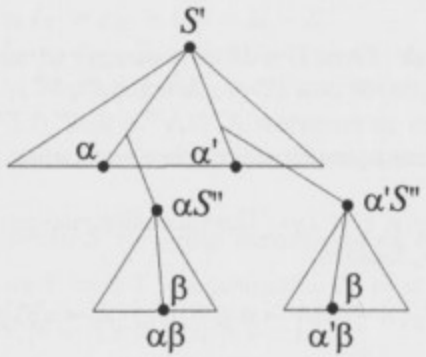
а) Нека $\omega \in L'.L''$. Тогава $\omega = \alpha\beta, \alpha \in L', \beta \in L''$, т.е. $S' \overset{\Gamma'}{\models} \alpha$ и $S'' \overset{\Gamma''}{\models} \beta$. Нека $S' \overset{\Gamma'}{\models} \alpha' A \overset{\Gamma'}{\vdash} \alpha' a = \alpha$. Всички правила, използвани в $S' \overset{\Gamma'}{\models} \alpha' A$ са в $\mathcal{P}$ и следователно $S' \overset{\Gamma}{\models} \alpha' A$. Вместо $A \to a \in \mathcal{P}'$ в $\mathcal{P}$ е добавено $A \to aS''$, следователно $S' \overset{\Gamma}{\models} \alpha' A \overset{\Gamma}{\vdash} \alpha' aS'' = \alpha S''$. Всички правила на $\mathcal{P}''$ са в $\mathcal{P}$ и следователно $S'' \overset{\Gamma}{\models} \beta$. Комбинираме получените два извода и получаваме $S' \overset{\Gamma}{\models} \alpha S'' \overset{\Gamma}{\models} \alpha\beta$, т.е. $\alpha\beta \in L_\Gamma$.

б) Нека $\omega \in L_\Gamma$, т.е. $S' \overset{\Gamma}{\models} \omega$. Но $S' \in \mathcal{N}'$ и тъй като $\mathcal{N}' \cap \mathcal{N}'' = \emptyset$, началото на този извод е от правила на $\mathcal{P}'$. В $\Gamma$ обаче няма къси правила от $\mathcal{P}'$. Следователно, последното приложено правило е от $\mathcal{P}''$ и краят на извода е с правила от $\mathcal{P}''$. Следователно във вътрешността на извода е приложено правило от вида $A \to aS'', A \in \mathcal{N}'$. Сега изводът може да бъде представен така:

$$S' \overset{\Gamma}{\models} \alpha' A \overset{\Gamma}{\vdash} \alpha' aS'' \overset{\Gamma}{\models} \alpha' a\beta = \alpha\beta,$$

откъдето $S' \overset{\Gamma}{\models} \alpha' aS''$ или $S' \overset{\Gamma'}{\models} \alpha' a = \alpha$ следователно $\alpha \in L_{\Gamma'}$. Освен това $S'' \overset{\Gamma}{\models} \beta$ или $S'' \overset{\Gamma''}{\models} \beta$ и $\beta \in L_{\Gamma''}$. Но $\omega = \alpha\beta$, т.е. $\omega \in L_{\Gamma'}.L_{\Gamma''}$.

На Фиг. 4.3 е показана конструкцията на безкрайното дърво на $\Gamma$, получена от безкрайното дърво на $\Gamma'$ и съответно количество копия на безкрайното дърво на $\Gamma''$, която илюстрира същността на доказаното.




Фигура 4.3: Произведение на автоматни езици.



б) Нека $\varepsilon \in L'$ и $L'_1 = L' \setminus \{\varepsilon\}$, а $\varepsilon \notin L''$. $L'_1$ е автоматен и $\varepsilon \notin L'_1$. Но $L'.L'' = (L'_1 \cup \{\varepsilon\}).L'' = L'_1.L'' + L''$. От доказаното по-горе $L'_1.L''$ е автоматен, от където и сумата $L'_1.L'' + L''$ е автоматен език. Случаят $\varepsilon \notin L', \varepsilon \in L''$ се доказва аналогично.

в) Нека $\varepsilon \in L', \varepsilon \in L''$. Нека $L'_1 = L' \setminus \{\varepsilon\}$ и $L''_1 = L'' \setminus \{\varepsilon\}$, т.е. езиците $L'_1$ и $L''_1$ са автоматни и не съдържат $\varepsilon$. Сега $L'.L'' = (L'_1 \cup \{\varepsilon\})(L''_1 \cup \{\varepsilon\}) = L'_1.L''_1 + L'_1 + L''_1 + \{\varepsilon\}$.

От доказаното по-горе, произведението $L'_1.L''_1$ е автоматен език. Езикът $\{\varepsilon\}$ е също автоматен, защото се поражда от автоматната граматика $\Gamma_\varepsilon =< \{S\}, \{\}, S, \{S \to \varepsilon\} >$. От тук $L'.L''$ е автоматен език, като сума на автоматните езици $L'_1.L''_1$, $L'_1$, $L''_1$ и $\{\varepsilon\}$. □

**Теорема 4.2.3** *Нека $L$ е автоматен език. Тогава $L^*$ е автоматен език.*

***Доказателство.*** Да означим с $\Gamma =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} >$ автоматната граматика, пораждаща $L$ и нека $\Gamma$ е без аксиома в дясна част на правило. Построяваме $\Gamma' =< \mathcal{N}, \mathcal{T}, S, \mathcal{P}' >$, където

$$\mathcal{P}' = (\mathcal{P} \setminus \{S \to \varepsilon\}) \cup \{A \to aS | \forall A \to a \in \mathcal{P}\}.$$

Граматиката $\Gamma'$ е автоматна. Ще покажем, че тя поражда езика $L^* \setminus \{\varepsilon\}$.

а) Нека $\omega \in L_{\Gamma'}$. Тогава $\omega \neq \varepsilon$, защото $S \to \varepsilon \notin \mathcal{P}'$. Разбиваме извода $S \overset{\Gamma'}{\models} \omega$ на части в местата, където се среща $S$. Тъй като $S$ не се среща в дясна част на правило от $\mathcal{P}$, това са точно местата, където са употребени новодобавени правила от вида $A \to aS$, за всяко от които в $\mathcal{P}$ има късо правило $A \to a$. Получаваме

$$S \overset{\Gamma'}{\models} \alpha_1 S \overset{\Gamma'}{\models} \alpha_1\alpha_2 S \overset{\Gamma'}{\models} \cdots \overset{\Gamma'}{\models} \alpha_1\alpha_2 \ldots \alpha_{k-1} S \overset{\Gamma'}{\models} \alpha_1\alpha_2 \ldots \alpha_{k-1}\alpha_k = \omega,$$

т.е.

$$S \overset{\Gamma'}{\models} \alpha_1 S, S \overset{\Gamma'}{\models} \alpha_2 S, \ldots, S \overset{\Gamma'}{\models} \alpha_{k-1} S, S \overset{\Gamma'}{\models} \alpha_k.$$

Изключваме новодобавените дълги правила, заместваики ги със съответни къси и получаваме

$$S \overset{\Gamma}{\models} \alpha_1, S \overset{\Gamma}{\models} \alpha_2, \ldots, S \overset{\Gamma}{\models} \alpha_{k-1}.$$

Последният извод в $\Gamma'$ няма новодобавено правило и директно го преобразуваме в $S \overset{\Gamma}{\models} \alpha_k$. Следователно $\alpha_i \in L, i = 1, 2, \ldots, k, k \geq 1$ и $\omega \in L^k$, т.е. $\omega \in L^* \setminus \varepsilon$.

б) Нека $\omega \in L^*\backslash\{\varepsilon\}$. Тогава $\exists k \geq 1$, такова че $\omega \in L^k$ и $\omega = \alpha_1\alpha_2\cdots\alpha_k$, $\alpha_i \in L, i = 1, 2, \ldots, k$, т.е.

$$S \overset{\Gamma}{\models} \alpha_1, S \overset{\Gamma}{\models} \alpha_2, \ldots, S \overset{\Gamma}{\models} \alpha_k.$$

Заместваме последните (късите) правила на $\Gamma$ в първите $k-1$ извода със съответните им новодобавени и получаваме

$$S \overset{\Gamma'}{\models} \alpha_1 S, S \overset{\Gamma'}{\models} \alpha_2 S, \ldots, S \overset{\Gamma'}{\models} \alpha_{k-1} S.$$

Последният извод директно пренасяме в $\Gamma'$, тъй като всички правила на $\Gamma$ са в $\Gamma'$ и получаваме $S \overset{\Gamma'}{\models} \alpha_k$. Така построихме извода

$$S \overset{\Gamma'}{\models} \alpha_1 S \overset{\Gamma'}{\models} \alpha_1\alpha_2 S \overset{\Gamma'}{\models} \cdots \overset{\Gamma'}{\models} \alpha_1\alpha_2\ldots\alpha_{k-1}S \overset{\Gamma'}{\models} \alpha_1\alpha_2\ldots\alpha_{k-1}\alpha_k = \omega,$$

т.е. $S \overset{\Gamma'}{\models} \omega$ и $\omega \in L_{\Gamma'}$. Следователно $L_{\Gamma'} = L^* \setminus \{\varepsilon\}$ е автоматен. Но в такъв случай и $L^*$ е автоматен. □

**Теорема 4.2.4** *Всеки краен език е автоматен.*

***Доказателство.***

а) Вече показахме, че $\{\varepsilon\}$ е автоматен.

б) Граматиката $\Gamma_\emptyset =< \{S\}, \{a\}, S, \{S \to aS\} >$ е автоматна граматика, която не извежда нито една дума, тъй като няма къси правила. Следователно $\emptyset$ е автоматен език.

в) Нека $L_\alpha = \{\alpha = a_{i_1}a_{i_2}\cdots a_{i_k} | k \geq 1\}$. За езика $L_\alpha$, състоящ се от една дума, построяваме автоматната граматика

$$\begin{aligned}\Gamma_\alpha \;=\; &< \{S, A_1, \ldots, A_{k-1}\}, \{a_{i_1}, a_{i_2}, \cdots a_{i_k}\}, S,\\ &\{S \to a_{i_1}A_1, A_1 \to a_{i_2}A_2, \ldots, A_{k-2} \to a_{i_{k-1}}A_{k-1}, A_{k-1} \to a_{i_k}\} > .\end{aligned}$$

Изводът на $\alpha$ от $S$ е единствено възможният извод на $\Gamma_\alpha$. Следователно всеки език, състоящ се от една дума, е автоматен.

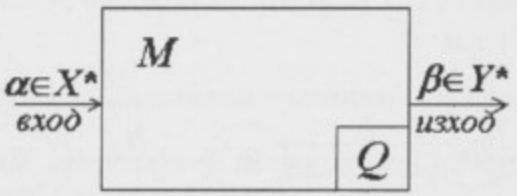
г) Нека $L = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$, т.е. $L = L_{\alpha_1} + L_{\alpha_2} + \ldots + L_{\alpha_m}$. Но всеки $L_{\alpha_i}$ е автоматен и $L$ е автоматен, като крайна сума на автоматни езици. □



### 4.2.2 Крайни автомати

Формалните граматики по естествен начин дефинират езици над множеството на терминалните си символи. Те обаче не позволяват да се даде по един достатъчно ясен начин отговор на въпроси от вида: "Даден е език $L$ чрез граматиката си $\Gamma =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} >$ и дума $\alpha \in \mathcal{T}^*$. Принадлежи ли $\alpha$ на $L$?" В някои отделни случаи (като в примерите от 4.1) можем да опитаме с индуктивни разсъждения да намерим вида на думите на $L_\Gamma$ и ако се окаже, че $\alpha$ е от същия вид, да се даде отговор на въпроса. В общия случай този подход не задоволява, особено като се има предвид, че за нуждите на практиката (например при транслацията на езици за програмиране) често се налага да се решава горната задача за доста сложни граматики и твърде дълги думи (програми).

Съвършено различен подход се получава с използването на *абстрактни математически машини*. Всяка такава машина (вж. Фиг. 4.4) можем да разглеждаме като "черна кутия", която чете от *входа* дума над дадена азбука и може да изведе на *изхода* дума над друга (не непременно различна от входната) азбука. Характерно е, че във всеки момент от работата си машината се намира в някое *състояние* $q$, принадлежащо на крайно множество $Q$ от състояния. Смяната на едно състояние с друго става в дискретно (изброимо) множество от моменти на времето, наричани *тактове*. Между всеки два такта машината остава в едно и също състояние. Новото състояние се определя еднозначно от текущото състояние и входната буква, която машината чете в тактовия момент. Изходната буква, когато машината действително извежда нещо, също е функция на тези




Фигура 4.4: Абстрактна математическа машина.

В началото на работата абстрактната математическа машина винаги се намира в едно и също състояние, наричано *начално* и работата ѝ се определя от цикличното повтаряне на няколко прости действия – прочитане на входна буква, определяне от тази буква и текущото състояние на изходна буква и следващо състояние, извеждане на изходната буква и

смяна на текущото състояние със следващото, изчакване до настъпване на следващия такт, когато действията се повтарят отново.

Абстрактната машина може да завърши работа, когато достигне някое от предварително фиксираните *заключителни състояния* или когато функцията, определяща следващо състояние е частична (машината не може да продължи да работи заради недефинираност). Разбира се, спирането може да се извърши и при прочитане на входната дума докрай или при някое друго, свързано с конкретната дефиниция събитие. Има машини, които са в състояние никога да не завършват работа при определени обстоятелства.

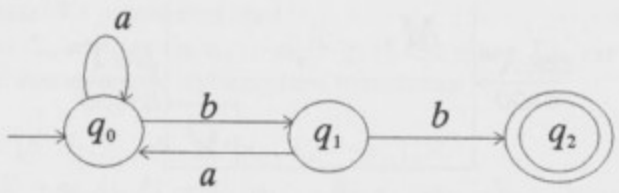
В този раздел ще дефинираме две прости абстрактни машини, краен детерминиран и краен недетерминиран автомат и ще покажем връзката им с автоматните езици.

***Дефиниция.*** *Краен детерминиран автомат (КДА)* наричаме петорката

$$A =< Q, X, q_0, \delta, F >,$$

в която: $Q$ е крайно множество от състояния; $X$ е крайна входна азбука; $q_0 \in Q$ е начално състояние; $\delta : Q \times X \to Q$ е частична *функция на преходите*, пресмятаща следващото състояние; $F \subseteq Q$ са заключителни състояния на КДА.

Представяме КДА с краен ориентиран мултиграф, с върхове елементите на $Q$, в който върхът $q_i \in Q$ и върхът $q_j \in Q$ са свързани с ребро, надписано с $x \in X$, ако $\delta(q_i, x) = q_j$. На Фиг. 4.5 е показан ориентираният мултиграф на автомат, за който $Q = \{q_0, q_1, q_2\}$, $X = \{a, b\}$, $F = \{q_2\}$. Началното състояние $q_0$ е посочено със свободна в единия край стрелка, а заключителното $q_2$ е отбелязано с двойно кръгче.




Фигура 4.5: Краен детерминиран автомат.

Забележете, че разглежданите автомати не произвеждат изход. Те са частен случай на по-общо понятие, което включва изходна азбука и функция, пресмятаща изходите. Това по-общо понятие не представлява



интерес за изложението, затова не се спираме на него тук.

Да разгледаме работата на КДА от Фиг. 4.5 под действието на думата $aaaba$. Когато автоматът е в началното състояние $q_0$ и чете буквата $a$, той остава в това състояние, докато прочете за пръв път буква $b$. Тогава автоматът преминава в $q_1$, а последната буква $a$ го връща в $q_0$. Следователно, започвайки в $q_0$ и прочитайки думата $aaaba$, този автомат ще достигне отново до $q_0$. Ако проследим по същия начин работата на автомата върху думата $abaabb$ ще видим, че след прочитането ѝ автоматът достига до заключителното състояние $q_2$, а при работа над думата $abbb$ автоматът ще спре поради недефинираност на функцията $\delta$ за $(q_2, b)$.

***Дефиниция.*** Определяме *разширена функция на преходите* $\Delta : Q \times X^* \to Q$ по следния начин:

$$\begin{aligned}
\Delta(q, \varepsilon) &= q \qquad \forall q \in Q \\
\Delta(q, x) &= \begin{cases} \delta(q, x) & \forall q, x, \text{ за които } \delta \text{ е дефинирана} \\ \text{недефинирана} & \text{ако } \delta(q, x) \text{ не е дефинирана} \end{cases} \\
\Delta(q, \alpha x) &= \begin{cases} \Delta(\Delta(q, \alpha), x) & \forall q, \alpha, x, \text{ за които } \Delta \text{ е дефинирана} \\ \text{недефинирана} & \text{ако } \Delta(\Delta(q, \alpha), x) \text{ не е дефинирана.} \end{cases}
\end{aligned}$$

Стойността $\Delta(q, \alpha)$ е състоянието, до което ще достигне автоматът $A$, ако започне работа в състояние $q$ и прочете входна дума $\alpha$. За КДА от Фиг. 4.5 с разширена функция на преходите $\Delta$ имаме $\Delta(q_0, aaaba) = q_0$, $\Delta(q_0, abaabb) = q_2 \in F$, а $\Delta(q_0, abbb)$ не е дефинирана, заради недефинираност на функцията $\delta$ за $(q_2, b)$.

**Лема 4.2.1** *Нека $\Delta$ е разширената функция на преходите на КДА $A =< Q, X, q_0, \delta, F >$, а $\alpha_1, \alpha_2 \in X^*$. Тогава, $\forall q \in Q$ е в сила $\Delta(q, \alpha_1\alpha_2) = \Delta(\Delta(q, \alpha_1), \alpha_2)$.*

***Доказателство.*** Ще докажем твърдението с индукция по дължината на $\alpha_2$:

а) Ако $\alpha_2 = \varepsilon$ , т.е. $d(\alpha_2) = 0$, тогава имаме $\Delta(q, \alpha_1\varepsilon) = \Delta(q, \alpha_1)$, а $\Delta(\Delta(q, \alpha_1), \varepsilon) = \Delta(q, \alpha_1)$ по дефиниция.

б) Нека твърдението е в сила за някоя дума $\alpha_2$.

в) Сега нека $x \in X$. $\Delta(q, \alpha_1\alpha_2 x) = \Delta(\Delta(q, \alpha_1\alpha_2), x)$ от дефиницията, а $\Delta(q, \alpha_1\alpha_2) = \Delta(\Delta(q, \alpha_1), \alpha_2)$ от индукционното предположение. Затова $\Delta(q, \alpha_1\alpha_2 x) = \Delta(\Delta(\Delta(q, \alpha_1), \alpha_2), x) = \Delta(\Delta(q, \alpha_1), \alpha_2 x)$ като за последното равенство сме приложили отново дефиницията. □

***Дефиниция.*** Казваме, че КДА $A =< Q, X, q_0, \delta, F >$ с разширена функция на преходите $\Delta$ *разпознава думата* $\alpha \in X^*$, ако $\Delta(q_0, \alpha) \in F$.

Множеството $L_A = \{\alpha | \alpha \in X^*, \Delta(q_0, \alpha) \in F\}$ наричаме *език, разпознаван от крайния детерминиран автомат* $A$.

Автоматът от Фиг. 4.5 разпознава думата $abaabb$, защото $\Delta(q_0, abaabb) = q_2 \in F$, не разпознава $aaaba$, защото $\Delta(q_0, aaaba) = q_0 \notin F$ и не разпознава $abbb$, защото не завършва работата си върху тази дума, поради недефинираността на $\Delta$.

Не е трудно да се съобрази, че езикът на автомата е съставен от всички думи от $\{a, b\}^*$, които завършват на $bb$ и в които няма други две последователни букви $b$.

Ако функцията $\delta$ на КДА $A =< Q, X, q_0, \delta, F >$ не е тотална, тогава можем да разширим $A$ до автомата $A' =< Q' = Q \cup \{q*\}, X, q_0, \delta', F >$, такъв че $q* \notin F$, $\delta' : Q' \times X \to Q'$, а

$$\delta'(q,x) = \begin{cases} \delta(q,x) & \text{ако } \delta(q,x) \text{ е дефинирана} \\ q^* & \text{ако } \delta(q,x) \text{ не е дефинирана} \end{cases}$$

Действително, всички думи, разпознавани от $A$ се разпознават и от $A'$. Всяка дума $\alpha$, за която $\Delta_A(q_0, \alpha) \notin F$, и значи не се разпознава от $A$ – не се разпознава и от $A'$, а за всяка дума $\beta$, за която $\Delta_A(q_0, \beta)$ не е дефинирана, в $A'$ имаме $\Delta_{A'}(q_0, \beta) = q^* \notin F$ и $\beta$ отново не се разпознава. Така $L_A = L_{A'}$ и следователно можем да работим с недодефинирани автомати или да ги додефинираме (без да изменяме езика им), когато това е необходимо.

Сега да видим каква е връзката между крайните детерминирани автомати и автоматните езици.

**Теорема 4.2.5** *За всеки КДА* $A =< Q, X, q_0, \delta, F >$ *съществува автоматна граматика* $\Gamma$ *такава, че* $L_\Gamma = L_A$.

***Доказателство.*** Строим $\Gamma =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} >$ такава, че $\mathcal{N} = Q$, $\mathcal{T} = X$, $S = q_0$, $\mathcal{P} = \{q_i \to xq_j | \text{ако } \delta(q_i, x) = q_j\} \cup \{q_i \to x | \text{ако } \delta(q_i, x) \in F\}$. Ако $\varepsilon \in L_A$, изчистваме аксиомата от дясна част на правилата и добавяме правилото $q_0 \to \varepsilon$ (без ограничение на общността означаваме и в този случай с $q_0$ аксиомата, а с $\Gamma$ получената граматика).

Очевидно $\Gamma$ е автоматна граматика. Ще покажем, че $L_\Gamma = L_A$.

а) Нека $\omega \in L_A$. Ако $\omega = \varepsilon$, тогава в $\Gamma$ сме включили $q_0 \to \varepsilon$ и следователно $\varepsilon \in L_\Gamma$.

Нека сега $\omega = x_{i_1} x_{i_2} \ldots x_{i_k} \neq \varepsilon$. От $\Delta(q_0, \omega) \in F$ получаваме: $\exists q_{i_1}, q_{i_2}, \ldots, q_{i_{k-1}}$ такива, че $\delta(q_0, x_{i_1}) = q_{i_1}$, $\delta(q_{i_1}, x_{i_2}) = q_{i_2}$, $\ldots$, $\delta(q_{i_{k-2}}, x_{i_{k-1}}) = q_{i_{k-1}}$ и $\delta(q_{i_{k-1}}, x_{i_k}) \in F$.



Следователно в $\mathcal{P}$ има правила

$$q_0 \to x_{i_1} q_{i_1}, q_{i_1} \to x_{i_2} q_{i_2}, \ldots, q_{i_{k-2}} \to x_{i_{k-1}} q_{i_{k-1}}, q_{i_{k-1}} \to x_{i_k},$$

т.е. можем да построим извода

$$q_0 \vdash x_{i_1} q_{i_1} \vdash x_{i_1} x_{i_2} q_{i_2} \vdash \ldots \vdash x_{i_1} x_{i_2} \ldots x_{i_{k-1}} q_{i_{k-1}} \vdash x_{i_1} x_{i_2} \ldots x_{i_k} = \omega$$

или $q_0 \models \omega$ и $\omega \in L_\Gamma$.

б) Нека $\omega \in L_\Gamma$. Ако $\omega = \varepsilon$, тогава в $\Gamma$ има правило $q_0 \to \varepsilon$, което е включено в $\mathcal{P}$, защото $\varepsilon \in L_A$.

Нека сега $\omega = x_{i_1} x_{i_2} \ldots x_{i_k} \neq \varepsilon$. От $q_0 \models \omega$ получаваме

$$q_0 \vdash x_{i_1} q_{i_1} \vdash x_{i_1} x_{i_2} q_{i_2} \vdash \ldots \vdash x_{i_1} x_{i_2} \ldots x_{i_{k-1}} q_{i_{k-1}} \vdash x_{i_1} x_{i_2} \ldots x_{i_k} = \omega,$$

т.е. съществуват правила

$$q_0 \to x_{i_1} q_{i_1}, q_{i_1} \to x_{i_2} q_{i_2}, \ldots, q_{i_{k-2}} \to x_{i_{k-1}} q_{i_{k-1}}, q_{i_{k-1}} \to x_{i_k},$$

които са били включени в $\mathcal{P}$, защото $\delta(q_0, x_{i_1}) = q_{i_1}$, $\delta(q_{i_1}, x_{i_2}) = q_{i_2}$, $\ldots$, $\delta(q_{i_{k-2}}, x_{i_{k-1}}) = q_{i_{k-1}}$ и $\delta(q_{i_{k-1}}, x_{i_k}) \in F$. Но това означава $\Delta(q_0, \omega) \in F$ и значи $\omega \in L_A$. □

За автомата от Фиг. 4.5 получаваме автоматната граматика $\Gamma =< \{q_0, q_1, q_2\}, \{a, b\}, q_0, \{q_0 \to aq_0, q_0 \to bq_1, q_1 \to aq_0, q_1 \to bq_2, q_1 \to b\} >$.

Би било добре, ако конструкцията на теоремата можеше да се обърне и да получим подобно твърдение и в другата посока. Това не може да стане директно по следната причина. Нека автоматна граматика съдържа правилата $A \to xB, A \to xC, A \to xD$ и т.н. Обратната конструкция не може еднозначно да определи стойността на функцията $\delta(A, x)$, защото в граматиката има повече от един нетерминал, кандидат за това значение. Това ни навежда на мисълта за следната, по-сложна абстрактна математическа машина.

***Дефиниция.*** Петорката $A =< Q, X, q_0, \delta, F >$, където $Q, X, q_0$ и $F$ са дефинирани както при КДА, а *функцията на преходите* е $\delta : Q \times X \to 2^Q$, наричаме *краен недетерминиран автомат (КНА)*.

Крайният недетерминиран автомат работи по следния начин, общ за всички недетерминирани абстрактни математически машини: Когато в резултат на изчисление на функцията $\delta$ се получи множество $Q'$ от състояния, в които КНА трябва да премине, той се размножава в $|Q'|$ копия и всяко копие преминава в едно от състоянията на $Q'$. Можем да си мислим, че автоматът едновременно се намира във всичките състояния на $Q'$. Когато КНА прочете следващата буква, всяко от копията

се размножава в толкова нови копия, колкото текущото му състояние, входната буква и функцията $\delta$ определят и всяко едно копие преминава в съответното състояние. Множеството от състояния, в които се намира КНА ще съдържа състоянията на всички получени копия. Тази не съвсем формална представа ще уточним със следната

***Дефиниция.*** Нека $A =< Q, X, q_0, \delta, F >$ е КНА. *Разширена функция на преходите* $\Delta : Q \times X^* \to 2^Q$ на $A$ дефинираме така:

$$\begin{aligned}
\Delta(q,\varepsilon) &= \{q\} \\
\Delta(q,x) &= \begin{cases} \delta(q,x) & \forall q, x, \text{ за които } \delta \text{ е дефинирана} \\ \text{недефинирана} & \text{ако } \delta(q,x) \text{ не е дефинирана} \end{cases} \\
\Delta(q,\alpha x) &= \bigcup_{i=1}^{l} \delta(q_{p_i}, x) \qquad \Delta(q,\alpha) = \{q_{p_1}, q_{p_2}, \ldots, q_{p_l}\}.
\end{aligned}$$

***Дефиниция.*** Казваме, че КНА $A$ *разпознава думата* $\alpha \in X^*$, ако $\Delta(q_0, \alpha) \cap F \neq \emptyset$. *Език, разпознаван от КНА* $A$ определяме като

$$L_A = \{\alpha | \alpha \in X^*, \Delta(q_0, \alpha) \cap F \neq \emptyset\}.$$

**Теорема 4.2.6** *За всяка автоматна граматика* $\Gamma$ *съществува КНА* $A$ *такъв, че* $L_\Gamma = L_A$.

***Доказателство.*** Нека $\Gamma =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} >$. Да изберем $E \notin \mathcal{N} \cup \mathcal{T}$ и да конструираме КНА $A =< \mathcal{N} \cup \{E\}, \mathcal{T}, S, \delta, F >$, където

$$F = \begin{cases} \{E\}, & \varepsilon \notin L_\Gamma \\ \{E, S\}, & \varepsilon \in L_\Gamma \end{cases}$$

а

$$\delta(A, x) = \{B_i | \forall A \to xB_i \in \mathcal{P}\} \cup \{E | \text{ако } A \to x \in \mathcal{P}\}.$$

В общия случай полученият автомат ще бъде недетерминиран. Ще покажем, че $L_\Gamma = L_A$.

а) Нека $\omega \in L_A$. Ако $\omega = \varepsilon$ тогава от $\Delta(S, \varepsilon) \cap F \neq \emptyset$ и $\Delta(S, \varepsilon) = \{S\}$ следва $S \in F$ и следователно $\varepsilon \in L_\Gamma$.

Нека $\omega \neq \varepsilon$ и $\omega = x_{i_1} x_{i_2} \ldots x_{i_k}$. Сега $\Delta(S, x_{i_1} x_{i_2} \ldots x_{i_k}) \cap F \neq \emptyset$ означава, че съществуват $A_{i_1}, A_{i_2}, \ldots, A_{i_{k-1}}$, такива че $\delta(S, x_{i_1}) \ni A_{i_1}, \delta(A_{i_1}, x_{i_2}) \ni A_{i_2}, \ldots, \delta(A_{i_{k-2}}, x_{i_{k-1}}) \ni A_{i_{k-1}}$ и $\delta(A_{i_{k-1}}, x_{i_k}) \cap F \neq \emptyset$. Това, съгласно конструкцията на КНА означава, че в $\mathcal{P}$ има правила

$$S \to x_{i_1} A_{i_1}, A_{i_1} \to x_{i_2} A_{i_2}, \ldots, A_{i_{k-2}} \to x_{i_{k-1}} A_{i_{k-1}}.$$

От $\delta(A_{i_{k-1}}, x_{i_k}) \cap F \neq \emptyset$ следва $\delta(A_{i_{k-1}}, x_{i_k}) \ni E$, защото или $F = \{E\}$ и $E$ е единствен елемент на $F$, от който може да се получи непразното



сечение, или $F = \{S, E\}$, но тогава $\varepsilon \in L_\Gamma, S \to \varepsilon \in \mathcal{P}$ и аксиомата не се среща в дясна част на правило. В последния случай не е възможно $\delta(A_{i_{k-1}}, x_{i_k}) \ni S$. От тук следва, че $\exists A_{i_{k-1}} \to x_{i_k} \in \mathcal{P}$. Получаваме извода

$$S \vdash x_{i_1} A_{i_1} \vdash x_{i_1} x_{i_2} A_{i_2} \vdash \cdots \vdash x_{i_1} \ldots x_{i_{k-1}} A_{i_{k-1}} \vdash x_{i_1} \ldots x_{i_{k-1}} x_{i_k} = \omega,$$

т.е. $S \models \omega$ и $\omega \in L_\Gamma$.

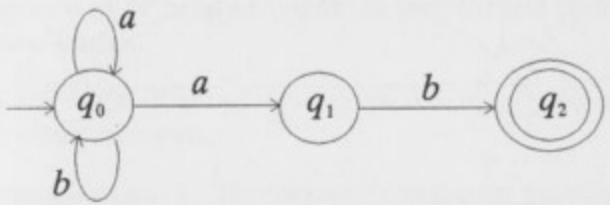
б) Нека $\omega \in L_\Gamma$. Ако $\omega = \varepsilon$ тогава $S \in F$ и тъй като $\Delta(S, \varepsilon) = \{S\}$, имаме $\Delta(S, \varepsilon) \cap F \neq \emptyset$ и следователно $\varepsilon \in L_A$.

Ако $\omega = x_{i_1} x_{i_2} \ldots x_{i_k} \neq \varepsilon$, от $S \models \omega$ получаваме

$$S \vdash x_{i_1} A_{i_1} \vdash x_{i_1} x_{i_2} A_{i_2} \vdash \cdots \vdash x_{i_1} \ldots x_{i_{k-1}} A_{i_{k-1}} \vdash x_{i_1} \ldots x_{i_{k-1}} x_{i_k} = \omega,$$

т.е. $\delta(S, x_{i_1}) \ni A_{i_1}, \delta(A_{i_1}, x_{i_2}) \ni A_{i_2}, \ldots, \delta(A_{i_{k-2}}, x_{i_{k-1}}) \ni A_{i_{k-1}}, \delta(A_{i_{k-1}}, x_{i_k}) \ni E$ или $\Delta(S, x_{i_1} x_{i_2} \ldots x_{i_k}) \ni E$ и $\Delta(S, \omega) \cap F \neq \emptyset$. Това означава, че $\omega \in L_A$. □

На Фиг. 4.6 е показан един краен недетерминиран автомат (забележете, че $\delta(q_0, a) = \{q_0, q_1\}$) с три състояния. Той разпознава езика, съставен от всички думи на азбуката $\{a, b\}$, които завършват на $ab$. Той е получен с конструкцията на доказаната теорема, от автоматната граматика $\Gamma =< \{S, A\}, \{a, b\}, S, \{S \to aS, S \to bS, S \to aA, A \to b\} >$ при $S = q_0, A = q_1$ и $E = q_2$.




Фигура 4.6: Краен недетерминиран автомат.

За да завършим описанието на връзката между автоматните езици и детерминираните крайни автомати, ще докажем следната

**Теорема 4.2.7** *За всеки КНА* $A$ *съществува КДА* $A'$ *такъв, че* $L_A = L_{A'}$.

***Доказателство.*** Нека $A =< Q, X, q_0, \delta, F >$. Построяваме КДА $A' =< Q', X, t_0, \delta', F' >$, където $Q' \subseteq 2^Q$. Нека множеството $\{q_{p_1}, q_{p_2}, \ldots, q_{p_l}\} \in Q'$. За по кратко ще го означаваме с $t_{[p_1, p_2, \ldots, p_l]}$. При това означение

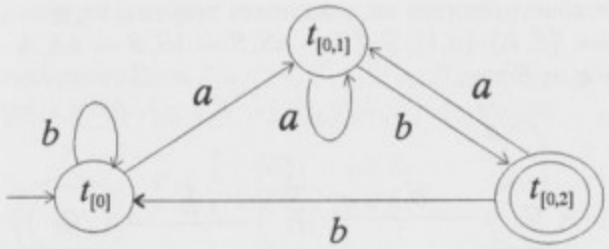
определяме $t_0 = \{q_0\} = t_{[0]}$. Нека $F' = \{t_{[p_1,p_2,\ldots,p_l]} | \{q_{p_1}, q_{p_2}, \ldots, q_{p_l}\} \cap F \neq \emptyset\}$, а $\delta'(t_{[p_1,p_2,\ldots,p_l]}, x) = t_{[r_1,r_2,\ldots,r_m]}$, ако $\{q_{r_1}, q_{r_2}, \ldots, q_{r_m}\} = \bigcup_{i=1}^{l} \delta(q_{p_i}, x)$. Забележете, че не можем да фиксираме в явен вид кои точно подмножества на $Q$ влизат в $Q'$. Те се определят от изчисляването на функцията $\delta'$, започвайки от $\delta'(t_{[0]}, x), \forall x \in X$. С индукция по дължината на $\alpha$ ще покажем, че $\Delta_{A'}(t_{[0]}, \alpha) = t_{[p_1,p_2,\ldots,p_l]}$ т.с.т.к. $\Delta_A(q_0, \alpha) = \{q_{p_1}, q_{p_2}, \ldots, q_{p_l}\}$.

а) Нека $\omega = \varepsilon$. Тогава $\Delta_{A'}(t_{[0]}, \varepsilon) = t_{[0]}$, а $\Delta_A(q_0, \varepsilon) = \{q_0\}$ и твърдението е в сила.

б) Допускаме верността на твърдението за някоя дума $\alpha$ и ще покажем, че $\Delta_{A'}(t_{[0]}, \alpha x) = t_{[r_1,r_2,\ldots,r_m]}$ т.с.т.к. $\Delta_A(q_0, \alpha x) = \{q_{r_1}, q_{r_2}, \ldots, q_{r_m}\}$.

в) Нека $\Delta_{A'}(t_{[0]}, \alpha x) = t_{[r_1,r_2,\ldots,r_m]}$, като $\Delta_{A'}(t_{[0]}, \alpha) = t_{[p_1,p_2,\ldots,p_l]}$. Тогава $\delta'(t_{[p_1,p_2,\ldots,p_l]}, x) = t_{[r_1,r_2,\ldots,r_m]}$ и съгласно построението на $\delta'$ $\{q_{r_1}, q_{r_2}, \ldots, q_{r_m}\} = \bigcup_{i=1}^{l} \delta(q_{p_i}, x)$. Но от индуктивното допускане $\Delta_A(q_0, \alpha) = \{q_{p_1}, q_{p_2}, \ldots, q_{p_l}\}$ и следователно $\Delta_A(q_0, \alpha x) = \{q_{r_1}, q_{r_2}, \ldots, q_{r_m}\}$. Разсъжденията в другата посока са подобни.

Като вземем предвид дефиницията на $F'$ и току що доказаното, получаваме $\Delta_{A'}(t_{[0]}, \alpha) \in F'$ т.с.т.к. $\Delta_A(q_0, \alpha) \cap F \neq \emptyset$ и значи $L_{A'} = L_A$. □




Фигура 4.7: Детерминизация на краен недетерминиран автомат.

На Фиг. 4.7 е показан резултатът от детерминизацията на КНА от Фиг. 4.6.

### 4.2.3 Регулярни изрази и езици

Ще разгледаме още един начин за задаване на езици – чрез думите на едно разширение на образуващата ги азбука. Нека $X = \{x_1, x_2, \ldots, x_n\}$ е крайна азбука. Разширяваме я до $\tilde{X}$ с буквите от $\{\varepsilon, \emptyset, *, +, (, )\}$ (ако някоя от тези букви е била елемент на $X$, можем да я заменим с друг знак, който не участва в $X$). Паралелно ще дефинираме понятията *регулярен израз* и *език, съпоставен на регулярен израз (регулярен език)*.



***Дефиниция.*** Нека $\tilde{X} = X \cup \{\varepsilon, \emptyset, *, +, (, )\}$

а) Думите $\varepsilon, \emptyset$ и $x_i, i = 1, 2, \ldots, n$ от $\tilde{X}^*$ са регулярни изрази. Съответните им регулярни езици над $X$ са $\{\varepsilon\}, \emptyset$ и $\{x_i\}, i = 1, 2, \ldots, n$.

б) Нека $\alpha, \beta \in \tilde{X}^*$ са регулярни изрази и съответните им регулярни езици са $L_\alpha$ и $L_\beta$. Тогава $(\alpha) + (\beta)$, $(\alpha).(\beta)$ и $(\alpha)^*$ са регулярни изрази, а съответните им езици са $L_\alpha + L_\beta, L_\alpha.L_\beta$ и $L_\alpha^*$.

в) Няма други регулярни изрази и регулярни езици.

Въвеждаме приоритет на операциите итерация, произведение и сума, намаляващ в зададения ред. Така употребата на скоби може да бъде силно ограничена.

Нека $X = \{a, b\}$. По-долу сме дали няколко примера на регулярни изрази и съответните им езици.

| Израз | Език |
|---|---|
| $a$ | $\{a\}$ |
| $a + b$ | $\{a, b\}$ |
| $a(a + b)$ | $\{aa, ab\}$ |
| $a^*$ | $\{\varepsilon, a, a^2, \ldots, a^i, \ldots\}$ |
| $a^*(a + b)$ | $\{a, b, aa, ab, a^2a, a^2b, \ldots, a^ia, a^ib, \ldots\}$ |
| $(a + b)^*$ | $X^*$ |
| $a^*b + b^*a$ | $\{b, a, ab, ba, a^2b, b^2a, \ldots, a^ib, b^ia, \ldots\}$ |

Интересно е да се определи видът на регулярните езици според йерархията на Чомски.

**Теорема 4.2.8 (Ст. Клини)** *Множествата на регулярните и автоматните езици съвпадат.*

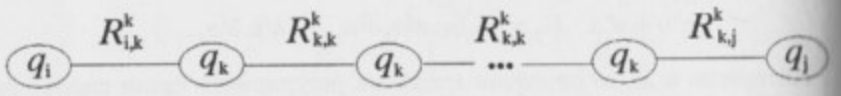
***Доказателство.*** 1) Ще покажем с индукция по дефиницията на регулярни езици, че всеки регулярен език е автоматен. Действително $\{\varepsilon\}, \emptyset$ и $\{x_i\}, i = 1, 2, \ldots, n$ са автоматни езици, защото са крайни. Ако допуснем, че регулярните езици $L_\alpha$ и $L_\beta$, съответни на регулярните изрази $\alpha$ и $\beta$ са автоматни, тогава $L_\alpha + L_\beta, L_\alpha.L_\beta$ и $L_\alpha^*$ са автоматни, защото са сума, произведение и итерация на автоматни езици, съответно. Тъй като други регулярни езици няма, всеки регулярен език е автоматен.

2) Нека езикът $L$ е автоматен. Ще докажем, че $L$ е регулярен език. Съществува КДА $A =< Q, X, q_0, \delta, F >$ такъв, че $L = L_A$. Нека автоматът $A$ е представен с крайния ориентиран мултиграф $G$. Нека състоянията на $A$ са $Q = \{q_0, q_1, \ldots, q_n\}$ и $F = \{q_{p_1}, q_{p_2}, \ldots, q_{p_r}\}$. Да означим с $R_{ij}^k$ множеството от маршрутите в $G$ от връх $q_i$ до връх $q_j$, които не използват като вътрешни върхове $q_k, q_{k+1}, \ldots, q_n$. Очевидно, всеки маршрут от

$q_i$ до $q_j$ еднозначно определя дума $\alpha \in X^*$, такава че $\Delta(q_i, \alpha) = q_j$. Така на множеството от маршрути $R_{ij}^k$ можем да гледаме като на множество от съответните думи от $X^*$, т.е. $R_{ij}^k$ е език над $X^*$ – *маршрутен език*. В автомата няма състояния с номера по-големи от $n$ и затова $R_{ij}^k$ е езикът на всички маршрути от $q_i$ до $q_j$, когато $k > n$.

**Лема 4.2.2** *За всеки $q_i, q_j \in Q$ е в сила $R_{ij}^{k+1} = R_{ij}^k + R_{ik}^k (R_{kk}^k)^* R_{kj}^k$, $k = 0, 1, \ldots, n$.*

***Доказателство.*** Маршрутите $R_{ij}^{k+1}$, не минаващи през $q_{k+1}, \ldots, q_n$ разбиваме на две подмножества: такива, които не минават през $q_k$ – $R_{ij}^k$ и такива които минават през $q_k$ – $R'$. Получаваме, че $R_{ij}^{k+1} = R_{ij}^k + R'$. Всеки маршрут от $q_i$ до $q_j$, минаващ през $q_k$ и не минаващ през $q_{k+1}, \ldots, q_n$ да разбием на части с краища срещанията на $q_k$ в маршрута (вж. Фиг. 4.8). Всеки такъв маршрут започва с подмаршрут от $R_{ik}^k$ и завършва с подмаршрут от $R_{kj}^k$, между които може да има произволен брой $(0, 1, 2, \ldots)$ подмаршрути от $R_{kk}^k$, т.е. $R' = R_{ik}^k (R_{kk}^k)^* R_{kj}^k$ и $R_{ij}^{k+1} = R_{ij}^k + R_{ik}^k (R_{kk}^k)^* R_{kj}^k$. □




Фигура 4.8: Разбиване на маршрут от $R_{ij}^{k+1}$.

**Лема 4.2.3** *За всеки $q_i, q_j \in Q$ $R_{ij}^k$, $k = 0, 1, \ldots, n+1$ е регулярен език.*

***Доказателство.*** Твърдението ще докажем с индукция по $k$.

а) Нека $k = 0$. Ако $i = j$, $R_{ii}^0$ са всички маршрути от $q_i$ до $q_i$, не минаващи през никой друг връх. Ако мултиграфът на автомата няма примки в $q_i$, единствен такъв маршрут е празният, без нито едно ребро и съответната му дума е $\varepsilon$. Тогава $R_{ii}^0 = \{\varepsilon\}$ е регулярен език. Другата възможност е в мултиграфа да има примки от $q_i$ до $q_i$, надписани с буквите $x_{i_1}, x_{i_2}, \ldots, x_{i_s}$. Тогава $R_{ii}^0 = \{\varepsilon, x_{i_1}, x_{i_2}, \ldots, x_{i_s}\}$ е регулярен език, защото се представя с регулярния израз $\varepsilon + x_{i_1} + x_{i_2} + \cdots + x_{i_s}$.

Ако $i \neq j$ тогава $R_{ij}^0$ са всички маршрути от $q_i$ до $q_j$, не минаващи през никой от останалите върхове, т.е. ребрата от $q_i$ до $q_j$. Ако няма такива ребра $R_{ij}^0 = \emptyset$ е регулярен език, а ако $x_{i_1}, x_{i_2}, \ldots, x_{i_s}$ са буквите



по ребрата от $q_i$ до $q_j$, тогава $R_{ij}^0 = \{x_{i_1}, x_{i_2}, \ldots, x_{i_s}\}$ и $R_{ij}^0$ е регулярен език, защото се представя с регулярния израз $x_{i_1} + x_{i_2} + \cdots + x_{i_s}$.

б) Да допуснем, че при някое $k$, $R_{ij}^k$ е регулярен език $\forall q_i, q_j \in Q$ т.е. съществува регулярен израз $\alpha_{ij}^k$ за него.

в) Езикът $R_{ij}^{k+1}$ е също регулярен, защото (съгласно Лема 4.2.2) се представя с регулярния израз $\alpha_{ij}^{k+1} = \alpha_{ij}^k + \alpha_{ik}^k (\alpha_{kk}^k)^* \alpha_{kj}^k$. □

Сега да се върнем към доказателството на Теорема 4.2.8. От дефиницията на език, разпознаван от КДА получаваме $L_A = R_{0p_1}^{n+1} + R_{0p_2}^{n+1} + \cdots + R_{0p_r}^{n+1}$, тъй като езикът на $A$ се състои от всички маршрути (разбирай съответните им думи), довеждащи автомата от началното състояние $q_0$ до някое от крайните $q_{p_1}, q_{p_2}, \ldots, q_{p_r}$. Следователно $L_A$ е регулярен език, защото е крайна сума на регулярни езици. □

С помощта на Лема 4.2.2 можем да решим задачата: "Зададен е автоматен език (с КДА, който го разпознава). Да се построи регулярен израз за този език." За автомата от Фиг. 4.7 при подреждане на състоянията $q_0 = t_{[0]}, q_1 = t_{[0,1]}, q_2 = t_{[0,2]}$, получаваме: $L_A = R_{02}^3 = R_{02}^2 + R_{02}^2 (R_{22}^2)^* R_{22}^2$, но

$$\begin{aligned} R_{02}^2 &= R_{02}^1 + R_{01}^1 (R_{11}^1)^* R_{12}^1 = \\ &= \emptyset + b^* a (a)^* b = b^* a a^* b, \\ R_{22}^2 &= R_{22}^1 + R_{21}^1 (R_{11}^1)^* R_{12}^1 = \\ &= \emptyset + (a + bb^* a)(a)^* b = b^* a a^* b. \end{aligned}$$

Сега $R_{02}^2 = R_{22}^2 = R$ и $L_A = R + RR^*R = R(\varepsilon + R^*R) = R(\varepsilon + R^+) = RR^* = R^*R = (b^*aa^*b)^*b^*aa^*b = (b^*a^*ab)^*b^*a^*ab$. Ще покажем, че $(b^*a^*ab)^*b^*a^* = (a+b)^*$, т.е. съответният му език съдържа всяка дума на азбуката $\{a, b\}$. Очевидно $\varepsilon$ е от езика на израза $(b^*a^*ab)^*b^*a^*$. Затова да разгледаме произволна дума $\alpha \neq \varepsilon$. Да намерим в $\alpha$ всички места, в които се среща комбинацията от букви $ab$. Очевидно е, че между всеки две двойки $ab$ може да стои само дума от вида $b^*a^*$, а след последното срещане на $ab$ – дума от същия вид. Следователно, всяка дума от $(a+b)^*$, различна от $\varepsilon$ се представя във вида $(b^*a^*ab)^*b^*a^*$. И така получаваме $L_A = (a+b)^*ab$ – факт, който се вижда лесно в недетерминирания автомат за същия език (вж. Фиг. 4.6).

### 4.2.4 Минимизация на КДА

***Дефиниция.*** Крайните детерминирани автомати $A$ и $A'$ наричаме *еквивалентни*, ако $L_A = L_{A'}$.

Естествено е желанието, когато език се разпознава от повече от един автомат, да се опитаме да намерим този разпознавател, който е най-прост.

***Дефиниция.*** КДА $A_0$, разпознаващ автоматния език $L$, наричаме *минимален* (за езика $L$), ако за всеки друг автомат $A$, разпознаващ $L$ $|Q_0| \leq |Q|$, където $Q_0$ и $Q$ са множествата от състояния на $A_0$ и $A$ съответно.

В този раздел ще се занимаем с намиране на минималния автомат за автоматния език $L$ по зададен КДА $A$, разпознаващ $L$. Тази задача обикновено наричаме задача за *минимизация* на автомата $A$.

***Дефиниция.*** Нека $A =< Q, X, q_0, \delta, F >$ е краен детерминиран автомат. Състоянието $q \in Q$ наричаме *недостижимо* , ако $\forall \alpha \in X^*$, $\Delta(q_0, \alpha) \neq q$, а *достижимо*, ако $\exists \alpha \in X^*, \Delta(q_0, \alpha) = q$.

За даден КДА лесно намираме достижимите състояния с обхождане на мултиграфа на автомата в ширина, започвайки от началното състояние. Всички обходени състояния са достижими, а всички необходени – не. Оставяме на читателя да докаже, че ако премахнем от автомата всички недостижими състояния и свързаните с тях ребра, няма да променим езика на автомата. Затова оттук нататък ще работим с навсякъде дефинирани КДА, без недостижими състояния.

***Дефиниция.*** Релацията $R \subseteq X^* \times X^*$, където $X$ е произволна крайна азбука наричаме *дясно-инвариантна*, ако $\forall \gamma \in X^*$ е в сила $(\alpha, \beta) \in R \Rightarrow (\alpha\gamma, \beta\gamma) \in R$.

***Дефиниция.*** Нека $A =< Q, X, q_0, \delta, F >$ е КДА с разширена функция на преходите $\Delta$. Релацията $R_A \subseteq X^* \times X^*$ определяме така

$$R_A = \{(\alpha, \beta) | \Delta(q_0, \alpha) = \Delta(q_0, \beta)\}.$$

**Лема 4.2.4** *Релацията $R_A$ за КДА $A$ е релация на еквивалентност и е дясно-инвариантна.*

***Доказателство.*** А) $(\alpha, \alpha) \in R_A$, $\forall \alpha \in X^*$, защото $\Delta(q_0, \alpha) = \Delta(q_0, \alpha)$. Ако $(\alpha, \beta) \in R_A$, то $(\beta, \alpha) \in R_A$ защото дефиницията е симетрична. Ако $(\alpha, \beta) \in R_A$ и $(\beta, \gamma) \in R_A$, то очевидно $(\alpha, \gamma) \in R_A$, защото $\Delta(q_0, \alpha) = \Delta(q_0, \beta) = \Delta(q_0, \gamma)$. Следователно, $R_A$ е релация на еквивалентност.

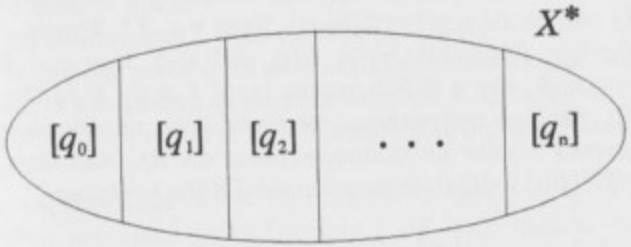
Б) Нека $(\alpha, \beta) \in R_A$ и $\gamma \in X^*$.

$$\Delta(q_0, \alpha\gamma) = \Delta(\Delta(q_0, \alpha), \gamma) = \Delta(\Delta(q_0, \beta), \gamma) = \Delta(q_0, \beta\gamma),$$

следователно $(\alpha\gamma, \beta\gamma) \in R_A$ и $R_A$ е дясно-инвариантна. □



Тъй като $A$ е без недостижими състояния, за всяко $q \in Q$ определяме всички думи от $X^*$, които довеждат автомата от $q_0$ до $q$. Очевидно те образуват клас на еквивалентност на $R_A$ и така с всяко състояние се свързва един клас на еквивалентност на релацията $R_A$ (вж. Фиг. 4.9).




Фигура 4.9: Класове на еквивалентност на $R_A$.

За индекса на релацията на еквивалентност $R_A$ на КДА $A$ очевидно имаме $IX(R_A) = |Q|$.

***Дефиниция.*** Нека $L \subseteq X^*$. Релацията $R_L \subseteq X^* \times X^*$ се състои от такива двойки $(\alpha, \beta)$, че $\forall \gamma \in X^*$ $\alpha\gamma$ и $\beta\gamma$ едновременно са или не са в $L$.

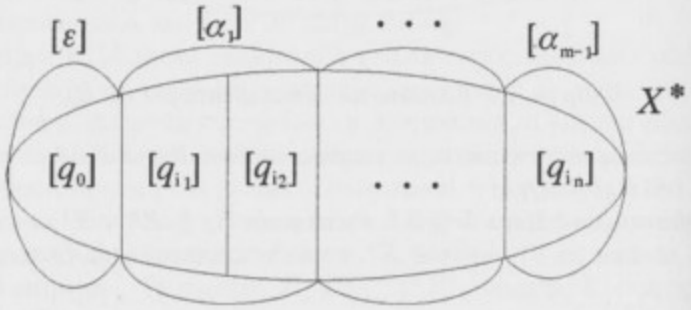
**Лема 4.2.5** *Релацията $R_L$ за езика $L \subseteq X^*$ е релация на еквивалентност и е дясно-инвариантна.*

***Доказателство.*** А) $(\alpha, \alpha) \in R_L, \forall \alpha \in X^*$, защото $\alpha\gamma$ и $\alpha\gamma$ едновременно са или не са в $L$, $\forall \gamma \in X^*$. Нека $(\alpha, \beta) \in R_L$, т.е. $\alpha\gamma$ и $\beta\gamma$ едновременно са или не са в $L$, $\forall \gamma \in X^*$. Тогава $\beta\gamma$ и $\alpha\gamma$ едновременно са или не са в $L$, $\forall \gamma \in X^*$ и следователно $(\beta, \alpha) \in R_L$. Нека $(\alpha, \beta) \in R_L$ и $(\beta, \phi) \in R_L$. Следователно, $\forall \gamma \in X^*$, $\alpha\gamma$ и $\beta\gamma$ едновременно са или не са в $L$ и $\beta\gamma$ и $\phi\gamma$ едновременно са или не са в $L$. Тогава $\alpha\gamma$ и $\phi\gamma$ едновременно са или не са в $L$, $\forall \gamma \in X^*$ и $(\alpha, \phi) \in R_L$. Следователно $R_L$ е релация на еквивалентност.

Б) Нека $(\alpha, \beta) \in R_L$, т.е. $\forall \gamma \in X^*$ $\alpha\gamma$ и $\beta\gamma$ едновременно са или не са в $L$. Да разбием $\gamma$ по произволен начин на $\gamma = \gamma'\gamma''$, т.е. $\forall \gamma'' \in X^*$ $(\alpha\gamma')\gamma''$ и $(\beta\gamma')\gamma''$ едновременно са или не са в $L$. Следователно $\forall \gamma' \in X^*$ $(\alpha\gamma', \beta\gamma') \in R_L$. Следователно $R_L$ е дясно-инвариантна. □

**Теорема 4.2.9 (Нерод-Майхил)** *Нека $L \subseteq X^*$. Релацията $R_L \subseteq X^* \times X^*$ има краен индекс т.с.т.к. $L$ е автоматен.*

***Доказателство.*** 1) Нека $L$ е автоматен език. Тогава съществува КДА $A =< Q, X, q_0, \delta, F >$, който разпознава $L$. Без ограничение на общността можем да считаме, че сме се освободили от недостижимите състояния на $A$, което не се отразява на разпознавания език. Образуваме релацията $R_A$. Нека $(\alpha, \beta) \in R_A$. Ще покажем , че $(\alpha, \beta) \in R_L$. От $(\alpha, \beta) \in R_A$ следва $\Delta(q_0, \alpha) = \Delta(q_0, \beta)$. Нека $\gamma \in X^*$. Тогава $\Delta(q_0, \alpha\gamma) = \Delta(\Delta(q_0, \alpha), \gamma) = \Delta(\Delta(q_0, \beta), \gamma) = \Delta(q_0, \beta\gamma) = q$. Или $q \in F$ и тогава $\alpha\gamma \in L$ и $\beta\gamma \in L$, или $q \notin F$ и тогава $\alpha\gamma \notin L$ и $\beta\gamma \notin L$. Следователно $(\alpha, \beta) \in R_L$. От тук получаваме, че всеки клас на еквивалентност на $R_A$ се съдържа в клас на еквивалентност на $R_L$ (вж. Фиг. 4.10) или $IX(R_L) \leq IX(R_A) = |Q_A|$. Следователно $IX(R_L)$ е краен.




Фигура 4.10: Класове на еквивалентност на $R_L$.

2) Нека $IX(R_L)$ е краен и $IX(R_L) = m$. Да означим с $[\alpha]$ класа на еквивалентност на $R_L$, съдържащ думата $\alpha \in X^*$. Да означим с $Q = \{[\varepsilon], [\alpha_1], \ldots, [\alpha_{m-1}]\}$ множеството от класовете на еквивалентност на $R_L$, където $\alpha_0 = \varepsilon, \alpha_1, \ldots, \alpha_{m-1}$ са представители на съответните класове. Образуваме КДА $A =< Q, X, [\varepsilon], \delta, F >$, където $F = \{[\alpha_{p_1}], [\alpha_{p_2}], \ldots, [\alpha_{p_r}]\}$ са всички класове на еквивалентност на $R_L$, съдържащи само думи от езика $L$. Ясно е от дефиницията на $R_L$, че всички думи от един и същ клас на $R_L$ едновременно са или не са в $L$, така че дефиницията на $F$ е коректна. Функцията на преходите $\delta$ дефинираме по следния начин: $\delta([\alpha], x) = [\alpha x]$. Тази дефиниция също е коректна, защото $\forall \beta \in [\alpha], [\beta x] = [\alpha x]$, заради дясната инвариантност на $R_L$.

Оставяме на читателя да докаже, с индукция по дължината на $\beta \in X^*$, че за разширената функция на преходите $\Delta$ на $A$ е в сила $\Delta([\alpha], \beta) = [\alpha\beta]$.

Ще докажем, че КДА $A$ разпознава точно езика $L$.



а) Нека $\omega \in L_A$, т.е. $\Delta([\varepsilon], \omega) \in F$. Но $\Delta([\varepsilon], \omega) = [\varepsilon\omega] = [\omega]$, т.е. съществува $p_j, [\omega] = [\alpha_{p_j}]$. Но тогава $\omega \in L$, защото всички думи на $[\alpha_{p_j}]$ са от $L$ според дефиницията на $[\alpha_{p_j}]$.

б) Нека $\omega \in L$. Тогава $[\omega] = [\alpha_{p_j}]$ за някое $p_j$ и $\Delta([\varepsilon], \omega) = [\omega] = [\alpha_{p_j}] \in F$. Следователно $\omega \in L_A$.

Следователно езикът $L$ е автоматен. □

На пръв поглед може и да не е ясна връзката между Теоремата на Нерод-Майхил и задачата за минимизация на КДА. Оказва се, че построеният в Теоремата автомат е минимален КДА, разпознаващ $L$ и даже нещо повече.

***Дефиниция.*** КДА $A =< Q, X, q_0, \delta, F >$ и $A' =< Q', X, q_0', \delta', F' >$ са *изоморфни*, ако съществува биекция $f : Q \to Q'$ такава, че $f(q_0) = q_0'$ и $\forall \alpha \in X$ е в сила $\Delta(q_0, \alpha) = q'$ т.с.т.к. $\Delta'(q_0'), \alpha) = f(q')$.

**Теорема 4.2.10** *Автоматът $A =< Q, X, q_0, \delta, F >$, построен в Теоремата на Нерод-Майхил е минимален за езика $L$ и всеки друг автомат за $L$ със същия брой състояния е изоморфен на $A$.*

***Доказателство.*** 1) Нека $A' =< Q', X, q_0', \delta', F' >$ е произволен автомат, разпознаващ $L$, навсякъде дефиниран и без недостижими състояния. Както в първата част на Теоремата на Нерод-Майхил доказваме, че $IX(R_L) \leq IX(R_{A'})$. Но $IX(R_L) = |Q|$, а $IX(R_{A'}) = |Q'|$. Следователно $A$ е минимален автомат за езика $L$.

2) Нека $A' =< Q', X, q_0', \delta', F' >$ е произволен автомат, разпознаващ $L$ и $|Q| = |Q'|$. Дефинираме биекцията $f : Q \to Q'$, $f([\alpha_j]) = q' \in Q'$, така че $\Delta'(q_0', \alpha_j) = q'$. Коректността на тази дефиниция следва от това, че всеки клас на еквивалентност на $R_{A'}$ е подмножество на един клас на еквивалентност на $R_L$ и след като $IX(R_{A'}) = IX(R_L)$, тогава всеки клас на еквивалентност на $R_{A'}$ съвпада с един клас на еквивалентност на $R_L$ и следователно кой елемент на $[\alpha_j]$ ще изберем за дефиниране на $f$ е без значение. Очевидно $f(q_0) = q_0'$. Функцията $f$ е изоморфизъм, защото $\forall \alpha \in X^*$ имаме $\Delta(q_0, \alpha) = [\alpha]$, а от дефиницията на $f$ следва, че $f([\alpha]) = \Delta'(q_0', \alpha)$, което е точно условието за изоморфизъм. □

За съжаление Теоремата на Нерод-Майхил не съдържа конструкция за построяване на минималния автомат. По-долу ще опишем процедура, която по зададен автомат $A$ на езика $L$ намира минималния $A_0$ за този език.

***Дефиниция.*** Състоянията $q_i$ и $q_j$ на автомата $A$ наричаме *еквивалентни*, ако съответните им класове на релацията $R_A$ са в един и същ клас на релацията $R_L$.

Всъщност, от Теоремата на Нерод-Майхил и горната дефиниция се вижда, че минимизацията на автомата $A$ се свежда до намирането на всички подмножества от еквивалентни състояния на $A$ (вж. Фиг. 4.10).

**Лема 4.2.6** *Ако състоянията $q'$ и $q''$ на КДА $A$ са такива, че $q' \notin F, q'' \in F$, то $q'$ и $q''$ не са еквивалентни.*

***Доказателство.*** Класът на $q'$ съдържа думи, които не са от езика $L_A$, класът на $q''$ – думи от езика на $L_A$. Всеки клас на $R_L$ се състои или само от думи на $L_A$ или само от думи, не принадлежащи на $L_A$. Следователно, класовете на $q'$ и $q''$ не са подмножества на един и същ клас на $R_L$, т.е. $q'$ и $q''$ не са еквивалентни. □

**Лема 4.2.7 (Тест на едната буква)** *Състоянията $q'$ и $q''$ на КДА $A$ не са еквивалентни, ако $\exists x \in X$ такова, че $\delta(q', x)$ и $\delta(q'', x)$ не са еквивалентни.*

***Доказателство.*** Да допуснем еквивалентността на $q'$ и $q''$ при условието на лемата. От дясната инвариантност на $R_L$ следва, че $\forall \gamma \in X^*$, $\Delta(q', \gamma)$ е еквивалентно на $\Delta(q'', \gamma)$ и следователно същото е вярно за $\gamma = x \in X$, което е противоречие. □

**Лема 4.2.8** *Нека $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_l\}$ е разбиване на множеството от състояния $Q$ на КДА $A$ такова, че $Q_i \subseteq F$ или $Q_i \subseteq Q \setminus F$, $i = 1, 2, \ldots, l$. Нека $\forall Q_j, \forall q', q'' \in Q_j, \forall x \in X$ е в сила $\delta(q', x) \in Q_k$ и $\delta(q'', x) \in Q_k$ за някое $k, 1 \leq k \leq l$. Тогава всяко $Q_j$ се състои само от еквивалентни състояния.*

***Доказателство.*** Нека $q' = \Delta(q_0, \alpha_1)$, $q'' = \Delta(q_0, \alpha_2)$ и $q', q'' \in Q_j$. Очевидно при условието на лемата $\forall \gamma \in X^*$ е в сила $\Delta(q', \gamma) = \Delta(\Delta(q_0, \alpha_1), \gamma) = \Delta(q_0, \alpha_1\gamma)$ и $\Delta(q'', \gamma) = \Delta(\Delta(q_0, \alpha_2), \gamma) = \Delta(q_0, \alpha_2\gamma)$ са в едно и също множество $Q_k$ на разбиването. Следователно $\alpha_1\gamma$ и $\alpha_2\gamma$ едновременно са или не са от $L$. Това означава че $(\alpha_1, \alpha_2) \in R_L$ и $q'$ и $q''$ са еквивалентни. □

Трите леми ни дават следната процедура:

**Алгоритъм за минимизация на КДА**

**Дадено:** КДА $A =< Q, X, q_0, \delta, F >$.

**Резултат:** КДА $A_0$ – минимален за езика $L_A$.

**Процедура:**

1. Образуваме разбиването $\mathcal{Q}^0 = \{Q_1^0, Q_2^0\}$ на $Q$, където $Q_1^0 = Q \setminus F, Q_2^0 = F$. Нека $i = 0$.

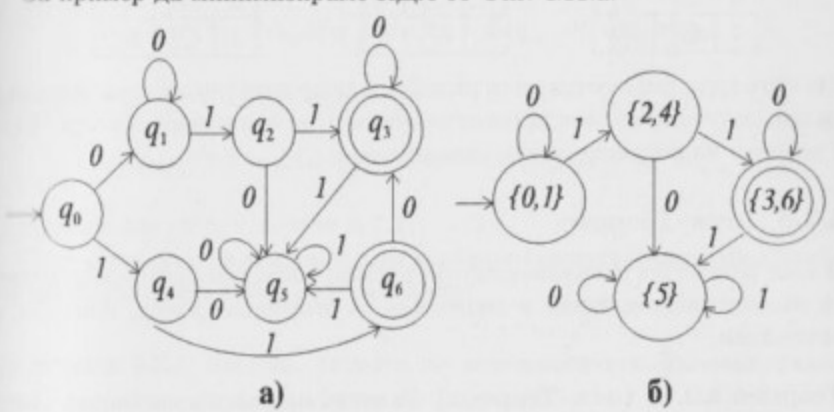


2. Нека сме построили разбиването $\mathcal{Q}^i = \{Q_1^i, Q_2^i, \ldots, Q_{l_i}^i\}$ на $Q$ за някое $i$. Всяко $Q_j^i$ разбиваме на подмножества $Q_{j_1}^{i+1}, Q_{j_1+1}^{i+1}, \ldots, Q_{j_1+r}^{i+1}$ такива, че елементите на всяко от тях не се проявяват като неекивалентни с теста на едната буква, за някое $x \in X$. Обединяваме получените разбивания и нека резултатът е разбиването $\mathcal{Q}^{i+1} = \{Q_1^{i+1}, Q_2^{i+1}, \ldots, Q_{l_{i+1}}^{i+1}\}$.

3. Ако $\mathcal{Q}^i$ и $\mathcal{Q}^{i+1}$ са едно и също разбиване – край. Иначе $i = i + 1$ и преминаваме към 2. □

Нека $\mathcal{Q}^r = \mathcal{Q}^{r+1} = \mathcal{Q}^{r+2} = \ldots$ е последното разбиване, получено от алгоритъма. От Теоремата на Нерод-Майхил и Леми 4.2.6, 4.2.7 и 4.2.8 следва, че всяко подмножество на $\mathcal{Q}^r$ се състои само от еквивалентни състояния. Освен това не е възможно еквивалентни състояния да са в различни подмножества на разбиването. Сега лесно построяваме минималния автомат $A_0 =< Q_0, X, t_0, \delta_0, F_0 >$, където $Q_0 = \mathcal{Q}^r$. Началното състояние $t_0$ е това $Q_i^r$, за което $q_0 \in Q_i^r$, защото началното състояние на минималния автомат е класът на $R_L$, съдържащ $\varepsilon$, а $\varepsilon$ е в класа на $q_0$. Множеството от заключителни състояния $F_0 = \{Q_k^r | Q_k^r \subseteq F\}$, защото заключителни състояния на минималния автомат са тези класове на $R_L$, които съдържат само думи от езика $L_A$ и следователно са образувани от класовете на $R_A$, породени от заключителни състояния. Функцията на преходите $\delta_0$ дефинираме по следния начин: $\delta_0(Q_i^r, x) = Q_j^r$, ако $\delta(q, x) \in Q_j^r$ за някое $q \in Q_i^r$. Тази дефиниция е коректна, заради дясната инвариантност на релацията $R_L$, т.е. защото буквата $x$ довежда автомата $A$ от произволно състояние от $Q_i^r$ в някое състояние на $Q_j^r$.

За пример да минимизираме КДА от Фиг. 4.11.а.




Фигура 4.11: КДА (а) и неговият минимален (б).

1) На първа стъпка разбиваме състоянията му на незаключителни $Q_1^0 = \{0,1,2,4,5\}$ и заключителни $Q_2^0 = \{3,6\}$.

2) Към всяко множество на разбиването прилагаме теста на едната буква:

| | 0 | 1 | 2 | 4 | 5 |
|---|---|---|---|---|---|
| 0 | $Q_1^0$ | $Q_1^0$ | $Q_1^0$ | $Q_1^0$ | $Q_1^0$ |
| 1 | $Q_1^0$ | $Q_1^0$ | $Q_2^0$ | $Q_2^0$ | $Q_1^0$ |

| | 3 | 6 |
|---|---|---|
| 0 | $Q_2^0$ | $Q_2^0$ |
| 1 | $Q_1^0$ | $Q_1^0$ |

Сега множеството $Q_1^0$ се разбива на две подмножества и получаваме $Q_1^1 = \{0,1,5\}, Q_2^1 = \{2,4\}, Q_3^1 = \{3,6\}$.

3) Тъй като поне едно подмножество се разби, продължаваме с теста на едната буква:

| | 0 | 1 | 5 |
|---|---|---|---|
| 0 | $Q_1^1$ | $Q_1^1$ | $Q_1^1$ |
| 1 | $Q_2^1$ | $Q_2^1$ | $Q_1^1$ |

| | 2 | 4 |
|---|---|---|
| 0 | $Q_1^1$ | $Q_1^1$ |
| 1 | $Q_3^1$ | $Q_3^1$ |

| | 3 | 6 |
|---|---|---|
| 0 | $Q_3^1$ | $Q_3^1$ |
| 1 | $Q_1^1$ | $Q_1^1$ |

В резултат получаваме разбиването $Q_1^2 = \{0,1\}$, $Q_2^2 = \{2,4\}$,$Q_3^2 = \{3,6\}$, $Q_4^2 = \{5\}$. Последното множество е от един елемент и не може да бъде повече разбивано.

4) Сега тестът на едната буква ни дава:

| | 0 | 1 |
|---|---|---|
| 0 | $Q_1^2$ | $Q_1^2$ |
| 1 | $Q_2^2$ | $Q_2^2$ |

| | 2 | 4 |
|---|---|---|
| 0 | $Q_4^2$ | $Q_4^2$ |
| 1 | $Q_3^2$ | $Q_3^2$ |

| | 3 | 6 |
|---|---|---|
| 0 | $Q_3^2$ | $Q_3^2$ |
| 1 | $Q_4^2$ | $Q_4^2$ |

т.е. нито едно множество не се разпада и следователно на предната стъпка сме получили множествата от еквивалентни състояния. На Фиг. 4.11.6 е показан полученият минимален автомат.

### 4.2.5 uvw-Теорема

В този раздел ще докажем едно твърдение, което показва ограничеността на автоматните езици и позволява да откриваме езици, които не са автоматни.

**Теорема 4.2.11 (uvw-Теорема)** *За всеки непразен автоматен език $L$ съществува цяло положително число $n$ такова, че ако $\alpha \in L$ и $d(\alpha) > n$, то $\alpha = uvw$ и 1) $d(v) \geq 1$; 2) $d(uv) \leq n$; 3) $\forall i = 0,1,2,\ldots$ думата $uv^iw \in L$.*



**Доказателство**. За автоматния език $L$ съществува КДА $A = < Q, X, q_0, \delta, F >$, който го разпознава. Избираме $n = |Q|$. Ще покажем, че това е търсеното цяло число. Нека $\alpha$ е дума от езика $L$, $\alpha = x_{i_1}x_{i_2}\ldots x_{i_k}$, $d(\alpha) = k > n$. Тъй като $A$ разпознава $\alpha$, то $\Delta(q_0,\alpha) \in F$, т.е.

$$\delta(q_0,x_{i_1}) = q_{i_1}, \delta(q_{i_1},x_{i_2}) = q_{i_2},\ldots,\delta(q_{i_{k-1}},x_{i_k}) = q_{i_k} \in F.$$

Ако разгледаме редицата от състояния $q_0, q_{i_1},\ldots,q_{i_k}, k > n$, през които автоматът преминава при работа върху думата $\alpha$, то от Принципа на Дирихле получаваме, че в редицата има поне една двойка съвпадащи състояния. Да изберем най-ляво разположената двойка $q_{i_m} = q_{i_l}, m < l$, т.е. вляво от $q_{i_l}$ няма друга такава двойка. Разбиваме $\alpha$ на три поддуми $u, v$ и $w$, такива че $\Delta(q_0,u) = q_{i_m}, \Delta(q_{i_m},v) = q_{i_l}, \Delta(q_{i_l},w) = q_{i_k} \in F$. Ще покажем, че тези три думи удовлетворяват твърденията на теоремата.

1) Тъй като $m \neq l$, $d(v) \geq 1$.

2) Тъй като $q_{i_m}, q_{i_l}$ е най-ляво разположената двойка, $d(uv) \leq n$. В противен случай отново щяхме да можем да приложим Принципа на Дирихле и да намерим друга двойка съвпадащи състояния в редицата $q_0, q_{i_1},\ldots,q_{i_{l-1}}$.

3) С индукция по $i$ доказваме, че $\Delta(q_0,uv^i) = q_{i_m} = q_{i_l}$. Действително

$$\Delta(q_0,uv^0) = \Delta(q_0,u) = q_{i_m} = q_{i_l}.$$

Допускаме, че твърдението е вярно за $i$ и да разгледаме

$$\Delta(q_0,uv^{i+1}) = \Delta(\Delta(q_0,uv^i),v) = \Delta(q_{i_m},v) = q_{i_l} = q_{i_m}.$$

Сега за всяко $i = 0,1,2,\ldots$ имаме

$$\Delta(q_0,uv^iw) = \Delta(\Delta(q_0,uv^i),w) = \Delta(q_{i_l},w) = q_{i_k} \in F$$

и следователно $uv^iw \in L, \forall i = 0,1,2,\ldots$. □

С помощта на uvw-теоремата ще докажем съществуването на контекстно-свободен език, който не е автоматен.

**Следствие 4.2.1** *Езикът, породен от контекстно-свободната граматика*

$$\Gamma =< \{S\}, \{a,b\}, S, \{S \to ab, S \to aSb\} >$$

*не е автоматен.*

***Доказателство.*** Както вече знаем, езикът на Γ е $L_\Gamma = \{a^i b^i | i = 1, 2, \ldots\}$. Да допуснем, че този език е автоматен, и нека $n$ е съответното му цяло число, определено от uvw-теоремата. Нека $i > n/2$, тогава $\alpha = a^i b^i$ е такава, че $\alpha \in L_\Gamma$ и $d(\alpha) > n$. Следователно $\alpha = uvw, d(v) \geq 1$ и $uv^2w \in L$. Да допуснем, че $v = a^r, r \geq 1$ (или $v = b^r, r \geq 1$). Тогава $uv^2w = a^{i+r}b^i$ (или $uv^2w = a^i b^{i+r}$), което е в противоречие с общия вид на думите от езика. Нека тогава допуснем, че $v = a^r b^s, i > r > 1, i > s > 1$. Тогава $uv^2w = a^{i-r}a^r b^s a^r b^s b^{i-s} = a^i b^s a^r b^i$, която също не е от езика. Тъй като друга възможност за думата $v$ не съществува, допускането ни е неверно и $L_\Gamma$ не е автоматен. □

И така автоматните езици са същинско подмножество на контекстно-свободните.

**Следствие 4.2.2** *Автоматният език $L$, разпознаван от КДА $A =< Q, X, q_0, \delta, F >$ е непразен ($L \neq \emptyset$) т.с.т.к. съдържа дума $\alpha, d(\alpha) \leq |Q|$.*

***Доказателство.*** 1) Нека $L$ съдържа дума $\alpha, d(\alpha) \leq |Q|$. Тогава, очевидно $L \neq \emptyset$.

2) Нека $L \neq \emptyset$. Да допуснем, че $\forall \alpha \in L$ е в сила $d(\alpha) > |Q| = n$. Тогава uvw-теоремата е в сила за всяка дума от езика и в частност за най-късата $\alpha \in L$, т.е. $\alpha = uvw, d(v) \geq 1$ и $uv^0w = uw \in L$. Това е противоречие с допускането, че $\alpha$ е най-късата дума на $L$. Следователно в $L$ има и думи с дължина $\leq n$. □

Сега ще покажем как с КДА може да бъде решена следната задача: „Автоматният език $L_A$ е зададен с КДА $A$, който го разпознава. Дадена е и дума $\alpha$ над входната азбука на автомата. Принадлежи ли $\alpha$ на $L_A$?"

Да оставим автомата $A$, намиращ се в начално състояние да работи върху думата $\alpha$ (Без ограничение можем да считаме, че $A$ е навсякъде дефиниран.) Когато $A$ завърши работата си над $\alpha$, достатъчно е да проверим дали състоянието, в което $A$ се намира е заключително или не. Ако е заключително, $\alpha$ е от езика на $A$, в противен случай $\alpha$ не е от езика на $A$.

Ето още една задача, решението на която може да се получи с помощта на КДА: „Автоматният език $L_A$ е зададен с $A =< Q, X, q_0, \delta, F >$, който го разпознава. Да се провери дали $L_A = \emptyset$."

Задачата решаваме, като построим всички думи $\alpha$ с дължина $d(\alpha) \leq n = |Q|$ и оставим автомата последователно да работи върху всяка такава дума. Ако намерим дума, която се разпознава от $A$, очевидно, $L_A$ не е празен език. В противен случай езикът е празен в съответствие със Следствие 4.2.2.



От своя страна задачата: „Даден е контекстно-свободен език $L$ и дума $\alpha$ над терминалната му азбука. Да се определи дали $\alpha \in L$." не може да бъде решена с помощта на КДА в общия случай. Действително, всеки език, за който горната задача се решава от КДА ще бъде автоматен език, а съгласно Следствие 4.2.1 съществуват контекстно-свободни езици, които не са автоматни.

## 4.3 Контекстно-свободни езици

### 4.3.1 Дърво на извод на КСГ

Всеки контекстно-свободен език се поражда от контекстно-свободна граматика $\Gamma =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} >$, правилата на която са от вида $A \to \alpha, A \in \mathcal{N}, \alpha \in (\mathcal{N} \cup \mathcal{T})^+$. Ако в граматиката няма аксиома в дясна част на правило, допустимо е и правилото $S \to \varepsilon$, позволяващо ни да извеждаме само празната дума. Както вече видяхме в 4.1., правилата от вида $A \to B, A, B \in \mathcal{N}$, наричани преименуващи, не са задължителни за контекстно-свободните граматики, въпреки че се допускат от общия вид на правилата. Затова можем да считаме, че граматиката, с която работим, е без преименуващи правила.

На всеки извод в контекстно-свободна граматика ще съпоставим дървовидна структура със следната

***Дефиниция.*** Нека $\Gamma =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} >$ е контекстно-свободна граматика. Дървото $D(V, E)$ с корен $r \in V$, с наредба на синовете и функция $f : V \to (\mathcal{N} \cup \mathcal{T})$, съпоставяща на всеки връх буква от азбуките на граматиката, наричаме *дърво на извод* в Γ, ако:

а) за всеки нелист $v$ на $D, f(v) \in \mathcal{N}$, като $f(r) = S$;

б) за всеки лист $l$ на $D, f(l) \in \mathcal{T}$;

в) за всеки нелист $v$ с наредена последователност от синове $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$, е в сила $f(v) \to f(v_{i_1})f(v_{i_2}) \ldots f(v_{i_k}) \in \mathcal{P}$.

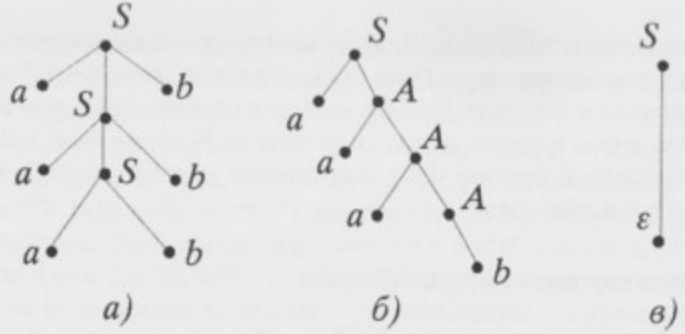
На Фиг. 4.12 ((а) и (б)) са показани две дървета на извод съответно от граматиките

$$\Gamma_1 =< \{S\}, \{a, b\}, S, \{S \to ab, S \to aSb\} >$$

и

$$\Gamma_2 =< \{S, A\}, \{a, b\}, S, \{S \to aA, A \to aA, A \to b\} > .$$

Специалният вид на дърветата, получавани от автоматни граматики се вижда от Фиг. 4.12.б. За да не изключваме случая, когато $S \to \varepsilon \in \mathcal{P}$ допускаме в множеството от дървета на извод и особеното дърво от Фиг. 4.12.в.




Фигура 4.12: Дървета на извод.

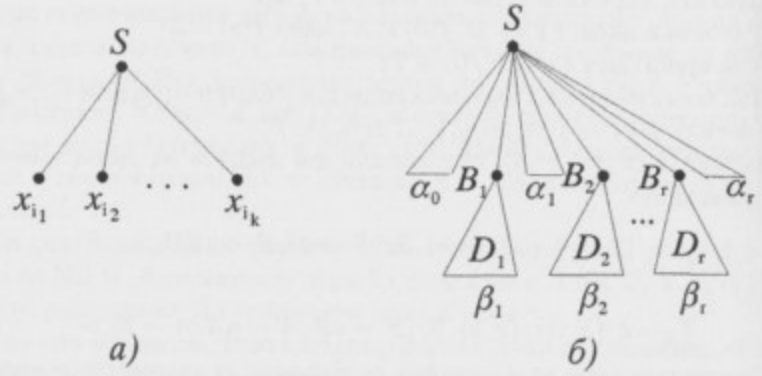
*Дума на дърво на извод* дефинираме индуктивно, следвайки една от дефинициите на понятието дърво.

***Дефиниция.*** Нека $D(V,E)$ е дърво на извод в граматиката $\Gamma = <\mathcal{N}, \mathcal{T}, S, \mathcal{P}>$.

а) ако $D$ е дървото от 4.12.в, дума на $D$ е $\varepsilon$;

б) Нека $D'(\{l\},\{\})$ е поддърво на дървото на извод $D$, състоящо се само от лист $l$. Дума на $D'$ е $f(l)$;

в) Нека $D'$ е поддърво на $D$ с корен $r'$, $D_1, D_2, \ldots, D_k$ са всички поддърветa на $D'$ с корени в синовете на $r'$, подредени в съответствие с наредбата на тези синове и $\alpha_i$ е думата на поддървото $D_i, i = 1,2,\ldots,k$. Тогава думата на $D'$ е $\alpha_1\alpha_2\ldots\alpha_k$.




Фигура 4.13: Дума на дърво на извод.



Ако $\Gamma =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} >$, то $\forall A \in \mathcal{N}$ образуваме граматиката $\Gamma_A = < \mathcal{N}, \mathcal{T}, A, \mathcal{P} >$. Очевидно $A \overset{\Gamma_A}{\models} \omega$ т.с.т.к. $A \overset{\Gamma}{\models} \omega$. Това се налага от необходимостта да разглеждаме поддърветата на едно дърво на извод като дървета на извод, независимо от факта, че коренът им не е означен с аксиомата на $\Gamma$. Сега всяко такова поддърво с корен, означен с $A$ е дърво на извод в $\Gamma_A$.

**Теорема 4.3.1** *Нека $\Gamma =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} >$ е КСГ. Съществува дърво на извод $D$ в $\Gamma$ с дума $\omega$ т.с.т.к. $S \overset{\Gamma}{\models} \omega$.*

***Доказателство.*** 1) За произволно $A \in \mathcal{N}$ с индукция по броя на нетерминалите в дървото на извод $D$ в $\Gamma_A$ с дума $\omega$ ще покажем, че $A \overset{\Gamma_A}{\models} \omega$.

а) Нека $D$ има един връх, означен с нетерминал. Тогава $D$ е или дървото от Фиг. 4.12.в или е дървото $D(\{u, v_1, \ldots, v_k\}, \{(u, v_i), i = 1,2,\ldots,k\})$ (подобно но това от Фиг. 4.13.а, като $S$ се замени с $A$). В първия случай $\omega = \varepsilon$, но тогава $S \to \varepsilon \in \mathcal{P}$ и следователно $S \overset{\Gamma}{\models} \varepsilon$. Във втория случай $\omega = x_{i_1}x_{i_2}\ldots x_{i_k}$, но съгласно дефиницията $A \to x_{i_1}x_{i_2}\ldots x_{i_k} \in \mathcal{P}$ и следователно $A \overset{\Gamma}{\models} \omega$.

б) Нека $B_i \in \mathcal{N}$ е корен на дърво на извод в $\Gamma_{B_i}$ с дума $\beta_i$ с не повече от $n \geq 1$ нетерминала. Да допуснем, че $B_i \overset{\Gamma_{B_i}}{\models} \beta_i$.

в) Нека $A \in \mathcal{N}$ е корен на дърво на извод с $n+1$ нетерминала в $\Gamma_A$ и с дума $\omega$. В общия случай то е подобно на дървото от Фиг. 4.13.б, като $S$ се замени с $A$. С корена му е свързано правилото $A \to \alpha_0 B_1 \alpha_1 B_2 \ldots B_r \alpha_r$, в което $\alpha_i \in \mathcal{T}^*, i \in \{0,1,\ldots,r\}$, а $B_j \in \mathcal{N}$ са корени на поддърветата $D_1, D_2, \ldots, D_r$ с думи $\beta_1, \beta_2, \ldots, \beta_r$ съответно и значи за думата му е в сила $\omega = \alpha_0\beta_1\alpha_1\beta_2\ldots\beta_r\alpha_r$. За всяко от поддърветата прилагаме индукционното предположение, тъй като броят на нетерминалите по върховете им е $\leq n$. Получаваме изводите $B_i \overset{\Gamma_{B_i}}{\models} \beta_i, i \in \{1,2,\ldots,r\}$. Тези изводи са идентични с $B_i \overset{\Gamma_A}{\models} \beta_i, i \in \{1,2,\ldots,r\}$ и можем да построим следния извод:

$$\begin{aligned} A &\overset{\Gamma_A}{\vdash} \alpha_0 B_1 \alpha_1 B_2 \ldots B_r \alpha_r \overset{\Gamma_A}{\models} \\ &\overset{\Gamma_A}{\models} \alpha_0\beta_1\alpha_1 B_2 \ldots B_r\alpha_r \overset{\Gamma_A}{\models} \\ &\overset{\Gamma_A}{\models} \alpha_0\beta_1\alpha_1\beta_2\ldots B_r\alpha_r \overset{\Gamma_A}{\models} \end{aligned}$$

$$\ldots$$
$$\overset{\Gamma_A}{\models} \alpha_0\beta_1\alpha_1\beta_2\ldots\beta_r\alpha_r = \omega$$

т.е. $A \overset{\Gamma_A}{\models} \omega$.

В частния случай $A = S$ получаваме исканото твърдение: ако съществува дърво на извод $D$ в $\Gamma$ с дума $\omega$, то $S \overset{\Gamma}{\models} \omega$.

2) Нека $A \overset{\Gamma_A}{\models} \omega \in \mathcal{T}^*$ за някой нетерминал $A \in \mathcal{N}$. С индукция по дължината на извода ще докажем, че съществува дърво на извод в $\Gamma_A$ с дума $\omega$.

а) Нека дължината на извода е 1. Тогава той е или $S \overset{\Gamma_S}{\vdash} \varepsilon$ и за него съществува дървото от Фиг. 4.12.в с дума $\varepsilon$ или е $A \overset{\Gamma_A}{\vdash} \omega = x_{i_1}x_{i_2}\ldots x_{i_k}$ и дървото му е това от Фиг. 4.13.а.

б) Допускаме, че за всеки $B_i \in \mathcal{N}$ от $B_i \overset{\Gamma_{B_i}}{\models} \beta i$, с дължина на извода $\leq n$, следва съществуването на дърво на извод в $\Gamma_{B_i}$ с дума $\beta_i$.

в) Сега нека $A \overset{\Gamma_A}{\models} \omega$ и този извод с дължина $n+1$ е

$$A \overset{\Gamma_A}{\vdash} \alpha_0 B_1\alpha_1 B_2\ldots B_r\alpha_r \overset{\Gamma_A}{\models} \alpha_0\beta_1\alpha_1\beta_2\ldots\beta_r\alpha_r = \omega.$$

От тук се вижда, че съществуват изводи $B_i \overset{\Gamma_A}{\models} \beta i, i = 1, 2, \ldots, r$, всеки от които е с дължина $\leq n$ (защото преди него е направена поне една стъпка). Обръщаме тези изводи в $B_i \overset{\Gamma_{B_i}}{\models} \beta i, i = 1, 2, \ldots, r$. Съгласно индукционното предположение съществуват дървета на извод $D_1, D_2, \ldots, D_r$ в граматиките $\Gamma_{B_i}$ с думи $\beta_i, i = 1, 2, \ldots, r$. Сега дървото от Фиг. 4.13.б е търсеното дърво за $\omega$ в $\Gamma_A$.

В частния случай $A = S$ получаваме исканото твърдение: ако $S \overset{\Gamma}{\models} \omega$, то съществува дърво на извод в $\Gamma$ с дума $\omega$. □

Дървото на извод не определя еднозначно извода. Затова ще уточним понятието извод така:

***Дефиниция.*** Изводът $S \vdash \alpha_1 \vdash \alpha_2 \vdash \cdots \vdash \alpha_k$ наричаме *ляв*, ако $\forall i = 1, 2, \ldots, k-1$ е в сила $\alpha_i = \alpha' A\alpha'', \alpha_{i+1} = \alpha'\beta\alpha''$, като $\alpha' \in \mathcal{T}^*, A \in \mathcal{N}$, т.е. на всяка стъпка е заместен най-левият нетерминал. Аналогично дефинираме понятието *десен извод*. Забележете, че в частния случай на автоматни граматики всеки извод е едновременно и ляв и десен.

**Теорема 4.3.2** *Нека $\Gamma =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} >$ е КСГ, а $D$ – дърво на извод в $\Gamma$ с дума $\omega$. Тогава съществува единствен ляв (десен) извод $S \models \omega$.*



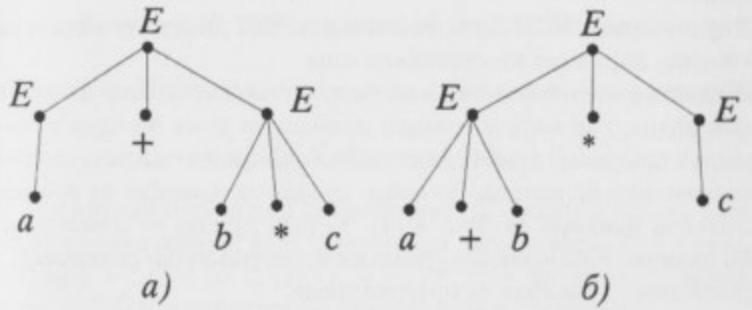
***Доказателство.*** За доказателство на тази теорема ще модифицираме процедурата за обхождане в дълбочина на кореново дърво с наредба на синовете по следния начин: Заменяме стъпката „избираме необходим син на текущия връх“ с „избираме най-левия (най-десния) необходим син на текущия връх“. Така модифицираната процедура наричаме *ляво (дясно) обхождане в дълбочина*.

Построяването на ляв извод става по следния начин: За наредба на синовете на връх използваме реда, по който се срещат (в дясната част на използваното в дървото правило) приписаните им букви. Текуща дума в началото е $S$. Започваме ляво обхождане на дървото на извод $D$. Когато за първи път „обхождаме“ връх, обозначен с нетерминал – прилагаме правилото, определено от дървото на извод за този нетерминал и получаваме нова текуща дума. От това, че измежду всички нетерминали, с които са обозначени синовете на даден връх винаги избираме най-левия, за да продължим обхождането и поради това, че обхождането е в дълбочина, полученият единствен извод е ляв.

Аналогично доказваме, че дясното обхождане в дълбочина поражда единствен десен извод. □

Може да се окаже, че за една и съща дума има две различни дървета на извод в КСГ Г. На Фиг. 4.14. са показани две такива дървета за думата $a + b * c$ в

$$\Gamma = < \{E\}, \{a, b, c, +, *\}, E,$$
$$\{E \to E * E, E \to E + E, E \to a, E \to b, E \to c\} > .$$



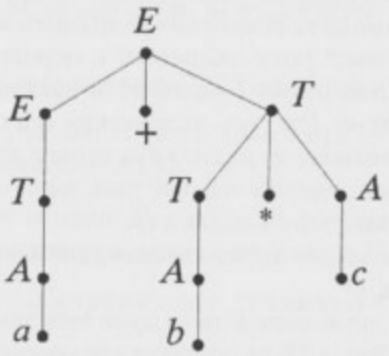

Фигура 4.14: Неоднозначна граматика.

***Дефиниция.*** Ако в КСГ Г съществува повече от едно дърво на извод за някоя дума $\omega$, тогава наричаме Г *неоднозначна*.

Това, че една граматика е неоднозначна не е фатално за езика, който тя поражда, защото може да се окаже, че съществува друга граматика за същия език, която е еднозначна. За примера от Фиг. 4.14 да разгледаме граматиката $\Gamma'$, пораждаща същия език:

$$\Gamma' = < \{E, T, A\}, \{a, b, c, +, *\}, E, \{E \to E + T, E \to T, T \to T * A, T \to A, A \to a, A \to b, A \to c\} > .$$

Думата $a+b*c$ има единствено дърво на извод в $\Gamma'$, показано на Фиг. 4.15.




Фигура 4.15: Дърво на извод в непротиворечива граматика.

Може да се окаже, обаче, че всяка КСГ пораждаща езика $L$ е неоднозначна.

***Дефиниция.*** КСЕ $L$, за който всяка КСГ, която го поражда е неоднозначна, наричаме *неоднозначен език*.

Неоднозначните езици не се използуват за реализация на системи за програмиране, тъй като във всяка разпозната дума на един такъв език (правилна програма) трябва да се вложи определен смисъл (*семантика*), силно зависеща от начина, по който думата се извежда от граматиката (вж. отново примера на Фиг. 4.14). Когато думата се извежда по различни начини, това означава възможност за различна семантика, а това е недопустимо в езиците за програмиране.

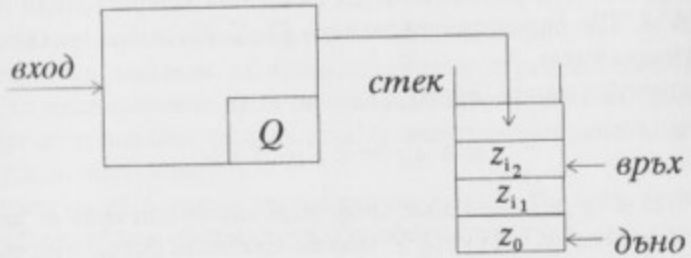
### 4.3.2 Недетерминирани стекови автомати

Както вече доказахме, КДА не могат да разпознават КСЕ. Затова ще дефинираме по-сложни машини, разпознаващи тези езици. Важно за дефиницията на тези машини е понятието *стек* (вж. Фиг. 4.16). Стекът е



запомнящо устройство, в което могат да се записват буквите на някаква азбука $Z$, и да се вземат обратно от стека, когато е необходимо. Дисциплината на работа със стека описваме неформално с фразата „влезлият пръв излиза последен". Това означава, че всеки новозапомнен елемент се поставя над елемента, обозначен като връх на стека и сам става връх. Когато трябва да се вземе елемент от стека, това е непременно върхът и връх става елементът, който е влязъл в стека непосредствено преди него. Освен тези две основни операции (поставяне в стека и вземане от стека) понякога се използва и операцията четене от стека, което означава, че се прочита буквата на върха, без да се изважда от стека.

Абстрактната машина, за която ще става дума в този подраздел (вж. Фиг. 4.16) разполага с обичаен вход, от който може да чете буквите на входна азбука $X$ и стек, от който може да чете и в който може да пише. Машината може да се намира в крайно множество от състояния.




Фигура 4.16: Недетерминиран стеков автомат.

***Дефиниция.*** *Недетерминиран стеков автомат* наричаме седморката

$$A = < Q, X, Z, q_0, z_0, \delta, F >,$$

в която: $Q$ е крайно множество от състояния; $X$ е крайна входна азбука; $Z$ е крайна стекова азбука; $q_0 \in Q$ е начално състояние; $z_0 \in Z$ е начална стекова буква; $\delta : Q \times (X \cup \{\varepsilon\}) \times Z \to 2^{Q \times Z^*}$ е частична функция на преходите; $F \subseteq Q$ е множество от заключителни състояния.

Ще разгледаме тези елементи на $A$, които го отличават от крайните автомати.

Специалната буква $z_0$ в стековата азбука е предназначена да пази стека от изпразване, тъй като НСА не може да работи, без да чете букви от стека. Противно на КДА четенето на входна буква от $X$ не е

задължително за НСА (забележете $X \cup \{\varepsilon\}$ в дефиниционната област на $\delta$). С други думи НСА може да работи без да чете входни данни, обработвайки запомнената в стека дума.

Втората особеност е по-сложната функция на преходите $\delta$, дефинирана върху тройки *(състояние, входна буква или празна дума, стекова буква)* и даваща в резултат множество от двойки *(състояние, стекова дума)*. В резултат, на всяка стъпка НСА може да премине недетерминирано в някое от определените от $\delta$ състояния, заместваики буквата във върха на стека със съответната, зададена от $\delta$, дума.

За формално дефиниране на работата на НСА, да определим неговото текущо положение с тройката $< q, \alpha, \gamma >$, наречена *конфигурация* на НСА, в която $q$ е текущото състояние на автомата, $\alpha$ – непрочетената още част от входната дума, а $\gamma$ – текущата стекова дума, четена от върха към дъното на стека. Очевидно, конфигурацията точно определя текущото положение на НСА и бъдещото му поведение.

Да означим с $\mathcal{K}_A$ множеството от възможни конфигурации на зададен НСА $A$. Ще определим релацията $R_{\vdash} \subseteq \mathcal{K}_A \times \mathcal{K}_A$ между двойки конфигурации така:

1) $\varepsilon$-трансформация ($\varepsilon$-преход):

$$< q, \alpha, z\gamma > \vdash < q', \alpha, \gamma'\gamma >,$$

ако $\delta(q, \varepsilon, z) \ni (q', \gamma')$. При тази трансформация остатъкът от входната дума остава същият, а думата $\gamma'$ заменя стековата буква $z$ на върха на стека;

2) нормален преход:

$$< q, x\alpha, z\gamma > \vdash < q', \alpha, \gamma'\gamma >,$$

ако $\delta(q, x, z) \ni (q', \gamma')$. При нормалния преход се чете входна буква, тя се изтрива от остатъка на входната дума и повече не може да се използва.

Ако $\kappa_1, \kappa_2 \in \mathcal{K}_A$ и $\kappa_1 \vdash \kappa_2$ казваме, че „конфигурацията $\kappa_1$ може да се трансформира (да премине) непосредствено в $\kappa_2$“.

Не е трудно да се види, че използваната операция „заместване на буквата от върха на стека със стекова дума“ е универсална. Заместването на $z_i$ с думата $z_j z_i$ е равносилно на добавяне на буквата $z_j$ в стека, заместването на $z_i$ с $\varepsilon$ – на изтриване на буквата $z_i$ от стека, а заместването на $z_i$ със $z_i$ – на прочитане на върха на стека.

***Дефиниция.*** Релацията $R_{\models} \subseteq \mathcal{K}_A \times \mathcal{K}_A$ дефинираме като рефлексивно и транзитивно затваряне на $R_{\vdash}$. Ако $\kappa_1, \kappa_2 \in \mathcal{K}_A$ и $\kappa_1 \models \kappa_2$ казваме, че „конфигурацията $\kappa_1$ може да се трансформира в $\kappa_2$“.



***Дефиниция.*** Нека $\alpha \in X^*$ е думата, подавана на входа на НСА $A =< Q, X, Z, q_0, z_0, \delta, F >$. Конфигурация от вида $\kappa_0 =< q_0, \alpha, z_0 >$ наричаме *начална конфигурация* за думата $\alpha$.

***Дефиниция.*** НСА $A =< Q, X, Z, q_0, z_0, \delta, F >$ *разпознава със заключително състояние* думата $\alpha$, ако съществува $q \in F$ такова, че $< q_0, \alpha, z_0 > \models < q, \varepsilon, \gamma >$, независимо от това каква е $\gamma$. Езикът

$$L_A^F = \{\alpha | \exists q \in F, < q_0, \alpha, z_0 > \models < q, \varepsilon, \gamma >\},$$

наричаме *език, разпознаван със заключителни състояния* от НСА $A$.

***Дефиниция.*** Казваме че НСА $A =< Q, X, Z, q_0, z_0, \delta, \emptyset >$ *разпознава с празен стек* думата $\alpha$, ако $< q_0, \alpha, z_0 > \models < q, \varepsilon, \varepsilon >$, независимо от това какво е състоянието $q$. Езикът

$$L_A^{\emptyset} = \{\alpha | < q_0, \alpha, z_0 > \models < q, \varepsilon, \varepsilon >\},$$

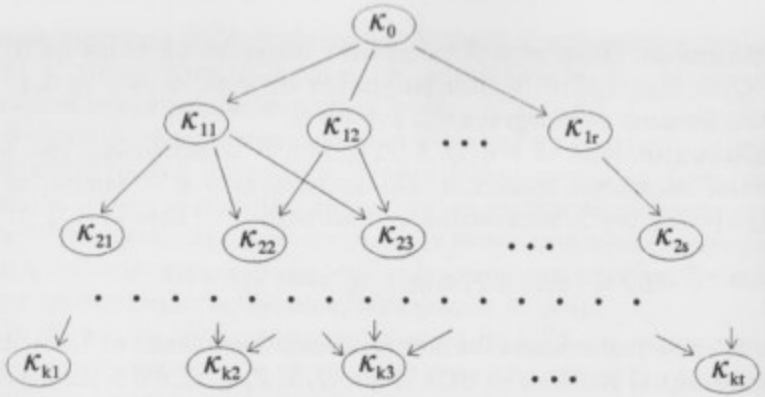
наричаме *език, разпознаван с празен стек* от НСА $A$.

По-късно ще покажем, че двете дефиниции за разпознаване на език от НСА са еквивалентни. Това не означава, обаче, че може да ги смесваме. При разпознаване на един език се използва или само едната, или само другата дефиниция.

Работата на НСА върху дадена входна дума $x_{i_1} x_{i_2} \ldots x_{i_k}$ може да се опише с (безкрайна в общия случай) графова структура, така както направихме с КНА. Нека при пресмятане на $\delta$ се получават повече от една двойка $(q, \gamma)$. Тогава НСА от положението, в което се намира, се размножава в толкова копия, колкото са различните значения на $\delta$ и всяко копие се трансформира съответно на едно от значенията (вж. Фиг. 4.17). Дефинициите за разпознаване означават, че поне една от конфигурациите от листата на получения ориентиран граф без цикли е разпознаваща (със заключителни състояния или с празен стек).

И при НСА можем да считаме непълнотите в дефиницията на $\delta$ като условие за неразпознаване. При НСА, обаче, поради наличието на стек и възможност машината да работи и без четене на входни букви, се появява нова принципна възможност за неразпознаване – възможността НСА никога да не завърши работа. В такъв случай казваме, че автоматът е зациклил.

За пример да разгледаме НСА $A_1 =< \{q_0, q_1, q_2\}, \{a, b\}, \{z_0, z_1\}, q_0, z_0, \delta, \emptyset >$, за който $\delta(q_0, a, z_0) = \{(q_1, z_1 z_0)\}$, $\delta(q_1, a, z_1) = \{(q_1, z_1 z_1)\}$, $\delta(q_1, b, z_1) = \{(q_2, \varepsilon)\}$, $\delta(q_2, b, z_1) = \{(q_2, \varepsilon)\}$, $\delta(q_2, \varepsilon, z_0) = \{(q_2, \varepsilon)\}$. Този НСА няма заключителни състояния и разпознава с празен стек. При работа




Фигура 4.17: Ориентиран граф без цикли на НСА.

върху думата *aaabbb* НСА $A_1$ последователно преминава през конфигурациите

$$\begin{aligned} < q_0, aaabbb, z_0 > &\vdash < q_1, aabbb, z_1 z_0 > \vdash < q_1, abbb, z_1 z_1 z_0 > \vdash \\ &\vdash < q_1, bbb, z_1 z_1 z_1 z_0 > \vdash < q_2, bb, z_1 z_1 z_0 > \vdash \\ &\vdash < q_2, b, z_1 z_0 > \vdash < q_2, \varepsilon, z_0 > \vdash < q_2, \varepsilon, \varepsilon >, \end{aligned}$$

разпознавайки зададената дума. При работа върху думата *aabab* НСА последователно преминава през конфигурациите

$$< q_0, aabab, z_0 > \vdash < q_1, abab, z_1 z_0 > \vdash < q_1, bab, z_1 z_1 z_0 > \vdash < q_2, ab, z_1 z_0 >$$

и не разпознава зададената дума. Линейният вид на тези преходи на конфигурации се дължи на простия факт, че НСА $A_1$ е детерминиран, т.е. множествата от стойности на функцията на преходите му винаги са с не повече от един елемент. Очевидно е че проверката за разпознаване с детерминиран стеков автомат е много по-проста.

Оставяме на читателя да покаже, че $L^{\emptyset}_{A_1} = \{a^i b^i | i = 1, 2, \ldots\}$.

Всъщност двата вида разпознаване с НСА са еквивалентни.

**Теорема 4.3.3** *За всеки НСА $A$, разпознаващ езика $L$ със заключителни състояния, съществува НСА $A'$, разпознаващ $L$ с празен стек.*

***Доказателство.*** Нека $A =< Q, X, Z, q_0, z_0, \delta, F >$. Построяваме $A' =< Q \cup \{t_0, t_1\}, X, Z \cup \{s_0\}, t_0, s_0, \delta', \emptyset >$, където $t_0, t_1 \notin Q, s_0 \notin Z$ и функцията на преходите $\delta'$ е дефинирана както следва:



1) $\delta'(t_0, \varepsilon, s_0) = \{(q_0, z_0 s_0)\}$;

2) $\delta'(q, \varepsilon, z) = \delta(q, \varepsilon, z)$ и $\delta'(q, x, z) = \delta(q, x, z)$ навсякъде, където $\delta$ е дефинирана;

3) $\delta'(q, \varepsilon, z)$ и $\delta'(q, x, z)$ са недефинирани, ако $\delta(q, \varepsilon, z)$ и $\delta(q, x, z)$ са недефинирани;

4) $\forall q \in F, \delta'(q, \varepsilon, z) \ni (t_1, \varepsilon)$;

5) $\delta'(t_1, \varepsilon, z) = \{(t_1, \varepsilon)\}$.

Работата на НСА $A'$ започва винаги с поставяне на върха на стека на началната стекова буква на $A$ – $z_0$ и преминаване в неговото начално състояние – $q_0(1)$. Тъй като $\delta'$ точно повтаря дефиницията на $\delta$ ((2) и (3)), автоматът $A'$ ще работи точно така, както автоматът $A$ върху произволна входна дума $\omega$: след прочитането ѝ ще попадне в състояние от $F$, ако $A$ попада в такова състояние, ще попадне в състояние не от $F$, ако $A$ направи същото, ще спре по недефинираност или ще зацикли, ако това е поведението на $A$. Заради (4) и (5), само в случай на разпознаване на $\omega$ от $A$ автоматът $A'$ ще попадне в състояние $t_1$, ще изпразни стека и ще разпознае $\omega$ с празен стек. Ако случайно дума $\beta \notin L$ е изпразнила стека (което е възможно при разпознаване със заключителни състояния), сега това е невъзможно, тъй като новият начален символ на стека $s_0$ не е познат на $\delta$, и тя не може да го почисти. Следователно $A'$ разпознава същите думи както и $A$. □

**Теорема 4.3.4** *За всеки НСА $A$, разпознаващ езика $L$ с празен стек съществува НСА $A'$, разпознаващ $L$ със заключителни състояния.*

***Доказателство.*** Нека $A =< Q, X, Z, q_0, z_0, \delta, \emptyset >$. Построяваме $A' =< Q \cup \{t_0, t_1\}, X, Z \cup \{s_0\}, t_0, s_0, \delta', \{t_1\} >$, където $t_0, t_1 \notin Q, s_0 \notin Z$ и функцията на преходите $\delta'$ е дефинирана както следва:

1) $\delta'(t_0, \varepsilon, s_0) = \{(q_0, z_0 s_0)\}$;

2) $\delta'(q, \varepsilon, z) = \delta(q, \varepsilon, z)$ и $\delta'(q, x, z) = \delta(q, x, z)$ навсякъде, където $\delta$ е дефинирана;

3) $\delta'(q, \varepsilon, z)$ и $\delta'(q, x, z)$ са недефинирани, ако $\delta(q, \varepsilon, z)$ и $\delta(q, x, z)$ са недефинирани;

4) $\delta'(q, \varepsilon, s_0) \ni (t_1, \varepsilon)$.

Аналогично на предната теорема, дефиницията на $\delta'$ от (1), (2) и (3) осигурява точното моделиране на работата на $A$ до прочитането на произволна входна дума $\omega$. Ако $\omega \in L$, т.е. $A$ изпразва стека, то $A'$ ще разкрие на върха на стека си $s_0$ и съгласно (4) ще премине в заключителното $t_1$, разпознавайки $\omega$. В противен случай преминаването в $t_1$ е невъзможно и $A'$ също не разпознава $\omega$. □

Разпознаването с празен стек е по-естествено за стековите автомати и тъй като е еквивалентно на разпознаване със заключителни състояния, по-нататък под разпознаване ще разбираме разпознаване с празен стек.

**Теорема 4.3.5** *За всеки КСЕ $L$ съществува НСА $A$, който го разпознава.*

***Доказателство.*** Нека КСГ $\Gamma =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} >$ поражда езика $L$. Построяваме НСА $A =< \{q\}, \mathcal{T}, \mathcal{N} \cup \mathcal{T}, q, S, \delta, \emptyset >$, за който функцията $\delta$ дефинираме така:

а) $\delta(q, \varepsilon, B) = \{(q, \alpha) | \forall \alpha, B \to \alpha \in \mathcal{P}\}, \forall B \in \mathcal{N}$;

б) $\delta(q, a, a) = \{(q, \varepsilon)\}, \forall a \in \mathcal{T}$.

Преди да докажем формално, че $A$ разпознава точно езика $L$, да разгледаме поведението на $A$ върху някоя входна дума. В началото на върха на стека е аксиомата $S$ и съгласно (а) с $\varepsilon$-преход $A$ я замества недетерминирано с всички възможни десни части $\alpha$ на правила от вида $S \to \alpha$, т.е. $A$ започва едновременно всички възможни изводи от $S$. Ако думата в стека започва с терминали и те съвпадат с поредните букви от входната дума, тогава НСА $A$ ги изтрива от стека (и от входа) в съответствие с дефиницията (б) до появата на нетерминал. Ако терминалите не съвпадат с тези от входната дума, разпознаването по съответния клон не може да продължи.

Ясно е, че при премахване на терминалите, нетерминалът, който първи се появява на върха на стека, е най-левият в току що изведената дума. Той, съгласно (а) се замества по всички възможни начини от съответните десни части на правила, за които е лява част и т.н. По този начин $A$ недетерминирано опитва да извърши всички леви изводи от $S$, докато се стигне до една от двете възможности: или съответния извод не може да породи входната дума $\omega$ (получаваните по време на извода терминали в началото на извежданата дума не съответстват на тези от входната дума) или стекът се изпразва, което означава разпознаване. Наистина, в този случай последователните терминали, появяващи се в началото на извежданата дума точно са съвпаднали с буквите на входната дума.

Сега ще докажем, че $L = L_A$.

1) Нека $B \overset{\Gamma}{\models} \omega$ е ляв извод, $\omega \in \mathcal{T}^*$. С индукция по дължината на този извод ще докажем, че $< q, \omega\beta, B\beta' > \overset{A}{\models} < q, \beta, \beta' >, \forall \beta, \beta' \in (\mathcal{N} \cup \mathcal{T})^*$.

а) Ако $B \overset{\Gamma}{\vdash} \omega$, то съществува $B \to \omega$ и следователно $\delta(q, \varepsilon, B) \ni (q, \omega)$. Тогава $< q, \omega\beta, B\beta' > \overset{A}{\vdash} < q, \omega\beta, \omega\beta' > \overset{A}{\models} < q, \beta, \beta' >$.



б) Да допуснем, че за всеки ляв извод $B_i \overset{\Gamma}{\models} \omega_i, \omega_i \in \mathcal{T}^*$, с дължина $\leq n$ е в сила $< q, \omega_i\beta_i, B_i\beta_i' > \overset{A}{\models} < q, \beta_i, \beta_i' >, \forall \beta_i, \beta_i' \in (\mathcal{N} \cup \mathcal{T})^*$.

Нека $B \overset{\Gamma}{\models} \omega$ е ляв извод с дължина $n+1$, $\omega \in \mathcal{T}^*$. Нека първото правило в този извод е $B \to \alpha_0 B_1 \alpha_1 B_2 \ldots \alpha_{k-1} B_k \alpha_k$, където $\alpha_i \in \mathcal{T}^*, i = 0, 1, \ldots, k$, а $B_j \in \mathcal{N}, j = 1, 2, \ldots, k$. Следователно $\delta(q, \varepsilon, B) \ni (q, \alpha_0 B_1 \alpha_1 B_2 \ldots \alpha_{k-1} B_k \alpha_k)$ и $B \overset{\Gamma}{\vdash} \alpha_0 B_1 \alpha_1 B_2 \ldots \alpha_{k-1} B_k \alpha_k \overset{\Gamma}{\models} \omega$. Значи $\omega = \alpha_0 \omega_1 \alpha_1 \omega_2 \ldots \alpha_{k-1} \omega_k \alpha_k$ и $B_i \overset{\Gamma}{\models} \omega_i, \omega_i \in \mathcal{T}^*$ са леви изводи с дължини $\leq n$. Означаваме $\beta_i = \alpha_i \omega_{i+1} \ldots \omega_k \alpha_k \beta$, а с $\beta_i' = \alpha_i B_{i+1} \ldots B_k \alpha_k \beta'$. От индукционното предположение получаваме $< q, \omega_i\beta_i, B_i\beta_i' > \overset{A}{\models} < q, \beta_i, \beta_i' >$, $\forall i \in \{1, 2, \ldots, k\}$, $\forall \beta_i, \beta_i' \in (\mathcal{N} \cup \mathcal{T})^*$. Сега можем да построим последователността от трансформации на конфигурации на $A$:

$$
\begin{aligned}
& < q, \alpha_0\omega_1\alpha_1\omega_2 \ldots \alpha_{k-1}\omega_k\alpha_k\beta, B\beta' > \overset{A}{\vdash} \\
\overset{A}{\vdash} & < q, \alpha_0\omega_1\alpha_1\omega_2 \ldots \alpha_{k-1}\omega_k\alpha_k\beta, \alpha_0 B_1\alpha_1 B_2 \ldots \alpha_{k-1} B_k\alpha_k\beta' > \overset{A}{\models} \\
\overset{A}{\models} & < q, \omega_1\alpha_1\omega_2 \ldots \alpha_{k-1}\omega_k\alpha_k\beta, B_1\alpha_1 B_2 \ldots \alpha_{k-1} B_k\alpha_k\beta' > \overset{A}{\models} \\
\overset{A}{\models} & < q, \alpha_1\omega_2 \ldots \alpha_{k-1}\omega_k\alpha_k\beta, \alpha_1 B_2 \ldots \alpha_{k-1} B_k\alpha_k\beta' > \overset{A}{\models} \\
\overset{A}{\models} & < q, \omega_2 \ldots \alpha_{k-1}\omega_k\alpha_k\beta, B_2 \ldots \alpha_{k-1} B_k\alpha_k\beta' > \overset{A}{\models} \\
& \ldots \\
\overset{A}{\models} & < q, \omega_k\alpha_k\beta, B_k\alpha_k\beta' > \overset{A}{\models} \\
\overset{A}{\models} & < q, \alpha_k\beta, \alpha_k\beta' > \overset{A}{\models} < q, \beta, \beta' > .
\end{aligned}
$$

При тази трансформация последователно премахваме $\alpha_i$ от входа и от стека съгласно дефиницията (2), а след това премахваме $\omega_{i+1}$ от входа и $B_i$ от стека съгласно индукционното предположение, $i = 0, 1, \ldots, k-1$. Накрая премахваме $\alpha_k$ от входа и стека.

В частния случай $B = S, \beta = \beta' = \varepsilon$ получаваме, че ако $S \overset{\Gamma}{\models} \omega$ е ляв извод, $\omega \in \mathcal{T}^*$, то $< q, \omega, S > \overset{A}{\models} < q, \varepsilon, \varepsilon >$, т.е. $\omega \in L \Rightarrow \omega \in L_A$.

2) Нека $< q, \omega\beta, \varphi\beta' > \overset{A}{\models} < q, \beta, \beta' >, \forall \beta, \beta' \in (\mathcal{N} \cup \mathcal{T})^*$. С индукция по броя на $\varepsilon$-преходите ще покажем, че $\varphi \overset{\Gamma}{\models} \omega$.

а) Ако $< q, \omega\beta, \varphi\beta' > \overset{A}{\models} < q, \beta, \beta' >$ без $\varepsilon$-преходи, то $\omega = \varphi$ и $\varphi \overset{\Gamma}{\models} \omega$ заради рефлексивността на релацията на извод.

б) Да допуснем, че ако $< q, \omega'\beta_i, \varphi'\beta_i' > \overset{A}{\models} < q, \beta_i, \beta_i' >$, с $\leq n$ $\varepsilon$-прехода, то $\varphi' \overset{\Gamma}{\models} \omega'$.

в) Нека сега $< q, \omega\beta, \varphi\beta' > \overset{A}{\models} < q, \beta, \beta' >$ с $n+1$ $\varepsilon$-прехода, $\forall \beta, \beta' \in (\mathcal{N} \cup \mathcal{T})^*$. Да отделим частта от извода до първия $\varepsilon$-преход.

$$< q, \omega\beta, \varphi\beta' > \overset{A}{\models} < q, \omega_1\beta, A\varphi_1\beta' > \overset{A}{\vdash} < q, \omega_1\beta, \alpha\varphi_1\beta' > \overset{A}{\models} < q, \beta, \beta' > .$$

Това означава, че $\omega = \omega_0\omega_1$ и $\varphi = \omega_0 A\varphi_1$. Значи $\exists A \to \alpha$, защото $\delta(q, \varepsilon, A) \ni (q, \alpha)$ и $< q, \omega_1\beta, \alpha\varphi_1\beta' > \overset{A}{\models} < q, \beta, \beta' >$ с $n$ $\varepsilon$-прехода. От тук, съгласно индукционното предположение $\alpha\varphi_1 \overset{\Gamma}{\models} \omega_1$. Сега можем да построим извода

$$\varphi = \omega_0 A\varphi_1 \overset{\Gamma}{\vdash} \omega_0\alpha\varphi_1 \overset{\Gamma}{\models} \omega_0\omega_1 = \omega$$

или $\varphi \overset{\Gamma}{\models} \omega$.

В частния случай $\varphi = S$, $\beta = \beta' = \varepsilon$ получаваме $< q, \omega, S > \overset{A}{\models} \}\{ < q, \varepsilon, \varepsilon > \Rightarrow S \overset{\Gamma}{\models} \omega$, т.е. $\omega \in L_A \Rightarrow \omega \in L$.

От частните случаи в (1) и (2) получаваме $L = L_A$. □

За граматиката $\Gamma =< \{S\}, \{a, b\}, S, \{S \to ab, S \to aSb\} >$, пораждаща езика $L = \{a^i b^i | i = 1, 2, \ldots\}$, с конструкцията на теоремата построяваме НСА $A =< \{q\}, \{a, b\}, \{S, a, b\}, q, S, \delta, \emptyset >$, където $\delta(q, \varepsilon, S) = \{(q, ab), (q, aSb)\}$, $\delta(q, a, a) = \{(q, \varepsilon)\}$, $\delta(q, b, b) = \{(q, \varepsilon)\}$.

В сила е и обратното твърдение.

**Теорема 4.3.6** *За всеки НСА $A$, разпознаващ езика $L$ съществува КСГ $\Gamma$ такава, че $L = L_\Gamma$.*

***Доказателство.*** Ще посочим само конструкцията на търсената граматика за НСА $A =< q, X, Z, q_0, z_0, \delta, \emptyset >$. Построяваме $\Gamma =< \mathcal{N}, X, S, \mathcal{P} >$, където $\mathcal{N} = \{S\} \cup \{S_{q,z,p} | \forall q, p \in Q, \forall z \in Z\}$, а $\mathcal{P}$ съдържа следните правила:

а) $S \to S_{q_0,z_0,p}, \forall p \in Q$;

б) $S_{q,z,p_k} \to aS_{p,z_1,p_1}S_{p_1,z_2,p_2}\ldots S_{p_{k-1},z_k,p_k}$, $\forall p \in Q$, $\forall (p_1, p_2, \ldots, p_k) \in Q^k$, $\forall a \in X \cup \{\varepsilon\}$ и $\forall z, z_1, \ldots, z_k$ такива, че $\delta(q, a, z) \ni (p, z_1 z_2 \ldots z_k)$;

в) $S_{q,z,p} \to a$, $\forall p, q \in Q$, $\forall a \in X$ такива, че $\delta(q, a, z) \ni (p, \varepsilon)$.

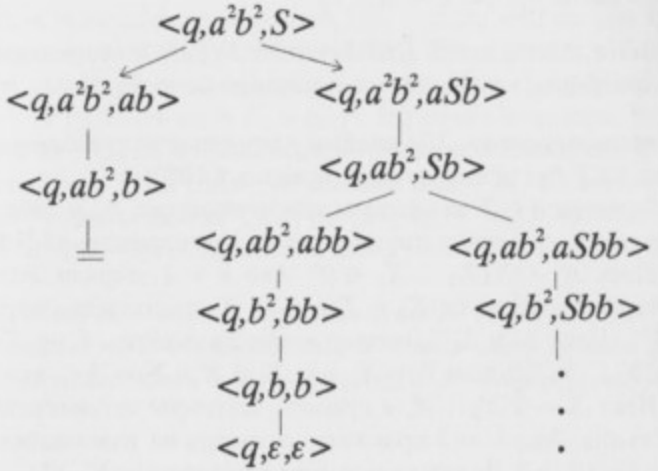
Оставяме на читателя да съобрази как тази доста тежка конструкция моделира със средствата на формалните граматики работата на недетерминирания стеков автомат. □

За получения по-горе НСА, разпознаващ езика $L = \{a^i b^i | i = 1, 2, \ldots\}$, с конструкцията на теоремата построяваме граматиката $\Gamma =< \{S, S_S, S_a, S_b\}, \{a, b\}, S, \{S \to S_S, S_S \to S_a S_b, S_S \to S_a S_S S_b, S_a \to a, S_b \to b\} >$, където за по-просто сме използвали означението $S_X$ за всеки нетерминал $S_{qXq}$ от теоремата.



Въпросът за детерминизация на НСА е малко по-сложен, тъй като няма проста конструкция за построяване на детерминиран стеков автомат по зададен произволен недетерминиран стеков автомат. Въпреки това въпросът за детерминизацията се решава задоволително.

Достатъчно е да си припомним, че НСА $A$, работейки върху произволна входна дума, строи графовата структура на възможните изводи от аксиомата. Не трябва да се бърка дървото на извод на думата $\omega$ с графовата структура на възможните изводи, управлявани от $\omega$ (вж. Фиг. 4.18 за $\Gamma =< \{S\}, \{a, b\}, S, \{S \to ab, S \to aSb\} >$ и $\omega = a^2b^2$), която в общия случай не е краен граф. Това не ни пречи да започнем динамичното построяване на близките до корена части на тази практически безкрайна структура по някоя от схемите за обхождане на граф – в дълбочина или в ширина.




Фигура 4.18: Дърво на анализа.

Да се спрем на обхождането в дълбочина, тъй като обхождането в ширина изисква повече памет. Търсенето в дълбочина на разпознаваща конфигурация за дадена дума $\omega$ е заплашено от потенциалната опасност да попаднем на безкраен клон (винаги надясно на Фиг. 4.18) и никога да не завършим детерминизацията. Сега изключително полезно се оказва свойството на разглежданите граматики да са несъкращаващи, т.е.

за всяко правило $\alpha \to \beta$ имаме $d(\alpha) \leq d(\beta)$. По този начин изводите, които правим са също несъкращаващи. Това ни позволява да контролираме дължината на стековата дума $\gamma$ и дължината на остатъка от входната дума $\omega'$ и ако $d(\omega') < d(\gamma)$ да спрем спускането в дълбочина и да направим стъпка назад. Така детерминизацията на недетерминиран стеков автомат се извършва с помощта на алгоритмична техника, вариант на обхождането в дълбочина, известна като „разклонения и граници" („branch and bound").

### 4.3.3 Нормални форми на КСГ

В този подраздел ще използваме следното означение. Когато имаме много правила с една и съща лява част $A \to \alpha_1$, $A \to \alpha_2, \ldots, A \to \alpha_m$, за по-просто ще ги записваме $A \to \alpha_1 | \alpha_2 | \ldots | \alpha_m$.

***Дефиниция.*** КСГ $\Gamma =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} >$ е в *нормална форма на Чомски (НФЧ)*, ако всяко правило (различно от $S \to \varepsilon$) е от вида $A \to BC$ или $A \to a$, където $A, B, C \in \mathcal{N}, a \in \mathcal{T}$.

**Теорема 4.3.7** *За всяка КСГ $\Gamma =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} >$ съществува КСГ $\Gamma'$ в нормална форма на Чомски, която поражда езика $L_\Gamma$.*

***Доказателство.*** Ще опишем алгоритъм за преобразуване на произволна КСГ без преименуващи правила в НФЧ:

1. За всеки $a \in \mathcal{T}$ въвеждаме нов нетерминал $A_a$ и добавяме към $\mathcal{P}$ правилото $A_a \to a$ което съответства на изискванията за НФЧ.

2. Нека $A \to X_1 X_2 \ldots X_k \in \mathcal{P}$. Ако $k = 1$, поради отсъствието на преименуващи правила, $X_1 \in \mathcal{T}$ и $A \to X_1$ съответства на изискването за НФЧ. Нека $k > 1$. Заменяме всяко правилото $A \to X_1 X_2 \ldots X_k$ с $A \to Y_1 Y_2 \ldots Y_k$, където $Y_i = X_i$, ако $X_i \in \mathcal{N}$ и $Y_i = A_{X_i}$, ако $X_i \in \mathcal{T}$.

3. Нека $A \to Y_1 Y_2 \ldots Y_k$ е правило, получено от замяната в предишната стъпка. Ако $k = 2$ правилото отговаря на изискванията за НФЧ. Затова нека $k \geq 3$. Въвеждаме новите нетерминали $Y'_1, Y'_2, \ldots, Y'_{k-2}$ (уникални за всяко отделно правило). Заместваме $A \to Y_1 Y_2 \ldots Y_k$ с правилата $A \to Y_1 Y'_1, Y'_1 \to Y_2 Y'_2, \ldots, Y'_{k-3} \to Y_{k-2} Y'_{k-2}, Y'_{k-2} \to Y_{k-1} Y_k$. Очевидно е, че получената граматика е в нормална форма на Чомски, като езикът който поражда е $L_\Gamma$. □

За граматиката $\Gamma =< \{S\}, \{a, b\}, S, \{S \to ab, S \to aSb\} >$ получаваме следната граматика в НФЧ

$$\begin{aligned} \Gamma' \;=\; & < \{S, A_a, A_b, A'\}, \{a, b\}, S, \\ & \{S \to A_a A_b, S \to A_a A', A' \to S A_b, A_a \to a, A_b \to b\} > . \end{aligned}$$



***Дефиниция.*** Всяко правило от вида $A \to A\alpha$ наричаме *ляво-рекурсивно.*

**Лема 4.3.1** *Нека $\Gamma$ е КСГ, в която $A \to A\alpha_1 | A\alpha_2 | \ldots | A\alpha_m$ са всички ляво-рекурсивни правила с $A$ в лява част, а $A \to \beta_1 | \beta_2 | \ldots | \beta_n$ са всички не ляво-рекурсивни правила с $A$ в лява част. Езикът, който $\Gamma$ поражда не се променя, ако заменим всички правила за $A$ с*

$$A \to \beta_1 | \beta_2 | \ldots | \beta_n | \beta_1 A' | \beta_2 A' | \ldots | \beta_n A'$$

*и*

$$A' \to \alpha_1 A' | \alpha_2 A' | \ldots | \alpha_m A' | \alpha_1 | \alpha_2 | \ldots | \alpha_m,$$

*където $A'$ е нов нетерминал, уникален за $A$.*

***Доказателство.*** Във всеки ляв извод в $\Gamma$ да отделим тези участъци, в които се използват само правила с $A$ в лява част. Във всеки такъв участък отначало се прилагат $0, 1, 2, \ldots$ ляво-рекурсивни правила и накрая едно не ляво-рекурсивно правило. Извежданата от $A$ поддума за всеки такъв участък се описва със следния (регулярен) израз: $(\beta_1 + \beta_2 + \ldots + \beta_n)(\alpha_1 + \alpha_2 + \ldots + \alpha_m)^*$. Но точно този израз описва и извежданите от $A$ думи в променената граматика. Разликата е, че при нея в извежданата дума първо се появява някое $\beta_i$, а след това последователно от ляво-надясно думи от $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$. Забележете, че полученият от изменената граматика извод може и да не е ляв. □

**Лема 4.3.2** *Нека $A \to \alpha_1 B \alpha_2$ е правило на КСГ $\Gamma$, а $B \to \beta_1 | \beta_2 | \ldots | \beta_m$ са всички правила на $\Gamma$ с нетерминала $B$ в лява част. Езикът на $\Gamma$ не се променя, ако заменим в нея $A \to \alpha_1 B \alpha_2$ с $A \to \alpha_1 \beta_1 \alpha_2 | \alpha_1 \beta_2 \alpha_2 | \ldots | \alpha_1 \beta_m \alpha_2$.*

Твърдението е очевидно.

***Дефиниция.*** КСГ $\Gamma =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} >$ е в *нормална форма на Грейбах* (НФГ), ако всяко правило на $\Gamma$ (различно от $S \to \varepsilon$) е от вида $A \to a$ или $A \to a B_1 B_2 \ldots B_k$, където $A, B_i \in \mathcal{N}, a \in \mathcal{T}$.

**Теорема 4.3.8** *За всяка КСГ $\Gamma =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} >$ съществува КСГ $\Gamma'$ в нормална форма на Грейбах, пораждаща езика $L_\Gamma$.*

***Доказателство.*** Без ограничение на общността можем да смятаме, че $\Gamma$ е без преименуващи правила и в НФЧ. Да означим с $A_0 =$

$S, A_1, \ldots, A_n$ нетерминалите на Г. Ще опишем алгоритъм, преобразуващ Г в НФГ.

*Първи етап.* На този етап оставяме настрана всяко правило, започващо с терминал в дясната част, което е от граматиката или се включва в нея при извършваните преобразувания.

1) Ще преобразуваме граматиката в еквивалентна, всички правила на която, започващи с нетерминал в дясна част, да имат вида $A_i \to A_j\alpha, i < j$. Построението ще направим с индукция по номера $i$ на нетерминала $A_i$.

а) Нека $i = 0$. Всяко правило от вида $A_0 \to A_j\alpha, j \neq 0$, е в търсения вид. Правилата $A_0 \to A_0\alpha$ са ляво-рекурсивни, и с помощта на Лема 4.3.1 изчистваме лявата рекурсия спрямо $A_0$. Оставяме настрана новите правила, започващи с терминал, както и тези в чиято лява част е новодобавеният нетерминал $A'_0$. Остават само правила от вида $A_0 \to A_j\alpha, j > 0$.

б) Нека правилата за $A_0, A_1, \ldots, A_k$, започващи с нетерминал в дясна част са представени в искания вид. Ще преобразуваме правилата от вида $A_{k+1} \to A_j\alpha$ така че $j$ да стане по-голямо от $k+1$. В съответствие с Лема 4.3.2 преобразуваме всяко правило от вида $A_{k+1} \to A_0\alpha$ така, че в получените правила имаме за начало на дясна част терминал или нетерминал $A_j, j > 0$. Аналогично преобразуваме правилата за $A_{k+1}$, започващи с $A_1, \ldots, A_k$ в дясна част (включително и получените от предишни подстъпки на тази стъпка). Накрая изчистваме лявата рекурсия в правилата $A_{k+1} \to A_{k+1}\alpha$ с помощта на Лема 4.3.1. Така получаваме искания вид за правилата с $A_{k+1}$ в лява част. Забележете, че в края на тази стъпка всички правила за нетерминала $A_n$ ще имат дясна част, започваща с терминал.

2) Индуктивно трансформираме граматиката, получена на предната стъпка, в еквивалентна граматика, за която всяко правило с $A_i$ в лява част има дясна част, започваща с терминал.

а) От предната стъпка е ясно, че исканият вид е получен за всички правила с $A_n$ в лявата част. Нека $i = n$.

б) Допускаме, че за $A_n, A_{n-1}, \ldots, A_i$ исканият вид е получен. Всяко правило с $A_{i-1}$ в лява част, започващо с нетерминал в дясно е от вида $A_{i-1} \to A_j\alpha, j > i-1$. Затова от индукционното предположение за правилата на $A_n, A_{n-1}, \ldots, A_i$ и Лема 4.3.2 трансформираме граматиката в нова, в която и правилата за $A_{i-1}$ започват с терминал в дясна част.

След завършване на тази стъпка всички правила за старите нетерминали $A_0, A_1, \ldots, A_n$ имат дясна част, започваща с терминал.

*Втори етап.* Да разгледаме правилата за новодобавените от Лема 4.3.1 нетерминали. Десните части на тези правила започват или с терминал или с някой от старите нетерминали. Всички правила от вида $A'_i \to A_j\alpha$ преобразуваме с помощта на Лема 4.3.2 в правила започващи с терминал в дясна част.



Сега всички правила на получената граматика, еквивалентна на дадената, са във вида, изискван от НФГ. □

За пример да разгледаме граматиката с правила $A_0 \to A_1A_0, A_1 \to A_2A_0, A_1 \to a, A_2 \to A_1A_1, A_2 \to b$.

*Първи етап.* 1) Единственото правило, което не съответства на изискванията на тази стъпка е $A_2 \to A_1A_1$. Заместваме го с $A_2 \to A_2A_0A_1$ и $A_2 \to aA_1$. Изчистваме лявата рекурсия за $A_2$ и получаваме правилата

$$\begin{aligned} A_2 &\to aA_1|b|aA_1A'_2|bA'_2 \\ A'_2 &\to A_0A_1|A_0A_1A'_2 \end{aligned}$$

които заместват старите правила за $A_2$.

2) На тази стъпка получаваме следните правила за $A_1$:

$$A_1 \to a|aA_1A_0|bA_0|aA_1A'_2A_0|bA'_2A_0,$$

а за $A_0$

$$A_0 \to aA_0|aA_1A_0A_0|bA_0A_0|aA_1A'_2A_0A_0|bA'_2A_0A_0$$

вместо старите правила за $A_1$ и $A_0$.

*Втори етап.*Преобразуваме правилата за единствения новодобавен нетерминал $A'_2$ в:

$$\begin{aligned} A'_2 \to\ & aA_0A_1|aA_1A_0A_0A_1|bA_0A_0A_1|aA_1A'_2A_0A_0A_1| \\ & |bA'_2A_0A_0A_1|aA_0A_1A'_2|aA_1A_0A_0A_1A'_2|bA_0A_0A_1A'_2| \\ & |aA_1A'_2A_0A_0A_1A'_2|bA'_2A_0A_0A_1A'_2. \end{aligned}$$

И така получихме НФГ за дадената граматика.

Нормалната форма на Грейбах дава възможност за ускоряване на детерминизацията на НСА по следната схема: Ако на върха на стека е нетерминалът $A$, а първата непрочетена входна буква е $x$, то от всички правила с лява част $A$, само тези, започващи в дясната част с терминала $x$ биха могли да доведат до разпознаване. В най-добрия случай можем да получим граматика, в която различните правила с $A$ в лява част започват с различни терминали в дясната част. Тогава стековият автомат, получен от Теорема 4.3.5, ще бъде детерминиран и разпознаването или неразпознаването на произволна дума $\omega$ ще става много бързо. Тази техника се нарича *анализ с поглеждане на една буква напред.* Естествено

може да се построи алгоритъм за анализ с поглеждане на $2, 3, \ldots$ букви напред, ако граматиката го позволява. За съжаление, съществуват КСГ, за които не съществува такова $n$, че да е възможно да се построи алгоритъм за анализ с поглеждане на $n$ букви.

За пример да разгледаме граматиката

$$\begin{aligned} S &\to aAB|bS \\ A &\to aA|bB \\ B &\to AB|c, \end{aligned}$$

която лесно се преобразува в НФГ:

$$\begin{aligned} S &\to aAB|bS \\ A &\to aA|bB \\ B &\to aAB|bBB|c, \end{aligned}$$

при което различните правила за кой да е нетерминал имат десни части, които започват с различни нетерминали. Затова анализът на произволна дума може да бъде направен бързо с поглеждане на една буква напред. Дървото на работа върху произволна дума се състои само от един клон и не се нуждае от детерминиране (връщане назад по време на анализа).

### 4.3.4 xyuvw-Теорема

**Теорема 4.3.9 (xyuvw-Теорема)** *За всеки непразен КСЕ $L \neq \{\varepsilon\}$ съществува цяло $n > 0$ такова, че ако $\alpha \in L, d(\alpha) > n$, то $\alpha = xyuvwu$*

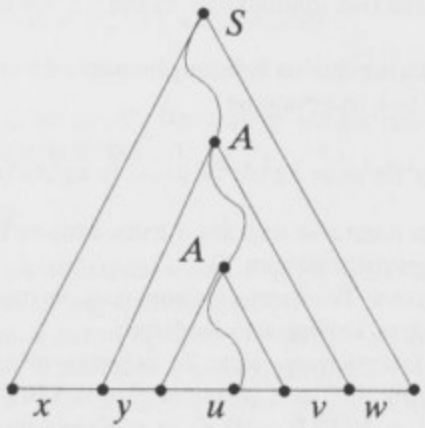
*1) $d(u) \geq 1$, $d(yv) \geq 1$, $d(yuv) \leq n$;*
*2) $xy^iuv^iw \in L$, $\forall i = 0, 1, 2, \ldots$.*

***Доказателство.*** Нека $\Gamma =< \mathcal{N}, \mathcal{T}, S, \mathcal{P} >$ е КСГ без преименуващи правила, която поражда езика $L$. Нека $l = |\mathcal{N}|$, а $m = \max\limits_{\forall A \to \beta \in \mathcal{P}} d(\beta)$, $m \geq 1$. Нека $n = m^{l+1}$. Ще покажем, че това е търсеното цяло положително число.

Нека $\alpha \in L, d(\alpha) > n$. Нека $D$ е дърво на извод за $\alpha$ в $\Gamma$ с височина $h$. Тъй като $D$ е дърво на извод, всеки връх има $\leq m$ сина и съгласно Теорема 3.2.4 броят на листата му е $d(\alpha) \leq m^h$. Следователно $m^{l+1} = n < d(\alpha) \leq m^h$ или $m^{l+1} < m^h$. От монотонността на показателната функция при $m \geq 1$ получаваме, че $l+1 < h$. Но височината на $D$ е равна на броя на нетерминалите, последователно срещащи се по най-дългия път от



корена до някой лист (забележете, че последният връх на пътя е означен с терминал). Следователно този брой е $> l$. От Принципа на Дирихле следва, че по този път поне един нетерминал $A$ ще се срещне поне два пъти. Да изберем най-близките до листа две срещания на $A$ (такива, че след по-близкото до корена срещане няма друг нетерминал, който се среща поне два пъти). На Фиг. 4.19 е показана получената ситуация.




Фигура 4.19: xyuvw-Теорема.

С $D'(V', E')$ означаваме поддървото с корен по-близкия до корена на $D$ нетерминал $A$, а с $D''(V'', E'')$ поддървото с корен по-далечният от корена на $D$ нетерминал $A$. Сега думата $\alpha$ на $D$ естествено се разпада на пет поддуми $x, y, u, v$ и $w$ така, че $u$ е думата на поддървото $D''$ (в граматиката $\Gamma_A$), $y$ – поддумата от листата на $D'$, стоящи вляво от листата на $D''$, $v$ – поддумата от листата на $D'$, стоящи вдясно от листата на $D''$, $x$ – поддумата от листата на $D$, стоящи вляво от листата на $D'$ и $w$ – поддумата от листата на $D$, стоящи вдясно от листата на $D'$.

При това за дървото $D''$ в граматиката $\Gamma_A$ имаме дума $u$, за дървото $D' \setminus D''$ в граматиката $\Gamma_A$ имаме дума $yAv$, а за дървото $D \setminus D'$ в граматиката $\Gamma$ – думата $xAw$. От това, че правилата на $\Gamma$ и $\Gamma_A$ съвпадат, следва, че $S \overset{\Gamma}{\models} xAw$, $A \overset{\Gamma}{\models} yAv$ и $A \overset{\Gamma}{\models} u$. Ще покажем че са в сила твърденията (1) и (2) на Теоремата.

1) $d(u) \geq 1$ защото в $D''$ има поне един лист. $d(yv) \geq 1$, защото в противен случай $y = v = \varepsilon$ и коренът на $D'$, означен с $A$, ще има само един син означен с нетерминал, например $B$. Тогава в $\Gamma$ ще се среща

преименуващо правило $A \to B$, което е невъзможно. Ако допуснем, че $d(yuv) \leq n$ не е вярно, с разсъждения подобни на по-горе направените, ще стигнем до извода, че след по-близкия до корена нетерминал $A$ ще има поне още една двойка повтарящи се нетерминали, което е изключено от избора на разглежданата двойка – най-отдалечената от корена на $D$.

2) За $i = 0$ имаме извода $S \overset{\Gamma}{\models} xAw \overset{\Gamma}{\models} xuw = xy^0uv^0w$. Следователно $xy^0uv^0w \in L$, като сме приложили първия и третия от получените изводи.

За $i = 1$ имаме $xy^1uv^1w = \alpha \in L$ по условие.

За произволно $i > 1$ получаваме

$$S \overset{\Gamma}{\models} xAw \overset{\Gamma}{\models} xy^1Av^1w \overset{\Gamma}{\models} xy^2Av^2w \overset{\Gamma}{\models} \ldots \overset{\Gamma}{\models} xy^iAv^iw \overset{\Gamma}{\models} xy^iuv^iw,$$

като сме приложили един път първия, $i$ пъти втория и накрая един път третия от горе получените изводи. □.

С помощта на xyuvw-Теоремата ще покажем, че съществуват контекстни езици, които не са контекстно-свободни.

Да разгледаме контекстния език $L$, породен от граматиката $\Gamma_3 = < \{S, A, B, C\}, \{a, b, c\}, S, \mathcal{P} >$ с правила $S \to aSBC, S \to abC$, $CB \to AB$, $AB \to AC, AC \to BC, bB \to bb, C \to c$. Както видяхме в 4.1 $L_{\Gamma_3} = \{a^ib^ic^i | i = 1, 2, \ldots\}$.

**Следствие 4.3.1** *Езикът $L = \{a^ib^ic^i | i = 1, 2, \ldots\}$ не е контекстно-свободен.*

***Доказателство.*** Да допуснем противното и да приложим xyuvw-Теоремата. Намираме $n$ и избираме $i > n/3$. Получаваме $\alpha = a^ib^ic^i$, $d(\alpha) = 3i > n$. Следователно $\alpha = xyuvw, d(yv) \geq 1$ и $xy^2uv^2w \in L$. Без ограничение на общността да допуснем, че $d(y) \geq 1$ (аналогично се разглежда случаят $d(v) \geq 1$, ако $d(y) = 0$). За $y$ има следните възможности:

- $y = a^k, k > 0$. За да не се наруши броят на буквите в $xy^2uv^2w$ трябва $v = b^kc^k$, но в такъв случай $xy^2uv^2w = xy^2ub^kc^kb^kc^kw$, а това не е възможно в $L$. Следователно $y$ не е от този вид;

- $y = b^k$ или $y = c^k, k > 0$. В такъв случай броят на буквите $b$ (или на буквите $c$) ще бъде по-голям от броя на буквите $a$ в $xy^2uv^2w$, което е недопустимо;

- $y$ е съставено от повече от една буква. Но тогава в $y^2$ще се нарушава последователността на срещане (букви $a$ след букви $b$ или $c$, или букви $b$ след букви $c$), което означава че и в този случай $xy^2uv^2w \notin L$.

Следователно езикът $L$ не е контекстно свободен. □



## Упражнения

**Упражнение 4.1**

Дадена е азбуката $A = \{a, b, c\}$ и правилата

$$ab \Rightarrow abc, a\alpha \Rightarrow a\alpha\alpha, \alpha bb\beta \Rightarrow \alpha c\beta, \alpha cc\beta \Rightarrow \alpha\beta,$$

където $\alpha$ и $\beta$ са думи от $A^*$. Проверете дали с горните правила е възможно от $ab$ да се изведе $ac$.

**Упражнение 4.2**

Постройте КДА, разпознаващ езика $L \subseteq \{0, 1\}^*$, състоящ се от:

а) всички думи, не завършващи на 11;

б) всички думи, в които броят на буквите 0 се дели на 3;

в) всички думи, в които от 4 последователни букви, поне една е 1;

г) всички думи започващи с 1, които са двоично представяне на естествени числа, делящи се на 5.

**Упражнение 4.3**

Постройте КДА с азбука $\{I, V, X, L\}$, разпознаващ римските представяния на числата от 1 до 89.

**Упражнение 4.4**

Опишете с думи езиците разпознавани от КДА:

а) $A =< \{q_0, q_1, q_2\}, \{a, b\}, q_0, \delta, \{q_0, q_1\} >$, където

$$\begin{aligned} \delta(q_0, a) &= q_0, & \delta(q_0, b) &= q_1, \\ \delta(q_1, a) &= q_2, & \delta(q_1, b) &= q_2, \\ \delta(q_2, a) &= q_2, & \delta(q_2, b) &= q_2; \end{aligned}$$

б) $A =< \{q_0, q_1, q_2, q_3\}, \{a, b\}, q_0, \delta, \{q_0\} >$, където

$$\begin{aligned} \delta(q_0, a) &= q_1, & \delta(q_0, b) &= q_2, \\ \delta(q_1, a) &= q_3, & \delta(q_1, b) &= q_0, \\ \delta(q_2, a) &= q_0, & \delta(q_2, b) &= q_3, \\ \delta(q_3, a) &= q_3, & \delta(q_3, b) &= q_3. \end{aligned}$$

**Упражнение 4.5**

Детерминирайте КНА:

а) $A =< \{q_0, q_1, q_2, q_3\}, \{a, b\}, q_0, \delta, \{q_3\} >$, където

$$\begin{aligned} &\delta(q_0, a) = \{q_0, q_1\}, && \delta(q_0, b) = \{q_0\}, \\ &\delta(q_1, a) = \{q_2\}, && \delta(q_1, b) = \{q_2\}, \\ &\delta(q_2, a) = \{q_3\}, && \\ &\delta(q_3, a) = \{q_3\}, && \delta(q_3, b) = \{q_3\}; \end{aligned}$$

б) $A =< \{q_0, q_1, q_2, q_3\}, \{a, b\}, q_0, \delta, \{q_1, q_3\} >$, където

$$\begin{aligned} &\delta(q_0, a) = \{q_1, q_3\}, && \delta(q_0, b) = \{q_1\}, \\ &\delta(q_1, a) = \{q_2\}, && \delta(q_1, b) = \{q_1, q_2\}, \\ &\delta(q_2, a) = \{q_3\}, && \delta(q_2, b) = \{q_1\}, \\ &\delta(q_3, b) = \{q_1\}. && \end{aligned}$$

**Упражнение 4.6**

Постройте регулярни изрази за езиците от Упражнение 4.2 и 4.4.

**Упражнение 4.7**

Постройте КДА, разпознаващ езика представен с регулярния израз:

а) $10 + (0 + 11)0^*1$;

б) $((0 + 1)(0 + 1))^* + ((0 + 1)(0 + 1)(0 + 1))^*$;

в) $(1 + 01 + 001)^*(\varepsilon + 0 + 00)$.

**Упражнение 4.8**

Постройте автоматни граматики за езиците от Упражнение 4.2 и Упражнение 4.4.

**Упражнение 4.9**

Определете броя на думите с дължина 4 в езиците от Упражнение 4.4.

**Упражнение 4.10**

Обратен език на $L \subseteq A^*$ наричаме $\overline{L} = \{a_{i_k} a_{i_{k-1}} \dots a_{i_1} | \; a_{i_1} a_{i_2} \dots a_{i_k} \in L\}$. Постройте КДА за езиците, обратни на тези от Упражнение 4.4.

**Упражнение 4.11**

За всеки от следните езици проверете дали е автоматен или не:

а) $\{0^{2n} | n \geq 1\}$;

б) $\{0^m 1^n 0^{m+n} | m \geq 1, n \geq 1\}$;

в) $\{0^p | p - \text{просто}\}$.

**Упражнение 4.12**

Минимизирайте автомата $A =< \{a, b, c, d, e, f, g, h\}, \{0, 1\}, a, \delta, \{d\} >$, където

$$\begin{gathered} \delta(a, 0) = b, \delta(a, 1) = a, \delta(b, 0) = a, \delta(b, 1) = c, \delta(c, 0) = d, \delta(c, 1) = b \\ \delta(d, 0) = d, \delta(d, 1) = a, \delta(e, 0) = d, \delta(e, 1) = f, \delta(f, 0) = g, \delta(f, 1) = e, \\ \delta(g, 0) = f, \delta(g, 1) = g, \delta(h, 0) = g, \delta(h, 1) = d. \end{gathered}$$

**Упражнение 4.13**

Кои са класовете на еквивалентност на релацията $R_L$ за $L = \{0^n 1^n\}$? Докажете, използвайки полученото, че $L$ не е автоматен език.

**Упражнение 4.14**

Постройте контекстно-свободна граматика за езика съставен от:

а) всички думи $\alpha = a_{i_1} a_{i_2} \dots a_{i_k}$, такива, че $\alpha = a_{i_k} a_{i_{k-1}} \dots a_{i_1}$;

б) всички правилни скобови изрази;

в) всички формули на променливите $x, y, z$ над множеството булеви функции $\{xy, x \vee y\}$;

г) всички думи над $\{a, b\}$, в които броят на буквите $a$ е два пъти по-голям от броя на буквите $b$.

**Упражнение 4.15**

За граматиката с правила $S \Rightarrow aB|bA, A \Rightarrow a|aS|bAA, B \Rightarrow b|bS|aBB$ постройте:

а) ляв извод; б) десен извод; в) дърво на извод
за думата *aaabbabbba*.

**Упражнение 4.16**

Постройте недетерминирани стекови автомати за езиците от Упражнение 4.14.

**Упражнение 4.17**

Постройте недетерминиран стеков автомат за езика породен от граматиката $S \Rightarrow AA|0, A \Rightarrow SS|1$

**Упражнение 4.18**

Преобразувайте граматиката от Упражнение 4.15 в нормална форма на Чомски.

**Упражнение 4.19**

Преобразувайте граматиката с правила $S \Rightarrow AA|0, A \Rightarrow SS|1$ в нормална форма на Грейбах.

**Упражнение 4.20**

Докажете, че следните езици не са контекстно-свободни:

а) $\{a^i b^j c^k | i < j < k\}$;

б) $\{a^i b^j | j = i^2\}$;

в) $\{a^i | i$ - просто$\}$.

# Глава 5

# Дискретни функции

## 5.1 Суперпозиция и формули

Всяка функция $f : A \to B$, където $A$ и $B$ са изброими, бихме могли да наречем дискретна, но в тази глава в понятието влагаме малко по-тесен смисъл.

**Дефиниция.** Нека $A$ е крайно множество с $q$ елемента, като без ограничение на общността ще считаме $A = J_q = \{0, 1, \ldots, q-1\}$. Всяка функция $f : A^n \to A$, където $n$ е цяло число, $n \geq 1$, наричаме *дискретна* (или $q$-ична ) функция.

Естествен начин за задаване на дискретна функция е таблицата на стойностите ѝ, състояща се от два стълба. В първия са изредени всички $q^n$ стойности от дефиниционната област $A^n$, а във втория стълб срещу всеки елемент $(a_1, a_2, \ldots, a_n)$ от първия стълб се намира стойността на функцията $f(a_1, a_2, \ldots, a_n)$. В Табл. 5.1 е показана схематично таблицата на функция $f : A^n \to A$. За по-просто вместо $(a_1, a_2, \ldots, a_n)$ понякога пишем $a_1 a_2 \ldots a_n$.

| $A^n$ | | | | | $f$ |
|---|---|---|---|---|---|
| 0 | 0 | ... | 0 | 0 | $f(0,0,\ldots,0,0)$ |
| 0 | 0 | ... | 0 | 1 | $f(0,0,\ldots,0,1)$ |
| 0 | 0 | ... | 0 | 2 | $f(0,0,\ldots,0,2)$ |
| | | ... | | | ... |
| $a_1$ | $a_2$ | ... | $a_{n-1}$ | $a_n$ | $f(a_1,a_2,\ldots,a_{n-1},a_n)$ |
| | | ... | | | ... |
| $q-1$ | $q-1$ | ... | $q-1$ | $q-1$ | $f(q-1,q-1,\ldots,q-1,q-1)$ |

Таблица 5.1: Таблица на дискретна функция $f : A^n \to A$.

Ще посочим няколко често употребявани $q$-ични функции:

а) $\max(x_1, x_2) = \begin{cases} x_1 & \text{ако } x_1 \geq x_2 \\ x_2 & \text{ако } x_1 < x_2 \end{cases}$;

б) $\min(x_1, x_2) = \begin{cases} x_1 & \text{ако } x_1 \leq x_2 \\ x_2 & \text{ако } x_1 > x_2 \end{cases}$;

в) $x_1 + x_2 (\mathrm{mod}\, q)$;

г) $x_1 . x_2 (\mathrm{mod}\, q)$;

д) $x_1^{x_2} = \begin{cases} q-1 & \text{ако } x_1 = x_2 \\ 0 & \text{ако } x_1 \neq x_2 \end{cases}$;

е) $\tilde{c} = f_c(x) = c, c \in \{0, 1, \ldots, q-1\}$.

Стойността $f(a_1, a_2, \ldots, a_n)$ можем да намерим, като последователно сравняваме дадения вектор $(a_1, a_2, \ldots, a_n)$ с векторите от левия стълб на таблицата, докато установим реда, в който той се намира. Търсената стойност е елементът от десния стълб в същия ред. Този начин на пресмятане на дискретни функции (приложим и в по-общия случай на произволна крайна дефиниционна област) понякога е единствената възможност за ефективно намиране на $f(a_1, a_2, \ldots, a_n)$. Наричаме го *пълно изчерпване*. В случай, че елементите на дефиниционната област са подредени по някакъв начин, процедурата може значително да се ускори, но в общия случай времето за пресмятане на функцията ще зависи от дължината на таблицата. Една от целите на тази глава е да посочим други начини за ефективно изчисляване на дискретни функции.

Да означим с $\mathcal{F}_q$ множеството $\{f : A^i \to A | i = 1, 2, \ldots\}$, за $A = J_q$. Функцията $f : A^n \to A$ разглеждаме като функция на $n$ независими променливи $x_1, x_2, \ldots, x_n$, всяка от които може да приема стойности от $A$. Означаваме с $\mathcal{F}_q^n$ функциите от $\mathcal{F}_q$, които са на $n$ променливи. Специалният вид на функциите от $\mathcal{F}_q$ ни позволява да дефинираме следната операция между функции, обобщение на композицията на едноаргументни функции:

***Дефиниция.*** Нека $f(x_1, x_2, \ldots, x_n) \in \mathcal{F}_q^n$ и $g(y_1, y_2, \ldots, y_m) \in \mathcal{F}_q^m$. Функцията

$$h(x_1, \ldots, x_{i-1}, y_1, \ldots, y_m, x_{i+1}, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, g(y_1, \ldots, y_m), x_{i+1}, \ldots, x_n)$$

наричаме *суперпозиция* на $g$ в $f$ на мястото на променливата $x_i$.

Пресмятането на суперпозицията $h$ на $g$ в $f$ за произволни стойности на променливите, например $h(a_1, \ldots, a_{i-1}, b_1, \ldots, b_m, a_{i+1}, \ldots, a_n)$, може да стане по следния начин. От таблицата на $g$ намираме $b = g(b_1, b_2, \ldots, b_m)$, а след това в таблицата на $f$ намираме $f(a_1, \ldots, a_{i-1}, b, a_{i+1}, \ldots, a_n)$ и това е търсената стойност на $h$.



| $x_1$ | $x_2$ | $f(x_1, x_2)$ | $h(x_1, x_2)$ |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 0 | 1 | 2 | 2 |
| 0 | 2 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 2 | 2 |
| 1 | 2 | 0 | 0 |
| 2 | 0 | 0 | 1 |
| 2 | 1 | 2 | 2 |
| 2 | 2 | 1 | 0 |

Таблица 5.2: Троични функции $f(x_1, x_2)$ и $h(x_1, x_2)$.

Обобщаваме операцията суперпозиция така:

***Дефиниция.*** Нека $f(x_1, x_2, \ldots, x_n) \in \mathcal{F}_q^n$ и $g_i(y_1, y_2, \ldots, y_m) \in \mathcal{F}_q^m$, $i = 1, 2, \ldots, n$. Функцията

$$h(y_1, y_2, \ldots, y_m) = f(g_1(y_1, \ldots, y_m), g_2(y_1, \ldots, y_m), \ldots, g_n(y_1, \ldots, y_m))$$

наричаме *суперпозиция* на $g_1, g_2, \ldots, g_n$ в $f$.

За да пресметнем стойността на суперпозицията $h$ на $g_1, g_2, \ldots, g_n$ в $f$ за вектора $(b_1, b_2, \ldots, b_m)$, постъпваме по следния начин. От таблиците на $g_1, g_2, \ldots, g_n$ пресмятаме вектора $(c_1, c_2, \ldots, c_n)$, където $c_i = g_i(b_1, b_2, \ldots, b_m)$, $i = 1, 2, \ldots, n$. След това от таблицата на $f$ пресмятаме $f(c_1, c_2, \ldots, c_n)$ и това е търсената стойност за $h(b_1, b_2, \ldots, b_m)$.

Нека $g : J_3 \to J_3$ е функция, която независимо от аргумента $a$ има стойност 1, т.е. $g(a) = 1$. Прието е такива функции да наричаме *константи* и да ги именуваме със стойността, която приемат, т.е. $g(x) = \tilde{1}$ е константата 1.

В Табл. 5.2 е зададена функцията $f(x_1, x_2) : J_3^2 \to J_3$. Да образуваме суперпозицията $h(x_1, x_2) = f(g(x_1), x_2) = f(1, x_2)$. Получената функция е показана също в Табл. 5.2. Не е трудно да се забележи, че тя не зависи от променливата $x_1$ в най-естественото разбиране на тази дума – както и да заместваме $x_1$ със стойности от $J_3$ при фиксирана стойност на $x_2$, резултатът е един и същ. При суперпозиция на константа на мястото на променлива (казваме още „при заместване на променлива с константа“) се получава нова функция, която не зависи от заместената променлива, но за удобство понякога я разглеждаме като функция на същия брой

променливи, както и началната функция. Таблицата ѝ има същия брой редове, макар че всъщност може да се представи с таблица, $q$ пъти по-къса от началната.

***Дефиниция.*** Казваме, че $f : A^n \to A$ *не зависи съществено* от променливата $x_i$, ако $\forall k, j \in A$

$$f(x_1, \ldots, x_{i-1}, k, x_{i+1}, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, j, x_{i+1}, \ldots, x_n).$$

Казваме още, че $x_i$ е *фиктивна* променлива.

Към всяка функция на $n$ променливи $f(x_1, x_2, \ldots, x_n)$ можем да добавим фиктивна променлива $x_{n+1}$, като дефинираме нова функция

$$f'(x_1, x_2, \ldots, x_n, x_{n+1}) = f(x_1, x_2, \ldots, x_n).$$

Това ще доведе до $q$-кратно увеличаване на редовете на таблицата на $f$, за да получим таблица на $f'$, в която всяка стойност на $f$ ще се повтаря $q$ пъти.

В нашите разглеждания няма да различаваме функциите $f'$ и $f'' \in \mathcal{F}_q$, ако едната се получава от другата с добавяне на фиктивни променливи, независимо от това, че таблиците им са с различен брой редове. Когато се налага, ще разглеждаме всички функции в дадено крайно множество като функции на един и същ брой променливи.

Ще фиксираме една пълна наредба на векторите от $A^n$, която ще наречем стандартна, като използуваме релацията $\mathcal{R}_<$ между елементите на $A$, разглеждани като естествени числа. Векторът $(a_1, a_2, \ldots, a_n)$ е преди вектора $(b_1, b_2, \ldots, b_n)$ в наредбата, ако $a_1 = b_1, \ldots, a_{i-1} = b_{i-1}, a_i < b_i$ за някое $i = 1, 2, \ldots, n$. Такава наредба е известна като *лексикографска*. В Табл. 5.2 векторите от $J_3^2$ са стандартно (лексикографски) наредени. Броят на векторите в $A^n$ е $q^n$, така че след фиксиране на наредбата в $A^n$ всяка дискретна $q$-ична функция на $n$ променливи се представя само с втория стълб на таблицата си, който е вектор с $q^n$ елемента и различните функции имат различни стълбове. Следователно броят на всички $q$-ични функции на $n$ променливи е $|\mathcal{F}_q^n| = q^{q^n}$.

При достатъчно големи $q$ и $n$ размерът на таблицата на една $q$-ична функция е толкова голям, че работата с нея ще бъде затруднена. Затова е естествено да се търсят други начини за представяне на дискретните функции, при които се запазва възможността за ефективното им изчисляване.

***Дефиниция.*** Функция от вида $f(x_1, x_2, \ldots, x_n) = x_i$ наричаме *идентитет* или просто *променлива*. Съгласно въведената по-горе еквивалентност, различните идентитети представляват една и съща функция.



Нека $X = \{x_0, x_1, x_2, \ldots\}$ е изброимо множество, от което избираме променливите за всяка функция от $\mathcal{F}_q$. Нека $F = \{f_i | i \in I\} \subseteq \mathcal{F}_q$. Нека $\mathbb{Z}$ е крайна азбука и $\iota : N \to Z^+$ е биекция. По-нататък, когато пишем $f_i$ или $x_j$ имаме предвид думите $f\alpha$ и $x\beta$, такива че $\alpha = \iota(i)$, $\beta = \iota(j)$, $\alpha, \beta \in Z^+$. Нека $H = \{f, x, (, ), <\text{запетая}>\} \cup I$. Понятието *формула* над множеството от функции $F$ ще дефинираме като съвкупността от думи над азбуката $H$, които удовлетворяват следната

***Дефиниция.*** а) $\forall f_i \in F$ думата $f_i(x_1, x_2, \ldots, x_n) \in H^*$ е формула над $F$;

б) нека $f_i \in F$ е функция на $n$ променливи, а $\varphi_1, \varphi_2, \ldots, \varphi_n$ са формули над $F$ или променливи от $X$. Тогава думата $f_i(\varphi_1, \varphi_2, \ldots, \varphi_n) \in H^*$ е формула над $F$;

в) няма други формули над $F$ освен определените в (а) и получените индуктивно с помощта на операцията от (б).

Например, ако $f_1(x_1, x_2)$ и $f_2(x_1)$ са функции от $F$, то думите $f_2(x_1)$, $f_1(x_1, x_2)$, $f_1(x_3, x_2)$, $f_1(f_2(x_1), x_2)$ и $f_1(f_1(x_3, x_2), f_2(x_1))$ са формули над $F$.

***Дефиниция.*** Нека $F = \{f_i | i \in I\} \subseteq \mathcal{F}_q$ и $\Phi_F$ е множеството от формули над $F$. На всяка формула $\varphi \in \Phi_F$ съпоставяме функция $f \in \mathcal{F}_q$ (пишем $\varphi : f$), следвайки дефиницията на $\varphi$ по следния начин:

а) ако $\varphi$ е $f_i \in F$, то $\varphi : f = f_i$;

б) нека $f_i \in F$ е функция на $n$ променливи, а $\varphi_1, \varphi_2, \ldots, \varphi_n$ са формули над $F$ или променливи от $X$, на които са съпоставени функциите $g_1, g_2, \ldots, g_n \in \mathcal{F}_q$ (ако $\varphi_i = x_j$, то $g_i$ е идентитетът $x_j$). Тогава на $\varphi = f_i(\varphi_1, \varphi_2, \ldots, \varphi_n)$ съпоставяме суперпозицията $h = f_i(g_1, g_2, \ldots, g_n)$. Формулите $\varphi_1, \varphi_2, \ldots, \varphi_n$ наричаме *подформули* на $\varphi$.

Ще означаваме с $[F]$ множеството от всички функции, съпоставени на формулите от $\Phi_F$ и ще го наричаме *затваряне* (по отношение на суперпозицията) на $F$.

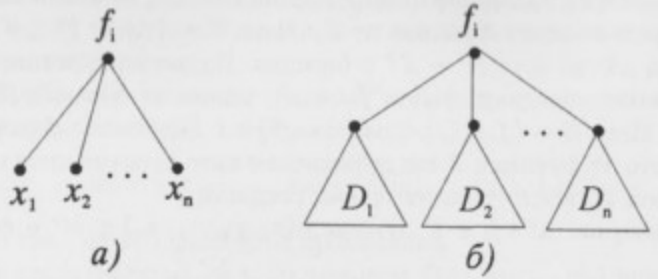
***Дефиниция.*** Функцията $\nu : \Phi_F \cup X \to N$ наричаме *дълбочина* на формула и определяме върху формулите от $\Phi_F$ и буквите на променливи така:

а) $\nu(x_j) = 0, \forall x_j \in X; \nu(f_i) = 1, \forall f_i \in F$;

б) ако $\varphi = f_i(\varphi_1, \varphi_2, \ldots, \varphi_n)$, то $\nu(\varphi) = \max_{i=1,2,\ldots,n} \nu(\varphi_i) + 1$.

Дълбочината на всяка подформула е по-малка от дълбочината на съответната ѝ формула.

За примери да разгледаме функциите: $MAX(x_1, x_2, \ldots, x_n)$, стойността на която е равна на най-големия ѝ аргумент и $MIN(x_1, x_2, \ldots, x_n)$, стойността на която е равна на стойността на най-малкия ѝ аргумент. Всъщност става дума за две изброими фамилии, съдържащи по една




Фигура 5.1: Формули, представени като дървета.

функция за всяко $n$. Очевидно за функцията $MAX$ имаме следното представяне с формула над $\max(x_1, x_2)$

$$\max(x_1, \max(x_2, \ldots, \max(x_{n-1}, x_n)\ldots)).$$

Аналогично $MIN$ се представя с формулата над $\min(x_1, x_2)$

$$\min(x_1, \min(x_2, \ldots, \min(x_{n-1}, x_n)\ldots)).$$

Разглежданите като операции функции $\max(x_1, x_2)$ и $\min(x_1, x_2)$ са асоциативни, затова редът, по който се прилагат при пресмятане на $MAX$ и $MIN$ е без значение. (В дадените по-горе формули е фиксирана една от многото възможности.) Това позволява да гледаме на $MAX$ и $MIN$ като на многократни операции и да пишем $MAX(x_1, x_2, \ldots, x_n) = MAX_{i=1,2,\ldots,n} x_i$, и $MIN(x_1, x_2, \ldots, x_n) = MIN_{i=1,2,\ldots,n} x_i$.

Ще дадем и алтернативна дефиниция на понятието формула, която често е по-полезна за илюстрация на свойствата на формулите и най-вече за теоретичната и приложна информатика.

***Дефиниция.*** Нека $F = \{f_i | i \in I\} \subseteq \mathcal{F}_q$ и $X = \{x_0, x_1, x_2, \ldots\}$.

а) всяко кореново дърво $D(V, E)$ с височина 1 (вж. Фиг. 5.1.а) и надписваща върховете му функция $t : V \to F \cup X$ такова, че коренът му е надписан с $f_i \in F$, а листата – с променливите $x_1, x_2, \ldots, x_n$ от $X$, е формула над $F$. Функцията от $\mathcal{F}_q$ съпоставена на тази формула $D$ е $f_i(x_1, x_2, \ldots, x_n)$, а дълбочината ѝ $\nu(D)$ е равна на височината на дървото, в случая $\nu(D) = h(D) = 1$;

б) нека $f_i(x_1, x_2, \ldots, x_n) \in F$, а $D_1(V_1, E_1), \ldots, D_n(V_n, E_n)$, $V_i \cap V_j = \emptyset, i \neq j$, са формули (съответно надписани по върховете коренови дървета) над $F$ или самостоятелни върхове, надписани с букви на променливи (т.е. тривиални коренови дървета), като $r_1, r_2, \ldots, r_n$ са съответните им корени, $t_1, t_2, \ldots, t_n$ – функциите надписващи върховете им, а $g_1, g_2, \ldots, g_n$ – съответните им функции от $\mathcal{F}_q$ ($g_i$ е идентитетът, ако $D_i$ е тривиално кореново дърво). Нека $r \notin V_1 \cup V_2 \cup \cdots V_n$. Тогава дървото $D(V_1 \cup \cdots \cup V_n \cup \{r\}, E_1 \cup \cdots \cup E_n \cup \{(r, r_1), \ldots, (r, r_n)\})$ от Фиг. 5.1.б с корен $r$, за което $t : V_1 \cup \cdots V_n \cup \{r\} \to F \cup X$ се определя като: $t(r) = f_i$ и $t(v) = t_k(v)$, ако $v \in V_k, 1 \leq k \leq n$, е също формула над $F$. Функцията, която ѝ съпоставяме, е суперпозицията $h = f_i(g_1, g_2, \ldots, g_n)$, а дълбочината ѝ $\nu(D)$ се определя от височината $h(D)$ и е очевидно $1 + \max_{i=1,\ldots,n} h(D_i) = 1 + \max_{i=1,\ldots,n} \nu(D_i)$.



Очевидно, двете дефиниции определят един и същ клас от обекти по два различни начина – чрез думи над крайна азбука или чрез коренови дървета с надписи по върховете. Понятието подформула във втория случай е тъждествено с понятието поддърво. Ще отбележим още, че листата на кореновите дървета, представящи формули, винаги са надписани с букви на променливи от $X$, а нелистата – с функции от съответното множество $F$ и това до голяма степен ги характеризира като дефиниращи формули.

Ще докажем няколко теореми, отнасящи се до трансформация на формули във формули при някои формални операции над формулите от зададено $\Phi_F$.

**Теорема 5.1.1** *Нека $\varphi \in \Phi_F$, $F = \{f_i | i \in I\} \subseteq \mathcal{F}_q$. Нека $G \subseteq \mathcal{F}_q$, $G = \{g_0, g_1, g_2, \ldots\}$ е такова, че $f_i$ и $g_i$ имат един и същ брой променливи, $i = 0, 1, 2, \ldots$. Операцията $\theta$, дефинирана $\forall \varphi \in \Phi_F$, се състои в замяна на всяка поддума $f_i$ в $\varphi$ с $g_i$. Тогава резултатът на операцията $\theta(\varphi)$ е формула над $G$, т.е. $\theta(\varphi) \in \Phi_G$.*

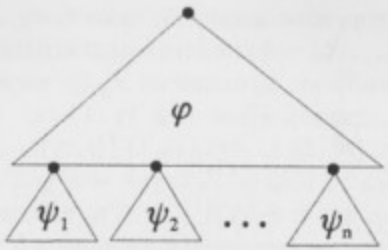
***Доказателство.*** С индукция по дълбочината $\nu$ на $\varphi$ ще докажем, че $\theta(\varphi) \in \Phi_G$.

а) Нека $\nu = 1$, т.е. $\varphi = f_i(x_1, \ldots, x_n)$. Тогава $\theta(\varphi) = \theta(f_i(x_1, \ldots, x_n)) = g_i(x_1, \ldots, x_n) \in \Phi_G$.

б) Да допуснем, че $\forall \varphi_i \in \Phi_F$, $\nu(\varphi_i) \leq k \Rightarrow \theta(\varphi_i) \in \Phi_G$.

в) Нека $\varphi \in \Phi_F$, $\nu(\varphi) = k + 1$. Следователно $\varphi = f_i(\varphi_1, \varphi_2, \ldots, \varphi_n)$, където $\varphi_i$ е формула с $\nu(\varphi_i) \leq k$ или променлива $x_j$. Ако $\varphi_i$ е формула, то от индукционното предположение $\theta(\varphi_i) \in \Phi_G$, а ако е променлива $x_j$, то $\theta(x_j) = x_j$. Следователно $\theta(\varphi_1), \theta(\varphi_2), \ldots, \theta(\varphi_n)$ са или формули над $G$ или променливи. Но $\theta(\varphi) = g_i(\theta(\varphi_1), \theta(\varphi_2), \ldots, \theta(\varphi_n))$ и следователно е формула над $G$. □

Ако формулата $\varphi$ е зададена във вид на кореново дърво, твърдението на теоремата е очевидно, защото трансформацията $\theta$ заменя всеки




Фигура 5.2: Заместване на формули във формули.

надпис на нелист $f_i$ с $g_i$ и полученото кореново дърво очевидно представлява формула над $G$.

**Теорема 5.1.2** *Нека $F = \{f_i | i \in I\} \subseteq \mathcal{F}_q$, $\varphi \in \Phi_F$ е формула, в която участват променливите $x_1, x_2, \ldots, x_n$ и $\psi_1, \psi_2, \ldots, \psi_n$ са формули от $\Phi_F$ или променливи. Операцията $\theta$ заменя във $\varphi$ всяко срещане на променливата $x_i$ с $\psi_i, i = 1, 2, \ldots, n$. Тогава $\theta(\varphi) \in \Phi_F$.*

***Доказателство.*** Отново ще приложим индукция по дълбочината на $\varphi$, за да докажем, че $\theta(\varphi) \in \Phi_F$.

а) Нека $\nu(\varphi) = 1$, т.е. $\varphi = f_i(x_1, \ldots, x_n)$. Тогава $\theta(\varphi) = f_i(\psi_1, \ldots, \psi_n)$, където $\psi_i$ са формули над $F$ или променливи и съгласно дефиницията $\theta(\varphi) \in \Phi_F$.

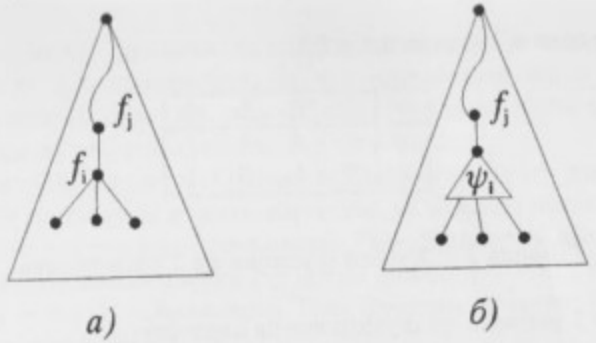
б) Да допуснем, че $\forall \varphi_i \in \Phi_F$, $\nu(\varphi_i) \leq k$, е в сила $\theta(\varphi_i) \in \Phi_F$.

в) Нека $\varphi \in \Phi_F, \nu(\varphi) = k + 1$. Следователно $\varphi = f_i(\varphi_1, \varphi_2, \ldots, \varphi_n)$, където $\varphi_i$ са или променливи или формули над $F$, $\nu(\varphi_i) \leq k$. Сега при $\varphi_i = x_j$ имаме $\theta(\varphi_i) = \psi_j \in \Phi_F$, ако $\psi_j \in \Phi_F$ и $\theta(\varphi_i) = x_l$, ако $\psi_j = x_l$. Ако $\varphi_i \in \Phi_F$, то $\nu(\varphi_i) \leq k$ и от индукционното предположение $\theta(\varphi_i) \in \Phi_F$ и следователно $\theta(\varphi_1), \ldots, \theta(\varphi_n)$ са формули над $F$ или променливи. Следователно $\theta(\varphi) = f_i(\theta(\varphi_1), \ldots, \theta(\varphi_n)) \in \Phi_F$. □

Така доказаната теорема разширява дефиницията на понятието формула и позволява да заместваме формули не само във функции от $F$, но и във вече построени формули от $\Phi_F$. Фиг. 5.2 илюстрира твърдението на теоремата чрез дърветата на съответните формули.

**Теорема 5.1.3** *Нека $F = \{f_i | i \in I\} \subseteq \mathcal{F}_q$, а $\varphi \in \Phi_F$ е формула за функцията $h$ . Нека $\psi_1, \psi_2, \ldots$ са формули от $\Phi_G$ такива, че $\psi_i$ е формула за $f_i$. Операцията $\theta$, дефинирана върху формулите от $\Phi_F$, заменя всяко $f_i$ с $\psi_i$. Тогава $\theta(\varphi) \in \Phi_G$ е формула за $h$.*






Фигура 5.3: Заместване на функции с формули.

Доказателството с индукция по дълбочината на $\varphi$ и с използване на Теорема 5.1.2 оставяме на читателя като упражнение. На Фиг. 5.3 е показана схематично замяната на върха, надписан с $f_i$, с формулата (дървото) $\psi_i$. Фигурата илюстрира частния случай, когато всяка от буквите на променливи е надпис на точно един лист на $\psi_i$. В общия случай една буква $x_k$ може да се среща многократно във формулата. В такъв случай копие от поддървото на върха с надпис $f_i$, съответно на $x_k$, ще бъде присъединено към всеки лист на $\psi_i$, надписан с $x_k$.

***Дефиниция.*** Формулите $\varphi_1$ и $\varphi_2$ са *еквивалентни*, ако им е съпоставена една и съща функция. Факта, че $\varphi_1$ е еквивалентна на $\varphi_2$, означаваме с $\varphi_1 = \varphi_2$.

Еквивалентността на формули можем да установим, като сравним таблиците на функциите, съпоставени на двете формули. Когато вече сме установили еквивалентността на две формули, можем да извършваме еквивалентни преобразувания, заместващи една формула с нейна еквивалентна и по този начин да установяваме еквивалентността и на други формули.

## 5.2 Булеви (двоични) функции

***Дефиниция.*** Функциите $\mathcal{F}_2 = \{f : J_2^n \to J_2 | n = 1, 2, \ldots\}$ наричаме *булеви* или *двоични*. Булевите функции на $n$ променливи означаваме с $\mathcal{F}_2^n$.

При стандартно подредени вектори от $J_2^n$, всяка булева функция на $n$ променливи се задава еднозначно с вектор-стълба си с $2^n$ елемента. Очевидно $|\mathcal{F}_2^n| = 2^{2^n}$. При $n = 1$ и $n = 2$ имаме съответно 4 и 16 функции,

които са дадени в Таблици 5.3 и 5.4:

| $x$ | $f_0$ | $f_1$ | $f_2$ | $f_3$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |

Таблица 5.3: Булеви функции на 1 променлива.

При $n = 1$ имената на функциите са следните:
- $f_0(x)$ – константата нула. Означаваме я с $\tilde{0}$ за разлика от $0 \in J_2$;
- $f_3(x)$ – константата единица. Означаваме я с $\tilde{1}$ за разлика от $1 \in J_2$;
- $f_1(x) = x$ – идентитетът;
- $f_2(x) = \overline{x}$ – *отрицание* на $x$.

Функциите $f_0$ и $f_3$ не зависят от променливата $x$, затова използваме означения, в които $x$ не участва.

| $xy$ | $f_0$ | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ | $f_6$ | $f_7$ | $f_8$ | $f_9$ | $f_{10}$ | $f_{11}$ | $f_{12}$ | $f_{13}$ | $f_{14}$ | $f_{15}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 01 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 10 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 11 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

Таблица 5.4: Булеви функции на 2 променливи.

При $n = 2$ за записване на функциите ще използваме инфиксния запис на двуместните операции. Функциите на 2 променливи са:

- $f_0(x, y) = \tilde{0}$. Тук константата $\tilde{0}$ е вече функция на две променливи, т.е. има стълб с височина 4, въпреки че не зависи нито от $x$, нито от $y$. Според направената уговорка ще я различаваме само по броя на променливите от константата $\tilde{0}$, разглеждана като функция на 1 променлива.

- $f_{15}(x, y) = \tilde{1}$. За константата $\tilde{1}$ са в сила съображенията, посочени по-горе за константата $\tilde{0}$.

- $f_3(x, y) = x$. Идентитетът от функциите на 1 променлива се появи като функция на 2 променливи. Очевидно тя не зависи от променливата $y$.

- $f_5(x, y) = y$. При наличие на две променливи има и втора форма на идентитет – функция, независеща от $x$.

- $f_{12}(x, y) = \overline{x}$ и $f_{10}(x, y) = \overline{y}$ са двете отрицания, разглеждани като функции на две променливи, независещи от едната си променлива.



Очевидно, всяка функция на $n$ променливи ще се появява и като функция на $n' > n$ променливи, ($n' - n$ променливи ще са фиктивни или несъществени), при това многократно – толкова пъти, по колкото начина можем да изберем фиктивните променливи.

Останалите функции от $\mathcal{F}_2^2$ зависят съществено и от двете си променливи. Ще ги означим и ще ги наричаме по следния начин:

- $f_1(x, y) = x \wedge y = xy$ – *конюнкция*. Тази функция е частен случай при $q = 2$ на функциите $\min(x_1, x_2)$ и $x_1.x_2 (\mathrm{mod}\, q)$;

- $f_7(x, y) = x \vee y$ – *дизюнкция*. Тази функция е частен случай при $q = 2$ на функцията $\max(x_1, x_2)$;

- $f_6(x, y) = x \oplus y$ – *събиране по модул 2*. Тази функция е частен случай при $q = 2$ на функцията $x_1 + x_2 (\mathrm{mod}\, q)$;

- $f_9(x, y) = x \equiv y$ – *еквивалентност*;

- $f_{13}(x, y) = x \to y$ – *импликация (от $x$ следва $y$)*;

- $f_{11}(x, y) = y \to x$ – обратната *импликация*(от $y$ следва $x$);

- $f_{14}(x, y) = x|y$ – *функция (черта) на Шефер*;

- $f_8(x, y) = x \downarrow y$ – *функция (стрелка) на Пирс*.

Функциите $f_2$ и $f_4$ нямат общоприето инфиксно записване.

При това записване остават в сила правилата от дефиницията на понятието формула. Вместо $f_7(f_1(x, f_{10}(x, y)), f_1(f_{12}(x, y), y))$ пишем инфиксния запис на същата формула $(x \wedge (\overline{y})) \vee ((\overline{x}) \wedge y) = (x(\overline{y})) \vee ((\overline{x})y)$. С въвеждането на приоритети на функциите този запис може да се опрости още. Общоприето е конюнкцията да има по-висок приоритет от дизюнкцията и събирането по модул 2, а отрицанието – по-висок приоритет от конюнкцията. Така формулата от горния пример става $x\overline{y} \vee \overline{x}y$.

Функциите $f_2$ и $f_4$ се представят с формулите $f_2 = \overline{x \to y}$ и $f_4 = \overline{y \to x}$. За да се убедим в това, е достатъчно да пресметнем функциите, съпоставени на тези формули и да сравним стълбовете им със стълбовете на $f_2$ и $f_4$.

За функциите на три и повече променливи инфиксното записване е невъзможно и няма друга подобна форма на запис. На практика за функциите на повече променливи най-често използваната форма за представяне са формулите над подходящо избрано множество. На този проблем ще се спрем в следващия раздел.

Някои важни свойства на често използваните двоични функции са събрани в следната

**Теорема 5.2.1** *В сила са следните:*

*а) комутативни свойства*

$$xy = yx, x \vee y = y \vee x, x \oplus y = y \oplus x;$$

*б) асоциативни свойства*

$(xy)z = x(yz), (x \vee y) \vee z = x \vee (y \vee z), (x \oplus y) \oplus z = x \oplus (y \oplus z);$

*в) дистрибутивни свойства*

$x(y \vee z) = xy \vee xz, x \vee yz = (x \vee y)(x \vee z), x(y \oplus z) = xy \oplus xz;$

*г) идемпотентни свойства*

$$xx = x, x \vee x = x \ (x \oplus x = \tilde{0});$$

*д) свойства на отрицанието*

$$x\overline{x} = \tilde{0}, x \vee \overline{x} = \tilde{1}, x \oplus \overline{x} = \tilde{1}, \overline{\overline{x}} = x;$$

*е) свойства на константите*

$$x\tilde{0} = \tilde{0}, x\tilde{1} = x, x \vee \tilde{0} = x, x \vee \tilde{1} = \tilde{1}, x \oplus \tilde{0} = x, x \oplus \tilde{1} = \overline{x};$$

*ж) закони на Де Морган*

$$\overline{x \vee y} = \overline{x} \wedge \overline{y}, \overline{x \wedge y} = \overline{x} \vee \overline{y}.$$

Ще илюстрираме произтичащия от дефиницията способ за доказателство на еквивалентност на формули, като покажем верността на първия от законите на Де Морган. Всички участващи в двете формули подформули са дадени в Таблица 5.5. От равенството на четвъртия и петия стълб следва верността на твърдението. Доказателството на останалите свойства оставяме на читателя за упражнение.

| $x$ | $y$ | $x \vee y$ | $\overline{x \vee y}$ | $\overline{x} \wedge \overline{y}$ | $\overline{x}$ | $\overline{y}$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |

Таблица 5.5: Доказателство на закона на Де Морган.

Ще използваме горните свойства, за да докажем някои други свойства на булевите функции чрез еквивалентни преобразувания. Ще опростим формулата $fx \vee f\overline{x}$, в която $f \in \mathcal{F}_2$. Последователно получаваме

$$\begin{aligned} & fx \vee f\overline{x} && = (\text{от дистрибутивния закон}) \\ = \ & f(x \vee \overline{x}) && = (\text{от свойствата на отрицанието}) \\ = \ & f\tilde{1} && = (\text{от свойствата на константата } \tilde{1}) \\ = \ & f \end{aligned}$$



Полученото свойство $fx \vee f\overline{x} = f$ наричаме *слепване* на $fx$ и $f\overline{x}$. С аналогичните разсъждения: $fg \vee f = f(g \vee \tilde{1}) = f\tilde{1} = f$ за произволни $f, g \in \mathcal{F}_2$ доказваме, че $fg \vee f = f$ – свойство, което наричаме *поглъщане* на $fg$ от $f$.

Наблюдателният читател навярно е забелязал съвпадението на означенията на конюнкцията, дизюнкцията и отрицанието с означенията на операциите в произволна булева алгебра. Това не е случайно. Да разгледаме тези функции като операции: конюнкцията и дизюнкцията от вида $\mathcal{F}_2 \times \mathcal{F}_2 \to \mathcal{F}_2$, а отрицанието от вида $\mathcal{F}_2 \to \mathcal{F}_2$. (Забележете разликата между булевите функции и съответните операции.) Резултатите от тези операции, извършени с операндите $f$ и $g$, се определят еднозначно и това са функциите, съпоставени на формулите $f \vee g, f \wedge g$ и $\overline{f}$. Комутативността, асоциативността и идемпотентността на функциите конюнкция и дизюнкция естествено се пренасят и върху съответните операции. Двата дистрибутивни закона водят до двете поглъщания, като едното вече доказахме, а другото се доказва аналогично. (Да си припомним принципа за двойственост!). И така $(\mathcal{F}_2; \vee, \wedge)$ е дистрибутивна решетка.

От свойствата на константите получаваме $f\tilde{0} = \tilde{0}, f \vee \tilde{1} = \tilde{1}, \forall f \in \mathcal{F}_2$. Следователно $\tilde{0}$ е нула, а $\tilde{1}$ – единица на решетката. От свойствата на отрицанието получаваме $f\overline{f} = \tilde{0}, f \vee \overline{f} = \tilde{1}, \forall f \in \mathcal{F}_2$. Следователно $\overline{f}$ е допълнение на $f$ в решетката, в сила са всички изисквания на дефиницията и $(\mathcal{F}_2; \vee, \wedge, \neg, \tilde{0}, \tilde{1})$ е булева алгебра.

В случая на булеви функции получаваме следния вариант на понятието фиктивна (несъществена променлива):

***Дефиниция.*** Променливата $x_i, 1 \leq i \leq n$ е фиктивна (несъществена) за функцията $f(x_1, x_2, \ldots, x_n) \in \mathcal{F}_2$, ако

$$f(x_1, \ldots, x_{i-1}, 0, x_{i+1} \ldots, x_n) = f(x_1, \ldots, x_{i-1}, 1, x_{i+1} \ldots, x_n).$$

Ще дефинираме по нов начин стандартното разполагане на векторите от $J_2^n$, като ще покажем, че то съвпада с лексикографската наредба.

***Дефиниция.*** а) Стандартното разполагане на векторите от $J_2^1$ е $\{(0), (1)\}$. Очевидно, то съвпада с лексикографската наредба на същите вектори.

б) Нека $\{\alpha_0, \alpha_1, \ldots, \alpha_{2^n-1}\}$ е стандартно разполагане на векторите от $J_2^n$ и да допуснем, че то съвпада с лексикографската наредба на същите вектори.

в) Стандартно разполагане на векторите от $J_2^{n+1}$ ще бъде

$$\{0\alpha_0, 0\alpha_1, \ldots, 0\alpha_{2^n-1}, 1\alpha_0, 1\alpha_1, \ldots, 1\alpha_{2^n-1}\}.$$

Лесно се вижда, че за произволни два вектора от първата или втората половина на това разполагане наредбата, съществуваща съгласно (б), се запазва, тъй като са разширени отляво с едно и също число. Но $0\alpha_{2^n-1}$ предхожда лексикографски $1\alpha_0$, защото се различават в първата си позиция и там $0\alpha_{2^n-1}$ има 0, а $1\alpha_0$ има 1.

Така стандартното разполагане на векторите от $J_2^2$ е $\{00, 01, 10, 11\}$, а на векторите от $J_2^3$ – $\{000, 001, 010, 011, 100, 101, 110, 111\}$. Редът, по който са изписани стълбовете на функции на една и две променливи в Табл. 5.3 и 5.4 по-горе, е всъщност стандартното разполагане на векторите от $J_2^2$ и $J_2^4$.

Асоциативността на дизюнкцията, конюнкцията и събирането по модул две ни позволяват да се освободим от скобите във формули, които са многократни дизюнкции, конюнкции и събирания по модул две. По аналогия с многократните събиране и умножение на числа ще използваме следните съкратени означения

$$\bigvee_{i=1}^{r} f_i = f_1 \vee f_2 \vee \cdots \vee f_r, \bigwedge_{i=1}^{r} f_i = f_1 \wedge f_2 \wedge \cdots \wedge f_r, \bigoplus_{i=1}^{r} f_i = f_1 \oplus f_2 \oplus \cdots \oplus f_r.$$

## 5.3 Пълни множества от функции

***Дефиниция.*** Множеството $F \subseteq \mathcal{F}_q$ е *пълно*, ако $[F] = \mathcal{F}_q$.

Очевидно $[\mathcal{F}_q] = \mathcal{F}_q$. Но съществуването на пълни множества, различни от $\mathcal{F}_q$, не е очевиден факт, а е изключително важно (особено ако са с неголям брой елементи), защото позволяват да се определят класове от формули, с които се оперира много по-удобно, отколкото със стълбовете на функциите.

Да се спрем първо на пълнотата на множества от булеви функции.

***Дефиниция.*** Функцията $f(x, \sigma) = x^\sigma$ дефинираме така

$$x^\sigma = \begin{cases} x & \text{ако } \sigma = 1 \\ \overline{x} & \text{ако } \sigma = 0 \end{cases}$$

**Лема 5.3.1** $x^\sigma = 1 \Leftrightarrow x = \sigma$.

***Доказателство.*** Достатъчно е да пресметнем стълба на $x^\sigma$ и да видим, че $x^\sigma = x \equiv \sigma$, което доказва твърдението. □

***Дефиниция.*** Формули от вида $x_{i_1}^{\sigma_1} x_{i_2}^{\sigma_2} \ldots x_{i_k}^{\sigma_k}$, в които $x_{i_j} \neq x_{i_l}, j \neq l$ наричаме *елементарни конюнкции*. При фиксирани $n$ променливи $x_1, x_2, \ldots, x_n$, елементарната конюнкция $x_1^{\sigma_1} x_2^{\sigma_2} \ldots x_n^{\sigma_n}$ наричаме *пълна елементарна конюнкция*. Тривиално следствие от Лема 5.3.1 е следната:



**Лема 5.3.2** $x_{i_1}^{\sigma_1} x_{i_2}^{\sigma_2} \ldots x_{i_k}^{\sigma_k} = 1 \Leftrightarrow x_{i_1} = \sigma_1, x_{i_2} = \sigma_2, \cdots, x_{i_k} = \sigma_k$.

**Теорема 5.3.1 (Разлагане на булева функция по $i$ променливи)** *Нека са избрани $i, 1 \leq i \leq n$ от променливите на функцията $f(x_1, x_2, \ldots, x_n) \in \mathcal{F}_2$. Без ограничение на общността, нека това са първите $i$ променливи. Тогава*

$$f(x_1, x_2, \ldots, x_n) = \bigvee_{\forall \sigma_1 \sigma_2 \ldots \sigma_i} x_1^{\sigma_1} x_2^{\sigma_2} \ldots x_i^{\sigma_i} f(\sigma_1, \sigma_2, \ldots, \sigma_i, x_{i+1}, \ldots, x_n).$$

***Доказателство.*** Да означим с $g(x_1, x_2, \ldots, x_n)$ функцията, определена от дясната част на равенството. Да пресметнем стойностите на функциите $f$ и $g$ за произволен вектор $(a_1, a_2, \ldots, a_n) \in J_2^n$. Вляво получаваме $f(a_1, a_2, \ldots, a_n)$. От Лема 5.3.2 следва, че от всички $2^i$ елементарни конюнкции $x_1^{\sigma_1} x_2^{\sigma_2} \ldots x_i^{\sigma_i}$, участващи в дясната част, само една има стойност 1 – тази, при която $\sigma_j = a_j, j = 1, 2, \ldots, i$. Останалите елементарни конюнкции имат стойност 0 и анулират съответните членове на многократната дизюнкция. Така за стойността на дясната част получаваме:

$$\begin{aligned} & g(a_1, a_2, \ldots, a_n) = \\ = \; & a_1^{a_1} a_2^{a_2} \ldots a_i^{a_i} f(a_1, a_2, \ldots, a_i, a_{i+1}, \ldots, a_n) \vee \tilde{0} = \\ = \; & \tilde{1} f(a_1, a_2, \ldots, a_i, a_{i+1}, \ldots, a_n) = f(a_1, a_2, \ldots, a_n). \end{aligned}$$

Следователно, функциите от двете части на равенството съвпадат. □

**Теорема 5.3.2 (Дж. Бул)** *Множеството $\{x \vee y, xy, \overline{x}\}$ е пълно.*

***Доказателство.*** Ще покажем, че $\forall f \in \mathcal{F}_2$ е в сила $f \in [\{x \vee y, xy, \overline{x}\}]$, т.е можем да представим $f$ с формула над $\{x \vee y, xy, \overline{x}\}$.

а) Нека $f = \tilde{0}$. Тогава $f(x) = x\overline{x}$ и следователно $f$ се представя с формула над $\{x \vee y, xy, \overline{x}\}$.

б) Нека $f \neq \tilde{0}$. Разлагаме $f(x_1, x_2 \ldots, x_n)$ по всичките $n$ променливи и получаваме

$$f(x_1, x_2, \ldots, x_n) = \bigvee_{\forall \sigma_1 \sigma_2 \ldots \sigma_n} x_1^{\sigma_1} x_2^{\sigma_2} \ldots x_n^{\sigma_n} f(\sigma_1, \sigma_2, \ldots, \sigma_n).$$

Ако $f(\sigma_1, \sigma_2, \ldots, \sigma_n) = 0$, съответният член в дясната част се анулира и съгласно свойствата, доказани по-горе, може да не участва във формулата. Ако пък $f(\sigma_1, \sigma_2, \ldots, \sigma_n) = 1$, то $x_1^{\sigma_1} x_2^{\sigma_2} \ldots x_n^{\sigma_n} f(\sigma_1, \sigma_2, \ldots, \sigma_n) =$

$x_1^{\sigma_1}x_2^{\sigma_2}\dots x_n^{\sigma_n}$. Така получаваме

$$f(x_1,x_2,\dots,x_n)=\bigvee_{\substack{\forall\sigma_1\sigma_2\dots\sigma_n\\ f(\sigma_1\sigma_2\dots\sigma_n)=1}} x_1^{\sigma_1}x_2^{\sigma_2}\dots x_n^{\sigma_n},$$

което е формула над $\{x\vee y, xy, \overline{x}\}$. □

Теоремата на Бул не само доказва пълнотата на множеството $\{x\vee y, xy, \overline{x}\}$, но ни дава в явен вид съответната формула. Когато $f\neq\tilde{0}$, формулата получена от Теоремата на Бул наричаме *Съвършена Дизюнктивна Нормална Форма (СъвДНФ)* на $f$. СъвДНФ на всяка различна от $\tilde{0}$ функция се състои само от пълни елементарни конюнкции. Формули, аналогични на СъвДНФ, в които могат да се съдържат и непълни елементарни конюнкции, наричаме *Дизюнктивни Нормални Форми (ДНФ)*. Константата $\tilde{0}$ няма нито една ДНФ. Всяка една от останалите булеви функции има една СъвДНФ, а може да има и повече ДНФ.

Като пример нека да построим СъвДНФ на дизюнкцията. Имаме три вектора – $(0,1),(1,0)$ и $(1,1)$, при които дизюнкцията има стойност 1. Затова $x\vee y = x^0y^1\vee x^1y^0\vee x^1y^1 = \overline{x}y\vee x\overline{y}\vee xy$. Забележете, че съгласно дефиницията $xy$ е СъвДНФ на конюнкцията.

За да построим и други пълни множества, ще използваме следната

**Теорема 5.3.3** *Нека $F\subseteq\mathcal{F}_2$ е пълно, $G\subseteq\mathcal{F}_2$ и $f_i\in[G], \forall f_i\in F$. Тогава и $G$ е пълно.*

***Доказателство.*** Всяка функция $h$ от $\mathcal{F}_2$ има формула $\varphi$ над $F$, защото $F$ е пълно. За да докажем, че $h\in[G]$ е достатъчно да заменим във $\varphi$ всяка функция $f_i$ от $F$ с формулата ѝ над $G$, да приложим Теорема 5.1.3 и да получим формула за $h$ над $G$. □

От доказаната теоремата следва, че:

а) $\{xy,\overline{x}\}$ е пълно, защото с формули над него можем да представим функциите на пълното множество $\{x\vee y, xy, \overline{x}\}$. Конюнкцията и отрицанието участват и в двете множества, затова ще бъде достатъчно да представим дизюнкцията чрез конюнкция и отрицание. От законите на Де Морган $\overline{x\vee y} = \overline{x}\,\overline{y}$. Като заместим двете страни в отрицанието и приложим закона за двойното отрицание, получаваме $x\vee y = \overline{\overline{x}\,\overline{y}}$, което е търсената формула за дизюнкцията.

б) Аналогично, с използване на другия закон на Де Морган, от пълнотата на $\{x\vee y, xy, \overline{x}\}$ получаваме и пълнотата на $\{x\vee y, \overline{x}\}$.

в) Множеството $\{\tilde{0},\tilde{1},xy,x\oplus y\}$ е пълно, защото всяка функция на пълното $\{xy,\overline{x}\}$ се представя с формула над него ($\overline{x} = x\oplus\tilde{1}$). Функцията



$\tilde{0}$ не е необходима за пълнотата на множеството (всъщност $\tilde{0} = \tilde{1}\oplus\tilde{1}$), но я добавяме, за да получим по-добра формула за самата $\tilde{0}$.

Да разгледаме формулите, които се получават от последното пълно множество. Тук полезна е аналогията с формули над събирането и умножението на числа, като събирането по модул 2 е аналог на събирането, а конюнкцията – на умножението. След разкриване на скобите (прилагане на дистрибутивния закон за конюнкцията по отношение на събирането по модул 2) и извършване на приведение по модул 2 получаваме представител на добре познат клас формули – многократна сума на едночлени, наричани *полиноми (многочлени)*. Получените в случая полиноми ще са многократна сума по модул 2 от елементарни конюнкции без отрицание. Наричаме ги *полиноми на Жегалкин*. При разкриването на скобите използваме идемпотентността на умножението $x = xx$, а при приведението факта, че $f\oplus f = \tilde{0}$. В резултат на тези операции всяка променлива участва в едночлен не повече от един път и никой едночлен не участва повече от един път в полинома на Жегалкин. В общия случай полиномът на Жегалкин на $n$ променливи има вида:

$$\begin{aligned} & f(x_1,x_2,\dots,x_n) = \\ = \;& a_0\oplus a_1x_1\oplus a_2x_2\oplus\dots\oplus a_nx_n\oplus \\ \oplus\;& a_{n+1}x_1x_2\oplus a_{n+2}x_1x_3\oplus\dots\oplus a_mx_{n-1}x_n\oplus \\ \oplus\;& a_{m+1}x_1x_2x_3\oplus\dots\oplus a_lx_{n-2}x_{n-1}x_n\oplus\dots\oplus a_Kx_1x_2\dots x_n \end{aligned}$$

т.е. сума по модул две на произволно подмножество елементарни конюнкции без отрицания. Коефициентите $a_i\in\{0,1\}$. Ако $a_i=0$, съответният едночлен се нулира и не участва в полинома, а ако $a_i=1$ – участва в полинома, като коефициентът не е нужно да се пише. В случай, че всички коефициенти са нули, полиномът на Жегалкин представяме с $\tilde{0}$.

От пълнотата на множеството $\{\tilde{0},\tilde{1},xy,x\oplus y\}$ следва, че всяка булева функция има полином на Жегалкин. Сега да намерим броя на различните полиноми на Жегалкин на $n$ променливи. В общия вид на полиномите на Жегалкин имаме $\binom{n}{0}=1$ коефициент пред едночлена от 0-ва степен (свободния член), $\binom{n}{1}=n$ коефициента пред едночлени от първа степен и т.н. Следователно, броят на свободните коефициенти е $\sum_{i=0}^{n}\binom{n}{i}=2^n$. Всеки от коефициентите може да приема стойности от $\{0,1\}$ и затова броят на полиномите на Жегалкин на $n$ променливи е $2^{2^n}$, колкото е и броят на всички функции на $n$ променливи. Следователно различните полиноми представят различни функции. Така доказахме следната

**Теорема 5.3.4** *Всяка булева функция има единствен полином на Жегалкин.*

За пример да намерим полинома на Жегалкин на дизюнкцията. Търсим полином от вида $a_0 \oplus a_1 x \oplus a_2 y \oplus a_3 xy = x \vee y$. Замествайки в горното равенство четирите възможни стойности за променливите $x$ и $y$, получаваме четири линейни уравнения в крайното поле с 2 елемента за четирите неизвестни коефициента:

$$\begin{aligned} a_0 \oplus a_1 0 \oplus a_2 0 \oplus a_3 0 &= 0 \\ a_0 \oplus a_1 1 \oplus a_2 0 \oplus a_3 0 &= 1 \\ a_0 \oplus a_1 0 \oplus a_2 1 \oplus a_3 0 &= 1 \\ a_0 \oplus a_1 1 \oplus a_2 1 \oplus a_3 1 &= 1 \end{aligned}$$

От първото уравнение получаваме $a_0 = 0$. Замествайки го във второто и третото, намираме $a_1 = a_2 = 1$. Четвъртото уравнение определя $a_3 = 1$. И така полиномът на Жегалкин на дизюнкцията е $x \oplus y \oplus xy$.

Не е трудно да се забележи че линейната система за произволна функция ще бъде в диагонален вид, ако уравненията се подредят по нарастване на броя на единиците на породилите ги вектори, а при равен брой единици по-напред поставяме това уравнение, най-дясната единица на чийто вектор е по-вляво. Такава линейна система, както видяхме, се решава лесно с последователно изключване на променливите.

Ето още един начин за намиране на полинома на Жегалкин на произволна булева функция $f \neq \tilde{0}$. В СъвДНФ на $f$ да заменим всяко $\overline{x_i}$ с $x_i + 1$ (еквивалентно преобразуване), а всяка дизюнкция – със събиране по модул 2. Ще покажем, че последната операция е допустима в конкретния случай. Всяка пълна елементарна конюнкция има стойност 1 точно върху един вектор от $J_2^n$, а различните пълни елементарни конюнкции имат стойност 1 върху различни вектори. Многократната дизюнкция на една 1 и няколко 0 дава стойност 1, както и многократното събиране по модул две на същите стойности. Да намерим и по този начин полинома на дизюнкцията.

$$\begin{aligned} & x \vee y = \overline{x}y \vee x\overline{y} \vee xy = (x \oplus 1)y \oplus x(y \oplus 1) \oplus xy = \\ = \; & xy \oplus y \oplus xy \oplus x \oplus xy = x \oplus y \oplus xy. \end{aligned}$$

В общия случай е в сила следният аналог на теоремата на Бул:

**Теорема 5.3.5** *Множеството от q-ични функции*

$$\{\tilde{0}, \tilde{1}, \ldots, \tilde{q-1}, \max(x_1, x_2), \min(x_1, x_2), x_1^{x_2}\}$$

*е пълно.*



***Доказателство.*** Аналогията с двоичния случай ни подсказва да разгледаме функциите от вида $MIN(x_1^{\sigma_1}, x_2^{\sigma_2}, \cdots, x_n^{\sigma_n})$, $(\sigma_1, \sigma_2, \ldots, \sigma_n) \in J_q^n$ – аналози на елементарните конюнкции. Очевидно, всяка такава функция има стойност $q-1$ т.с.т.к. $x_1 = \sigma_1, x_2 = \sigma_2, \cdots, x_n = \sigma_n$. Следователно $\forall a \in J_q^n \; MIN(a, x_1^{\sigma_1}, x_2^{\sigma_2}, \cdots, x_n^{\sigma_n}) = a$ т.с.т.к. $x_1 = \sigma_1, x_2 = \sigma_2, \cdots, x_n = \sigma_n$. С директна проверка се убеждаваме, че $\forall f \in \mathcal{F}_q^n$ е в сила представянето

$$f(x_1, \ldots, x_n) = MAX_{\forall \sigma_1, \ldots, \sigma_n}(MIN(f(\sigma_1, \ldots, \sigma_n), x_1^{\sigma_1}, \cdots, x_n^{\sigma_n}))$$

и теоремата е доказана, тъй като $MIN$ и $MAX$ на произволен краен брой аргументи се представят с формули над даденото множество.

## 5.4 Затворени множества от функции

Проблемът за съществуването на пълни множества беше решен в предния раздел чрез директна конструкция на пълно множество или чрез свеждане към известно пълно множество. От този подход не става ясно, възможно ли е да се определи ефективно пълнотата на зададено множество. Пряко отношение към проблема за пълнота има понятието, на което е посветен този раздел.

***Дефиниция.*** Множеството $F \subseteq \mathcal{F}_q$ е *затворено*, ако $[F] = F$.

Например, $\mathcal{F}_q$ е затворено, защото $[\mathcal{F}_q] = \mathcal{F}_q$. Затворени са и множествата булеви функции $\{\tilde{0}\}, \{\tilde{1}\}$ и $\{x, \overline{x}\}$.

Връзката между затвореност и пълнота е очевидна, защото не може да бъде пълно затвореното множество $F \subset \mathcal{F}_q$, а така също и никое негово подмножество.

**Лема 5.4.1** *Нека* $F, G \subseteq \mathcal{F}_q$.

*а)* $F \subseteq [F]$;
*б)* $F \subseteq G \Rightarrow [F] \subseteq [G]$;
*в)* $[F] \cup [G] \subseteq [F \cup G]$;
*г)* $[[F]] = [F]$.

***Доказателство.*** Твърденията от (а), (б) и (в) са очевидни. Ще докажем (г).

Първо ще покажем, че ако $\{f_0, f_1, f_2, \ldots\} \subseteq [F]$, то $[F \cup \{f_0, f_1, f_2, \ldots\}] = [F]$. Очевидно $F \subseteq F \cup \{f_0, f_1, f_2, \ldots\}$ и съгласно (б) $[F] \subseteq [F \cup \{f_0, f_1, f_2, \ldots\}]$. Ще докажем обратното. Нека $f \in [F \cup \{f_0, f_1, f_2, \ldots\}]$, т.е. $\exists \varphi = h(\varphi_1, \varphi_2, \ldots, \varphi_n)$ – формула за $f$ над $F \cup \{f_0, f_1, f_2, \ldots\}$.

Да допуснем, че $f \notin [F]$. Тогава $\exists i \in \{1,2,\ldots,n\}$, такова, че съпоставената на $\varphi_i$ функция $g_i \notin [F]$. Ако това не беше така и $h \in F$, то от дефиницията $\varphi$ е формула над $F$ и $f \in [F]$, което противоречи на допускането ни. Ако пък $h = f_j \in [F]$, то съществува формула $\psi_j$ за $h$ и от Теорема 5.1.3 отново получаваме формула $\psi_j(\varphi_1, \varphi_2, \ldots, \varphi_n)$ за $f$ над $F$, което е противоречие. Да отбележим, че $\nu(\varphi) > \nu(\varphi_i)$. И така за функцията $g_i$ получихме, че $g_i \in [F \cup \{f_0, f_1, f_2, \ldots\}]$, $g_i \notin [F]$. Но това бяха условията, на които отговаряше $f$. Затова с аналогични разсъждения получаваме, че във формулата $\varphi_i$ на $g_i$ над $F \cup \{f_0, f_1, f_2, \ldots\}$ ще се намери подформула $\varphi_{ij}$ над $F \cup \{f_0, f_1, f_2, \ldots\}$ такава, че съответната ѝ функция $g_{ij} \notin [F]$. При това $\nu(\varphi_i) > \nu(\varphi_{ij})$. Този процес може да продължи до безкрайност и в резултат ще получим безкрайната редица от формули $\varphi, \varphi_i, \varphi_{ij}, \ldots$ такива, че $\nu(\varphi) > \nu(\varphi_i) > \nu(\varphi_{ij}) > \ldots$. Но това противоречи на факта, че дълбочината на формула е цяло положително число. Следователно допускането не е вярно, $f \in [F]$, $[F \cup \{f_0, f_1, f_2, \ldots\}] \subseteq [F]$ и $[F \cup \{f_0, f_1, f_2, \ldots\}] = [F]$.

Нека сега $[F] = F \cup \{f_0, f_1, f_2, \ldots\}$. Тогава $[[F]] = [F \cup \{f_0, f_1, f_2, \ldots\}] = [F]$, което трябваше да докажем. □

**Теорема 5.4.1 (Критерий за затвореност)** *Нека $F \subseteq \mathcal{F}_q$ е такова, че*

*а) $f(x) = x \in F$;*

*б) $\forall f, g_1, \ldots, g_n \in F \Rightarrow h = f(g_1, \ldots, g_n) \in F$.*

*Тогава $F$ е затворено.*

***Доказателство.*** С индукция по дълбочината на всяка формула $\varphi$ над $F$ ще докажем, че съответната ѝ функция $f \in F$.

а) нека $\nu(\varphi) = 1$, тогава $f = f_i \in F$;

б) нека $\forall \varphi_i$ над $F$ и $\nu(\varphi_i) \leq k$ съответната функция $g_i \in F$. Нека $\varphi$ е формула над $F, \nu(\varphi) = k+1$. Следователно $\varphi = f_l(\varphi_1, \varphi_2, \ldots, \varphi_n)$, където $\varphi_j, j = 1, 2, \ldots, n$ са формули с $\nu(\varphi_j) \leq k$ или променливи. Ако $\varphi_j$ е формула, то от индукционното предположение съответната ѝ $g_j \in F$. Ако $\varphi_j$ е променлива, тогава $g_j = x \in F$ по условие. Но сега $f_l, g_1, g_2, \ldots, g_n \in F$ и по условие $h = f_l(g_1, g_2, \ldots, g_n) \in F$. □

***Дефиниция.*** Формулите от вида $f(g_1, g_2, \ldots, g_n)$ наричаме *сложни функции.*

Съвкупността от всички затворени множества булеви функции е детайлно изследвана от американския математик Е. Пост. Той доказа, че фамилията от затворените множества булеви функции е изброима безкрайна.



Ще разгледаме някои затворени множества от булеви функции, представляващи интерес за изследването на пълнотата на произволно множество булеви функции.

### 5.4.1 Булеви функции, запазващи константите

***Дефиниция.*** Казваме, че $f(x_1, x_2, \ldots, x_n) \in \mathcal{F}_2$ *запазва константата* $\tilde{c}$, ако $f(c, c, \ldots, c) = c$. Означаваме с $T_c$ множеството от функции, които запазват константата $\tilde{c}$, а с $T_c^n$ тези от тях, които са на $n$ променливи. Получаваме две множества функции, $T_0$ – функциите, запазващи нулата и $T_1$ – функциите, запазващи единицата.

Всяка функция, запазваща константата $\tilde{c}$ се дефинира по произволен начин върху векторите различни от $(c, c, \ldots, c)$ и следователно $|T_0^n| = |T_1^n| = 2^{2^n - 1}$. Булеви функции запазващи нулата са $xy, x \vee y$ и $x \oplus y$. Константата $\tilde{1}$ и $\overline{x}$ не запазват нулата. Булеви функции запазващи единицата са $xy$ и $x \vee y$. Константата $\tilde{0}, x \oplus y$ и $\overline{x}$ не запазват единицата. За нас важно е, че $T_0 \neq \mathcal{F}_2$ и $T_1 \neq \mathcal{F}_2$.

**Теорема 5.4.2** *Множеството $T_c$ от двоични функции, които запазват константата $\tilde{c}$ е затворено.*

***Доказателство.*** Прилагаме критерия за затвореност:

а) $f(x) = x \in T_c$, защото $f(c) = c$;

б) Нека $f, g_1, g_2, \ldots, g_n \in T_c$. Тогава $h = f(g_1, g_2, \ldots, g_n) \in T_c$, защото

$$h(c) = f(g_1(c), g_2(c), \ldots, g_n(c)) = f(c, c, \ldots, c) = c. \square$$

От написаното по-горе е ясно, че не е възможно едно пълно множество да се състои изцяло от функции, запазващи константа, т.е. за да е пълно едно множество, в него трябва непременно да има функция $\notin T_0$ и функция $\notin T_1$.

### 5.4.2 Самодвойнствени булеви функции

***Дефиниция.*** Нека $f(x_1, x_2, \ldots, x_n) \in \mathcal{F}_2$. Функцията $f^*(x_1, x_2, \ldots, x_n) = \overline{f}(\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n})$ наричаме *двойнствена* на $f$.

***Дефиниция.*** Векторите $\alpha(a_1, a_2, \ldots, a_n)$ и $\overline{\alpha}(\overline{a_1}, \overline{a_2}, \ldots, \overline{a_n}) \in J_2^n$ наричаме *противоположни.*

**Лема 5.4.2** *В стандартната наредба на векторите от $J_2^n$ противоположните вектори са симетрично разположени относно средата на таблицата.*

***Доказателство.*** Доказателството извършваме с индукция по $n$:

а) Нека $n = 1$. Стандартната наредба на $J_2^1$ е $\{0,1\}$, двата вектора са противоположни и са симетрично разположени относно средата.

б) Нека твърдението е в сила за $J_2^k$, т.е. $\forall \alpha \in J_2^k$ $\alpha$ и $\overline{\alpha}$ са симетрично разположени:

$$r\begin{bmatrix} & 0 & 0 & \dots & 0 \\ & & & \dots & \\ \alpha: & a_1 & a_2 & \dots & a_k \end{bmatrix}$$
$$\vdots$$
$$r\begin{bmatrix} \overline{\alpha}: & \overline{a_1} & \overline{a_2} & \dots & \overline{a_k} \\ & & & \dots & \\ & 1 & 1 & \dots & 1 \end{bmatrix}$$

в) Тогава за $J_2^{k+1}$ стандартното разположение има, съгласно конструкцията, вида

$$r\begin{bmatrix} & 0 & 0 & 0 & \dots & 0 \\ & & & & \dots & \\ 0\alpha: & 0 & a_1 & a_2 & \dots & a_k \end{bmatrix}$$
$$\vdots$$
$$r\begin{bmatrix} 0\overline{\alpha}: & 0 & \overline{a_1} & \overline{a_2} & \dots & \overline{a_k} \\ & & & & \dots & \\ & 0 & 1 & 1 & \dots & 1 \end{bmatrix}$$
$$r\begin{bmatrix} & 1 & 0 & 0 & \dots & 0 \\ & & & & \dots & \\ 1\alpha: & 1 & a_1 & a_2 & \dots & a_k \end{bmatrix}$$
$$\vdots$$
$$r\begin{bmatrix} 1\overline{\alpha}: & 1 & \overline{a_1} & \overline{a_2} & \dots & \overline{a_k} \\ & & & & \dots & \\ & 1 & 1 & 1 & \dots & 1 \end{bmatrix}$$

и очевидно противоположните $0\alpha$ и $1\overline{\alpha}$ са симетрично разположени относно средата, както и противоположните $0\overline{\alpha}$ и $1\alpha$. □

От това свойство получаваме следната процедура:

**Алгоритъм (за намиране двойствената $f^*$ на $f$).**

**Дадено: Стълбът на $f(x_1, x_2, \dots, x_n)$.**

**Резултат: Стълбът на $f^*(x_1, x_2, \dots, x_n)$.**

**Процедура:**

1) Инвертираме (правим отрицание на) всяка стойност в стълба на $f$.



2) Завъртаме симетрично относно средата получения в предната стъпка стълб. □

Забележете, че двете стъпки на алгоритъма са взаимно заменими, т.е. няма значение в какъв ред ще се прилагат. Като следствие от алгоритъма получаваме факта, че $\forall f \in \mathcal{F}_2$ е в сила $(f^*)^* = f$.

С прилагане на алгоритъма установяваме следните двойствености: $\bar{0}^* = \bar{1}$, $x^* = x$, $\overline{x}^* = \overline{x}$, $(xy)^* = x \vee y$, $(x \oplus y)^* = x \equiv y$, $(x|y)^* = x \downarrow y$, $(x \to y)^* = \overline{y \to x}$, $(y \to x)^* = \overline{x \to y}$.

**Лема 5.4.3 (Двойствена на сложна функция)** *Ако $h = f(g_1, g_2, \dots, g_n)$, то $h^* = f^*(g_1^*, g_2^*, \dots, g_n^*)$.*

***Доказателство.***

$$\begin{aligned} & h^*(x_1, x_2, \dots, x_m) = \\ = \; & \overline{f(g_1(\overline{x_1}, \dots, \overline{x_m}), \dots, g_n(\overline{x_1}, \dots, \overline{x_m}))} = \\ = \; & \overline{f(\overline{g_1}(\overline{x_1}, \dots, \overline{x_m}), \dots, \overline{g_n}(\overline{x_1}, \dots, \overline{x_m}))} = \\ = \; & \overline{f}(\overline{g_1^*}, \dots, \overline{g_n^*}) = f^*(g_1^*, g_2^*, \dots, g_n^*). \square \end{aligned}$$

Да означим с $F^* = \{f^* | f \in F \subseteq \mathcal{F}_2\}$.

**Теорема 5.4.3 (Принцип за двойственост)** *Нека $\varphi$ е формула над $F \subseteq \mathcal{F}_2$ за функцията $h$. Нека $\theta$ е трансформация, която заменя във $\varphi$ всяко срещане на $f_i \in F$ с $f_i^*$. Тогава $\theta(\varphi) = \varphi^* \in F^*$ е формула на функцията $h^*$.*

***Доказателство.*** Твърдението ще докажем с индукция по дълбочината на $\varphi$.

а) Нека $\nu(\varphi) = 1$, т.е. $\varphi = f_i = h$. Тогава $\varphi^* = f_i^* = h^*$.

б) Да допуснем, че твърдението е в сила $\forall \varphi_i$ над $F, \nu(\varphi_i) \leq k$.

в) Нека $\varphi$ е формула над $F$ за функцията $h, \nu(\varphi) = k+1$. Тогава $\varphi = f_i(\varphi_1, \varphi_2, \dots, \varphi_n)$, $\nu(\varphi_i) \leq k, i = 1, 2, \dots, n$ и $h = f_i(g_1, g_2, \dots, g_n)$, където $g_1, g_2, \dots, g_n$ са функциите, представяни от $\varphi_1, \varphi_2, \dots, \varphi_n$. От индукционното предположение $\varphi_1^* = \theta(\varphi_1), \varphi_2^* = \theta(\varphi_2), \dots, \varphi_n^* = \theta(\varphi_n)$ са формули на функциите $g_1^*, g_2^*, \dots, g_n^*$. Следователно $\theta(\varphi) = \varphi^* = f^*(\varphi_1^*, \dots, \varphi_n^*)$, която определя функцията $f^*(g_1^*, \dots, g_n^*) = h^*$ съгласно Лема 5.4.3. □

Директно от твърдението на Теоремата получаваме

**Следствие 5.4.1** *Нека $\varphi_1 = \varphi_2$. Тогава $\varphi_1^* = \varphi_2^*$.*

Това твърдение е обобщение на формулирания в 1.4. Принцип за двойственост за булевата алгебра $(\mathcal{F}_2; x \vee y, xy, \overline{x}, \tilde{0}, \tilde{1})$. Предлагаме на читателя да докаже сам този факт.

Принципът за двойственост позволява да се доказват лесно нови твърдения, изхождайки от доказани вече твърдения. Ето няколко примера.

Пример 1. Вече доказахме Закона на Де Морган $\overline{x \vee y} = \overline{x}\,\overline{y}$. С Принципа за двойственост от него получаваме директно $(\overline{x \vee y})^* = (\overline{x}\,\overline{y})^*$ или, $\overline{xy} = \overline{x} \vee \overline{y}$, което е другият закон на Де Морган.

Пример 2. Лесно се вижда, че всяка запазваща нулата булева функция има полином на Жегалкин без свободен член и обратно. От тук получаваме, че $[\{xy, x \oplus y\}] = T_0$. За $T_1$ е много трудно да се получи директно аналогичен резултат, защото за функциите от $T_1$ няма подобно представяне. Не е трудно да се установи обаче, че $T_0^* = T_1$. Сега от принципа за двойственост получаваме $[\{xy, x \oplus y\}^*] = T_0^*$, т.е. $[\{x \vee y, x \equiv y\}] = T_1$.

Пример 3. Нека $f \neq \tilde{1}$. Тогава $f^* \neq \tilde{0}$ и можем да я представим със СъвДНФ

$$f^*(x_1, x_2, \ldots, x_n) = \bigvee_{\substack{\forall \sigma_1 \sigma_2 \ldots \sigma_n \\ f^*(\sigma_1 \sigma_2 \ldots \sigma_n)=1}} x_1^{\sigma_1} x_2^{\sigma_2} \ldots x_n^{\sigma_n}.$$

Прилагаме принципа за двойственост и получаваме

$$f(x_1, x_2, \ldots, x_n) = \bigwedge_{\substack{\forall \sigma_1 \sigma_2 \ldots \sigma_n \\ f^*(\sigma_1 \sigma_2 \ldots \sigma_n)=1}} x_1^{\sigma_1} \vee x_2^{\sigma_2} \vee \ldots \vee x_n^{\sigma_n}.$$

Да направим в дясната част на горното равенство замяната $\sigma_i = \overline{\tau_i}, i = 1, 2, \ldots, n$. Получаваме

$$f(x_1, x_2, \ldots, x_n) = \bigwedge_{\substack{\forall \tau_1 \tau_2 \ldots \tau_n \\ f^*(\overline{\tau_1}, \overline{\tau_2}, \ldots, \overline{\tau_n})=1}} x_1^{\overline{\tau_1}} \vee x_2^{\overline{\tau_2}} \vee \ldots \vee x_n^{\overline{\tau_n}}$$

или, като приложим дефиницията на двойствена функция в описанието на многократната конюнкция,

$$f(x_1, x_2, \ldots, x_n) = \bigwedge_{\substack{\forall \tau_1 \tau_2 \ldots \tau_n \\ f(\tau_1, \tau_2, \ldots, \tau_n)=0}} x_1^{\overline{\tau_1}} \vee x_2^{\overline{\tau_2}} \vee \ldots \vee x_n^{\overline{\tau_n}}.$$

Така $\forall f \neq \tilde{1}$ получихме формула, двойствена на СъвДНФ, която наричаме *Съвършена Конюнктивна Нормална Форма (СъвКНФ)* на $f$.



Алгоритъмът за построяване на СъвКНФ е двойствен на този за построяване на СъвДНФ – за всеки вектор $(\tau_1, \tau_2, \ldots, \tau_n)$, за който $f$ е нула, пишем по една формула от вида $x_1^{\overline{\tau_1}} \vee x_2^{\overline{\tau_2}} \vee \ldots \vee x_n^{\overline{\tau_n}}$ (наричаме я *пълна елементарна дизюнкция*), като поставянето на отрицания е обратно на това при елементарните конюнкции. След това свързваме с многократна конюнкция елементарните дизюнкции. СъвКНФ се състои само от пълни елементарни дизюнкции. В общия случай, конюнкцията на *елементарни дизюнкции* (включително и не пълни) наричаме *Конюнктивна Нормална Форма* на $f$ (КНФ). Разбира се, функцията $\tilde{1}$ няма нито една КНФ.

За СъвКНФ на конюнкцията получаваме $xy = (x \vee y)(x \vee \overline{y})(\overline{x} \vee y)$. Дизюнкцията $x \vee y$ е в СъвКНФ.

***Дефиниция.*** Функцията $f(x_1, x_2, \ldots, x_n)$ наричаме *самодвойствена*, ако $f^* = f$.

От функциите на една и две променливи самодвойствени са само идентитетът и отрицанието. Да означим с $S$ множеството на самодвойствените функции, а с $S^n$ – тези от тях, които са на $n$ променливи.

**Лема 5.4.4** *Функцията $f \in S$ т.с.т.к. $\forall \alpha$ е в сила $f(\alpha) \neq f(\overline{\alpha})$.*

Доказателството е просто следствие от дефинициите, защото $f(\alpha) = f^*(\alpha) = \overline{f}(\overline{\alpha})$. Това ни дава възможност да намерим $|S^n| = 2^{2^{n-1}}$, защото всяка самодвойствена функция се дефинира свободно върху точно един от всеки два противоположни вектора, т.е. върху половината $2^{n-1}$ от всички $2^n$ вектори на $J_2^n$.

**Теорема 5.4.4** *Множеството $S$ е затворено.*

***Доказателство.*** Прилагаме критерия за затвореност.

а) $f(x) = x \in S$.

б) Нека $f, g_1, g_2, \ldots, g_n \in S$ и $h = f(g_1, g_2, \ldots, g_n)$. Тогава $h^* = f^*(g_1^*, g_2^*, \ldots, g_n^*)$ от Лема 5.4.3. От самодвойствеността на $f, g_1, g_2, \ldots, g_n$ получаваме $h^* = f(g_1, g_2, \ldots, g_n) = h$ и $h \in S$.

Следователно $S$ е затворено. □

### 5.4.3 Монотонни булеви функции

В 1.3. въведохме частична наредба в $J_2^n$, като с $\alpha \preceq \beta$ означавахме, че $\alpha(a_1, \ldots, a_n)$ предшества $\beta(b_1, \ldots, b_n)$. Лесно се вижда, че тя може да се дефинира като рефлексивно и транзитивно затваряне на релацията

$\mathcal{R}_{\prec\cdot} = \{\alpha \prec\cdot \beta | \exists i \in I_n, a_i < b_i$ и $a_j = b_j, j \neq i\}$, наречена „непосредствено предшестване". С други думи $\alpha \prec\cdot \beta$, ако $\alpha \preceq \beta$ и $\rho(\alpha, \beta) = 1$.

***Дефиниция.*** Булевата функция $f(x_1, x_2, \ldots, x_n)$ наричаме *монотонна*, ако $\forall \alpha, \beta \in J_2^n$, $\alpha \preceq \beta$ е в сила $f(\alpha) \leq f(\beta)$.

Функциите $x, x \vee y, xy, \tilde{0}, \tilde{1}$ са монотонни. Функцията $\overline{x}$ не е мотнотонно, защото $0 \preceq 1$, но $\overline{0} > \overline{1}$. Не е монотонна и функцията $x \oplus y$, защото $01 \preceq 11$, но $0 \oplus 1 > 1 \oplus 1$.

Означаваме с $M$ множеството на монотонните функции, а с $M^n$ – множеството на тези от тях, които са на $n$ променливи. Задачата за намиране броя на монотонните функции на $n$ променливи, поставена още в края на 19-ти век, не е решена задоволително. Известни са асимптотични оценки за $|M^n|$, но няма формула за точното му пресмятане. Проверката на монотонността на функция изисква доста усилия, тъй като по дефиниция трябва да бъдат сравнени стойностите на функцията за всяка двойка $\alpha \preceq \beta$. Затова и компютърното пресмятане на броя на монотонните функции не е лесно, дори за неголеми стойности на $n$. Следващото твърдение малко опростява нещата, но не дотолкова, че да позволи да бъде точно пресметнат $|M^n|$.

**Лема 5.4.5** *Нека $f \notin M, \alpha \preceq \beta$ и $f(\alpha) > f(\beta)$. Нека $\alpha = \alpha_1 \prec\cdot \alpha_2 \prec\cdot \cdots \prec\cdot \alpha_r = \beta$. Тогава $\exists i, 1 \leq i < r$ такова, че $f(\alpha_i) > f(\alpha_{i+1})$.*

***Доказателство.*** $f(\alpha) > f(\beta)$ означава $f(\alpha) = 1, f(\beta) = 0$. Да допуснем, че твърдението на лемата не е вярно, т.е. $\forall i, 1 \leq i < r$, $f(\alpha_i) \leq f(\alpha_{i+1})$. Разсъждавайки индуктивно, ще получим $1 = f(\alpha) = f(\alpha_1) = f(\alpha_2) = \cdots = f(\alpha_r) = f(\beta)$, което е в противоречие с $f(\beta) = 0$. □

От лемата се вижда, че за установяване принадлежността или непринадлежността на една функция към множеството $M$ е достатъчно да сравняваме само двойките $\alpha \prec\cdot \beta$. Ще оценим грубо отдолу $|M^n|$.

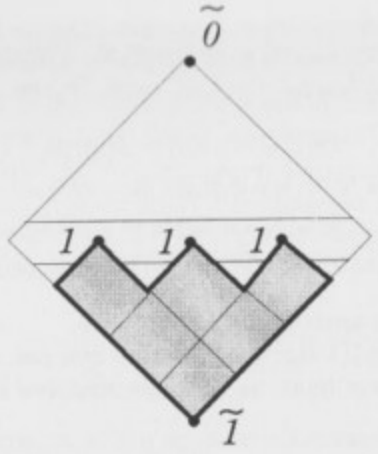
**Теорема 5.4.5** $|M^n| > 2^{\binom{n}{\lfloor \frac{n}{2} \rfloor}}$.

***Доказателство.*** $\binom{n}{\lfloor \frac{n}{2} \rfloor}$ е броят на векторите с тегло $\lfloor \frac{n}{2} \rfloor$ в $J_2^n$. Затова определяме множеството $M'$ от всички функции $f$ на $n$ променливи, за които

$$f(\alpha) = \begin{cases} 0 & wt(\alpha) < \lfloor \frac{n}{2} \rfloor \\ \text{произволна} & wt(\alpha) = \lfloor \frac{n}{2} \rfloor \\ 1 & wt(\alpha) > \lfloor \frac{n}{2} \rfloor \end{cases}$$

(вж. Фиг 5.4). Очевидно всяка такава функция е монотонна и затова $|M^n| > |M'| = 2^{\binom{n}{\lfloor \frac{n}{2} \rfloor}}$. □






Фигура 5.4: Монотонна функция.

**Теорема 5.4.6** *Множеството $M$ е затворено.*

***Доказателство.*** Прилагаме критерия за затвореност.

а) $f(x) = x \in M$.

б) Нека $f, g_1, g_2, \ldots, g_n \in M, \alpha \preceq \beta$ и нека $\tilde{\alpha} = (g_1(\alpha), g_2(\alpha), \ldots, g_n(\alpha))$, $\tilde{\beta} = (g_1(\beta), g_2(\beta), \ldots, g_n(\beta))$. От монотонността на $g_i$ следва, че $g_i(\alpha) \leq g_i(\beta), i = 1, 2, \ldots, n$, следователно $\tilde{\alpha} \preceq \tilde{\beta}$. Означаваме $h = f(g_1, g_2, \ldots, g_n)$. Сега $h(\alpha) = f(\tilde{\alpha}), h(\beta) = f(\tilde{\beta})$. Но $f \in M$ и от $\tilde{\alpha} \preceq \tilde{\beta}$ следва $f(\tilde{\alpha}) \leq f(\tilde{\beta})$. Следователно $h(\alpha) \leq h(\beta)$, $h \in M$ и $M$ е затворено. □

### 5.4.4 Линейни функции

***Дефиниция.*** Всяка функция $f(x_1, x_2, \ldots, x_n)$, с полином на Жегалкин от вида $a_0 \oplus a_1 x_1 \oplus \cdots \oplus a_n x_n$ наричаме *линейна*.

Очевидно $x, \overline{x} = x \oplus 1$ са линейни, $xy$ и $x \vee y = x \oplus y \oplus xy$ не са линейни. Означаваме с $L$ множеството на линейните функции, а с $L^n$ множеството на тези от тях, които са на $n$ променливи. Тъй като в общия вид на линейните полиноми на Жегалкин на $n$ променливи има $n + 1$ свободни коефициента, то $|L^n| = 2^{n+1}$.

**Теорема 5.4.7** *Множеството $L$ е затворено.*

***Доказателство.*** Прилагаме критерия за затвореност.

а) $f(x) = x \in L$.

б) Нека $f(x_1, x_2, \ldots, x_n = a_0 \oplus a_1 x_1 \oplus \cdots \oplus a_n x_n$, а $g_i(y_1, y_2, \ldots, y_m) = b_{i,0} \oplus b_{i,1} y_1 \oplus \cdots \oplus b_{i,m} y_m$, $i = 1, 2, \ldots, n$. Тогава

$$\begin{aligned} & h(y_1, y_2, \ldots, y_m) = f(g_1, g_2, \ldots, g_n) = \\ = \quad & a_0 \oplus a_1(b_{1,0} \oplus b_{1,1} y_1 \oplus \cdots \oplus b_{1,m} y_m) \oplus \cdots \oplus \\ & \oplus a_n(b_{n,0} \oplus b_{n,1} y_1 \oplus \cdots \oplus b_{n,m} y_m) = \\ = \quad & (a_0 \oplus \bigoplus_{i=1}^{n} a_i b_{i,0}) \oplus (\bigoplus_{i=1}^{n} a_i b_{i,1}) y_1 \oplus \cdots \oplus (\bigoplus_{i=1}^{n} a_i b_{i,m}) y_m \in L. \end{aligned}$$

Следователно $L$ е затворено. □

Очевидно $L = [\{\tilde{1}, \tilde{0}, x \oplus y\}]$, като и тук сме добавили без да е необходимо константата нула, за да я представяме по естествен начин, а не като $\tilde{1} \oplus \tilde{1}$.

## 5.5 Критерий за пълнота на множество булеви функции

Както забелязахме в предния раздел, никое пълно множество $F$ от булеви функции не може да бъде подмножество на затворено множество, не съвпадащо с $\mathcal{F}_2$. Изненадващ е фактът, че при наличието на изброима безкрайна фамилия от затворени множества, за пълнотата на $F$ е достатъчно то да не е подмножество само на множествата $T_0, T_1, S, M$ и $L$. Този силен резултат беше доказан по различен начин от Е. Пост и С. В. Яблонский.

**Теорема 5.5.1 (Е. Пост-С. В. Яблонский)** *Нека $F \subseteq \mathcal{F}_2$. Множеството $F$ е пълно т.с.т.к. не е подмножество на нито едно от множествата $T_0, T_1, S, M$ и $L$.*

***Доказателство.*** 1) Нека $F$ е пълно и $K \in \{T_0, T_1, S, M, L\}$. Да допуснем, че $F \subseteq K$. Но тогава от Лема 5.4.1.6 $[F] \subseteq [K]$. От пълнотата на $F$ следва $[F] = \mathcal{F}_2$, а от затвореността на $K$ – $[K] = K \neq \mathcal{F}_2$. Получаваме противоречие, следователно $F \not\subseteq K, \forall K \in \{T_0, T_1, S, M, L\}$.

2) Нека $F \not\subseteq K, \forall K \in \{T_0, T_1, S, M, L\}$. Тогава $\exists F' = \{f_0, f_1, f_s, f_m, f_l\} \subseteq F$, такова, че $f_0 \notin T_0$, $f_1 \notin T_1$, $f_s \notin S$, $f_m \notin M$, $f_l \notin L$. (Не е задължително петте функции да са различни.) Ще покажем, че $\{xy, \overline{x}\} \subseteq [F']$.

Функциите $g_0(x) = f_0(x, x, \ldots, x)$ и $g_1(x) = f_1(x, x, \ldots, x)$ са формули над $F'$, защото във функции от $F'$ сме заместили променлива. $g_0(0) = f_0(0, 0, \ldots, 0) = 1$, защото $f_0 \notin T_0$. Аналогично $g_1(1) = f_1(1, 1, \ldots, 1) = 0$, защото $f_1 \notin T_1$. За $g_0(1)$ и $g_1(0)$ имаме четири възможности:

- $g_0(1) = 0, g_1(0) = 0 \Rightarrow g_0(x) = \overline{x}, g_1(x) = \tilde{0}$ и $g_0(g_1(x)) = \overline{\tilde{0}} = \tilde{1}$;



- $g_0(1) = 0, g_1(0) = 1 \Rightarrow g_0(x) = \overline{x}, g_1(x) = \overline{x}$;
- $g_0(1) = 1, g_1(0) = 0 \Rightarrow g_0(x) = \tilde{1}, g_1(x) = \tilde{0}$;
- $g_0(1) = 1, g_1(0) = 1 \Rightarrow g_0(x) = \tilde{1}, g_1(x) = \overline{x}$ и $g_1(g_0(x)) = \overline{\tilde{1}} = \tilde{0}$.

В три от случаите построихме двете константи. Да разгледаме случая, при който получихме само отрицанието. Функцията $f_s \notin S$, следователно $\exists \alpha(a_1, a_2, \ldots, a_n), f_s(\alpha) = f_s(\overline{\alpha})$. Построяваме като формула над $F'$ функцията $g_s(x) = f_s(x^{a_1}, x^{a_2}, \ldots, x^{a_n})$. (Забележете, че $x^{a_i}$ е $x$ или $\overline{x}$, а отрицанието е вече построено.) Сега

$$\begin{aligned} g_s(0) = f_s(0^{a_1}, 0^{a_2}, \ldots, 0^{a_n}) &= f_s(\overline{a_1}, \overline{a_2}, \ldots, \overline{a_n}) = f(\overline{\alpha}) = \\ = f(\alpha) = f_s(a_1, a_2, \ldots, a_n) &= f_s(1^{a_1}, 1^{a_2}, \ldots, 1^{a_n}) = g_s(1). \end{aligned}$$

Следователно $g_s$ е константа, а $\overline{g_s}$ е другата константа. Така с формули над $f_0, f_1$ и $f_s$ успяхме да построим във всички случаи двете константи.

Функцията $f_m \notin M$. Следователно $\exists \alpha \prec \cdot \beta, f_m(\alpha) = 1, f_m(\beta) = 0$. Нека $\alpha = (a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_n)$ и $\beta = (a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_n)$. С помощта на построените вече константи образуваме като формула над $F'$ функцията $g_m(x) = f_m(a_1, \ldots, a_{i-1}, x, a_{i+1}, \ldots, a_n)$. Очевидно $g_m(0) = f_m(\alpha) = 1$ и $g_m(1) = f_m(\beta) = 0$. Следователно $g_m = \overline{x}$. Така вече имаме константите и отрицанието, представени с формули над $F'$.

Нека $xyz_1 \cdots z_k$ е най-късият нелинеен член в полинома на Жегалкин на нелинейната функция $f_l$. Да заместим в $f_l$ променливите $z_1, \ldots, z_k$ с 1, а всички останали променливи (без $x$ и $y$) с 0. Получаваме функцията $g_l(x, y)$ като формула над $F'$. Полиномът ѝ на Жегалкин ще има вида $xy \oplus ax \oplus by \oplus c$. Едночленът $xy$ се получава от най-късия нелинеен член и не може да бъде унищожен от друг подобен при събирането по модул 2, защото при избраното заместване с константи всеки друг нелинеен ще се анулира. Да направим в $g_l$ смяната на променливите $x = u \oplus b, y = v \oplus a$. Забележете, че $u \oplus b$ е $u$ или $\overline{u}$, а $v \oplus a$ – $v$ или $\overline{v}$. Следователно, получената функция $h_l(u, v)$ ще бъде формула над $F'$ с полином на Жегалкин

$$\begin{aligned} h_l(u, v) &= (u \oplus b)(v \oplus a) \oplus a(u \oplus b) \oplus b(v \oplus a) \oplus c = \\ &= uv \oplus au \oplus bv \oplus ab \oplus au \oplus ab \oplus bv \oplus ab \oplus c = \\ &= uv \oplus ab \oplus c = uv \oplus d, \end{aligned}$$

където с $d$ сме означили $ab \oplus c$. Ако $d = 0$, то $h_l(u, v)$ е конюнкцията, в противен случай конюнкцията построяваме като $\overline{h_l(u, v)}$.

И така $\{xy, \overline{x}\} \subseteq [F'] \subseteq [F]$. Но $\{xy, \overline{x}\}$ е пълно и съгласно Теорема 5.3.3 и $F$ е пълно. □

Доказаното достатъчно условие за пълнота на множество от булеви функции, наричано още *Критерий за пълнота* или *Критерий на Пост-Яблонский* ще приложим, за да установим още веднъж пълнотата на множеството $\{x \vee y, xy, \overline{x}\}$. В таблица (вж. Таблица 5.6), стълбовете на която са надписани с $T_0, T_1, S, M$ и $L$, отбелязваме непринадлежността на интересуващите ни функции към всяко от затворените множества със знак в съответния ред и стълб.

| | $T_0$ | $T_1$ | $S$ | $M$ | $L$ |
|---|---|---|---|---|---|
| $xy$ | | | * | | * |
| $x \vee y$ | | | * | | * |
| $\overline{x}$ | * | * | | * | |
| $\tilde{0}$ | | * | * | | |
| $\tilde{1}$ | * | | * | | |
| $x \oplus y \oplus z$ | | | | * | |
| $xy \oplus xz \oplus yz$ | | | | | * |

Таблица 5.6: Тест за пълнота и предпълнота.

От таблицата се вижда, че $\{x \vee y, xy, \overline{x}\}$ е пълно множество, защото съдържа поне по една функция, непринадлежаща на всяко едно от петте затворени множества. От таблицата се вижда, че $\{xy, \overline{x}\}$ и $\{x \vee y, \overline{x}\}$ също са пълни.

***Дефиниция.*** Пълно множество, което е минимално по включване с това свойство, наричаме *базис*.

Множествата $\{xy, \overline{x}\}$ и $\{x \vee y, \overline{x}\}$ са базиси.

***Дефиниция.*** Булевата функция $f$ наричаме *Шеферова*, ако $[\{f\}] = \mathcal{F}_2$, т.е. $f$ сама образува пълно множество. Съгласно Теоремата на Пост-Яблонский това означава, че $f \notin T_0, f \notin T_1, f \notin S, f \notin M, f \notin L$.

**Теорема 5.5.2 (Критерий за Шеферовост)** *Ако $f \in \mathcal{F}_2, f \notin T_0, f \notin T_1, f \notin S$, то $f$ е Шеферова.*

***Доказателство.*** От $f \notin T_0 \Rightarrow f(\tilde{0}) = 1$. От $f \notin T_1 \Rightarrow f(\tilde{1}) = 0$. Тогава $f \notin M$, защото $\tilde{0} \preceq \tilde{1}$, а $f(\tilde{0}) > f(\tilde{1})$.

Да допуснем, че $f \in L$ и $f = a_0 \oplus x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k}$. От $f(\tilde{0}) = 1$ получаваме $a_0 = 1$ и $f = 1 \oplus x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k}$. От $f(\tilde{1}) = 0$ пък следва, че $k$ е нечетно. Затова

$$\begin{aligned} f^* &= 1 \oplus (1 \oplus (x_{i_1} \oplus 1) \oplus (x_{i_2} \oplus 1) \oplus \cdots \oplus (x_{i_k} \oplus 1)) = \\ &= 1 \oplus x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k} \oplus (k+1) (\mathrm{mod}\, 2) = \\ &= 1 \oplus x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k} = f \end{aligned}$$



Но това противоречи на $f \notin S$. Следователно допускането е невярно, $f \notin L$ и значи $f$ е Шеферова. □

Като последно приложение на Теоремата за пълнота ще дадем отговор на въпроса по какво се отличават множествата $T_0, T_1, S, M, L$ от останалите затворени множества.

***Дефиниция.*** Множеството $F \subset \mathcal{F}_2$ наричаме *предпълно*, ако

а) $[F] \neq \mathcal{F}_2$;

б) $\forall f \notin F, [F \cup \{f\}] = \mathcal{F}_2$.

**Теорема 5.5.3** *Множествата $T_0, T_1, S, M, L$ и само те са предпълни в $\mathcal{F}_2$.*

***Доказателство.*** 1) Ще докажем, че $T_0, T_1, S, M, L$ са предпълни.

а) В предния раздел за всяко от петте множества видяхме, че е затворено и не съвпада с $\mathcal{F}_2$. Следователно $[K] \neq \mathcal{F}_2, \forall K \in \{T_0, T_1, S, M, L\}$.

б) Второто изискване на дефиницията ще покажем, като $\forall K \in \{T_0, T_1, S, M, L\}$ покажем, че съдържа поне по една функция непринадлежаща на всяко от останалите четири множества. Тогава $\forall f \notin K$ множеството $K \cup \{f\}$ ще съдържа по една функция, непринадлежаща на всяко от петте множества и съгласно Критерия за пълнота $[K \cup \{f\}] = \mathcal{F}_2$. В Таблица 5.6 са дадени булеви функции, които са достатъчни за доказателство по горната схема, като за всяка от функциите в таблицата с * са отбелязани множествата, на които функцията не принадлежи.

Функциите $\tilde{0}, x \oplus y \oplus z, xy \oplus xz \oplus yz \in T_0, \tilde{0} \notin T_1$, $\tilde{0} \notin S$, $x \oplus y \oplus z \notin M$, $xy \oplus xz \oplus yz \notin L$. Следователно $\forall f \notin T_0$, $[\{f, \tilde{0}, x \oplus y \oplus z, xy \oplus xz \oplus yz\}] = \mathcal{F}_2$ и следователно $T_0$ е предпълно.

Функциите $\tilde{1}, x \oplus y \oplus z, xy \oplus xz \oplus yz \in T_1, \tilde{1} \notin T_0$, $\tilde{1} \notin S$, $x \oplus y \oplus z \notin M$, $xy \oplus xz \oplus yz \notin L$. Следователно $\forall f \notin T_1$, $[\{f, \tilde{1}, x \oplus y \oplus z, xy \oplus xz \oplus yz\}] = \mathcal{F}_2$ и следователно $T_1$ е предпълно.

Функциите $\overline{x}, xy \oplus xz \oplus yz \in S, \overline{x} \notin T_0 \cup T_1 \cup M$, $xy \oplus xz \oplus yz \notin L$. Следователно $\forall f \notin S, [\{f, \overline{x}, xy \oplus xz \oplus yz\}] = \mathcal{F}_2$ и следователно $S$ е предпълно.

Функциите $\tilde{0}, \tilde{1}, xy \in M$, $\tilde{0} \notin T_1 \cup S$, $\tilde{1} \notin T_0$, $xy \notin L$. Следователно $\forall f \notin M, [\{f, \tilde{0}, \tilde{1}, xy\}] = \mathcal{F}_2$ и следователно $M$ е предпълно.

Функциите $\tilde{0}, \overline{x} \in L$, $\tilde{0} \notin T_1 \cup S$, $\overline{x} \notin T_0 \cup M$. Следователно $\forall f \notin L, [\{f, \tilde{0}, \overline{x}\}] = \mathcal{F}_2$ и следователно $L$ е предпълно.

2) Сега ще покажем, че няма други предпълни множества. Да допуснем противното и нека $H \subseteq \mathcal{F}_2$ е предпълно и различно от $T_0, T_1, S, M$ и $L$.

Не е възможно $K \subset H, K \in \{T_0, T_1, S, M, L\}$, защото тогава $\exists f \in H, f \notin K$ и $[K \cup \{f\}] = \mathcal{F}_2$ от предпълнотата на $K$, но $K \cup \{f\} \subseteq H$ и следователно $[K \cup \{f\}] \subseteq [H] \neq \mathcal{F}_2$ (от предпълнотата на $H$), което е противоречие.

Не е възможно $H \subset K \in \{T_0, T_1, S, M, L\}$, защото тогава $\exists f \notin H, f \in K$ и от предпълнотата на $H$ следва $[H \cup \{f\}] = \mathcal{F}_2$, което е противоречие с $H \cup \{f\} \subseteq K$ и следователно $[H \cup \{f\}] \subseteq [K] = K \neq \mathcal{F}_2$.

От това следва, че ако $H$ е предпълно, то $\exists F' = \{f_0, f_1, f_s, f_m, f_l,\} \subseteq F$, такова, че $f_0 \notin T_0$, $f_1 \notin T_1$, $f_s \notin S$, $f_m \notin M$, $f_l \notin L$ (не е задължително петте функции да са различни). Тогава от Теоремата на Пост-Яблонский това води до пълнотата на $H$, което е отново противоречие.

Тъй като други възможности за взаимното положение на $H$ с множествата $T_0, T_1, S, M, L$ не съществуват, то няма други предпълни множества. □

## 5.6 Задача за минималните покрития

В този раздел ще покажем възможностите на монотонните булеви функции за получаване на по-прегледно решение на трудна комбинаторна задача.

***Дефиниция.*** Нека $A = \{a_1, a_2, \ldots, a_n\}$ е крайно множество, $\mathcal{P} = \{S_1, S_2, \ldots, S_m\}$ е също крайно множество, а $T = ||b_{ij}||_{m \times n}$ е матрица, в която $b_{ij} \in \{0, 1\}$. Казваме, че $S_i \in \mathcal{P}$ *покрива* $a_j \in A$ т.с.т.к. $b_{ij} = 1$. Казваме, че $\mathcal{P}$ е *покритие* на $A$, ако $\forall a_j \in A\ \exists S_i \in \mathcal{P}$, такова че $S_i$ покрива $a_j$. Казваме, че $\mathcal{P}$ е *минимално покритие* на $A$, ако $\mathcal{P}$ е покритие, но никое $\mathcal{P}' \subset \mathcal{P}$ не е покритие на $A$.

Например, нека $A = \{T_0, T_1, S, M, L\}$, а $\mathcal{P} = \{\tilde{1}, \overline{x}, xy, x \vee y, x \oplus y\}$. Казваме, че $f \in \mathcal{P}$ покрива $K \in A$, ако $f \notin K$. Получаваме матрицата на покритие от Таблица 5.7.

| | $T_0$ | $T_1$ | $S$ | $M$ | $L$ |
|---|---|---|---|---|---|
| $\tilde{1}$ | 1 | 0 | 1 | 0 | 0 |
| $\overline{x}$ | 1 | 1 | 0 | 1 | 0 |
| $xy$ | 0 | 0 | 1 | 0 | 1 |
| $x \vee y$ | 0 | 0 | 1 | 0 | 1 |
| $x \oplus y$ | 0 | 1 | 1 | 1 | 0 |

Таблица 5.7: Матрица на покритие.

Очевидно $\mathcal{P}$ покрива $A$ и в конкретния случай това е еквивалентно



на твърдението, че $\mathcal{P}$ е пълно множество. Но $\mathcal{P}$ не е минимално покритие на $A$, защото $\mathcal{P}' = \{\overline{x}, xy, x \vee y\}$ е също покритие (пълно множество).

Следната задача наричаме *задача за минималните покрития*: Дадени са $A, \mathcal{P}$ и $T$, така че $T$ задава покритие на $A$ от $\mathcal{P}$. Да се намерят всички подмножества на $\mathcal{P}$, които са минимални покрития на $A$.

В нашия пример задачата за минималните покрития се формулира така: дадено е пълно множество от булеви функции. Да се намерят всички базиси, съдържащи се в това множество.

Тъй като се търсят подмножества на $\mathcal{P}$, $|\mathcal{P}| = m$, да представим всяко подмножество $\mathcal{P}'$ на $\mathcal{P}$ с $m$-мерен двоичен характеристичен вектор $\alpha_{\mathcal{P}'}(\sigma_1, \sigma_2, \ldots, \sigma_m) \in J_2^m$, където $\sigma_i = 1$ т.с.т.к. $S_i \in \mathcal{P}', i = 1, 2, \ldots, m$. Да образуваме, в съответствие с матрицата $T$, булева функция $f(x_1, x_2, \ldots, x_m) = D_1 \wedge D_2 \wedge \cdots \wedge D_m$, където $D_i = x_{i_1} \vee x_{i_2} \vee \cdots \vee x_{i_r}$, $x_{i_j}$ участва в $D_i$ т.с.т.к. $b_{i_j i} = 1$, т.е. $S_{i_j}$ покрива $a_i$.

За разглеждания пример получаваме функцията

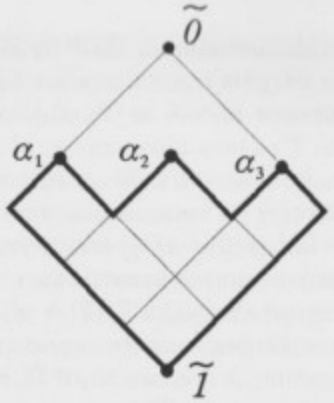
$$\begin{aligned} & f(x_1, x_2, x_3, x_4, x_5) = \\ = \ & (x_1 \vee x_2)(x_2 \vee x_5)(x_1 \vee x_3 \vee x_4 \vee x_5)(x_2 \vee x_5)(x_3 \vee x_4) = \\ = \ & (x_1 \vee x_2)(x_2 \vee x_5)(x_1 \vee x_3 \vee x_4 \vee x_5)(x_3 \vee x_4). \end{aligned}$$

**Теорема 5.6.1** *Нека $f(x_1, x_2, \ldots, x_m) = D_1 \wedge D_2 \wedge \cdots \wedge D_m$ е булевата функция за таблицата $T$ на покриване на $A$ от $\mathcal{P}$ и $\alpha_{\mathcal{P}'}(\sigma_1, \sigma_2, \ldots, \sigma_m) \in J_2^m$ е характеристичен вектор на подмножеството $\mathcal{P}'$ на $\mathcal{P}$. $f(\sigma_1, \sigma_2, \ldots, \sigma_m) = 1$ т.с.т.к. $\mathcal{P}'$ е покритие на $A$.*

***Доказателство.*** 1) Нека $f(\sigma_1, \sigma_2, \ldots, \sigma_m) = 1$. Тогава $\forall i, D_i(\sigma_1, \sigma_2, \ldots, \sigma_m) = 1$. Нека $D_i = x_{i_1} \vee x_{i_2} \vee \cdots \vee x_{i_r}$. Съществува $x_{i_j}$ в $D_i$, такова че $\sigma_{i_j} = 1$. Следователно $S_{i_j}$ покрива $a_i$ и $\mathcal{P}'$, определено от $\alpha_{\mathcal{P}'}(\sigma_1, \sigma_2, \ldots, \sigma_m)$ е покритие.

2) Нека $(\sigma_1, \sigma_2, \ldots, \sigma_m)$ е характеристичен вектор на покритието $\mathcal{P}' \subseteq \mathcal{P}$. Тогава $\forall i, \exists S_j \in \mathcal{P}'$, който покрива $a_i$. От $S_j \in \mathcal{P}'$ следва, че $\sigma_j = 1$, а фактът, че $S_j$ покрива $a_i$ означава, че $x_j$ участва в $D_i$. Но тогава $D_i(\sigma_1, \sigma_2, \ldots, \sigma_m) = 1, \forall i$ ($\sigma_j$ замества $x_j$). Следователно $f(\sigma_1, \sigma_2, \ldots, \sigma_m) = 1$. □

Функцията на покритието е монотонна, защото е представена с формула над монотонните функции $xy$ и $x \vee y$. В термините на задачата за минимални покрития това означава, че не е възможно за характеристичните вектори $\alpha, \beta \in J_2^m$ да е изпълнено $\alpha \preceq \beta$, а $f(\alpha) = 1$ и $f(\beta) = 0$, т.е. не е възможно за определените от $\alpha$ и $\beta$ множества $\mathcal{P}_\alpha, \mathcal{P}_\beta \subseteq \mathcal{P}$ да е изпълнено $\mathcal{P}_\alpha \subseteq \mathcal{P}_\beta$, $\mathcal{P}_\alpha$ да е покритие, а $\mathcal{P}_\beta$ да не е.




Фигура 5.5: Горни единици на монотонна булева функция.

***Дефиниция.*** Вектора $\alpha$ наричаме *горна единица* на монотонната функция $f$, ако $f(\alpha) = 1$ и $\forall \gamma, \gamma \preceq \alpha, f(\gamma) = 0$.

Ако преведем дефиницията в термините на покрития, ще получим, че горните единици на монотонната функция на едно покритие са точно характеристичните вектори на минималните покрития, определени от това покритие. Ще покажем прост начин за намиране на горните единици (минималните покрития) на монотонна булева функция, която е в ДНФ. Очевидна е следната

**Лема 5.6.1** *Нека $f \in M, f(\alpha) = 1$ и $i_1, i_2, \ldots, i_r$ са позициите, в които $\alpha$ има единици. Тогава $K = x_{i_1} x_{i_2} \ldots x_{i_r}$ е елементарна конюнкция, такава, че $K(\beta) = 1$ $\forall \beta, \alpha \preceq \beta$.*

Всъщност всички $\beta$, такива че $\alpha \preceq \beta$, образуват $(n-r)$-мерен подкуб на $n$-мерния двоичен куб.

На Фиг. 5.5 схематично е представена монотонна булева функция с три горни единици $\alpha_1, \alpha_2, \alpha_3$.

**Лема 5.6.2** *Нека $K'$ и $K''$ са елементарни конюнкции без отрицания. Тогава $K' \vee K'K'' = K'$.*

Доказателството следва от свойството „поглъщане“, което доказахме в 5.2, в частния случай $f = K', g = K''$.

Поглъщането е операция, която приложена към елементарни конюнкции без отрицания води до намиране на съответните горни единици.



Действително, ако $\mathcal{P}_1$ и $\mathcal{P}_2$ са покрития, такива, че $\mathcal{P}_1 \subseteq \mathcal{P}_2$, т.е. $\mathcal{P}_2$ не е минимално, за елементарните им конюнкции без отрицания $K_1$ и $K_2$ очевидно имаме $K_2 = K_1 K$ за някоя елементарна конюнкция без отрицания $K$. Това означава, че ако в ДНФ без отрицания на монотонната функция $f$ приложим всички поглъщания, ще останат само елементарни конюнкции, представящи минимални покрития. От друга страна, тъй като всеки характеристичен вектор на минимално покритие е единица на функцията, то непременно поне една елементарна конюнкция, приемаща единица върху този вектор, ще участва в ДНФ. От вече казаното следва, че ако са направени всички поглъщания, това не може да бъде друга елементарна конюнкция, освен елементарната конюнкция на самото минимално покритие. Така доказахме следната

**Теорема 5.6.2** *ДНФ без отрицания на $f \in M$, в която са направени всички възможни поглъщания, съдържа елементарните конюнкции на минимални покрития (горни единици) и само тях.*

За да намерим минималните покрития, е достатъчно да обърнем функцията на покритията от КНФ в ДНФ, като разкрием скобите и направим съответните поглъщания. Забележете, че операцията поглъщане може да се прилага в ранни стадии на разкриването на скоби, тъй като, ако $K_1$ и $K_1 K$ са се оказали в една ДН подформа на $f$, то при всяко следващо разкриване на скобите към тях ще се свързват с конюнкция едни и същи елементарни конюнкции от друга ДН подформа и получената от $K_1$ елементарна конюнкция винаги ще поглъща получената от $K_1 K$.

Да приложим така разработената техника към функцията от разглеждания пример. За да си осигурим по бързо поглъщане и опростяване на пресмятанията, ще разкриваме с предимство скобите на такива две ДФ, които имат общи елементарни конюнкции. В примера това съображение ни насочва към разкриване на скобите в първата и втората, в третата и четвъртата елементарни дизюнкции, защото в тези двойки има съответно 1 и 2 общи променливи. И така

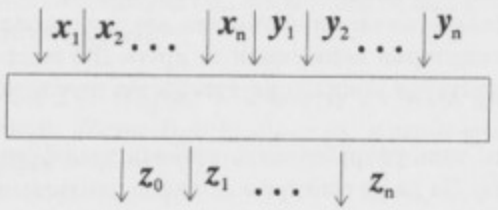
$$
\begin{aligned}
& f(x_1, x_2, x_3, x_4, x_5) = \\
= \; & (x_1 \vee x_2)(x_2 \vee x_5)(x_1 \vee x_3 \vee x_4 \vee x_5)(x_3 \vee x_4) = \\
= \; & (x_1 x_2 \vee x_1 x_5 \vee x_2 \vee x_2 x_5)(x_1 \vee x_3 \vee x_4 \vee x_5)(x_3 \vee x_4) = \\
= \; & (x_2 \vee x_1 x_5)(x_1 x_3 \vee x_1 x_4 \vee x_3 \vee x_3 x_4 \vee x_3 x_4 \vee x_4 \vee x_3 x_5 \vee x_4 x_5) = \\
= \; & (x_2 \vee x_1 x_5)(x_3 \vee x_4) = x_2 x_3 \vee x_2 x_4 \vee x_1 x_3 x_5 \vee x_1 x_4 x_5
\end{aligned}
$$

Съгласно Теорема 5.6.2 минимални покрития образуват подмножествата $\{\overline{x}, xy\}, \{\overline{x}, x \vee y\}$, $\{\tilde{1}, x \oplus y, xy\}$ и $\{\tilde{1}, x \oplus y, x \vee y\}$. Следователно това са всички базиси, които се съдържат в пълното множество $\{\tilde{1}, \overline{x}, xy, x \vee y, x \oplus y\}$. Първите три вече са ни познати. Четвъртият базис ни дава за всяка функция формула, аналог на полинома на Жегалкин.

## 5.7 Схеми от функционални елементи

Вече се запознахме с два начина за представяне на дискретни функции – с таблицата на стойностите или с формули над някое пълно множество. В този раздел ще разгледаме едно друго представяне, което е свързано с реалното построяване на изчислителни устройства.

При създаването на първите изчислителни машини актуална е била следната задача: да се построи устройство с $2n$ двоични входа $x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n$ и $n+1$ двоични изхода $z_0, z_1, z_2, \ldots, z_n$ (вж. Фиг. 5.6). Това устройство ще наричаме *$n$-разряден двоичен суматор*. По зададени в двоичен вид числа $x = \overline{x_1 x_2 \ldots x_n}_{(2)}$ и $y = \overline{y_1 y_2 \ldots y_n}_{(2)}$, $n$-разрядният двоичен суматор намира сумата им $x + y = \overline{z_0 z_1 z_2 \ldots z_n}_{(2)}$ в двоичен вид. За простота да разгледаме случая $n = 2$, който принципно не се различава от всеки друг случай.




Фигура 5.6: $n$-разряден двоичен суматор.

Търсим $\overline{z_0 z_1 z_2}_{(2)} = \overline{x_1 x_2}_{(2)} + \overline{y_1 y_2}_{(2)}$. Тъй като има четири двоични аргумента, построяваме Таблица 5.8, в която за всеки две двоични числа, сме определили резултата от сумирането им. Очевидно $z_0, z_1, z_2$ са три двоични функции на четирите променливи $x_1, x_2, y_1, y_2$.

Ако можехме за всяка булева функция $f(x_1, x_2, \ldots, x_n)$ да построим устройство, което по зададени стойности на двоичните променливи $x_1, x_2, \ldots, x_n$ да пресмята $f$, то търсеният суматор може да бъде построен като механичен сбор на трите устройства, пресмятащи $z_0(x_1, x_2, y_1, y_2)$,



| $x_1$ | $x_2$ | $y_1$ | $y_2$ | $z_0$ | $z_1$ | $z_2$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 |

Таблица 5.8: 2-разряден двоичен суматор, представен с булеви функции.

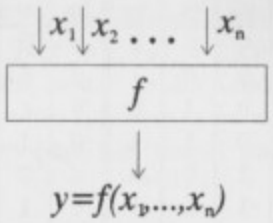
$z_1(x_1, x_2, y_1, y_2)$ и $z_2(x_1, x_2, y_1, y_2)$. Така задачата за построяване на произволно устройство, преобразуващо входните в изходни стойности по строго определени правила (алгоритъм), се свежда до построяването на съответна дискретна функция.

Конструктивните обекти, от които ще построяваме дискретните функции трябва да приличат на исканите устройства и да имат съответните входове и изходи. Не е възможно за всяка $q$-ична функция да има специализиран конструктивен елемент, който я изчислява, поради безкрайността на множеството от $q$-ични функции.

Аналогията с понятието формула ни навежда на мисълта да работим с ограничен брой такива устройства, реализиращи някои функции, а функциите, за които нямаме готови устройства, да построяваме, като комбинираме базовите устройства по някакъв начин, аналогичен на начина за построяване на формули.

***Дефиниция.*** Устройство (каквото и да съдържа в себе си) с $n$ $q$-ични входа $x_1, x_2, \ldots, x_n$ и един $q$-ичен изход $y$, такова, че $\forall \alpha \in J_q^n$ подаден на съответните входове, на изхода се получава $f(\alpha)$, за някоя $f \in \mathcal{F}_q$, наричаме ***функционален елемент*** и го означаваме с $e_f$ (вж. Фиг. 5.7).

Функционалните елементи са математическа абстракция на реални




Фигура 5.7: Функционален елемент.

устройства. Физическата им природа, начинът, по който се представят стойностите от $J_q$ и начинът, по който се пресмята $f(\alpha)$ във вътрешността на устройството не ни интересуват. Ще се абстрахираме и от много важен параметър на реалните (електронни) устройства – времето от подаването на стойностите на аргументите на входа до получаване на $y = f(\alpha)$, което ще считаме равно на 0, макар че в реалните устройства това не е така.

Комбинирането на функционални елементи ще извършваме в съответствие със следната дефиниция на *схема от функционални елементи*:

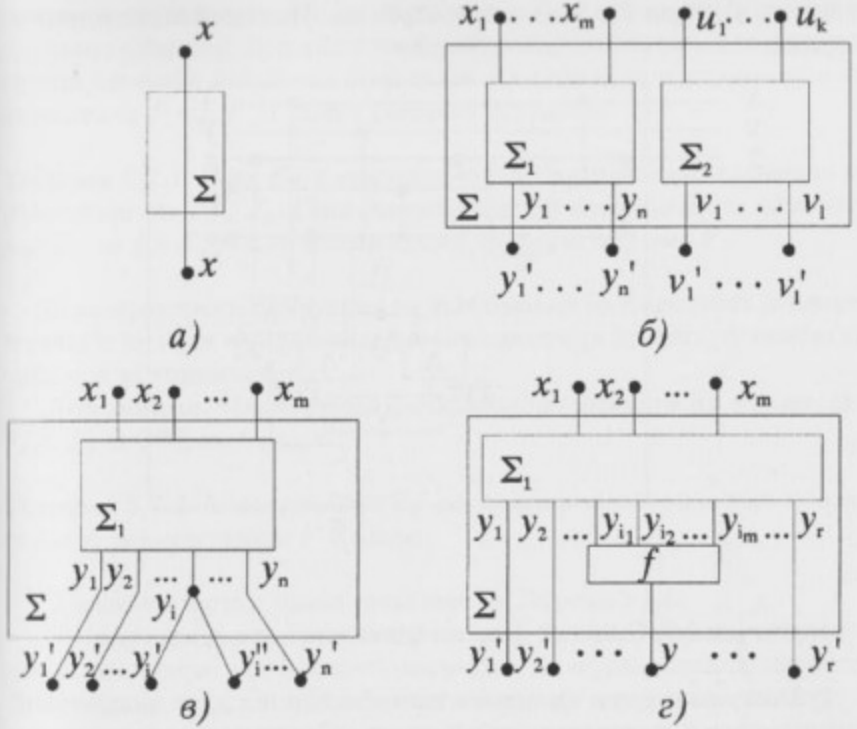
***Дефиниция.*** Нека $F = \{f_1, f_2, \ldots\} \subseteq \mathcal{F}_q$ и $E_F = \{e_{f_1}, e_{f_2}, \ldots\}$ е множеството от съответните им функционални елементи.

а) устройството $\Sigma$ от Фиг. 5.8.а (в което не се използва нито един елемент, а получената на входа стойност се предава директно на изхода) е схема от функционални елементи над $E_F$ – *тривиална схема*. Тази схема пресмята идентитета $f(x) = x$.

б) Нека $\Sigma_1$ и $\Sigma_2$ са схеми от функционални елементи (Фиг. 5.8.б) с входове съответно $x_1, \ldots, x_m$ и $u_1, \ldots, u_k$, пресмятащи функциите $y_i(x_1, \ldots, x_m), i = 1, 2, \ldots, n$ и $v_j(u_1, \ldots, u_k), j = 1, 2, \ldots, l$. Устройството $\Sigma$ от същата фигура наричаме *обединение на схемите* $\Sigma_1$ и $\Sigma_2$. То е схема от функционални елементи, като входовете ѝ са обединение на входовете на двете схеми, а изходите – обединение на изходите им. Функциите, които пресмята получената схема са $y_i'(x_1, \ldots, x_m, u_1, \ldots, u_k) = y_i(x_1, \ldots, x_m), i = 1, 2, \ldots, n$, а $v_j'(x_1, \ldots, x_m, u_1, \ldots, u_k) = v_j(u_1, \ldots, u_k)$, $j = 1, 2, \ldots, l$.

в) Нека $\Sigma$ е схема от функционални елементи с входове $x_1, x_2, \ldots, x_m$ и изходи $y_1, y_2, \ldots, y_n$ (вж. Фиг. 5.8.в). Устройството $\Sigma'$ от същата Фигура, получено от $\Sigma$ с *разклоняване на изхода* $y_i$, е също схема от функционални елементи. Входовете ѝ съвпадат с входовете на $\Sigma$, а изходите ѝ – с изходите на $\Sigma$, като $y_j' = y_j, j \neq i$ и $y_i' = y_i'' = y_i$.





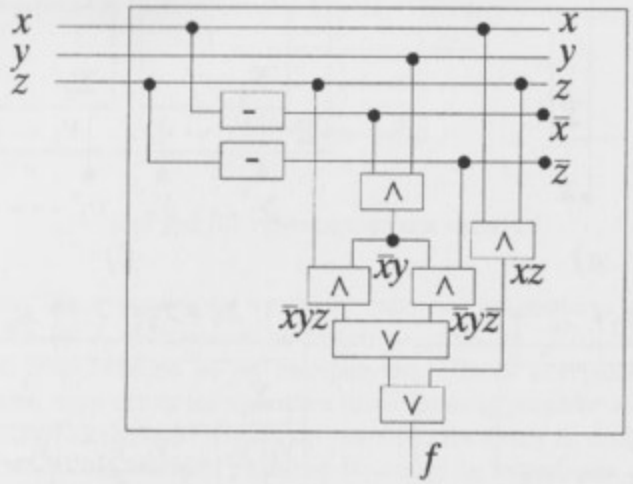

Фигура 5.8: Дефиниция на схеми от функционални елементи.

г) Нека $\Sigma$ е схема от функционални елементи с входове $x_1, x_2, \ldots, x_m$ и изходи $y_1, y_2, \ldots, y_r$ (вж. Фиг. 5.8.г) и $y_{i_1}, y_{i_2}, \ldots, y_{i_n}, n \leq r$ са избрани $n$ изхода на $\Sigma$. Нека $f(u_1, \ldots, u_n) \in F$. Устройството $\Sigma'$ от същата Фигура, получено от $\Sigma$ с *присъединяване на функционалния елемент* $e_f \in E_F$, е схема от функционални елементи. Входовете ѝ съвпадат с входовете на $\Sigma$, а изходите ѝ са $y_j' = y_j, j \notin \{i_1, i_2, \ldots, i_n\}$ и $y = f(y_{i_1}, y_{i_2}, \ldots, y_{i_n})$, суперпозицията на функциите, съпоставени на избраните $n$ изхода във функцията $f$ на присъединения елемент.

д) Няма други схеми от функционални елементи, освен тривиалните и индуктивно построените от тях с обединение, разклоняване на изходи и присъединяване на функционални елементи.

За пример да разгледаме множеството от функционални елементи $E_F = \{e_{\overline{x}}, e_{xy}, e_{x \vee y}\}$, съответно на множеството булеви функции $F =$

$\{\overline{x}, xy, x \vee y\}$. Нека $f(x,y,z) = \overline{x}yz \vee \overline{x}y\overline{z} \vee xz$. Прилагайки допустимите операции:




Фигура 5.9: Схема от функционални елементи $e_{\overline{x}}, e_{xy}, e_{x\vee y}$.

1) Построяваме три тривиални схеми за $x, y$ и $z$ и ги обединяваме. Разклоняваме $x$ и на единия от получените изходи присъединяваме отрицание. Разклоняваме два пъти $z$ и на единия от трите получени изхода присъединяваме отрицание.

2) Присъединяваме конюнкция към изходите $\overline{x}$ и $y$, както и конюнкция към $x$ и един от изходите $z$. Разклоняваме получения изход $\overline{x}y$ и на единия от получените изходи заедно с изхода $z$ присъединяваме конюнкция, а на другия, заедно с изхода $\overline{z}$ – друга конюнкция.

3) Присъединяваме дизюнкция към изходите $\overline{x}yz$ и $\overline{x}y\overline{z}$. Към получения изход, заедно с $xz$ присъединяваме още една дизюнкция. Получаваме схемата $\Sigma$ за $f$ (вж. Фиг. 5.9).

***Дефиниция.*** *Сложност на схемата* $\Sigma$ (означаваме я с $s(\Sigma)$) наричаме броя на участващите в нея функционални елементи.

В разгледания пример получихме схема със сложност 8 (две отрицания, четири конюнкции и две дизюнкции).

От дефиницията на схема от функционални елементи се вижда, че единственият начин за построяване на нови функции, различни от тези, за които има функционални елементи и идентитета, е операцията присъединяване на елемент. При тази операция новополучената функция е



суперпозиция на функции, за които вече са построени схеми от функционални елементи. В такъв случай построяването на схема от функционални елементи над $E_F$ за функцията $f$ е аналог на построяването на формула за $f$ над $F$. В сила е следното твърдение:

**Теорема 5.7.1** *Нека $E_F$ е множество от функционални елементи за функциите от $F \subseteq \mathcal{F}_q$. Съществува схема от функционални елементи над $E_F$ за $f \in \mathcal{F}_q$ т.с.т.к. съществува формула за $f$ над $F$.*

Доказателството с индукция по дълбочината на формулата (в едната посока) и по броя на присъединените елементи (в другата) оставяме на читателя за упражнение.

***Дефиниция.*** Множеството функционални елементи $E_F$ е *пълно*, ако $\forall f \in \mathcal{F}_q$, $\exists$ СФЕ за $f$ над $F$.

**Теорема 5.7.2** *Множеството $E_F$ от функционални елементи е пълно т.с.т.к. множеството $F$ е пълно.*

Доказателството е пряко следствие от Теорема 5.7.1.

От тук нататък ще разглеждаме само булеви функции и техните схеми, построени над пълното множество от функционални елементи $E_F = \{e_{\overline{x}}, e_{xy}, e_{x\vee y}\}$. За всяка функция можем да построим схема от функционални елементи над $E_F$, следвайки построяването на някоя нейна формула над $F$. Интерес представляват алгоритмите (или, както в теорията на схемите ги наричат, *методите* ) за построяване на схема за произволна булева функция.

В разгледания пример използвахме техника, която лесно можем да уточним и да получим прост метод за построяване схема на произволна функция, който наричаме *Метод на Дизюнктивните Нормални Форми*. Ще разгледаме по-простия *Метод на Съвършените Дизюнктивни Нормални Форми*.

**Метод на СъвДНФ.**

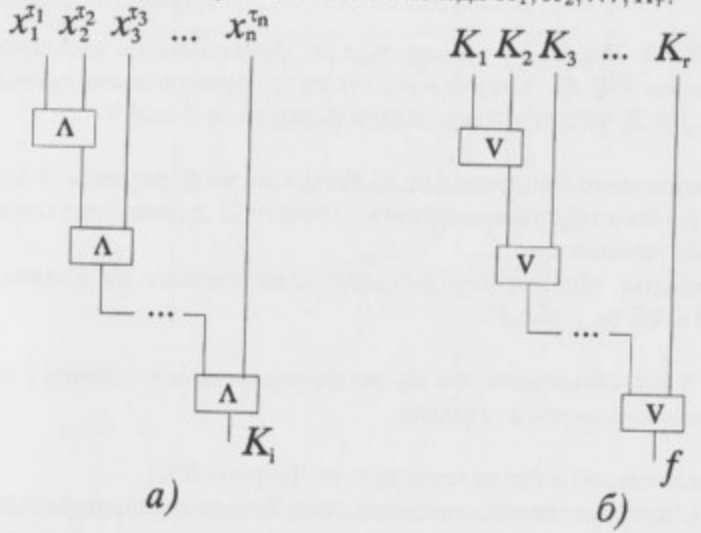
**Дадено:** СъвДНФ на булевата функция $f(x_1, x_2, \ldots, x_n)$.

**Резултат:** Схема от функционални елементи над $E_F = \{e_{\overline{x}}, e_{xy}, e_{x\vee y}\}$ за $f$.

**Процедура:**

1) Построяваме $n$ тривиални схеми за променливите $x_1, x_2, \ldots, x_n$. Разклоняваме всеки изход и на всеки един от получените нови изходи присъединяваме отрицание. Получаваме схема с изходи $x_1, x_2, \ldots, x_n$ и $\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}$.

2) Нека $f = K_1 \vee K_2 \vee \cdots \vee K_r$, където всяко $K_i$ е пълна елементарна конюнкция $K_i = x_1^{\tau_1} x_2^{\tau_2} \cdots x_n^{\tau_n}$. За всяко $K_i$ с последователно присъединяване на конюнкции към изходите $x_1^{\tau_1}, x_2^{\tau_2}, \cdots, x_n^{\tau_n}$, строим схема за $K_i$ (вж. Фиг. 5.10.а). Получихме схема с изходи $K_1, K_2, \ldots, K_r$.




Фигура 5.10: Метод на СъвДНФ.

3) С последователно присъединяване на дизюнкции към изходите $K_1$, $K_2$, ..., $K_r$ получаваме схема за $f$ (вж. Фиг. 5.10.б). □

На Фиг. 5.11 е показана схема за функцията $f(x, y, z) = \overline{x}yz \vee \overline{x}y\overline{z} \vee xyz$, построена по Метода на СъвДНФ.

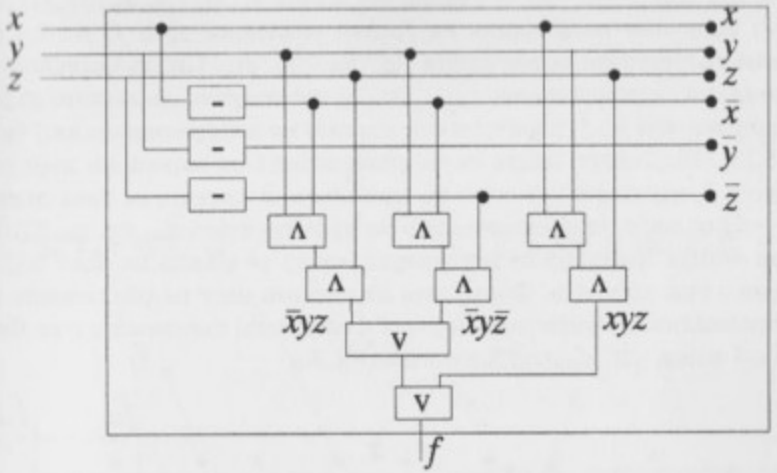
***Дефиниция.*** Нека $M$ е метод за построяване на всяка функция и със $\Sigma(f, M)$ сме означили схемата за функцията $f$, построена по метода $M$. Функцията $s_M(n) = \max_{f \in \mathcal{F}_2^n} s(\Sigma(f, M))$ наричаме *сложност на метода* $M$.

Да намерим $s_{\text{СъвДНФ}}(n)$. За целта ще забележим, че най-голяма сложност по Метода на СъвДНФ ще имаме за функция, която има максимален брой единици, т.е. $\tilde{1}$, като функция на $n$ променливи. Затова $s_{\text{СъвДНФ}}(n) = s(\tilde{1}, \text{СъвДНФ}) = n + 2^n(n-1) + 2^n - 1$, където $n$ е броят на отрицанията, $2^n(n-1)$ – броят на конюнкциите, а $2^n - 1$ – броят на дизюнкциите. Следователно $s_{\text{СъвДНФ}}(n) = n2^n + n - 1$.

**Модифициран метод на СъвДНФ (МСъвДНФ)**

**Дадено:** СъвДНФ на булевата функция $f(x_1, x_2, \ldots, x_n)$.






Фигура 5.11: Схема, построена по Метода на СъвДНФ.

**Резултат:** Схема от функционални елементи над $E_F = \{e_{\overline{x}}, e_{xy}, e_{x \vee y}\}$ за $f$.

**Процедура:** 1) Ако $f$ има $\leq 2^{n-1}$ единични стойности, строим схемата за $f$ по Метода на СъвДНФ.

2) Ако $f$ има $> 2^{n-1}$ единични стойности, тогава строим схема за $\overline{f}$ по Метода на СъвДНФ и към получения изход присъединяваме отрицание. □

Най-сложно при Модифицирания метод на СъвДНФ ще се построи функция $h$ с точно $2^{n-1}$ единични стойности и тя се реализира по Метода на СъвДНФ. Затова

$$s_{\text{МСъвДНФ}}(n) = n + 2^{n-1}(n-1) + 2^{n-1} - 1 = n2^{n-1} + n - 1.$$

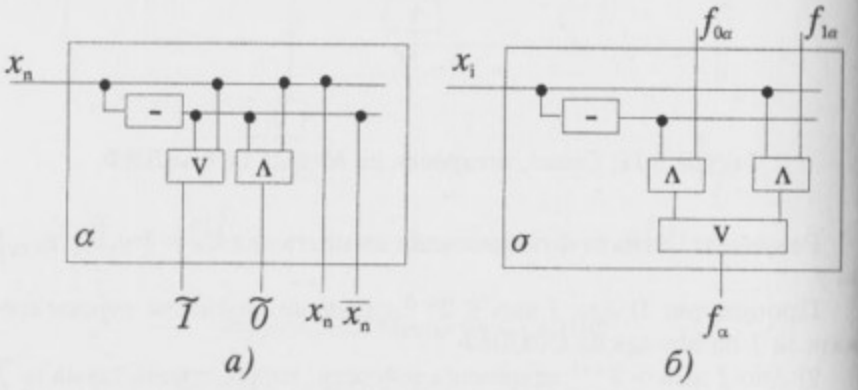
Тъй като $n2^{n-1}$ и $n2^n$ нарастват твърде бързо в сравнение с $(n-1)$, може да се твърди, че Модифицираният метод на СъвДНФ е около два пъти по-евтин от Метода на СъвДНФ.

Сега ще разгледаме метод с още по-ниска сложност.

Да приложим към булевата функция $f(x_1, x_2, \ldots, x_n)$ Теорема 5.3.1, като я разложим по първата ѝ променлива:

$$f(x_1, x_2, \ldots, x_n) = \overline{x_1} f(0, x_2, \ldots, x_n) \vee x_1 f(1, x_2, \ldots, x_n).$$

Функциите $f_0(x_2, \ldots, x_n) = f(0, x_2, \ldots, x_n)$ и $f_1(x_2, \ldots, x_n) = f(1, x_2, \ldots, x_n)$ наричаме *подфункции* на $f$. Ако разложим $f_0$ и $f_1$ по $x_2$, ще получим съответните подфункции $f_{00}$, $f_{01}$, $f_{10}$, $f_{11}$. Продължавайки по същия начин с променливите $x_3, \ldots, x_{n-1}$, ще получим двоичното дърво на подфункциите на $f$, наричано още *каскада на подфункциите* на $f$ (вж. Фиг. 5.13). Върховете, които са на разстояние $i$ от корена на дървото, означен с $f$, наричаме $i$-то ниво на каскадата. В листата на това дърво, на $(n-1)$-во ниво, са функции само на променливата $x_n$, т.е. $x_n, \overline{x_n}, \tilde{0}$ и $\tilde{1}$. Тези четири функции се реализират, както се вижда на Фиг. 5.12.а, от схема с три елемента. Формулата от дясната част на разлагането по една променлива реализираме лесно (с 4 елемента) със схемата $\sigma$ от Фиг. 5.12.б с 4 входа – $\overline{x_i}, f_{\alpha 0}, x_i, f_{\alpha 1}$ и изход $f_\alpha$.




Фигура 5.12: Базови схеми на Метода на каскадите.

**Метод на каскадите.**

**Дадено:** Формула на булевата функция $f(x_1, x_2, \ldots, x_n)$.

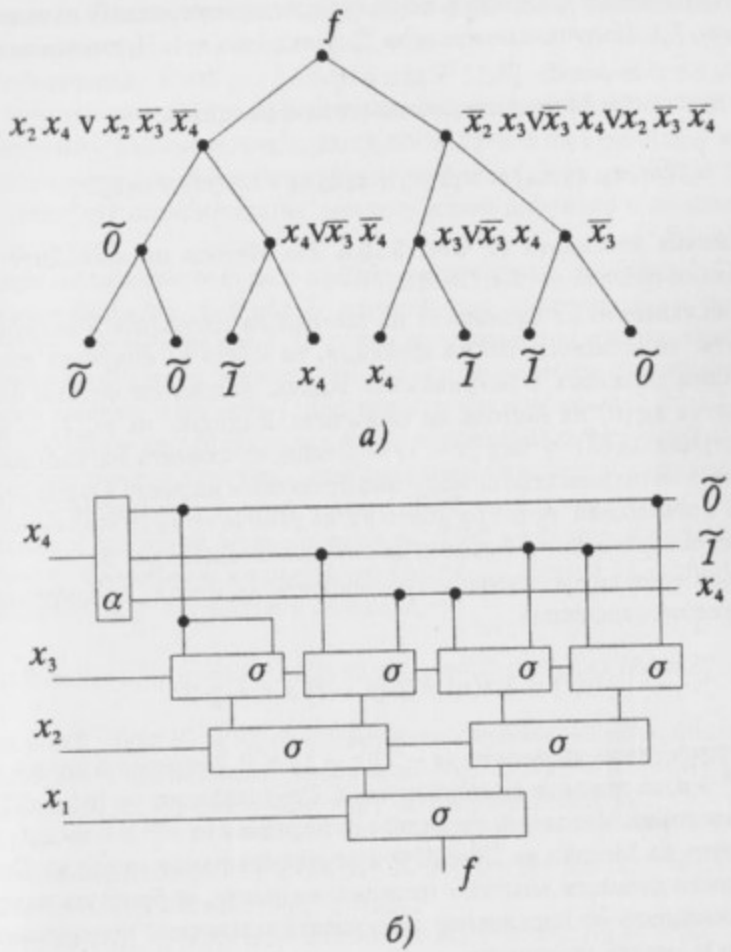
**Резултат:** Схема от функционални елементи над $E_F = \{e_{\overline{x}}, e_{xy}, e_{x \vee y}\}$ за $f$.

**Процедура:** 1) Построяваме каскада на подфункциите на $f$.

2) Построяваме $(n-1)$-во ниво на схемата за $f$, което е схемата от Фиг. 5.12.а за функциите на променливата $x_n$. Всеки от четирите изхода разклоняваме в толкова екземпляра, колкото пъти съответната функция на една променлива е подфункция на някоя от функциите от $(n-2)$-то ниво на каскадата. Нека $i = n-1$.

3) Ако $i = 0$, край. Иначе строим $(i-1)$-во ниво на схемата за $f$, като разклоняваме входа $x_i$ и към единия от получените изходи присъеди-






Фигура 5.13: Каскада (а) и схема (б) по Метода на каскадите.

няваме отрицание. Получените изходи $x_i$ и $\overline{x_i}$ разклоняваме в толкова екземпляра, колкото са подфункциите от $(i-1)$-то ниво на каскадата. За всяка подфункция $f_\alpha$ от $(i-1)$-то ниво на каскадата поставяме на $(i-1)$-то ниво на схемата за $f$ по един екземпляр от схемата $\sigma$ от Фиг. 5.12.б., като с четирите ѝ входа свързваме построените вече изходи $\overline{x_i}, f_{\alpha 0}, x_i, f_{\alpha 1}$. Получаваме изход за $f_\alpha$. Нека $i = i-1$. Преминаваме към 3. □

Да приложим Метода на каскадите за функцията

$$f(x_1, x_2, x_3, x_4) = x_1\overline{x_2}x_3 \vee x_1\overline{x_3}x_4 \vee \overline{x_1}x_2x_4 \vee x_2\overline{x_3}\,\overline{x_4}.$$

Построяваме каскадата от Фиг. 5.13.а. По Метода на каскадите за $f$ получаваме схемата от Фиг. 5.13.б.

Изчисляването на сложността на Метода на каскадите е затруднено от факта, че определянето на функция, за която се получава схема с най-голяма сложност, е нетривиална задача. Затова ще оценим отгоре сложността $s_K(n)$ на Метода на каскадите. Видяхме, че $s_K(1) = 3$. От друга страна $s_K(n) \leq 2s_K(n-1) + 4$, защото схемата на най-лошата функция на $n$ променливи се получава от схемите на двете ѝ подфункции на $n-1$ променливи, всяка от които е с не повече от $s_K(n-1)$ елемента, а четири елемента са необходими за свързването им чрез $\sigma$.

Затова една груба оценка на сложността на метода се получава от рекурентното отношение

$$t(1) = 3; t(n) = 2t(n-1) + 4, n \geq 2,$$

с характеристично уравнение $(x-2)(x-1) = 0$. Решението му ще е от вида $c2^n + d$, за някакви константи $c$ и $d$. Следователно, $s_K(n) \leq c2^n + d$ и сложността на Метода на каскадите се определя от $c2^n$ и е по-добра от сложността на Метода на СъвДНФ и Модифицирания метод на СъвДНФ. Сложен детайлен анализ, с отчитане на факта, че броят на подфункциите намалява от определено ниво нататък, показва, че константата $c$ е доста по-малка от единица.

Най-добрият резултат в областта на методите за представяне на произволна булева функция чрез СФЕ бе постигнат от С. В. Яблонский и О. Б. Лупанов, които с разработения от тях Метод на локалното кодиране успяха да покажат, че в най-лошия случай за реализация на булева функция на $n$ променливи са необходими и достатъчни от порядъка на $2^n/n$ елемента.



## 5.8 Минимизация на булеви функции

Както видяхме в предния раздел, построяването на схема за дадена булева функция може да стане чрез повтаряне на конструкцията на съответна формула. Затова колкото по-проста е формулата, толкова по-проста ще бъде и построената схема.

***Дефиниция.*** Нека $\varphi$ е формула над $F \subseteq \mathcal{F}_2$. *Сложност* на $\varphi$ наричаме броя на срещанията на букви на променливи във $\varphi$.

Например, сложността на $x\overline{y}z \vee xy\overline{z} \vee x\overline{yz}$ е 9, а на $1 \oplus x \oplus xy \oplus xyz$ е 6. Както се вижда, сложността на формулата може да се различава (макар и не много) от сложността на схемата, която директно е построена от тази формула.

Дори множеството $F$ да е крайно, задачата, по зададена $f \in [F]$ да се намери формула за $f$ над $F$ с минимална сложност, е изключително трудна, макар и решима. С пълно изчерпване по нарастващия брой срещания на букви на променливи, можем да намерим формулата с минимална сложност, но това е практически неизпълнимо.

Ще се спрем на следната задача за минимизация на формули, която има задоволително решение с доказана минималност: по зададена ДНФ на булевата функция $f$ да се намери ДНФ на $f$ с най-малка сложност. Наричаме я задача за *минимизация на ДНФ на булеви функции.*

Нека с $N_f$ означим множеството $\{\alpha | \alpha \in J_2^n, f(\alpha) = 1\}$ – *единичното множество* на $f$. Очевидна е следната

**Лема 5.8.1** *а)* $N_{f \vee g} = N_f \cup N_g$; *б)* $N_{f \wedge g} = N_f \cap N_g$; *в)* $N_{\overline{f}} = J_2^n \setminus N_f$.

**Лема 5.8.2** *Нека $K_1$ и $K_2$ са елементарни конюнкции на $n$ променливи. Необходимо и достатъчно условие за $N_{K_1} \subseteq N_{K_2}$ е $K_1 = K_2K'$ (т.е. всички букви на $K_2$ се срещат в $K_1$ на същите степени).*

***Доказателство.*** 1) Нека $K_1 = K_2K'$. Тогава $N_{K_1} = N_{K_2K'} = N_{K_2} \cap N_{K'}$. Следователно $N_{K_1} \subseteq N_{K_2}$.

2) Нека $N_{K_1} \subseteq N_{K_2}$. Не е възможно $K_1 = x^a K_1', K_2 = x^{\overline{a}} K_2'$, защото тогава $K_1K_2 = x^a x^{\overline{a}} K_1'K_2' = \tilde{0}$, т.е. $N_{K_1K_2} = N_{\tilde{0}}$ и $N_{K_1} \cap N_{K_2} = \emptyset$. Това противоречи на $N_{K_1} \subseteq N_{K_2}$, защото $N_{K_1} \neq \emptyset$.

Да допуснем, че в $K_2$ се съдържа буква $y$, която не участва в $K_1$. Нека $K_1 = x_{i_1}^{\sigma_1} \cdots x_{i_k}^{\sigma_k}$, $K_2 = y^\sigma K_2'$ и $y \neq x_{i_j}, j = 1, 2, \ldots, k$. Да построим вектор $\alpha$, така че $x_{i_1} = \sigma_1, \cdots, x_{i_k} = \sigma_k, y = \overline{\sigma}$ (забележете, че стойността за $y$ можем да изберем след определянето на стойностите за $x_{i_1}, \cdots, x_{i_k}$, защото $y$ е различна от всички тях). За останалите променливи избираме

произволни стойности. Очевидно $K_1(\alpha) = 1, K_2(\alpha) = 0$, т.е. $\alpha \in N_{K_1}$, но $\alpha \notin N_{K_2}$, а това е противоречие с $N_{K_1} \subseteq N_{K_2}$. Следователно $K_1 = K_2 K'$. □

Всъщност колкото по-малко букви има една елементарна конюнкция, толкова по-голям е броят на единиците ѝ. На този прост факт се основава решението на задачата за минимизация.

Ако си припомним простото свойство $xK \vee \overline{x}K = K$ (наречено поглъщане), ще видим, че добавянето на буквата $x$ към елементарната конюнкция $K$, несъдържаща $x$, дава елементарна конюнкция, чието единично множество съдържа половината от елементите на $N_K$. Останалата половина е единично множество на $\overline{x}K$. С други думи $N_{xK}$ и $N_{\overline{x}K}$ образуват разбиване на $N_K$.

***Дефиниция.*** Елементарната конюнкция $K$ наричаме *импликанта* на $f$, ако $N_K \subseteq N_f$. Не е трудно да се покаже, че ако $K$ е импликанта на $f$, то е в сила $(K \to F) = \tilde{1}$, откъдето и наименованието импликанта. Импликантата $K$ на $f$ се нарича *проста*, ако не съществува импликанта $K'$, такава че $N_K \subset N_{K'} \subseteq N_f$.

От Лема 5.8.2 следва, че $K$ е проста, ако не можем да премахнем от нея буква, без получената елементарна конюнкция да престане да бъде импликанта на $f$. Освен това, ако $xK$ и $\overline{x}K$ са импликанти на $f$, то от $xK \vee \overline{x}K = K$ следва, че те не са прости импликанти на $f$.

Всъщност представянето на коя да е булева функция $f$ с ДНФ можем да разгледаме като покриване на единиците на $f$ с импликанти. СъвДНФ е тривиално решение – всяка единица на $f$ се покрива от отделна пълна елементарна конюнкция. Ще докажем следната

**Теорема 5.8.1** *Минималната ДНФ на $f \in \mathcal{F}_2$ се състои само от прости импликанти.*

***Доказателство.*** Нека $f = D_{\min} = K_1 \vee K_2 \vee \cdots \vee K_r$ и да допуснем, че $K_1$ (без ограничение на общността) не е проста импликанта. Тогава $\exists K'$, такова че $N_{K_1} \subset N_{K'} \subseteq N_f$ и $K'$ е с по-малко букви от $K$. Но сега $D = K' \vee K_2 \vee \cdots \vee K_r$ е ДНФ на $f$. Наистина

$$\begin{aligned} N_D &= N_{K' \vee K_2 \vee \cdots \vee K_r} = N_{K'} \cup N_{K_2} \cup \cdots \cup N_{K_r} \subseteq N_f \\ N_f &= N_{D_{\min}} = N_{K_1 \vee K_2 \vee \cdots \vee K_r} = N_{K_1} \cup N_{K_2} \cup \cdots \cup N_{K_r} \subseteq \\ &\subseteq N_{K'} \cup N_{K_2} \cup \cdots \cup N_{K_r} = N_D. \end{aligned}$$

Формата $D$ обаче има по-малко букви от $D_{\min}$, което е противоречие. Следователно $D_{\min}$ се състои само от прости импликанти. □



**Теорема 5.8.2** *Нека $K_1, K_2, \ldots, K_s$ са всички прости импликанти на $f \in \mathcal{F}_2$. Тогава ДНФ $D = K_1 \vee K_2 \vee \cdots \vee K_s$ е ДНФ на $f$.*

***Доказателство.*** 1) Очевидно

$$N_D = N_{K_1 \vee K_2 \vee \cdots \vee K_s} = N_{K_1} \cup N_{K_2} \cup \cdots \cup N_{K_s} \subseteq N_f.$$

2) Ще покажем, че $N_f \subseteq N_D$. Действително нека $\alpha = (\sigma_1, \sigma_2, \ldots, \sigma_n) \in N_f$. Ако $K = x_1^{\sigma_1} x_2^{\sigma_2} \ldots x_n^{\sigma_n}$ е проста импликанта, то $K$ участва в $D$ и $\alpha \in N_D$. В противен случай $\exists$ проста импликанта $K_j$ в $D$ такава, че $N_K \subseteq N_{K_j}$ и отново $\alpha \in N_D$. □

***Дефиниция.*** ДНФ на $f$, образувана от всички прости импликанти, наричаме *Съкратена ДНФ* на $f$ (СДНФ).

От Теорема 5.8.1 следва, че минимална ДНФ можем да получим, ако по подходящ начин отстраним някои от простите импликанти на СДНФ. Ако се върнем към терминологията на задачата за минималните покрития, можем да гледаме на $N_f$ като на покривано множество, а на простите импликанти на СДНФ – като на покриващи елементи. Тогава минималните покрития ще имат важното свойство, че от тях не може да се премахне проста импликанта, без те да престанат да бъдат ДНФ на $f$. Ако една ДНФ не е минимално покритие, тя не може да бъде минимална ДНФ.

***Дефиниция.*** ДНФ на $f$, съставена от прости импликанти, от която не можем да премахнем нито една елементарна конюнкция, без получената ДНФ да престане да бъде ДНФ на $f$, наричаме *неприводима* ДНФ на $f$.

От направените по-горе разсъждения е ясно, че минимални ДНФ са някои от Неприводимите ДНФ на $f$. Затова задачата за минимизация на булева функция решаваме, като построим всички нейни неприводими ДНФ. Това става сравнително лесно с техниката, разработена в 5.6, но преди това е необходимо да можем да намираме СДНФ на функцията. Ще опишем два начина за това.

**Алгоритъм на Куайн-МакКласки**

**Дадено:** СъвДНФ на булева функция $f$.

**Резултат:** СДНФ на $f$.

**Процедура:** Строим последователно стълбове $(n), (n-1), \ldots, (r)$ на Таблицата на уайн-Мак ласки (вж. Фиг. 5.9), като броят им не е предварително определен.

1) В стълба $(n)$ записваме всички импликанти от СъвДНФ. Нека $i = n$.

| (4) | (3) | (2) |
|---|---|---|
| $\overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_4}$ * | $\overline{x_1}\,\overline{x_2}\,\overline{x_4}$ * | $\overline{x_2}\,\overline{x_4}$ |
| $\overline{x_1}\,\overline{x_2}\,x_3\,\overline{x_4}$ * | $\overline{x_2}\,\overline{x_3}\,\overline{x_4}$ * | |
| $x_1\,\overline{x_2}\,\overline{x_3}\,\overline{x_4}$ * | $\overline{x_2}\,x_3\,\overline{x_4}$ * | |
| $x_1\,\overline{x_2}\,x_3\,\overline{x_4}$ * | $\overline{x_1}\,x_3\,\overline{x_4}$ | |
| $x_1\,\overline{x_2}\,\overline{x_3}\,x_4$ * | $x_1\,\overline{x_2}\,\overline{x_3}$ | |
| $\overline{x_1}\,x_2\,x_3\,\overline{x_4}$ * | $x_1\,\overline{x_2}\,\overline{x_4}$ * | |
| $\overline{x_1}\,x_2\,x_3\,x_4$ * | $\overline{x_1}\,x_2\,x_3$ | |
| $x_1\,x_2\,\overline{x_3}\,x_4$ * | $x_1\,\overline{x_3}\,x_4$ | |
| $x_1\,x_2\,x_3\,x_4$ * | $x_2\,x_3\,x_4$ | |
| | $x_1\,x_2\,x_4$ | |

Таблица 5.9: Таблица на Куайн-МакКласки.

2) Нека е построен стълбът $(i)$ на Таблицата на Куайн-МакКласки. За всяка двойка импликанти $K_1$ и $K_2$ от $(i)$, такива че $K_1 = xK, K_2 = \overline{x}K$:

а) отбелязваме $xK$ и $\overline{x}K$ в стълба $(i)$;

б) записваме $K$ в стълба $(i-1)$.

3) Ако в стълба $(i-1)$ има поне две импликанти, то нека $i = i-1$ и преминаваме към 2. В противен случай – край, като $r = i$. □

Ще докажем някои свойства на таблицата.

**Лема 5.8.3** *В стълба $(i)$ се съдържат всички импликанти на $f$ с $i$ букви и само те.*

***Доказателство.*** 1) С индукция по $i$ ще докажем, че всички импликанти с $i$ букви са в стълба $(i)$. За $i = n$ твърдението е очевидно, защото всички импликанти на $f$ с $n$ букви са в СъвДНФ, от която образувахме стълба $(n)$. Допускаме верността на твърдението за $(i)$. Нека сега предположим, че в $(i-1)$ липсва импликантата $K$ с $i-1 < n$ букви. Нека $x_j$ е буква, която не участва в $K$. Тогава $x_jK$ и $\overline{x_j}K$ са импликанти на $f$ съгласно Лема 5.8.2 и от индукционното предположение следва, че са в стълба $(i)$. За тях алгоритъмът на Куайн-МакКласки ще постави $K$ в стълба $(i-1)$, което противоречи на предположението. Следователно всички импликанти с $i$ букви се съдържат в стълба $(i), i = n-1, \ldots, r$.

2) Също с индукция по $i$ ще докажем, че в стълба $(i)$ се съдържат само импликанти. Действително в стълба $(n)$ са само импликанти на $f$, а в резултат на слепване се получават импликанти, затова в цялата таблица се съдържат само импликанти на $f$.



Следователно в Таблицата се съдържат всички импликанти на $f$ и само импликанти на $f$. □

**Лема 5.8.4** *Неотбелязаните елементарни конюнкции в Таблицата на Куайн-МакКласки са всички прости импликанти на $f$.*

***Доказателство.*** От Лема 5.8.3 следва, че всички прости импликанти на $f$ са в Таблицата. Очевидно, всяка отбелязана импликанта не е проста. Остава да докажем, че ако $K$ не е отбелязана, то $K$ непременно е проста. Да допуснем противното, т.е. $K$ не е отбелязана, но не е проста. Следователно съществува буква $x_j$ в $K$, $K = x_j^\sigma K'$ и $K'$ също е импликанта на $f$. Нека $K$ се съдържа в $(i)$, тогава $K'$ е в $(i-1)$ съгласно Лема 5.8.3. Но сега $x_j^{\overline{\sigma}}K'$ също е в $(i)$, защото е импликанта. Алгоритъмът на Куайн-МакКласки слепва $x_j^\sigma K'$ и $x_j^{\overline{\sigma}}K'$, като ги отбелязва, а това е противоречие с допускането, че $K$ не е отбелязана. Следователно всяка неотбелязана импликанта е проста. □

Така доказахме следната

**Теорема 5.8.3** *Съкратената ДНФ на булевата функция $f$ се състои от всички неотбелязани в Таблицата на Куайн-МакКласки импликанти.*

Вторият алгоритъм за построяване на СДНФ на $f$, започвайки от СъвДНФ или от таблицата на функцията, се приписва от различни автори на Уейч или на Карно. Затова го наричаме *алгоритъм на Карно-Уейч*. Този алгоритъм е много удобен за $n = 3, 4$ и практически неприложим за $n \geq 5$. На Фиг. 5.10 са показани *картите на Карно-Уейч* за $n = 3, 4$.

| $x_1$ \ $x_2$ | 0 0 | 1 1 | $x_2$ |
|---|---|---|---|
| | 0 1 | 1 0 | $x_3$ |
| 0 | 1 | 1 | 1 1 |
| 1 | | | 1 |

а)

| $x_1$ $x_2$ \ $x_3$ | 0 | 1 | 1 | 0 | $x_4$ |
|---|---|---|---|---|---|
| 0 0 | 1 | | | 1 | |
| 0 1 | | | 1 | 1 | |
| 1 1 | | 1 | 1 | | |
| 1 0 | 1 | 1 | | 1 | |

б)

Таблица 5.10: Карти на Карно-Уейч.

Картите на Карно-Уейч са начин за представяне на стойностите на функцията като двумерна таблица, като надписите по редове и стълбове са подредени в код на Грей и осигуряват по този начин геометрична

близост за единичните стойности на $f$, формиращи импликанти. Заради цикличността на кода на Грей, картите на Карно-Уейч можем да разглеждаме като циклично „съединени“, както по хоризонтала (левият и десният край са съединени), така и по вертикала (горният и долният край са съединени). По този начин всеки две съседни единици в ред или стълб образуват импликанта. Импликантите с четири единици се появяват като четири единици в ред или стълб (за $n = 4$) или в квадрат със страна 2. Импликантите с 8 единици се образуват от два съседни реда или стълба. Прости са тези импликанти, които не се съдържат в други импликанти.

Така на Фиг. 5.10.а се виждат една проста импликанта, образувана от единиците на третия стълб и една от единиците на първия ред. Определянето на съответните елементарни конюнкции е просто. Достатъчно е да се намерят тези променливи, които запазват стойността си за всички вектори, образуващи импликантата, и да се вземе тази стойност за степен на съответната променлива. Третият стълб е образуван от векторите 011 и 111, следователно простата импликанта, която той представлява, е $x_2x_3$. За първия ред постоянна е само стойността на $x_1$ и тя е 0. И така СДНФ на функцията, представена в картата на Фиг. 5.10.а е $\overline{x_1} \vee x_2x_3$.

За функцията на четири променливи, представена с картата от Фиг. 5.10.б имаме следните прости импликанти:

- квадратът от четирите ъгъла на картата – $\overline{x_2}\,\overline{x_4}$;
- двойките съседни единици във втори, трети и четвърти ред – $\overline{x_1}\,x_2\,x_3$, $x_1\,x_2\,x_4$ и $x_1\,\overline{x_2}\,\overline{x_3}$;
- двойките съседни единици във втори, трети и четвърти стълб – $x_1\,\overline{x_3}\,x_4$, $x_2\,x_3\,x_4$ и $\overline{x_1}\,x_3\,\overline{x_4}$.

И така СДНФ на функцията, представена с картата от Табл. 5.10.б е $\overline{x_2}\,\overline{x_4} \vee \overline{x_1}\,x_2\,x_3 \vee x_1\,x_2\,x_4 \vee x_1\,\overline{x_2}\,\overline{x_3} \vee x_1\,\overline{x_3}\,x_4 \vee x_2\,x_3\,x_4 \vee \overline{x_1}\,x_3\,\overline{x_4}$. След като сме намерили СДНФ на $f$, ще построим всички неприводими форми на $f$, а това е задача за минимални покрития. Редовете на таблицата на покритие съответстват на простите импликанти (от СДНФ), а стълбовете на таблицата – на единичните стойности на функцията (от СъвДНФ). Една проста импликанта $K$ покрива единица на функцията $\alpha$, ако $K(\alpha) = 1$. Задачата за покритие решаваме по известния вече начин и намираме всички неприводими форми на $f$. Забележете, че минималността на покритието не гарантира минималност на ДНФ съгласно дефиницията. Достатъчно е обаче да преброим срещанията на буквите в неприводимите ДНФ и тази (тези) от тях, която (които) имат минимален брой букви, ще бъде (бъдат) минимална (минимални).

За съкратената ДНФ на $f$ представена в Табл. 5.10.б получаваме



задачата за минимални покрития от Табл. 5.11. За по-просто, вместо $x_i$ навсякъде в таблицата сме поставили $i$. Преди да приложим алгоритъма за намиране на минимални покрития, да отбележим, че простата импликанта $\overline{x_2}\,\overline{x_4}$ е единствена, която покрива две от единиците на функцията. Следователно тази импликанта участва във всяка НДНФ, а следователно и в минималната.

| | $\overline{1}\,\overline{2}\,\overline{3}\,\overline{4}$ | $\overline{1}\,\overline{2}\,3\,\overline{4}$ | $1\,\overline{2}\,\overline{3}\,\overline{4}$ | $1\,\overline{2}\,3\,\overline{4}$ | $1\,\overline{2}\,3\,4$ | $\overline{1}\,2\,3\,\overline{4}$ | $\overline{1}\,2\,3\,4$ | $1\,2\,\overline{3}\,4$ | $1\,2\,3\,4$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| $\overline{2}\,\overline{4}$ | * | * | * | * | | | | | | |
| $\overline{1}\,3\,\overline{4}$ | | * | | | | * | | | | $t_1$ |
| $1\,\overline{2}\,\overline{3}$ | | | * | | * | | | | | $t_2$ |
| $\overline{1}\,2\,3$ | | | | | | * | * | | | $t_3$ |
| $1\,\overline{3}\,4$ | | | | | * | | | * | | $t_4$ |
| $2\,3\,4$ | | | | | | | * | | * | $t_5$ |
| $1\,2\,4$ | | | | | | | | * | * | $t_6$ |

Таблица 5.11: Минимизацията на БФ е задача за минимални покрития.

***Дефиниция.*** Импликанта, която се съдържа във всяка НДНФ (а следователно и във всяка МДНФ), наричаме *ядрова*.

Очевидно премахването на редовете на ядровите импликанти и всички стълбове на елементи, които те покриват, силно опростява таблицата на задачата за минималните покрития. Възможно е в таблицата да не остане нито един ред и нито един стълб, ако всяка проста импликанта се окаже ядрова. В нашия пример непокрити от ядровата импликанта $2\,\overline{4}$ са елементите в последните пет стълба на таблицата на покритие. Затова присвояваме на неядровите импликанти булевите променливи $t_1, t_2, t_3, t_4, t_5$ и $t_6$. Получаваме следната функция на покритието

$$(t_1 \vee t_3)(t_2 \vee t_4)(t_3 \vee t_5)(t_4 \vee t_6)(t_5 \vee t_6).$$

Преобразуваме я в ДНФ, като правим възможните поглъщания:

$$\begin{aligned} & (t_1 \vee t_3)(t_2 \vee t_4)(t_3 \vee t_5)(t_4 \vee t_6)(t_5 \vee t_6) = \\ = \; & (t_3 \vee t_1t_5)(t_4 \vee t_2t_6)(t_5 \vee t_6) = (t_3t_5 \vee t_3t_6 \vee t_1t_5)(t_4 \vee t_2t_6) = \\ = \; & t_3t_4t_5 \vee t_3t_4t_6 \vee t_1t_4t_5 \vee t_2t_3t_6 \vee t_1t_2t_5t_6. \end{aligned}$$

Следователно функцията има пет неприводими ДНФ, а четири от тях:

$\overline{x_2}\,\overline{x_4} \vee \overline{x_1}\,x_2\,x_3 \vee x_1\,\overline{x_3}\,x_4 \vee x_2\,x_3\,x_4$,
$\overline{x_2}\,\overline{x_4} \vee \overline{x_1}\,x_2\,x_3 \vee x_1\,\overline{x_3}\,x_4 \vee x_1\,x_2\,x_4$,
$\overline{x_2}\,\overline{x_4} \vee \overline{x_1}\,x_3\,\overline{x_4} \vee x_1\,\overline{x_3}\,x_4 \vee x_2\,x_3\,x_4$,

и

$\overline{x_2}\,\overline{x_4} \vee x_1\,\overline{x_2}\,\overline{x_3} \vee \overline{x_1}\,x_2\,x_3 \vee x_1\,x_2\,x_4$
са минимални.

---

## Упражнения

### Упражнение 5.1

Пресметнете:

а) десетичното число, на което векторът $(11010101101) \in J_2^{11}$ е двоично представяне;

б) естественото число $n$ и вектора $\alpha \in J_2^n$, който е двоично представяне на десетичното число 2005.

### Упражнение 5.2

Намерете броя на:

а) подмножествата от булеви функции $\{f, g\}$ на $n$ променливи такива, че за стълбовете $\alpha_f$ и $\alpha_g$ е в сила $\rho(\alpha_f, \alpha_g) = k, 1 \leq k \leq 2^n$;

б) булевите функции $f$ на $n$ променливи такива, че за стълбовете им $\alpha_f$ е в сила $\alpha_g \preceq \alpha_f \preceq \alpha_h$, където $\alpha_g \preceq \alpha_h$ са стълбовете на две дадени булеви функции на $n$ променливи;

в) булевите функции $f$ на $n$ променливи такива, че за зададени $l$ вектора от $J_2^n$, $1 \leq l \leq 2^n$ стойностите им са фиксирани, а на останалите вектори – произволни.

г) булевите функции $f$ на $n$ променливи, които при транспозиция на дадени две от променливите не се изменят;

д) булевите функции $f$ на $n$ променливи, които са *симетрични*, т.е. при произволна пермутация на променливите не се изменят.

### Упражнение 5.3

Шестима експерти $E_1, E_2, E_3, E_4, E_5, E_6$ извършват оценки на проекти, като проектът се приема, ако болшинството от експертите гласуват „за“ или пък ако „за“ са гласували трима от експертите, между които председателят на експертния съвет – $E_1$. Напишете булева функция на променливите $x_1, x_2, x_3, x_4, x_5, x_6$, която приема стойност 1 т.с.т.к. проектът е приет, при условие, че $x_i = 1$ т.с.т.к. експертът $E_i$ е гласувал „за“.



### Упражнение 5.4

Три машини – $X_1, X_2, X_3$, консумиращи еднакво количество енергия, се захранват от два агрегата: $F$, който може да захрани една машина и $G$, който може да захрани две машини. Напишете булеви функции $f(x_1, x_2, x_3)$ и $g(x_1, x_2, x_3)$, които да имат стойност единица т.с.т.к. съответните агрегатите $F$ и $G$ са включени (така, че разумно да се изразходва енергията). Променливата $x_i$ приема стойност 1 т.с.т.к. машината $X_i$ е включена.

### Упражнение 5.5

Представете с кореново дърво следните формули и намерете стълба на съответната им булева функция:

а) $(\overline{x \to \overline{y}}) \oplus ((z \vee (y \wedge x))|y)$;

б) $(((((x \oplus y) \oplus x) \oplus y) \oplus z) \oplus (x \oplus z))$;

в) $(((x \downarrow y)|(y \downarrow x))|z) \downarrow (x|z)$.

### Упражнение 5.6

Ако $\alpha_f = (0010)$ и $\alpha_g = (0100)$ са стълбовете на булевите функции $f$ и $g$, намерете стълба на булевата функция $h(x, y) = f(x, g(y, x)) \vee g(y, f(x, x))$.

### Упражнение 5.7

Проверете следните еквивалентности на формули:

а) $x \vee y = (x \to y) \to y$;

б) $x \equiv y = (x \to y) \wedge (y \to x)$;

в) $x \to (y \wedge z) = (x \to y) \wedge (x \to z)$;

г) $x \to (y \to z) = (x \to y) \to (x \to z)$;

д) $x \wedge (\overline{y} \to \overline{x}) \to y = \tilde{1}$.

### Упражнение 5.8

Определете съществените и фиктивните променливи на функцията:

а) $f(x_1, x_2, x_3, x_4)$ със стълб $(1100001111000011)$;

б) $f(x_1, x_2, x_3, x_4) = (x_1 \to x_2x_3)(x_2 \to x_1x_3) \vee (x_1 \equiv x_2)$.

**Упражнение 5.9**

За кои $n, n \geq 2$ функцията $f \in \mathcal{F}_2$ зависи съществено от всичките си променливи:

а) $f = (x_1 \vee x_2 \vee \cdots \vee x_n) \to ((x_1x_2)(x_2x_3)\cdots(x_{n-1}x_n)(x_nx_1))$;

б) $f = (x_1x_2 \vee x_2x_3 \vee \cdots \vee x_{n-1}x_n \vee x_nx_1) \to (x_1x_2 \oplus x_2x_3 \oplus \cdots \oplus x_{n-1}x_n \oplus x_nx_1)$.

**Упражнение 5.10**

Напишете СъвДНФ, СъвКНФ и полином на Жегалкин на функцията със стълб (0000100010010000).

**Упражнение 5.11**

Докажете, че променливата $x$ е съществена за функцията $f \in \mathcal{F}_2$ т.с.т.к. участва в полинома на Жегалкин на $f$.

**Упражнение 5.12**

Преобразувайте от:

а) ДНФ в КНФ функцията $x_1x_2\overline{x_3x_4} \vee \overline{x_1}x_2\overline{x_3} \vee \overline{x_2}x_4$;

б) КНФ в ДНФ функцията $(x_1 \vee \overline{x_2} \vee x_3)(\overline{x_1} \vee x_2 \vee \overline{x_3})(x_1 \vee x_2 \vee x_3)$.

**Упражнение 5.13**

Нека $G_1$ и $G_2$ са затворени подмножества на $\mathcal{F}_2$. Затворено множество ли е:

а) $G_1 \cup G_2$; б) $G_1 \cap G_2$; в) $G_1 \setminus G_2$; г) $\overline{G_1}^{\mathcal{F}_2}$.

**Упражнение 5.14**

Кои от следните функции са самодвойнствени:

а) $f = x_1x_2 \oplus x_1x_3 \oplus x_2x_3 \oplus x_1x_2x_3$;

б) $f$ със стълб (1100100101101100).

**Упражнение 5.15**

Докажете, че всяка самодвойнствена функция на три променливи е или линейна, или от вида $x^\alpha y^\beta \vee x^\alpha z^\gamma \vee y^\beta z^\gamma$.



**Упражнение 5.16**

Самодвойнствени ли са множествата от булеви функции:

а) $[\{\tilde{1}, xy, x \oplus y\}]$; б) $[\{\tilde{1}, x \oplus y\}]$; в) $[\{x \oplus y\}]$.

**Упражнение 5.17**

Докажете, че функцията $f \in \mathcal{F}_2^n$, която зависи съществено от всичките си променливи, е линейна т.с.т.к. при произволно заместване на $k (1 \leq k \leq n)$ променливи с константи, получената функция зависи съществено от всички останали $n - k$ променливи.

**Упражнение 5.18**

Постройте от нелинейната функция $f$ дизюнкция с помощта на константите и отрицанието.

**Упражнение 5.19**

Докажете, че не съществува монотонна самодвойнствена функция с точно 2 горни единици.

**Упражнение 5.20**

Докажете, че всяка проста импликанта на монотонна функция не съдържа отрицания.

**Упражнение 5.21**

Проверете пълнотата на множеството булеви функции

$$\{\tilde{0}, \tilde{1}, x \oplus y \oplus z, xy \oplus yz \oplus xz, xy \oplus z, x \vee y\}.$$

Определете всички съдържащи се в него базиси.

**Упражнение 5.22**

При какви $n$ са Шеферови функциите на $n$ променливи:

а) $1 \oplus x_1x_2 \oplus x_2x_3 \oplus \cdots \oplus x_{n-1}x_n$;

б) $1 \oplus x_1x_2 \oplus x_2x_3 \oplus \cdots \oplus x_{n-1}x_n \oplus x_nx_1$;

в) $1 \oplus \bigoplus_{1 \leq i < j \leq n} x_ix_j$.

**Упражнение 5.23**

Намерете броя на Шеферовите функции на $n$ променливи.

**Упражнение 5.24**

Проверете пълно ли е множеството булеви функции:
а) $(S \cap M) \cup (L \setminus M)$;
б) $(M \setminus T_0) \cup (L \setminus S)$;
в) $((L \cap M) \setminus T_1) \cup (S \cap T_1)$.

**Упражнение 5.25**

Постройте схеми с колкото може по-малко функционални елементи за функциите от Упражнения 5.3 и 5.4.

**Упражнение 5.26**

Постройте по Метода на каскадите схема за множеството функции $\{x_1, 1 \oplus x_1, x_2 \oplus x_3, 1 \oplus x_2 \oplus x_3, x_1 \oplus x_2 \oplus x_3, 1 \oplus x_1 \oplus x_2 \oplus x_3\}$.

**Упражнение 5.27**

Нека $n_1, n_2, \ldots, n_k$ са естествени числа, такива че $\sum_{i=1}^{k} 2^{n_i} = 2^n$. Докажете че съществува разбиване на $J_2^n$ на подкубове $C_1, C_2, \ldots, C_k$ с размерности $n_1, n_2, \ldots, n_k$, съответно.

**Упражнение 5.28**

Минимизирайте булевата функция $f$ със стълб:
а) $\alpha_f = (1110100001101000)$;
б) $\alpha_f = (1110011000010101)$;
в) $\alpha_f = (0001011110101110)$.

**Упражнение 5.29**

Нека $f$ е булева функция със стълб (0101110000111010). Докажете, че функцията

$$h(x_1, x_2, \ldots, x_n) = f(x_1, x_2, x_3, x_4) \oplus x_5 \oplus \cdots \oplus x_n$$

има $10^{2^{n-4}}$ неприводими и $2^{2^{n-4}}$ минимални ДНФ.



**Упражнение 5.30**

Понякога се налага да се минимизира булева функция $f$, някои от стойностите на която не са от значение. Такива стойности означаваме с $\star$. За всяка от дадените по-долу със стълба си $\alpha_f$ булеви функции, намерете това заместване на стойностите без значение с 0 или 1, при което съответната минимална ДНФ е най-проста:
а) $\alpha_f = (11101000 \star 1101000)$;
б) $\alpha_f = (1110 \star 11000 \star 10101)$;
в) $\alpha_f = (0001 \star 11110 \star \star 1110)$.

# Глава 6

# Кодиране на информацията

Под кодиране на информацията обикновено се разбира нейното засекретяване, представянето ѝ в такъв вид, че за неупълномощени лица да бъде непонятен вложеният в нея смисъл. Днес с това понятие обозначаваме няколко, доста различаващи се една от друга области на границата на дискретната математика и теоретичната информатика. Обща за тези области е технологията, която не зависи от конкретната цел. Тази технология се състои в замяна на думи от някаква азбука, по някакви правила, с думи над друга азбука (не непременно различна от първата). Съществено е, че задължително трябва да имаме възможност за възстановяване на първоначалната дума. Всяка от областите, за които говорим, има специфична цел, която я отличава от другите области и налага специфични понятия и методи.

Известната още от дълбока древност област, за която споменахме в началото, се нарича *криптография*. С въвеждането на компютрите като средство за съхраняване на информацията и особено с широкото навлизане в практиката на компютърните мрежи, криптографските методи за защита непрекъснато повишават актуалността си. Към класическата цел на криптографията – да се скрие съдържанието на определено съобщение от лицата, нямащи съответните пълномощия, се прибавят и други интересни цели, като например възможността да се установи по недвусмислен начин кой е авторът на дадено съобщение, т.н. *електронен подпис*. Тази задача е от изключителна важност в условията на електронни комуникации между лица и институции, особено в сфери, в които източникът на информацията е обект на правови взаимоотношения.

Друга изключително важна област от кодирането е *компресията на информацията* – представянето ѝ в колкото може по-компактен вид. Независимо от това, че се създават запомнящи устройства и носители с все по-големи възможности, скоростта с която нараства обемът на „про-

извежданата" информация и ускорението на този растеж са такива, че актуалността на компресията като метод за обработка на информацията ще се запази за дълъг период (освен ако не бъде извършена революция в способите за запомняне на информация). Съвършено различни са задачите за компресия на текстове, при която възстановяването на първоначалния текст е задължително, и задачата за компресия на изображения, където се допускат неголеми отклонения от първоначалния вид.

Трета област на кодирането, интересът към която не намалява, независимо от качествения скок в използваните технологии за пренос на информация, е *шумозащитното кодиране* . Каналите за пренасяне на данни внасят нежелани изменения в обработваната информация – те „шумят". В повече от случаите шумът в получените данни е абсолютно недопустим – например при предаване на изпълними кодове на компютърни програми, при които незначителни изменения могат да доведат до катастрофални последици. Но дори и в области, където абсолютната точност на възстановяване на началната информация не е жизнено важна, шумозащитното кодиране допринася за по-високото качество на пренасяне на информацията. Например в транслирането на качествени видеоизображения или звукозаписи.

В тази глава се дават някои начални теоретични познания от дискретната математика, необходими за по-бързото навлизане в съответните области.

## 6.1 Побуквено кодиране

Възможни са различни подходи за трансформиране на информацията, които нарекохме с общото име кодиране. Ще се спрем на един от най-разпространените – *побуквеното кодиране*.

Нека $A = \{a_1, a_2, \ldots, a_n\}$ е крайна азбука, наричана *изходна*. Информационните единици, които ще кодираме, се представят като думи над тази азбука и се наричат *изходни текстове* или *съобщения*. Нека $B = \{b_1, b_2, \ldots, b_m\}$ е друга крайна азбука (без да изключваме възможността $A = B$), наричана *кодова азбука*, с помощта на която ще кодираме изходните съобщения.

***Дефиниция.*** Функцията $\kappa : A \to B^+$, такава че $\kappa(a_i) \neq \kappa(a_j)$ при $i \neq j$ наричаме *побуквен код*, а думите $\kappa_1 = \kappa(a_1), \kappa_2 = \kappa(a_2), \ldots, \kappa_n = \kappa(a_n)$ – *кодови думи*.

В компютърните системи, например, се използват побуквени кодове (EBCDIC, ASCII, Unicode и т.н.). На всяка буква от азбуката, с която си служим в компютрите (клавиатурните и някои специални знаци), те съпоставят дума от азбуката $\{0,1\}$ с фиксирана дължина (при EBCDIC и ASCII тя е 8, а при Unicode – 16 двоични букви).



Побуквеният код $\kappa : A \to B^+$ наричаме *равномерен*, ако $d(\kappa_1) = d(\kappa_2) = \cdots = d(\kappa_n) = d$. Числото $d$ наричаме *дължина* на равномерния код.

Функцията $\kappa : A \to B^+$ естествено разширяваме до $\kappa^* : A^+ \to B^+$ по следния начин. Нека $\alpha \in A^+, \alpha = a_{i_1} a_{i_2} \ldots a_{i_l}$. Тогава $\kappa^*(\alpha) = \kappa_{i_1} \kappa_{i_2} \ldots \kappa_{i_l}$. Думата $\kappa^*(\alpha)$ наричаме *кодирано съобщение*. Процедурата, построяваща по зададено съобщение съответното кодирано съобщение, наричаме *кодиране*.

***Дефиниция.*** Казваме, че побуквеният код $\kappa : A \to B^+$ е *разделим* или, че *осигурява еднозначност на декодирането*, ако $\forall \beta \in B^+$, $\exists$ не повече от една $\alpha \in A^+$ такава, че $\kappa^*(\alpha) = \beta$.

Процедурата, която по зададена $\beta \in B^+$ намира (ако съществува) съответната $\alpha \in A^+$, такава че $\kappa^*(\alpha) = \beta$ наричаме *декодиране*.

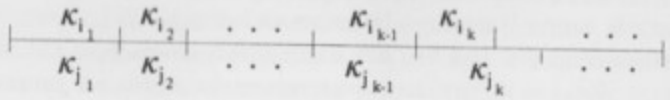
Равномерните кодове осигуряват еднозначно декодиране. Процедурата за декодиране в този случай е много проста. Кодираното съобщение се разбива на поддуми с дължина, равна на дължината на кода. Всяка такава поддума се замества с първообраза си в съответствие с дефиницията на кода. В случай, че някоя от поддумите не е кодова дума, декодирането става невъзможно, но в никакъв случай не могат да се получат две различни декодирания. Да разгледаме обаче неравномерния код с изходна азбука $A = \{a, b, c\}$, кодова азбука $B = \{0, 1\}$ и кодови думи $\kappa_a = 01, \kappa_b = 11, \kappa_c = 0111$. Да разгледаме кодираното съобщение 01110111. Лесно се вижда, че то може да бъде декодирано като $abab, abc, cab$ или $cc$. Следователно неравномерните кодове не винаги осигуряват еднозначност на декодирането.

***Дефиниция.*** Думата $\alpha$ наричаме *префикс* на думата $\beta$, ако $\beta = \alpha\gamma, \gamma \neq \varepsilon$. Кода $\kappa$ наричаме *префиксен*, ако никое $\kappa_i$ не е префикс на $\kappa_j, i \neq j$.

**Теорема 6. .** *Всеки префиксен код е разделим.*

***Доказателство.*** Нека $\kappa$ е префиксен. Да допуснем, че не е разделим. Следователно $\exists$ кодирано съобщениие $\beta$, което се декодира поне по два различни начина – например $a_{i_1}, a_{i_2}, \ldots, a_{i_r}$ и $a_{j_1}, a_{j_2}, \ldots, a_{j_s}$. Очевидно $\exists k, 1 \leq k \leq \min(r, s)$, такова че $a_{i_1} = a_{j_1}, \cdots a_{i_{k-1}} = a_{j_{k-1}}, a_{i_k} \neq a_{j_k}$ (вж. Фиг. 6.1).

Тъй като $\kappa_{i_k}$ и $\kappa_{j_k}$ започват от едно и също място на $\beta$, единствената възможност да са различни е $d(\kappa_{i_k}) \neq d(\kappa_{j_k})$. Нека (без ограничение на



Фигура 6.1: Разделимост на префиксен код.

общността) $d(\kappa_{i_k}) < d(\kappa_{j_k})$. Тогава $\kappa_{i_k}$ е префикс на $\kappa_{j_k}$, в противоречие с префиксността на кода. Следователно, допускането ни е невярно и кодът е разделим. □

Обратното твърдение не е вярно. Например, кодът $\kappa_a = 01, \kappa_b = 00, \kappa_c = 0111$ за азбуката $A = \{a, b, c\}$ не е префиксен, но е разделим. Въпреки това, както ще видим по-долу, можем да се ограничим с разглеждането само на префиксни кодове.

**Теорема 6.1.2 (Неравенство на МакМилан-Крафт)** *Нека $\kappa : A \to B^+$ е разделим код, $|A| = n, |B| = m$ и $d(\kappa_i) = l_i$. Тогава $\sum_{i=1}^{n} m^{-l_i} \leq 1$.*

***Доказателство.*** Да означим с $C$ константата $\sum_{i=1}^{n} m^{-l_i}$. Нека $t \in N, t \neq 0$. Да пресметнем $C^t$:

$$\begin{aligned} C^t &= (\sum_{i=1}^{n} m^{-l_i})^t = \\ &= (\sum_{i_1=1}^{n} m^{-l_{i_1}})(\sum_{i_2=1}^{n} m^{-l_{i_2}}) \cdots (\sum_{i_t=1}^{n} m^{-l_{i_t}}) = \\ &= \sum_{\substack{\forall i_1, i_2, \ldots, i_t \\ 1 \leq i_j \leq n}} m^{-(l_{i_1} + l_{i_2} + \cdots + l_{i_t})}. \end{aligned}$$

Да означим с $L(a_{i_1}, a_{i_2}, \ldots, a_{i_t}) = \sum_{k=1}^{t} l_{i_k}$ - дължината на кодираното съобщение $\kappa_{i_1}\kappa_{i_2}\ldots\kappa_{i_t}$. Тогава последната сума добива вида

$$\sum_{\forall a_{i_1}, a_{i_2}, \ldots, a_{i_t}} m^{-L(a_{i_1}, a_{i_2}, \ldots, a_{i_t})}$$

Ако с $M$ означим $\max_{i=1,2,\ldots,n} \kappa_i$, като имаме предвид, че $d(\kappa_i) \geq 1$, получаваме за $L(a_{i_1}, a_{i_2}, \ldots, a_{i_t})$, че $t \leq L(a_{i_1}, a_{i_2}, \ldots, a_{i_t}) \leq Mt$, а последната сума преписваме във вида

$$\sum_{r=t}^{Mt} N(t, r) m^{-r},$$



където с $N(t, r)$ сме означили броя на кодираните съобщения $\kappa_{i_1}, \kappa_{i_2}, \ldots, \kappa_{i_t}$ с дължина $L(a_{i_1}, a_{i_2}, \ldots, a_{i_t}) = r$. Тъй като $\kappa$ е разделим код, очевидно е, че $N(t, r) \leq m^r$, защото $m^r$ е броят на различните думи с дължина $r$ от $B^+$. Ако допуснем противното, тогава от Принципа на Дирихле ще следва, че има поне две съобщения с $t$ букви, които се кодират еднакво с $r$ букви. Следователно

$$\begin{aligned} C^t &= \sum_{r=t}^{Mt} N(r, t) m^{-r} \leq \sum_{r=t}^{Mt} m^r m^{-r} = \\ &= \sum_{r=t}^{Mt} 1 = (M-1)t + 1. \end{aligned}$$

С други думи $\forall t \in N, t > 0$ получихме $C^t \leq (M-1)t + 1$, а това е възможно само при $C \leq 1$, тъй като при $C > 1$ показателната функция $C^t$ расте по-бързо от линейната $(M-1)t+1$. Следователно $\sum_{i=1}^{n} k^{-l_i} \leq 1$. □

**Теорема 6.1.3** *За всеки разделим код $\kappa : A \to B^+, |A| = n, |B| = m$ с дължини на кодовите думи $l_1 \leq l_2 \leq \cdots \leq l_n$ съществува префиксен код $\kappa'$, думите на който имат същите дължини.*

***Доказателство.*** Без ограничение на общността можем да смятаме, че $B = \{0, 1, \ldots, m-1\}$. Всяко дробно число $q, 0 \leq q < 1$ има единствено представяне (евентуално безкрайно) като сума от отрицателни степени на $m$ ($m$-ично представяне на $q$)

$$q = c_1 m^{-1} + c_2 m^{-2} + \cdots + c_i m^{-i} + \cdots, 0 \leq c_i \leq m-1.$$

Да образуваме следната последователност от $n$ числа

$$\begin{aligned} q_1 &= 0 \\ q_2 &= m^{-l_1} \\ &\cdots \\ q_i &= m^{-l_1} + m^{-l_2} + \cdots + m^{-l_{i-1}} \\ &\cdots \\ q_n &= m^{-l_1} + m^{-l_2} + \cdots + m^{-l_{n-1}} \end{aligned}$$

Очевидно $\forall i = 1, 2, \ldots, n, q_i \geq 0$, тъй като е сума от положителни числа и $q_i < 1$ от неравенството на МакМилан-Крафт, което е в сила за

разделимия код $\kappa$. Очевидно, ако $j > i$, то

$$q_j - q_i = m^{-l_i} + m^{-l_{i+1}} + \cdots + m^{-l_{j-1}}$$

е положително дробно число и поради наредбата на дължините на кодовите думи първата различна от 0 цифра в $m$-ичното му представяне е не по-надясно от позиция $l_i$ (броени от $m$-ичната точка надясно).

Образуваме кода $\kappa'$ по следния начин: за $\kappa'_i$ избираме първите $l_i$ цифри в дробната част на $k$-ичното представяне на $q_i$. Ще покажем, че $\kappa'$ е префиксен и тъй като дължините на думите му съвпадат с тези на $\kappa$, това е търсеният код.

Да допуснем, че $\kappa'$ не е префиксен. Тогава $\exists \kappa_i, \kappa_j, i < j$ такива, че $\kappa_i$ е префикс на $\kappa_j$, т.е. първите $l_i$ букви ($m$-ични цифри) на $\kappa_j$ съвпадат с буквите на $\kappa_i$. Но това е противоречие с пресметнатия вече вид на положителното число $q_j - q_i$, тъй като при направеното предположение, при пресмятане на $q_j - q_i$ първата различна от нула $k$-ична цифра на разликата не може да се появи по-наляво от $(l_i + 1)$-ва позиция (числата съвпадат в първите си $l_i$ позиции). □

Забележете, че равномерните кодове са префиксни и следователно осигуряват еднозначност на декодирането. В повечето случаи при неравномерните кодове ни интересуват дължините на кодовите думи, затова във всеки такъв случай можем да се задоволим със съответния префиксен аналог на разделимия, но не префиксен код, който ни интересува. Още повече, че в редица случаи префиксността е свойство, от което можем да извлечем допълнителни ползи.

## 6.2 Оптимално побуквено кодиране

В този раздел ще разгледаме една задача за побуквена компресия на информацията в следната постановка: Дадена е съвкупност от текстове (думи над изходната азбука $A$). Например, програмите написани на даден алгоритмичен език или художествените текстове, написани на български език. Искаме да построим разделим побуквен код $\kappa : A \to \{0,1\}^+$ такъв, че ако кодираме всички текстове от съвкупността с този код, сумарната дължина на получените кодирани текстове да не е по-голяма от сумарната дължина на текстовете, кодирани с произволен друг разделим код $\kappa' : A \to \{0,1\}^+$.

Тъй като в общия случай съвкупността от текстове не е крайна, такава задача може да се решава само със статистически подход.

Нека $\alpha \in A^+$ е произволен текст от съвкупността. Да означим с $\#_i\alpha$ броя на срещанията на $a_i \in A$ в $\alpha$, а с $\#_i\alpha/d(\alpha)$ – честотата на срещане



на $a_i$ в $\alpha$. Очевидно

$$\sum_{i=1}^{n} \frac{\#_i\alpha}{d(\alpha)} = \frac{1}{d(\alpha)} \sum_{i=1}^{n} \#_i\alpha = 1$$

Когато меним $\alpha$ в съвкупността от текстове, ще се мени и стойността на $\#_i\alpha/d(\alpha)$. За някои добре дефинирани съвкупности от текстове, обаче, $\#_i\alpha/d(\alpha)$ варира в много тесни граници. Например, при художествените и публицистични текстове на даден естествен език. Таблица 6.1 показва честотите на буквите от кирилицата, употребени в този текст. Това ни позволява да въведем вероятностно пространство с елементарни събития буквите на азбуката по следния начин. На всяка буква $a_i$ от азбуката $A, |A| = n$, да съпоставим вероятност $p_i, 0 < p_i < 1, i = 1, 2, \ldots, n$, $\sum_{i=1}^{n} p_i = 1$, като $p_i$ е колкото може по-близо до реалните стойности на $\#_i\alpha/d(\alpha)$. Когато $\alpha$ приема различни стойности от множеството на текстовете, тогава $p_i d(\alpha)$ е очакваният брой букви $a_i$ в текста $\alpha$.

| | | | | | |
|---|---|---|---|---|---|
| А – 12.68 | Б – 1.08 | В – 4.77 | Г – 1.01 | Д – 3.33 | Е – 9.64 |
| Ж – 0.76 | З – 2.21 | И – 8.54 | Й – 0.48 | К – 3.43 | Л – 3.04 |
| М – 3.30 | Н – 8.17 | О – 9.05 | П – 2.50 | Р – 4.67 | С – 4.41 |
| Т – 7.80 | У – 1.44 | Ф – 0.78 | Х – 0.35 | Ц – 0.87 | Ч – 1.62 |
| Ш – 0.18 | Щ – 0.60 | Ъ – 1.56 | Ь – 0.02 | Ю – 0.10 | Я – 1.61 |

Таблица 6.1: Честоти на буквите от кирилицата (в проценти).

***Дефиниция.*** Нека $A = \{a_1, a_2, \ldots, a_n\}$ е крайна азбука, а $p_1, p_2, \ldots, p_n, 0 < p_i < 1, \sum_{i=1}^{n} p_i = 1$ са вероятностите за поява на съответните букви в думите от езика $T \subseteq A^*$. Тогава $\mathcal{I}_T = (A; p_1, p_2, \ldots, p_n)$ ще наричаме *източник на информация* или *източник*.

Изключваме възможността $p_i = 0$, защото това означава, че на практика $a_i$ не се среща въобще в текст от $T$ и можем да считаме, че не е елемент на азбуката $A$. Изключваме и възможността $p_i = 1$, което пък означава, че всички текстове от $T$ са съставени само от една буква, което е много частен и тривиален случай.

Сега нека $\kappa : A \to \{0,1\}^+$ е разделим побуквен код за $A, |A| = n$, с дължини на кодовите думи $l_1, l_2, \ldots, l_n$, а $\alpha$ е съобщение от $T$. Очевидно $l_i p_i d(\alpha)$ е очакваният брой букви, които $a_i$ внася в дължината на кодираното съобщение $\kappa^*(\alpha)$. Сумата

$$\sum_{i=1}^{n} l_i p_i d(\alpha) = d(\alpha) \sum_{i=1}^{n} l_i p_i$$

е очакваната дължина на $\kappa^*(\alpha)$. И така, очакваната дължина на всяко кодирано съобщение $\kappa^*(\alpha)$ зависи от $d(\alpha)$ и константата $\sum_{i=1}^n l_i p_i$, определена еднозначно от източника $\mathcal{I}_T$ и кода $\kappa$. Статистически погледнато, за да имат кодираните съобщения сумарно минимална дължина, е необходимо и достатъчно (при фиксиран $\mathcal{I}_T$) да изберем $\kappa$ така, че $\sum_{i=1}^n l_i p_i$ да е минималната възможна.

***Дефиниция.*** Нека $\mathcal{I}_T = (A; p_1, p_2, \ldots, p_n)$ е източник на информация, а $\kappa : A \to \{0,1\}^+$ е разделим побуквен код за $\mathcal{I}_T$ с дължини на кодовите думи $l_1, l_2, \ldots, l_n$. Сумата $C(\kappa, \mathcal{I}_T) = \sum_{i=1}^n l_i p_i$ наричаме *цена на кода $\kappa$ за източника $\mathcal{I}_T$*. Тъй като най-често източникът се подразбира, за по-кратко ще пишем $C(\kappa)$ и ще я наричаме просто *цена на кода*.

***Дефиниция.*** Нека $\mathcal{I}_T = (A; p_1, p_2, \ldots, p_n)$ е източник на информация, а $\kappa^o : A \to \{0,1\}^+$ е разделим код за $\mathcal{I}_T$, такъв че $\forall$ друг разделим код $\kappa$ за $\mathcal{I}_T$ е в сила $C(\kappa^o) \leq C(\kappa)$. Тогава $\kappa^o$ наричаме *оптимален (побуквен) код* за източника $\mathcal{I}_T$.

От изложеното до тук не е ясно дали съществува оптимален код за даден източник. Затова първо ще докажем следната

**Теорема 6.2.1** *За всеки източник на информация $\mathcal{I} = (A; p_1, p_2, \ldots, p_n)$ съществува оптимален код $\kappa^o$.*

***Доказателство.*** Нека $\kappa : A \to \{0,1\}^+$ е разделим код за $\mathcal{I}$ (например, равномерен код с подходяща дължина) и нека $l_1, l_2, \ldots, l_n$ са дължините на кодовите му думи. Ще покажем, че ако $\kappa^o$ съществува и $l_1^o, l_2^o, \ldots, l_n^o$ са дължините на кодовите му думи, то са в сила неравенствата:

$$1 \leq l_i^o \leq C(\kappa)/p_i, i = 1, 2, \ldots, n.$$

Наистина, всички дължини са $\geq 1$, а ако допуснем, че за някое $i$ е в сила $l_i^o > C(\kappa)/p_i$, получаваме

$$C(\kappa^o) = \sum_{j=1}^n l_j^o p_j > l_i^o p_i > C(\kappa) p_i / p_i = C(\kappa),$$

т.е. $C(\kappa^o) > C(\kappa)$, което противоречи на оптималността на $\kappa^o$.

Освен това, $\kappa^o$ трябва да е разделим, следователно $l_i^o$ трябва да удовлетворяват неравенството на МакМилан-Крафт при $m = 2$. Получаваме следната система от $2n + 1$ неравенства за неизвестните $l_1^o, l_2^o, \ldots, l_n^o$:

$$\left| \begin{array}{l} 1 \leq l_i^o \leq C(\kappa)/p_i, i = 1, 2, \ldots, n \\ \sum_{i=1}^n 2^{-l_i} \leq 1 \end{array} \right.$$



Очевидно е, че има краен брой възможности целите $l_1^o, l_2^o, \ldots, l_n^o$ да удовлетворяват системата. Освен това, $l_1, l_2, \ldots, l_n$ е решение на системата неравенства, защото кодът $\kappa$ е разделим и удовлетворява неравенството на МакМилан-Крафт, $l_i \geq 1$, а ако допуснем, че $l_i > C(\kappa)/p_i$, получаваме

$$C(\kappa) = \sum_{j=1}^n l_j p_j > l_i p_i > C(\kappa) p_i / p_i = C(\kappa),$$

т.е. $C(\kappa) > C(\kappa)$, което е противоречие.

Получихме, че системата неравенства има поне едно решение и тъй като има краен брой решения, съществува решение $l_1^o, l_2^o, \ldots, l_n^o$, при което $\sum_{j=1}^n l_j^o p_j$ е минимална. □

Ако знаем $\{l_i^o\}$, за да построим оптимален (префиксен) код е достатъчно да повторим процедурата от Теорема 6.1.3 с намерените дължини $l_1^o, l_2^o, \ldots, l_n^o$. Това не е най-добрият начин за построяване на оптимален код, тъй като решаването на системата в общия случай може да стане само с пълно изчерпване.

Да се спрем на следния изключително важен факт. При зададен източник $I = (A; p_1, p_2, \ldots, p_n)$ цената на оптималния побуквен код не може да бъде по-малка от една константа, еднозначно определена от източника.

***Дефиниция.*** Нека $I = (A; p_1, p_2, \ldots, p_n)$ е източник на информация. Сумата $H(I) = \sum_{i=1}^n p_i \log_2 \frac{1}{p_i}$ наричаме *ентропия* на $I$.

**Теорема 6.2.2 (Кл. Шенън)** *За всеки разделим код $\kappa$ на източника $I = (A; p_1, p_2, \ldots, p_n)$, с дължини на кодовите думи $l_1, l_2, \ldots, l_n$ е в сила $C(\kappa) \geq H(I)$.*

***Доказателство.*** Да пресметнем разликата

$$H(I) - C(\kappa) = \sum_{i=1}^n p_i \log_2 \frac{1}{p_i} - \sum_{i=1}^n p_i l_i = \sum_{i=1}^n p_i (\log_2 \frac{1}{p_i} - \log_2 2^{l_i}) =$$

$$= \sum_{i=1}^n p_i \log_2 \frac{2^{-l_i}}{p_i} = \frac{1}{\ln 2} \sum_{i=1}^n p_i \ln \frac{2^{-l_i}}{p_i}$$

Ще използваме известното от анализа неравенство $\ln x \leq x - 1$. Получаваме

$$H(I) - C(\kappa) =$$

$$= \frac{1}{\ln 2}\sum_{i=1}^{n} p_i \ln \frac{2^{-l_i}}{p_i} \leq \frac{1}{\ln 2}\sum_{i=1}^{n} p_i\left(\frac{2^{-l_i}}{p_i} - 1\right) =$$

$$= \frac{1}{\ln 2}\left(\sum_{i=1}^{n} p_i \frac{2^{-l_i}}{p_i} - \sum_{i=1}^{n} p_i\right) = \frac{1}{\ln 2}\left(\sum_{i=1}^{n} 2^{-l_i} - 1\right) \leq \frac{1}{\ln 2}(1-1) = 0.$$

Следователно $H(I) \leq C(\kappa)$. □

В най-лошия случай, $p_1 = p_2 = \cdots = p_n = 1/n$, ентропията е равна на сумата $\sum_{i=1}^{n} \frac{1}{n}\log_2 n = \log_2 n$. Тъй като минималната дължина на равномерен код за кодиране на азбуката $A, |A| = n$ е $\lceil \log_2 n \rceil$ бита, то в този случай компресия не е възможна.

Цената на оптималния код е ограничена отгоре, както показва следващата

**Теорема 6.2.3** *За оптималния код $\kappa^o$ на източника $I = (A; p_1, p_2, \ldots, p_n)$ е в сила $C(\kappa^o) \leq H(I) + 1$.*

***Доказателство.*** Нека $l_i = \lceil \log_2 \frac{1}{p_i} \rceil$, $i = 1, 2, \ldots, n$. Получаваме

$$\sum_{i=1}^{n} 2^{-l_i} = \sum_{i=1}^{n} 2^{-\lceil \log_2 \frac{1}{p_i} \rceil} \leq \sum_{i=1}^{n} 2^{-\log_2 \frac{1}{p_i}} = \sum_{i=1}^{n} 2^{\log_2 p_i} =$$

$$\sum_{i=1}^{n} p_i = 1.$$

Следователно $l_i, i = 1, 2, \ldots, n$, удовлетворяват неравенството на МакМилан-Крафт и можем да построим (с процедурата от Теорема 6.1.3) префиксен код $\kappa$ с дължини на кодовите думи $l_1, l_2, \ldots, l_n$. Но тогава

$$C(\kappa^o) \leq C(\kappa) = \sum_{i=1}^{n} p_i l_i = \sum_{i=1}^{n} p_i \lceil \log_2 \frac{1}{p_i} \rceil \leq \sum_{i=1}^{n} p_i\left(\log_2 \frac{1}{p_i} + 1\right) =$$

$$= \sum_{i=1}^{n} p_i \log_2 \frac{1}{p_i} + \sum_{i=1}^{n} p_i = H(I) + 1. \ \square$$

Исторически един от първите опити да се строят кодове, близки до оптималните е следният

**Алгоритъм на Шенън-Фано.**

**Дадено:** Източник $I = (A; p_1, p_2, \ldots, p_n)$.

**Резултат:** Префиксен код за $I$.

**Процедура:** 1. Нека вероятностите са сортирани по големина и нека $r = 1$ и $m = n$.



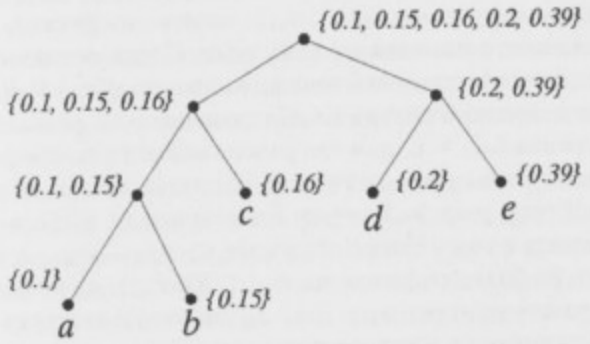
2. Разбиваме множеството от вероятности на две части $(p_1, p_2, \ldots, p_j)$ и $(p_{j+1}, p_{j+2}, \ldots, p_m)$ така, че $|\sum_{i=1}^{j} p_i - \sum_{i=j+1}^{m} p_i|$ да е минимална.

3. На буквите, чиито вероятности са попаднали в първата група, присвояваме $r$-та кодова буква 0, а на останалите – $r$-та кодова буква 1.

4. За всяка група от вероятности с един елемент процедурата завършва, а за всяка от останалите групи означаваме с $m$ броя на елементите ѝ, след което се обръщаме рекурсивно към стъпка 2 с $r = r + 1$. □

Алгоритъмът на Шенън-Фано по естествен начин съпоставя на всеки източник двоично кореново дърво с наредба на синовете. Всички върхове с изключение на корена, съответстват на подмножествата от изходни букви, които имат еднакви първи $r$ кодови букви в построения от алгоритъма код. Ляв и десен син на съответния връх са подмножествата от изходни букви на съпоставеното му множество, които имат $(r+1)$-ва кодова буква 0 и 1, съответно.

Да разгледаме работата на алгоритъма на Шенън-Фано върху източника $I = (\{a, b, c, d, e\}; 0.1, 0.15, 0.16, 0.2, 0.39)$. Дървото на Фиг. 6.2 илюстрира резултата от работата на алгоритъма. Търсеният код е $\kappa_a = 000, \kappa_b = 001, \kappa_c = 01, \kappa_d = 10, \kappa_e = 11$. Цената му е $C(\kappa) = 3(0.1+0.15) + 2(0.16 + 0.2 + 0.39) = 0.75 + 1.5 = 2.25$.




Фигура 6.2: Дърво на Шенън-Фано.

Алгоритъмът, предложен от Шенън и Фано, не строи оптимални кодове. Ще покажем как може да бъде построен оптимален код на зададен източник.

**Теорема 6.2.4** *Нека $I = (A; p_1 \geq p_2 \geq \ldots \geq p_n)$ е източник на информация. Съществува оптимален префиксен код $\kappa^o : A \to \{0,1\}^+$ за $I$ с дължини на кодовите думи $l_1^o \leq l_2^o \leq \cdots \leq l_{n-1}^o = l_n^o$, като $\kappa_{n-1}^o = \alpha 0, \kappa_n^o = \alpha 1$.*

***Доказателство.*** Нека $\kappa$ е оптимален префиксен код на $I$, с дължини на кодовите думи $l_1, l_2, \ldots l_n$. Да допуснем, че $i < j$, но $l_i > l_j$. Ако $p_i = p_j$, можем да разменим $\kappa_i$ с $\kappa_j$ без да променяме цената на кода и в получения код $\kappa'$ ще имаме $l_i' < l_j'$. Ако $p_i > p_j$, не е възможно $l_i > l_j$, защото след като направим същата размяна и построим кода $\kappa'$, ще получим

$$\begin{aligned} C(\kappa) - C(\kappa') &= l_i p_i + l_j p_j - l_i' p_i - l_j' p_j = \\ &= l_i p_i + l_j p_j - l_j p_i - l_i p_j = \\ &= (l_i - l_j)(p_i - p_j) > 0. \end{aligned}$$

т.е. $C(\kappa) > C(\kappa')$, което е в противоречие с оптималността на $\kappa$.

И така, съществува оптимален код, в който $l_1 \leq l_2 \leq \cdots \leq l_n$. Да допуснем, че $l_{n-1} < l_n$. Тогава можем да заменим $\kappa_n$ с $\kappa_n'$, такава че $\kappa_n = \kappa_n' \kappa_n''$, $d(\kappa_n') = l_{n-1}$. Никоя от кодовите думи $\kappa_1, \kappa_2, \ldots, \kappa_{n-1}$ не е префикс на $\kappa_n'$, защото би била префикс и на $\kappa_n$. Обратно, $\kappa_n'$ не е префикс на никоя от кодовите думи $\kappa_1, \kappa_2, \ldots, \kappa_{n-1}$, защото дължината ѝ е не-малка от дължините на всяка от тези думи. Следователно полученият код е префиксен и има по-ниска цена от $\kappa$, защото $d(\kappa_n') < d(\kappa_n)$. Това е противоречие с оптималността на $\kappa$. Следователно $l_{n-1} = l_n$. Ако между думите с дължина $l_{n-1} = l_n$ има две различаващи се по последния бит – $\alpha 0$ и $\alpha 1$, правим ги $\kappa_{n-1}$ и $\kappa_n$ и търсеният оптимален код $\kappa^o$ е построен. В противен случай, нека $\kappa_{n-1} = \alpha a$. Заместваме $\kappa_n$ с $\alpha \bar{a} = \kappa_n'$ – дума, която не се среща в кода. Никоя от думите $\kappa_1, \kappa_2, \ldots, \kappa_{n-1}$ не е префикс на $\kappa_n'$, защото би била префикс и на $\kappa_{n-1}$. Обратно, $\kappa_n'$ не е префикс на никоя от кодовите думи $\kappa_1, \kappa_2, \ldots, \kappa_{n-1}$, защото дължината ѝ не е по-малка от дължините на всяка от тези думи. С евентуално разместване на $\alpha a$ и $\alpha \bar{a}$ получаваме търсения оптимален код $\kappa^o$. □

**Теорема 6.2.5** *Нека $\kappa = \{\kappa_1, \kappa_2, \ldots, \kappa_n\}$, с дължини на кодовите думи $l_1, l_2, \ldots, l_n$ е оптимален код за източника $I = (A; p_1 \geq p_2 \geq \ldots \geq p_n)$ и $p_i = q_1 + q_2$ такива, че $p_n \geq q_1 \geq q_2$. Тогава $\kappa' = \{\kappa_1, \ldots, \kappa_{i-1}, \kappa_{i+1}, \ldots, \kappa_n, \kappa_i 0, \kappa_i 1\}$ е оптимален код за източника $I' = (B; p_1 \geq \cdots \geq p_{i-1} \geq p_{i+1} \geq \cdots \geq p_n \geq q_1 \geq q_2)$, където $B$ е произволна азбука с $n+1$ букви.*



***Доказателство.*** От конструкцията на $\kappa'$ получаваме, че

$$\begin{aligned} C(\kappa') - C(\kappa) &= q_1(l_i + 1) + q_2(l_i + 1) - p_i l_i = \\ &= (q_1 + q_2)(l_i + 1) - p_i l_i = \\ &= p_i(l_i + 1) - p_i l_i = p_i. \end{aligned}$$

Нека $\kappa''$ е оптимален префиксен код за източника $I'$. От Теорема 6.2.4 следва, че можем да изберем $\kappa''$ така, че кодовите думи на $\kappa''$ да са с дължини $l_1'' \leq l_2'' \leq \cdots \leq l_n'' = l_{n+1}''$ и $\kappa'' = \{\kappa_1'', \ldots, \kappa_{i-1}'', \kappa_{i+1}'', \ldots, \kappa_n'', \kappa_i'' 0, \kappa_i'' 1\}$ Тогава построяваме префиксния код $\kappa'''$ за източника $I - \kappa''' = \{\kappa_1'', \ldots, \kappa_{i-1}'', \kappa_i'', \kappa_{i+1}'', \ldots, \kappa_n''\}$. Аналогично (както за $\kappa'$ и $\kappa$) имаме $C(\kappa'') - C(\kappa''') = p_i$. Но $\kappa$ е оптимален за $I$, следователно $C(\kappa) \leq C(\kappa''')$. Затова

$$C(\kappa') = C(\kappa) + p_i \leq C(\kappa''') + p_i = C(\kappa'')$$

и тъй като $\kappa''$ е оптимален, то и $\kappa'$ е оптимален за източника $I'$. □

Така доказаната Теорема ни дава следната процедура за построяване на оптимален код:

**Алгоритъм на Хафмън.**

**Дадено:** Източник $I = (A; p_1 \geq p_2 \geq \ldots \geq p_n)$

**Резултат:** Оптимален код $\kappa$ за $I$

**Процедура:** 1) Започвайки от $I$, последователно строим източниците $I_{n-1}, I_{n-2}, \ldots, I_2$ с $n-1, n-2, \ldots, 2$ букви, като всеки път събираме най-малките две вероятности на текущия източник, заместваме тези две вероятности със сбора и запомняме от кои две стойности се е получил.

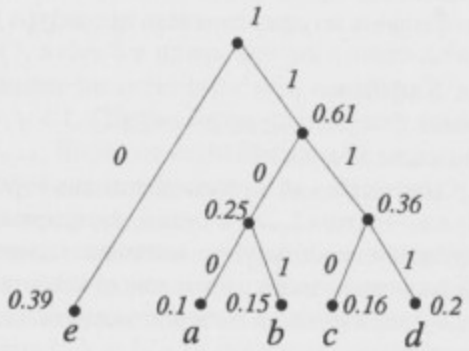
2) За източника $I_2$, каквито и да са вероятностите, оптималният код се състои от думите 0 и 1.

3) Започвайки от оптималния код за източника $I_2$, последователно строим оптимални кодове за източниците $I_3, I_4, \ldots, I_{n-1}, I$, като думата, избрана за вероятността-сбор на двете най-малки от предния източник разширяваме отдясно с 0 и 1, и присвояваме получените две думи на двете най-малки вероятности на този източник. Всички останали думи остават присвоени на старите вероятности. □

С алгоритъма на Хафмън естествено се свързва двоично кореново дърво с наредба на синовете, подобно на дървото от Алгоритъма на Шенън-Фано. На върховете му приписваме по една вероятност (на стъпка 1 на процедурата) и по една дума (на стъпка 2 на процедурата) по следните правила: Започваме с $n$ върха, като $n$ отделни дървета с началните вероятности. Всеки път, когато сумираме двете най-малки вероятности, приписани на корени $r_1$ и $r_2$ на дърветата, избираме нов връх – $r$,

на който приписваме получената сума, като дърветата с корени $r_1$ и $r_2$ правим ляво и дясно поддърво на новия връх. Когато получим само две поддървета, процедурата завършва и в корена на полученото двоично дърво е приписано числото 1. В обратната посока: за всеки нелист на дървото, да надпишем с 0 реброто към левия му, а с 1 – реброто към десния му син. Кодовата дума $\kappa_i$ на оптималния *код на Хафмън*, съответна на буквата $a_i$ с вероятност $p_i$, е думата от $\{0,1\}^*$, която се получава по пътя от корена на дървото до листа надписан с $a_i$.

Да разгледаме работата на алгоритъма на Хъфман върху източника $I = (\{a,b,c,d,e\}; 0.10, 0.15, 0.16, 0.20, 0.39)$, който използвахме за пример и при алгоритъма на Шенън-Фано. Дървото на Фиг. 6.3 илюстрира получения резултат. Търсеният оптимален код е $\kappa_a^o = 100, \kappa_b^o = 101, \kappa_c^o = 110, \kappa_d^o = 111, \kappa_e^o = 0$. Цената му е $C(\kappa^o) = 3(0.1 + 0.15 + 0.16 + 0.2) + 1(0.39) = 2.22$. Макар и съвсем малко, кодът на Хафмън е по-добър от кода на Шенън-Фано за същия източник.




Фигура 6.3: Дърво на Хафмън.

Кореновите дървета, породени от алгоритмите на Шенън-Фано и Хафмън, решават сравнително просто и въпроса за декодиране. Започваме от началото на кодираното съобщение и от корена на дървото. Ако текущата буква на кодираното съобщение е 0, преминаваме към левия син на текущия връх, в противен случай – към десния син. Ако върхът, в който сме попаднали не е лист, продължаваме обхождането на дървото със следващата буква от кодираното съобщение. В противен случай, декодираме частта от кодираното съобщение в буквата, приписана на листа. Ако кодираното съобщение е завършило, то и декодирането е завършило. В противен случай преминаваме към корена на дървото и



повтаряме описаните по-горе действия.

Например, кодираното с кода на Хафман от Фиг.6.3 съобщение

01110110101

описаната процедура декодира в изходното съобщение *edecb*.

За съжаление, статистическите характеристики на някои източници не благоприятстват компресията на породените от тях съобщения. Типичен пример са изпълнимите компютърни програми, разглеждани като думи над азбуката на 256-те различни байта на компютърната памет. При тези съобщения вероятностите на различните байтове са почти изравнени и алгоритъмът на Хафман почти не намалява обема на текста. В същото време известните компресиращи програми успяват да съкратят, и то доста, такива текстове. Създава се впечатлението, че нещо не е наред в теорията на оптималните кодове. Всъщност теорията е абсолютно вярна, но е приложена върху неподходящ източник.

За случаи като горния е необходимо да се опитат други източници за същите съобщения, например за азбуки да се вземат декартовите степени на азбуката, т.е. различните наредени двойки от букви, тройки от букви и т.н. или пък доста често срещащи се последователности от букви с различна дължина.

За пример да разгледаме съобщението с 45 букви:

```
if(a[i]>a[i+1]){b=a[i];a[i]=a[i+1];a[i+1]=b;}
```

за което получаваме следните честоти на участващите букви:

| $i$ | $f$ | ( | $a$ | [ | + | 1 | ] | > | ) | $b$ | = | ; | { | } |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\frac{7}{45}$ | $\frac{1}{45}$ | $\frac{1}{45}$ | $\frac{6}{45}$ | $\frac{6}{45}$ | $\frac{3}{45}$ | $\frac{3}{45}$ | $\frac{6}{45}$ | $\frac{1}{45}$ | $\frac{1}{45}$ | $\frac{2}{45}$ | $\frac{3}{45}$ | $\frac{3}{45}$ | $\frac{1}{45}$ | $\frac{1}{45}$ |

За ентропията на този източник получаваме:

$$6\frac{1}{45}\log_2\frac{45}{1} + 1\frac{2}{45}\log_2\frac{45}{2} + 4\frac{3}{45}\log_2\frac{45}{3} + 3\frac{6}{45}\log_2\frac{45}{6} + 1\frac{7}{45}\log_2\frac{45}{7} =$$
$$= \frac{6}{45}5.49 + \frac{2}{45}4.49 + \frac{12}{45}3.91 + \frac{18}{45}2.91 + \frac{7}{45}2.68 = 3.58$$

Нека сега променим източника на същото съобщение, като обявим някои комбинации от букви за букви на новия източник. Новата азбука е с 9 букви и следните честоти в промененото съобщение с 18 букви:

| $if($ | $a[i+1]$ | $a[i]$ | $>$ | $)\{$ | $b$ | $=$ | $;$ | $\}$ |
|---|---|---|---|---|---|---|---|---|
| $\frac{1}{18}$ | $\frac{3}{18}$ | $\frac{3}{18}$ | $\frac{1}{18}$ | $\frac{1}{18}$ | $\frac{2}{18}$ | $\frac{3}{18}$ | $\frac{3}{18}$ | $\frac{1}{18}$ |

За ентропията на този източник получаваме:

$$4\frac{1}{18}\log_2\frac{18}{1}+1\frac{2}{18}\log_2\frac{18}{2}+4\frac{3}{18}\log_2\frac{18}{3}=\frac{4}{18}4.16+\frac{2}{18}3.17+\frac{12}{18}2.58=3.00$$

Така с елементарни промени построихме източник с по-ниска ентропия от началния и повишихме възможността за компресиране от 45 на около 37 процента от първоначалната дължина, която би се получила при използване на стандартна 8-битова кодова таблица.

Горният пример дава, макар и опростена, представа за един от пътищата, по които се търси възможност за увеличаване степента на компресия при зададен текст. Съвременните програмни и хардуерни средства за компресия извършват детайлни статистически анализи на текстовете, за да намерят източник с колкото може по-ниска ентропия и така постигат намаляване на обема на текста близко до оптималното. Намирането на оптимален източник е трудна задача. Пълното ѝ решаване изисква големи ресурси и не трябва да се очаква да бъде направено за разумно време.

## 6.3 Шумозащитно кодиране

Електронната обработка на информацията предполага предаването ѝ по канали за връзка, в които са възможни повреди на предаваната информация – *шум*. Ако се знае характерът на шума в канала, тогава е възможно, с подходящо кодиране, да се установи, че при предаването са настъпили нежелани изменения. При определени условия е възможно тези изменения да се отстранят. За описване на поведението на канала ще използваме характеристика от вида „каналът изменя не повече от $t$ букви на всеки $n$“, което накратко ще записваме $\leq t/n$. За простота и



тук ще се спрем на двоични кодове, които са и най-често използваните в практиката.

Да въведем някои основни понятия от теорията на *шумозащитните кодове*, наричани още *кодове, откриващи и поправящи грешки*. Ще разглеждаме само равномерни кодове $\kappa : A \to \{0,1\}^n$. Ще означаваме с $n(\kappa)$ дължината на кодовите думи и ще я наричаме *дължина на кода*. Ще означаваме с $K(\kappa)$ броя на кодовите думи (броя на буквите в $A$) и ще го наричаме *обем на кода* $\kappa$. В теорията на шумозащитното кодиране (особено при равномерни кодове) при задаването на кода ни интересува не толкова изображението $\kappa : A \to \{0,1\}^n$, колкото множеството на кодовите думи $C = \{\kappa_1, \kappa_2, \ldots, \kappa_K\}$. В такъв случай за дължината на кода $n(\kappa)$ пишем $n(C)$ или просто $n$, когато кодът се подразбира. Аналогично за обема, вместо $K(\kappa)$ пишем $K(C)$ или просто $K$, когато кодът се подразбира.

И така, нека $C \subseteq \{0,1\}^n$ е двоичен код с дължина $n$ и обем $K$. Отношението $V(C) = \frac{\log_2 K}{n}$ наричаме *скорост на кода*. В числителя на това отношение стои число, близко до броя на двоичните позиции, необходими и достатъчни за записване на думите на кой да е двоичен код с обем $K$, а в знаменателя – реалният брой използвани позиции. Затова скоростта на кода $C$ показва загубите на време, които ще понесем, заради защитните свойства на кода, получени чрез добавяне на излишък от информация.

За всеки от следващите примери ще считаме, че каналът, по който се изпращат данните е с характеристика $1 \leq h$.

Пример 1. Нека $m$ е цяло положително число такова, че $2 \leq 2m \leq h$. Образуваме кода $C_1 = \{\alpha\alpha | \alpha \in \{0,1\}^m\}$. Очевидно $n(C_1) = 2m, K(C_1) = 2^m$, следователно $V(C_1) = \frac{\log_2 2^m}{2m} = \frac{m}{2m} = \frac{1}{2}$. Нека е изпратена кодовата дума $\alpha\alpha$ и е получена думата $\beta, d(\beta) = 2m$. Разделяме $\beta$ на две думи с дължина $m$: $\beta = \alpha'\alpha''$. Ако няма грешки при предаването по канала това ще се установи от $\alpha' = \alpha''$ и $\alpha'\alpha''$ е изпратената дума. Казваме, че сме *декодирали* изпратеното съобщение. Ако пък каналът е направил една грешка (инвертирал е един бит), тогава $\alpha' \neq \alpha''$ и установяваме, че е допусната грешка, без да можем да определим коя от двете поддуми е вярна и коя е грешна. Тази ситуация наричаме *отказ от декодиране*.

***Дефиниция.*** Процедурата, при която (при условие, че каналът не е направил повече от допустимите грешки) можем да установим правилността или неправилността на полученото съобщение, но не можем да го поправим, ако е неправилно, наричаме *откриване на грешка (грешки)*.

Кодът $C_1$ открива една грешка при канал с характеристика $\leq 1/h$.

Пример 2. Нека $m$ е цяло положително число такова, че $3 \leq 3m \leq h$. Образуваме кода $C_2 = \{\alpha\alpha\alpha | \alpha \in \{0,1\}^m\}$. Очевидно $n(C_2) = 3m, K(C_2) = 2^m$, следователно $V(C_2) = \frac{\log_2 2^m}{3m} = \frac{m}{3m} = \frac{1}{3}$. Нека е изпратена кодовата дума $\alpha\alpha\alpha$ и е получена думата $\beta, d(\beta) = 3m$. Разделяме $\beta$ на три думи с дължина $m$: $\beta = \alpha'\alpha''\alpha'''$. Ако няма грешки при предаването по канала, това ще се установи от $\alpha' = \alpha'' = \alpha'''$ и $\alpha'\alpha''\alpha'''$ е изпратената дума. Казваме, че сме *декодирали* изпратеното съобщение. Ако пък каналът е направил една грешка (инвертирал е един бит), тогава очевидно точно една от трите поддуми се различава от другите две. Нека без ограничение на общността $\alpha' = \alpha''$, а $\alpha' \neq \alpha''' \neq \alpha''$. Отново можем да декодираме изпратената дума – изпратена е $\alpha'\alpha'\alpha'$ (или $\alpha''\alpha''\alpha''$). В този пример, ако каналът е имал поведение съответно на характеристиката му, декодираме полученото съобщение, независимо от това допусната ли е грешка или не.

***Дефиниция.*** Процедурата, при която (при условие, че каналът не е направил повече от допустимите грешки) можем да установим правилността или неправилността на полученото съобщение и да го поправим, ако е неправилно, наричаме *поправяне на грешка (грешки).*

Кодът $C_2$ поправя една грешка при канал с характеристика $\leq 1/h$.

Пример 3. Нека $m$ е цяло положително число, $m < h$. Образуваме кода $C_3 = \{\alpha a | \alpha \in \{0,1\}^m, a = wt(\alpha) (\mathrm{mod}\ 2)\}$. Очевидно $n(C_3) = m + 1, K(C_3) = 2^m$, следователно $V(C_3) = \frac{\log_2 2^m}{m+1} = \frac{m}{m+1} = 1 - \frac{1}{m+1}$, т.е. скоростта на този код е много по-добра от скоростта на $C_1$. Наричаме го *код с проверка на четност*. Аналогично можем да построим код с проверка на нечетност.

Нека е изпратена кодовата дума $\alpha a$ и е получена думата $\beta, d(\beta) = m + 1$. Проверяваме четността на $wt(\beta)$. При условие, че каналът не е допуснал грешка $wt(\beta) = 0 (\mathrm{mod}\ 2)$ и следователно изпратеното съобщение е $\beta$. Ако $wt(\beta) = 1 (\mathrm{mod}\ 2)$, някъде в $\beta$ е допусната грешка, но не е възможно да се установи кой е повреденият бит. И така, кодът с проверка на четност открива 1 грешка, както и $C_1$, при много по-висока скорост.

Пример 4. Нека $m$ е цяло положително число, $m^2 + 2m \leq h$. Образуваме кода

$$C_4 = \{\alpha b_1 \dots b_m c_1 \dots c_m | \alpha \in \{0,1\}^{m^2}, b_i, c_j \in \{0,1\}\},$$

като $\alpha = a_{11} \dots a_{1m} a_{21} \dots a_{2m} \dots a_{m1} \dots a_{mm}$, $b_i = wt(a_{i1} \dots a_{im}) (\mathrm{mod}\ 2)$, $i = 1, 2, \dots, m$, а $c_j = wt(a_{1j} \dots a_{mj}) (\mathrm{mod}\ 2), j = 1, 2, \dots, m$ (вж. Табл. 6.2). Очевидно $n(C_4) = m^2 + 2m, K(C_4) = 2^{m^2}$, следователно $V(C_4) =$



$\frac{\log_2 2^{m^2}}{m^2+2m} = \frac{m^2}{m^2+2m} = 1 - \frac{2}{m+2}$. Скоростта на този код е много по-добра от скоростта на $C_2$. Той е двумерен аналог на кода с проверка на четност. Аналогично можем да построим двумерен аналог на кода с проверка на нечетност.

| | | | | |
|---|---|---|---|---|
| $a_{11}$ | $a_{12}$ | $\cdots$ | $a_{1m}$ | $b_1$ |
| $a_{21}$ | $a_{22}$ | $\cdots$ | $a_{2m}$ | $b_2$ |
| $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ |
| $a_{m1}$ | $a_{m2}$ | $\cdots$ | $a_{mm}$ | $b_m$ |
| $c_1$ | $c_2$ | $\cdots$ | $c_m$ | |

Таблица 6.2: Двумерен код с проверка на четност.

Да изпратим по канала кодовата дума $\alpha b_1 \dots b_m c_1 \dots c_m$ и нека $\beta$ е получената дума. Да подредим битовете на $\beta$, както е показано на Табл. 6.2 и да извършим всички $2m$ проверки на четност. Ако каналът не е допуснал грешка, всички проверки ще завършат успешно и изпратената дума е $\beta$. Ако е допусната точно една грешка в $\alpha$, тогава точно в един ред и точно в един стълб ще се получи нарушаване на четността. Декодирането се състои в инвертиране на бита, намиращ се в съответния ред и стълб. Ако грешката е в добавените (*проверочните*) битове, точно в един ред или точно в един стълб проверката ще завърши неуспешно и декодирането е състои в инвертиране на съответния проверочен бит. Очевидно кодът $C_4$ поправя една грешка, както и $C_2$, но с много по-висока скорост.

От Примерите се вижда, че шумозащитните качества на един код се постигат с подходящ избор на някои от думите на $\{0,1\}^n$ за кодови думи. По отклонението от някоя кодова дума съдим за правилността или неправилността на получената дума.

В множеството $J_2^n$ на $n$-мерните двоични вектори въвеждаме известното ни вече Хеминговото разстояние $\rho : J_2^n \times J_2^n \to N$, $\rho(\alpha, \beta)$ е броят на различаващите се компоненти на $\alpha$ и $\beta$.

***Дефиниция.*** Нека $C \subseteq J_2^n$ е код. Означаваме с $d(C) = \min \rho(\alpha, \beta)$, $\alpha, \beta \in C, \alpha \neq \beta$ и наричаме $d(C)$ *минимално разстояние* на кода $C$.

Ще изразим коригиращите възможности на кода $C$ чрез минималното му разстояние. Нека по канала е изпратена думата $\alpha \in C$ и е получена $\beta$. Очевидно $\rho(\alpha, \beta)$ е точно броят на грешките, които каналът е допуснал при предаването на $\alpha$.

**Теорема 6.3.1** *Кодът $C \subseteq J_2^n$ открива $t$ грешки т.с.т.к. $d(C) \geq t + 1$ (при използване на канал с характеристика $\leq t/n$).*

***Доказателство.*** 1) Нека кодът открива $t$ грешки. Да допуснем, че $d(C) \leq t$. Тогава $\exists \alpha, \beta \in C, \alpha \neq \beta$, такива че $\rho(\alpha, \beta) \leq t$. Възможно е при предаване по канала на думата $\alpha$ той да измени точно тези $\leq t$ позиции, в които $\alpha$ се различава от $\beta$. Така при $\leq t$ допуснати грешки ще получим кодовата дума $\beta$ и по никакъв начин не можем да установим, че са направени грешки. Това е противоречие с условието, че кодът $C$ открива $t$ грешки.

2) Нека $d(C) \geq t+1$. Която и дума $\alpha \in C$ да изпратим и които и $\leq t$ позиции да промени каналът, получената дума $\beta \notin C$. Процедурата за декодиране е следната: Ако е получена кодова дума, това е изпратената дума. Ако получим дума, която не е от кода, това означава, че са направени грешки и отказваме декодирането. Следователно $C$ открива $t$ грешки. □

За следващата теорема използваме дискретен вариант на геометричното понятие кълбо.

***Дефиниция.*** Множеството $K(\alpha, t) = \{\beta | \beta \in J_2^n, \rho(\alpha, \beta) \leq t\}, \alpha \in J_2^n, t \in N$ наричаме *кълбо* с център $\alpha$ и радиус $t$.

**Лема 6.3.1** *Нека $\alpha, \beta \in J_2^n, t \in N$. Тогава $K(\alpha, t) \cap K(\beta, t) \neq \emptyset$ т.с.т.к $\rho(\alpha, \beta) \leq 2t$.*

***Доказателство.*** 1) Ако $K(\alpha, t) \cap K(\beta, t) \neq \emptyset$, то $\exists \gamma \in J_2^n$, $\gamma \in K(\alpha, t)$, $\gamma \in K(\beta, t)$, т.е. $\rho(\alpha, \gamma) \leq t$ и $\rho(\beta, \gamma) \leq t$. От симетричността на $\rho$ следва, че $\rho(\gamma, \beta) \leq t$. Сега, от неравенството на триъгълника $\rho(\alpha, \beta) \leq \rho(\alpha, \gamma) + \rho(\gamma, \beta) \leq t + t = 2t$.

2) Нека $\rho(\alpha, \beta) \leq 2t$ и $i_1, i_2, \ldots, i_r$ са позициите, в които $\alpha$ и $\beta$ се различават, $r \leq 2t$. Да инвертираме в $\alpha$ битовете в позиции $i_1, i_2, \ldots, i_{\lceil \frac{r}{2} \rceil}$ и да означим с $\gamma$ получения вектор. Тъй като $\lceil \frac{r}{2} \rceil \leq t$, то $\rho(\alpha, \gamma) \leq t$. Но $r - \lceil \frac{r}{2} \rceil \leq t$ и затова и $\rho(\beta, \gamma) \leq t$. Следователно $\gamma \in K(\alpha, t)$ и $\gamma \in K(\beta, t)$, т.е $\gamma \in K(\alpha, t) \cap K(\beta, t)$. □

**Теорема 6.3.2** *Кодът $C \subseteq J_2^n$ поправя $t$ грешки т.с.т.к. $d(C) \geq 2t+1$ (при използване на канал с характеристика $\leq t/n$).*

***Доказателство.*** 1) Нека кодът $C$ поправя $t$ грешки. Да допуснем, че $d(C) \leq 2t$. Тогава $\exists \alpha, \beta \in C, \alpha \neq \beta$ такива, че $\rho(\alpha, \beta) \leq 2t$. От Лема 6.3.1 следва, че $\exists \gamma \in K(\alpha, t) \cap K(\beta, t)$. Възможно е при предаване по канала на думата $\alpha$ той да измени точно тези $\leq t$ позиции, в които $\alpha$ се различава от $\gamma$. Получава се думата $\gamma \in K(\alpha, t)$. Тъй като $\gamma \in K(\beta, t)$, векторът $\gamma$ може да бъде получен и при изпращане на $\beta$, с направени



$\leq t$ грешки. Така не е възможно поправяне, защото не може да се определи дали $\gamma$ е получен от $\alpha$ или от $\beta$, което противоречи на условието. Следователно $d(C) \geq 2t+1$.

2) Нека $d(C) \geq 2t+1$. Нека сме изпратили по канала $\alpha$ и сме получили $\alpha'$ такъв, че $\rho(\alpha, \alpha') \leq t$. От $d(C) \geq 2t+1$ следва $\rho(\alpha', \beta) > t, \forall \beta \in C, \beta \neq \alpha$. Получаваме следната процедура за декодиране: Пресмятаме разстоянията от получената $\alpha'$ до всички кодови думи. Ако каналът не е направил повече от допустимите $t$ грешки, точно едно от разстоянията $\rho(\alpha', \alpha_i)$ ще бъде $\leq t$. Тогава $\alpha_i$ е изпратената кодова дума и кодът $C$ поправя $t$ грешки. □

Минималното разстояние на кода $C_1$ е 2, защото при произволна дума $\alpha$ най-близка до нея ще бъде такава дума $\beta$, че $\rho(\alpha, \beta) = 1$. Тогава $\rho(\alpha\alpha, \beta\beta) = 2$. Аналогично получаваме, че $d(C_2) = 3$. Минималното разстояние на $C_3$ е също 2, защото при произволни $\alpha$ и $\beta$, $\rho(\alpha, \beta) = 1$ битовете за проверка на четност $a$ на $\alpha$ и $b$ на $\beta$ ще са различни и $\rho(\alpha a, \beta b) = 2$. Ако пък $\rho(\alpha, \beta) = 2$ битовете за проверка на четност $a$ на $\alpha$ и $b$ на $\beta$ ще са еднакви и отново $\rho(\alpha a, \beta b) = 2$. Аналогично получаваме за $d(C_4) = 3$. Следователно, $C_1$ и $C_3$ откриват 1 грешка, а $C_2$ и $C_4$ поправят 1 грешка.

Ще разгледаме важен клас двоични кодове, поправящи грешки. За целта да разгледаме $J_2^n$ като линейно векторно пространство $L_2$ – понятие, което въведохме в Глава 1. С $wt(\alpha)$ означихме теглото на $\alpha \in J_2^n$ – броя на ненулевите му компоненти или $\rho(\tilde{0}, \alpha)$. Ще отбележим, че в $L_2$ е в сила $\rho(\alpha, \beta) = wt(\alpha + \beta)$.

***Дефиниция.*** Нека $C$ е линейно подпространство с размерност $k$ на линейното пространство $J_2^n$. Наричаме $C$ *линеен* $[n, k]$ *код* и го означаваме с $C[n, k]$.

Дължината на кода $C[n, k]$ е $n$. Нека $\{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ е базис на $C$. Тогава всеки вектор от $C$ е линейна комбинация $a_1\alpha_1 + a_2\alpha_2 + \ldots + a_k\alpha_k$ на базисните вектори, при това различните линейни комбинации дават различни вектори. Затова обемът $K(C[n, k]) = 2^k$, а скоростта $V(C[n, k]) = \frac{\log_2 2^k}{n} = \frac{k}{n}$.

***Дефиниция.*** Матрицата $G(C) = ||\alpha_1, \alpha_2, \ldots, \alpha_k||^T$, образувана от произволни $k$ линейно независими вектори на $C$, наричаме *пораждаща матрица* на $C$. С $A^T$ означаваме операцията *транспониране* на матрицата $A$, при която редовете на матрицата се обръщат в стълбове.

Да означим с $C^\perp$ ортогоналното на $C$ подпространство. То очевидно е линеен $[n, n-k]$ код. Наричаме го *ортогонален* код на $C$.

***Дефиниция.*** Нека $\{\beta_1, \beta_2, \ldots, \beta_{n-k}\}$ е базис на $C^\perp$. Матрицата $H(C) = G(C^\perp) = ||\beta_1, \beta_2, \ldots, \beta_{n-k}||^T$ наричаме *проверочна матрица* на кода

$C$.

Името е лесно обяснимо, защото от дефиницията за ортогонално пространство имаме $H(C)\alpha^T = \tilde{0}$ т.с.т.к. $\alpha \in C$, т.е. проверката дали $\alpha \in C$ извършваме с пресмятане на $H(C)\alpha^T$.

***Дефиниция.*** *Минимално тегло* на кода $C[n,k]$ наричаме теглото на най-лекия $\neq \tilde{0}$ вектор $\alpha$ на кода – $wt(C) = \min wt(\alpha), \alpha \in C, \alpha \neq \tilde{0}$.

Ако линейният $[n,k]$ код $C$ има минимално разстояние $d$, казваме, че той е $[n,k,d]$ код.

**Теорема 6.3.3** *За всеки линеен $[n,k,d]$ код $C$ е в сила $d(C) = wt(C)$.*

***Доказателство.*** 1) Нека минималното разстояние на кода се достига за някои $\alpha, \beta \in C$, $\alpha \neq \beta$. Следователно

$$d(C) = \rho(\alpha,\beta) = wt(\alpha+\beta) = wt(\gamma), \gamma \in C, \gamma \neq \tilde{0}$$

Но $wt(\gamma) \geq wt(C)$ следователно $d(C) \geq wt(C)$.

2) За някое $\alpha \in C, \alpha \neq \tilde{0}$ имаме

$$wt(C) = wt(\alpha) = wt(\tilde{0}+\alpha) = \rho(\tilde{0},\alpha).$$

Но $\alpha \in C, \tilde{0} \in C$ следователно $\rho(\tilde{0},\alpha) \geq d(C)$, т.е. $wt(C) \geq d(C)$.

От 1) и 2) следва, че $d(C) = wt(C)$. □

Заместващи полученото в Теорема 6.3.1 и Теорема 6.3.2, получаваме необходими и достатъчни условия линейният код $C$ да открива или поправя грешки, изразени с минималното тегло на кода. А ето и едно друго необходимо и достатъчно условие за поправяне на грешки от линеен код.

**Теорема 6.3.4** *Линейният код $C[n,k]$ поправя $t$ грешки т.с.т.к всеки $2t$ вектор-стълба на проверочната матрица $H(C)$ са линейно независими.*

***Доказателство.*** Нека $\alpha \in C, wt(\alpha) = w$, а

$$H(C) = \begin{Vmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ & & \dots & \\ b_{(n-k)1} & b_{(n-k)2} & \dots & b_{(n-k)n} \end{Vmatrix}.$$

Тогава

$$H(C)\alpha^T = \begin{Vmatrix} b_{1i_1} \\ . \\ . \\ . \\ b_{(n-k)i_1} \end{Vmatrix} + \begin{Vmatrix} b_{1i_2} \\ . \\ . \\ . \\ b_{(n-k)i_2} \end{Vmatrix} + \dots + \begin{Vmatrix} b_{1i_w} \\ . \\ . \\ . \\ b_{(n-k)i_w} \end{Vmatrix} = \tilde{0},$$



където $i_1, i_2, \dots, i_w$ са позициите, в които $\alpha$ има ненулеви компоненти (в случая на двоичен код – единици). Това означава, че стълбовете на $H(C)$ с номера $i_1, i_2, \dots, i_w$ са линейно зависими и следователно в матрицата $H(C)$ има $w \neq 0$ линейно зависими стълба т.с.т.к. в $C$ има ненулев вектор с тегло $w$. Но $C$ поправя $t$ грешки т.с.т.к. $d(C) = wt(C) \geq 2t+1$, т.е. т.с.т.к всеки $2t$ вектор-стълба на $H(C)$ са линейно независими. □

Тъй като $(J_2^n; \oplus)$ е група, то всеки линеен код $C[n,k] \subseteq J_2^n$ е подгрупа на $(J_2^n; \oplus)$ и можем да факторизираме $(J_2^n, \oplus)$ по $C[n,k]$.

Ще направим факторизацията с едно допълнително изискване: всеки път, когато избираме елемент за образуване на нов съседен клас на $C$, това да бъде вектор с най-малко тегло, който още не е включен в съседен клас. Полученото разбиване на $J_2^n$ на съседни класове наричаме *стандартно разлагане на Слепян* (вж. Табл. 6.3). Както знаем от 1.4, всеки елемент на групата $(J_2^n, \oplus)$ ще се появи точно веднъж в таблицата на разлагането и броят на съседните класове (включително и кода $C$) е точно $2^n/2^k = 2^{n-k}$.

| | $H\beta_1^T$ | $H\beta_2^T$ | $\dots$ | $H\beta_{2^{n-k}-1}^T$ |
|---|---|---|---|---|
| $\tilde{0}$ | $\beta_1$ | $\beta_2$ | $\dots$ | $\beta_{2^{n-k}-1}$ |
| $\alpha_1$ | $\alpha_1+\beta_1$ | $\alpha_1+\beta_2$ | $\dots$ | $\alpha_1+\beta_{2^{n-k}-1}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $\alpha_{2^k-1}$ | $\alpha_{2^k-1}+\beta_1$ | $\alpha_{2^k-1}+\beta_2$ | $\dots$ | $\alpha_{2^k-1}+\beta_{2^{n-k}-1}$ |

Таблица 6.3: Стандартно разлагане на Слепян.

***Дефиниция.*** Векторите $\beta_1, \beta_2, \dots, \beta_{2^{n-k}-1}$, които са с минимални тегла в съседните класове на разлагането на Слепян, наричаме *лидери* на съседните класове.

**Теорема 6.3.5** *Кодът $C[n,k]$ поправя $t$ грешки т.с.т.к. всеки вектор $\beta \in J_2^n, wt(\beta) \leq t$, е лидер на съседен клас в разлагането на Слепян.*

***Доказателство.*** 1) Нека всеки $\beta \in J_2^n, wt(\beta) \leq t$, е лидер на съседен клас. Очевидно е, че $K(\alpha,t) = \{\gamma | \gamma = \alpha+\beta, wt(\beta) \leq t\}$. Но тогава $K(\alpha,t)$ се намира изцяло в ред на разлагането на Слепян и когато $\alpha_1, \alpha_2 \in C, \alpha_1 \neq \alpha_2$, тогава $K(\alpha_1,t) \cap K(\alpha_2,t) = \emptyset$. Следователно $C$ поправя $t$ грешки.

2) Нека $C$ поправя $t$ грешки, т.е. $wt(C) \geq 2t+1$. Да допуснем, че съществува $\beta \in J_2^n, 0 < wt(\beta) \leq t$, който не е лидер на съседен клас ($\beta \notin C$, защото $wt(\beta) < wt(C)$). Тогава $\exists \alpha_i \in C$ и лидер на съседен клас $\beta_j$ такива, че $\beta = \alpha_i + \beta_j$ и $wt(\beta_j) \leq wt(\beta)$. Но тогава $\alpha_i = \beta + \beta_j$ и $wt(\alpha_i) = wt(\beta + \beta_j) \leq wt(\beta) + wt(\beta_j) \leq 2wt(\beta) \leq 2t$ , а това противоречи на условието, че $wt(C) \geq 2t+1$. Следователно всеки ненулев вектор с тегло $\leq t$ е лидер на съседен клас. □

***Дефиниция.*** Нека $C[n,k]$ е линеен код. Вектора $\sigma = H(C)\gamma^T \in J_2^{n-k}$ наричаме *синдром* на $\gamma$ по отношение на $C$, $\forall \gamma \in J_2^n$.

**Теорема 6.3.6** *Нека $C[n,k]$ е линеен код. За векторите $\gamma_1, \gamma_2 \in J_2^n$, $H(C)\gamma_1^T = H(C)\gamma_2^T$ т.с.т.к. $\gamma_1$ и $\gamma_2$ са в един и същ съседен клас на $C$.*

***Доказателство.*** 1) Нека $\gamma_1$ и $\gamma_2$ са в един и същ съседен клас на $C$ с лидер $\beta_i$. Тогава $\gamma_1 = \alpha_j + \beta_i, \gamma_2 = \alpha_k + \beta_i$ и $H(C)\gamma_1^T = H(C)\alpha_j^T + H(C)\beta_i^T = H(C)\beta_i^T$, и $H(C)\gamma_2^T = H(C)\alpha_k^T + H(C)\beta_i^T = H(C)\beta_i^T$. Следователно $H(C)\gamma_1^T = H(C)\gamma_2^T$.

2) Нека $H(C)\gamma_1^T = H(C)\gamma_2^T$, т.е. $H(C)(\gamma_1^T + \gamma_2^T) = \tilde{0}$.Тогава $\gamma_1 + \gamma_2 = \alpha \in C$. Да допуснем, че $\gamma_1$ и $\gamma_2$ са в различни съседни класове, т.е. $\exists \beta_i \neq \beta_j, \gamma_1 = \alpha_m + \beta_i$, $\gamma_2 = \alpha_k + \beta_j$. Сега $\alpha = \gamma_1 + \gamma_2 = \alpha_m + \beta_i + \alpha_k + \beta_j$. От тук $\beta_i = \alpha + \alpha_m + \alpha_k + \beta_j$. Но $\alpha + \alpha_m + \alpha_k \in C$, следователно $\beta_i$ е елемент от съседния клас на $\beta_j$. Това противоречи на факта, че $\beta_i$ също е лидер на съседен клас. Следователно $\gamma_1$ и $\gamma_2$ са в един и същ съседен клас. □

Теоремите 6.3.5 и 6.3.6 дават следната процедура за декодиране на кода $C[n,k]$, поправящ $t$ грешки: Нека сме изпратили кодовата дума $\alpha_i \in C$ и при преминаването през канала към нея се е прибавил допустим, но неизвестен шум $\varepsilon \in J_2^n, wt(\varepsilon) \leq t$. Получен е векторът $\alpha = \alpha_i + \varepsilon$. Пресмятаме $H(C)\alpha^T = H(C)(\alpha_i + \varepsilon)^T = H(C)\varepsilon^T = \sigma$. Достатъчно е в разлагането на Слепян да имаме и синдромите $\sigma_j = H(C)\beta_j^T$ за всеки лидер на съседен клас $\beta_j$. Сега намираме такова $\beta_j$, че $\sigma = \sigma_j$ (съгласно Теорема 6.3.6 различните лидери имат различни синдроми). Очевидно $\beta_j = \varepsilon$ и изпратената дума е $\alpha + \beta_j$.

Като примери ще разгледаме два безкрайни класа линейни кодове.

Пример 5. Кодове на Хеминг

Нека $l \geq 2$. Да разгледаме матрицата $H(\mathcal{H}_l)$ с $l$ реда и $2^l - 1$ стълба, състояща се от всички различни от $\tilde{0}$ вектор-стълба от $J_2^l$. Тя съдържа всички $l$ вектор-стълба с тегло 1, следователно редовете ѝ са линейно независими. Дефинираме $l$-тия *код на Хеминг* $\mathcal{H}_l$ като линеен код с проверочна матрица $H(\mathcal{H}_l)$. Очевидно $\mathcal{H}_l$ е $[2^l - 1, 2^l - 1 - l]$-код и значи



скоростта му е $V = \frac{2^l - l - 1}{2^l - 1} \approx 1 - \frac{log_2 n}{n}$ – малко по-лоша от скоростта на кодовете с проверка на четност. Коригиращите възможности на Хеминговите кодове получаваме от Теорема 6.3.4. Лесно се вижда, че всеки два стълба в проверочната матрица на Хемингов код са линейно независими. От друга страна, ако вземем два различни стълба $\alpha$ и $\beta$ на проверочната матрица, то $\alpha, \beta$ и $\alpha + \beta$ са линейно зависими ($\alpha + \beta + (\alpha + \beta) = \tilde{0}$) и следователно $\mathcal{H}_l$ поправя една грешка.

Кодовете на Хеминг имат ефектно и просто декодиране. За пример да разгледаме $\mathcal{H}_3$ с проверочна матрица

$$H(\mathcal{H}_3) = \begin{Vmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{Vmatrix},$$

в която вектор-стълбовете с височина три са подредени в нарастващ ред на съответните цели числа, на които са двоични представяния. Нека е изпратен векторът $\alpha_i$ и към него в канала се е прибавил векторът-грешка $\varepsilon, wt(\varepsilon) = 1$. Получен е векторът $\alpha = \alpha_i + \varepsilon = (1,1,0,1,1,0,1)$. Пресмятаме $H(\mathcal{H}_3)\alpha^T = H(\mathcal{H}_3)\varepsilon = (1,0,1)$. Тъй като това е двоичното представяне на числото 5, то очевидно $\varepsilon$ е имал единица само в петата компонента. За декодирането на $\alpha$ е достатъчно да инвертираме петата му компонента. Така установяваме, че е изпратен векторът $(1,1,0,1,0,0,1)$. Тази процедура е много по-проста от декодирането на кода $C_4$, който има същите коригиращи възможности. Разбира се, за да може да бъде използван $\mathcal{H}_l$ в канал с характеристика $\leq 1/h$ е необходимо $2^l - 1 \leq h$.

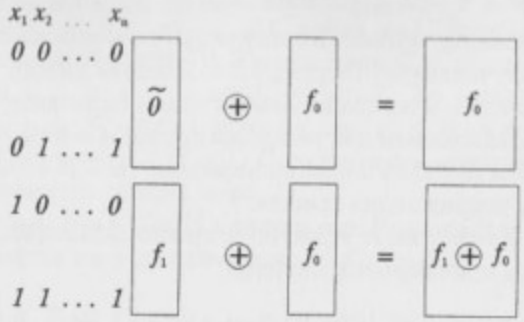
Пример 6. Кодове на Рид-Малер

Нека $m$ е цяло положително число и $l$ е цяло $1 \leq l \leq m$. Да означим с $\mathcal{RM}(m,l)$ множеството от всички булеви функции на $m$ променливи, полиномите на Жегалкин на които имат степен $\leq l$. Множеството $\mathcal{F}_2^m$ на булевите функции на $m$ променливи съвпада с $J_2^{2^m}$ и следователно е линейно пространство с размерност $2^m$. Тогава $\mathcal{RM}(m,l)$ е линейно подпространство на $\mathcal{F}_2^m$, защото $\forall f, g \in \mathcal{RM}(m,l)$ полиномът на Жегалкин на $f \oplus g$ е също от степен $\leq l$.

Кодовете $\mathcal{RM}(m,l), l = 1,2,\ldots,m$, наричаме *кодове на Рид-Малер* от $l$-ти ред. Дължината на $\mathcal{RM}(m,l)$ е $n = 2^m$, а размерността $k = \sum_{i=0}^{l} \binom{m}{i}$, като естествен базис на $\mathcal{RM}(m,l)$ са елементарните конюнкции без отрицания с $1,2,\ldots,l$ букви и $\tilde{1}$.

**Теорема 6.3.7** $wt(\mathcal{RM}(m,l)) = 2^{m-l}$.

***Доказателство.*** 1) Функцията $f = x_1 x_2 \ldots x_l \in \mathcal{F}_2^m$ е от кода на




Фигура 6.4: Кодове на Рид-Малер.

Рид-Малер $\mathcal{RM}(m,l)$, но $|N_f| = 2^{m-l}$, следователно $wt(\mathcal{RM}(m,l)) \leq 2^{m-l}$.

2) С индукция по $m$ ще докажем, че $wt(\mathcal{RM}(m,l)) \geq 2^{m-l}, 1 \leq l \leq m$.

Нека $m = 1$. Тогава имаме само един код $\mathcal{RM}(1,1)$. Базис на този код е $\{\tilde{1}, x_1\}$ или $\mathcal{RM}(1,1) = \{\tilde{0}, \tilde{1}, x_1, x_1 \oplus 1\}$ и очевидно $wt(\mathcal{RM}(1,1)) = 1 = 2^{1-1} = 2^{m-l}, m = 1, l = 1$.

Нека твърдението е в сила за $\mathcal{RM}(1,l), \ldots, \mathcal{RM}(m-1,l)$. С индукция по $l$ ще докажем, че то е в сила и за $\mathcal{RM}(m,l)$.

Нека $l = m$. Тогава $\mathcal{RM}(m,m) = J_2^{2^m}$ и $wt(\mathcal{RM}(m,m) = 1 = 2^{m-m}$.

Да допуснем, че твърдението е в сила за $\mathcal{RM}(m,m), \ldots, \mathcal{RM}(m,i+1)$. Ще докажем, че $wt(\mathcal{RM}(m,i)) \geq 2^{m-i}$. Нека $f(x_1, x_2, \ldots, x_m) \in \mathcal{RM}(m,i)$ е представена с полинома си на Жегалкин. Разлагаме $f$ на две подфункции $f_1$ и $f_0$ такива, че

$$f(x_1, x_2, \ldots, x_m) = x_1 f_1(x_2, x_3, \ldots, x_m) \oplus f_0(x_2, x_3, \ldots, x_m),$$

като $f_1 \in \mathcal{RM}(m-1, i-1), f_0 \in \mathcal{RM}(m-1,i)$. На Фиг. 6.4 е показано как стълбът (с височина $2^m$) на $f$ се образува от стълбовете (с височини $2^{m-1}$) на $f_0$ и $f_1$, съгласно полученото по-горе разлагане.

Сега $wt(f) = wt(f_0) + wt(f_0 \oplus f_1)$, но $wt(f_0) \geq 2^{(m-1)-i}$, защото $f_0 \in \mathcal{RM}(m-1,i)$, а $wt(f_0 \oplus f_1) \geq 2^{(m-1)-i}$, защото $f_0 \oplus f_1 \in \mathcal{RM}(m-1,i)$. Следователно

$$wt(f) \geq 2^{(m-1)-i} + 2^{(m-1)-i} = 2.2^{(m-1)-i} = 2^{m-i}.$$

Тъй като това е в сила $\forall f \in \mathcal{RM}(m,i)$, то $wt(\mathcal{RM}(m,i)) \geq 2^{m-i}$. □



Ортогонален на $\mathcal{RM}(m,l)$ е точно $\mathcal{RM}(m, m-l-1) = \mathcal{RM}(m, m-(l+1))$. Действително, да представим $\mathcal{RM}(m,l)$ с базисни елементи – елементарни конюнкции от вида $x_{i_1} x_{i_2} \ldots x_{i_r}, r \leq l$ или $\tilde{1}$. Аналогично $\mathcal{RM}(m, m-(l+1))$ можем да представим с базисни елементи – елементарни конюнкции от вида $x_{i_1} x_{i_2} \ldots x_{i_s}, s \leq m-(l+1)$ или $\tilde{1}$. Покомпонентното произведение на двоични вектори е тъждествено на конюнкцията на булеви функции. Затова произведението на два базисни елемента (по един от всеки от кодовете) ще бъде елементарна конюнкция от степен $p \leq l+m-(l+1) = m-1$ или $\tilde{1}$ ($p = 0$) и ще има в стълба си $2^{m-p}, 1 \leq m-p \leq m$, т.е. четен брой единици. Скаларното произведение на всеки два такива базисни елемента е 0 и следователно всеки вектор на единия код е ортогонален на всеки вектор от другия код. За да довършим доказателството на твърдението, ще отбележим, че размерността на $\mathcal{RM}(m,l)$ е $k(\mathcal{RM}(m,l)) = \sum_{i=0}^{l} \binom{m}{i}$, а размерността на $\mathcal{RM}(m, m-(l+1))$ е $k(\mathcal{RM}(m, m-(l+1))) = \sum_{i=0}^{m-(l+1)} \binom{m}{i} = \sum_{i=m}^{l+1} \binom{m}{i} = \sum_{i=l+1}^{m} \binom{m}{i}$. Следователно $k(\mathcal{RM}(m,l)) + k(\mathcal{RM}(m, m-(l+1))) = \sum_{i=0}^{m} \binom{m}{i} = 2^m = n$ и двата кода са ортогонални.

Интересна е процедурата за декодиране на кодовете на Рид-Малер. За пример да разгледаме кода $\mathcal{RM}(3,1)$. $d(\mathcal{RM}(3,1)) = wt(\mathcal{RM}(3,1)) = 2^2 = 4$ и от този код можем да очакваме (вж. Теорема 6.3.2) малко повече от поправяне на една грешка. Всъщност $\mathcal{RM}(3,1)$ открива някои по-сложни грешки, без да може да ги поправи. Базови елементи на $\mathcal{RM}(3,1)$ са $\tilde{1}, x_1, x_2$ и $x_3$ и

$$G(\mathcal{RM}(3,1)) = \left\| \begin{array}{cccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{array} \right\| \begin{array}{c} \tilde{1} \\ x_1 \\ x_2 \\ x_3 \end{array}$$

е пораждаща матрица на $\mathcal{RM}(3,1)$. Нека $\alpha(a_1, a_2, \ldots, a_8) \in \mathcal{RM}(3,1)$, т.е. $\alpha = b_0 \oplus b_1 x_1 \oplus b_2 x_2 \oplus b_3 x_3$. Използвайки пораждащата матрица, получаваме за неизвестните коефициенти системата

$$\left| \begin{array}{l} b_0 = a_1 \\ b_0 \oplus b_3 = a_2 \\ b_0 \oplus b_2 = a_3 \\ b_0 \oplus b_2 \oplus b_3 = a_4 \end{array} \right| \quad \left| \begin{array}{l} b_0 \oplus b_1 = a_5 \\ b_0 \oplus b_1 \oplus b_3 = a_6 \\ b_0 \oplus b_1 \oplus b_2 = a_7 \\ b_0 \oplus b_1 \oplus b_2 \oplus b_3 = a_8 \end{array} \right|$$

Събирайки (по модул 2) подходящо избрани двойки уравнения (лесно се вижда кои, по индексите на компонентите от дясната част), получаваме

$$b_1 = a_1 \oplus a_5, \quad b_1 = a_2 \oplus a_6, \quad b_1 = a_3 \oplus a_7, \quad b_1 = a_4 \oplus a_8,$$

$$b_2 = a_1 \oplus a_3, \quad b_2 = a_2 \oplus a_4, \quad b_2 = a_5 \oplus a_7, \quad b_2 = a_6 \oplus a_8,$$
$$b_3 = a_1 \oplus a_2, \quad b_3 = a_3 \oplus a_4, \quad b_3 = a_5 \oplus a_6, \quad b_3 = a_7 \oplus a_8.$$

Нека при изпращането на $\alpha$ по канал с характеристика $\leq 1/8$ не е допусната нито една грешка. Тогава изчисляването на стойностите за всяко от $b_1, b_2$ и $b_3$ по горните формули ще даде по четири еднакви стойности. Нека при предаването е сгрешена една компонента, например $a_1$. Тогава първите стойности за $b_1, b_2$ и $b_3$ ще се различават от останалите 3 и по това, че в пресмятането и на трите участва $a_1$, можем да заключим, че или $a_1$ е сгрешена или са сгрешени едновременно $a_2, a_3$ и $a_5$. Знаейки поведението на канала, втората възможност изглежда невероятна, затова заключаваме, че е сгрешена $a_1$ и можем да я поправим. Тази техника на декодиране наричаме *мажоритарно декодиране*. Оставяме на читателя да съобрази какви по-сложни комбинации от грешки кодът открива без да може да ги поправи.

Ето и някои твърдения за параметрите на линейни двоични кодове.

**Теорема 6.3.8 (Граница на Хеминг)** *За всеки линеен двоичен код с дължина $n$ и размерност $k$, който поправя $t$ грешки е в сила неравенството* $2^{n-k} \geq \sum_{i=0}^{t} \binom{n}{i}$.

***Доказателство.*** Кълбата с центрове кодовите думи ($2^k$ на брой) и радиус $t$ не се пресичат, защото кодът поправя $t$ грешки. Всяко кълбо се състои от $\sum_{i=0}^{t} \binom{n}{i}$ вектора и в обединението на всички кълба има $2^k \sum_{i=0}^{t} \binom{n}{i}$ различни вектора. Този брой не може да надхвърля $2^n$ – обема на цялото пространство, откъдето следва $2^{n-k} \geq \sum_{i=0}^{t} \binom{n}{i}$. □

***Дефиниция.*** Кодове, за които имаме равенство в границата на Хеминг, наричаме *съвършени*.

**Теорема 6.3.9 (Граница на Варшамов-Джилбърт)** *Ако е изпълнено неравенството*

$$1 + \binom{n-1}{1} + \cdots + \binom{n-1}{d-2} \leq 2^{n-k},$$

*то съществува линеен $[n, k]$ код с минимално разстояние поне $d$.*

***Доказателство.*** При условията на Теоремата ще построим матрица $H$ с $n-k$ реда и $n$ стълба, всеки $d-1$ стълба на която са линейно независими и тогава, съгласно Теорема 6.3.4, $H$ ще бъде проверочна матрица на линеен код с минимално разстояние поне $d$.



За първи стълб на $H$ можем да изберем кой да е вектор от $J_2^{n-k}$. Затова да допуснем, че за някое $i$, $1 \leq i \leq n-1$ сме избрали $i$ стълба, никои $d-1$ от които не са линейно зависими. От тези $i$ можем да образуваме най-много $\binom{i}{1} + \cdots + \binom{i}{d-2}$ линейни комбинации от не повече от $d-2$ стълба. Съгласно условието на теоремата, при $i \leq n-1$ този брой е по-малък от $2^{n-k} - 1$ и можем да изберем ненулев стълб, който е различен от всяка от горните линейни комбинации. Добавяме този стълб към вече избраните $i$ и получаваме матрица с $i+1$ стълба, никои $d-1$ от които не са линейно зависими. Продължавайки разсъжденията индуктивно построяваме търсената $H$ с $n$ стълба. □

## 6.4 Криптология

По причини от най-различен характер – политически, военни, икономически, личностни и т.н., при това още от дълбока древност, се е налагало да се кодира информация по такъв начин, че вложеният в нея смисъл да остава скрит за лица или институции, които не са упълномощени да я ползват. В същото време някои неупълномощени лица или институции (обикновено тези, от които е скрита информацията), по същите или други причини, са полагали сериозни усилия да декодират интересуващата ги информация. И в единия, и в другия случай могат да се използват методи и алгоритми от различни дискретни математически теории. Днес тази тематика е обединена в математическо направление, наречено *криптология*. Разделът на криптологията, занимаващ се с методите за надеждно кодиране (криптиране) и нормалното декодиране (декриптиране) на информацията, носи името *криптография* , а този, занимаващ се със способите за несанкционирано декодиране („разбиване" на кодовата система) – *криптоанализ*.

Тук няма да се спираме на нематематически методи за криптиране и декриптиране на информация, като използване на кодови книги от изпращача и получателя, при което отделни думи и цели фрази на изходното съобщение се заменят с думи или фрази от кодовата книга. Такива начини на криптиране не се поддават лесно на анализ, но са неудобни, заради наличието на елемент в системата – кодовата книга, която се съхранява трудно и не осигурява бързи и точни процедури за криптиране и декриптиране. Затова, в стила на разглежданията в тази глава, ще се спрем на побуквените криптосистеми, които в редица случаи не са толкова надеждни, но процедурите за криптиране и декриптиране са алгоритмични.

***Дефиниция.*** Нека крайните множества $A$ и $B$ са съответно изходна

и кодова азбука, а $\mathcal{K}$ – крайно множество, елементите на което наричаме *ключове*. Нека $E : A^+ \times \mathcal{K} \to B^+$, $D : B^+ \times \mathcal{K} \to A^+$ и $\kappa : \mathcal{K} \to \mathcal{K}$ са такива, че $d(E(\alpha, K)) = d(\alpha)$, $d(D(\beta, K')) = d(\beta)$ и ако $K' = \kappa(K)$, то $D(E(\alpha, K), K') = \alpha$. Тогава $< \mathcal{K}, E, D >$ наричаме *криптосистема* или *шифър*. Думите от $A^+$ наричаме *открити съобщения*, а стойностите на функцията $E$ – *шифровани съобщения* или *криптограми*. Пресмятането на $E$ наричаме *криптиране* (или *шифриране*), а пресмятането на $D$ – *декриптиране* (или *дешифриране*).

В горната дефиниция не е указано явно, но се предполагат следните свойства на криптосистемата:

1. Функциите $E$ и $D$ се изчисляват просто.

2. Ако не е известна функцията $D$, декриптирането на шифровано съобщение е трудно.

3. Даже когато е известна функцията $D$, но не е известен декриптиращият ключ $K'$, съответен на ключа $K$, с който е шифрирано съобщението, декриптирането на шифровано съобщение остава трудно.

Днес в математическата криптология се разглеждат главно два типа криптосистеми.

***Дефиниция.*** Криптосистема, в която криптиращият и декриптиращият ключ се получават лесно един от друг (например – съвпадат), наричаме *симетрични криптосистеми* или *криптосистеми с общ ключ*. Криптосистема, в която криптиращият и декриптиращият ключ са не просто различни, а се получават трудно един от друг, наричаме *асиметрични криптосистеми* или *криптосистеми с публичен ключ*.

### 6.4.1 Криптосистеми с общ ключ

При създаване на криптосистеми с общ ключ се използват главно две техники – пермутации и субституции, както и комбинации на тези две техники. Пермутационните криптосистеми запазват буквите в шифрирания текст, но променят местата им, докато субституционните шифри запазват местата на буквите, но ги заменят с други букви. Една и съща буква може да бъде заменена от субституционен шифър с различни букви в зависимост от мястото ѝ в текста. Субституционните шифри са по-разнообразни и с повече възможности да осигурят надеждно шифриране.

Да разгледаме първо някои пермутационни шифри. Най-прост пермутационен шифър получаваме, ако разменим (пермутираме) първата буква на открития текст с последната, втората с предпоследната и т.н. Ако шифрираме текста „дискретна математика“ ще получим криптограмата „акитаметам антерксид“. Тази криптосистема практически не използва ключ. За да декриптираме полученото съобщение, достатъчно е да го четем отдясно наляво. Разбиването на такава криптосистема е елементарно.



Малко по-сложна пермутационна криптосистема получаваме по следния начин. За множество от ключове избираме естествените числа, различни от 0. Нека $n$ е избраният ключ, а $m$ – дължината на открития текст. В правоъгълна таблица с $n$ реда и $\lceil m/n \rceil$ стълба записваме буквите на открития текст по стълбове, започвайки от най-левия стълб и вписвайки буквите в стълба отгоре надолу. При $n = 1$ криптограмата ще съвпада с открития текст, затова е добре ключът да е различен от 1. За съобщението „дискретна математика“, при ключ $n = 3$ получаваме Табл. 6.4.

| | | | | | | |
|---|---|---|---|---|---|---|
| д | к | т | | т | а | к |
| и | р | н | м | е | т | а |
| с | е | а | а | м | и | |

Таблица 6.4: Таблица на пермутационен шифър

Шифрирането извършваме, като четем буквите отляво надясно по редове, започвайки от първия. За нашия пример получаваме криптограмата „дкт такирнметасеаами“.

Декриптирането се извършва, като се построи таблицата на криптирането отново. Криптограмата се вписва в таблицата по редове, а се чете по стълбове – обратно на процеса на криптиране. И при този шифър разбиването не е много трудно. Достатъчно е да се направят последователно опити с ключовете $1, 2, \ldots, n$.

В горните два примера се използват шифри, множеството $K$ от ключовете на които се състои от пермутации, $K = \{\pi_i | i \in N, \pi_i \in \mathcal{P}_i\}$. За шифриране на открит текст с дължина $m$ се използва пермутацията $\pi_m$. Не е възможно да се помнят твърде много елементи на $K$ и на практика $K$ се състои от пермутации, които лесно се описват с думи и не е много трудно за криптоаналитика да се досети за вида на такава пермутация. Затова такива криптосистеми се разбиват лесно.

Нека сега да вземем за множество от ключове $\mathcal{P}_n$ – пермутациите на $n$ променливи, като $n > 2$ е константа, параметър на множеството от ключове (т.е. част от ключа). Нека $\alpha \in A^*$ е откритият текст, като за удобство ще считаме дължината му за кратна на $n$, т.е.

$$\alpha = a_{1,1} \cdots a_{1,n} a_{2,1} \cdots a_{2,n} \ldots a_{k,1} \cdots a_{k,n}.$$

Криптиращата функция $E : A^* \times \mathcal{P}_n \to A^*$ дефинираме така, че

$$E(\alpha, \pi) = a_{1,\pi(1)} \cdots a_{1,\pi(n)} a_{2,\pi(1)} \cdots a_{2,\pi(n)} \cdots a_{k,\pi(1)} \cdots a_{k,\pi(n)}, \pi \in \mathcal{P}_n.$$

Да шифрираме съобщението „дискретна математика", като $n = 5$, а за ключ сме избрали пермутацията $\pi = (3\,5\,1\,4\,2)$. Получаваме криптограмата „срдкин еаттммеаиаакт".

Нека при получателя е пристигнал криптираният текст

$$\beta = b_{1,1} \cdots b_{1,n} b_{2,1} \cdots b_{2,n} \ldots b_{k,1} \cdots b_{k,n}.$$

От ключа $\pi$ пресмятаме лесно обратната пермутация $\pi^{-1}$. Декриптиращата функция $E : A^* \times \mathcal{P}_n \to A^*$ дефинираме така

$$D(\beta, \pi) = b_{1,\pi^{-1}(1)} \cdots b_{1,\pi^{-1}(n)} b_{2,\pi^{-1}(1)} \cdots b_{2,\pi^{-1}(n)} \ldots b_{k,\pi^{-1}(1)} \cdots b_{k,\pi^{-1}(n)}.$$

Пресмятането на криптиращата и декриптиращата функция при известен ключ не представлява никаква трудност. Криптоаналитикът, дори да се досеща, че е използван пермутационен шифър, ще трябва първо да познае числото $n$, а след това да провери, в най-лошия случай, $n!$ пермутации, за да разбие системата. Това вече е значителна трудност даже при не много големи $n$. Дължината $m$ на криптограмата може да бъде използвана при криптоанализа, тъй като най-често $n$ дели $m$. Тази неприятност лесно се избягва като се допълни откритият текст с произволни букви до текст с дължина просто число. При декриптиране излишните букви лесно се определят от контекста на съобщението.

Да разгледаме някои примери на субституционни шифри. В изложението използваме текстове написани на български език, т.е. с букви на кирилицата. Тъй като някои от криптиращите функции са числови, ще използваме изображението на кирилицата в $J_{30}$, запазващо естествената наредба на буквите: а – 0, б – 1, в – 2,..., я – 29. Всъщност представянето на буквите с числа също е субституция и може да бъде различно от фиксираното по-горе. Така, то може да се разглежда като част от криптосистемата. По традиция в субституционните шифри не се шифрират препинателните знаци и интервалите.

***Дефиниция.*** Взаимно-еднозначната функция $e_k : A \to B$ зависеща от ключа $k \in K$ наричаме *субституционен шифър*.

Шифрирането на съобщението $\alpha = a_{i_1} a_{i_2} \ldots a_{i_l} \in A^+$ извършваме, като заместим всяка буква $a_{i_j}$ с $b_{i_j} = e_k(a_{i_j})$. Дешифрирането на криптограмата $\beta = b_{i_1} b_{i_2} \ldots b_{i_l} \in A^+$ извършваме, като заместим всяка буква $b_{i_j}$ с $a_{i_j} = e_k^{-1}(b_{i_j})$.



Нека $A = B = J_n$. Да разгледаме функции от вида $e_{k,k'}(a) = ka + k' (\mathrm{mod}\, n)$, в която $a \in J_n$, а ключът е наредена двойка цели числа $(k, k')$, $k, k' \in J_n$.

**Лема 6.4.1** *Функцията $f(a) = ka (\mathrm{mod}\, n)$, в която $a, k \in J_n$ е взаимно-еднозначна, ако $k$ и $n$ са взаимно прости.*

***Доказателство.*** Нека $k$ и $n$ са взаимно прости. Да допуснем нееднозначността на $f$, т.е. $\exists a_1 > a_2 \in J_n, f(a_1) = f(a_2)$, т.е. $ka_1 (\mathrm{mod}\ n) = ka_2 (\mathrm{mod} n)$ или $ka_1 = p_1 n + q$, а $ka_2 = p_2 n + q$. Тогава $k(a_1 - a_2) = n(p_1 - p_2)$ и $n | k(a_1 - a_2)$. Но $a_1 - a_2 < n$ и следователно $n$ дели $k$, което е в противоречие с взаимната простота на $k$ и $n$. Следователно $f$ е еднозначна. □

**Следствие 6.4.1** *Ако $k$ и $n$ са взаимно прости, тогава $e(a) = ka + k' (\mathrm{mod}\, n)$ е взаимно-еднозначна.*

Оставяме на читателя да покаже, че прибавянето на константата $k'$ към $n$-те различни стойности на еднозначната функция $ka (\mathrm{mod}\, n)$ дава $n$ различни стойности.

***Дефиниция.*** Биекцията $e_{k,k'}(a) = ka + k' (\mathrm{mod}\, n)$, където $k$ и $n$ са взаимно прости, наричаме *модулярен шифър* с ключ $(k, k')$. Модулярен шифър с $k = 1$ наричаме *циклическо изместване* на азбуката $J_n$ на $k'$ букви.

Циклическото изместване на 3 букви на латинската азбука е субституционен шифър, който се свързва с името на римския император Гай Юлий Цезар. Да разгледаме като пример вариант на шифъра на Цезар с $k' = 3$ и $n = 30$ за азбуката на използваните в българския език букви на кирилицата. Да криптираме с този шифър прочутата фраза на императора „дойдохвидяхпобедих" (Напомняме, че в субституционните шифри пропускаме препинателните знаци и интервалите.) След замяна на буквите с числа от $J_{30}$ получаваме съобщението

$$4, 14, 9, 4, 14, 21, 2, 8, 4, 29, 21, 15, 14, 1, 5, 4, 8, 21$$

което криптираме в

$$7, 17, 12, 7, 17, 24, 5, 11, 7, 2, 24, 18, 17, 4, 8, 7, 21, 24$$

или върнато обратно в кирилица – „зсмзсшелзвштсдизтш".

За дешифриране на криптограма, шифрирана с модулярен шифър, пресмятаме за всяка получена буква $b$ стойността $b' = b + (n - k') (\mathrm{mod}\, n)$

и решаваме уравнението $b' = ka (\mathrm{mod}\, n)$ спрямо $a$. Съгласно Лема 6.4.1 то има единствено решение по модул $n$. При $k \neq 1$ можем да имаме таблица с $n$ елемента, в която елементът индексиран с $b'$ е решението $a$ на $b' = ka (\mathrm{mod}\, n)$. При шифрите, изместващи азбуката на $k'$ цифри, очевидно, $a = b'$.

***Дефиниция.*** Нека $n$ е естествено положително число. Функцията $\varphi : N \to N$ такава, че $\varphi(n)$ е броят на естествените положителни числа $m \leq n$, което са взаимно прости с $n$, наричаме *функция на Ойлер*.

Например, $\varphi(6) = 2$, защото от естествените положителни числа помалки от 6 взаемно прости с 6 са 1 и 5.

Модулярните шифри се поддават лесно на криптоанализ. Броят на различните ключове е $n\varphi(n)$ и не е трудно да бъдат опитани всички възможни ключове. На практика нещата се опростяват още повече от факта, че може да бъде направен честотен анализ на източника на съобщенията. (Честотен анализ на текста на тази книга дадохме в раздел 7.2.) Забележете, че модулярните шифри запазват честотните характеристики на източника и затова при криптоанализ на достатъчно много шифрограми не е трудно да се направи съответствие между буквите на източника и буквите на криптограмите.

За да се избегне тази неприятна страна на модулярните шифри, се използват т.н. *субституционни шифри с много азбуки*. Тези криптоситеми са въведени от френския криптограф Б. дьо Виженер и се наричат още *шифри на Виженер*. Най-простият шифър на Виженер с $p$ азбуки дефинираме с $p$ функции $f_i(a) = a + k_i (\mathrm{mod}\, n)$, $i \in I_p$, където $(k_1, k_2, \ldots, k_p)$ е ключ на криптосистемата, $k_i \in J_n$. Тази система се състои от $p$ циклически измествания на изходната азбука. Криптограмата на открития текст

$$\alpha = a_1 a_2 \ldots a_p a_{p+1} a_{p+2} \ldots a_{2p} \ldots$$

е

$$f_1(a_1) f_2(a_2) \ldots f_p(a_p) f_1(a_{p+1}) f_2(a_{p+2}) \ldots f_p(a_{2p}) \ldots .$$

При шифриране е удобно да се използва таблицата на Виженер с нулев ред – изходната азбука и още толкова реда, колкото е размерът $p$ на ключа. Редът $i$ на таблицата е циклически изместената на $k_i$ позиции изходна азбука. При $n = 5$ и ключ „шифър“, част от таблицата на Виженер е дадена в Табл. 6.5

За да шифрираме с нея открит текст, записваме над буквите му последователно ключа толкова пъти, колкото е необходимо, за да може всяка буква на текста да получи съответна буква от шифъра. Всяка буква



| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| а | б | в | г | д | е | ж | з | и | й | к | л | м | н | о | ... |
| ш | щ | ъ | ь | ю | я | а | б | в | г | д | е | ж | з | и | ... |
| и | й | к | л | м | н | о | п | р | с | т | у | ф | х | ц | ... |
| ф | х | ц | ч | ш | щ | ъ | ь | ю | я | а | б | в | г | д | ... |
| ъ | ь | ю | я | а | б | в | г | д | е | ж | з | и | й | к | ... |
| р | с | т | у | ф | х | ц | ч | ш | щ | ъ | ь | ю | я | а | ... |

Таблица 6.5: Таблица на Виженер

$a_i$ от открития текст, на която е съпоставена буква $c$ от ключа, шифрираме с буквата от стълба на $a_i$ и реда на $c$. За съобщението „дискретна математика“ получаваме криптограмата „юрзжвяъгъюшъцирмраъ“.

За дешифриране, записваме по същия начин ключа над криптограмата толкова пъти, колкото е необходимо, за да може всяка буква на криптограмата да получи съответна буква от ключа. Всяка буква $b_i$ от криптограмата, със съпоставена буква $c$ от ключа, дешифрираме в тази буква от изходната азбука, която е надпис на стълба, който съдържа $b_i$, в реда надписан с $c$.

Забележете как еднакви букви от открития текст, например двете букви „м“, се криптират с различни букви. Това съществено променя честотните характеристики на криптирания текст и прави разбиването на шифъра по-трудно. Във всяка от отделните $p$ подпоследователности от съобщението, обаче, честотните характеристики на източника се запазват. Достатъчно е да се познае дължината $p$ на ключа на криптосистемата и тогава честотният анализ позволява бързото ѝ разбиване. През 1838 г. Ф. Касиски предлага прост способ за намиране дължината на ключа.

Използвани са варианти на шифрите с много азбуки. Например, шифрите с *автоматичен избор на ключ* определят изместването на поредната буква като функция на предходната. За да стартира шифрирането (и дешифрирането), е достатъчен прост ключ, определящ изместването на първата буква. Този вариант спестява не много безопасната размяна на ключове между кореспондентите. В 1883 г. Ж.-Г. Керкофс предлага сравнително прост начин за разбиване на шифрите с много азбуки, който не зависи от дължината на ключа. Наистина достатъчно е да разполагаме с много криптограми и да ги напишем една над друга. Тогава всички букви на криптограмите в позиция $i$ са шифрирани с един и същ щифър. Т.е. буквите в позиция $i$ образуват криптограма на шифър с просто отместване и криптосистемата може да бъде разбита с техниката на честотния анализ.

| а | з | б | у | к | и |
|---|---|---|---|---|---|
| в | й | г | х | м | л |
| д | н | е | ч | о | п |
| ж | р | с | т | ф | ц |
| ш | щ | ъ | ь | ю | я |

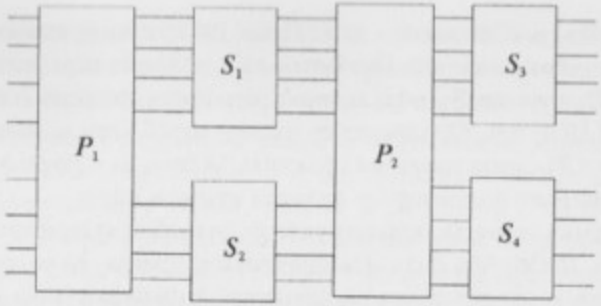
Таблица 6.6: Правоъгълник на Плайфер

Както се вижда, слабото място на субституционните шифри е многократното използване на ключа. Това дава предпоставки за статистическа атака и води до разбиване на шифъра. Единственият начин за осигуряване на абсолютна надеждност е да се използва ключ с потенциално безкрайна дължина. Този вариант на шифрите с много азбуки (шифър с безкрайно много азбуки) е известен с името *еднократен блокнот*. Той е абсолютно надежден, тъй като дори и да разшифрова изпратените до момента криптограми, криптоаналитикът не получава никакво знание за начина, по който ще бъдат криптирани следващите съобщения. Така всяко ново съобщение трябва да се дешифрира само за себе си, т.е. статистическите методи са неприложими. Неприятното на този шифър е необходимостта на обмяна на много дълги ключови последователности между кореспондентите.

Алтернатива на субституционните шифри, които разгледахме по-горе, са шифри, които заместват едновременно по няколко букви на изходната азбука с толкова букви на кодовата азбука – *полиграфовите шифри*. За пример да разгледаме един от най-популярните диграфови шифри, шифърът на Плайфер. Да подредим всички букви на българската азбука по случаен начин в правоъгълник с 5 реда и 6 стълба (вж. Табл 6.6):

Този правоъгълник е ключ на системата. При криптиране разделяме открития текст на последователни двойки букви. Да допуснем, че двете букви $a_i$ и $a_j$ от открития текст са в различни редове и стълбове на таблицата, т.е. определят правоъгълник. Например буквите „а“ и „я“ образуват такъв правоъгълник. Тогава заменяме двойката $a_i a_j$ с двойката букви $a_k a_l$, стоящи на другия диагонал на правоъгълника, като $a_k$ е в стълба на $a_i$, а $a_l$ – в стълба на $a_j$. В примера „ая“ трябва да заменим с „ши“. Ако двете букви са в един ред или един стълб, заменяме всяка от тях със съседната ѝ отдясно в реда, съответно – отдолу в стълба. Съседна на последната в реда или стълба буква е първата буква в този ред или стълб.






Фигура 6.5: Криптираща машина.

Да шифрираме с правоъгълника на Плайфер съобщението „дискретенматематик“. Получаваме криптограмата „апбфнсчсйожугожуку“. Дешифрирането става аналогично, с тази разлика, че за двойките от един ред (или един стълб) се вземат съседите им отляво (отгоре).

Шифрите, които разгледахме до тук, са по-скоро от исторически интерес. Сега да се спрем на съвременното състояние на криптосистемите с общ ключ.

***Дефиниция.*** Схема с $m$ входа и $m$ изхода такава, че при подаване на входовете

$$(a_1, a_2, \ldots, a_m) \in J_n^m$$

на изходите получаваме

$$(a_{\pi(1)}, a_{\pi(2)}, \ldots, a_{\pi(m)}) \in J_n^m,$$

където $\pi$ е пермутация от $\mathcal{P}_m$, наричаме *$P$-схема*. Схема с $m$ входа и $m$ изхода такава, че при подаване на входовете на вектора $\alpha \in J_n^m$ на изходите получаваме вектор $f(\alpha) = \beta \in J_n^m$, като функцията $f$ е взаимно еднозначна, наричаме *$S$-схема*.

$P$-схемите извършват пермутации на $m$-те цифри на въвежданото число, записано в $n$-ична бройна система, а $S$-схемите – субституции на такова число с друго $m$-цифрено число. Така с подходящо комбиниране на $P$-схеми и $S$-схеми може да бъде построена практически всяка от разглежданите по-горе криптосистеми с общ ключ.

През 1948-1949 г. Кл. Шенън предлага създаването на шифровални машини, в които последователно се редуват $P$-схеми и $S$-схеми, както е показано на Фиг. 6.5. На този принцип IBM разработва криптиращата машина LUCIFER. В LUCIFER $P$-схемите са с 64 или 128 двоични

входа/изхода, а $S$-схемите – с 4. През 1977 г. Американската Национална Стандартизационна Организация стандартизира под името DES (от Data Encryption Standard), криптографска система, аналог на 64-битовата LUCIFER. Системата се състои от 16 слоя и всеки от тях се управлява с 64-битов ключ (на практика 56 бита за ключ и 8 проверочни бита) генерирани вътрешно от зададен външен ключ.

На теория изглежда много трудно да се разбие криптосистема с ключ с дължина 16x56=896 бита. На практика се счита, че системата DES е ненадеждна от гледна точка на днешните възможности на компютрите. Критиците ѝ смятат, че ако се премине към 128-битови ключове, ще се получи неразбиваема система, но техническата реализация ще бъде много сложна.

Както се вижда от казаното по-горе, криптосистемите с общ ключ (симетричните криптосистеми) не са абсолютно надеждни или стават такива при много големи ключове. Като се има предвид, че в тези криптосистеми е необходимо двамата кореспонденти предварително да си разменят по достатъчно надежден начин ключове, става ясно защо криптоложките изследвания през последните години се ориентират в друга посока. Така през 1976 г. Дъфи и Хелман достигат до идеята за принципно други криптосистеми.

### 6.4.2 Криптосистеми с публичен ключ

Дъфи и Хелман предлагат да се построи криптосистема, в която шифриращият и дешифриращият ключ не се получават лесно един от друг (асиметрична криптосистема). Всеки участник $X$ в обмена на съобщения депозира в публичен каталог шифрираща процедура $E_X$, която е едновременно и неговият публичен ключ. В същото време $X$ пази в тайна лична процедура (а същевременно и ключ) за дешифриране $D_X$ такава, че $D_X(E_X(\alpha)) = \alpha$, за всеки открит текст $\alpha$ над изходната азбука. Освен да удовлетворяват изискванията, посочени по-горе, добре е криптосистемите с публичен ключ да са такива, че:

а) знанието на $E_X$ да не дава никаква информация за $D_X$;

б) $E_X(D_X(\alpha)) = \alpha$, за всеки открит текст $\alpha$ над изходната азбука.

Последното изискване не е абсолютно задължително, но както ще видим по-късно, дава допълнителни възможности, неприсъщи на симетричните криптосистеми.

Системата с публичен ключ се използва по следния начин. Кореспондентът $Y$ желае да изпрати конфиденциално съобщение $\alpha$ на $X$. Той намира в публичния каталог процедурата $E_X$, пресмята $\beta = E_X(\alpha)$ и



изпраща криптограмата $\beta$ по линията за връзка. Когато $X$ получи $\beta$, изчислява $D_X(\beta) = D_X(E_X(\alpha)) = \alpha$. Очевидно, при такава процедура никой освен $X$ не може лесно да дешифрира получената криптограма.

Криптосистемите с публичен ключ, за които е изпълнено и условието (б), могат да бъдат използвани за идентифициране на автора на съобщението. Този въпрос е важен, защото в системите с публичен ключ всеки може да изпрати шифровано съобщение и в него да представи като автор трето лице. С помощта на двустъпковата шифрираща техника, наричана *електронен подпис*, проблемът се решава напълно от системите с публичен ключ. $X$ иска да изпрати на $Y$ съобщение $\alpha$ така, че $Y$ да е абсолютно уверен, че $X$ е авторът на съобщението. $X$ пресмята $D_X(\alpha)$ с помощта на собствената си дешифрираща процедура, а след това $\beta = E_Y(D_X(\alpha))$. Криптограмата $\beta$ се изпраща по канала за връзка. При получаването ѝ $Y$ първо пресмята с личната си дешифрираща функция $D_Y(\beta) = D_Y(E_Y(D_X(\alpha))) = D_X(\alpha)$. След това с публичната криптираща функция на $X$ той пресмята $E_X(D_X(\alpha)) = \alpha$. Очевидно е, че на последната стъпка на дешифрирането ще се получи безсмислен текст, ако на първата стъпка на шифрирането не е използвана дешифриращата функция, за която се предполага, че само $X$ знае.

Ето и още едно приложение на системите с публичен ключ. За нуждите на взаимния контрол върху опитите с ядрени оръжия, страните $X$ и $Y$ се договарят да поставят, всяка на територията на другата страна, компютърни измерителни системи, които отчитат сеизмичните отклонения и могат с точност да установят всеки ядрен опит. Компютърните системи не могат да бъдат поставени под контрол без страната собственик да разбере, но не е трудно да се комутира каналът за връзка и да се подмени изпращаната информация с неотговаряща на истинската. Ако страната $X$ криптира информацията с тайната си дешифрираща функция $D_X$, тогава не е възможно да бъде подменена изпращаната информация, защото обработката ѝ с $E_X$ няма да даде смислен текст. Ако страната $Y$ предяви претенции, че под формата на сеизмични се изпращат и други данни, за събирането на които $X$ не е упълномощена, тогава е достатъчно да ѝ бъде предадена процедурата $E_X$, с която да се дешифрират изпращаните данни и така да отпаднат всички съмнения за техния характер.

Първата реална система с публичен ключ е създадена в 1978 година от Меркл и Хелман. Тя е построена на базата на следната

**Задача за раницата.** Дадени са $k, R \in N$ и множеството естествени числа $S = \{n_1, n_2, \ldots, n_k\}$. Да се намери $S' \subseteq S$ такова, че $\sum_{i \in S'} i = R$.

За пример нека $k = 10, R = 19758$ и

$$S = \{3313, 409, 7586, 1265, 924, 1995, 9192, 4804, 3612, 8256\}.$$

Предлагаме на читателя да се опита да реши сам тази задача за раницата и да прецени усилията, които трябва да положи при $k = 10$. В най-лошия случай всички $2^k$ подмножества на $S$ трябва да бъдат проверени. Сега да разгледаме следната

**Модифицирана задача за раницата.** Дадени са $k, R \in N$, множеството естествени числа $S = \{n_1, n_2, \ldots, n_k\}$ и $S' \subseteq S$. Да се провери дали $S'$ е такова, че $\sum_{i \in S'} i = R$.

Тази задача се решава много лесно. Достатъчно е да се сумират числата от $S'$ и да се сравни полученото с $R$. За горния пример лесно се вижда, че 3313+1265+924+8256=19758. Задачата за раницата принадлежи на множество от задачи, решаването на които е трудно, но ако има кандидат за решение, лесно може да се провери дали това е така.

Сега да разгледаме задача за раницата, в която числата на $S$ са такива, че $\sum_{j=1}^{i-1} n_j < n_i$, $i = 1, 2, \ldots, k$. Множеството от числа $S$ в този случай наричаме *хиперрастящо*. Задачата за раницата се решава тривиално за хиперрастящо множество от числа. Ще демонстрираме това за $S = \{7, 11, 47, 98, 201, 425, 2011, 3220\}$ и $R = 2230$. Числото 3220 не може да участва в търсеното подмножество, защото е по-голямо от $R$. Следващото число – 2011, може и трябва да участва, защото сумата на всички останали е по-малка от $R$. По същия начин отхвърляме 425, включваме задължително в конструираното множество 201, отхвърляме 98 и 47, а включваме 11 и 7. Получаваме 2230=7+11+201+2011. Така с около $|S|$ събирания и още толкова сравнения успяхме да решим задачата за раницата при хиперрастящо множество от числа.

Идеята за построяване на система с публичен ключ от задачата за раницата е да се преобразува една задача за раница с хиперрастящо множество в „скрито еквивалентна" задача, множеството от числа на която не е хиперрастящо. За целта Меркл и Хелман предлагат следния метод. Да изберем две взаимнопрости числа $m$ и $w$. Както видяхме в Лема 6.4.1, функцията $f(a) = wa (\mathrm{mod}\ m), a \in J_m$, е еднозначна, следователно съществува единствен елемент $a \in J_m$ такъв, че $f(a) = 1$. Означаваме този елемент с $w_m^{-1}$, или просто с $w^{-1}$, когато $m$ се подразбира. Наричаме го *мултипликативен обратен по модул* $n$ на $w$.

Нека $S^h = \{n_1^h, n_2^h, \ldots, n_k^h\}$ е хиперрастящо множество и нека $m > \sum_{i=1}^{k} n_i^h$. Да преобразуваме множеството $S^h$ в $S = \{n_1, n_2, \ldots, n_k\}$, където $n_i = wn_i^h (\mathrm{mod}\ m)$. С голяма вероятност може да се очаква, че след



преминаване през тази операция по модул $m$ полученото множество $S$ вече не е хиперрастящо. Избираме една наредба на числата от $S$, например зададената по-горе и публикуваме вектора $n_S = (n_1, n_2, \ldots, n_k)$ в каталога на публичните ключове, като запазваме в тайна $m, w$, мултипликативния обратен $w^{-1}$ на $w$ по модул $m$ и съответния вектор $n_S = (n_1^h, n_2^h, \ldots, n_k^h)$.

Нека изходната ни азбука е предварително кодирана с равномерен двоичен код, така че всички открити текстове са думи над $\{0, 1\}$. Нека $\alpha = x_{1,1}x_{1,2}\cdots x_{1,k}x_{2,1}x_{2,2}\cdots x_{2,k}\ldots x_{r,1}x_{r,2}\cdots x_{r,k}$ е открито съобщение, което кореспондент иска да ни изпрати. За всяка $k$-орка битове $x_{i,1}x_{i,2}\cdots x_{i,k}$, $i = 1, 2, \ldots, r$, той пресмята числото $p_i = \sum_{j=1}^{k} n_j x_{i,j}$ и изпраща по канала за връзка криптограмата $p_1, p_2, \ldots, p_r$. За декриптирането пресмятаме

$$\begin{aligned} p_i^h &= w^{-1}p_i (\mathrm{mod}\ m) = \\ &= w^{-1}n_1x_{i,1} + w^{-1}n_2x_{i,2} + \cdots + w^{-1}n_kx_{i,k} (\mathrm{mod}\ m) = \\ &= w^{-1}wn_1^hx_{i,1} + w^{-1}wn_2^hx_{i,2} + \cdots + w^{-1}wn_k^hx_{i,k} (\mathrm{mod}\ m) = \\ &= n_1^hx_{i,1} + n_2^hx_{i,2} + \cdots + n_k^hx_{i,k} (\mathrm{mod}\ m) = \\ &= n_1^hx_{i,1} + n_2^hx_{i,2} + \cdots + n_k^hx_{i,k}. \end{aligned}$$

На последната стъпка сме използвали, че $m > \sum_{i=1}^{k} n_i^h$. Остава да се намерят $x_{i,1}, x_{i,2}, \ldots x_{i,k} \in \{0, 1\}$ такива , че $p_i^h = n_1^hx_{i,1} + n_2^hx_{i,2} + \cdots + n_k^hx_{i,k}$. Но това е задача за раницата с хиперрастящо множество и лесно намираме съответните стойности на $x_i$, а с това е дешифрирана поредната порция от криптограмата.

Да съставим криптосистема на Меркл-Хелман за азбуката

$$\{\text{а,д,е,и,к,м,н,р,с,т}\},$$

която предварително сме кодирали с равномерен двоичен код по следния начин

$$\begin{gathered} \text{а} - 0011,\ \text{д} - 0101,\ \text{е} - 0110,\ \text{и} - 1001,\ \text{к} - 1010 \\ \text{м} - 1100,\ \text{н} - 0111,\ \text{р} - 1011,\ \text{с} - 1101,\ \text{т} - 1110. \end{gathered}$$

За декриптиране ще използваме хиперрастящото множество

$$S = \{7, 11, 47, 98, 201, 425, 2011, 3220\},$$

разгледано по-горе. Сумата на елементите му е 6020, затова да изберем $m = 6021 = 3^3.223$ и взаимно простото с него $w = 10000 = 2^4.5^4$.

Мултипликативният обратен на $w$ по модул $m$ е $w^{-1} = 2179$, защото $2179.10000 = 3619.6021 + 1 \equiv 1(\text{mod } 6021)$.

Да трансформираме хиперрастящото множество, както посочихме по- горе:

$$\begin{aligned}
10000.7 &= 70000 \equiv 3769(\text{mod } 6021)\\
10000.11 &= 110000 \equiv 1622(\text{mod } 6021)\\
10000.47 &= 470000 \equiv 362(\text{mod } 6021)\\
10000.98 &= 980000 \equiv 4598(\text{mod } 6021)\\
10000.201 &= 2010000 \equiv 5007(\text{mod } 6021)\\
10000.425 &= 4250000 \equiv 5195(\text{mod } 6021)\\
10000.2011 &= 20110000 \equiv 5881(\text{mod } 6021)\\
10000.3220 &= 32200000 \equiv 5713(\text{mod } 6021)
\end{aligned}$$

След пресмятанията ясно се вижда ефектът на трансформацията – множеството от получените числа вече не е хиперрастящо. Публичният ключ на построената криптосистема е

$$3769, 1622, 362, 4598, 5007, 5195, 5881, 5713.$$

За да криптираме текста „дискретнаматематика“, предварително го преобразуваме в двоичен вид с по-горе дефинираната трансформация, като разделяме текста на поддуми с дължина 8, колкото е дължината на ключа. Недостигащите битове попълваме с 0. Получаваме съобщението

$$\begin{array}{c}
01011001\ 11011010\ 10110110\ 11100111\ 00111100\\
00111110\ 01101100\ 00111110\ 10011010\ 00110000.
\end{array}$$

За криптирането трябва да заместим всеки 8-мерен двоичен вектор $\alpha_i$ със сумата от числата на подмножество на ключа, за което $\alpha$ е характеристичен вектор. За първата осмица получаваме $1662 + 4598 + 5007 + 5713 = 16940$, за втората осмица – $3769+1622+4598+5007+5881 = 20877$ и т. н., докато криптираме цялото съобщение и получим криптограмата

$$16940, 20877, 19805, 22542, 15162, 21043, 12186, 21043, 19255, 4960.$$

За да я декриптираме, умножаваме всяко от получените числа с $w^{-1} = 2179$ по модул 6021. За първото число резултатът е $16940.2179 = 3530(\text{mod } 6021)$. Решаваме задачата за раницата за 3530 и хиперрастящото множество. Получаваме

$$3530 = 0.7 + 1.11 + 0.47 + 1.98 + 1.201 + 0.425 + 0.2011 + 1.3220$$



и следователно първите 8 бита на изпратеното съобщение са 01011001, което съответства на двойката букви „ди“. За останалите числа получаваме

$$\begin{aligned}
20877.2179 &\equiv 2328(\text{mod } 6021),\\
19805.2179 &\equiv 2588(\text{mod } 6021),\\
22542.2179 &\equiv 5721(\text{mod } 6021),\\
15162.2179 &\equiv 771(\text{mod } 6021),\\
21043.2179 &\equiv 2782(\text{mod } 6021),\\
12186.2179 &\equiv 684(\text{mod } 6021),\\
19255.2179 &\equiv 2317(\text{mod } 6021),\\
4960.2179 &\equiv 145(\text{mod } 6021),
\end{aligned}$$

за всяко от тях решаваме задачата за раницата с хиперрастящото множество и получените коефициенти образуват съответните декриптирани последователности.

Криптосистемата на Меркл-Хелман не може да бъде използвана за електронен подпис. Нейната криптираща функция превръща $n$-орки битове в цели числа, а декриптиращата – целите числа, получени от криптирането, в $n$-орки битове. Тъй като не всяко цяло число може да се обърне в $n$-орка битове (с помощта на задачата за раницата), изпращачът не може да обработи с декриптиращата си функция изпращаното съобщение.

Пример на криптосистема с публичен ключ, която може да бъде използвана и за електронен подпис, е системата на Ривъст-Шамир-Аделман (известна като RSA), разработена в 1978г. Тук накратко ще изложим същността на тази криптосистема. За целта ще са необходими някои резултати от теорията на числата.

**Теорема 6.4.1** *Нека $p_1, p_2, \ldots, p_k$ са всички различни прости делители на естественото положително число $n$. Тогава*

$$\phi(n) = n(1 - \frac{1}{p_1})(1 - \frac{1}{p_2}) \cdots (1 - \frac{1}{p_k}).$$

***Доказателство.*** Очевидно $\frac{n}{p_{i_1} p_{i_2} \cdots p_{i_r}}$ е броят на числата, по-малки от $n$, които се делят едновременно на $p_{i_1}, \cdots, p_{i_{r-1}}$ и $p_{i_r}$, а $\phi(n)$ е броят на числата, по-малки от $n$, които не се делят на никое от $p_1, \cdots, p_{k-1}$ и $p_k$. Затова от Принципа за включване и изключване получаваме

$$\phi(n) = n - \frac{n}{p_1} - \frac{n}{p_2} - \cdots - \frac{n}{p_k} +$$

$$+\frac{n}{p_1p_2}+\frac{n}{p_1p_3}+\cdots+\frac{n}{p_{k-1}p_k}+$$
$$+\cdots+(-1)^k\frac{n}{p_1p_2\dots p_k}=$$
$$= n(1-\frac{1}{p_1}-\frac{1}{p_2}-\cdots-\frac{1}{p_k}+$$
$$+\frac{1}{p_1p_2}+\frac{1}{p_1p_3}+\cdots+\frac{1}{p_{k-1}p_k}+$$
$$+\cdots+(-1)^k\frac{1}{p_1p_2\dots p_k})=$$
$$= n(1-\frac{1}{p_1})(1-\frac{1}{p_2})\cdots(1-\frac{1}{p_k}).\ \square$$

Да приложим тази теорема за $n=pq$, където $p$ и $q$ са прости числа. Получаваме $\phi(n)=n(1-\frac{1}{p})(1-\frac{1}{q})=\frac{n}{pq}(p-1)(q-1)=(p-1)(q-1)$.

Без доказателство ще посочим следната важна

**Теорема 6.4.2 (Л. Ойлер)** *Ако $m$ и $a$ са взаимно прости числа, тогава $a^{\phi(m)}\equiv 1(\mathrm{mod}\ m)$.*

и непосредствено получаващото се от нея

**Следствие 6.4.2** *Ако $m$ и $a$ са взаимно прости числа, тогава за всяко естествено $k$ е в сила $a^{k\phi(m)}\equiv 1(\mathrm{mod}\ m)$.*

Сега да построим криптосистема на Ривъст-Шамир-Аделман:

1. Избираме две големи прости числа $p$ и $q$, които запазваме в тайна. Колкото по-големи са $p$ и $q$, толкова по-надеждна е криптосистемата.

2. Пресмятаме произведението $n$ на $p$ и $q$. Пресмятаме в съответствие с Теорема 6.4.1 $\phi(n)=(p-1)(q-1)$.

3. Избираме естествено число $E$, взаимно просто с $\phi(n)$. Поместваме двойката $(n,E)$ в каталога на публичните ключове. Тя е нашият публичен ключ.

4. Тъй като $E$ и $\phi(n)$ са взаимно прости, можем да пресметнем $D$ – мултипликативния обратен на $E$ по модул $\phi(n)$, т.е. $ED\equiv 1(\mathrm{mod}\ \phi(n))$ или $ED=1+k\phi(n)$ за някое $k$. Запазваме и $D$ в тайна. То е нашият скрит ключ.

5. За криптиране на информацията е необходимо да я преобразуваме по произволен начин в цифров вид. Най-просто е да заместим всяка буква на открития текст с нейния пореден номер в азбуката (с водещата нула, ако има такава). Получената последователност от цифри разбиваме на *блокове* $M_1,M_2,\dots,M_r$ така, че всяко от числата $m_1,m_2,\dots,m_r$,



които се образуват от цифрите на тези блокове, да не надхвърля $n$. Криптирането се състои в пресмятане на последователността $c_1,c_2,\dots,c_r$, където $c_i=m_i^E(\mathrm{mod}\ n)$.

6. За декриптирането пресмятаме

$$c_i^D\equiv m_i^{ED}(\mathrm{mod}\ n)=m_i^{1+k\phi(n)}(\mathrm{mod}\ n)=m_i^1m_i^{k\phi(n)}(\mathrm{mod}\ n)$$

Прилагаме Следствие 6.4.2 и получаваме

$$c_i^D\equiv m_i(\mathrm{mod}\ n)=m_i,$$

тъй като $0\le m_i\le n-1$. От получената цифрова последователност лесно възстановяваме открития текст.

За пример да построим криптосистема с $p=3,q=11$. (Напомняме, че за реални криптосистеми $p$ и $q$ трябва да са много големи!). Сега $n=33$, а $\phi(n)=20$. Ако за $E$ изберем 3, което е взаимно просто с 20, то мултипликативният обратен на 3 по модул 20 е $D=7$, защото $7.3=21\equiv 1(\mathrm{mod}\ 20)$. Тъй като буквите на българската азбука са 30, то номерът на всяка буква образува отделен блок. За съобщението „дискретнаматематика“, след заместване на буквите с поредните им номера в азбуката (за буквата „а“ вместо 1 вземаме 31) получаваме:

$$M=5,9,18,11,17,6,19,14,31,13,31,19,6,13,31,19,9,11,31.$$

Криптираме съобщението като пресмятаме последователно

$5^3=125\equiv 26(\mathrm{mod}\ 33)$,

$9^3=739\equiv 3(\mathrm{mod}\ 33)$,

$18^3=5832\equiv 24(\mathrm{mod}\ 33)$ и т.н.

Получаваме криптираното съобщение

$$26,3,24,11,29,18,28,5,25,19,25,28,29,19,25,28,3,11,25.$$

За декриптиране пресмятаме последователно:

$26^7=8031810176\equiv 5(\mathrm{mod}\ 33)$,

$3^7=2187\equiv 9(\mathrm{mod}\ 33)$,

$24^7=4586471424\equiv 18(\mathrm{mod}\ 33)$

и т.н. до получаване на открития текст.

Трудността на разбиването на криптосистемата на Ривъст-Шамир-Аделман се определя от това колко трудно по известното $n$ се пресмятат множителите му $p$ и $q$. Съвременните компютри с най-ефективни алгоритми разлагат 200-цифрени десетични числа за няколко часа. Очевидно за пълна сигурност трябва да се вземат много по-големи числа.

## Упражнения

**Упражнение 6.1**

Постройте пример на разделим код на азбука с произволен, зададен брой букви, който не е префиксен.

**Упражнение 6.2**

Постройте кодове на Фано-Шенън и Хафман за източниците:
а) $p_1 = 1/2, p_2 = 1/3, p_3 = 1/6$;
б) $p_1 = 1/3, p_2 = 1/4, p_3 = 1/5, p_4 = 1/6, p_5 = 1/20$;
в) $p_1 = 1/2, p_2 = 1/4, p_3 = 1/8, p_4 = 1/16, p_5 = 1/16$;
г) $p_1 = 27/40, p_2 = 9/40, p_3 = 3/40, p_4 = 1/40$.

**Упражнение 6.3**

За източника $I = (A = \{a, b\}; p_a = 1/3, p_b = 2/3)$ постройте източници за азбуките с 4 и 8 букви, съответни на произволни думи с 2 или 3 букви от $A$. Пресметнете ентропията на получените източници.

**Упражнение 6.4**

Колко грешки може да открива код, който поправя $t$ грешки.

**Упражнение 6.5**

Постройте множество $C$ от 5-мерни двоични вектори с максимален брой елементи такова, че разстоянието между всеки два различни вектора на $C$ е поне 3.

**Упражнение 6.6**

Нека

$$G_C = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

е пораждаща матрица на линеен двоичен [5,2] код $C$. Постройте проверочна матрица на $C$. Определете коригиращите му свойства.



**Упражнение 6.7**

Постройте стандартно разлагане на Слепян за кода от упражнение 6.6. Пресметнете синдромите на съседните класове. Декодирайте получената дума 10101.

**Упражнение 6.8**

Докажете, че кодовете на Хеминг са съвършени.

**Упражнение 6.9**

Докажете, че всеки код с r-кратно повторение

$$\mathcal{D}_{r\times k} = \{\alpha\alpha\ldots\alpha | \alpha \in J_2^k\}$$

е линеен $[rk, k]$ код. Постройте пораждаща и проверочна матрица на $\mathcal{D}_{2\times 3}$.

**Упражнение 6.10**

Докажете, че всеки код с проверка по четност

$$\mathcal{PC}_k = \{\alpha a | \alpha \in J_2^k, a \equiv wt(\alpha) (\mathrm{mod}\, 2)\}$$

е линеен $[k+1, k]$ код. Постройте пораждаща и проверочна матрица на $\mathcal{PC}_3$.

**Упражнение 6.11**

Линеен код $C[2k, k]$, който съвпада с ортогоналния си $C^\perp$, наричаме *самоортогонален*. Постройте самоортогонален двоичен $[4, 2]$ код.

**Упражнение 6.12**

Докажете, че в линеен двоичен код или всички вектори са с четно тегло, или точно половината са с четно тегло.

**Упражнение 6.13**

Докажете, че в линеен код или всички вектори имат нулева $i$-та компонента, или точно половината имат нулева $i$-та компонента.

### Упражнение 6.14

Нека $G_1$ и $G_2$ са пораждащи матрици на двоичен $[n_1, k, d_1]$ и $[n_2, k, d_2]$ код, съответно. Докажете, че

$$\left\| \begin{array}{cc} G_1 & O \\ O & G_2 \end{array} \right\| \text{ и } \|G_1\, G_2\|$$

са съответно $[n_1+n_2, 2k, \min\{d_1, d_2\}]$ и $[n_1+n_2, k, d]$ двоични кодове, като $O$ е матрица съставена само от нули, а $d \geq d_1 + d_2$.

### Упражнение 6.15

Нека $C_1$ и $C_2$ са линейни кодове, а

$$C_1 + C_2 = \{\alpha_1 + \alpha_2 | \alpha_1 \in C_1, \alpha_2 \in C_2\}.$$

Докажете, че $C_1 + C_2$ е линеен код такъв, че $(C_1 + C_2)^{\perp} = C_1^{\perp} \cap C_2^{\perp}$.

### Упражнение 6.16

Криптирайте съобщението „съвременнакриптология“ с шифъра на Виженер и с ключ думата „компютър“. Декриптирайте със същия ключ криптограмата „дцвлобъттфсюгд“.

### Упражнение 6.17

Криптирайте с правоъгълника на Плайфер от Табл. 6.6 съобщението „краткооткритосъобщение“. Декриптирайте със същия правоъгълник криптограмата „жгчдфепмдряацкзц“.

### Упражнение 6.18

Декриптирайте с помощта на статистиката от раздел 6.2 получената с циклическо изместване криптограма:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| дгдеь | мьвьг | звуюе | уъаьм | швкуе | уязше | дгаьз | ьмшжя |
| ьхгшв | вььяг | вгбьм | шжяьь | взьбв | ььзвд | еьзгх | угошг |
| зчпаф | гяуче | шхвгж | зжшшв | ууацу | агчуж | шягчь | еуьвй |
| гебул | ьтдгз | уяпхв | умьвм | шхагщ | швьтз | хвштж | бьжпа |
| чугжз | ухужя | еьзъу | аьлуь | ааьвж | зьзил | ььягъ | згвшж |
| уидпа | вгбго | швьчу | тъъдг | аъхуз. | | | |



### Упражнение 6.19

Зашифрирайте с модулярен шифър с $n = 7, k = 19$ съобщението „строгосекретно“.

### Упражнение 6.20

С дефинираната в раздел 6.4 криптосистема на Меркъл-Хелман шифрирайте като подател съобщението „секретнимисии“. Дешифрирайте като получател криптограмата 26590,19449,16473,16111,26072,18625,19087.

### Упражнение 6.21

С дефинираната в раздел 6.4. криптосистема на Ривъст-Шамир-Аделман шифрирайте като подател съобщението „публиченключ“. Дешифрирайте като получател криптограмата 18,12,18,11,28,29,9,5,18,5,4,9,26,9,3,24.

# Глава 7

# Масови задачи и алгоритми

В тълковния Речник на българския език (т.5, Издателство на БАН, 1987 г.) намираме следното тълкуване на думата *задача* (в математически смисъл): „Упражнение по математика, физика и др., което се разрешава чрез разсъждения и изчисления". Без преувеличение може да се каже, че същността на математиката е решаването на задачи. В Доклада си пред Втория международен конгрес на математиците в Париж, 6-12 август 1900г., Давид Хилберт посочва, че състоянието на всяка наука до голяма степен зависи от формулирането на сериозни задачи и от развитието на средствата за тяхното решаване. „Всяка научна област е жизнеспособна", твърди Хилберт, „докато в нея има излишък от нови задачи. Недостигът от нови задачи означава отмиране или прекратяване на самостоятелното развитие." Друг от корифеите на съвременната математика, ненадминатият „решавач на задачи" Дьорд Пойя пише: „Да имаме задача означава да търсим съзнателно някое действие, годно за постигането на една ясно схващана, но не и непосредствено достижима цел. Да се реши една задача означава да се намери това действие."

В тази глава ще разгледаме неформалните понятия *задача* и *решение* и ще се спрем на възможностите за формализация на тези понятия и изследване с математически апарат на формалните им модели.

## 7.1 Масови задачи

По-горе в изложението посочихме много задачи и техните решения. Тук ще обърнем внимание на една важна особеност на математическите задачи, като въведем неформалното понятие *масова задача*. Нека е зададено произволно множество от математически обекти с добре дефинирани свойства и съвкупност от операции между обекти, даващи в резултат обект от същото множество. Под масова задача ще разбираме всяка дос-

татъчно ясна цел, състояща се в намиране (с последователно прилагане на допустими операции) на някакви обекти $\tilde{y} = (y_1, y_2, \ldots)$ от множеството, с предварително определени свойства, започвайки от зададени обекти $\tilde{x} = (x_1, x_2, \ldots)$, също с предварително определени свойства. Всеки от параметрите $\tilde{x} = (x_1, x_2, \ldots)$ се мени в някакви, по-широки или по-тесни, области от стойности. С всяко стесняване на областта на някои от параметрите или ограничаването им до единствено възможни стойности, се получава нова масова задача, която наричаме *екземпляр* на първата масова задача. При стесняване на всички параметри до единствени стойности ще стигнем до масова задача, която съвпада с единствения си екземпляр.

Ето някои примери на масови задачи:

Пример 1. *„Дадени са елементите $a, b$ и $c$ на крайното поле $GF(q)$. Да се намерят всички $x \in GF(q)$ такива, че $ax^2+bx+c = 0$.“* и *„Дадени са елементите $a, b, c$ и $x$ на крайното поле $GF(q)$. Да се провери дали $ax^2 + bx + c = 0$.“*
са масови задачи, а
*„Да се намерят всички елементи $x$ на $GF(3)$, такива че $x^2 + 2 = 0$.“* и *„Да се провери, че за $x = 2$ в полето $GF(3)$ е в сила $x^2 + 2 = 0$.“*
са екземпляри на тези масови задачи.

Пример 2. *„Дадени са естествените числа $i$ и $j$. Да се намери НОД на $i$ и $j$.“* и *„Дадени са естествените числа $i, j$ и $k$. Да се провери дали $k$ е НОД на $i$ и $j$.“*
са масови задачи, а
*„Да се намери НОД на 12 и 30.“* и *„Да се провери дали 6 е НОД на 12 и 30.“*
са екземпляри на тези масови задачи.

Пример 3. *„Дадени са множество от елементи (линейни и ъглови) на равнинен (планиметричен) обект. Да се построи този обект само с линийка и пергел.“*
е масова задача. Задачата
*„Да се построи триъгълник по зададени две страни $a, b$ и ъгъл $\alpha$ между тях.“*
е неин екземпляр и също е масова задача. При задаване на конкретни страни и ъгъл получаваме екземпляр на тази масова задача.

В първия пример множеството от обекти е съставено от елементите на някакво поле, във втория пример – от естествените числа, а в третия – от геометричните обекти: точки, прави, ъгли, отсечки и т.н. Операциите в първите два примера са аритметичните и операциите за сравняване, а в третия – изпълнимите с линийка и пергел: начертаване на (част от) права, начертаване на окръжност, определяне пресечната точка на два обекта и т.н.



Ще обърнем внимание, че под решение на дадена задача в математиката разбираме не само крайния резултат, но и последователността от операции, с които сме го намерили.

В първите два примера са дадени двойки масови задачи, които твърде много си приличат, но имат и съществени различия. Д. Пойя нарича първите *„задачи за намиране* (на решение, б.а.)“, а вторите - „задачи за доказателство“. Поради честата употреба на термина „доказателство“, могат да се посочат масови задачи за намиране на решение, във формулировката на които се среща фразата „докажете, че...“. За да избегнем недоразуменията, ще наричаме задачите от втория тип *„задачи за проверка* (на решение)“. Като още един пример на такава двойка задачи ще посочим двата варианта на Задачата за раницата, разгледани в 6.4.2.

Ако сравним масовите задачи от Пример 1, с тези от Пример 2 ще забележим още една разлика, важна за класифицирането на задачите. В Пример 2 се говори не просто за общ делител на две естествени числа, а за най-големия им общ делител, т.е. общ делител с екстремалното свойство да бъде по-голям от всички останали общи делители на двете числа. Такива масови задачи наричаме „задачи за оптимизация“. Няколко оптимизационни задачи разгледахме в предходните глави. Задачите за оптимизация имат своя специфика и за нашите цели е по-удобно да ги заменим с неоптимизационни, като си даваме сметка, че получените задачи са различни от началните. Една достатъчно обща техника за извършване на тази замяна се състои в следното: въвеждаме нов параметър $B$ на задачата и вместо оптималното решение търсим решение, което е в лесно проверимо отношение с $B$, например, ограничено (отдолу или отгоре) от $B$.

С по-горе описаната техника задачите от Пример 2 се трансформират в: *„Дадени са естествените числа $i, j$ и $B$. Да се намери ОД $k$ на $i$ и $j$ такъв, че $k \geq B$.“* и *„Дадени са естествените числа $i, j, k$ и $B$. Да се провери дали $k$ е ОД на $i$ и $j$ такъв, че $k \geq B$.“*

Понятието масова задача е неформално и не може да бъде изследвано с математически средства. За целта е необходимо да изберем математически формализъм, който достатъчно добре да отразява същността на това понятие. Такъв формализъм може да бъде понятието функция. Ще се ограничим до масови задачи с изброимо множество екземпляри. Нека $A$ е крайна азбука. Да представим всеки екземпляр $\tilde{x}$ на масовата задача $\pi$ за намиране на решение и съответния резултат $\tilde{y}$ с думи $\alpha_{\tilde{x}}$ и $\alpha_{\tilde{y}}$ от $A^*$. Масовата задача $\pi$ можем да представим с функцията

$f_\pi : A^* \to A^*$ такава, че $f_\pi(\alpha_{\tilde{x}}) = \alpha_{\tilde{y}}$, $\forall$ екземпляри $\tilde{x}$ на $\pi$. Ако пък $\pi$ е задача за проверка на решение можем да я представим с функцията $f_\pi : A^* \to \{true, false\}$ такава, че $f_\pi(\alpha_{\tilde{x}}) = true$, $\forall$ екземпляри $\tilde{x}$ на $\pi$, за които проверката завършва с успех. Това означава, че задачите за проверка на решение можем да наречем и „задачи за разпознаване на език" – понятие, което читателят познава добре от Глава 4. За задачите за намиране на решение пък ще използваме по-естественото от гледна точка на формализма понятие „задачи за изчисляване на функция". Очевидно, разпознаването на език е частен случай на изчисляването на функция.

Изчисляването на $f_\pi(\alpha_{\tilde{x}})$ по зададен екземпляр $\tilde{x}$, с помощта на последователност от допустими операции, е математическият модел на неформалното понятие „решаване на задачата $\pi$".

## 7.2 Алгоритми и разрешимост на масови задачи

В този раздел ще класифицираме масовите задачи с изброимо множество екземпляри по много съществен признак. За масовите задачи за изчисляване на функции от Пример 1 и Пример 2 на предния раздел лесно можем да посочим процедури (последователности от позволените операции), които ни позволяват с краен брой *стъпки* да намерим решение на задачата, независимо от това кой от екземплярите ѝ е зададен. В първия случай процедурата се задава от формулата за корените на квадратното уравнение, а за втория пример процедурата е известният алгоритъм на Евклид. Съответните им задачи за разпознаване на език се решават с елементарни проверки. За втората масова задача от Пример 3 съществува добре известна процедура за построяване на триъгълник по произволни зададени две страни и ъгъл между тях.

По-различно е положението с първата масова задача от Пример 3. Не е известна достатъчно обща процедура за решаването на тази масова задача, която не зависи от конкретния екземпляр. За много от екземплярите знаем специфични решения, но те са толкова различни едно от друго, че за всеки нов екземпляр решаването трябва да започне отново. Знаем също, че за някои екземпляри такова решение не съществува.

Процедурите, които решават масови задачи, независимо от това кой екземпляр е зададен, наричаме *алгоритми* (от името на средновековния математик Ибрахим ал Хорезми). Тълковният Речник на българския език описва неформалното понятие алгоритъм така: „Система от правила, които определят последователност от изчислителни операции, прилагането на които води до решаването на дадена задача. ... ".

Масова задача, за която съществува алгоритъм, който я решава, ще



наричаме *алгоритмически разрешима*, а такава, за която не съществува алгоритъм – *алгоритмически неразрешима*. Да се определи дали една масова задача е алгоритмически разрешима е изключително трудно и в класификацията на масовите задачи освен разрешими и неразрешими алгоритмически, има и такива, за които все още не знаем разрешими ли са или не.

Класификацията на масовите задачи според алгоритмичната им разрешимост се пренася и върху моделиращите ги функции. Функциите съответни на алгоритмически разрешимите задачи ще наречем *ефективно (алгоритмически) изчислими*. Както не е съвсем ясно очертано множеството от алгоритмически разрешимите масови задачи, така не е добре дефинирано и съответното множество от формалните им математически модели – множеството на ефективно изчислимите функции.

С помощта на вече изучения материал лесно можем да посочим алгоритмични процедури за цели класове от функции и по този начин да установим тяхната ефективна изчислимост, т.е. алгоритмическата разрешимост на съответните масови задачи.

Например, нека масовата задача $\pi$ се представя с функция $f_\pi : B \to A^*$ с крайна дефиниционна област $B \subseteq A^*$. Тогава $f$ може да бъде зададена с таблицата си с два стълба и $|B|$ реда. В първия елемент на всеки ред е записана някоя от стойностите на дефиниционната област, като в различните редове са записани различни стойности. Във втория елемент – съответната стойност на функцията. Тривиалният алгоритъм за ефективното изчисляване на $F_\pi(a)$ се състои в последователна проверка на съдържанията на първия стълб до откриване на реда съдържащ $a$. Тогава съдържанието на втория стълб в този ред е стойността на функцията. Тази процедура е известна като *пълно изчерпване*. При всичките си недостатъци, тя решава въпроса за ефективната изчислимост на функции с крайна дефиниционна област, зададени таблично. Ние вече знаем и други начини за изчисляване на такива функции. С просто кодиране на елементите на дефиниционната област и областта на изменение, всяка такава функция може да бъде представена като крайно множество дискретни (например булеви) функции, за които да бъдат построени съответни формули (ДНФ или съответните им $q-$ични аналози) или схеми от функционални елементи. Процедурите за пресмятане на функция по формула или чрез схема от функционални елементи, дискутирани в съответните раздели на Глава 5, са примери на алгоритмични процедури.

Алгоритмични процедури за решаване на някои масови задачи с изброимо множество от екземпляри разгледахме в Глава 4. Задачата за разпознаване на произволен автоматен език се решава от прост алгори-

тъм, който проследява работата на съответния краен автомат. С подобна процедура, моделираща работата на недетерминиран стеков автомат, разпознаваме и контекстно-свободните езици.

Както се вижда, някои масови задачи могат да бъдат решавани в рамките на ограничена съвкупност от алгоритми. В такъв случай казваме, че тези задачи са разрешими в рамките на тясната съвкупност. Например, задачата за разпознаване на автоматен език е *автоматно разрешима*. Задачата за разпознаване на произволен контекстно-свободен език не е автоматно разрешима, но е *разрешима с недетерминиран стеков автомат* и т.н.

Основен въпрос е какъв формализъм за изчисление на функции да се избере, така че с голяма доза увереност да може да твърдим, че множеството на изчислимите в рамките на този формализъм функции съвпада с множеството на ефективно изчислимите функции. Математиката предлага няколко такива формализма, на един от които ще се спрем в следващия раздел.

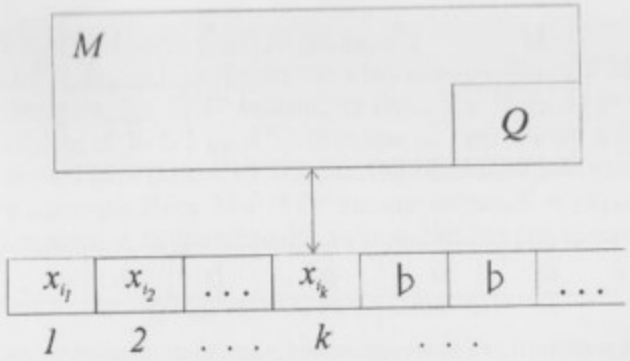
## 7.3 Машини на Тюринг

На Фиг. 7.1 е представена схематично абстрактна математическа машина. Тя чете входа си от безкрайно в едната посока (например, надясно) запомнящо устройство, наричано *лента* и записва изхода си на същото устройство. Входно-изходната лента е разделена на изброимо множество еднотипни клетки, във всяка от които може да бъде записана някоя от буквите на крайна (входно-изходна) азбука. Клетките са номерирани (отляво надясно) с естествените числа $1, 2, \ldots, k, \ldots$. Четенето от лентата и писането на нея се извършва от четящо-пишещо устройство, наричано *глава*. Главата е подвижна и в края на всеки такт може да се премести с една клетка наляво (ако не е била на най-лявата клетка), една клетка надясно или да остане на място. Означаваме с $L$, $R$ и $S$ тези възможности за преместване на главата.

***Дефиниция.*** Петорката $M =< Q, X, q_0, \delta, F >$, където $Q$ е крайно множество от състояния; $X$ е крайна азбука; $q_0 \in Q$ е начално състояние; $F$ са заключителни състояния, $F \cap Q = \emptyset$; $\delta : Q \times X \to (Q \cup F) \times X \times \{L, R, S\}$ е функция на преходите, наричаме *машина на Тюринг (МТ)* (с безкрайна в едната посока лента).

За разлика от вече разгледаните абстрактни машини, заключителните състояния на МТ са отделени от останалите състояния и са такива, че попадайки в заключително състояние машината не може да продължи да работи – stop-състояния.






Фигура 7.1: Машина на Тюринг.

Една от буквите на азбуката $X$ има специално предназначение. Наричаме я *запълваща буква (бленк)* и я бележим с $\flat$. Ще разглеждаме машините на Тюринг с условието, че преди започване на работата *крайно множество от клетки на лентата не съдържат запълващата буква*. При това условие, след краен брой тактове, клетките несъдържащи бленк ще продължат да бъдат крайно множество.

***Дефиниция.*** Нека клетката с номер $k$ е последната несъдържаща бленк клетка на лентата на МТ $M$. Думата $x_{i_1}x_{i_2}\ldots x_{i_k} \in X^*$ такава, че $x_{i_j}$ е записана в клетката с номер $j$ наричаме текуща *лентова дума* на $M$.

***Дефиниция.*** Нека МТ $M =< Q, X, q_0, \delta, F >$ е в състояние $q \in Q$, текущата лентова дума е $\alpha' a \alpha'' \in X^*$, а главата на машината се намира над буквата $a$ (вж. Фиг. 7.2). Тройката $(\alpha', q, a\alpha'')$ наричаме *конфигурация* на МТ $M$.

Очевидно, конфигурацията на МТ еднозначно определя бъдещото ѝ поведение. Конфигурациите от вида $(\varepsilon, q_0, \alpha)$ наричаме *начални конфигурации*. Ще опишем формално работата на $M$ чрез трансформация на конфигурации.

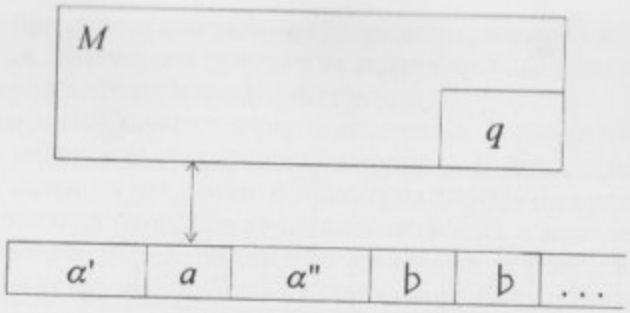
***Дефиниция.*** Релацията $R_{\vdash} \subseteq (X^* \times Q \times X^*) \times (X^* \times Q \times X^*)$, съставена от наредени двойки конфигурации дефинираме така:

а) $(\alpha' b, q, a\alpha'') \vdash (\alpha', q', ba'\alpha'')$, ако $\delta(q, a) = (q', a', L)$;

б) $(\alpha', q, a\alpha'') \vdash (\alpha' a', q', \alpha'')$, ако $\delta(q, a) = (q', a', R)$;

в) $(\alpha', q, a\alpha'') \vdash (\alpha', q', a'\alpha'')$, ако $\delta(q, a) = (q', a', S)$.

С други думи, ако $\delta(q, a) = (q', a', m)$, то в края на такта машината на Тюринг записва $a'$ на мястото на $a$, преминава в състояние $q'$ и премества




Фигура 7.2: Конфигурация на машина на Тюринг.

главата на една клетка вляво, на една клетка вдясно или я оставя на място, ако $m = L, R$ или $S$, съответно. Ако $\kappa_1$ и $\kappa_2$ са две конфигурации на $M$ и $\kappa_1 \vdash \kappa_2$ казваме, че $\kappa_1$ *преминава непосредствено* в $\kappa_2$.

Нека $R_{\models} \subseteq (X^* \times Q \times X^*) \times (X^* \times Q \times X^*)$ е рефлексивно и транзитивно затваряне на $R_{\vdash}$. Ако $\kappa' \models \kappa''$ казваме, че $\kappa'$ *преминава* в $\kappa''$. Последователността от непосредствени преминавания $\kappa' \vdash \kappa_1 \vdash \kappa_2 \vdash \ldots \vdash \kappa_k \vdash \kappa''$ наричаме *изчисление* с машината на Тюринг $M$.

Естествен начин за завършване на едно изчисление на машината на Тюринг $M$ е тя да достигне някое от заключителните състояния. В такъв случай казваме, че $M$ *спира*. Машината може да спре и когато се намира в конфигурация $(\alpha, q, a\alpha')$ и стойността $\delta(q, a)$ не е дефинирана, или пък ако в конфигурация $(\varepsilon, q, a\alpha)$ имаме $\delta(q, a) = (q', a', L)$, т.е. машината се опитва да премести главата си вляво от най-лявата клетка на лентата.

Възможно е при зададена лентова дума $\alpha$ машината на Тюринг $M$ да не спре работа никога. Например, ако $\forall x \in X, \delta(q_0, x) = (q_0, x, R)$. В такъв случай казваме, че $M$ *зацикля*.

Възможни са различни видове изчисления с машини на Тюринг. Ще се спрем на някои от тях:

***Дефиниция.*** 1) Нека $M$ е МТ. Нека $\alpha$ е лентова дума, за която има спиращо изчисление $(\varepsilon, q_0, \alpha) \models (\beta', q, \beta'')$ и $\beta = \beta'\beta''$. Тогава функцията $f : X^* \to X^*$ такава, че

$$f(\alpha) = \begin{cases} \beta, & \text{ако } (\varepsilon, q_0, \alpha) \models (\beta', q, \beta'') \\ \text{неопределена}, & \text{ако М зацикли} \end{cases}$$

наричаме *изчислима по Тюринг*.

2) Нека $F = \{q_y, q_n\}$. Дефинираме език *разпознаван* от МТ $M$ като множеството от думи $L_M \subseteq X^*$ такова, че $\forall \alpha \in L_M, (\varepsilon, q_0, \alpha) \models (\alpha', q_y, \alpha'')$, а $\forall \beta \notin L_M, (\varepsilon, q_0, \beta) \models (\beta', q_n, \beta'')$.



3) Нека $F = \{q_y, q_n\}$. Дефинираме език *допускан* от МТ $M$ като множеството от думи $L_M \subseteq X^*$ такова, че $\forall \alpha \in L_M, (\varepsilon, q_0, \alpha) \models (\alpha', q_y, \alpha'')$, а $\forall \beta \notin L_M, (\varepsilon, q_0, \beta) \models (\beta', q_n, \beta'')$, $M$ спира по друг начин или зацикля.

За по-лесно представяне на МТ, ще дефинираме прост език, състоящ се от два оператора. Нека $M$ е МТ с входна азбука $X = \{x_1, x_2, \ldots, x_n\}$. Първият оператор е за представяне на незаключителни състояния и има вида:

$$q)\ \delta(q, x_1); \delta(q, x_2); \ldots; \delta(q, x_n).$$

Вторият оператор представя заключителни състояния и има вида:

$$q)\ stop.$$

Последователност от оператори, в която всяко $q_i \in Q \cup F$ се среща точно веднъж, наричама *програма* за МТ.

За простота ще разглеждаме азбуката с две букви $X = \{0, 1\}$, в която 0 играе ролята на бленк.

Пример 1. Програмата

$$\begin{array}{lllll} M_1: & q_0) & (q_1, 0, L) & ; & (q_0, 1, R) \\ & q_1) & - & ; & (q_1, 0, L) \end{array}$$

винаги спира при опит да премести главата си вляво от най-лявата клетка на лентата. Започвайки от лентовата дума $\alpha = 1^i 0\beta$, тя оставя на лентата думата $0^{i+1}\beta$, а произволна дума $\alpha = 0^i\beta$ оставя непроменена. Следователно машината $M_1$ изчислява функцията

$$f_{M_1}(\alpha) = \begin{cases} 0^{i+1}\beta, & \alpha = 1^i 0\beta \\ 0^i\beta, & \alpha = 0^i\beta \end{cases}$$

Пример 2. Програмата

$$\begin{array}{lllll} M_2: & q_0) & (q_1, 0, R) & ; & - \\ & q_1) & (q_2, 1, R) & ; & (q_1, 1, R) \\ & q_2) & (q_3, 0, L) & ; & (q_3, 0, L) \\ & q_3) & (q_4, 0, S) & ; & (q_3, 1, L) \\ & q_4) & stop & & \end{array}$$

винаги завършва в заключителното състояние $q_4$, ако лентовата ѝ дума започва с 0, и заради недефинираност, ако лентовата ѝ дума има за първа буква 1. В първия случай, започвайки от думата $01^i 0a\alpha$, машината

изчислява думата $01^{i+1}0\alpha$, т.е. добавя една единица в края на първия блок от единици. Във втория случай началната дума остава непроменена. Следователно машината $M_2$ изчислява функцията

$$f_{M_2}(\alpha) = \begin{cases} 01^{i+1}0\alpha, & \alpha = 01^i0a\alpha \\ \alpha, & \alpha = 1\alpha' \end{cases}$$

Пример 3. Програмата

$$\begin{array}{lllll} M_3: & q_0) & (q_2, 0, S) & ; & (q_1, 1, R) \\ & q_1) & (q_3, 0, S) & ; & (q_0, 1, R) \\ & q_2) & stop & & \\ & q_3) & stop & & \end{array}$$

винаги завършва в едно от състоянията $q_2$ или $q_3$, като в първия случай това става за думи от вида $1^{2k}0\beta, k = 0, 1, 2, \ldots$, а във втория случай – за останалите думи $(1^{2k+1}0\beta, k = 0, 1, 2, \ldots)$. Ако приемем $q_2 = q_y$ (допускащо заключително), а $q_3 = q_n$ (отхвърлящо заключително), можем да заключим, че $M_2$ разпознава езика $L_{M_2} = \{1^{2k}0\beta | k = 0, 1, 2, \ldots\}$. Ако разменим ролите на $q_2$ и $q_3$, ще получим нова машина $M_2'$, която разпознава допълнението $L_{M_2'}$ на $L_{M_2}$ до $X^*$, $L_{M_2'} = X^* \setminus L_{M_2}$. Последното ни довежда до сравнително проста конструкция (размяна на ролите на допускащото и отхвърлящото заключителни състояния), която доказва следната

**Теорема 7.3.1** *Ако $L \subseteq X^*$ се разпознава от МТ $M$, то и езикът $X^* \setminus L$ се разпознава от някоя МТ.*

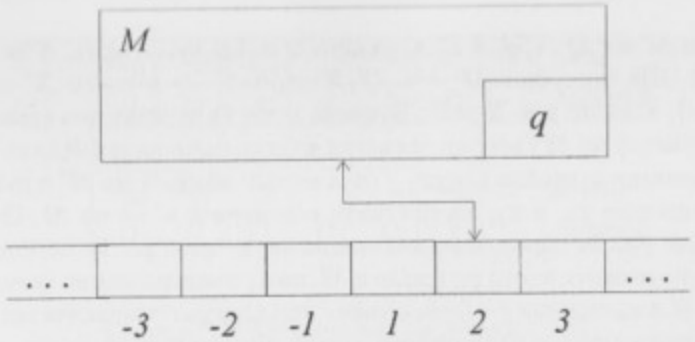
Забележете, че подобно твърдение не е в сила за езиците, допускани от МТ.

Известни са много опити за обобщаване на машините на Тюринг. Ще разгледаме накратко някои от тях.

**МТ с безкрайна в двете посоки лента ($\infty$–МТ)**

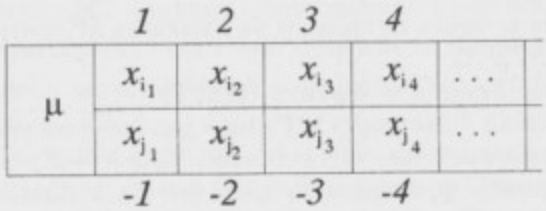
Формалната дефиниция на тази машина не се различава от дефиницията на МТ с безкрайна в едната посока лента. Във функционирането ѝ има следната разлика: отсъства спирането при опит за преместване на главата вляво от най-лявата клетка, тъй като при това обобщение няма най-лява клетка. И тук е в сила правилото за крайния брой букви различни от бленк, а за дефиниране на различните изчисления е достатъчно да си мислим, че при условната номерация на клетките (вж. Фиг. 7.3) началната лентова дума е винаги в клетките с номера $1, 2, \ldots, k$, а при






Фигура 7.3: $\infty$-Машина на Тюринг (с двустранно безкрайна лента).

завършване на работата крайната лентова дума остава на произволно място на лентата (в случая, когато изчисляваме функция), но главата сочи първата буква на думата резултат.




Фигура 7.4: Моделиране на $\infty$–МТ с МТ.

Ще изложим накратко идеята за доказателство на следната

**Теорема 7.3.2** *За всяка машина на Тюринг $M$ с безкрайна в двете посоки лента съществува машина на Тюринг $M'$, изчисляваща функцията $f_M$ (разпознаваща/допускаща езика $L_M$).*

***Доказателство.*** Ще построим машината $M'$, която моделира работата на $M$, без да доказваме формално идентичността на резултатите от изчисленията на $M$ и $M'$ при една и съща начална лентова дума. На Фиг. 7.4 е показана лентата на моделиращата машина $M'$. Тя е условно разделена на две ивици – горна, съответна на дясната част на лентата на $M$ и долна – съответна на лявата част.

Нека $M =< Q, X, q_0, \delta, F >$, където $Q = \{q_0, q_1, \ldots, q_m\}$, $X = \{x_1, x_2, \ldots, x_n\}$. Ще построим $M' =< Q', X', q'_0, \delta', F' >$ в която $X' = (X \times X) \cup \{\mu\}$, така че $\mu \notin X \times X$. Буквата $\mu$ ще бъде записана само в най-лявата клетка на $M'$, всички останали клетки съдържат букви от $X \times X$, като наредената двойка $(x_{i_k}, x_{j_k})$ от клетка с номер $k$ на $M'$ е съставена от стойностите $x_{i_k}$ и $x_{j_k}$ на клетките с номера $k$ и $-k$ на $M$. От всяко състояние $q$ на $M$ образуваме два екземпляра – $q_+$ и $q_-$. Те запазват в $M'$ предназначението, което $q$ е имало в $M$, но $q_+$ съответства на ситуациите, когато $M$ в състояние $q$ е била с глава над клетка с положителен номер, а $q_-$ – над клетка с отрицателен номер. Разбира се, $q'_0 = q_{0+}$, а $F' = F_+ \cup F_-$, където $F_+$ и $F_-$ са съответните версии на $F$. Сега функцията $\delta'$ дефинираме така:

$$\begin{aligned}
\delta(q_+, (x_i, x_j)) &= (q'_+, (x'_i, x_j), m), \text{ако } \delta(q, x_i) = (q', x'_i, m),\\
\delta(q_-, (x_i, x_j)) &= (q'_-, (x_i, x'_j), -m), \text{ако } \delta(q, x_j) = (q', x'_j, m),\\
\delta(q_+, \mu) &= (q_-, \mu, R),\\
\delta(q_-, \mu) &= (q_+, \mu, R).
\end{aligned}$$

Оставяме на читателя да провери, че машината $M'$ извършва същите изчисления като $M$. □

**Машина на Тюринг с $k$ ленти ($k$-МТ)**

В общия случай $k$-лентовата МТ има $k$ различни азбуки $X_1, X_2, \ldots, X_k$, всяка от които съдържа собствен бленк. Нека $X = X_1 \times X_2 \times \ldots \times X_k$. При това положение функцията на преходите на $k$-лентовата МТ е от вида $\delta : Q \times X \to (Q \cup F) \times X \times \{L, R, S\}^k$, т.е. всяка глава има свое самостоятелно поведение на всяка стъпка. За дефиниране на различните изчисления, използваме първата лента, като дефинициите са аналогични на тези при еднолентовата машина. Една трилентова машина е показана на Фиг. 7.8. За моделирането на работата на $k$-лентова МТ, се използва конструкция подобна на тази при МТ с двустранно безкрайна лента, разбира се доста по-сложна.

На Фиг. 7.5 е показан видът на лентата на моделиращата машина. За всяка от лентите върху нея са отделени по две подленти, първата е за записване на лентовата дума, а втората сочи мястото на съответната глава. Във всеки момент, мястото на главата е определено с 1 в точно една от клетките на втората подлента, а всички останали нейни клетки съдържат 0. Моделирането се извършва, като за всеки такт на $k$-лентовата машина еднолентовата повтаря следните действия: $k$ пъти обхожда лентата (по един път за всяка лента на $k$-лентовата МТ) и по мястото на главите, с помощта на специални състояния, събира $k$-торката от букви. След



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $x_{11}$ | $x_{12}$ | $x_{13}$ | $\cdots$ | $\cdots$ | $x_{1m}$ | $\cdots$ | *1-ва* |
| | *0* | *0* | *1* | *0* | *0* | *0* | $\cdots$ | *лента* |
| $\mu$ | | | | $\cdots$ | | | | |
| | $x_{k1}$ | $x_{k2}$ | $x_{k3}$ | $\cdots$ | $\cdots$ | $x_{km}$ | $\cdots$ | *k-та* |
| | *0* | *0* | *0* | *0* | *0* | *1* | $\cdots$ | *лента* |

Фигура 7.5: Моделиране на $k$–МТ с МТ.

като пресметне следващо състояние, $k$-торка от букви, които трябва да запише и движенията на главите, моделиращата машина извършва още $k$ обхождания на лентата, като при $i$-то обхождане записва съответната буква и премества съответната единица (мястото на главата) за $i$-тата лента на моделираната машина. Така е доказано, че изчисляваните с $k$-лентова МТ функции (разпознаваните/допусканите езици) съвпадат с изчисляваните от МТ функции (разпознаваните/допусканите езици).

**Недетерминирана $k$-лентова машина на Тюринг (НМТ)**

Недетерминираната $k$-лентова машина на Тюринг $M =< Q, X, q_0, \delta, F >$, при която $X = X_1 \times \cdots \times X_k$, а $\delta : Q \times X \to 2^{(Q \cup F) \times (X_1 \times \ldots \times X_k) \times \{L,R,S\}^k}$ има поведение, аналогично на КНА и НСА. Всяка начална лентова дума поражда (в общия случай безкрайно) дърво на поведението на НМТ. „Детерминизацията“ може да се извърши чрез обхождане (в ширина) на полученото дърво. Така функциите, изчислявани от $k$-лентова НМТ (разпознаваните/допусканите езици) също съвпадат с изчисляваните от МТ функции (разпознаваните/допусканите езици).

Доказано е, че машините на Тюринг разпознават езиците от общ тип, дефинирани в Глава 4. Доказателството, че за всеки език от общ тип съществува $k$-лентова недетерминирана МТ, се извършва по схема, подобна на тази в Теорема 4.33. Недетерминираната машина моделира всевъзможните изводи на граматиката, като използва допълнителна лента за прилагане на правилата от общ тип, при които няколко букви образуват лява част на правилата (сравнете с правилата на контекстно-свободните граматики). За да илюстрираме идеята за доказателство на твърдението в другата посока, ще посочим конструкция, показваща връзката между МТ и граматиките от общ тип. Нека $M =< Q, X, q_0, \delta, F >$ е машина на Тюринг. Нека състоянията от $Q$ са нетерминали, а входните букви от $X$ – терминали на граматика от общ тип Г с правила, определени от

функцията на преходите на $M$:

$$\begin{aligned}\mathcal{P} &= \{qa \to q'a' | \text{ ако } \delta(q,a) = (q',a',S)\} \cup \\ &\cup \{qa \to a'q' | \text{ ако } \delta(q,a) = (q',a',R)\} \cup \\ &\cup \{bqa \to q'ba' | \forall b \in X, \text{ ако } \delta(q,a) = (q',a',L)\}.\end{aligned}$$

При тези правила конфигурацията $(\alpha, q, \beta)$ на $M$ преминава в конфигурацията $(\alpha', q', \beta')$ т.с.т.к. $(\alpha, q, \beta) \overset{\Gamma}{\models} (\alpha', q', \beta')$, т.е. всяко изчисление на МТ може да бъде моделирано с извод в граматика от общ тип.

Правени са и други опити за разширяване и ограничаване възможностите на МТ за изчисление, които са зъвършили без успех. Както и в горните случаи, всяка нова конструкция се е оказала еквивалентна на класическата МТ. Например, доказано е, че опростяването на МТ до МТ с азбука $X = \{0, 1\}$ е несъществено, защото всяка МТ е еквивалентна по изчислителни възможности на МТ с две букви.

Освен с МТ и формални граматики, са правени и много други опити за формализиране на неформалното понятие алгоритъм. Между най-известните са: *частично-рекурсивните функции,* въведени от Ст. Клини; *$\lambda$-смятането,* въведено от А. Чърч; *машините на Пост,* въведени от Е. Пост, *нормалните алгоритми на Марков,* въведени от А.А. Марков и др. Доказана е еквивалентността на всички известни до днес формални модели, т.е. функциите, които тези модели естествено пораждат (или изчисляват), са едни и същи - изчислимите по Тюринг функции. Еквивалентността на формалните модели дава основание да бъде формулирано следното неформално твърдение:

***Тезис на Тюринг-Чърч:*** *Множеството на ефективно (алгоритмично) изчислимите функции съвпада с множеството на изчислимите по Тюринг функции.*

Това твърдение не може да бъде доказвано, заради използваното в него неформално понятие „ефективно (алгоритмично) изчислими функции". Затова можем само да приведем още един довод в негова полза. Най-естественият от практическа гледна точка формализъм за представяне на алгоритми е формално дефинираният език за програмиране. Ще покажем, че създаването на програми за машини на Тюринг не се отличава съществено от процеса на реалното програмиране.

Командата

$$q)\,(q', 0, S);\ (q', 1, S),$$

независимо от четената буква предизвиква преминаване в състояние $q'$.



Затова можем да я означим с

$$q)\ goto\ q'.$$

Аналогично, командата

$$q)\,(q', 0, S);\ (q'', 1, S)$$

можем да запишем като

$$q)\ if\ x = 0\ then\ goto\ q'\ else\ goto\ q''.$$

Следователно, МТ може да изпълнява управляващите конструкции на алгоритмичните езици.

В Пример 1 показахме, как можем да нулираме началото на лентата (т.е. да пресметнем аналог на функцията $f(x_1, x_2, \ldots) = 0$), в Пример 2 – как можем да увеличим с 1 дължината на блок от единици (т.е. да изчислим по Тюринг аналог на функцията $f(x) = x + 1$), а в Пример 3 – как може да се изчисли условие (в случая, четен или нечетен брой единици има в началото на лентата). За пресмятане на по-сложни функции се налага да се създават по-сложни МТ. По аналогия с използваните в програмирането техники, това може да става с композиране на готови вече машини (подпрограми) в нова машина (програма). В разглежданията по-долу, без ограничаване на общността, ще предполагаме, че състоянията на всяка МТ са избрани последователно от множеството $\{q_0, q_1, q_2, \ldots\}$ и $q_0$ винаги е началното състояние. Без ограничение на общността можем да считаме също, че изчисляващите функции МТ имат точно едно, а разпознаващите езици – точно две заключителни състояния.

***Дефиниция.*** Нека $M$ е машина на Тюринг, зададена с програмата си. *Редактиране от адрес $r$* на $M$ ще наричаме замяната на всяко състояние $q_i$ на $M$ със състояние $q_{i+r}$.

Нека $M_1$ и $M_2$ са МТ изчисляващи функции, зададени със съответните си програми и състоянието с най-голям номер в $M_1$ е $q_n$. Да редактираме $M_2$ от адрес $n+1$ и да заменим заключителното състояние $q_f)\ stop$ на $M_1$ с $q_f)\ goto\ q_{n+1}$. Получаваме нова машина на Тюринг $M$, която наричаме *последователна композиция* на $M_1$ и $M_2$. Очевидно, ако $M_1$ изчислява функцията $f_{M_1}$, а $M_2$ - функцията $f_{M_2}$, то $M$ изчислява композицията $f_{M_2} \circ f_{M_1}$ на тези две функции. Така доказахме следната

**Лема 7.3.1** *Ако $f_{M_1}$ и $f_{M_2}$ са изчислими по Тюринг функции, то и функцията $g = f_{M_2} \circ f_{M_1}$ е изчислима по Тюринг.*

Аналогично, можем да дефинираме и други по-сложни композиции на МТ, в подкрепа на твърдението, че създаването на програми на МТ е твърде близко до общоприетото понятие за програмиране. Средствата, с които се изграждат МТ са ограничени, което прави процеса на програмиране много тежък, но това не касае принципния въпрос: може или не може една функция да бъде изчислена с МТ.

След всичко казано досега, да приемем верността на Тезиса на Чърч и по-нататък в изложението под алгоритъм да разбираме машина на Тюринг.

## 7.4 Неразрешими масови задачи

Ще покажем, че съществуват масови задачи, които не могат да бъдат решени алгоритмично (т.е. с машина на Тюринг, а значи и с произволен друг есвивалентен изчислителен формализъм).

**Теорема 7.4.1** *Множеството от машините на Тюринг е изброимо.*

***Доказателство.*** Ще дефинираме функция, съпоставяща на всяка МТ дума от $\{0,1\}^*$ или, което е същото, естествено число. Нека $\mathcal{Q} = \{q_0, q_1, q_2, \ldots\}$ е изброимо множество от състояния и за дефинирането на всяка МТ да използваме първите няколко от тези състояния. Състоянието $q_0$ винаги ще е начално, а $q_1$ – заключително за машините, изчисляващи функции. Ако машината разпознава език, заключителните й състояния ще са $q_1 = q_y$ и $q_2 = q_n$. Аналогично, нека $\mathcal{X} = \{x_0, x_1, x_2, \ldots\}$ е изброимо множество от букви и за дефинирането на всяка МТ да използваме краен брой от тези букви. Буквата $x_0$ винаги ще е бленк на машината.

Кодираме състоянията, буквите и движенията на главата с кодираща функция

$$\kappa : \mathcal{Q} \cup \mathcal{X} \cup \{L, S, R\} \to \{1\}^+$$

по следния начин:

$\kappa(q_i) = 1^{i+1}$, $\forall q_i \in \mathcal{Q}$;

$\kappa(x_i) = 1^{i+1}$, $\forall x_i \in \mathcal{X}$;

$\kappa(L) = 1$, $\kappa(S) = 11$, $\kappa(R) = 111$.

Нека сега разгледаме машината на Тюринг $M =< Q, X, q_0, \delta, F >$, където $Q \subset \mathcal{Q}$, $X \subset \mathcal{X}$. Дефиницията на функцията на преходите представлява крайно множество $\{p_1, p_2, \ldots, p_r\}$, елементите на което са от вида $p_i = (q, x, q', x', m) \in Q \times X \times (Q \cup F) \times X \times \{L, S, R\}$ и $\delta(q, x) = (q', x', m)$. Дефинираме разширение на кодиращата функция с $\kappa(p_i) = \kappa(q)0\kappa(x)0\kappa(q')0\kappa(x')0\kappa(m)$. Машината $M$ кодираме с думата $\kappa(M) = \kappa(p_1)00\kappa(p_2)00\ldots00\kappa(p_r)$.



И така, за всяка машина на Тюринг имаме еднозначно определена дума от $\{0,1\}^*$, която елементарно се декодира до съответната й машина на Тюринг. Естествено, някои двоични думи не се декодират до смислена машина на Тюринг или въобще не се декодират, но важното е, че машините на Тюринг не са повече от думите на крайната азбука $\{0,1\}^*$, т.е. множеството им е изброимо. □

Процедурата, при която се установява еднозначно съответствие между елементите на зададено множество и естествените числа (думите над крайна азбука), с цел да се докаже неговата изброимост, наричаме *номериране* или *гьоделизация* (в чест на К. Гьодел, за големия му принос в теорията на изчислимостта).

Както вече добре знаем, езиците над крайна азбука не са изброимо много. Ако допуснем, че всеки език може да бъде разпознаван с машина на Тюринг, ще влезем в противоречие със заключението на Теорема 7.4.1. Така доказахме следната

**Теорема 7.4.2** *Съществува език над крайна азбука, който не се разпознава с машина на Тюринг.*

С други думи, съществуват масови задачи, които са неразрешими алгоритмично (с машина на Тюринг). Интересно е да се посочи конкретна такава масова задача.

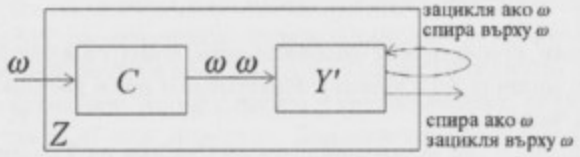
Нека $\alpha = x_{i_1}x_{i_2}\ldots x_{i_k}$ е лентова дума на машина на Тюринг $M$. С по-горе дефинираната функция кодираме $\alpha$ с $\kappa(\alpha) = \kappa(x_{i_1})0\kappa(x_{i_2})0\ldots0\kappa(x_{i_k})$ над $\{0,1\}^*$. Двойката $M$ и $\alpha$ кодираме с $\kappa(M)000\kappa(\alpha)$, но за по-кратко пишем просто $M\alpha$. Задачата:

„По зададени машина на Тюринг $M$ и лентова дума $\alpha$ да се определи дали $M$ спира (в заключително състояние или по недефинираност) или зацикля при работата си върху $\alpha$."

наричаме *стоп-проблем за машини на Тюринг*. Стоп-проблемът за МТ е задача за разпознаване на език, защото при произволен вход трябва да се даде отговор „да" (машината спира) или „не" (машината зацикля). Ще докажем, че стоп-проблемът не е разрешим алгоритмически.

**Теорема 7.4.3** *Не съществува машина на Тюринг, разпознаваща езика от всички думи $M\alpha$ такива, че машината $M$ спира при работа върху лентовата дума $\alpha$.*

***Доказателство.*** Да допуснем, че съществува машина на Тюринг $Y$, с азбука $\{0,1\}$, която разпознава зададения език, като $q_1$ е допускащото ѝ, а $q_2$ – отхвърлящото ѝ заключително състояние, т.е. $Y$ спира в $q_1$, ако $M$ спира при работа върху $\alpha$ и спира в $q_2$ в противен случай. Построяваме машината $Y'$, като в $Y$ заменим оператора $q_1)$ *stop* с оператора $q_1)$ *goto* $q_1$. Машината $Y'$ спира върху $M\alpha$, ако $M$ зацикля върху $\alpha$ и зацикля в противния случай. Нека $C$ е машина на Тюринг, която започвайки от лентова дума $\omega$ спира винаги и оставя на лентата $\omega\omega$, т.е. две копия на кода на $\omega$, разделени с 000. (Доказателството за съществуване на копираща машина е добро упражнение по програмиране на МТ.) Композираме последователно машините $C$ и $Y'$ и получаваме машина на Тюринг $Z$ (вж. Фиг. 7.6) със следното поведение: започва работа върху произволна лентова дума $\omega$ и спира, ако машината на Тюринг, кодирана с $\omega$ зацикля върху лентовата дума, кодирана с $\omega$ и зацикля в противен случай. Да поставим върху лентата на $Z$ собственото ѝ описание и да я оставим да го обработи. Ако $Z$ е била такава, че трябва да спре на собственото си описание, тя се зацикля. Ако пък е такава, че трябва да се зацикли – спира. Този абсурд е резултат на допускането, че съществува машината $X$. Следователно стоп-проблемът е неразрешим. □




Фигура 7.6: Неразрешимост на стоп-проблема за МТ.

Наблюдателният читател навярно е забелязал в горното доказателство по-важните елементи на познатия ни диагонален метод на Кантор.

Сега ще посочим друга мощна техника за доказателства на твърдения за формалните модели на понятието алгоритъм.

***Дефиниция.*** Функцията $f : A^* \to A^*$ наричаме *алгоритмически сводима* към $g : B^* \to B^*$, ако съществува изчислима по Тюринг биекция $\varphi : A^* \to B^*$ такава, че $\varphi^{-1}$ е също изчислима и $f(\alpha) = \varphi^{-1}(g(\varphi(\alpha)))$. За случая на разпознаване на език аналогичната дефиниция гласи: функцията $f : A^* \to \{true, false\}$ наричаме *алгоритмически сводима* към $g : B^* \to \{true, false\}$, ако съществува изчислима по Тюринг биекция $\varphi : A^* \to B^*$ такава, че $f(\alpha) = g(\varphi(\alpha))$.



С използване на понятието алгоритмична сводимост можем да докажем неразрешимост на масова задача $\pi_g$, като сведем алгоритмично към функцията ѝ $g$ функцията $f$ на известна неразрешима масова задача $\pi_f$. Наистина, ако допуснем че $\pi_g$ е разрешима, т.е. $g$ е изчислима по Тюринг, ще се окаже, че и $f = \varphi^{-1} \circ g \circ \varphi$ е изчислима по Тюринг, а това е противоречие с неразрешимостта на $\pi_f$.

Ще използваме алгоритмичната сводимост в един важен частен случай в следващия раздел.

## 7.5 Сложност на алгоритми и масови задачи

В класическата математика, след като е установена разрешимостта на една масова задача, интересът към нея значително спада. Наличието на алгоритъм прави решаването на кой да е екземпляр на задачата, повече или по-малко, рутинно. С развитието на изчислителната техника стана ежедневие да се решават огромни по размери екземпляри на алгоритмически разрешими задачи, които не са по силите на никой човек или даже на голяма група от хора. На фона на този значителен прогрес се открояват някои задачи с дискретен характер, решаването на които е затруднено. Те са алгоритмично разрешими, с помощта на схемата „пълно изчерпване“, която дава алгоритъм за решаване на всяка такава задача. Проблемът е, че когато расте обемът на входните данни, нараства твърде много времето, необходимо за изпълнение на алгоритъма, дори и на много мощни компютри. Например, броят на монотонните булеви функции на $n$ променливи е пресметнат за не много големи стойности на $n$.

Ще представим основите на математическа теория, занимаваща се с посочения проблем. От казаното по-горе е ясно колко важен за разглежданията е обемът на входните данни. Този въпрос е много сложен, затова тук ще поставим само едно важно условие, без което теоретичното изследване на проблема губи смисъл. Ще поискаме представянето на екземплярите с думи над избрана азбука да е *разумно*, т.е. всичко необходимо за решаването на съответната задача да се съдържа в думата, представяща екземпляра и да няма излишък от информация, такава, че ако я премахнем от думата, същността на екземпляра не се променя.

Основен ресурс, спрямо който ще изследваме качествата на алгоритмите, е времето за работа, но успоредно с това ще обърнем внимание и на количеството използвана памет. Първо ще разгледаме сложността на алгоритмите по отношение на необходимите им време и памет, а после ще се спрем на изключително важния въпрос за сложността на масовите

задачи.

***Дефиниция.*** Нека $f_M : A^* \to A^*$ е изчислима с машината на Тюринг $M$ тотална функция. Функцията

$$t_M(n) = \max_{\alpha \in A^*, d(\alpha)=n} [\text{брой стъпки на } M \text{ при работа върху } \alpha]$$

наричаме *сложност по време на $M$ в най-лошия случай*, а функцията

$$\tilde{t}_M(n) = \frac{\sum_{\alpha \in A^*, d(\alpha)=n} [\text{брой стъпки на } M \text{ при работа върху } \alpha]}{|A^n|}$$

наричаме *средна сложност по време на МТ $M$*.

Аналогично за сложността по памет имаме

***Дефиниция.*** Нека $f_M : A^* \to A^*$ е изчислима с машината на Тюринг $M$ тотална функция. Функцията

$$s_M(n) = \max_{\alpha \in A^*, d(\alpha)=n} [\text{брой използвани клетки от } M \text{ при работа върху } \alpha]$$

наричаме *сложност по памет на $M$ в най-лошия случай*, а функцията

$$\tilde{s}_M(n) = \frac{\sum_{\alpha \in A^*, d(\alpha)=n} [\text{брой използвани клетки от } M \text{ при работа върху } \alpha]}{|A^n|}$$

наричаме *средна сложност по памет на МТ $M$*.

За практически нужди е възможно да се дефинира по-привичен и ясен формализъм, като се изберат понятията „размер на входната дума" и „стъпка" така, че получените функции на сложност да съответстват на интуитивната ни представа за сложността на процедурата. Например, когато се подреждат по големина естествени числа, под стъпка се разбира сравнението на две такива числа. Това е естествено, защото броят на останалите стъпки лесно може да бъди оценен като функция от броя на сравненията. В същото време за размер на входните данни вместо дължината на думата, с която са представени всички числа, се взема броят им. И това също е разбираемо, защото ако числата не надхвърлят по големина определена граница, все едно е с колко цифри са представени във входната дума.

Като втори пример да разгледаме алгоритъма на Евклид за намиране на най-голям общ делител на две естествени числа. Ако подходим при определяне размера на входа както при задачата за подреждане, ще



получим, че всички екземпляри на тази масова задача имат размер на входните данни 2. Ако по този начин изчислим сложността по време, ще получим абсолютно неестествен резултат – стойност 0 за всички $n \neq 2$ и недефинирана, ако $n = 2$, защото всички екземпляри са с този размер и няма максимум на необходимия брой стъпки. Тук очевидно броят на цифрите, с които двете числа са представени, е от значение за намиране на добра функция на сложност.

И така за нуждите на практиката е наложително да се дефинират адекватно размер на входа и стъпка. Най-важното е, когато сравняваме сложността на два алгоритъма за една и съща задача, да прилагаме и към двата един и същ подход.

Между сложността по време и памет в най-лошия случай съществува следната важна връзка:

**Лема 7.5.1** *За всяка машина на Тюринг $M$ и за всяко естествено число $n \in N$, $s_M(n) \leq t_M(n)$.*

Твърдението е очевидно, защото не е възможно за $t_M(n)$ стъпки да бъдат използвани повече от $t_M(n)$ клетки на лентата.

В общия случай функциите на сложност са растящи функции, но с особено поведение. Затова вместо с реалните сложности обикновено работим с техни приближения, които са по-прости за изследване.

***Дефиниция.*** Нека $f : N \to R$ и $g : N \to R$. Казваме, че $f \in O(g)$, ако съществуват $n_0 \in N$ и $c \in R, c \geq 0$ такива, че $f(n) \leq cg(n)$, $\forall n \in N, n \geq n_0$.

Лесно се вижда, че $f \in O(f)$, и че от $f \in O(g), g \in O(h)$ следва $f \in O(h)$, т.е. релацията $\in O$ е рефлексивна и транзитивна, но не е симетрична, нито антисиметрична. Например, $n \in O(n^2)$, а $n^2 \notin O(n)$, докато $n^2 \in O(n^2 + 2n + 3)$ и $n^2 + 2n + 3 \in O(n^2)$.

**Лема 7.5.2** *$O(f) = O(g)$ т.с.т.к. $f \in O(g)$ и $g \in O(f)$.*

***Доказателство.*** 1) Нека $O(f) = O(g)$. Тъй като $f \in O(f)$ и $g \in O(g)$, то $f \in O(g)$ и $g \in O(f)$.

2) Нека $f \in O(g)$. Тогава $\forall h \in O(f)$ от транзитивността следва $h \in O(g)$, т.е. $O(f) \subseteq O(g)$. Другото включване се доказва аналогично. Следователно, $O(f) = O(g)$. □

Да означим с $N_R$ множеството $\{f | f : N \to R\}$. Релацията $R_\asymp \subseteq N_R \times N_R$ дефинираме с $R_\asymp = \{(f, g) | f \in O(g), g \in O(f)\}$. От доказаното по-горе следва, че $R_\asymp$ е релация на еквивалентност и разбива $N_R$ на

класове на еквивалентност. Класът на $f$ означаваме с $\Theta(f)$ и наричаме *порядък на растеж* на $f$. Очевидно $\Theta(f) \subseteq O(f)$. За представители на класовете избираме характерни функции – $n$, $n^2$, $\log n$, $2^n$ и т.н.

Нека $\mathcal{O}$ се състои от всички различни класове на еквивалентност на релацията $R_{\asymp}$. Релацията $R_{\preceq} \subseteq \mathcal{O} \times \mathcal{O}$ дефинираме с $R_{\preceq} = \{(\Theta(f), \Theta(g)) | f \in O(g)\}$. Ако $X \preceq Y$ и $X \neq Y$, тогава пишем $X \prec Y$. Оставяме на читателя да провери, че тази релация е частична наредба. Не е задължително, но е интересно да се покаже с пример, че тази наредба не е пълна. Наредбата $R_{\preceq}$ ни позволява да сравняваме порядъците на растеж на сложностите на различни алгоритми. За сортиране на елементи от някакво напълно наредено множество са популярни алгоритми от два типа – със сложност от $\Theta(n^2)$ и със сложност от $\Theta(n \log n)$. Тъй като $\Theta(n \log n) \prec \Theta(n^2)$, алгоритмите от втория вид са за предпочитане.

В сила е следната

**Лема 7.5.3** *За всяко цяло положително число $d$, $\Theta(n^d) \prec \Theta(2^n)$.*

Лемата ни дава основание да разграничим според сложността им по време две основни категории алгоритми. Да означим с $S$ множеството на всички разрешими масови задачи, като всяка задача се представя от съответната си изчислима функция.

***Дефиниция.*** Класът на *полиномиално-разрешимите задачи* $\mathcal{P}$ се състои от всички $f \in S$ такива, че съществува МТ $M$, която изчислява $f$ и $t_M(n) \in O(n^d)$, за някое $d \in N$.

В изложението имаме примери на алгоритми с полиномиална сложност: алгоритъмът на Прим, алгоритъмът на Крускал, алгоритъмът на Ойлер, алгоритъмът на Дейкстра, алгоритъмът за минимално съчетание в двуделен граф и др. Задачите, които те решават, са в класа $\mathcal{P}$. Имаме и множество примери на задачи, за които не е известен алгоритъм с полиномиална сложност. Например, задачите за намиране на Хамилтонов цикъл, за намиране на минимално оцветяване на граф, за минималните покрития и т.н. Ще разгледаме важен частен случай на алгоритмична сводимост на функции, определен от следната

***Дефиниция.*** Нека $f$ и $g$ са изчислими функции. Казваме, че $f$ се *свежда полиномиално* към $g$, ако $f$ се свежда алгоритмично към $g$ и свеждащите функции $\varphi$ и $\varphi^{-1}$ са алгоритми с полиномиална сложност. При свеждане на задачи за разпознаване на езици, разбира се, ще изискваме полиномиална сложност само на свеждащия алгоритъм $\varphi$.

Ще означаваме с $f \propto g$ полиномиалната сводимост на $f$ към $g$. Важността на релацията $\propto$ се вижда от следните свойства:



**Лема 7.5.4** *Ако $f \propto g$, то от $g \in \mathcal{P}$ следва $f \in \mathcal{P}$ (или еквивалентното от $f \notin \mathcal{P}$ следва $g \notin \mathcal{P}$).*

***Доказателство.*** Нека сложността, с която $f$ се свежда към $g$, и сложността, с която след изчисляване на $g$ полученият резултат се трансформира в стойност на $f$, са с порядъка на $p(n)$, за някакъв полином $p(n)$. Нека $g$ се изчислява със сложност от порядъка на $q(n)$, за някакъв полином $q(n)$. Тъй като дължината на резултата от свеждането не надхвърля сложността на пресмятането му, получаваме алгоритъм за предсмятане на $f$ със сложност $p(q(p(n)))$, т.е. $f$ е полиномиално-разрешима. □

**Лема 7.5.5** *Ако $f \propto g$ и $g \propto h$, то $f \propto h$.*

***Доказателство.*** Както в предната лема, свеждането на $f$ към $h$ става с композицията на два полиномиални алгоритъма в едната посока и на обратните им – в другата посока. Това довежда до суперпозиция на съответните полиноми и в резултат се получават полиномиални функции на сложност. □

Още веднъж ще отбележим, че когато се свеждат полиномиално задачи за разпознаване на езици нещата силно се опростяват, защото не се изисква конструиране на обратни на свеждащите алгоритми.

***Дефиниция.*** Масова задача, представена с изчислимата функция $f : A^* \to A^*$, наричаме *полиномиално-проверима*, ако съществува МТ $M$, разпознаваща полиномиално езика $\{\alpha\beta | f(\alpha) = \beta\}$. Думата $\beta$ наричаме *кандидат-решение*. Масова задача, представена с изчислимата функция $f : A^* \to \{true, false\}$, наричаме полиномиално-проверима, ако съществува $\beta_\alpha \in B^*$ и МТ $M$, разпознаваща полиномиално езика $\{\alpha\beta_\alpha\}$. Думата $\beta_\alpha$ наричаме *догадка*.

Означаваме с $\mathcal{NP}$ класа на полиномиално-проверимите задачи.

Полиномиалната проверимост на една задача се установява сравнително лесно. Да разгледаме задачата за намиране на Хамилтонов цикъл в зададен граф. Не е известен полиномиален алгоритъм за решаване на тази задача. Проверката, обаче, дали зададена последователност от върхове на граф образува Хамилтонов цикъл е елементарна. Достатъчно е да преминем последователността от началото към края и да проверим дали всеки два съседни върха образуват ребро (с матрица на съседства проверката се извършва с една операция), дали началото и краят съвпадат и дали всеки връх се среща точно веднъж. Последната проверка също се извършва лесно, ако се отбелязва в списък всеки срещнат връх. Така проверката ще отнеме време, пропорционално на броя на върховете в зададения граф.

Очевидно $\mathcal{P} \subseteq \mathcal{NP}$. За полиномиалната проверимост на всяка задача от $\mathcal{P}$ е достатъчно да я решим за полиномиално време и да сравним за полиномиално време получения резултат с кандидат-решението или думата догадка. Последното можем да направим, защото дължината на резултата винаги е ограничена от сложността по време, както вече отбелязахме. Генералният въпрос е *дали $\mathcal{P}$ е свщинско подмножество на $\mathcal{NP}$ или не*. Засега всяка една от двете хипотези е възможна, макар че от практически съображения по-вероятна изглежда първата възможност. Затова останалите разглеждания в този раздел ще направим при предположението, че $\mathcal{P} \neq \mathcal{NP}$.

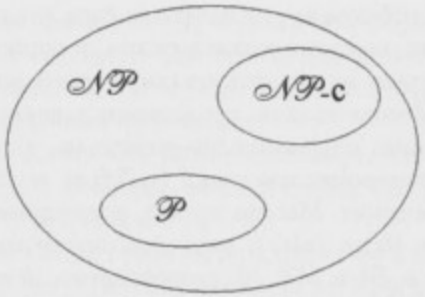
***Дефиниция.*** Задача $f$, за която:

а) $f \in \mathcal{NP}$;

б) $\forall g \in \mathcal{NP}, g \propto f$,

наричаме *$\mathcal{NP}$-пълна*. Означаваме с $\mathcal{NP}$-с класа на $\mathcal{NP}$-пълните задачи.

При предположението, че $\mathcal{P} \neq \mathcal{NP}$, взаимното разположение на $\mathcal{P}$, $\mathcal{NP}$ и $\mathcal{NP}$-с е показано на Фиг. 7.7.




Фигура 7.7: Класове на сложност.

**Лема 7.5.6** *Нека $f, g \in \mathcal{NP}$, $f \in \mathcal{NP}$-с и $f \propto g$. Тогава $g \in \mathcal{NP}$-с.*

***Доказателство.*** Тъй като $g \in \mathcal{NP}$, остава да докажем, че $\forall h \in \mathcal{NP}$, $h \propto g$. Но $\forall h \in \mathcal{NP}$, $h \propto f$, защото $f \in \mathcal{NP}$-с. От транзитивността на полиномиалната сводимост следва, че $h \propto g$, $\forall h \in \mathcal{NP}$. □

Ако имаме поне една $\mathcal{NP}$-пълна задача, тогава с полиномиално свеждане на тази задача към задачата $\pi$ можем да покажем, че и $\pi$ е $\mathcal{NP}$-пълна.

Да разгледаме следната задача: „Дадена е булева функция $f(x_1, x_2, \ldots, x_n)$ в конюнктивна нормална форма. Съществуват ли стойности на променливите $\sigma_1, \sigma_2, \ldots, \sigma_n$ такива, че $f(\sigma_1, \sigma_2, \ldots, \sigma_n) = 1$?." Ще я наричаме задача за *удовлетворимост на булева функция (УБФ)* и казваме, че векторът $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ *удовлетворява* $f$.



**Теорема 7.5.1 (С. Кук)** *Задачата УБФ е $\mathcal{NP}$-пълна.*

***Доказателство.*** 1) Лесно се проверява, че УБФ е от $\mathcal{NP}$. За да проверим, че даден вектор $\alpha$ удовлетворява функцията $f$ е достатъчно да заместим всяка променлива със съответната стойност от вектора. Ако всяка елементарна дизюнкция приема стойност 1 върху $\alpha$, тогава той удовлетворява $f$. Тази проверка изисква време пропорционално на сложността на КНФ, т.е. сложността ѝ е линейна функция на дължината на входа.

2) Остава да покажем, че всяка друга задача от $\mathcal{NP}$ се свежда полиномиално към УБФ. Нека $\pi$ е произволна задача от $\mathcal{NP}$, т.е. съществува машина на Тюринг $M =< Q, X, q_0, \delta, F >$, която за полиномиално време проверява дали някаква хипотеза е решение на задачата $\pi$. Нека $Q = \{q_0, q_1 = q_y, q_2 = q_n, \ldots, q_r\}$, $X = \{x_0 = \flat, x_1, \ldots, x_s\}$. Да означим с $L_M$ езика от всички входни думи, за които $M$ дава положителен отговор. Нека $\omega = x_{i_1}, x_{i_2}, \ldots, x_{i_n}$ е дума от азбуката на $M$. Нека $t_M(n) \in \Theta(p(n))$, където $p(n)$ е полиномиална функция. Без ограничение на общността ще считаме, че $t_M(n) = p(n)$. Ще посочим полиномиален алгоритъм, който от $M$ и входната дума $\omega$ (екземпляр на задачата $\pi$, заедно с кандидат-решението или думата догадка) образува булева функция $f$ в КНФ (екземпляр на задачата УБФ) такава, че $\omega \in L_M$ т.с.т.к. съществува вектор $\alpha$ от стойности на променливите на $f$ и $f(\alpha) = 1$. За време $p(n)$ машината на Тюринг може да промени само клетките с номера $1, 2, \ldots, p(n)$.

Ще използваме следните групи от променлиии за построяване на булевата функция:

- $Q[i,k], 1 \leq i \leq p(n), 0 \leq k \leq r$ – приема стойност единица т.с.т.к в момент $i$ машината е била в състояние $q_k$.

- $H[i,j], 1 \leq i \leq p(n), 1 \leq j \leq p(n)$ – приема стойност единица т.с.т.к в момент $i$ главата на машината е била над клетка с номер $j$.

- $S[i,j,l], 1 \leq i \leq p(n), 1 \leq j \leq p(n), 0 \leq l \leq s$ – приема стойност единица т.с.т.к в момент $i$ в клетката с номер $j$ се е намирала буквата $x_l$.

Ще построим шест групи от елементарни дизюнкции, които съставят исканата КНФ. Всяка група е описана с предназначението си, така че от конюнкцията на предназначенията да се вижда ясно, че това е функцията, която търсим.

1) Във всеки момент от време $M$ се намира в точно едно състояние.

Изискването $M$ да бъде в поне едно състояние, за всяко $i$, $1 \leq i \leq p(n)$ отразяваме в елементарната дизюнкция:

$$Q[i,0] \vee Q[i,1] \vee \ldots \vee Q[i,r],$$

а за да не бъде в повече от едно състояние, за всяко $i$, $1 \leq i \leq p(n)$ и за всяка двойка $k, k'$, $0 \leq k < k' \leq r$ създаваме елементарна дизюнкция

$$\overline{Q[i,k]} \vee \overline{Q[i,k']} = \overline{Q[i,k]Q[i,k']}.$$

Построяването на тези дизюнкции внася в конюктивната форма $p(n)(r+1) + 2p(n)\binom{r+1}{2} = p(n)(r+1)^2$ букви.

2) Във всеки момент от време главата на $M$ се намира точно над една клетка.

В тази група подобно на първата строим за всяко $i$, $1 \leq i \leq p(n)$ дизюнкциите

$$H[i,1] \vee H[i,2] \vee \ldots \vee H[i,p(n)]$$

и

$$\overline{H[i,j]} \vee \overline{H[i,j']}, 1 \leq j < j' \leq p(n).$$

Построяването на тези дизюнкции внася в конюктивната форма $p(n)^2 + 2p(n)\binom{p(n)}{2} = p(n)^3$ букви.

3) Във всеки момент от време, във всяка от клетките, които могат да бъдат достигнати от главата на $M$, се намира точно една буква.

Тази група строим както първите две, като за всяко $i$, $1 \leq i \leq p(n)$ и за всяко $j$, $1 \leq j \leq p(n)$ включваме дизюнкциите

$$S[i,j,0] \vee S[i,j,1] \vee \ldots \vee S[i,j,s]$$

и

$$\overline{S[i,j,l]} \vee \overline{S[i,j,l']}, 0 \leq l < l' \leq s.$$

Построяването на тези дизюнкции внася в конюктивната форма $p(n)^2(s+1) + 2p(n)^2\binom{s+1}{2} = p(n)^2(s+1)^2$ букви.

4) В началния момент машината $M$ е готова да започне работа върху думата $\omega$.

В началния момент $M$ се намира в състояние $q_0$, главата е над клетката с номер 1, в клетките с номера от 1 до $n$ е записана думата $\omega$, а клетките с номера от $n+1$ до $p(n)$ са запълнени с бленкове. Тази ситуация представяме със следните $p(n)+2$ еднобуквени дизюнкции

$$Q[0,0], H[0,1], S[0,1,i_1], \ldots, S[0,n,i_n], S[0,n+1,0], \ldots, S[0,p(n),0].$$



5) След не повече от $p(n)$ стъпки машината ще бъде в допускащото заключително състояние $q_y = q_1$

Тази група съдържа само една еднобуквена дизюнкция – $Q[p(n),1]$.

6) За всеки момент $i$ от времето, $0 \leq i \leq p(n)$, описанието на конфигурацията на $M$ за момента $i+1$ се получава от описанието за момента $i$ по правилата, определени от $\delta$.

Тази група е най-сложна. Разделяме я на две подгрупи от елементарни дизюнкции. В първата група са дизюнкциите, които ни гарантират, че не са възможни изменения, нерегламентирани с функцията на преходите. Т.е. не е възможно главата да не е над клетка $j$ в момент $i$, а в момент $i+1$ съдържанието на тази клетка да се е променило. За всяко $i, 1 \leq i \leq p(n)$, за всяко $j, 1 \leq j \leq p(n)$ и за всяко $l, 0 \leq l \leq s$ построяваме дизюнкцията

$$\overline{S[i,j,l]} \vee H[i,j] \vee S[i+1,j,l] = \overline{S[i,j,l]}\,\overline{H[i,j]}\,\overline{S[i+1,j,l]}.$$

От тази група в конюнктивната форма се добавят $3p(n)^2(s+1)$ букви.

Втората група се грижи за това, измененията на конфигурациите да се извършват по правилата, зададени от $\delta$. За всяка четворка $(i,j,k,l)$, $1 \leq i \leq p(n)$, $1 \leq j \leq p(n)$, $0 \leq k \leq r$ $l, 0 \leq l \leq s$ в тази подгрупа поставяме следните три елементарни дизюнкции:

$$\overline{H[i,j]} \vee \overline{Q[i,k]} \vee \overline{S[i,j,l]} \vee H[i+1, j+m],$$
$$\overline{H[i,j]} \vee \overline{Q[i,k]} \vee \overline{S[i,j,l]} \vee Q[i+1,k'],$$
$$\overline{H[i,j]} \vee \overline{Q[i,k]} \vee \overline{S[i,j,l]} \vee S[i+1,j,l'],$$

където $\delta(q_k, x_l) = (q_{k'}, x_{l'}, m)$, $m = -1, 0, 1$, ако $q \in Q$. Ако $q \in \{q_y, q_n\}$, тогава $k' = k, l' = l$ и $m = 0$.

От тази подгрупа в КНФ, която строим, ще влязат $12p(n)^2(r+1)(s+1)$ букви.

Да свържем с конюнкции всички построени дизюнкции. Получаваме конюнктивна нормална форма на булева функция $f$. Така, на всеки екземпляр $\omega$ на задачата, решавана от $M$ за полиномиално време, съпоставяме екземпляр на задачата УБФ – булева функция в КНФ. При това лесно се вижда, че $M$ завършва в $q_y$ при работа върху $\omega$ т.с.т.к. съществува вектор от стойности на променливите $\alpha$, за който $f(\alpha) = 1$.

Остава да покажем, че конструирането на $f$ става за време, ограничено от някакъв полином на $n = d(\omega)$. За целта да отбележим, че общият брой букви, които участват в $f$, е $p\rho^2 + p^3 + p^2\sigma^2 + p + 3 + 3p^2\sigma + 12p^2\rho\sigma =$

$p^3+(\sigma^2+3\sigma+12\rho\sigma)p^2+2p+3$, където с $p$, $\rho$ и $\sigma$ сме означили за по-кратко $p(n)$ и константите $r+1$ и $s+1$, съответно. Полученото е очевидно полином от $n$. Може да се посочи константа $c$ такава, че за построяването на всяка от буквите на КНФ (заедно със съседните ѝ знаци на операции) са необходими не повече от $c$ стъпки на алгоритъма. И така построеният свеждащ алгоритъм е полиномиален.

Следователно задачата УБФ е $\mathcal{NP}$-пълна. □

След като имаме $\mathcal{NP}$-пълната задача УБФ, с използването на Лема 7.5.6, можем да докажем $\mathcal{NP}$-пълнотата на друга задача от $\mathcal{NP}$, като сведем полиномиално УБФ към новата задача. Чрез такова свеждане е доказана принадлежността на стотици интересни за теоретичната и приложна информатика задачи към класа $\mathcal{NP}$-с. От познатите ни задачи в този клас попадат: задачата за намиране на Хамилтонов цикъл, за минимално оцветяване на върховете на граф, за максималната клика (или антиклика) на граф, задачата за минималните покрития, задачата за минимизация на ДНФ и т.н.

Голямото предизвикателство се състои в това, че всички тези задачи са полиномиално сводими една към друга, т.е. ако за една от тях бъде намерен полиномиален алгоритъм, такъв алгоритъм има за всяка друга задача от класа. Ако за една от тези задачи се докаже, че не съществува полиномиален алгоритъм, за никоя задача от класа няма такъв алгоритъм.

## 7.6 Универсална машина на Тюринг

Ще използваме означенията, въведени в 7.3.

***Дефиниция.*** Ще казваме, че машината на Тюринг $U$ е *универсална*, ако за всяка машина $M$ и лентова дума $\alpha$, $U$ работи върху $M\alpha$ така както $M$ върху $\alpha$, т.е. $U$ спира върху $M\alpha$, когато $M$ спира върху $\alpha$ и зацикля върху $M\alpha$, когато $M$ зацикля върху $\alpha$. При това, ако $M$ спира в заключително състояние при работата върху $\alpha$ и върху лентата ѝ остава думата $\beta$, то след спирането на $U$ върху $M\alpha$ на определено място на лентата ѝ също се намира думата $\beta$.

Доколкото машините на Тюринг са формален модел на понятието алгоритъм, от изключителен методологичен интерес е въпросът: „Съществува ли универсална машина на Тюринг?“ Наличието на универсален алгоритъм би предопределило възможността да се създаде реално изчислително устройство, реализиращо този алгоритъм и способно да изпълни работата на произволен друг алгоритъм, при произволни негови

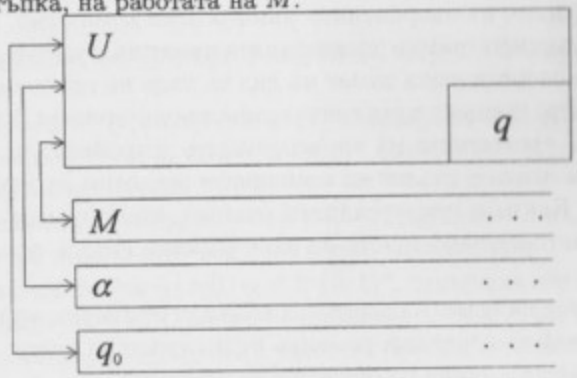


входни данни, при това да получи същия резултат. Отсъствието на универсален алгоритъм би означавало, че за нуждите на автоматизираното изпълнение на всеки отделен алгоритъм би трябвало да се построява специфично устройство, което го реализира.

Доказателството на следната

**Теорема 7.6.1** *Съществува универсална машина на Тюринг.*

е доста дълго, затова ще изложим накратко неговата идея. Да разгледаме трилентова МТ $U$ (вж. Фиг. 7.8). На първата лента ще поставим кодираното описание $\kappa(M)$ на зададената машина на Тюринг, а на втората лента – кодираното описание $\kappa(\alpha)$ на зададената лентова дума. По време на работата втората лента ще се променя така, както би се променяла лентата на $M$ при работа върху $\alpha$ и върху нея ще остане резултатът, получен при нормално завършващо изчисление. В началото главата на втората лента се намира над най-лявата буква. Третата лента е работна и отначало върху нея е записан кодът $\kappa(q_0)$ на началното състояние на $M$. Състоянието, записано в началото на трета лента, ще наричаме текущо. Работата на универсалната машина се състои в буквално повтаряне, стъпка по стъпка, на работата на $M$.




Фигура 7.8: Универсална машина на Тюринг.

За всяка стъпка на $M$ машината $U$ извършва едни и същи действия. От втората лента се чете кодът на текущата входна буква $x$ и се записва до текущото състояние $q$ на третата лента. Търси се върху първата лента (след комбинация от две нули) описание на петорка, започваща с $q, x$. Остатъкът от петорката, т.е. $(q', x', m), \delta(q, x) = (q', x', m)$ се извличат от първата на третата лента. След това съдържанието на втората лента се

модифицира с $x'$, главата на втората лента се мести в съответствие с $m$, а $q'$ се копира в началото на третата лента. С това е моделирана една стъпка от работата на $M$ и работата на $U$ продължава с моделиране на следващата стъпка.

Ако машината $M$ достига до заключително състояние при работа върху $\alpha$, универсалната машина преминава в съответното заключително състояние и също преустановява работа. Ако $M$ се зацикли при работата над $\alpha$, то и $U$ ще направи същото, моделирайки работата на $M$. Ако $M$ спре при опит да премести главата си наляво от най-лявата клетка, то и $U$ ще спре при опита да премести главата на втората лента наляво от най-лявата ѝ клетка. Ако $M$ спира заради недефинираност на функцията на преходите, тогава $U$ лесно ще установи това при претърсване на описанието на $M$ и ще преустанови работа в „аварийно заключително състояние", различно от заключителните състояния на $M$.

За да завърши конструкцията на теоремата, остава да трансформираме 3-лентовата машина в еквивалентна еднолентова по начина, който посочихме по-горе.

Не е трудно да се направи аналогия между конструкцията и функционирането на универсалната машина на Тюринг и конструкцията и функционирането на съвременния универсален компютър. Първите две ленти на $U$ съответстват на оперативната памет на компютъра, като разделянето на оперативната памет на дял за кода на програмата и дял за данните е често срещано в реалните компютърни системи. Управляващата част на $U$ съответства на управляващото устройство на компютъра, като третата лента е аналог на помощните регистри на управляващото устройство. Както и универсалната машина, компютърът на практика изпълнява автоматично тялото на един работен цикъл, основните стъпки на който са:

- извличане на адрес на следваща команда от текущо изпълняваната;
- извличане на следваща команда от програмния текст;
- извличане на данни от областта на данните;
- изпълняване на командата, с последващо връщане на редултата в областта за данните и т.н.

Доказателството за съществуване на УМТ е математическата обосновка на принципната възможност да се създаде реално универсално устройство за автоматично изпълнение на алгоритми. Сега, когато такова устройство е широко разпространено, тази обосновка няма същата тежест, но в годините на работа върху първите образци на универсални компютри въпросът за съществуването на УМТ е бил от първостепенно значение.



Не е трудно да се посочат съществени различия между абстрактните и реалните изчислители. На първо място това е безкрайната памет на УМТ, която никога няма да имаме в компютъра, колкото и да разширяваме обема на запомнящите устройства. Този недостатък се компенсира с много по-бързия достъп до данните в паметта на съвременните компютри – *произволен достъп*, за разлика от машините на Тюринг, при които за да се прочете буква от лентата е необходимо да се премине през всички предшестващи съответната клетка – *последователен достъп*. Не е за пренебрегване и силно развитата система от специфични операции, която съвременните компютри предлагат като алтернатива на простата (но универсална) операция на машините на Тюринг – смяна на едно състояние с друго.

---

## Упражнения

### Упражнение 7.1

Постройте машина на Тюринг с азбука $\{\flat, 0, 1\}$ и запълваща буква $\flat$, която:

а) започвайки от $(\flat, q_0, \alpha)$, $\alpha \in \{0,1\}^*$, завършва в $(\flat, q, \alpha\flat\alpha)$;

б) започвайки от $(\flat\alpha, q_0, \flat\beta)$, $\alpha, \beta \in \{1\}^*$, завършва в $q_y$, ако $d(\alpha) | d(\beta)$ и в $q_n$ – в противен случай;

в) започвайки от $(\flat\alpha, q_0, \flat\beta)$, $\alpha, \beta \in \{1\}^*$, завършва в $q_=$, ако $d(\alpha) = d(\beta)$, в $q_<$ – ако $d(\alpha) < d(\beta)$ и в $q_>$ – в противен случай;

г) започвайки от $(\flat\alpha, q_0, \flat\beta)$, $\alpha, \beta \in \{0,1\}^*$, завършва в $q_=$, ако $\nu(\alpha) = \nu(\beta)$, в $q_<$ – ако $\nu(\alpha) < \nu(\beta)$ и в $q_>$ – в противен случай (с $\nu(x)$ означаваме естественото число, представено в двоична бройна система от $x$);

д) започвайки от $(\flat, q_0, \alpha)$, $\alpha \in \{0,1\}^*$, завършва в $(\flat, q_0, \beta)$ така, че $\nu(\beta) = \nu(\alpha) + 1 (\mathrm{mod}\ 2^{d(\alpha)})$.

### Упражнение 7.2

Постройте машина на Тюринг, която разпознава езика:

а) $L_1 = \{a^i b^i c^i | i = 1, 2, \ldots\}$;

б) $L_2$, разпознаван от КДА $A =< Q, X, q_0, \delta, F >$;

в) $L_3$, разпознаван от НСА $A =< Q, X, Z, q_0, z_0, \delta, F >$ със заключителни състояния;

г) $L_4$, състоящ се от всички думи на $\{0,1\}$, които са палиндроми.

**Упражнение 7.3**

Нека $L_1$ и $L_2$ са езици разпознавани от машини на Тюринг. Докажете, че езиците:

а) $L_1 \cup L_2$; б) $L_1 \cap L_2$; в) $L_1 L_2$.

се разпознават от машини на Тюринг.

**Упражнение 7.4**

Постройте машина на Тюринг $M$ с азбука $\{0,1,\flat\}$ и двустранно безкрайна лента, със следното поведение. В началото лентата ѝ съдържа 0 във всичките си клетки, с изключение на една, в която е буквата 1, а главата на $M$ е върху произволна клетка на лентата. При това начално състояние, $M$ винаги спира и главата ѝ е върху клетката, в която в началото е имало единица.

**Упражнение 7.5**

Да означим с $\mathcal{M}^i, i = 1,2,\ldots$ множеството от всички машини на Тюринг на азбуката $\{0,1\}$, с 1 заключително и $i$ незаключителни състояния.

а) постройте различните машини от $\mathcal{M}^1$;

б) колко машини от $\mathcal{M}^1$ спират и колко зациклят при входна лента, запълнена изцяло с нули?

**Упражнение 7.6**

Постройте краен детерминиран автомат, който разпознава правилно кодирани (в смисъла на Теорема 7.4.1) машини на Тюринг.

**Упражнение 7.7**

Докажете неразрешимостта на задачата: „По зададена машина на Тюринг $M$ да се определи дали $M$ спира или цикли, върху лента запълнена изцяло с бленкове".

**Упражнение 7.8**

Функцията $f$ дефинираме за всяко $n \in \{1,2,3,\ldots\}$ и $f(n)$ е максималния брой единици, които спираща машина от $\mathcal{M}^i$ (вж. Упражнение 7.5)



оставя след спирането, започвайки от лента запълнена само с нули. Докажете, че $f$ не е изчислима по Тюринг.

**Упражнение 7.9**

Вирус наричаме машина на Тюринг $V$, която по зададена машина на Тюринг $M$ построява последователната композиция $M \circ V$. Докажете, че не съществува машина, която разпознава всички вируси.

**Упражнение 7.10**

Кои от следните твърдения са верни:

а) $n^2 \in O(n^3)$; б) $n^3 \in O(n^2)$; в) $2^{n+1} \in O(2^n)$; г) $(n+1)! \in O(n!)$;

д) $\forall f : N \to R^+, f(n) \in O(n) \Rightarrow f^2(n) \in O(n^2)$;

е) $\forall f : N \to R^+, f(n) \in O(n) \Rightarrow 2^{f(n)} \in O(2^n)$.

**Упражнение 7.11**

Посочете функции $f, g : N \to R^+$ такива, че $f \notin O(g)$ и $g \notin O(f)$.

**Упражнение 7.12**

Оценете порядъка на растеж на броя на сравненията (в най-лошия случай) за:

а) алгоритъма за последователно търсене на число $x$ в масив с $n$ елемента;

б) алгоритъма за двоично търсене на число $x$ в сортиран масив с $n$ елемента;

в) алгоритъма за подреждане на $n$ числа по „метода на мехурчето";

г) алгоритъма за сливане в подреден масив с $n$ числа на два подредени масива с $m_1$ и $m_2$ числа $(m_1 + m_2 = n)$.

**Упражнение 7.13**

Оценете порядъка на растеж на средния брой сравнения за:

а) алгоритъма за последователно търсене на число $x$ в масив с $n$ елемента;

б) алгоритъма за двоично търсене на число $x$ в сортиран масив с $n$ елемента.

**Упражнение 7.14**

Множествата $A_1$ и $A_2$, с $m_1$ и $m_2$ елемента, съответно, са представени като масиви от различни цели числа. Предложете „бързи“ (по брой сравнения) алгоритми за извършване на операциите обединение, сечение и симетрична разлика на $A_1$ и $A_2$.

**Упражнение 7.15**

Изберете подходящо представяне за свързания граф $G(V, E)$ с тегла на ребрата определени от $c : E \to N$ и оценете броя на сравненията (в най-лошия случай), които са необходими на:

а) алгоритъма на Прим;

б) алгоритъма на Крускал;

в) алгоритъма на Дейкстра.

**Упражнение 7.16**

За рекурсивните алгоритми за сортиране на $n$ числа „сортиране чрез сливане“ и „бързо сортиране“ напишете рекурентни отношения за броя на необходимите сравнения (съответни на рекурсивните извиквания на задачи с по-малък размер). Оценете сложността на тези алгоритми.

**Упражнение 7.17**

Нека $G(V, E)$ е граф. Множеството $V' \subseteq V$ наричаме *върхово покритие* на $G$, ако $\forall (v_i, v_j) \in E$ или $v_i \in V'$, или $v_j \in V'$. Докажете, че задачите:

а) за намиране на максимална клика на $G$;

а) за намиране на максимална антиклика на $G$;

в) за намиране на минимално върхово покритие на $G$;

се свеждат полиномиално една към друга.

**Упражнение 7.18**

Докажете, че ако $\pi$ е полиномиално-проверима задача, то $\pi$ е разрешима за време от порядъка на $2^{p(n)}$, за някакъв полином $p(n)$ на дължината на входа $n$.

## Библиография


[1] М. Айгнер. *Комбинаторная теория.* Мир, Москва, 1982.

[2] А. Акритас. *Основы компютърной алгебры с приложениями.* Мир, Москва, 1994.

[3] А. Ахо, Дж. Хопкрофт и Дж. Ульман. *Построение и анализ вычислительных алгоритмов.* Мир, Москва, 1979.

[4] Дж. Барвайс, под ред. *Справочная книга по математической логике, ч.II Теория множеств.* Наука, Москва, 1982.

[5] Э.Р. Берлекэмп. *Алгебраическая теория кодирования.* Мир, Москва, 1971.

[6] Д.А. Владимиров. *Булевы алгебры.* Наука, Москва, 1969.

[7] Г.П. Гаврилов и А.А. Сапоженко. *Сборник задач по дискретной математике.* Наука, Москва, 1977.

[8] М. Гаврилов и В. Чуканов. *Основни алгебрични структури.* Наука и изкуство, София, 1973.

[9] Р. Галлагер. *Теория информации и надежная связь.* Советское радио, Москва, 1974.

[10] Г. Гретцер. *Общая теория решеток.* Мир, Москва, 1982.

[11] Д. Грин и Д. Кнут. *Математические методы анализа алгоритмов.* Мир, Москва, 1987.

[12] М. Гэри и Д. Джонсон. *Вычислительные машины и труднорешаемые задачи.* Мир, Москва, 1982.

[13] Й. Денев, Р. Павлов и Я. Деметрович. *Дискретна математика.* Наука и изкуство, София, 1984.

[14] Ф. Картеси. *Введение в конечные геометрии.* Наука, Москва, 1979.

[15] Н. Кристофидес. *Теория графов. Алгоритмический подход.* Мир, Москва, 1978.

[16] О.П. Кузнецов и Г.М. Адельсон-Вельский. *Дискретная математика для инженера.* Энергоатомиздат, Москва, 1988.

[17] Ф.Дж. Мак-Вилямс и Н.Дж.А. Слоэн. *Теория кодов, исправляющих ошибок.* Связь, Москва, 1979.

[18] З. Манна. *Математическа теория на информатиката.* Наука и изкуство, София, 1983.

[19] М. Минский. *Вычисления и автоматы.* Мир, Москва, 1971.

[20] В.Н. Нефедов и В.А. Осипова. *Курс дискретной математики.* Издателство МАИ, Москва, 1992.

[21] Р. Павлов. *Математическа лингвистика.* Народна просвета, София, 1982.

[22] Д. Пойа. *Математическото откритие.* Народна просвета, София, 1968.

[23] Э. Рейнгольд, Ю. Нивергельт и Н. Део. *Комбинаторные алгоритмы. Теория и практика.* Мир, Москва, 1980.

[24] В.Дж. Рейуорд-Смит. *Теория формальных языков. Вводный курс.* Радио и связь, Москва, 1988.

[25] Дж. Риордан. *Комбинаторные тождества.* Наука, Москва, 1982.

[26] Рыбников, К.А., под ред. *Комбинаторный анализ. Задачи и упражнения.* Наука, Москва, 1982.

[27] В. Славов. *Увод в алгоритмите.* Издателство на Нов Български Университет, София, 1995.

[28] Ф. Харари. *Теория графов.* Мир, Москва, 1979.

[29] М. Холл. *Комбинаторика.* Мир, Москва, 1970.

[30] Р.В. Хэмминг. *Теория кодирования и теория информации.* Радио и связь, Москва, 1983.



[31] В. Чуканов. *Комбинаторика.* Народна просвета, София, 1977.

[32] С.В. Яблонский. *Введение в дискретную математику.* Наука, Москва, 1974.

[33] A.V. Aho, J.E. Hopcroft and J.D. Ullman. *Data Structures and Algorithms.* Addison-Wesley, Reading MA, 1983.

[34] G. Brassard and P. Bratly. *ALGORITHMIC. Theory and Practice.* Printice Hall, 1988.

[35] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading MA, 1979.

[36] B. Korte, L. Lovasz and R. Schrader. *Greedoids.* Springer-Verlag, 1991.

[37] H.R. Lewis and Ch.H. Papadimitriou. *Elements of the Theory of Computation.* Prentice-Hall, 1981.

[38] M. Nelson. *The Data Compression Book.* M&T Books, 1991.

[39] D.J.A. Welsh. *Matroid Theory.* Academic Press, New York, 1976.

## Списък на фигурите

## Списък на таблиците

## Азбучен указател