Записки по дискретна математика

Христо Ганчев

17 януари 2023 г.

Съдържание

ГЛАВА 1

Множества и функции

1.1 Съждителна логика

Съждителните формули са формални думи (последователности от символи), изградени от символите $P_0, P_1, \ldots, P_n, \ldots$ (които наричаме съждителни променливи), $\neg, \vee, \&, \to, \leftrightarrow$ (които наричаме логически връзки) и $(,)$ (лява и дясна скоба), използвайки следните правила:

1. Всяка съждителна променлива е съждителна формула.
2. Ако $\mathbf{A}$ и $\mathbf{B}$ са съждителни формули, то и последователностите $\neg\mathbf{A}$, $(\mathbf{A} \vee \mathbf{B})$, $(\mathbf{A} \,\&\, \mathbf{B})$, $(\mathbf{A} \to \mathbf{B})$, $(\mathbf{A} \leftrightarrow \mathbf{B})$ също са съждителни формули.

Съгласно първото правило P_0 и P_1 са съждителни формули. Използвайки последователно второто правило, можема да получим следните съждителни формули:

$$\neg P_0,\ (P_0 \vee P_1),\ (P_0 \,\&\, (P_0 \vee P_1)),\ \neg(P_0 \,\&\, (P_0 \vee P_1),\ (\neg(P_0 \,\&\, (P_0 \vee P_1) \to P_1).$$

С оглед по-кратък и читаем запис на формулите ще използваме следните съкращения: изпускаме най-външните скоби на формулата; връзките $\vee$ и $\&$ свързват по-силно от връзките $\to$ и $\leftrightarrow$; при многократно използване на една и съща връзка групирането е от дясно наляво. По този начин формулата

$$P_0 \,\&\, P_1 \,\&\, P_2 \to P_0 \vee P_1 \vee P_2$$

е съкращение на формулата

$$((P_0 \,\&\, (P_1 \,\&\, P_2)) \to (P_0 \vee (P_1 \vee P_2)),$$

а формулата

$$\neg P_0 \to P_0 \vee P_1 \to P_1$$

е съкращение за формулата

$$(\neg P_0 \to ((P_0 \vee P_1) \to P_1)).$$

Интуитивно, съждителните променливи са заместители на елементарни съждения, които са или истина, или лъжа. Логическата връзка $\neg$ играе ролята на отрицание и обръща истинността на формулата, пред която стои. Логическата връзка $\vee$ се нарича „логически или“ (дизюнкция) и изказва това, че е истина поне една от формулите, които свързва. Логическата връзка $\&$ се нарича „логически и“ (конюнкция) и изказва това, че са истина и двете формули, които свързва. Логическата връзка $\to$ се нарича импликация и изказва това, че в случай, че формулата отляво е истина, то вярна е и формулата отдясно е истина. Логическата връзка $\leftrightarrow$ се нарича еквивалентност и изказва това, че двете формули, които свързва, имат една и съща истинност. Накрая нека да отбележим, че формулата

$$P_0 \to P_1 \to P_2$$

е съкращения за

$$(P_0 \to (P_1 \to P_2)),$$

чийто смисъл е: ако P_1 е истина, то и ако P_1 е истина, то и P_2 е истина, или по кратко, ако P_1 и P_2 са истина, то и P_2 е истина.

Използвайки горната интуиция, смисълът на формулата $P_0 \to P_0 \vee P_1$ е: ако е съждението P_0 е истина, то поне едно от съжденията P_0 и P_1 е истина. Смисълът на формулата $P_0 \,\&\, P_1 \to P_1$ е: ако P_0 и P_1 са едновременно истина, то P_2 е истина.

Горната интуиция формализираме по следния начин. Оценка на променливите наричаме всяко изображение, съпоставящо на съждителните променливи $\mathbb{T}$ или $\mathbb{F}$. Ако V е оценка на променливите, то тя поражда оценка $\tilde{V}$ на всички съждителни формули по следната схема:

1. $\tilde{V}(P_i) = V(P_i)$ за $i = 0, 1, \ldots,$;

2. $\tilde{V}(\neg\mathbf{A}) = \begin{cases} \mathbb{T}, & \tilde{V}(\mathbf{A}) = \mathbb{F} \\ \mathbb{F}, & \tilde{V}(\mathbf{A}) = \mathbb{T} \end{cases}$, $\tilde{V}(\mathbf{A} \vee \mathbf{B}) = \begin{cases} \mathbb{F}, & \tilde{V}(\mathbf{A}) = \tilde{V}(\mathbf{B}) = \mathbb{F} \\ \mathbb{T}, & \text{иначе} \end{cases}$, $\tilde{V}(\mathbf{A} \,\&\, \mathbf{B}) = \begin{cases} \mathbb{T}, & \tilde{V}(\mathbf{A}) = \tilde{V}(\mathbf{B}) = \mathbb{T} \\ \mathbb{F}, & \text{иначе} \end{cases}$,

$\tilde{V}(\mathbf{A} \to \mathbf{B}) = \begin{cases} \mathbb{F}, & \tilde{V}(\mathbf{A}) = \mathbb{T},\ \tilde{V}(\mathbf{B}) = \mathbb{F} \\ \mathbb{T}, & \text{иначе} \end{cases}$, $\tilde{V}(\mathbf{A} \leftrightarrow \mathbf{B}) = \begin{cases} \mathbb{T}, & \tilde{V}(\mathbf{A}) = \tilde{V}(\mathbf{B}) \\ \mathbb{F}, & \text{иначе} \end{cases}$.

Например, при оценка на променливите V, за която $V(P_0) = V(P_1) = \mathbb{T}$, имаме

$$\tilde{V}(P_0 \to P_0 \,\&\, P_1) = \mathbb{T},$$

а при оценка V', за която $V(P_0) = \mathbb{T}$ и $V(P_1) = \mathbb{F}$, имаме

$$\tilde{V}'(P_0 \to P_0 \,\&\, P_1) = \mathbb{F}.$$

Можем да обобщим значенията на логическите връзки в следната таблица

$\mathbf{A}$	$\mathbf{B}$	$\neg\mathbf{A}$	$\mathbf{A} \vee \mathbf{B}$	$\mathbf{A} \,\&\, \mathbf{B}$	$\mathbf{A} \to \mathbf{B}$	$\mathbf{A} \leftrightarrow \mathbf{B}$
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$
$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{F}$
$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$
$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$

Стойностите на формулата $P_0 \to \neg(P_0 \,\&\, P_1)$ при всевъзможните оценки на променливите са следните

P_0	P_1	$P_0 \,\&\, P_1$	$\neg(P_0 \,\&\, P_1)$	$P_0 \to \neg(P_0 \,\&\, P_1)$
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$
$\mathbb{F}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$
$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$
$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$

Казваме, че една формула е (съждителна) *тавтология*, ако тя приема стойност истина за всяка оценка на променливите. Така например, формулите $\neg P_0 \vee P_0$, $P_0 \to P_1 \vee P_0$ и $P_0 \,\&\, P_1 \to P_1$ са тавтологии, а формулите P_0, $P_0 \vee P_0$ и $P_0 \to P_1$ не са. В сила е следната теорема

Теорема 1.1 (за замяната)**.** Нека $\mathbf{A}$ е формула със съждителни променливи измежду $P_0, P_1, \ldots, P_k$ и нека $\mathbf{A}'$ се получава от $\mathbf{A}$, замествайки едновременно всички срещания на $P_0, P_1, \ldots, P_k$ съответно със съждителните формули $\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_k$. Тогава, ако $\mathbf{A}$ е тавтология, то и $\mathbf{A}'$ също е тавтология.

Прилагайки горната теорема към тавтологиите $\neg P_0 \vee P_0$, $P_0 \to P_1 \vee P_0$ и $P_0 \,\&\, P_1 \to P_1$ получаваме, че формулите $\neg\mathbf{A} \vee \mathbf{A}$, $\mathbf{A} \to \mathbf{B} \vee \mathbf{A}$ и $\mathbf{A} \,\&\, \mathbf{B} \to \mathbf{B}$ са тавтологии за произволни формули $\mathbf{A}$ и $\mathbf{B}$.

Казваме, че формулата $\mathbf{A}$ е тавтологично следствие на формулите $\mathbf{A}_1, \ldots, \mathbf{A}_n$ $(n \geq 0)$, ако формулата $\mathbf{A}_1 \to \mathbf{A}_2 \to \cdots \to \mathbf{A}_n \to \mathbf{A}$ е тавтология. Ще пишем $\dfrac{\mathbf{A}_1, \ldots, \mathbf{A}_n}{\mathbf{A}}$. Директно от дефинициите следва следната теорема.

Теорема 1.2. Нека $\mathbf{A}$ е тавтологично следствие на формулите $\mathbf{A}_1, \ldots, \mathbf{A}_n$ и V е оценка на променливите. Тогава, ако $\tilde{V}(\mathbf{A}_1) = \cdots = \tilde{V}(\mathbf{A}_n) = \mathbb{T}$, то $\tilde{V}(\mathbf{A}) = \mathbb{T}$.

С други думи, ако $\mathbf{A}$ е тавтологично следствие на $\mathbf{A}_1, \ldots, \mathbf{A}_n$, то $\mathbf{A}$ приема стойност истина винаги, когато $\mathbf{A}_1, \ldots, \mathbf{A}_n$ са едновременно истина. Примери за едни от най-използваните тавтологични следствия са следните:

$\dfrac{}{\neg\mathbf{A} \vee \mathbf{A}}$ (правило за изключеното трето), $\qquad \dfrac{\mathbf{A} \vee \mathbf{B}}{\mathbf{B} \vee \mathbf{A}}$ (комутативност на дизюнкцията),

$\dfrac{\mathbf{A} \,\&\, \mathbf{B}}{\mathbf{B} \,\&\, \mathbf{A}}$ (комутативност на конюнкцията), $\qquad \dfrac{\mathbf{A}, \mathbf{A} \to \mathbf{B}}{\mathbf{B}}$ (модус поненс),

$\dfrac{\neg(\mathbf{A} \vee \mathbf{B})}{\neg\mathbf{A} \,\&\, \neg\mathbf{B}}$ и $\dfrac{\neg(\mathbf{A} \,\&\, \mathbf{B})}{\neg\mathbf{A} \vee \neg\mathbf{B}}$ (правила на Де Морган),

$\dfrac{\mathbf{A} \to \mathbf{B}}{\neg\mathbf{B} \to \neg\mathbf{A}}$ и $\dfrac{\neg\mathbf{B} \to \neg\mathbf{A}}{\mathbf{A} \to \mathbf{B}}$ (правила за контрапозицията),

$\dfrac{\mathbf{A} \vee (\mathbf{B} \,\&\, \mathbf{C})}{(\mathbf{A} \vee \mathbf{B}) \,\&\, (\mathbf{A} \vee \mathbf{C})}$ и $\dfrac{\mathbf{A} \,\&\, (\mathbf{B} \vee \mathbf{C})}{(\mathbf{A} \,\&\, \mathbf{B}) \vee (\mathbf{A} \,\&\, \mathbf{C})}$ (дистрибутивност на дизюнкзията и конюнкцията).

1.2 Множества. Основни операции върху множества.

Наивната, класическа, представа за множество е, че множество е всяка колекция от вида $\{x \mid \varphi(x)\}$, където φ е произволно свойство. В тази категория попадат всички добре познати множества, като

$$\{x \mid x \text{ е естествено просто число}\};$$

$$\{x \mid x \text{ положително реално число}\};$$

$$\{f \mid f \text{ е напрекъсната функция, дефинирана в } (0,1)\}.$$

Тази представа е напълно достатъчна в рамките на разглежданията, които се правят в повечето отделни математически дисциплини, като анализ, алгебра и геометрия. Това обаче не е достатъчно за да бъде прието това, като дефиниция на понятието множество, тъй като твърде голямата общност, която се допуска при избора на свойството φ, води много бързо до непреодолими абсурди. Наистина, нека разгледаме съвкупността

$$R = \{x \mid x \notin x\}.$$

Съгласно класическата представа R е множество. Нещо повече, то съдържа всички множества, използвани в математиката. Така например елементите на $\mathbb{N}$ са числата 0, 1, 2, ..., но самото множество $\mathbb{N}$ не е естествено число, т.е. $\mathbb{N} \notin \mathbb{N}$ и следователно $\mathbb{N} \in R$. По същия начин $\mathbb{Z} \notin \mathbb{Z}$, $\mathbb{Q} \notin \mathbb{Q}$, $\mathbb{R} \notin \mathbb{R}$, $\mathbb{C} \notin \mathbb{C}$ и значи $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C} \in R$. За това е естествено да предположим, че $R \notin R$. Но това автоматично означава, че R удовлетворява условието за принадлежност в R и значи $R \in R$, което е абсурдно. Следователно трябва да бъде в сила $R \in R$. Оттук и условието за принадлежност в R получаваме $R \notin R$ и отново стигаме до абсурд. Така получаваме противоречие, което не може да бъде преодоляно в рамките на наивната представа за множество. То е известно като парадокс на Ръсел.

За да се преодолее парадоксът на Ръсел, се използва така наречения аксиоматичен подход към понятието множество. В него колекциите от вида $\{x \mid \varphi(x)\}$ наричаме класове, а множеството е първично понятие (понятие, което не се дефинира), а трябва да се подчинява на някои основни свойства, които наричаме аксиоми:

Аксиома за обемност:

$$A = B \iff \forall x(x \in A \Leftrightarrow x \in B).$$

Интуитивно тази аксиома казва, че всяко множество еднозначно се определя от своите елементи. Нека тук отбележим, че за две множества A и B, казваме че A е подмножество на B и записваме $A \subseteq B$, ако всеки елемент на A е елемент и на B, т.е. ако е в сила твърдението

$$\forall x(x \in A \Rightarrow x \in B).$$

Така, аксиомата за обемност може да се изкаже още и чрез следната формула

$$A = B \iff A \subseteq B \;\&\; B \subseteq A.$$

Аксиома за празното множество: Класът

$$\emptyset = \{x \mid x \neq x\}$$

е множество.

Празното множество е единственото конкретно множество, чието съществуване е гарантирано непосредствено от аксиомите. То играе ролята на изграждащ елемент за цялата аксиоматична теория на множества. Ясно е, че за всяко множество A е в сила

$$\emptyset \subseteq A.$$

Аксиома за отделянето (обхващането): Нека A е множество, а φ е свойство. Тогава класът

$$\{x \mid x \in A \;\&\; \varphi(x)\}$$

е множество.

За краткост ще пишем $\{x \in A \mid \varphi(x)\}$, вместо $\{x \mid x \in A \ \& \ \varphi(x)\}$. Най-често аксиомата за отделянето се използва в следната форма: нека φ е свойство, такова че

$$\forall x(\varphi(x) \Rightarrow x \in A)$$

за някое фиксирано множество A. С други думи нека е изпълнено, че класът $\{x \mid \varphi(x)\}$ се съдържа изцяло в множеството A, т.е.

$$\{x \mid \varphi(x)\} \subseteq A.$$

Тогава аксиомата за отделянето ни гарантира, че класът $\{x \mid \varphi(x)\}$ е множество[1].

Аксиомата за отделянето ни позволява да въведем операциите сечение на множества чрез

$$A \cap B = \{x \in A \mid x \in B\}.$$

Директно от дефиницията следва, че $(A \cap B) \cap C = A \cap (B \cap C)$ и $A \cap B = B \cap A$. Действително

$$x \in (A \cap B) \cap C \iff x \in A,\ x \in B,\ x \in C \iff x \in A \cap (B \cap C)$$

и

$$x \in A \cap B \iff x \in A \text{ и } x \in B \iff x \in B \text{ и } x \in A \iff x \in B \cap A.$$

Освен сечение на две множества можем да въведем и сечение на произволен клас по следния начин: Ако φ е свойство, то дефинираме сечението на (всички множества от) класа $\{Y \mid \varphi(Y)\}$ да бъде класът

$$\bigcap\{Y \mid \varphi(Y)\} = \bigcap_{\varphi(Y)} Y \stackrel{def}{=} \{x \mid \forall Y(\varphi(Y) \Rightarrow x \in Y)\}.$$

В случай, че класът $\{Y \mid \varphi(Y)\}$ е непразен, т.е. съществува множество, да речем A, такова че $\varphi(A)$, то сечението $\bigcap\limits_{\varphi(Y)} Y$ е множество, защото $\bigcap\limits_{\varphi(Y)} Y \subseteq A$.

Освен операцията сечение можем да дефинираме операцията разлика на две множества чрез

$$A \setminus B = \{x \in A \mid x \notin B\}.$$

Аксиома за чифта: За всеки две множества A и B, класът

$$\{A, B\} = \{x \mid x = A \text{ или } x = B\}$$

е множество.

Множеството $\{A, B\}$ наричаме чифт (или още ненаредена двойка). В случай, че $A = B$, аксиомата за чифта ни дава, че едноелементния клас $\{A\}$ е множество. Множеството $\{A\}$ наричаме синглетон на A.

Чифтовете ни позволяват да въведем понятието наредена двойка (a, b) чрез

$$(a, b) = \{\{a\}, \{a, b\}\}.$$

За наредените двойки е в сила

$$(a, b) = (c, d) \iff a = c \text{ и } b = d.$$

Твърдението е очевидно в посоката отдясно-наляво. За да докажем твърдението в обратната посока, нека първо предположим, че $a = b$. Тогава $(a, b) = (a, a) = \{\{a\}\}$. Оттук и $(a, b) = (c, d)$ получаваме, че (c, d) е едноелементно множество, откъдето $c = d$ и значи $(c, d) = \{\{c\}\}$. Сега, използвайки отново $(a, b) = (c, d)$, получаваме $a = c$, откъдето $a = b = c = d$.

Нека сега предположим, че $a \neq b$. Тогава (a, b) и $\{a, b\}$ са двуелементни монжества. Оттук и $(a, b) = (c, d)$ следва, че множествата (c, d) и $\{c, d\}$ са също двуелементни множества, откъдето $\{a\} = \{c\}$ и $\{a, b\} = \{c, d\}$. Следователно $a = c$ и $b = d$.

Разполагайки с понятието наредени двойки, можем да дефинираме и понятието наредени тройки, четворки и т.н. чрез следната схема:

$$\begin{aligned} (a_1, a_2, a_3) &= ((a_1, a_2), a_3) \\ (a_1, a_2, a_3, a_4) &= ((a_1, a_2, a_3), a_4) \\ &\vdots \end{aligned}$$

[1] Защото в този случай $\{x \mid \varphi(x)\} = \{x \mid x \in A \ \& \ \varphi(x)\}$

Аксиома за обединението: За всяко множество A класът

$$\bigcup A = \bigcup_{Y \in A} Y = \{x \mid \exists Y \in A(x \in Y)\}$$

е множество.

Нека отбележим, че в света на аксиоматичната теория на множества всчки обекти, с евентуално изключение на класовете, са множества. Така елементите на множествата също са множества и не е изненадващо, че можем да обединим елементите на едно множество. Благодарение на аксиомата за обединението можем да дефинираме операцията обединение на две множества, чрез

$$A \cup B = \bigcup \{A, B\}.$$

Ясно е, че

$$A \cup B = \{x \mid x \in A \text{ или } x \in B\}$$

и

$$A \cup B = B \cup A$$

Основните три операции (обединение, сечение и разлика) върху множества имат следните свойства:

1.	$A \cap \emptyset = \emptyset$	$A \cup \emptyset = A$
2.	$A \cap B = B \cap A$	$A \cup B = B \cup A$
3.	$(A \cap B) \cap C = A \cap (B \cap C)$	$(A \cup B) \cup C = A \cup (B \cup C)$
4.	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
5.	$(A \cup B) \setminus C = (A \setminus C) \cup (B \setminus C)$	$(A \cap B) \setminus C = (A \setminus C) \cap (B \setminus C)$
6.	$A \setminus (B \cap C) = (A \setminus B) \cup (A \setminus C)$	$A \setminus (B \cup C) = (A \setminus B) \cap (A \setminus C)$

Използвайки операциите обединение на две множества и чифт можем да дефинираме операция, която условно наричаме операция наследник и означаваме с $S(A)$, а именно

$$S(A) = A \cup \{A\}.$$

Благодарение на операцията S можем да дефинираме отделните естествени числа чрез следната схема:

$$\begin{aligned}
0 &= \emptyset \\
1 &= S(0) = \{0\} \\
2 &= S(1) = \{0, 1\} \\
3 &= S(2) = \{0, 1, 2\} \\
&\vdots \\
n+1 &= S(n) = \{0, 1, 2 \ldots, n\} \\
&\vdots
\end{aligned}$$

В горния запис използвахме съкращения. Разписани, първите няколко естествени числа изглеждат така:

$$\begin{aligned}
0 &= \emptyset \\
1 &= \{\emptyset\} \\
2 &= \{\emptyset, \{\emptyset\}\} \\
3 &= \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\} \\
&\vdots
\end{aligned}$$

Възможността да дефинираме всички отделни естествени числа показва, че в аксоматичната теория на множествата има неограничен (некраен) брой множества. Независимо от това аксиомите, споменати до момента, не гарантират съществуването на безкрайно множество. Следващата аксиома прави именно това.

Аксиома за безкрайността: Съществува множество, съдържащо празното множество и затворено относно операцията S.

Множествата, които съдържат празното множество и са затворени относно операцията S се наричат индуктивни. Аксиомата за безкрайността казва, че класът $\{A \mid A \text{ е индуктивно}\}$ е непразен. Да забележим, че всяко индуктивно множествосъдържа всички естествени числа, дефинирани по-горе. Така аксиомата ни гарантира, че съществува множество, съдържащо всички естествени числа. Да разгледаме множеството

$$\mathbb{N} = \bigcap \{A \mid A \text{ е индуктивно}\}$$

Такова множество съществува, съгласно аксиомата за отделянето. Нека първо да видим, че $\mathbb{N}$ е индуктивно. Трябва да докажем, че $\emptyset \in \mathbb{N}$, и че за всяко $x \in \mathbb{N}$ имаме $S(x) \in \mathbb{N}$. Първото следва от това, че $\emptyset$ принадлежи на всяко индуктивно множество. За второто нека фиксираме $x \in \mathbb{N}$. Тогава $x \in A$ за всяко индуктивноA и значи $S(x) \in A$ за всяко индуктивно A, откъсето $S(x) \in \mathbb{N}$. Следователно $\mathbb{N}$ е индуктивно.

Така, $\mathbb{N}$ е най-малкото (по отношение на включването) индуктивно множество и значи $\mathbb{N}$ е най-малкото множество, съдържащо естествените числа. За това наричаме $\mathbb{N}$ *множество на естествените числа.*

Аксиома за множество от подмножествата: За всяко множество A класът $\mathcal{P}(A) = \{B \mid B \subseteq A\}$ е множество.

Благодарение на тази аксиома можем да дефинираме понятието декартово произведение на две множества, а именно: за две множества A и B с $A \times B$ ще означаваме класът

$$A \times B = \{(a, b) \mid a \in A,\ b \in B\}$$

От дефиницията за наредена двойка следва, че за всяко $a \in A$ и $b \in B$ е в сила $(a, b) \subseteq \mathcal{P}(A \cup B)$. Следователно $A \times B \subseteq \mathcal{P}(\mathcal{P}(A \cup B))$ и съгласно аксиомата за отделянето $A \times B$ е множество.

Аксиома за замяната: Нека Φ е правило, което на всяко множество x съпоставя най-много едно множество $\Phi(x)$, т.е.

$$\forall x((\Phi(x) = y \ \&\ \Phi(x) = y') \Rightarrow y = y')$$

Тогава за всяко множество A, класът

$$\{\Phi(x) \mid x \in A\} = \{y \mid \exists x(x \in A \ \&\ \Phi(x) = y)\}$$

е множество.

1.3 Бинарни релации.

Определение 1.3. *Бинарна релация* ще наричаме всяко множество от наредени двойки.

Понятието бинарна релеция е едно от основните понятия в математиката. То се появява непрекъснато както в математиката, така и в ежедневието ни. Обичайно, релациите отразяват наличието на връзка между елементите, съставляващи наредените двойки, принадлежащи на релацията. Така например, бихме могли да разглеждаме релацията, която казва, че между два града се осъществяват директни самолетни полети, или пък релацията, казваща че двама души имат роднинска връзка.

Ако R е бинарна релация, ще означаваме

$$\begin{aligned} \mathrm{dom}(R) &= \{a \mid (a,b) \in R \text{ за някое } b\}, \\ \mathrm{range}(R) &= \{b \mid (a,b) \in R \text{ за някое } a\} \end{aligned}$$

Ясно е, че за всяка бинарна релация R е в сила

$$\mathrm{dom}(R), \mathrm{range}(R) \subseteq \bigcup\bigcup R$$

и следователно $\mathrm{dom}(R)$ и $\mathrm{range}(R)$ са множества (защото R е множество). В частност $R \subseteq \mathrm{dom}(R) \times \mathrm{range}(R)$. Множеството $\mathrm{dom}(R)$ наричаме дефиниционна област на R, а $\mathrm{range}(R)$ — множество от стойности на R.

Да отбележим, че $\emptyset$ е бинарна релация с $\mathrm{dom}(\emptyset) = \mathrm{range}(\emptyset) = \emptyset$. Освен това, за всяко множество A с id_A ще бележим бинарната релация

$$\mathrm{id}_A = \{(x,x) \mid x \in A\}.$$

За id_A е в сила $\mathrm{dom}(\mathrm{id}_A) = \mathrm{range}(\mathrm{id}_A) = A$.

За две бинарни релации R и Q с $Q \circ R$ ще бележим релацията

$$Q \circ R = \{(a,b) \mid \exists x((a,x) \in R \text{ и } (x,b) \in Q\}.$$

Релацията $Q \circ R$ ще наричаме композиция на релациите R и Q.

Ясно е, че

$$\begin{aligned} \mathrm{dom}(Q \circ R) &\subseteq \mathrm{dom}(R) \\ \mathrm{range}(Q \circ R) &\subseteq \mathrm{range}(Q) \end{aligned}$$

При това

$$\begin{aligned} \mathrm{range}(R) \subseteq \mathrm{dom}(Q) &\implies \mathrm{dom}(Q \circ R) = \mathrm{dom}(R), \\ \mathrm{dom}(Q) \subseteq \mathrm{range}(R) &\implies \mathrm{range}(Q \circ R) = \mathrm{range}(Q). \end{aligned}$$

Нека още отбележим, че за всяка бинарна релация R

$$\emptyset \circ R = R \circ \emptyset = \emptyset$$

и за всяко множество A

$$\begin{aligned} R \circ \mathrm{id}_A &= \{(a,b) \in R \mid a \in A\}, \\ \mathrm{id}_B \circ R &= \{(a,b) \in R \mid b \in B\} \end{aligned}$$

В частност

$$\begin{aligned} R \circ \mathrm{id}_A = R &\iff \mathrm{dom}(R) \subseteq A, \\ \mathrm{id}_B \circ R = R &\iff \mathrm{range}(R) \subseteq B. \end{aligned}$$

Твърдение 1.4. Композицията на релации е асоциативна, т.е.

$$P \circ (Q \circ R) = (P \circ Q) \circ R.$$

Доказателство. За всяко a и b имаме

$$\begin{aligned}
(a,b) \in P \circ (Q \circ R) &\iff (a,x) \in Q \circ R \text{ и } (x,b) \in P \text{ за някое } x \\
&\iff (a,y) \in R,\ (y,x) \in Q \text{ и } (x,b) \in P \text{ за някои } x, y \\
&\iff (a,y) \in R \text{ и } (y,b) \in P \circ Q \text{ за някое } y \\
&\iff (a,b) \in (P \circ Q) \circ R.
\end{aligned}$$

□

Ако R е бинарна релация, то с R^{-1} ще означаваме релацията

$$R^{-1} = \{(y,x) \mid (x,y) \in R\}.$$

R^{-1} наричаме обратна релация на R.

Ясно е, че

$$\begin{aligned}
\mathrm{dom}(R^{-1}) &= \mathrm{range}(R) \\
\mathrm{range}(R^{-1}) &= \mathrm{dom}(R).
\end{aligned}$$

Твърдение 1.5. Нека R и Q са бинарни релации. Тогава

$$(Q \circ R)^{-1} = R^{-1} \circ Q^{-1}.$$

Доказателство. В сила е

$$\begin{aligned}
(a,b) \in (Q \circ R)^{-1} &\iff (b,a) \in Q \circ R \\
&\iff (b,x) \in R \text{ и } (x,a) \in Q \text{ за някое } x \\
&\iff (a,x) \in Q^{-1} \text{ и } (x,b) \in R^{-1} \text{ за някое } z \\
&\iff (a,b) \in R^{-1} \circ Q^{-1}.
\end{aligned}$$

□

1.4 Функции.

Казваме, че бинарната релация f е *функция* (или още *изображение*), ако f е *еднозначна* бинарна релация, т.е.

$$((x,y) \in f \text{ и } (x,y') \in f) \Longrightarrow y = y'$$

за всяко x, y и y'.

Най-простите примери за функции са $\emptyset$ и релациите от вида id_A. Освен това, ако Φ е правило, което на всяко x съпоставя най-много едно $\Phi(x)$, то съгласно аксиомата за замяната за всяко множество A класът $\{\Phi(x) \mid x \in A\}$ е множество и следователно

$$\{(x, \Phi(x)) \mid x \in A\}$$

е функция за всяко множество A.[2]

Ще пишем $f : A \to B$ в случай, че f е функция, $\mathrm{dom}(f) = A$ и $\mathrm{range}(f) \subseteq B$. В този случай казваме, че f е функция дефинирана в A и приемаща стойности в B (или по-просто — f е функция от A в B).

С други думи означението $f : A \to B$ означава, че са в сила следните три свойства:

1. $f \subseteq A \times B$;
2. f е еднозначна релация;
3. f е *тотална* релация, т.е. за всяко $a \in A$ съществува $b \in B$, такова че $(a,b) \in f$.

Да отбележим, че свойствата тоталност и еднозначност могат да бъдат изказани и като едно единствено свойство, а именно — за всяко $x \in A$ съществува единствено $y \in B$, такова че $(x,y) \in f$. От дефиницията директно следва, че ако $f : A \to B$, то $f : A \to C$ за всяко C, за което $B \subseteq C$. В частност $\emptyset : \emptyset \to B$ за всяко B и $\mathrm{id}_A : A \to B$ за всяко B, такова че $A \subseteq B$.

Ако f е функция, то за всяко $x \in \mathrm{dom}(f)$ с $f(x)$ ще означаваме единствения елемент на $\mathrm{range}(f)$, за който $(x, f(x)) \in f$. $f(x)$ наричаме *образ* на x, а x — *първообраз* на $f(x)$ (под действието на f).

За функцията f и множествата X и Y дефинираме

$$\begin{aligned} f(X) &= \{f(x) \mid x \in X\}, \\ f^{-1}(Y) &= \{x \mid f(x) \in Y\}. \end{aligned}$$

Множеството $f(X)$ наричаме *образ* на X под действието на f, а множеството $f^{-1}(Y)$ наричаме *първообраз* на Y спрямо f. Ясно е, че

$$\begin{aligned} f(\mathrm{dom}(f)) &= \mathrm{range}(f), \\ f^{-1}(\mathrm{range}(f)) &= \mathrm{dom}(f). \end{aligned}$$

Множеството от всички функции от A в B ще бележим с B^A, т.е.

$$B^A = \{f \mid f : A \to B\}.$$

Особен интерес представлява множеството 2^A, т.е. функциите изобразяващи A в $\{0,1\}$. Елементите на 2^A ще наричаме характеристични функции (на подмножествата на A). Множеството 2^A е тясно свързано с множеството от подмножествата на A. Наистина, ако $B \subseteq A$, с B свързваме неговата характеристична функция $\chi_B \in 2^A$, дефинирана чрез правилото

$$\chi_B(x) = \begin{cases} 1, & x \in B \\ 0, & x \notin B \end{cases}, \text{ за } x \in A.$$

Обратно, ако $f \in 2^A$, с f свързваме подмножеството P_f на A, дефинирано с $P_f = \{x \in A \mid f(x) = 1\}$. Тогава

$$\chi_{P_f} = f \text{ и } P_{\chi_B} = B.$$

Поради тази естествена връзка между множествата 2^A и $\mathcal{P}(A)$ те често се отъждествяват, като в различни направления на математиката с 2^A се означава множеството от подмножествата на A.

---

[2] Ако f е функция, то можем да разгледаме правилото Φ

$$\Phi : \begin{cases} x \mapsto y, & \text{ако } x \in \mathrm{dom}(f) \text{ и } (x,y) \in f \\ \text{не е дефинирано}, & \text{иначе} \end{cases}$$

Така неформално можем да кажем, че функциите са точно еднозначните правила Φ, за които графиката им $\{(x,y) \mid y = \Phi(x)\}$ е множество.

Твърдение 1.6. Нека f и g са функции. Тогава $f \circ g$ също е функция. При това

$$\mathrm{dom}(f \circ g) = \{x \in \mathrm{dom}(g) \mid g(x) \in \mathrm{dom}(f)\}$$

и за всяко $x \in \mathrm{dom}(f \circ g)$ е в сила

$$(f \circ g)(x) = f(g(x)).$$

В частност, ако $g : A \to B$ и $f : B \to C$, то $f \circ g : A \to C$

Доказателство. За да докажем, че $f \circ g$ е функция, нека $(x, y) \in f \circ g$ и $(x, y') \in f \circ g$. Тогава съществуват z и z', такива че $(x, z) \in g$ и $(z, y) \in f$, и $(x, z') \in g$ и $(z', y) \in f$. Оттук и това, че f и g са функции последователно получаваме $z = z'$ и $y = y'$. Следователно $f \circ g$ е еднозначна релация и значи $f \circ g$ е функция.

За да определим дефиниционната област на $f \circ g$ да забележим, че за всяко x е в сила

$$\begin{aligned} x \in \mathrm{dom}(f \circ g) &\iff (x, y) \in f \circ g \text{ за някое } y \\ &\iff (x, z) \in g \text{ и } (z, y) \in f \text{ за някои } z, y \\ &\iff x \in \mathrm{dom}(g) \text{ и } g(x) \in \mathrm{dom}(f). \end{aligned}$$

и следователно

$$\mathrm{dom}(f \circ g) = \{x \in \mathrm{dom}(g) \mid g(x) \in \mathrm{dom}(f)\}.$$

Нека $x \in \mathrm{dom}(f \circ g)$ и нека $y = (f \circ g)(x)$, т.е. $(x, y) \in f \circ g$. Тогава за някое z имаме $(x, z) \in g$ и $(z, y) \in f$, т.е. $z = g(x)$ и $y = f(z)$, откъдето

$$(f \circ g)(x) = y = f(z) = f(g(x)).$$

Накрая нека $g : A \to B$ и $f : B \to C$. Тогава $\mathrm{range}(g) \subseteq B = \mathrm{dom}(f)$ и следоватлено $\mathrm{dom}(f \circ g) = \mathrm{dom}(f) = A$. От друга страна $\mathrm{range}(f \circ g) \subseteq \mathrm{range}(f) \subseteq C$ и следователно $f \circ g : A \to C$.

□

От свойствата на композиция на релации имаме

1. $(f \circ g) \circ h = f \circ (g \circ h)$;
2. $f \circ \mathrm{id}_A = \mathrm{id}_B \circ f = f$ за всеки две множества A и B, такива че $\mathrm{dom}(f) \subseteq A$ и $\mathrm{range}(f) \subseteq B$.

1.5 Обратна функция. Инекция, сюрекция и биекция.

Казваме, че финкцията f е *обратима*, ако релацията f^{-1} е функция. С други думи функцията f е обратима тогава и само тогава, когато f^{-1} е еднозначна релация. В случай, че f е обратима, функцията f^{-1} наричаме *обратна* на f.

Казваме, че функцията f е *инекция*, ако за всяко $x, x' \in \mathrm{dom} f$ е в сила

$$f(x) = f(x') \Rightarrow x = x'$$

или еквивалентно

$$x \neq x' \Rightarrow f(x) \neq f(x')$$

Твърдение 1.7. Една функция f е обратима тогава и само тогава, когато f е инекция.

Доказателство. За произволни y, y' и x свойството

$$((y, x) \in f^{-1} \ \& \ (y, x') \in f^{-1}) \Rightarrow x = x')$$

е еквивалентно на

$$((x, y) \in f \ \& \ (x', y) \in f) \Rightarrow x = x'),$$

т.е.

$$(f(x) = y \text{ и } f(x') = y) \Rightarrow x = x'.$$

Оттук релацията f^{-1} е функция тогава и само, когато f е инекция.

□

От дефиницията на обратна релация имаме

$$\left(f^{-1}\right)^{-1} = f$$

и следователно, ако f е обратима функция, то и f^{-1} също е обратима, като обратната ѝ е точно f.

От свойствата на композиция на релации знаем, че

$$(f \circ g)^{-1} = g^{-1} \circ f^{-1}$$

и следователно, ако f и g са обратими функции, то $f \circ g$ също е обратима функция. В частност, композиция на инективни функции също е инективна.

Твърдение 1.8. Нека f е обратима. Тогава

$$\begin{aligned} f \circ f^{-1} &= \mathrm{id}_{\mathrm{range}(f)} \\ f^{-1} \circ f &= \mathrm{id}_{\mathrm{dom}(f)} \end{aligned}$$

Доказателство. Имаме

$$\begin{aligned} \mathrm{dom}(f^{-1}) &= \mathrm{range}(f) \\ \mathrm{range}(f^{-1}) &= \mathrm{dom}(f) \end{aligned}$$

откъдето

$$\mathrm{dom}(f \circ f^{-1}) = \mathrm{dom}(f^{-1}) = \mathrm{range}(f) = \mathrm{dom}(\mathrm{id}_{\mathrm{range}(f)}),$$

като при това за всяко $y \in \mathrm{range}(f)$, ако с x означим елемента на $\mathrm{dom}(f)$, за който $f(x) = y$, то тогава

$$f \circ f^{-1}(y) = f(f^{-1}(y)) = f(x) = y = \mathrm{id}_{\mathrm{range}(f)}(y).$$

Следователно

$$f \circ f^{-1} = \mathrm{id}_{\mathrm{range}(f)}.$$

Равенството

$$f^{-1} \circ f = \mathrm{id}_{\mathrm{dom} f}$$

се доказва аналогично.

□

Да забележим, че ако $f : A \to B$ и f е инекция не е задължително $f^{-1} : B \to A$. Това е така, защото

$$\mathrm{dom}(f^{-1}) = \mathrm{range}(f) \subseteq B$$

като в общия случай $\mathrm{range}(f) \neq B$.

Казваме, че f е *сюрекция на A върху B*, ако $f : A \to B$ и $\mathrm{range}(f) = B$, т.е. $f(A) = B$. Да забележим, че ако $g : A \to B$ и $f : B \to C$ са сюрекции, то $f \circ g : A \to C$ също е сюрекция. Наистина, имаме $g(A) = B$ и $f(B) = C$, то $f \circ g(A) = f(g(A)) = f(B) = C$.

Казваме, че f е *биекция на A върху B*, ако $f : A \to B$ е едновременно инекция и сюрекция на A върху B. Така, ако $f : A \to B$, то

$$f \text{ е биекция на } A \text{ върху } B \iff f^{-1} : B \to A.$$

При това, ако $g : A \to B$ и $f : B \to C$ са биекции, то $f \circ g : A \to C$ също е биекция.

ГЛАВА 2

Естествени числа. Крайни множества. Комбинаторика

2.1 Естествени числа

Както е обичайно, множеството на естествените числа ще означаваме с $\mathbb{N}$. Да напомним, че $\mathbb{N}$ е най-малкото (по-отношение на $\subseteq$) индуктивно множество. Оттук следва следната теорема:

Теорема 2.1 (непълна математическа индукция)**.** Нека $A \subseteq \mathbb{N}$ е такова, че

1. $0 \in A$;
2. $x \in A \Rightarrow S(x) \in A$.

Тогава $A = \mathbb{N}$.

Доказателство. Свойствата (1) и (2) означават, че A е индуктивно множество. Оттук $\mathbb{N} \subseteq A$ и значи $A = \mathbb{N}$. □

Обичайно, непълната математическа индукция се прилага по следния начин:

Нека φ е свойство за естествените числа (т.е. $\varphi(x) \Rightarrow x \in \mathbb{N}$) и нека са в сила

1. $\varphi(0)$;
2. $\varphi(x) \Rightarrow \varphi(S(x))$.

Тогава всяко естествено число има свойството φ.

Наистина, ако означим с $A = \{x \mid \varphi(x)\}$, то $A \subseteq \mathbb{N}$ и A удовлетворява условията на теоремата. Следователно $A = \mathbb{N}$, т.е. за всяко $x \in \mathbb{N}$ е в сила $\varphi(x)$.

Лема 2.2. Ако $x \in \mathbb{N}$ и $y \in x$, то $y \subseteq x$

Доказателство. Ще докажем твърдението с индукция по x. Твърдението е очевидно за числото 0. Нека сега предположим, че твърдението е вярно за числото x. Ще докажем твърдението за $S(x)$. За целта нека

$$y \in S(x).$$

Тогава

$$y \in x \text{ или } y = x.$$

Тъй като $x \subseteq S(x)$,

$$\text{ако } y = x, \text{ то } y \subseteq S(x).$$

От друга страна,

$$\text{ако } y \in x, \text{ то } y \subseteq x, \text{ откъдето } y \subseteq S(x).$$

Така и в двата възможни случая $y \subseteq S(x)$, което доказва индукционната стъпка.

□

Лема 2.3. Ако $x \in \mathbb{N}$, то $x \subseteq \mathbb{N}$. С други думи, елементите на едно естествено число са естествени числа.

Доказателство. Индукция по x. Твърдението е очевидно за $x = 0$. Нека сега $x \in \mathbb{N}$ е такова, че $x \subseteq \mathbb{N}$. Тогава, ако $y \in S(x)$, то или $y = x$ и значи $y \in \mathbb{N}$, или $y \in x \subseteq \mathbb{N}$ и отново имаме $y \in \mathbb{N}$, което доказва индукционната стъпка.

□

Лема 2.4. Ако $x \in \mathbb{N}$, то $x \notin x$. В частност $x \neq S(x)$.

Доказателство. Индукция по x. Твърдението е очевидно за числото 0. Нека сега $x \in \mathbb{N}$ е такова, че $x \notin x$. Да допуснем, че

$$S(x) \in S(x),$$

т.е.

$$S(x) \in x \text{ или } S(x) = x.$$

Тогава,

$$\text{ако } S(x) \in x, \text{ то } S(x) \subseteq x, \text{ откъдето } x \in x.$$

От друга страна,

$$\text{ако } S(x) = x, \text{ то отново } x \in x.$$

Така допускането е грешно и следователно $S(x) \notin S(x)$, с което индукционната стъпка е доказана.

□

Оттук нататък за две естествени числа x и y вместо $x \in y$ ще пишем $x < y$. Съгласно втората лема $<$ е транзитивна релация в $\mathbb{N}$, т.е.

$$\text{ако } x < y \text{ и } y < z, \text{ то } x < z.$$

Настина, от $x < y$ и $y < z$ (т.е. $x \in y$ и $y \in z$) следва, че $y \subseteq z$, откъдето $x < z$.

Съгласно последната лема, релацията $<$ е ирефлексивна, т.е.

$$x \not< x (\text{за всяко естествено } x).$$

Релации, които са транзитивни и ирефлексивни наричаме частични наредби. Така следователно $<$ е частична наредба, както в $\mathbb{N}$, така и във всяко естествено число. Да отбележим, че за всяко естествено x, най-големият елемент на $S(x)$ е x и

$$y < S(x) \iff (y < x \text{ или } y = x), \text{ т.е. } y \leq x$$

Лема 2.5. Ако $y < x$, то $S(y) \leq x$

Доказателство. Индукция по x. Твърдението е очевидно за числото 0. Нека съга твърдението е вярно за x. Нека

$$y < S(x).$$

Тогава

$$y < x \text{ или } y = x.$$

Сега

$$\text{ако } y < x, \text{ то } S(y) < x \text{ и значи } S(y) < S(x).$$

От друга страна,

$$\text{ако } y = x, \text{ то } S(y) = S(x).$$

Следователно $S(y) \leq S(x)$.

□

Лема 2.6. Релацията $<$ е добра наредба във всяко естествено число. С други думи, ако A е непразно подмножество на естественото число x, то в A има най-малък елемент относно релацията $<$, т.е. съществува $m \in A$, такова че $m \leq n$ за всяко $n \in x$.

Доказателство. Твърдението е очевидно за числото 0. Нека сега $x \in \mathbb{N}$ е такова, че $<$ е добра наредба в x. Нека $A \subseteq S(x)$ е непразно множество и нека $A' = x \cap A$. Ако $A' = \emptyset$, то $A = \{x\}$ и тогава x е най-малкият елемент на A. Нека сега A' е непразно. Тогава от избора на x следва, че в A' има най-малък елемент m, т.е. $m \in A'$ и $m \leq y$ за всяко $y \in A'$. Но за всяко $y \in A$ е в сила или $y \in A'$, или $y = x$. Следователно, тъй като $m < x$, m е най-малкият елемент на A, с което индукционната стъпка е доказана.

□

Лема 2.7. Релацията $<$ е линейна наредба в $\mathbb{N}$, т.е. за всеки две естествени числа е изпълнено

$$x < y \text{ или } x = y, \text{ или } y < x$$

Доказателство. Нека $x \neq y$ са естествени числа. Без ограничение можем да считаме, че $x \not\subseteq y$. Тогава множеството $x \setminus y$ е непразно и нека m е най-малкият му елемент. Ако $m = 0$, то $0 \not< y$ и значи $y = 0 < x$. В противен случай $m = S(z)$ за някое естествено z. Тогава $z < m$ и значи $z < x$ (т.е. $z \in x$). Оттук $z < y$ (тъй като m е най-малкият елемент на $x \setminus y$). Сега от предната лема имаме или $m < y$ или $m = y$. Но, ако $m < y$ (т.е. $m \in y$), то бихме имали $m \notin x \setminus y$ и следователно $y = m < x$.

□

Теорема 2.8. Релацията $<$ е добра наредба в $\mathbb{N}$.

Доказателство. Нека сега $A \subseteq N$ е непразно и нека фиксираме $x \in A$. Нека $A' = x \cap A$. Нека първо A' е непразно и нека m е най-малкият му елемент. Нека $y \in A$. Ако $y < x$, то $y \in A'$ и значи $m \leq y$. В противен случай $m < x \leq y$ и значи m е най-малкият елемент на A.

Нека сега $A' = \emptyset$. Тогава за всяко $y \in A$ е в сила $y \not< x$, откъдето $x \leq y$ и следователно x е най-малкият елемента на A.

□

Теорема 2.9 (пълна математическа индукция)**.** Нека $A \subseteq \mathbb{N}$ е такова, че за всяко естествено x е в сила

$$\{y | y < x\} \subseteq A \Rightarrow x \in A.$$

Тогава $A = \mathbb{N}$.

Доказателство. Да допуснем, че $A \neq \mathbb{N}$. Тогава множеството $\mathbb{N} \setminus A$ е непразно и нека m е най-малкият му елемент. Тогава за всяко $y < m$ е в сила $y \in A$, т.е. $\{y \mid y < m\} \subseteq A$. Оттук $m \in A$, което противоречи с условието на теоремата. Следователно $A = \mathbb{N}$.

2.2 Крайни редици. Дефиниции чрез рекурсия

Ще казваме, че α е *крайна редица* с елементи от B, ако $\alpha : n \to B$ за някое $n \in \mathbb{N}$. Множеството от всички крайни редици с елементи от B ще бележим с $B^{<\mathbb{N}}$. Ще казваме, че редицата $\alpha \in B^{<\mathbb{N}}$ има дължина n и ще пишем $\mathrm{lh}(\alpha) = n$ (и още $|\alpha| = n$), ако $\mathrm{dom}(\alpha) = n$.

Ако $\alpha \in B^{<\mathbb{N}}$ и $b \in B$, то с

$$\alpha \cdot b$$

(или още $\alpha^* b$ и $\alpha^\wedge b$) ще означаваме

$$\alpha \cdot b = \alpha \cup (\mathrm{lh}(\alpha), b).$$

Ясно е, че $\alpha \cdot b \in B^{<\mathbb{N}}$, като $\mathrm{lh}(\alpha \cdot b) = S(\mathrm{lh}(\alpha))$. При това $\alpha(x) = (\alpha \cdot b)(x)$ за всяко $x < \mathrm{lh}(\alpha)$, а $(\alpha \cdot b)(\mathrm{lh}(\alpha)) = b$.

Ако α е крайна редица с $\mathrm{lh}(\alpha) = n$, а $i < n$, вместо $\alpha(i)$ пишем α_i, а за редицата α използваме записа

$$\alpha = \alpha_0, \alpha_1, \ldots, \alpha_{n-1}.$$

Теорема 2.10 (дефиниция чрез рекурсия)**.** Нека B е множество, $b \in B$ и $g : B \to B$. Тогава съществува единствена $f : \mathbb{N} \to B$, такава че

(1) $f(0) = b$;

(2) $f(S(n)) = g(f(n))$.

Доказателство. Първо ще докажем единствеността. Нека $f, f' : \mathbb{N} \to B$ удовлетворяват (1) и (2). Да допуснем, че $f \neq f'$ и нека m е най-малкото, за което $f(m) \neq f'(m)$. Очевидно $m \neq 0$ и значи $m = S(m')$ за някое $m' \in \mathbb{N}$. Тогава $m' < m$, откъдето $f(m') = f'(m')$. Оттук (съгласно свойство (2)) получаваме

$$f(m) = g(f(m')) = g(f'(m')) = f'(m),$$

което противоречи на избора на m. Следователно $f = f'$.

За да докажем съществуването нека разгледаме релацията $f \subseteq \mathbb{N} \times B$, за която $(n, y) \in f$ тогава и само тогава, когато съществува крайна редица $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_n$ с елементи от B, такава че

$$\alpha_0 = b,\ \alpha_{S(i)} = g(\alpha_i) \text{ за всяко } i < n,\ \text{ и } \alpha_n = y.$$

Първо ще докажем, че f е функция. За целта нека $(n, y) \in f$ и $(n, y') \in f$. Тогава съществуват крайни редици $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_n$ и $\alpha' = \alpha'_0, \alpha'_1, \ldots, \alpha'_n$, такива че $\alpha_0 = \alpha'_0 = b$, $\alpha_{S(i)} = g(\alpha_i)$ и $\alpha'_{S(i)} = g(\alpha'_i)$ за $i < n$, и $\alpha_n = y$, $\alpha'_n = y'$. Да допуснем, че $\alpha \neq \alpha'$ и нека m е най-малкото, за което $\alpha_m \neq \alpha'_m$. Очевидно $m \neq 0$ и значи $m = S(k)$ за някое k. Но тогава $k < m$ и значи $\alpha_k = \alpha'_k$, откъдето $\alpha_m = g(\alpha_k) = g(\alpha'_k) = \alpha'_m$, което противоречи с избора на m. Следователно $\alpha = \alpha'$ и значи $y = y'$. Следователно f е функция.

Нека сега предположим, че f е дефинирана за някое естествено n. Тогава съществува крайна редица $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_n$, такава че $\alpha_0 = b$, $\alpha_{S(i)} = g(\alpha_i)$ за $i < n$, и $\alpha_n = f(n)$. Тогава за крайната редица $\alpha' = \alpha_0, \alpha_1, \ldots, \alpha_n, g(\alpha_n)$ имаме $\alpha'_0 = \alpha_0 = b$, $\alpha'_{S(i)} = \alpha_{S(i)} = g(\alpha_i) = g(\alpha'_i)$ за $i < n$, и $\alpha'_{S(n)} = g(\alpha_n) = g(\alpha'_n)$ и следователно f е дефинирана за $S(n)$, като при това $f(S(n)) = g(\alpha_n) = g(f(n))$. Така доказахме, че ако f е дефинирана за естественото число n, то f е дефинирана и за $S(n)$, като при това $f(S(n)) = g(S(n))$. Оттук, $f(0) = b$ и принципа за математическата индукция следва, че f е функция, изпълняваща изискванията на теоремата.

□

Теорема 2.11. Нека A и B са множества, $h : A \to B$ и $g : A \times B \to B$. Тогава съществува единствена $f : A \times \mathbb{N} \to B$, такава че

(1) $f(a, 0) = h(0)$;

(2) $f(a, S(n)) = g(a, f(a, n))$.

Доказателство. Първо ще докажем единствеността. Нека f и f' удовлетворяват условията на теоремата и да допуснем, че $f \neq f'$. Тогава за някое $a \in A$ има $x \in \mathbb{N}$, такова че $f(a, x) \neq f'(a, x)$. Нека m е най-малкото, за което

$f(a,m) \neq f'(a,m)$. Очевидно $m = 0$ (т.к. $f(a,0) = h(a) = f'(a,0)$) и значи $m = S(n)$ за някое $n \in \mathbb{N}$. Тогава $n < m$ и следователно $f(a,n) = f'(a,n)$. Оттук

$$f(a,m) = f(a,S(n)) = g(a,f(a,n)) = g(a,f'(a,n)) = f'(a,S(n)) = f'(a,m),$$

което противоречи на избора на m. Следователно $f = f'$.

За да докажем съществуването нека за $a \in A$ означим $b_a = h(a)$, а g_a е функцията $g_a : B \to B$, дефинирана чрез $g_a(x) = g(a,x)$. Нека f_a е (единствената) функция от $\mathbb{N}$ в B, за която $f_a(0) = b_a$ и $f_a(S(n)) = g_a(f_a(n))$. Нека $f : A \times \mathbb{N} \to B$ е дефинирана чрез

$$f(a,n) = f_a(n).$$

Тогава за произволно $a \in A$ имаме $f(a,0) = f_a(0) = b_a = h(0)$ и $f(a,S(n)) = f_a(S(n)) = g_a(f_a(n)) = g(a,f(a,n))$ и значи f изпълнява изискванията на теоремата.

□

С помощта на горната теорема дефинираме операциите събиране и умножение на естествени числа чрез следните рекурсии:

(1) $m + 0 = m$;

(2) $m + S(n) = S(m+n)$

и

(1') $m0 = 0$;

(2') $mS(n) = mn + m$.[1]

Да отбележим, че (тъй като $1 = S(0)$) за всяко естествено m имаме

$$m + 1 = m + S(0) = S(m+0) = S(m).$$

Тогава рекурентната дефиниця на събирането приема вида

(1) $m + 0 = m$;

(2) $m + (n+1) = (m+n) + 1$,

а тази на умножението:

(1') $m0 = 0$;

(2') $m(n+1) = mn + m$.

Ще докажем следните основни свойства на операциите събиране и умножение:

Твърдение 2.12. За операцията събиране са в сила:

(i) $(k+m) + n = k + (m+n)$;

(ii) $m + n = n + m$.

Доказателство. Всички твърдения доказваме с индукция по n (при фиксирани k и m).

(i) Асоциативността на събирането следва от

$$(k+m) + 0 = k + m = k + (m+0).$$

и това, че ако $(k+m) + n = k + (m+n)$, то

$$\begin{aligned} (k+m) + (n+1) &= ((k+m)+n) + 1 \\ &= (k + (m+n)) + 1 \\ &= k + ((m+n)+1) \;=\; k + (m + (n+1)). \end{aligned}$$

[1] С други думи, за да дефинираме събирането полагаме $A = B = \mathbb{N}$, $h = \mathrm{id}_{\mathbb{N}}$, а g действа по правилото $g(a,x) = S(x)$. За умножението отново $A = B = \mathbb{N}$, но този път h е функцията, която на всяко число съпоставя 0, а g действа по правилото $g(a,x) = x + a$.

(ii) От $0+0=0$ и това, че ако $0+n=n$, то $0+(n+1)=(0+n)+1=n+1$ имаме

$$0+n=n.$$

От друга страна $1+0=1=0+1$ и, ако $1+n=n+1$, то $1+(n+1)=(1+n)+1=(n+1)+1$. Следователно

$$1+n=n+1.$$

Сега комутативността на събирането следва от $m+0=m=0+m$ и от това, че ако $m+n=n+m$, то

$$\begin{aligned} m+(n+1) &= (m+n)+1 \\ &= (n+m)+1 \\ &= 1+(n+m) \\ &= (1+n)+m \;=\; (n+1)+m. \end{aligned}$$

□

Твърдение 2.13. За операцията умножение са в сила:

(i) $k(m+n)=km+kn$;

(ii) $(km)n=k(mn)$;

(iii) $mn=nm$.

Доказателство. Всички твърдения доказваме с индукция по n (при фиксирани k и m).

(i) Дистрибутативността (отляво) на умножението следва от $k(m+0)=km=km+0=km+k0$ и това, че ако $k(m+n)=km+kn$, то

$$\begin{aligned} k(m+(n+1)) &= k((m+n)+1) \\ &= k(m+n)+k \\ &= (km+kn)+k \\ &= km+(kn+k) \;=\; km+k(n+1). \end{aligned}$$

(ii) Асоциативността на умножението следва от $(km)0=0=k0=k(m0)$ и от това, че

$$\begin{aligned} (km)(n+1) &= (km)n+km \\ &= k(mn)+km \\ &= k(mn+m) \;=\; k(m(n+1)). \end{aligned}$$

(iii) От $00=0$ и това, че ако $0n=0$, то $0(n+1)=0n+0=0+0=0$, следва, че

$$0n=0$$

От друга страна, от $(m+1)0=0=0+0=m0+0$ и това, че ако $(m+1)n=mn+n$, то

$$\begin{aligned} (m+1)(n+1) &= (m+1)n+(m+1) \\ &= (mn+n)+m+1 \\ &= (mn+m)+(n+1) \;=\; m(n+1)+(n+1) \end{aligned}$$

следва, че

$$(m+1)n=mn+n.$$

Сега комутативността на умножението следва от $m0=0=0m$ и това, че ако $mn=nm$, то

$$\begin{aligned} m(n+1) &= mn+m \\ &= nm+m \;=\; (n+1)m. \end{aligned}$$

□

2.3 Крайни множества

Лема 2.14. Нека n и m са естествени числа. Тогава $n \leq m$ тогава и само тогава, когато съществува инекция $f : n \to m$.

Доказателство. Ако $n \leq m$, очевидно съществува инекция от n в m. За обратната посока ще използваме индукция по n. Твърдението е очевидно за числото 0. Нека сега n е такова, че за произволно естествено m', ако съществува инекция от n в m', то $n \leq m'$. Трябвя да докажем, че ако съществува инекция от $n+1$ в m, то $n+1 \leq m$. Нека m е естествено и $g : n+1 \to m$ е инекция. Очевидно $m \neq 0$ и значи $m = m' + 1$ (т.е. $m = m' \cup \{m'\}$) за някое естествено m'. Ако $g(k) \neq m'$ за всяко $k < n$, то $f : n \to m'$, дефинирана чрез $f(x) = g(x)$ за $x < n$ е инекция, откъдето $n \leq m'$ и значи $n+1 \leq m$. От друга страна, ако $m' = g(k)$ за някое $k < n$, то $g(n) = s < m'$ (тъй като g е инекция) и тогава функцията $f : n \to m'$, дефинирана чрез

$$f(x) = \begin{cases} g(x) & x \neq k, \\ s & x = k \end{cases}$$

е инекция, откъдето $n \leq m'$ и значи $n + 1 \leq m$.

□

Следствие 2.15. Нека m и n са естествени числа, за които съществува биекция от n върху m. Тогава $m = n$.

Доказателство. Нека $f : n \to m$ е биекция. Тогава f е инекция и следователно $n \leq m$. От друга страна $f^{-1} : m \to n$ също е биекция, а значи и инекция, откъдето $m \leq n$. Следователно $m = n$.

□

Ще казваме, че множеството A е *n-елементно* ($n \in \mathbb{N}$), ако съществува биекция $f : n \to A$. Да забележим, че ако едно множество A е едновременно m-елементно и n-елементно, то $m = n$. Наистина, ако $g : n \to A$ и $h : m \to A$ са биекции, то $h^{-1} \circ g : n \to m$ е биекция и следователно $m = n$. Това ни дава право да пишем $|A| = n$, ако A е n-елементно множество.

Твърдение 2.16. Нека $|A| = n$ и $|B| = m$. Тогава

(i) $n \leq m$ тогова и само тогава, когато съществува инекция от A в B;

(ii) $n = m$ тогава и само тогава, когато съществува биекция на A върху B.

Доказателство. Нека $f : n \to A$ и $g : m \to B$ са биекции.

(i) Нека първо $n \leq m$. Тогава $n \subseteq m$ и значи $g \circ f^{-1} : A \to B$ е инекция. Обратно, нека $h : A \to B$ е инекция. Тогава $g^{-1} \circ h \circ f : n \to m$ е инекция и следователно $n \leq m$.

(ii) Нека първо $n = m$. Тогава $g \circ f^{-1}$ е биекция от A върху B. Обратно, нека $h : A \to B$ е биекция. Тогава $g^{-1} \circ h \circ f : n \to m$ е биекция и следователно $n = m$.

□

Следствие 2.17 (Принцип на Дирихле)**.** Нека $|A| = n$, $|B| = m$ и $m < n$. Тогава не съществува инекция от A в B.

Ще казваме, че множеството A е *крайно*, ако $|A| = n$ за някое естествено n.

Лема 2.18. Нека A е крайно и $b \notin A$. Тогава $|A \cup \{b\}| = |A| + 1$

Доказателство. Нека $f : n \to A$ е биекция. Тогава $f' = f \cup (n, b)$ е биекция от $n + 1$ върху $A \cup \{b\}$. □

Теорема 2.19. Нека A крайно и $B \subseteq A$. Тогава B е крайно и $|B| \leq |A|$. При това $|B| = |A|$ тогава и само тогава, когато $B = A$.

Доказателство. Ще проведем индукция по $|A|$. Ако $|A| = 0$, то $A = \emptyset$, откъдето $B = \emptyset$ и значи $|B| = 0 = |A|$.

Нека сега n е такова, че всяко подмножество на n-елементно множество е k-елементно за някое $k \leq n$, като $k = n$ тогава и само тогава, когато двете множества съвпадат. Нека A е $n + 1$-елементно множество и $B \subseteq A$. Ако $B = A$, то B е $n + 1$-елементно множество. Нека сега $B \neq A$, $b \in A \setminus B$ и $A' = A \setminus \{b\}$. Тогава $A = A' \cup \{b\}$, $b \notin A'$ и значи

$$n + 1 = |A| = |A'| + 1$$

, т.е. $|A'| = n$. От друга страна $B \subseteq A'$ и следователно $|B| = k$ за някое $k \leq n < n + 1$.

□

Твърдение 2.20. Нека A е крайно множество. Тогава

(i) ако съществува биекция от A върху C, то C е крайно и $|C| = |A|$;

(ii) ако съществува инекция от C в A, то C е крайно и $|C| \leq |A|$.

Доказателство. (i) Нека $|A| = n$ и $g : n \to A$ е биекция. Тогава, ако $f : C \to A$ е биекция, то $f^{-1} \circ g : n \to C$ е биекция и значи $|C| = n = |A|$.

(ii) Нека $f : C \to A$ е инекция. Нека $B = \text{range}(f)$. Тогава $B \subseteq A$ и значи B е крайно и $|B| \leq |A|$. Но $f : C \to B$ е биекция и следоветлно (съгласно (i)) $|B| = |C| \leq |A|$.

□

Теорема 2.21. Нека A е крайно множество и $f : A \to A$. Тогава, ако f е инекция, то f е биекция.

Доказателство. Нека $B = f(A)$. Тогава $f : A \to B$ е биекция и значи $|A| = |B|$. Оттук и $B \subseteq A$ получаваме $B = A$, откъдето f е сюрекция върху A и значи е биекция.

□

2.4 Основни комбинаторни принципи

Теорема 2.22 (Принцип за сумата)**.** Нека A и B са крайни и $A \cap B = \emptyset$. Тогава $|A \cup B| = |A| + |B|$.

Доказателство. Нека A и B са крайни множества, такива че $A \cap B| = 0$ и $|A| = m$ и $|B| = n$. Нека $f : n \to A$ и $g : m \to B$ са биекции. Да разгледаме $h : m + n \to A \cup B$, дефинирана чрез

$$h(i) = \begin{cases} f(i), & i < n \\ g(i-n) & i > n \end{cases}.$$

Функцията h е сюрекция, тъй като ако $x \in A$, то $x = f(i)$ за някое $i < m$ и значи $x = h(i)$, а ако $x \in B$, то $x = g(j)$ за някое $j < m$ и значи $x = h(n + j)$.

Ще докажем, че h е инекция, откъдето следва, че h е биекция и $|A \cup B| = |A| + |B|$. Нека $h(i) = x = h(i')$. Ако $i < n$, то тогава $x = h(i) = f(i) \in A$. Оттук и $A \cap B = \emptyset$ следва, че $h(i') = f(i')$, откъдето $i = i'$ (защото f е инекция). Ако $i > n$ разсъждаваме аналогично.

□

Следствие 2.23 (Принцип за разликата)**.** Нека A е крайно и $B \subseteq A$. Тогава $|A \setminus B| = |A| - |B|$.

Доказателство. Имаме $A = (A \setminus B) \cup B$ и $(A \setminus B) \cap B = \emptyset$, откъдето $|A| = |A \setminus B| + |B|$ и значи $|A \setminus B| = |A| - |B|$.

Следствие 2.24 (Принцип за разбиването)**.** Нека A_1, A_2,..., A_n $(n \geq 1)$ са крайни множества, такива че $A_i \cap A_j = \emptyset$ за $i \neq j$. Тогава

$$|A_1 \cup A_2 \cup \cdots \cup A_n| = |A_1| + |A_2| + \cdots + |A_n|,$$

или по-кратко

$$|\bigcup_{1 \leq i \leq n} A_i| = \sum_{i=1}^{n} |A_i|.$$

Доказателство. Доказателството се извършва с индукция по $n \geq 1$, като се използва горното твърдение и това, че от $A_i \neq A_j$ за $i \neq j$ следва, че множеството $A_1 \cup A_2 \cup \cdots \cup A_{n-1}$ и множеството A_n нямат общи елементи.

□

Следствие 2.25 (Принцип за обединението)**.** Нека A и B са крайни. Тогава $|A \cup B| = |A| + |B| - |A \cap B|$.

Доказателство. В сила е $A \cup B = A \cup (B \setminus (A \cap B))$ и $A \cap (B \setminus (A \cap B)) = \emptyset$. Тъй като B е крайно, то $B \setminus (A \cap B)$ е крайно. Тогава

$$|A \cup B| = |A| + |B \setminus (A \cap B)| = |A| + |B| - |A \cap B|.$$

□

Нека B е произволно крайно множество и $X \subseteq B$. Тогава характеристичната функция $\chi_X : B \to \{0, 1\}$ също е крайна и

$$|X| = \sum_{x \in B} \chi(x).$$

Нека забележим още, че

$$\chi_{B \setminus X}(x) = 1 - \chi_X(x)$$

и ако $X_1, \ldots, X_k \subseteq B$, то

$$\chi_{X_1 \cap \cdots \cap X_k} = \prod_{1 \leq i \leq k} \chi_{X_i}.$$

Теорема 2.26 (Принцип за включването и изключването)**.** Нека A_1, A_2,..., A_n $(n \geq 2)$ са крайни множества. Тогава $A_1 \cup A_2 \cup \cdots A_n$ е крайно и

$$|\bigcup_{1\leq i\leq n} A_i| = \sum_{1\leq k\leq n} (-1)^{k+1} \sum_{1\leq i_1<i_2<\cdots<i_k\leq n} |A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_k}|.$$

Доказателство. Нека $B = A_1 \cup A_2 \cup \cdots \cup A_n$. Тъй като A_1, A_2, ..., A_n са крайни, то B също е крайно. Да забележим, че

$$\emptyset = B \setminus B = B \setminus \bigcup_{1\leq i\leq k} A_i = \bigcap_{1\leq i\leq n} (B \setminus A_i). \tag{2.1}$$

Нека

$$\chi_i : B \to \{0, 1, \}$$

е характеристичната функция на множеството A_i. Тогава, съгласно (2.1), за всяко $x \in B$ имаме

$$\begin{aligned} 0 = \prod_{1\leq i\leq n} (1 - \chi_i(x)) &= 1 + \sum_{1\leq k\leq n} (-1)^k \sum_{1\leq i_1<i_2<\cdots<i_k\leq n} \chi_{i_1}(x)\chi_{i_2}(x)\cdots\chi_{i_k}(x) \\ &= 1 + \sum_{1\leq k\leq n} (-1)^k \sum_{1\leq i_1<i_2<\cdots<i_k\leq n} \chi_{A_{i_1}\cap A_{i_2}\cap\cdots\cap A_{i_k}}(x) \end{aligned}$$

и значи

$$1 = \sum_{1\leq k\leq n} (-1)^{k+1} \sum_{1\leq i_1<i_2<\cdots<i_k\leq n} \chi_{A_{i_1}\cap A_{i_2}\cap\cdots\cap A_{i_k}}(x).$$

Оттук

$$\begin{aligned} |B| = \sum_{x\in B} 1 &= \sum_{x\in B} \sum_{1\leq k\leq n} (-1)^{k+1} \sum_{1\leq i_1<i_2<\cdots<i_k\leq n} \chi_{A_{i_1}\cap A_{i_2}\cap\cdots\cap A_{i_k}}(x) \\ &= \sum_{1\leq k\leq n} (-1)^{k+1} \sum_{1\leq i_1<i_2<\cdots<i_k\leq n} \sum_{x\in B} \chi_{A_{i_1}\cap A_{i_2}\cap\cdots\cap A_{i_k}}(x) \\ &= \sum_{1\leq k\leq n} (-1)^{k+1} \sum_{1\leq i_1<i_2<\cdots<i_k\leq n} |A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_k}| \end{aligned}$$

□

Ще казваме, че α е *крайна редица* с елементи от B, ако $\alpha : n \to B$ за някое $n \in \mathbb{N}$. Множеството от всички крайни редици с елементи от B ще бележим с $B^{<\mathbb{N}}$. Ще казваме, че редицата $\alpha \in B^{<\mathbb{N}}$ има дължина n и ще пишем $\mathrm{lh}(\alpha) = n$ (и още $|\alpha| = n$), ако $\mathrm{dom}(\alpha) = n$.

Ако $\alpha \in B^{<\mathbb{N}}$ и $b \in B$, то с

$$\alpha \cdot b$$

(или още $\alpha^* b$ и $\alpha^\frown b$) ще означаваме

$$\alpha \cdot b = \alpha \cup (\mathrm{lh}(\alpha), b).$$

Ясно е, че $\alpha \cdot b \in B^{<\mathbb{N}}$, като $\mathrm{lh}(\alpha \cdot b) = S(\mathrm{lh}(\alpha))$. При това $\alpha(x) = (\alpha \cdot b)(x)$ за всяко $x < \mathrm{lh}(\alpha)$, а $(\alpha \cdot b)(\mathrm{lh}(\alpha)) = b$.

Ако α е крайна редица с $\mathrm{lh}(\alpha) = n$, а $i < n$, вместо $\alpha(i)$ пишем α_i, а за редицата α използваме записа

$$\alpha = \alpha_0, \alpha_1, \ldots, \alpha_{n-1}.$$

Теорема 2.27 (Принцип за умножението)**.** Нека $k \geq 1$ е естествено число. Нека n_1, n_2,..., n_k са естествени числа, а φ е свойство (за редици), такова че за всяка редица α с $\mathrm{lh}(\alpha) < k$, която има свойството φ, е в сила

$$|\{a \mid \varphi(\alpha \cdot a)\}| = n_{\mathrm{lh}(\alpha)+1}.$$

Тогава, ако от $\varphi(\alpha \cdot a)$ следва $\varphi(\alpha)$, то

$$|\{\alpha \mid \mathrm{lh}(\alpha) = k \text{ и } \varphi(\alpha)\}| = n_1 n_2 \ldots n_k.$$

Доказателство. Ще проведем доказателството с индукция по k. При $k = 1$ твърдението е очевидно. Нека сега k е такова, че твърдението е вярно за $k-1$ и нека свойството φ и числата n_1, n_2,..., n_k удовлетворяват условието на теоремата. Нека означим

$$\begin{aligned} A &= \{\alpha \mid \varphi(\alpha) \text{ и } \mathrm{lh}(\alpha) = k\} \\ A' &= \{\alpha \mid \varphi(\alpha) \text{ и } \mathrm{lh}(\alpha) = k-1\} \end{aligned}$$

Съгласно индукционното предположение $|A'| = n_1 n_2 \dots n_{k-1}$. Нека за $\alpha \in A'$ означим

$$A_\alpha = \{\alpha \cdot a \mid \varphi(\alpha \cdot a)\}.$$

Съгласно условието на теоремата $|A_\alpha| = n_k$. От друга страна, съгласно условието за φ имаме

$$\alpha \cdot a \in A \Longrightarrow \alpha \in A'.$$

и значи

$$A = \bigcup_{\alpha \in A'} A_\alpha.$$

Но

$$A_\alpha \cap A_\beta = \emptyset \text{ при } \alpha \neq \beta,$$

откъдето, съгласно принципа за разбиването, получаваме

$$|A| = \sum_{\alpha \in A'} |A_\alpha| = (n_1 n_2 \dots n_{k-1}) n_k.$$

□

Принципът за умножението обикновенно се прилага по следния начин. Нека S е крайно множество от редици с дължина k. Формулираме схема, при която последователно строим редици

$$\begin{aligned} &\alpha_0 \\ &\alpha_0, \alpha_1 \\ &\vdots \\ &\alpha_0, \alpha_1, \dots, \alpha_{k-1} \end{aligned}$$

като

1. всеки елемент на S може да се получи по единствен начин по схемата.
2. съществуват числа n_1, ..., n_k, такива че за всяко $i < k$ и за всяка редица $\alpha_1, \alpha_2, \dots, \alpha_i$, построена по схемата, имаме точно n_i избора на елемент α_{i+1}.

Тогава

$$|S| = n_1 n_2 \dots n_k.$$

2.5 Основни комбинаторни конфигурации

Твърдение 2.28. Нека A_1, A_2,..., A_k са крайни множества. Тогава

$$|A_1 \times A_2 \times \ldots \times A_k| = |A_1||A_2| \cdots |A_k|.$$

Доказателство. Нека $|A_i| = n_i$ за $1 \leq i \leq k$. Елементите на $A_1 \times A_2 \times \ldots \times A_k$ съответстват еднозначно на редици с дължина k, в които i-тият член е елемент на A_i за $1 \leq i \leq k$. Следователно елементите на $A_1 \times A_2 \times \ldots \times A_k$ се получават еднозначно по следната схема: избираме последователно по един елемент на A_1, A_2, ..., A_k. Оттук и принципа за умножението

$$|A_1 \times A_2 \times \ldots \times A_k| = n_1 n_2 \cdots n_k.$$

□

Ако в горното твърдение $A_1 = A_2 = \ldots = A_k = A$, имаме

$$|A^k| = |A|^k.$$

Множеството A^k съответства на множеството от всички редици с дължина k от елементи на A, т.е. всички изображение от k в A. Всяка такава редица се нарича вариация с повторения от k-ти клас с елементи от A. Така, ако A е n-елементно множество то броят на вариациите с повторения от k-ти клас с елементи от A е точно n^k. Да отбележим, че броят на вариациите с повторения от k-ти клас на n-елементно множество, отразява точно броя на начините, по които можем да изберем последователно k елемента от n елементно множество, отчитайки реда на избора и позволявайки един и същи елемент да бъде избран повече от веднъж.

Дефинираме функцията факториел по следната рекурсивна схема:

$$\begin{aligned} 0! &= 1 \\ (n+1)! &= n!(n+1) \end{aligned}$$

При $n > 0$ числото $n!$ представлява произведението на числата 1, 2, ..., n.

Твърдение 2.29. Нека A е n-елементно множество и нека $0 < k \leq n$. Тогава броят на инекциите от k в A е

$$n(n-1)\cdots(n-k+1) = \frac{n!}{(n-k)!}.$$

Доказателство. Всяка инекция от k в A е редица $\alpha_0, \alpha_1, \ldots, \alpha_{k-1}$, за която $\alpha_i \neq \alpha_j$ за $0 \leq i < j < k$. Всяка такава редица можем да построим еднозначно по следната схема

1) Избираме произволно

$$\alpha_0 \in A.$$

2) Избираме произволно $\alpha_1 \in A$, различно от α_0, т.е.

$$\alpha_1 \in A \setminus \{\alpha_0\}.$$

3) Избираме произволно $\alpha_2 \in A$, различно от α_0,α_1, т.е.

$$\alpha_2 \in A \setminus \{\alpha_0, \alpha_1\}.$$

$\vdots$

k) Избираме произволно $\alpha_{k-1} \in A$, различно от α_0,..., α_{k-2}, т.е.

$$\alpha_{k-1} \in A \setminus \{\alpha_0, \ldots, \alpha_{k-2}\}$$

Така (независимо от избора на предходните стъпки) на стъпка i, $1 \leq i \leq k$ можем да изберем α_i по $n-i+1$ начина. Оттук и принципа за умножението получаваме, че броят на всички такива редици е

$$n(n-1)\cdots(n-k+1).$$

□

Редиците с дължина k от различни елементи на A наричаме вариации (без повторения) от k-ти клас с елементи на A. Следователно броят на вариациите от k-ти клас на едно n елементно множество ($1 \leq k \leq n$) е $\frac{n!}{(n-k)!}$. Да отбележим, че този брой, отразява точно броя на начините, по които можем да изберем последователно k елемента от n елементно множество, отчитайки реда на избора.

При $k = n$ горното твърдение приема вида

Твърдение 2.30. Нека A е n-елементно множество. Тогава броят на биекциите от n върху A е $n!$.

При n-елементно множество A, биекциите от n върху A наричаме пермутации на елементите на A. Така всяка пермутация на елементите на A е всъщност подреждане на елементите на A. Съгласно горното твърдение, броят на пермутациите на едно n-елементно множество е $n!$.

Ако n и k са естествени числа и $k < n$, дефинираме биномния коефициент $\binom{n}{k}$ чрез

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} = \frac{n(n-1)\cdots(n-k+1)}{k!}.$$

Твърдение 2.31. Нека A е n-елементно множество и $k \leq n$. Тогава броят на k-елементните подмножества на A е $\binom{n}{k}$

Доказателство. Нека означим с m броят на k-елементните подмножества на A. За да построим еднозначно всички инекции от k в A (т.е. всички вариации k-ти клас с елементи от A), можем да приложим следната схема:

1) Избираме k-елементно подмножество на A.

2) Избираме биекция от k в избраното в предната стъпка множество.

Независимо от избора на множество в първата стъпка, втората стъпка можем да осъществим по $k!$ начина. Оттук, това че броят на инекциите от k в A е $\frac{n!}{(n-k)!}$ и принципа за умножението имаме

$$\frac{n!}{(n-k)!} = m.k!,$$

откъдето

$$m = \frac{n!}{(n-k)!k!} = \binom{n}{k}.$$

□

Всяко k-елементно подмножество на едно n-елементно множество съответства на това да изберем (без повторения) k елемента от множеството. Всеки такъв избор се нарича комбинация k-ти клас. Следователно от n-елемнтно множество можем да изберем k различни елемента по $\binom{n}{k}$ начина.

Твърдение 2.32 (Формула на Нютон)**.** Нека x и y са елементи на множество, в което са дефинирани операции събиране и умножение, които са асоциативни, комутативни и умножението е дистрибутивно относно събирането. Тогава за всяко естествено $n > 0$

$$(x+y)^n = \sum_{0 \leq k \leq n} \binom{n}{k} x^{n-k}y^k = \binom{n}{0}x^n + \binom{n}{1}x^{n-1}y + \cdots + \binom{n}{n}y^n.$$

Доказателство. Имаме

$$(x+y)^n = \underbrace{(x+y)(x+y)\cdots(x+y)}_{n}.$$

Разкривайки скобите (и възползвайки се от комутативността и асоциативността на умножението) получаваме сума на едночлени от вида $x^{n-k}y^k$ за $0 \leq k \leq n$. При това всеки такъв едночлен се получава еднозначно избирайки k от множителите $(x+y)$, от които взимаме y (а останалите множители взимаме x). Следователно всеки едночлен $x^{n-k}y^k$ участва $\binom{n}{k}$ пъти в получената сума. Оттук и асоциативността и комутативността на събирането получаваме

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

□

Твърдение 2.33. Броят на всички подмножества на една n-елементно множество е 2^n.

Доказателство. Нека A е n-елементно множество и нека с P_k, $0 \leq k \leq n$, означим множеството от всички k-елементни подмножества на A. Тогава $|P_k| = \binom{n}{k}$. При това $P_i \cap P_j = \emptyset$ за $i \neq j$ и $\mathcal{P}(A) = P_0 \cup P_1 \cup \cdots P_n$. Оттук, принципа за разбиването и формулата на Нютон получаваме

$$\begin{aligned} |\mathcal{P}(A)| &= |P_0| + |P_1| + \cdots + P_n \\ &= \binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{n} \\ &= \binom{n}{0}1^n + \binom{n}{1}1^{n-1}1 + \cdots + \binom{n}{n}1^n \\ &= (1+1)^n = 2^n \end{aligned}$$

□

2.6 Равномощни множества

Ще казваме, че A и B са *равномощни* (имат равен брой елементи) и ще пишем $||A|| = ||B||$, ако съществува биекция от A върху B.

От свойствата на биекциите получаваме следните свойства на равномощността:

1. $||A|| = ||A||$;
2. Ако $||A|| = ||B||$, то $||B|| = ||A||$;
3. Ако $||A|| = ||B||$ и $||B|| = ||C||$, то $||A|| = ||C||$.

Ще казваме, че мощността A е по-малка или равна на мощността на B (броят на елементите на A е по-малък или равен на броя на елементите на B) и ще пишем $||A|| \leq ||B||$, ако съществува инекция от A в B.

От свойствата на инекциите получаваме следните свойства:

1. $||A|| \leq ||A||$;
2. Ако $||A|| \leq ||B||$ и $||B|| \leq ||C||$, то $||A|| \leq ||C||$.

Оказва се, че сравняването на мощности на множества е и антисиметрично, т.е. от $||A|| \leq ||B||$ и $||B|| \leq ||A||$ следва $||A|| = ||B||$. Това твърдение не е тривиално и за това ще го формулираме, като отделна теорема.

Теорема 2.34 (Кантор-Шрьодер-Бернщайн)**.** Ако $||A|| \leq ||B||$ и $||B|| \leq ||A||$, то $||A|| = ||B||$.

Първо ще докажем следната лема.

Лема 2.35. Нека $C \subseteq A$ и нека $||A|| \leq ||C||$. Тогава $||A|| = ||C||$.

Доказателство. Нека $C \subseteq A$ и нека $f : A \to C$ е инекция. Да разгледаме редиците от множества D_0, D_1,..., D_n, ..., дефинирани с

$$D_0 = A \setminus C \text{ и } D_{n+1} = f(D_n) \text{ за } n \geq 0$$

Да разгледаме изображението $h : A \to A$, дефинирано чрез правилото

$$h(x) = \begin{cases} f(x), & x \in D_n \text{ за някое } n \in \mathbb{N}, \\ x, & \text{иначе.} \end{cases}$$

Ще докажем, че h е биекция от A върху C. Нека първо да видим, че $h(x) \in C$ за всяко $x \in A$. Наистина, ако $x \in D_n$ за някое n, то $h(x) = f(x) \in C$. В противен случай $h(x) = x$, като в сила е $x \notin D_0$, откъдето $x \in C$.

Нека сега докажем, че h е инекция. Да фиксираме $x, y \in A$, такива че $x \neq y$. Ако $x \in D_n$ и $y \in D_k$ за някои n и k, то $h(x) = f(x) \neq f(y) = h(y)$. Ако $x, y \notin D_n$ за всяко $n \in \mathbb{N}$, то $h(x) = x \neq y = h(y)$. Нека сега $x \in D_n$ за някое n и $y \notin D_k$ за всяко естествено k. Тогава $h(x) = f(x) \in D_{n+1}$, а $h(y) = y \notin D_{n+1}$ и следователно $h(x) \neq h(y)$.

Остава да докажем, че h е сюрекция. За целта нека фисдираме $z \in C$. Възможни са два случая. Нека първо $z \notin D_n$ за всяко естествено n. Тогава $z = h(z)$. Нека сега $z \in D_n$ за някое естествено n. Тъй като $D_0 \cap C = \emptyset$, имаме $n > 0$ и следоветлно $z = g(x)$ за някое $x \in D_{n-1}$. Тогава $z = h(x)$.

□

Доказателство. (Теорема на Кантор-Шрьодер-Бернщайн) Нека $g_1 : A \to B$ и $g_2 : B \to A$ са инекции. Нека означим с C множеството $C = g_2(B)$. Ясно е, че $C \subseteq A$ и че $||C|| = ||B||$. Функцията $g_2 \circ g_1 : A \to C$ е инекция и значи $||A|| \leq ||C||$. Тогава съгласно лемата $||A|| = ||C||$ и значи $||A|| = ||B||$.

□

2.7 Изброими и неизброими множества

Ще казваме, че множеството A е *безкрайно*, ако A не е крайно, т.е. за всяко естествено n не съществува биекция между A и n. Ще казваме, че множеството A е *изброимо*, ако съществува биекция между A и $\mathbb{N}$. Ясно е, че изброимите множества са безкрайни.

Примери за изброими множества:

1) $\mathbb{N}$;

2) Нека с $2\mathbb{N}$ означим множеството от четните естествени числа, т.е.

$$2\mathbb{N} = \{2x \mid x \in \mathbb{N}\}.$$

$2\mathbb{N}$ е изброимо, тъй като функцията $f : \mathbb{N} \to 2\mathbb{N}$, дефинирано $f(x) = 2x$ е биекция;

3) Нека с $2\mathbb{N}+1$ означим множеството от нечетните естествени числа , т.е.

$$2\mathbb{N}+1 = \{2x+1 \mid x \in \mathbb{N}\}.$$

$2\mathbb{N}+1$ е изброимо, тъй като функцията $f : \mathbb{N} \to 2\mathbb{N}+1$, дефинирано $f(x) = 2x+1$ е биекция;

4) $\mathbb{Z}$. Изображението $f : \mathbb{N} \to \mathbb{Z}$, дефинирано с

$$f(n) = \begin{cases} \dfrac{n}{2}, & \text{ако } n \text{ е четно,} \\ -\dfrac{n+1}{2}, & \text{ако } n \text{ е нечетно} \end{cases}$$

е биекция;

5) $\mathbb{N} \times \mathbb{N}$. Изображението $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, дефинирано с

$$f(x, y) = 2^x(2y+1) - 1$$

е биекция.

6) $\mathbb{Q}$.

Твърдение 2.36. Всяко безкрайно подмножество на $\mathbb{N}$ е изброимо.

Доказателство. Нека $A \subseteq \mathbb{N}$ е безкрайно. Да разгледаме функцията $f : \mathbb{N} \to A$, дефинирана чрез рекурсията

$$f(n) = \mu a\,[a \in A \setminus \{f(i) \mid i < n\}],$$

където за всяко непразно множество от естествени числа X с $\mu a\,[a \in X]$ означаме най-малкият елемент на A. Функцията f е коректно дефинирана, тъй като

1. най-малкият елемент на всяко непразно множество от естествени числа е еднозначо определен
2. за всяко естествено n, множесвото $\{f(i) | i < n\}$ е най-много n-елементно и значи $A \setminus \{f(i) \mid i < n\}$ е непразно (тъй като A е безкрайно).

Функцията f е инекция, тъй като съгласно дефиницята $f(n) \notin \{f(i) \mid i < n\}$, т.е. $f(n) \neq f(i)$ за $i < n$.[2] Следонателно $||\mathbb{N}|| \leq ||A||$. Оттук, $A \subseteq \mathbb{N}$ и теоремата на Кантор-Шройдер-Бернщайн[3] следва, че A е равномощно на $\mathbb{N}$ и значи A е изброимо.

□

Следствие 2.37. Ако A е безкрайно и $||A|| \leq ||\mathbb{N}||$, то A е изброимо.

Доказателство. Нека $f : A \to \mathbb{N}$ е инекция и $B = f(A)$. Тогава B е безкрайно подмножество на $\mathbb{N}$ и значи B е изброимо. Но $f : A \to B$ е биекция и следователно A също е изброимо. □

Твърдение 2.38. Обединение на две, а значи и на краен брой, изброими множества е изброимо.

[2] С индукция по n можем да докажем, че $f(i) < f(n)$ за $i < n$.

[3] Всъщност, f е сюрекция, тъй като $f(0) < f(1) < \cdots < f(n) < \cdots$ е изреждане (номерация) в нарастващ ред на елементите на A.

Доказателство. Нека A и B са изброими и нека $f : A \to \mathbb{N}$ и $g : B \to \mathbb{N}$ са биекции. Тогава функцията $h : A \cup B \to N$, дефинирана чрез

$$h(x) = \begin{cases} 2f(x), & \text{ако } x \in A \\ 2g(x) + 1, & \text{ако } x \in B \setminus A \end{cases}$$

е инекция и значи $A \cup B$ е изброимо (съгласно предното следствие).

□

Твърдение 2.39. Множеството $\mathbb{Q}$ на рационалните числа е изброимо.

Доказателство. Да разгледаме множеството $\mathbb{Q}^+ \cup \{0\}$ от неотрицателните рационални числа. Изображението $f : \mathbb{Q}^+ \cup \{0\} \to \mathbb{N} \times \mathbb{N}$, дефинирано чрез

$$f(x) = \begin{cases} (0,0), & x = 0; \\ (p,q), & x = \dfrac{p}{q},\ p \text{ и } q \text{ са взаимно прости} \end{cases}$$

е инективно и следователно $\mathbb{Q}^+ \cup \{0\}$, а също и $\mathbb{Q}^+$ и $\mathbb{Q}^-$ са изброими. Следователно $\mathbb{Q}$ е изброимо.

□

Ще казваме, че безкрайното множество A е *неизброимо*, ако A не е изброимо. Най-простият пример за неизброимо множество е $2^{\mathbb{N}}$.

Теорема 2.40. Множеството $2^{\mathbb{N}}$ е неизброимо.

Доказателство. Ясно е, че $2^{\mathbb{N}}$ е безкрайно множество. Да допуснем, че $2^{\mathbb{N}}$ е изброимо и нека $f : \mathbb{N} \to 2^{\mathbb{N}}$ е биекция. Нека за простота на означения, $\phi_n = f(n)$ за $n \in \mathbb{N}$ ($\phi_n \in 2^{\mathbb{N}}$ и значи $\phi_n : \mathbb{N} \to \{0,1\}$). Да разгледаме функцията $\phi : \mathbb{N} \to \{0,1\}$, дефинирана с

$$\phi(k) = 1 - \phi_k(k).$$

Тъй като f е биекция, в сила е $\phi = \phi_n$ за някое n. Тогава

$$\phi_n(n) = \phi(n) = 1 - \phi_n(n),$$

което е невъзможно. Следоветлно не съществува биекция между $\mathbb{N}$ и $2^{\mathbb{N}}$ и значи $2^{\mathbb{N}}$ е неизброимо.

□

Горното доказателство е известно като *диагонален метод на Кантор* и можем да илюстрираме по следния начин. Нека първо да забележим, че функциите от $\mathbb{N}$ в A, т.е. елементите на $A^{\mathbb{N}}$, са точно безкрайните редици от елементи на A. Така $2^{\mathbb{N}}$ е множеството от всички безкрайни редици от 0 и 1. Допускането за съществуване на биекция от $\mathbb{N}$ върху $2^{\mathbb{N}}$ означава, че можем да подредим безкрайните редици от 0 и 1 в редица α_0, α_1, α_2, …, α_n, …, като за всяко естествено k, $\alpha_k = \{a_k i\}_{i=0}^{\infty}$. Така получаваме следната безкрайна матрица:

$$\begin{array}{lccccccc}
\alpha_0 = & a_{00}, & a_{01}, & a_{02}, & \ldots, & a_{0n}, & \ldots \\
\alpha_1 = & a_{10}, & a_{11}, & a_{12}, & \ldots, & a_{1n}, & \ldots \\
\alpha_2 = & a_{20}, & a_{21}, & a_{22}, & \ldots, & a_{2n}, & \ldots \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\alpha_n = & a_{n0}, & a_{n1}, & a_{n2}, & \ldots, & a_{nn}, & \ldots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}$$

Нека сега да разгледаме редицата $\alpha = \{a_i\}_{i=0}^{\infty}$, дефинирана чрез $a_i = 1 - a_{ii}$, т.е. α е точно противоположната редица на диагоналната редица $\{a_{ii}\}_{i=0}^{\infty}$. Така за всяко естествено i

$$a_i = 1 \iff a_{ii} = 0$$

и следователно редицата α не съвпада с нито една от редиците α_0, α_1, α_2, …, α_n, …, което пък противоречи на това, че редицата α_0, α_1, α_2, …, α_n, …, изчерпва всички елементи на $2^{\mathbb{N}}$.

Множества, равномощни с $2^{\mathbb{N}}$, наричаме множества с мощността на континуума. Да напомним, че съществува естествен изоморфизъм[4] между множеството от характеристичните функции 2^A, дефинирани в A, и множеството $\mathcal{P}(A)$ от подмножества на A. Следователно $\mathcal{P}(\mathbb{N})$ също е с мощността на континуума. Нека още да отбележим, че ако A и B са равномощни, то $\mathcal{P}(A)$ и $\mathcal{P}(B)$ също са равномощни. Наистина, ако $g : A \to B$ е биекция, то $G : \mathcal{P}(A) \to \mathcal{P}(B)$, дефинирано чрез

$$G(X) = \{g(x) \mid x \in X\}$$

също е биекция. Следователно $2^{\mathbb{Q}}$ и $\mathcal{P}(\mathbb{Q})$ също са с мощността на континуума.

Теорема 2.41. Множествата $\mathbb{R}$ и $\mathbb{C}$ са с мощността на континуума.

Доказателство. Първо ще докажем, че $\mathbb{R}$ е с мощността на континуума. Всяко реално число r се определя еднозначно от рационалните числа по-малки от него и следователно изображението $f : \mathbb{R} \to \mathcal{P}(\mathbb{Q})$, дефинирано чрез

$$f(r) = \{q \in \mathbb{Q} \mid q < r\}$$

е инекция. Следователно $||\mathbb{R}|| \leq ||2^{\mathbb{N}}||$.

За обратната посока отново ще се възползваме от това, че елементите на $2^{\mathbb{N}}$ са безкрайни редици от 0 и 1. Ще казваме, че една редица $\alpha \in 2^{\mathbb{N}}$ е финитна, ако само краен брой от членовете ѝ са различни от нула[5]. Нека $\mathcal{F}$ е множеството от всички финитни редици от $2^{\mathbb{N}}$ и нека $\mathcal{P} = 2^{\mathbb{N}} \setminus \mathcal{F}$. Ще докажем, че съществуват инективни изображения от $\mathcal{P}$ в интервала $(0, 1]$ и от $\mathcal{F}$ в множеството от реалните числа (и по точно в множеството от естествените числа) по-големи от две, откъдето ще следва, че съществува инективно изображение от $2^{\mathbb{N}}$ в $\mathbb{R}$, което заедно с $||\mathbb{R}|| \leq ||2^{\mathbb{N}}||$ ще ни даде, че $\mathbb{R}$ е с мощността на континуума.

Изображението $g : \mathcal{P} \to \mathbb{R}$, дефинирано чрез

$$g(\alpha) = \sum_{n=0}^{\infty} \frac{1}{2^{n+1}} \alpha(n).$$

е инективно. Действително, нека $\alpha = \alpha_0, \ldots, \alpha_i, \ldots$ и $\beta = \beta_0, \ldots, \beta_i, \ldots$ са две редици от $\mathcal{P}$ и нека $\alpha \neq \beta$. Нека k_0 е първата позиция на различие на α и β, т.е. $\alpha_i = \beta_i$ за $i < k_0$ и $\alpha_{k_0} \neq \beta_{k_0}$, като например $\alpha_{k_0} = 1$, а $\beta_{k_0} = 0$. Тъй като α не е финитна, то $\alpha_n = 1$ за поне едно $n > k_0$ и следователно

$$\begin{aligned}
g(\alpha) = \sum_{n=0}^{\infty} \frac{1}{2^{n+1}} \alpha_n &= \sum_{n<k_0} \frac{1}{2^{n+1}} \alpha_n + \frac{1}{2^{k_0+1}} \alpha_{k_0} + \sum_{n>k_0} \frac{1}{2^{n+1}} \alpha_n \\
&> \sum_{n<k_0} \frac{1}{2^{n+1}} \alpha_n + \frac{1}{2^{k_0+1}} \\
&= \sum_{n<k_0} \frac{1}{2^{n+1}} \beta_n + \frac{1}{2^{k_0+1}} .0 + \sum_{n>k_0} \frac{1}{2^{n+1}} \\
&\geq \sum_{n<k_0} \frac{1}{2^{n+1}} \beta_n + \frac{1}{2^{k_0+1}} \beta_{k_0} + \sum_{n>k_0} \frac{1}{2^{n+1}} \beta_n \\
&= g(\beta).
\end{aligned}$$

и значи $g(\alpha) \neq g(\beta)$.

Нека сега разгледаме множеството $\mathcal{F}$ и нека $h : \mathcal{F} \to R$ е дефинирана чрез правилото

$$h(\alpha) = 2 + \sum_{n=0}^{\infty} 2^n \alpha(n).$$

Тъй като всяка редица $\alpha \in \mathcal{F}$ е финитна, то h е коректно дефинирана инективна функция[6]. При това $h(\alpha) \geq 2$, с което доказателството на $||\mathbb{R}|| = ||2^{\mathbb{N}}||$ е завършено.

За да докажем, че $||\mathbb{C}|| = ||2^{\mathbb{N}}||$, да припомним, че съгласно алтебричаната дефиниция, множеството $\mathbb{C}$ е всъщност множеството $\mathbb{R} \times \mathbb{R}$ и значи достатъчно е да докажем, че $||2^{\mathbb{N}} \times 2^{\mathbb{N}}|| = ||2^{\mathbb{N}}||$.

[4] На $B \subseteq A$ съпоставяме характеристичната функция $\chi_B : A \to \{0, 1\}$.

[5] Финитните редици, съответстват на характеристчните функции на крайните множества от естествени числа

[6] Нека α_k е последният ненулев член на редицата α. Тогава $\sum_{n=0}^{\infty} 2^n \alpha(n) = \alpha_0 + \alpha_1 2 + \alpha_2 2^2 + \cdots + \alpha_k 2^k$, което е числото с двоичен запис $\alpha_k \alpha_{k-1} \ldots \alpha_0$. В частност изображението, което на финитната двоична редица α съпоставя числото $\sum_{n=0}^{\infty} 2^n \alpha(n)$ е биективно изображение между $\mathcal{F}$ (а значи и между множеството на крайните множества от естествени числа) и $\mathbb{N}$.

Нека за $\alpha, \beta \in 2^{\mathbb{N}}$ с $\langle \alpha, \beta \rangle$ означим редицата

$$\alpha(0), \beta(0), \alpha(1), \beta(1), \ldots$$

Ясно е, че ако $(\alpha, \beta) \neq (\alpha', \beta')$, то $\langle \alpha, \beta \rangle \neq \langle \alpha', \beta' \rangle$. При това за всяко $\alpha \in 2^{\mathbb{N}}$

$$\alpha = \langle \alpha_{\text{четна}}, \alpha_{\text{нечетна}} \rangle,$$

където $\alpha_{\text{четна}}$ е редицата $\alpha(0), \alpha(2), \alpha(4), \ldots$, а $\alpha_{\text{нечетна}}$ е редицата $\alpha(1), \alpha(3), \alpha(5), \ldots$. Следователно функцията $\varphi : 2^{\mathbb{N}} \times 2^{\mathbb{N}} \to 2^{\mathbb{N}}$, действаща по правилото

$$\varphi(\alpha, \beta) = \langle \alpha, \beta \rangle$$

е биекция и следователно $\mathbb{C}$ е с мощността на континуума.

□

ГЛАВА 3

Релации на еквивалентност. Частични наредби

3.1 Видове релации

Ако R е бинарна релация и $(a,b) \in R$, ще казваме, че a е *в релация* R с b и ще пишем $a\,R\,b$. Ако a не е в релация R с b ще пишем $a \not R\, b$. Ако R е бинарна релация и $R \subseteq A \times A$, ще казваме, че R е *бинарна релация в* A.

Казваме, че бинарната релация R в A е *рефлексивна*, ако

$$aRa$$

за всяко $a \in A$. Казваме, че бинарната релация R в A е *ирефлексивна*, ако

$$a \not R\, a$$

за всяко $a \in A$.

Примери.

1. Релацията R_1 в $\mathbb{C}$, дефинирана чрез
$$z_1\, R_1 z_2 \iff |z_1| = |z_2|$$
е рефлексивна.

2. Релацията R_2 в $\mathbb{Z}$, дефинирана чрез
$$x_1\, R_2 x_2 \iff \exists y(y \neq 0 \;\&\; x_1^2 + y^2 = x_2^2)$$
е ирефлексивна.

3. Релацията R_3 в $\mathbb{Z}$, дефинирана чрез
$$x_1\, R_3 x_2 \iff \exists y(x_1 = -x_2 y^2)$$
не е нито рефлексивна (защото например $1 \not R_3\, 1$), нито ирефлексивна (защото например $0\, R_3\, 0$).

От дефиницията на рефлексивна и ирефлексивна релация директно следва следното твърдение:

Твърдение 3.1. Нека R е бинарна релация в A. Тогава

(i) R е рефлексивна в A тогава и само тогава, когато $\mathrm{id}_A \subseteq R$;

(ii) R е ирефлексивна в A тогава и само тогава, когато $\mathrm{id}_A \cap R = \emptyset$;

Казваме, че бинарната релация R в A е симетрична, ако за всеки избор на $a, b \in A$ е в сила

$$a\,R\,b \Rightarrow b\,R\,a.$$

Примери.

1. Релацията R_1 в $\mathbb{C}$, дефинирана чрез
$$z_1 \, R_1 \, z_2 \iff z_1^4 = z_2^4$$
е симетрична.

2. Релацията R_2 в $\mathbb{C}$, дефинирана чрез
$$z_1 \, R_2 \, z_2 \iff z_1^3 = z_2^4$$
не е симетрична, защото $1 \, R_2 \, -1$, но $-1 \not R_2 \, 1$

Твърдение 3.2. Една бинарна релация R е симетрична тогава и само тогава, когато $R = R^{-1}$.

Доказателство. Нека първо R е симетрична. Тогава
$$(x,y) \in R^{-1} \overset{def}{\iff} (y,x) \in R \overset{\text{сим.}}{\iff} (x,y) \in R$$
и значи $R = R^{-1}$.

Обратно, нека $R = R^{-1}$. Тогава, ако $x \, R \, y$, т.е. $(x,y) \in R$, то $(x,y) \in R^{-1}$, откъдето $(y,x) \in R$, т.е. $y \, R \, x$. Следователно R е симетрична.

□

Казваме, че релацията R в A е *антисиметрична*, ако за всеки избор на $a, b \in A$ е в сила
$$a \, R \, b \;\&\; b \, R \, a \Longrightarrow a = b.$$

Примери.

1. Релацията R_1 в $\mathbb{N}$, дефинирана чрез
$$n_1 \, R_1 \, n_2 \iff n_2 = x n_1 \text{ за някое } x \in \mathbb{N}$$
е антисиметрична.

2. Релацията R_2 в $\mathbb{N}$, дефинирана чрез
$$n_1 \, R_2 \, n_2 \iff n_1 + n_2 \text{ е четно}$$
не е антисиметрична, защото $3 \, R_2 \, 5$, $5 \, R_2 \, 3$, но $3 \neq 5$.

Твърдение 3.3. Една бинарна релация R в A е антисиметрична тогава и само тогава, когато $R \cap R^{-1} \subseteq \mathrm{id}_A$.

Доказателство. Нека първо R е антисиметрична. Нека $(a,b) \in R \cap R^{-1}$. Тогава $a \, R \, b$ (тъй като $(a,b) \in R$) и $b \, R \, a$ (тъй като $(a,b) \in R^{-1}$) и значи $a = b$, т.е. $(a,b) \in \mathrm{id}_A$. Следователно $R \cap R^{-1} \subseteq \mathrm{id}_A$.

Обратно, нека $R \cap R^{-1} \subseteq \mathrm{id}_A$. Тогава, ако $a \, R \, b$ и $b \, R \, a$, т.е. $(a,b), (b,a) \in R$, то $(a,b) \in R \cap R^{-1}$, откъдето $(a,b) \in \mathrm{id}_A$ и значи $a = b$. Следователно R е антисиметрична.

□

Казваме, че бинарната релация R в A е *транзитивна*, ако за всеки избор на $a, b, c \in A$ е в сила
$$a \, R \, b \;\&\; b \, R \, c \Longrightarrow a \, R \, c.$$

Примери.

1. Релацията R_1 в $\mathbb{N}$, дефинирана чрез
$$n_1 \, R_1 \, n_2 \iff n_1 + n_2 \text{ е четно}$$
е транзитивна.

2. Релацията R_2 в $\mathbb{N}$, дефинирана чрез
$$n_1 \, R_2 \, n_2 \iff n_1 + n_2 \text{ е нечетно}$$
не е транзитивна, защото $1 \, R_2 \, 2$, $2 \, R_2 \, 3$, но $1 \not R_2 \, 3$

Твърдение 3.4. Една бинарна релация R в A е транзитивна тогава и само тогава, когато $R \circ R \subseteq R$.

Доказателство. Нека първо R е транзитивна. Нека $(a,b) \in R \circ R$. Тогава за някое x имаме $(a,x),(x,b) \in R$, т.е. $a\,R\,x$, $x\,R\,b$, откъдето $a\,R\,b$, т.е. $(a,b) \in R$. Следователно $R \circ R \subseteq R$.

Обратно, нека $R \circ R \subseteq R$. Тогава, ако $a\,R\,b$ и $b\,R\,c$ (т.е. $(a,b) \in R$ и $(b,c) \in R$), то $(a,c) \in R \circ R$, откъдето $(a,c) \in R$, т.е. $a\,R\,c$. Следователно R е транзитивна.

□

Дефинираме *степен* на релация по следната рекурсивна схема. Нека R е бинарна релация в A. Тогава

$$\begin{aligned} R^0 &= \mathrm{id}_A; \\ R^{n+1} &= R \circ R^n. \end{aligned}$$

Ясно е, че $R^1 = R$. В сила е следното твърдение.

Твърдение 3.5. Нека R е бинарна релация в A и $a, a' \in A$. Тогава, за всяко $n > 0$,

$$a\,R^n\,a' \iff \text{съществуват } a_1, \ldots, a_{n-1} \in A, \text{такива че } a\,R\,a_1,\ a_1\,R\,a_2, \ldots, a_{n-1}\,R\,a'.$$

Доказателство. Ще извършим доказателството с индукция по n. При $n = 1$ твърдението е очевидно. Нека сега n е естествено, число за което твърдението е вярно. Тогава

$$\begin{aligned} a\,R^{n+1}\,a' &\iff a\,R \circ R^n\,a' \\ &\iff \text{съществува } a_n \in A,\ \text{такова че } a\,R^n\,a_n \text{ и } a_n\,R\,a' \\ &\overset{\text{и.п.}}{\iff} \text{съществуват } a_1, \ldots, a_{n-1}, a_n \in A, \text{такива че } a\,R\,a_1,\ a_1\,R\,a_2, \ldots, a_{n-1}\,R\,a_n, a_n\,R\,a'. \end{aligned}$$

□

От горното твърдение директно следва, че

$$R^n \circ R^m = R^{m+n}.$$

Ако R е бинарна релация, то с R^+ ще означаваме релацията

$$R^+ = \bigcup_{n>0} R^n.$$

Теорема 3.6. Нека R е бинарна релация. Тогава R^+ е транзитивна релация, съдържаща R. При това, ако Q е транзитивна релация, съдържаща R, то $R^+ \subseteq Q$.

Доказателство. Първо ще докажем, че R е транзитивна. За целта нека $a\,R^+\,b$ и $b\,R^+\,c$. Тогава $a\,R^m\,b$ и $b\,R^n\,c$ за някои $m, n > 0$, откъдето $a\,R^n \circ R^m\,c$, т.е. $a\,R^{n+m}\,c$ и значи $a\,R^+\,c$. Следователно R^+ е транзитивна и тъй като $R^1 = R$, то R^+ съдържа R.

Нека сега Q е транзитивна релация, съдържаща R. Ще докажем, че $R^n \subseteq Q$ за $n > 0$. При $n = 1$ твърдението следва от $R^1 = R$ и $R \subseteq Q$. Нека сега n е естествено число, такова че $R^n \subseteq Q$. Тогава $R^{n+1} \subseteq Q \circ Q$, тъй като $R \subseteq Q$. Но Q е транзитивна и значи $Q \circ Q \subseteq Q$, откъдето $R^{n+1} \subseteq Q$. Следователно $R^+ \subseteq Q$.

□

Съгласно горната теорема R^+ е най-малката (по отношение на $\subseteq$) транзитивна релация, съдържаща R^+. Релацията R^+ наричаме *транзитивно затваряне* на R.

Релацията

$$R^* = \bigcup_{n \geq 0} R^n$$

е най-малката (по отношение на $\subseteq$) рефлексивна и транзитивна релация, съдържаща R (защото $R^* = R^0 \cup R^+$). Релацията R^* наричаме *рефлексивно и транзитивно затваряне* на R.

3.2 Релации на еквивалентност

Ще казваме, че бинарната релация R в A е *релация на еквивалентност в* A, ако R е рефлексивна, симетрична и транзитивна релация в A, т.е. ако за всеки избор на $a, b, c \in A$ в сила са

1. aRa (*рефлексивност*);
2. $aRb \Rightarrow bRa$ (*симетричност*);
3. $aRb, bRc \Rightarrow aRc$ (*транзитивност*).

Обичайно релациите на еквивалентност се означават със символите $\equiv$, $\sim$, $\simeq$.

Пример.

1. Релацията $\sim_1$ в $\mathbb{C}$, дефинирана чрез
$$z_1 \sim z_2 \iff |z_1| = |z_2|$$
е релация на еквивалентност.

2. Релацията $\sim_2$ в $\mathbb{C} \setminus \{0\}$, дефинирана чрез
$$z_1 \sim z_2 \iff \frac{z_1}{|z_1|} = \frac{z_2}{|z_2|}$$
е релация на еквивалентност.

3. Ако $n > 1$ е естествено число, то релацията $\equiv \mod n$ в $\mathbb{Z}$, дефинирана чрез
$$a \equiv b \mod n \iff n|(a-b)$$
е релация на еквивалентност.

4. Ако G е група (с операция умножение) и $H \leq G$, то релацията $\equiv \mod H$ в G, дефинирана чрез
$$g_1 \equiv g_2 \mod H \iff g_1^{-1} g_2 \in H$$
е релация на еквивалентност.

5. Нека $\mathcal{C}$ е множеството от всички редици на Коши от рационални числа. Релацията $\sim$ в $\mathcal{C}$, дефинирана чрез
$$\alpha \sim \beta \iff \lim |a_n - \beta_n| = 0$$
е релация на еквивалнентност.

6. Равенството между (геометрични) насочени отсечки е релация на еквивалентност между насочени отсечки.

7. Релацията $\sim$ в $\mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$, дефинирана чрез
$$(a, b) \sim (c, d) \iff ad = bc$$
е релация на еквивалентност.

Нека $\equiv$ е релация на еквивалентност в A. Множеството
$$[a]_{\equiv} = \{b \in A \mid a \equiv b\}$$
ще наричаме клас на еквивалентност с представител (или още, породен от) $a \in A$. Когато релацията на еквивалентност се подразбира, ще пишем само $[a]$.

Твърдение 3.7. Нека $\equiv$ е релация на еквивалентност в A. Тогава за всяко $a, b \in A$ в сила са:

1. $a \in [a]$;
2. $a \equiv b \iff [a] = [b]$;
3. $[a] \cap [b] \neq \emptyset \Rightarrow [a] = [b]$.

Доказателство. (1) следва от $a \equiv a$.

(2) Нека първо $a \equiv b$. Тогава от транзитивността и симетричността на $\equiv$ имаме

$$x \in [a] \iff x \equiv a \iff x \equiv b \iff x \in [b]$$

и значи $[a] = [b]$.

Обратно, ако $[a] = [b]$, то $a \in [b]$ (т.к. $a \in [a]$) и следователно $a \equiv b$.

(3) Нека $c \in [a] \cap [b]$. Тогава $a \equiv c$ и $b \equiv c$ и значи $[a] = [c] = [b]$.

□

От (2) и (3) следва, че множеството $A/_{\equiv} = \{[a] \mid a \in A\}$ се състои от взаимно непресичащи се множества и $\bigcup A/_{\equiv} = A$. Ще казваме, че сме факторизирали A по релацията $\equiv$, а $A/_{\equiv}$ ще наричаме факторможество на A по $\equiv$.

Пример. При означенията на предния пример имаме

1. $[z]_{\sim_1}$ са всички комплексни числа, които лежат на окръжнсост с център 0 и радиус $|z|$.
2. $[z]_{\sim_2}$ е лъчът с начало 0, който съдържа числото z.
3. $[a]_{\mod n}$ е класът остатъци при делене на n.
4. $[g]_{\mod H}$ е левият съседен клас gH.
5. $[\alpha]_{\sim}$ е реалното число, което границата на редицата α.
6. $\left[\overrightarrow{AB}\right]$ е свободния вектор с представител $\overrightarrow{AB}$.
7. $[(a,b)]_{\sim}$ е рационалтото число $\dfrac{a}{b}$.

В случай, че факторможеството $A/_{\equiv}$ е крайно ще казваме, че релацията $\equiv$ има *краен индекс*. В този случай $|A/_{\equiv}|$ наричаме индекс на релацията $\equiv$.

Ще казваме, че множеството F от *непразни* множества е *разбиване* на множеството A, ако $\bigcup_{E \in F} E = A$ и $E_1 \cap E_2 = \emptyset$ за всеки две различни $E_1, E_2 \in F$.

Съгласно Твърдение 3.7, ако $A/\equiv$ е разбиване на A, т.е. всяка релация на еквивалнтност в A задава разбиване на A. Сега ще видим, че е вярно и обратното:

Твърдение 3.8. Нека F е разбиване на A. Тогава релацията $\equiv_F$, зададена чрез

$$a \equiv_F b \iff a, b \in E \text{ за някое } E \in F,$$

е релация на еквиваленност. При това $F = A/\equiv_F$.

Доказателство. Очевидно $\equiv_F$ е симетрична релация, така че остава да докажем само, че тя е рефлексивна и транзитивна. Тъй като $A = \bigcup_{E \in F} E$, то за всяко $a \in A$ съществува $E \in F$, такова че $a \in E$. Следователно $a \equiv_F a$ за всяко $a \in A$ и значи $\equiv_F$ е рефлексивна. Нека сега $a, b, c \in A$ са такива, че $a \equiv_F b$ и $b \equiv_F c$, т.е. $a, b \in E$ и $b, c \in E'$ за някои $E, E' \in F$. Тогава $c \in E \cap E' \neq \emptyset$, откъдето $E = E'$ и значи $a \equiv_F c$. Следователно $\equiv_F$ е транзитивна.

Остава да докажем, че елементите на F са точно класовете на еквивалентност по $\equiv_F$. За целта нека фиксираме $a \in A$ и нека $E \in F$ е такова, че $a \in E$ (такова има, тъй като $A = \bigcup_{E \in F} E$). Очевидно $E \subseteq [a]$. За обратното включване нека $b \in [a]$, т.е. $b \equiv_F a$. Тогава $b, a \in E'$ за някое $E' \in F$. Но $a \in E \cap E'$, откъдето следва, че $E = E'$ и значи $b \in E$. Следователно $[a] \subseteq E$ и значи $[a] = E$. Така всеки клас на еквивалентност е елемент на F. Обратно, ако $E \in F$, то E е непразно и значи $a \in E$ за някое $a \in A$. Тогава $E = [a]$ и значи всеки елемент на F е клас на еквивалентност по $\equiv_F$.

□

3.3 Частични наредби

Ще казваме, че бинарната релация R в A е частична наредба (в A), ако R е *ирефлексивна* и *транзитивна* релация, т.е. ако за всеки избор на $a, b, c \in A$ в сила са

1. не е вярно aRa;
2. ако aRb и bRc, то aRc.

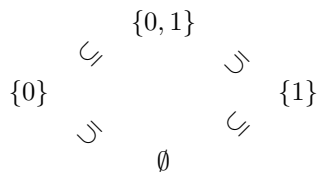
Обичайно частичните наредби се означават със символа $<$. Ако $<$ е частична наредба в A, то с $\leq$ ще означаваме релацията дефинирана чрез

$$a \leq b \iff a < b \text{ или } a = b.$$

Така дефинираната релацията $\leq$ е *рефлексивна*, *антисиметрична* и *транзитивна*, т.е.

1. $a \leq a$;
2. ако $a \leq b$ и $b \leq a$, то $a = b$;
3. ако $a \leq b$ и $b \leq c$, то $a \leq c$.

Релацията $<$ се нарича строга, а релацията $\leq$ нестрога частична наредба. Примери за частични наредби са стандартните наредби на естествените, целите, рационалните и реалните числа. Особеното при тези наредби е, че за всеки избор на елементи a и b в сила е или $a \leq b$, или $b < a$. Наредби с това свойство наричаме *линейни*. Разбира се, не всички наредби са линейни. Да разгледаме например положителните цели числа $\mathbb{Z}^+$ и релацията делимост. Ясно е, че тази релация е (нестрога) частична наредба. Но спрямо делимостта всеки две прости числа са несравними помежду си и следователно наредбата не е линейна. По-естествен пример за нелинейна наредба е релацията $\subseteq$. Да разгледаме множеството от подмножествата на $\{0, 1\}$. Спрямо $\subseteq$ то е наредено по следния начин



Ясно е, че $\{0\} \not\subseteq \{1\}$ и $\{1\} \not\subseteq \{0\}$ и следователно наредбата не е линейна.

Нека $<$ е частична наредба в A и $X \subseteq A$. Ще казваме, че $x \in X$ е *минимален* (*максимален*) елемент в X, ако за всяко $y \in X$ е в сила $y \not< x$ ($x \not< y$), т.е. ако в X няма елемент по-малък (по-голям) от x. Ще казваме, че $x \in X$ е *най-малкият* (*най-големият*) елемент в X, ако за всяко $y \in X$ е в сила $x \leq y$ ($y \leq x$), т.е. ако всеки елемент на X, различен от x, е по-голям (по-малък) от x.

Ясно е, че ако в X има най-малък (най-голям) елемент, то той е единствен. При това, той е и минимален (максимален) елемент на X. От друга страна в X може да има и повече от един минимален (максимален) елемент. Така например, в множеството $\{\{0\}, \{1\}, \{0, 1\}\}$ по отношение на включването елементите $\{0\}$ и $\{1\}$ са минимални, а елемента $\{0, 1\}$ е най-голям.

Твърдение 3.9. Ако $<$ е линейна наредба, то понятията минимален (максимален) елементи и най-малък (максимален) елемент са еквивалентни.

Доказателство. Нека $<$ е линейна наребда в A, $X \subseteq A$ и $m \in X$. Тогава

$$\begin{aligned} m \text{ е минимален в } X &\iff x \not< m \text{ за всяко } x \in X \\ &\iff m \leq x \text{ за всяко } x \in X \\ &\iff m \text{ е най-малък в } X. \end{aligned}$$

□

Пример.

1. Относно делимостта в $\mathbb{N}$, 0 е най-голям, а 1 е най-малък елемент. От друга страна, всяко просто число е минимален елмент в $\mathbb{N} \setminus \{1\}$.
2. В $\mathbb{Z}$ няма минимално (максимално) цяло число по отношение на обичайната наредба.

Лема 3.10. Нека $<$ е частична наредба в A. Тогава във всяко непразно, крайно подмножество на A има минимален (максимален) елемент.

Доказателство. Нека $X \subseteq A$ е крайно, $|X| = n$ и $X = \{x_1, x_2, \dots x_n\}$. Да допуснем, че в X няма минимален елемент. Тогава в X има елемент, по-малък от x_1 и нека например това е x_2. Елементът x_2 също не е минимален и следователно в X има елемент, по-малък от x_2 и нека например това е x_3. Повтаряйки това разсъждение още $n-1$ пъти получаваме

$$x_n < x_{n_1} < \dots < x_2 < x_1.$$

Но x_n не е минимален елемент на X и значи $x_i < x_n$ за някое i, $1 \leq i \leq n$, откъдето $x_i < x_n \leq x_i$ и значи $x_i < x_i$, противоречие. Следователно в X има минимален елемент.

□

Ще казваме, че частичната наредба $<$ в A е *добра наредба*, ако всяко непразно подмножество на A има най-малък елемент.

Пример.

1. Множеството $\mathbb{N}$ относно обичайната наредба на естествени числа е добре наредено.
2. Всяко крайно, линейно наредено множество е добре наредено.

Теорема 3.11 (за тополгичната сортировка)**.** Нека $<$ е частична наредба в крайното множество A. Тогава съществува линейна наредба $\prec$ в A, такава че $<\subseteq\prec$. С други думи, всяка частична наредба в едно крайно множество може да бъде допълнена до линейна (а значи и добра) наредба.

Доказателство. Нека $|A| = n$ и $A = \{a_1, \dots, a_n\}$. Тъй като A е крайно, в A има минимален елемент. Нека b_1 е първият елемент в редицата $a_1, a_2, \dots, a_n$, който е минимален в A. Множеството $A \setminus \{b_1\}$ е крайно (и непразно при $n > 1$) и значи в него има минимален елемент. Нека b_2 е първият елемент в редицата $a_1, a_2, \dots, a_n$, който е минимален в $A \setminus \{b_1\}$. Продължавайки така още $n-2$ пъти, получаваме пермутация

$$b_1, b_2, \dots, b_n$$

на елементите на A, такава че b_i, $1 \leq i \leq n$, е минимален елемент в множеството $A \setminus \{b_1, b_2, \dots, b_{i-1}\}$. Тогава, ако $j < i$, то $b_i \in A \setminus \{b_1, b_2, \dots, b_{j-1}\}$ и значи $b_i \not< b_j$. Така,

$$b_i < b_j \Longrightarrow i < j.$$

Дефинираме релацията $\prec$ в A чрез

$$b_i \prec b_j \iff i < j, \quad 1 \leq i, j \leq n.$$

Тогава $\prec$ е линейна наредба в A и от казаното по-горе следва, че $<\subseteq\prec$.

□

ГЛАВА 4

Булеви функции

4.1 Дефиниция на булеви функции

Под булева функция на n-аргумента ($n \geq 1$) ще разбираме изображение $f : \{0,1\}^n \to \{0,1\}$. С $\mathbb{B}_n$ ще оначаваме колекцията на всички функции на n-аргумента. Да забележим, че елементите на $\mathbb{B}_n$ могат да бъдат интерпретирани като характеристичните функции на подмножествата на $\{0,1\}^n$. В частност множеството $\mathbb{B}_n$ е равномощно с множеството $\mathcal{P}\big(\{0,1\}^n\big)$. Оттук и това, че в $\{0,1\}^n$ има 2^n елемента, следва, че в $\mathbb{B}_n$ има 2^{2^n} елемента.

Нека първо разгледаме множеството $\mathbb{B}_1$. В $\mathbb{B}_1$ има четири различни елемента, които ние ще означаваме с $\mathbf{0}$, $\mathbf{1}$, x и $\overline{x}$. Считаме, че всяка една от тези функции е на аргумента x, който приема стойности 0 и 1. Стойностите на функциите в зависимост от стойността на аргумента x дефинираме, чрез следната таблица:

x	$\mathbf{0}$	$\mathbf{1}$	x	$\overline{x}$
0	0	1	0	1
1	0	1	1	0

Функциите $\mathbf{0}$ и $\mathbf{1}$ ще наричаме константни, функцията x — идентитет, а функцията $\overline{x}$ — отрицание (да отбележим, че понякога тази функция се означава с $\neg x$).

Накрая да разгледаме множеството от двоичните функции $\mathbb{B}_2$. В $\mathbb{B}_2$ има общо 16 различни функции. Позабележителните шест от тях означаваме с

$$x \vee y,\ xy,\ x \to y,\ x \equiv y,\ x|y \text{ и } x \downarrow y$$

и дефинираме с помощта на следната таблица (в зависимост от стойностите на арументите x и y)

x	y	$x \vee y$	xy	$x \to y$	$x \equiv y$	$x\|y$	$x \downarrow y$	$x + y$
0	0	0	0	1	1	1	1	0
0	1	1	0	1	0	1	0	1
1	0	1	0	0	0	1	0	1
1	1	1	1	1	1	0	0	0

Функцията $x \vee y$ наричаме *дизюнкция* (или още *логическо или*), функцията xy — *конюнкция* (или още *логическо и*), функцията $x \to y$ — *импликация*, функцията $x \equiv y$ — *еквивалентност*, функцията $x|y$ — *черта на Шефър*, а функцията $x \downarrow y$ — *стрелка на Пиърс*. Накрая да отбележим, че понякога конюнкцията се означава и чрез символите & и $\wedge$, а еквивалентността — с $\leftrightarrow$.

В сила са следните равенства:

$$x \vee \mathbf{0} = x \tag{4.1}$$
$$x\mathbf{0} = \mathbf{0} \tag{4.2}$$
$$x\mathbf{1} = x \tag{4.3}$$
$$x \vee \mathbf{1} = \mathbf{1} \tag{4.4}$$
$$x \vee (y \vee z) = (x \vee y) \vee z \tag{4.5}$$
$$x \vee x = x \tag{4.6}$$
$$xx = x \tag{4.7}$$
$$x(yz) = (xy)z \tag{4.8}$$
$$x(y \vee z) = xy \vee xz \tag{4.9}$$
$$x \vee yz = (x \vee y)(x \vee z) \tag{4.10}$$
$$\overline{x \vee y} = \overline{x}\,\overline{y} \tag{4.11}$$
$$\overline{xy} = \overline{x} \vee \overline{y} \tag{4.12}$$
$$x \to y = \overline{x} \vee y \tag{4.13}$$
$$x \equiv y = (x \to y)(y \to x) \tag{4.14}$$
$$x|y = \overline{xy} \tag{4.15}$$
$$x \downarrow y = \overline{x \vee y} \tag{4.16}$$
$$x + y = \overline{x \equiv y} \tag{4.17}$$
$$\overline{x} = x + \mathbf{1} \tag{4.18}$$

4.2 Съвършена дизюнктивна нормална форма. Теорема на Бул

Нека $a \in \{0,1\}$. Въвеждаме означението x^a по следния начин

$$x^a = \begin{cases} x, & a = 1, \\ \overline{x}, & a = 0. \end{cases}$$

Да забележим, че $b^a = 1$ тогава и само тогава, когато $b = a$. Наистина

$$b^a = 1 \iff a = b = 0 \text{ или } a = b = 1 \iff b = a.$$

Нека сега $f \in \mathbb{B}_n$ приема стойност 1 за поне една стойност на аргументите си. Да разгледаме n-местната функция h, дефинирана с

$$h(x_1, x_2, \ldots, x_n) = \bigvee_{f(a_1,a_2,\ldots,a_n)=1} x_1^{a_1} x_2^{a_2} \ldots x_n^{a_n}$$

Ще докажем, че $f = h$. Да разгледаме една n-орка $(b_1, b_2, \ldots, b_n) \in \{0,1\}^n$. Имаме

$$b_1^{a_1} b_2^{a_2} \ldots b_n^{a_n} = 1 \iff b_1 = a_1,\ b_2 = a_2,\ \ldots,\ b_n = a_n.$$

Оттук

$$\begin{aligned} h(b_1, b_2, \ldots, b_n) = 1 \quad &\iff \bigvee_{f(a_1,a_2,\ldots,a_n)=1} b_1^{a_1} b_2^{a_2} \ldots b_n^{a_n} = 1 \\ &\iff f(b_1, b_2, \ldots, b_n) = 1. \end{aligned}$$

Следователно $h = f$.

Така доказахме следната теорема:

Теорема 4.1. Нека $f \in \mathbb{B}_n$ приема стойност 1 за поне една стойност на аргументите си. Тогава

$$f(x_1, x_2, \ldots, x_n) = \bigvee_{f(a_1,a_2,\ldots,a_n)=1} x_1^{a_1} x_2^{a_2} \ldots x_n^{a_n}.$$

Ще казваме, че сме представили f в съвършена дизюнктивна нормална форма.

Като следствие на тази теорема получаваме, следната теорема

Теорема 4.2 (Бул)**.** Всяка булева функция може да се получи от функциите отрицание и дизюнкция.

Доказателство. Тъй като $xy = \overline{\overline{x} \vee \overline{y}}$, достатъчно е да докажем, че всяка булева функция може да се изрази чрез отрицанието, дизюнкцията и конюнкцията. Нека $f \in \mathbb{B}_n$. Ако f приема стойност 1 за поне една стойност на аргументите си, то съгласно предната теорема f се изразява чрез отрицанието, дизюнкцията и конюнкцията. От друга страна, ако $f(a_1, a_2, \ldots, a_n) = 0$ за всяко $a_1, a_2, \ldots, a_n \in \{0, 1\}$, то

$$f(x_1, x_2, \ldots, x_n) = x_1\overline{x_1}.$$

□

ГЛАВА 5

Графи и дървета

5.1 Неориентирани и ориентирани графи. Мултиграфи.

Определение 5.1. Под неориентиран прост граф (или само граф) ще разбираме наредена двойка $G = (V, E)$, където

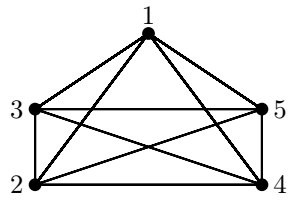
$$E \subseteq \{\{u, v\} \mid u, v \in V \text{ и } u \neq v\}.$$

Елементите на V ще наричаме върхове, а тези на E — ребра на графа. Ако $e = \{u, v\} \in E$ ще казваме, че върховете u и v са краища на реброто e.

Неориентираните графи онагледяваме, като върховете на графа рисуваме като точки, а ребрата между тях като дъги (или съответно отсечки) свързващи точките. Например, графът

$$K_5 = (\{1, 2, 3, 4, 5\}, \{\{i, j\} \mid 1 \leq i < j \leq 5\}),$$

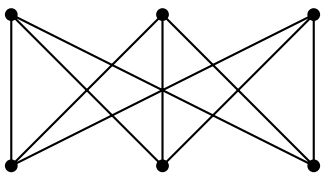
известен като пълния граф с 5 върха, можем да изобразим по следния начин



Графът

$$K_{3,3} = (\{1, 2, 3, 4, 5, 6\}, \{\{i, j\} \mid 1 \leq i \leq 3 < j \leq 6\})$$

можем да изобразим по следния начин



Определение 5.2. Под ориентиран граф ще разбираме наредена двойка $G = (V, E)$, където

$$E \subseteq V \times V.$$

Елементите на V ще наричаме върхове, а тези на E — ребра на графа. Ако $e = (u, v) \in E$ ще казваме, че върхът u е начало, а върхът v — край, на реброто e.

Понякога в литературата се среща следната вариация на неориентиран граф — неориентиран граф е ориетиран граф $G = (V, E)$, за който E е ирефлексивна и симетрична релация, т.е. за всеки избор на върхове $u, v \in V$

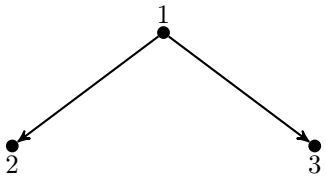
$$(u, u) \notin E$$

и

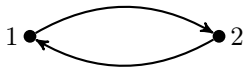
$$(u, v) \in E \iff (u, v) \in E.$$

Доколкото обаче по принцип в неориентирания граф ребрата нямат начало и край, т.е. двата края на реброто са неотличими, по-добре е да считаме, че ребрата на графа са ненаредени двойки, така както е в Дефиниция 5.1.

Ориентираните графи онагледяваме, като върховете на графа рисуваме като точки, а ребрата между тях като стрелки, свързващи точките. Например, графът $G = \{\{1, 2, 3\}, \{(1, 2), (1, 3)\}\}$ можем да изобразим по следния начин:



Графът $G = \{\{1, 2\}, \{(1, 2), (2, 1)\}\}$ можем да изобразим по следния начин



Нека да отбележим, че в неоритираните графи между два върха има най-много едно ребро, а в ориентираните — най-много две. За да позволим наличието на повече от две ребра между два върха въвеждаме понятието *мултиграф*.

Определение 5.3. Под мултиграф ще разбираме наредена четворка $G = (V, E, \pi_1, \pi_2)$, където

$$\pi_1, \pi_2 : E \to V.$$

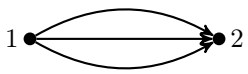
Елементите на V ще наричаме върхове, а тези на E — ребра на графа. За всяко ребро $e \in E$, върхът $\pi_1(e)$ ще наричаме начало на e, върхът $\pi_2(e)$ — край на e.

Мултиграфите онагледяваме, по същия начин както неориентираните графи.

Пример. Нека G е мултиграфът

$$G = \{\{1, 2\}, \{A, B, C\}, \{(A, 1), (B, 1), (C, 1)\}, \{(A, 2), (B, 2), (C, 2)\}\}.$$

G можем да изобразим по следния начин



Под *мултиграф с етикети* от A ще разбираме петорката $G = (V, E, \pi_1, \pi_2, \lambda)$, където $G = (V, E, \pi_1, \pi_2)$ е мултиграф, а λ е функция от E в A.

Нека $G = (V, E, \pi_1, \pi_2, \lambda)$ е етикетиран мултиграф, за който на всеки две различни ребра с едно и също начало и един и същи край се съпоставят различни етикети, т.е. за всеки две различни $e_1, e_2 \in E$,

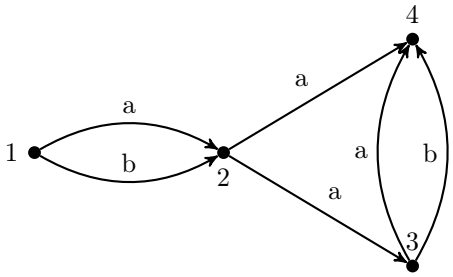
$$\pi_1(e_1) = \pi_1(e_2) \text{ и } \pi_2(e_1) = \pi_2(e_2) \Longrightarrow \lambda(e_1) \neq \lambda(e_2).$$

В G всяко ребро се определя еднозначно от своето начало, своя край и своя етикет. В такъв случай, можем да считаме, че $G = (V, \Delta)$, където

$$\Delta \subseteq V \times A \times V.$$

Така всяко ребро е от вида (u, a, v), където u и v са върхове, a е етикет. Върхът u ще наричаме начало, върхът v — край, а a —етикет на реброто.

Следващата къртинка онагледява пример за мултиграф с етикети:


4
a
a
1
b
a
b
2
a
3

5.2 Път в граф. Матрица на съседство.

Казваме, че редицата

$$v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n$$

е (краен) *nъm* в G с начало v_0, край v_n и дължина n ($n \geq 0$), ако за всяко $1 \leq i \leq n$, e_i е ребро на G с начало v_{i-1} и край v_i.

Да отебележим, че за всеки връх v, редицата

$$v$$

е път с начало v, край v и дължина 0. Да отбележим, че пътищата на G могат да бъдат описани и чрез следната индуктивна дефиниция:

1. За всеки връх v на G, v е път в G с начало v, край v и дължина 0.
2. Нека π е път в G с начало v_0, край v_n и дължина n. Тогава за всяко ребро e_{n+1} с начало v_n и край, да речем, v_{n+1},

   $$\pi, e_{n+1}, v_{n+1}$$

   е път с начало v_0, край v_{n+1} и дължина $n+1$.

Ясно е, че за пътищата в един граф са в сила следните елементарни свойства:

1. Ако $v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n$ е път в G, то за всяко $0 \leq i \leq j \leq n$

   $$v_i, e_{i+1}, v_{i+1}, \ldots, e_j, v_j$$

   е път с начало v_i, край v_j и дължина $j - i$.
2. Ако $v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n$ и $v'_0, e'_1, v'_1 \ldots, e'_k, v'_k$ са пътища в G и $v_n = v'_0$, то

   $$v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n, e'_1, v'_1 \ldots, e'_k, v'_k$$

   е път в G.

В случая на неориентиран и ориентиран граф, всеки път е изцяло определен от върховете, през които минава, тъй като между всеки два върха съществува най-много едно ребро. Така за неориентиран и ориентиран граф, в записа на даден път можем да изпускаме ребрата, по които пътя минава и следователно за път в неориентиран (ориентиран) граф можем да считаме редиците от върхове

$$v_0, v_1, v_2, \ldots, v_n,$$

за които за всяко $0 \leq i \leq n-1$, съществува ребро с начало v_i и край v_{i+1}.

От друга страна в случая на ориентиран граф и мултиграф всяко ребро еднозначно определя своите начало и край. Така в този случай всеки път с ненулева дължина е еднозначно определен от ребрата, по които минава и следоветлно за пътища с ненулева дължина можем да считаме редиците от ребра

$$e_1, e_2, \ldots, e_n,$$

за които краят на e_i съвпада с началото на e_{i+1} за $1 \leq i \leq n-1$.

Накрая нека да отбележим, че ако G е етикетиран мултиграф от вида $G = (V, \Delta)$, ребрата еднозначно се определят от началото, края и етикета си и следователно, за пътища в G можем да считаме редиците

$$v_0, a_1, v_1, a_2, v_2, \ldots, a_n, v_n,$$

където $(v_{i-1}, a_i, v_i) \in \Delta$ за $1 \leq i \leq n$.

Ще казваме, че пътят $v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n$ е *прост*, ако $v_i \neq v_j$ за всички i и j, $0 \leq i \neq j \leq n$, такива че $\{i, j\} \neq \{0, n\}$, т.е. ако пътят не минава два пъти през един и същи връх.

Твърдение 5.4. Нека в графа G има път с начало върха v и край върха v'. Тогава в G има прост път с начало v и край v'.

Доказателство. Нека L е множеството от всички естествени числа n, за които съществува път с начало v, край v' и дължина n. Тогава, съгласно условието на твърдението, множеството L е непразно. Нека m е най-малкият елемент на L и нека $v = v_0, e_1, v_1, e_2, v_2, \ldots, e_m, v_m = v'$ е път с начало v, край v' и дължина m (т.е. това е път, свързващ v и v' с възможно най-малка дължина). Ще докажем, че този път е прост. Да допуснем противното. Тогава $v_i = v_j$ за някои $0 \leq i < j \leq n$. Тогава

$$v_0, e_1, v_1, \ldots, e_i, v_i, e_{j+1}, v_{j+1}, \ldots, e_m, v_m$$

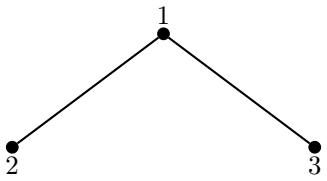
е път с начало v, край v' и дължина $m - (j - i) < m$, което противоречи на избора на m. Следователно пътят $v_0, e_1, v_1, e_2, v_2, \ldots, e_m, v_m$ е прост.

□

Ще казваме, че графът G е *краен*, ако G има крайно много върхове и ребра. Да отбележим, че един неориентиран или ориентиран граф е краен тогава и само тогава, когато G има краен брой върхове.

Нека G е краен граф с n върха и нека u_0, u_1, …, u_n е фиксирано изреждане (пермутация) на върховете на G. Квадратната матрица $A = (a_{ij})$ от ред n, за която a_{ij} е равен на броя на ребрата с начало u_i и край u_j, ще наричаме *матрица на съседство* на G.

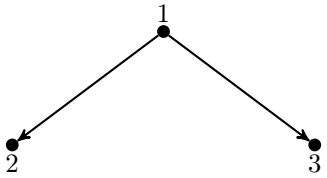
Неориентираният граф



има матрица на съседство

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

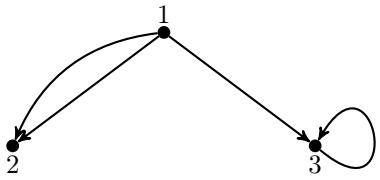
Ориентираният граф



има матрица на съседство

$$\begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Мултиграфът



има матрица на съседство

$$\begin{pmatrix} 0 & 2 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Да отбележим, че матрицата на съседство не е еднозначно определена, тъй като тя зависи от фиксираното изреждане на върховете на графа. Така, например, за последния граф имаме 6 различни възможни изреждания на върховете му, като съответните матрици на съседство са:

$$\begin{pmatrix} 0 & 2 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 2 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 2 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 2 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 2 \\ 0 & 0 & 0 \end{pmatrix}$$

Те се получават една от друга извършвайки една и съща пермутация върху редовете и стълбовете на матрицата (например, третата матрица се получава от първата разменяйки местата на първи и втори ред, и съответно първи и втори стълб).

Теорема 5.5. Нека G е краен граф с матрица на съседство A. Тогава за всяко $k \geq 0$, (i,j)-тият елемент на A^k е равен на броя на различните пътища с дължина k, начало върха, съответстващ на i-тия ред (стълб) на A, и край върха, съответстващ на j-тия ред (стълб) на матрицата A.

Доказателство. Без ограничение можем да считаме, че върховете на G (и тяхното изреждане, при което се получава матрицата A) са $1, 2, \ldots, n$. Ще докажем твърдението с индукция по k. За $k \leq 1$ твърдението е очевидно. Нека сега k е такова, че за всяко $1 \leq i, j \leq n$, (i,j)-тият елемент на A^k е равен на броя на различните пътища с начало i, край j и дължина k. Нека $A^k = (b_{ij})$ и $A^{k+1} = (c_{ij})$. Тогава

$$c_{ij} = \sum_{s=1}^{n} b_{is}a_{sj} = b_{i1}a_{1j} + b_{i2}a_{2j} + \cdots + b_{in}a_{nj}.$$

Нека сега означим с

$$P_{ij} = \{\pi \mid \pi \text{ е път на начало } i \text{ край } j \text{ и дължина } k+1\}$$

и с

$$P_{ij}^s = \{\pi \in P_{ij} \mid \text{предпоследният връх в } \pi \text{ е } s\}.$$

Тогава $P_{ij}^s \cap P_{ij}^t = \emptyset$ за $s \neq t$ и $P_{ij} = P_{ij}^1 \cup P_{ij}^2 \cup \cdots \cup P_{ij}^n$ и значи

$$|P_{ij}| = |P_{ij}^1| + |P_{ij}^2| + \cdots + |P_{ij}^n|.$$

От друга страна всеки елемент на P_{ij}^s се получава еднозначно като изберем път с начало i, край s и дължина k, и в края му добавим ребро с начало s и край j. Но пътищата с i, край s и дължина k са точно b_{is}, а ребрата с начало s и край j са a_{sj} на брой и следователно (съгласно принципа за умножението)

$$|P_{ij}^s| = b_{is}a_{sj}.$$

Оттук

$$|P_{ij}| = b_{i1}a_{1j} + b_{i2}a_{2j} + \cdots + b_{in}a_{nj} = c_{ij},$$

което трябваше да докажем.

□

Ще казваме, че един граф G е свързан, ако между всеки два върха съществува път. Използвайки матрицата на съседство и прости пътища, получаваме следното необходимо и достатъчно условие за това един граф да е свързан.

Твърдение 5.6. Нека G е граф с n върха и матрица на съседство A. Тогава G е свързан тогава и само тогава, когато в матрицата

$$A^0 + A^1 + A^2 + \cdots + A^{n-1}$$

няма нулеви елементи.

Доказателство. От твърдението за матрицата на съседство следва, че ij-ият елемент на матрицата $A^0 + A^1 + A^2 + \cdots + A^{n-1}$ е равен на броя на различните пътища между i-тия и j-тия връх на G с дължина най-много $n-1$.

Ясно е, че ако в G има n върха, то всеки прост път е с дължина по-малка от n. Следоветлно, съгласно предното твърдение, G е свързан тогава и само тогава, когато между всеки два върха съществува път с дължина най-много $n-1$, т.е. когато в матрицата $A^0 + A^1 + A^2 + \cdots + A^{n-1}$ няма нулевите елементи.

□

5.3 Степен на връх. Формула на Ойлер

Под *степен* на връх v в графа G ще разбираме броя на ребрата в G, с които v е инцидентен, т.е. за които v е начало или край. Степента на v в G ще означаваме с $d_G(v)$ (когато G се подразбира, ще пишем просто $d(v)$).

Теорема 5.7 (Формула на Ойлер)**.** Нека G е краен граф с върхове V и ребра E. Тогава

$$\sum_{v \in V} d_G(v) = 2|E| - m,$$

където m е броят на примките (т.е. броят на ребрата e, такива че началото и краят на e са едни и същи). В частност, ако G е неориентиран, то

$$\sum_{v \in V} d_G(v) = 2|E|,$$

Доказателство. Да разгледаме множеството

$$I = \{(v, e) \mid v \in V, e \in E \text{ и } v \text{ е инцидентен с } e\}.$$

Всяко ребро e, което не е примка, е инцидентно с два различни върха и следователно се среща в точно два елемента на I. От друга страна всяка примка се среща точно в един елемент на I. Следователно

$$|I| = 2(|E| - m) + m = 2|E| - m.$$

От друга страна всеки връх v се среща в точно $d_G(v)$ елемента на I и следователно

$$|I| = \sum_{v \in V} d_G(v).$$

□

5.4 Свързани графи. Ациклични графи

До края на тази глава ще разглеждаме само неориентирани прости графи.

Твърдение 5.8. Нека G е свързан граф с n върха. Тогава в G има поне $n-1$ ребра.

Доказателство. Ще докажем твърдението с индукция по n. За $n=1$ и $n=2$ твърдението е очевидно. Нека сега предположим, че n е естествено число, такова че всеки свързан граф с n върха има поне $n-1$ ребра и нека $G=(V,E)$ е свързан граф с $n+1$ върха и m ребра. Ще разгледаме два случая.

Нека първо предположим, че $d_G(v)\geq 2$ за всеки връх v на G. Тогава, съгласно формулата на Ойлер,

$$2m=\sum_{v\in V} d_G(V)\geq 2(n+1)$$

и следователно $m\geq n+1>(n+1)-1$.

Нека сега предположим, че някой връх на G, да речем v, е от степен по-малка от 2. Тъй като G е свързан, $d_G(v)\geq 1$ и значи $d_G(v)$. Нека e е единственото ребро инцидентно с v. Да разгледаме графа $G'=(V\setminus\{v\},E\setminus\{e\})$. Твърдим, че G' е свързан. Наистина, нека v' и v'' са два върха на G', т.е. v' и v'' са върхове на G, различни от v. Нека

$$v=v_0,e_1,v_1,\ldots,e_k,v_k=v''$$

е прост път в G (такъв има защото G е свързан). Тогава $e_s\neq e_t$ за всяко $1\leq s<t\leq k$ и значи e не е сред ребрата e_1, e_2, …, e_k. Оттук v не е сред върховете v_0, v_1, …, v_k и следователно $v_0,e_1,v_1,\ldots,e_k,v_k$ е път в G'. Следователно G' е свързан граф с $n-1$ върха и $m-1$ ребра и съгласно индукционното предположение имаме $m-1\geq n-1$, т.е. $m>(n+1)-1$.

□

Ще казваме, че пътят $v_0,e_1,v_1,\ldots,e_n,v_n$ е *цикъл*, ако $v_0=v_n$. При това ще казваме, че цикълът е прост, ако пътят пътят $v_0,e_1,v_1,\ldots,e_n,v_n$ е прост. Ще казваме, че един граф е *ацикличен*, ако в него няма прости цикли с дължина по-голяма от 2.

Лема 5.9. Нека G е граф с краен брой върхове, в който всеки връх е от спепен поне 2. Тогава G не е ацикличен.

Доказателство. Нека $G=(V,E)$ е с n върха. Строим път с дължина n по следната схема. За v_0 изибираме произволен връх на G и нека v_1 е връх, за който $e_1=\{v_0,v_1\}\in E$. Избираме v_2, така че $e_2=\{v_1,v_2\}$ и $v_2\neq v_0$, което можем да направим, тъй като $\deg(v_1)\geq 2$. След това избираме v_3, така че $e_3=\{v_2,v_3\}$ и $v_3\neq v_1$, което отново можем да направим, тъй като $\deg(v_1)\geq 2$. Повтаряйки последоветлено този избор $n-1$ пъти, получаваме път

$$v_0,e_1,v_1,e_2,v_2,\ldots,e_n,v_n,$$

такъв че $v_{i-1}\neq v_{i+1}$ за $1\leq i\leq n-1$. Оттук и от това, че в G има n върха следва, че можем да изберем i и j, $0\leq i<i+2<j\leq n$, такива че $v_i,e_{i+1},v_{i+1},\ldots,e_j,v_j$ е прост цикъл с дължина $j-i>3$.

□

Твърдение 5.10. Нека G е ацикличен граф с n върха. Тогава в G има най-много $n-1$ ребра.

Доказателство. Ще докажем твърдението с индукция по n. За $n=1$ и $n=2$ твърдението е очевидно. Нека сега предположим, че n е естествено число, такова че всеки ацикличен граф с n върха има най-много $n-1$ ребра. Нека $G=(V,E)$ е ацикличен граф с $n+1$ върха и m ребра. Съгласно предната лема, в G има връх от степен най-много 1. Ако всеки връх е от степен 0, то твърдението е очевидно. За това нека предположим, че в G има връх от степен поне 1 и нека този връх е v. Тогава $d_G(v)=1$, т.е. v е инцидентен само с едно ребро и нека това ребро е e. Тогава $G'=\big(V\setminus\{v\},E\setminus\{e\}\big)$ е граф с n върха и $m-1$ ребра. При това всеки път в G' е път в G и значи G' е ацикличен. Оттук $m-1\leq n-1$ и значи $m+\leq(n+1)-1$.

□

5.5 Дървета. Покриващи дървета. Компоненти на свързаност

Ще казваме, че графът G е *дърво*, ако G е ацикличен и свързан. В частност, ако G е дърво с n върха, то в G има точно $n-1$ ребра.

Определение 5.11. Ще казваме, че графът $G' = (V', E')$ е *подграф* на графа $G = (V, E)$, ако $V' \subseteq V$ и $E' \subseteq E$.

С други думи, ако $G = (V, E)$ е граф, а $V' \subseteq V$ и $E' \subseteq E$ са такива, че $E' \subseteq \big\{\{v_1, v_2\} \mid v_1, v_2 \in V' \text{ и } v_1 \neq v_2\big\}$, т.е. ребрата в E' свързват върхове от V', то наредената двойка (V', E') е подграф на G.

Ще казваме, че G' е *покриващо дърво* на G, ако G' е подграф на G и G' и G имат едни и същи върхове.

Твърдение 5.12. Всеки краен свързан граф има покриващо дърво.

Доказателство. Нека $G = (V, E)$ е свързан граф с n върха и m ребра. Ако G е ацикличен, то G е дърво и няма какво да доказваме. За това нека предположим, че $v_0, e_1, v_1, \ldots, e_k, v_k$, където $n \geq 3$ е прост цикъл в G. Да разгледаме графът $G_1 = (V, E_1)$, където $E_1 = E \setminus \{e_1\}$. Ясно е, че G_1 е подграф на G. Твърдим, че G_1 е свързан. Наистина нека $v', v'' \in V$. и нека $v' = v'_0, e'_1, v'_1, \ldots, e'_s, v'_s = v''$ е път в G от v' до v''. Нека от този път образуваме нов път, замествайки всяко срещане на реброто e_1 с пътя $v_n, e_n, v_{n-1} \ldots, e_2, v_1$ или с пътя $v_1, e_2, v_2, \ldots, e_n, v_n$. Така получаваме път в G' с начало v' и край v'' и значи G' е свързан.

Сега, ако G_1 е ацикличен, то G_1 е търсеното покриващо дърво. В противен случай, можем да повторим горната конструкция и да получим свързан подграф $G_2 = (V, E_2)$ на G_1. Ако той е ацикличен, значи той е търсеното покриващо дърво. Продължавайки по тази схема, получаваме редица

$$G_0 = (V, E), G_1 = (V, E_1), G_2 = (V, E_2), \ldots$$

където G_{i+1} е свързан подграф на G_i, съдържащ точно едно ребро по-малко от G_i. Тъй като G съдържа n върха и m ребра, тази редица е крайна. По-точно G_i съдържа $m - i \geq n - 1$, т.е. $i \leq m - n + 1$ и следоветлно редицата съдържа най-много $m - n + 2$ графа. Нека G_k е последния граф в редицата. Тогава G_k е търсеното покриващо дърво.

□

Нека $G = (V, E)$ е граф. Във V въвеждаме следната релация $\sim_G$:

$$v' \sim_G v'' \iff \text{съществува път в } G \text{ с начало } v' \text{ и край } v''.$$

Ясно е, че $\sim_G$ е рефлексивна, симетрична и транзитивна релация и значи $\sim_G$ е релация на еквивалентност в V. Нека $V' \subseteq V$ е един клас на еквивалентност по отношение на $\sim_G$ и да разгледаме едно ребро $e \in E$. Ако e е инцидентно с връх $v' \in V'$, то втория край v'' на e също ще принадлежи на V', тъй като v', e, v'' е път в G и значи $v' \sim v''$. Следователно всяко ребро на G звързва два върха от един и същи клас на еквивалентност по релацията $\sim_G$. Така, ако в G има краен брой върхове, то $\sim_G$ разделя G на краен брой свързани подграфи

$$G_1 = (V_1, E_1), G_2(V_2, E_2), \ldots, G_k(V_k, E_k),$$

където V_1, V_2, …, V_k са различните класове на еквивалентност по релацията $\sim_G$, а $E_i = \{e \in E \mid e \subseteq V_i\}$, като при това

$$E_i \cap E_j = \emptyset,\ i \neq j \qquad \text{и} \qquad E = \bigcup_{i=1}^{k} E_i.$$

Графите G_1, G_2, … G_k наричаме *компоненти на свързаност* на G.

Твърдение 5.13. Нека G е граф с n върха и k компоненти на свързаност. Тогава в G ими поне $n - k$ ребра. При това, ако G е ацикличен, то в G има точно $n - k$ ребра.

Доказателство. Нека G_1, G_2, …, G_k са компонентите на свързаност на G, и нека G_i, $1 \leq i \leq k$, има n_i върха и m_i ребра. Тъй като G_i е свързан, в сила е $m_i \geq n_i - 1$. Тогава броят на ребрата на G е

$$m = m_1 + m_2 + \cdots + m_k \geq n_1 - 1 + n_2 - 1 + \cdots + n_k - 1 = n - k.$$

При това в случай, че G е ацикличен, всеки един от графите G_1, G_2, …, G_k е ацикличен и значи е дърво. В частност $m_i = n_i - 1$ за $1 \leq i \leq k$ и значи $m = n - k$.

□

5.6 Ациклични ориентирани графи

Ще казваме, че един ориентиран граф е *ацикличен*, ако в него няма цикли.

Твърдение 5.14. Нека $G = (V, E)$ е ориентиран граф. Тогава G е ацикличен тогава и само тогава, когато транзитивното затваряне E^+ на релацията E е (строга) частична наредба.

Доказателство. Да напомним, че $E^+ = \bigcup_{n>0} E^n$ е транзитивна релация. За произволно $v, v' \in V$ е в сила, че

$$(v, v') \in E^+ \iff (v, v') \in E^n \text{ за някое } n > 0 \iff \text{съществува път в } G \text{ с начало } v \text{ и край } v'.$$

Оттук

$$E^+ \text{ е ирефлексивна} \iff G \text{ е ацикличен.}$$

□

ГЛАВА 6

Крайни автомати

6.1 Формални езици. Основни операции

Под *азбука* ще разбираме произволно непразно множество, чиито елементи ще наричаме *букви* (или *символи*). Освен кирилицата, латиницата и гръцката азбука, като познат пример за азбука можем да посочим и цифрите $0, 1, \ldots, 9$, използвани за записване на естествените, целите и рационалните числа.

Ако Σ е азбука, под *дума* над Σ ще разбираме крайна редица от елементи на Σ. Празната редица (редицата без елементи) ще наричаме *празна дума* и ще бележим с ε (или λ). Така всяка дума от тълковния речник е дума над кирилицата. От друга страна думи над кирилицата са и

$$\varepsilon,\ \text{яояжфшг}, \text{ююююююююююююю}$$

и следователно не е необходимо една дума да има смисъл. Добавяйки към кирилицата препинателните знаци и знаци за интервал, всяко изречение на български език се превръща в дума. Ако добавим още и символ за нов ред, то тогава всеки текст, написан на български език също става една дума. Нека още отбележим, че десетичният запис на естествените числа са думи над азбуката $0, 1, \ldots, 9$. Множеството от всички думи над азбуката Σ ще бележим със Σ^*

Под *конкатенация* uv на думите $u = \alpha_1 \ldots \alpha_m$ и $v = \beta_1 \ldots \beta_n$, където $\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_n$ са букви ще разбираме думата $uv = \alpha_1 \ldots \alpha_m \beta_1 \ldots \beta_m$. Така думата 0123 може да се представи като конкатенация на следните непразни думи: 0 и 123, 01 и 23, 012 и 3.

Празната дума е неутрален елемент относно конкатенацията, т.е.

$$w = w\varepsilon = \varepsilon w, \quad \text{за всяка дума } w.$$

При това, операцията конкатенация е асоциативна, т.е.

$$(uv)w = u(vw).$$

Така за всяка азбука Σ, множеството Σ^* е моноид относно операцията конкатенация. Асоциативността на операцията конкатенация и неутрабността на думата ε ни позволяват да дефинираме степен на дума чрез следната рекурсия:

$$\begin{aligned} w^0 &= \varepsilon \\ w^{n+1} &= w^n w. \end{aligned}$$

Нека отбележим, че тъй като операцията конкатенация не е комутативна, то в общия случай $(uv)^n \neq u^n v^n$. Например,

$$(01)^5 = 0101010101 \neq 0000011111 = 0^5 1^5.$$

Под *дължина* $\operatorname{lh} w$ на думата w ще разбираме броят на буквите, които тя съдържа. Така

$$\operatorname{lh}(uv) = \operatorname{lh} u + \operatorname{lh} v$$

и

$$\operatorname{lh} w = 0 \iff w = \varepsilon.$$

Думите над азбуката Σ, т.е. елементите на Σ^* се получават и чрез следната рекурсия

1. $\varepsilon \in \Sigma^*$;

2. Ако $w \in \Sigma^*$ и $a \in \Sigma$, то $wa \in \Sigma^*$.

Така за да докажем, че всички думи над азбуката Σ имат дадено свойство P, достатъчно е да докажем, че

1. ε има свойството P;

2. Ако $w \in \Sigma^*$ има свойството P и $a \in \Sigma$, то тогава wa има свойството P.

В горния случай ще казваме, че сме извършили индукция по построението (или още по дължината) на думите над Σ.

Под *език* над азбуката Σ ще разбираме всяко множество от думи над азбуката Σ, т.е. езиците над Σ са точно подмножествата на Σ^*. Езиците над Σ са затворени относно теоретико-множествените операции обединение, сечение и разлика. Дефинираме конкатенацията на езиците L_1 и L_2 чрез

$$L_1L_2 = \{uv \mid u \in L_1 \text{ и } v \in L_2\}.$$

Ясно е, че езиците над дадена азбука са затворени относно операцията конкатенация на езици. Освен това операцията конкатенация е асоциативна. При това

$$L.\emptyset = \emptyset.L = \emptyset, \qquad L.\{\varepsilon\} = \{\varepsilon\}.L = L.$$

Асоциативността на операцията конкатенация и неутралността на езика $\{\varepsilon\}$ ни позволяват да дефинираме степен на език чрез следната рекурсия

$$\begin{aligned} L^0 &= \{\varepsilon\}, \\ L^{n+1} &= L^nL. \end{aligned}$$

Да отбележим, че $L^1 = \{\varepsilon\}L = L$. Нещо повече

$$w \in L^n \iff w = u_1u_2\ldots u_n, \text{ за някои } u_1, \ldots, u_n \in L. \tag{6.1}$$

Ще докажем (6.1) с индукция по n. За $n = 0$ и $n = 1$ твърдението е очевидно. Нека сега n е такова, че еквивалентност (6.1) е в сила. Тогава

$$\begin{aligned} w \in L^{n+1} &\iff w \in L^nL \\ &\iff w = uu_{n+1} \text{ за някои } u \in L^n,\ u_{n+1} \in L \\ &\iff w = \underbrace{u_1\ldots u_n}_{u}u_{n+1}, \text{ за някои } u_1, \ldots, u_n, u_{n+1} \in L. \end{aligned}$$

За всеки език L дефинираме L^* чрез

$$L^* = \bigcup_{n\in\mathbb{N}} L^n$$

Да отбележим, че тъй като L^0 и L^1 участват в обединението, то $\varepsilon \in L^*$ и $L \subseteq L^*$. По общо, съгласно (6.1) имаме

$$w \in L^* \iff w = u_1u_2\ldots u_n, \text{ за някои } n \geq 0 \text{ и } u_1, \ldots, u_n \in L.$$

За всеки език L дефинираме L^+ чрез

$$L^+ = \bigcup_{n>0} L^n.$$

Ясно е, че $L^* = \{\varepsilon\} \cup L^+$. Освен това,

$$w \in L^+ \iff w = u_1u_2\ldots u_n, \text{ за някои } n \geq 1 \text{ и } u_1, \ldots, u_n \in L.$$

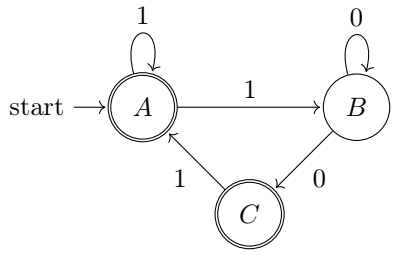
6.2 Дефиниция на краен автомат. Език на краен автомат

Краен автомат над азбуката Σ ще наричаме всяка четворка

$$\mathcal{A} = (Q, q_0, \Delta, F),$$

където Q е крайно множество, $q_0 \in Q$, $\Delta \subseteq Q \times \Sigma \times Q$ и $F \subseteq Q$. Елементите на Q ще наричаме *състояния*, елементите на F — *заключителни състояния*, а q_0 — *начално състояние*. Елементите на Δ ще наричаме *преходи* в автомата. Ако (q, a, q') е преход, ще казваме, че от q излиза преход с буквата a към q'. Ясно е, че двойката (Q, Δ) е етикетиран мултиграф. Ще бележим този граф с $G_{\mathcal{A}}$.

Пример 6.1. Нека $\Sigma = \{0, 1\}$ и нека $\mathcal{A}$ е крайният автомат над Σ имащ състояния $\{A, B, C\}$, начално състояние — A, заключителни състояния — $\{A, C\}$ и преходи $\{(A, 1, A), (A, 1, B), (B, 0, B), (B, 1, C), (C, 1, A)\}$. Тогава графът $G_{\mathcal{A}}$ изглежда така:



където заключителните състояния са отбелязани с удвоени окръжности, а началното състояние със стрелка без начало. Ясно е, че по този начин, графът еднозначно определя автомата $\mathcal{A}$.

Да разгледаме думата 111. Тогава, започвайки от началното състояние и движейки се по стрелките, съгласно техните етикети, тази дума задава два пътя в $G_{\mathcal{A}}$:

$$A, 1, A, 1, A, 1, A$$

$$A, 1, A, 1, A, 1, B.$$

От друга страна думата 101 задава един единствен път с начало A, а именно

$$A, 1, B, 0, C, 1, A,$$

а думата 1010 не задава нито един път с начало A. Формално, дефинираме понятието *състояние достижимо от дадено състояние посредством дума* по следния начин:

Определение 6.1. Нека $\mathcal{A}$ е краен автомат над Σ. Ще казваме, че състоянието q' *е достижимо в* $\mathcal{A}$ *от състояние* q *посредством думата* w *над* Σ и ще пишем $q \overset{w}{\underset{\mathcal{A}}{\longmapsto}} q'$, ако съществува път $q, a_1, q_1, \ldots, a_n, q_n = q'$ в $G_{\mathcal{A}}$, за който $w = a_1 a_2 \ldots a_n$.

Така за автомата от горния пример имаме

$$A \overset{111}{\underset{\mathcal{A}}{\longmapsto}} A,\ A \overset{111}{\underset{\mathcal{A}}{\longmapsto}} B,\ C \overset{111}{\underset{\mathcal{A}}{\longmapsto}} A,\ A \overset{101}{\underset{\mathcal{A}}{\longmapsto}} A,$$

но от друга страна посредством думата 1010 от нито едно състояние не може да се достигне нито едно състояние.

От дефиницията за достижимост и свойствата на пътища в граф директно следва

$$q \overset{\epsilon}{\underset{\mathcal{A}}{\longmapsto}} q;$$

$$q \overset{a}{\underset{\mathcal{A}}{\longmapsto}} q' \iff (q, a, q') \text{ е преход в } \mathcal{A};$$

$$q \overset{w'w''}{\underset{\mathcal{A}}{\longmapsto}} q'' \iff q \overset{w'}{\underset{\mathcal{A}}{\longmapsto}} q' \text{ и } q' \overset{w''}{\underset{\mathcal{A}}{\longmapsto}} q'' \text{ за някое състояние } q' \text{ на } \mathcal{A}.$$

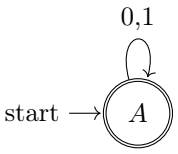
Ще казваме, че крайният автомат $\mathcal{A}$ *разпознава* думата w, ако от началното състояние на $\mathcal{A}$ посредством думата w може да се достигне до заключително състояние на $\mathcal{A}$. Под език $L(\mathcal{A})$ на автомата, ще разбираме множеството от всички думи, разпознавани от $\mathcal{A}$.

С други думи, езикът на крайния автомат $\mathcal{A}$ с начално състояние q_0 и заключителни състояния F е множеството

$$L(\mathcal{A}) = \{w \mid q_0 \overset{w}{\underset{\mathcal{A}}{\longmapsto}} q \text{ за някое } q \in F\}.$$

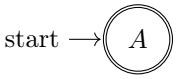
Ще казваме, че езикът L е *автоматен*, ако L е езикът на някой автомат.

Пример 6.2. Да разгледаме автомата



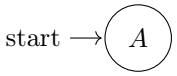
Ясно е, че посредством произволна дума $w \in \{0,1\}^*$ от началното състояние A се достига до същото състояние, което е заключително и следователно езикът на този автомат е $\{0,1\}^*$.

Пример 6.3. Да разгледаме автомата



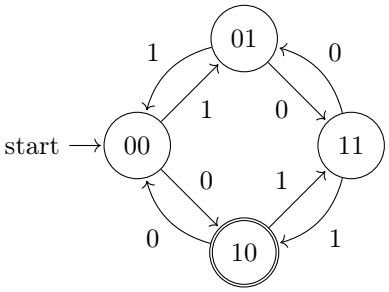
Единствената дума, с която от начално състояние може да се достигне до състояние, е ϵ. Тъй като началното състояние е заключително, езикът на евтомата е $\{\epsilon\}$.

Пример 6.4. Да разгледаме автомата



Той няма заключителни състояния и следователно неговият език е $\emptyset$.

Пример 6.5. Да разгледаме автомата $\mathcal{A}$



За да разберем, кои са думите, разпознавани от автомата, трябва да определим инвариантите на състоянията, т.е. със всяко състояние трябва да свържем свойство, такова че състоянието се достига посредством дадена дума от началното състояние тогава и само тогава, когато думата удовлетворява свойството. Нека първо забележим следното елементарно свойство на думите над азбуката $\{0,1\}$: за всяка дума $w \in \{0,1\}^*$ броят на нулите (единиците) в думите w и $w0$ ($w1$) са с различна четност. Сега да забележим, че преходите от всяко състояние на автомата са към друго състояние, но при два последователни прехода с една и съща буква се връщаме в състоянието, от което сме тръгнали. Това ни навежда на мисълта със състоянието ij да свържем свойството

$$P_{ij}(w) \overset{\text{деф}}{\iff} N_0(w) \equiv i(\text{mod}2) \text{ и } N_1(w) \equiv j(\text{mod}2),$$

където $N_0(w)$ и $N_1(w)$ са съответно броят на 0 и 1 в w.

Ще докажем, че P_{ij} е инварианта за състоянието ij с индукция по дължината на w. Ако $|w| = 0$, то $w = \epsilon$ и следователно в сила е $P_{00}(w)$. От друга страна единственото състояние, достижимо от началното, посредством ϵ е 00. Нека сега предположим, че n е такова, че за всяка дума $w \in \{0,1\}^*$ с дължина n е в сила

$$P_{ij}(w) \iff 00 \overset{w}{\underset{\mathcal{A}}{\longmapsto}} ij$$

и нека $w' \in \{0,1\}^*$ е с дължина $n+1$. Тогава $w' = wa$ за някоя дума w с дължина n и буква $a \in \{0,1\}$. Нека за определеност $a = 0$ (случаят $a = 1$ се разглежда аналогично). Да забележим, че за всеки две състояния kl и st имаме

$$kl \overset{0}{\underset{\mathcal{A}}{\longmapsto}} st \iff s = 1 - k \text{ и } l = t.$$

Следователно

$$P_{ij}(w') \iff P_{(1-i)j}(w) \iff 00 \overset{w}{\underset{\mathcal{A}}{\longmapsto}} (1-i)j \iff 00 \overset{w'}{\underset{\mathcal{A}}{\longmapsto}} (1-(1-i))j$$

и значи

$$P_{ij}(w') \iff 00 \xmapsto[\mathcal{A}]{w'} ij,$$

което трябваше да докажем.

Следователно

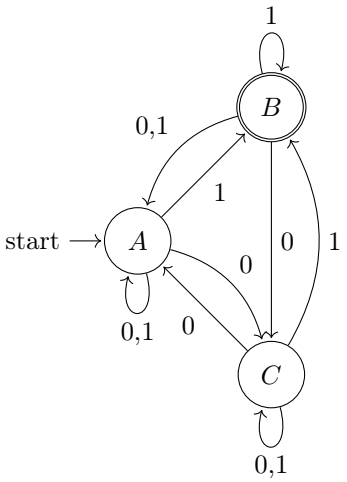
$$w \in L(\mathcal{A}) \iff 00 \xmapsto[\mathcal{A}]{w} 10 \iff P_{10}(w)$$

и значи

$$L(\mathcal{A}) = \{w \in \{0,1\}^* \mid w \text{ съдържа нечетен брой 0 и четен брой 1}\}.$$

6.3 Детерминирани и тотални крайни автомати

Да разгледаме автомата



За да докажем, че думата 101 е от езика на автомата, достатъчно е да посочим един път с начало A, край B и етикети 1, 0, 1, като например

$$A, 1, A, 0, A, 1, B.$$

От друга страна за да се уверим, че думата 110 не принадлежи на езика трябва да разгледаме всички 8 пътища, която тя поражда, а именно

$$A, 1, A, 1, A, 0, A$$

$$A, 1, A, 1, A, 0, C$$

$$A, 1, A, 1, B, 0, A$$

$$A, 1, A, 1, B, 0, C$$

$$A, 1, B, 1, B, 0, A$$

$$A, 1, B, 1, B, 0, C$$

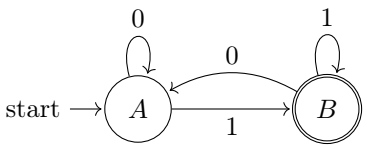
$$A, 1, B, 1, A, 0, A$$

$$A, 1, B, 1, A, 0, C$$

Нито един от тези пътища не завършва в заключителното състояние B и следователно думата не принадлежи на езика на автомата.

По-общо, за да се убедим, че една дума с дължина n не принадлежи на езика, ще трябва да разгледаме всичките 2^n пътя, които тя поражда.

Нека сега разгледаме автомата



Този автомат отново приема думата 101 и отхвърля думата 110, но този път всяка една от тези думи пораждат само един път в автомата. Изобщо, всяка дума поражда един единствен път в автомата и следователно въпросът за нейната принадлежност към езика на автомата се разрешава за точно толкова стъпки, колкото е дължината ѝ (стъпките необходими за прочитането ѝ).

Разликата между двата автомата е в това, че от всяко състояние на първия автомат излиза повече от един преход с всяка буква, а във втория автомат такива състояния няма. Така в първия автомат от всяко състояние имаме недерминизъм по кое ребро да тръгнем, докато във втория такъв недерминизъм няма — състоянието и буквата еднозначно определят следващото състояние. Автомати, подобни на втория автомат, ще наричаме детерминирани.

По-точон, ще казваме, че крайният автомат $\mathcal{A}$ е *детерминиран*, ако в $\mathcal{A}$ няма състояние, от което да излизат два прехода с една и съща буква, т.е. за всяка буква a и състояния q, q' и q'' на $\mathcal{A}$ от $q \xmapsto[\mathcal{A}]{a} q'$ и $q \xmapsto[\mathcal{A}]{a} q''$ следва $q' = q''$.

Твърдение 6.2. Нека $\mathcal{A}$ е детерминиран автомат, q е състояние на $\mathcal{A}$, а w е дума. Тогава съществува *най-много едно* състояние на $\mathcal{A}$, достижимо от q посредством w.

Доказателство. Нека означим с $R_{q,w}$ монажеството от всички състояния на $\mathcal{A}$, достижими от q посредством w, т.е.

$$R_{q,w} = \{q' \mid q \xmapsto[\mathcal{A}]{w} q'\}.$$

Ще докажем, че $|R_{q,w}| \leq 1$, с индукция по $|w|$. Нека първо $|w| = 0$. Тогава $w = \epsilon$ и следователно

$$|R_{q,w}| = |\{q\}| = 1.$$

Ако от друга страна $|w| = 1$, т.е. $w = a$ за някоя буква a, то неравенството $|R_{q,w}| \leq 1$, следва от детерминираността на $\mathcal{A}$.

Нека сега предположим, че твърдението е вярно за думи с дължина $n \geq 1$ и нека $|w| = n + 1$. Тогава $w = w'a$ за някоя дума w' и буква a и следователно

$$q \xmapsto[\mathcal{A}]{w} q' \iff q \xmapsto[\mathcal{A}]{w'} q'' \text{ и } q'' \xmapsto[\mathcal{A}]{a} q' \text{ за някое състояние } q''$$

Тогава

$$R_{q,w} = \bigcup_{q'' \in R_{q,w'}} R_{q'',a},$$

откъдето, предвид $|R_{q,w'}| \leq 1$ (съгласно индукционното предположение) и $|R_{q',a}| \leq 1$, получаваме $|R_{q,w}| \leq 1$.

□

Казваме, че крайният автомат $\mathcal{A}$ над Σ е *тотален*, ако от всяко състояние на $\mathcal{A}$ излиза преход с всяка буква от Σ, т.е. за всяко състояния q на $\mathcal{A}$ и всяка буква $a \in \Sigma$ съществува състояние q' на $\mathcal{A}$, такова че $q \xmapsto[\mathcal{A}]{a} q'$.

Аналогично на твърдението за детерминирани автомати, можем да докажем следното:

Твърдение 6.3. Нека $\mathcal{A}$ е тотален автомат над Σ, q е състояние на $\mathcal{A}$, а w е дума над Σ. Тогава съществува *поне едно* състояние на $\mathcal{A}$, достижимо от q посредством w.

Съчетавайки предните две твърдения, получаваме, че ако $\mathcal{A}$ е тотален и детерминиран автомат над Σ, q е състояние на $\mathcal{A}$, а w е дума над Σ, то съществува точно едно състояние, достижимо от q посредством w.

Теорема 6.4. Всеки автоматен език се разпознава от тотален и детерминиран краен автомат.

Доказателство. Нека L се разпознава от автомата $\mathcal{A} = (Q, q_0, \Delta, F)$. Да разгледаме автомата $\mathcal{A}_D$, чийто състояния са подмножествата на Q, началното му състояние е $\{q_0\}$, заключителните му състояния са онези подмножества на Q, които съдържат поне едно заключително състояние на $\mathcal{A}$ и

$$X \xmapsto[\mathcal{A}_D]{a} Y \iff Y = \{q' \mid q \xmapsto[\mathcal{A}]{a} q' \text{ за някое } q \in X\},$$

т.е. в $\mathcal{A}_D$ има преход с буквата a от състоянието X към състоянието Y тогава и само тогава, когато Y е колекцията от състоянията в $\mathcal{A}$, достижими с буквата a, от състояние, принадлежащо на X.

Ясно е, че $\mathcal{A}_D$ е тотален и детерминиран автомат. Нека първо разгледаме едно произволно състояние X на $\mathcal{A}_D$. Ще докажем, че за произволна дума $w \in \Sigma^*$ то

$$X \xmapsto[\mathcal{A}_D]{w} \{q' \mid q \xmapsto[\mathcal{A}]{w} q' \text{ за някое } q \in X\},$$

Ще докажем твърдението с индукция по $n = |w|$. За $n \leq 1$ твърението е очевидно. Нека сега предположим, че твърдението е вярно за $n \geq 1$ и нека w е дума с дължина $n + 1$, т.е. $w = w'a$ за някое $w' \in \Sigma^*$ и някое $a \in \Sigma$. Нека

$$X \xmapsto[\mathcal{A}_D]{w} Y$$

Тогава

$$X \xmapsto[\mathcal{A}_D]{w'} Z, \quad Z \xmapsto[\mathcal{A}_D]{a} Y.$$

Съгласно индукционното предположение

$$Z = \{q'' \mid q \xmapsto[\mathcal{A}]{w'} q'' \text{ за някое } q \in X\}$$

От друга страна, предвид дефиницията на преходите в $\mathcal{A}_D$

$$Y = \{q' \mid q'' \xmapsto[\mathcal{A}]{a} q' \text{ за някое } q'' \in Z\},$$

откъдето

$$Y = \{q' \mid q \xmapsto[\mathcal{A}]{w'} q'' \text{ и } q'' \xmapsto[\mathcal{A}]{a} q' \text{ за някои } q \in X,\ q'' \in Q\}.$$

Но

$$q \xmapsto[\mathcal{A}]{w} q' \iff q \xmapsto[\mathcal{A}]{w'} q'' \text{ и } q'' \xmapsto[\mathcal{A}]{a} q' \text{ за някое } q'' \in Q$$

и следонателно

$$X \xmapsto[\mathcal{A}_D]{w} \{q' \mid q \xmapsto[\mathcal{A}]{w} q' \text{ за някое } q \in X\},$$

с което индукцията е завършена.

Нека сега $w \in \Sigma^*$ и Y е състоянието, достижимо посредством w от началното състояние на $\mathcal{A}_D$ ($q_D = \{q_0\}$). Тогава

$$Y = \{q' \mid q_0 \xmapsto[\mathcal{A}]{w} q'\}$$

и следователно

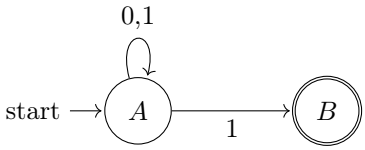
$$w \in L(\mathcal{A}_D) \iff Y \in F_D \iff \{q' \mid q_0 \xmapsto[\mathcal{A}]{w} q'\} \cap F \neq \emptyset \iff w \in L(\mathcal{A}),$$

откъдето

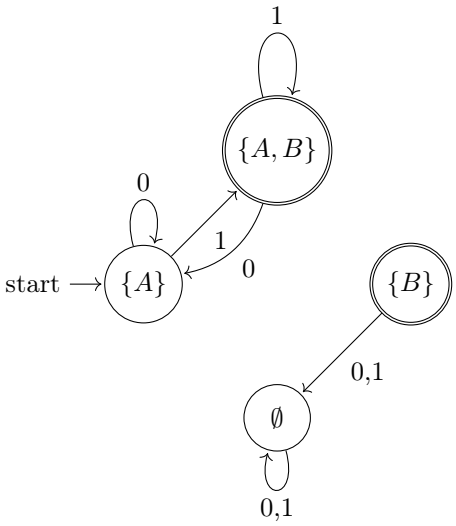
$$L(\mathcal{A}_D) = L(\mathcal{A}) = L.$$

□

Пример 6.6. Нека $\mathcal{A}$ е автоматът



Тогава $\mathcal{A}_D$ е автоматът



Следствие 6.5. Нека $L \subseteq \Sigma^*$ е автоматен. Тогава $\overline{L} = \Sigma^* \setminus L$ също е автоматен.

Доказателство. Нека $\mathcal{A} = (Q, q_0, \Delta, F)$ е тотален и детерминиран краен автомат с $L(\mathcal{A}) = L$. Нека $\mathcal{A}' = (Q, q_0, \Delta, GF')$, където $F' = Q \setminus F$. Нека $w \in \Sigma^*$ е произволна и нека $q \in Q$ е единственото състояние, за което $(q_0, w) \vdash_{\mathcal{A}}^* q$ (автоматът $\mathcal{A}$ е тотален и детерминиран). Тогава

$$w \notin L \iff q \notin F \iff q \in F' \iff q \in L(\mathcal{A}').$$

Следователно $\overline{L} = L(\mathcal{A}')$.

□

6.4 Обединение, конкатенация, звезда на Клини и сечение на автоматни езици

Теорема 6.6. Обединението на автоматни езици е автоматен език.

Доказателство. Нека $L_i \subseteq \Sigma^*$, $i = 1, 2$, са автоматни езици и нека $\mathcal{A}_i = (Q_i, q_{0i}, \Delta_i, F_i)$, $i = 1, 2$, са крайни автомати над Σ, такива че $L_i = L(\mathcal{A}_i)$, $i = 1, 2$. Нека още $Q_1 \cap Q_2 = \emptyset$. Да разгледаме крайния автомат $\mathcal{A} = (Q, q_0, \Delta, F)$, където $q_0 \notin Q_1 \cup Q_2$ и

$$\begin{aligned} Q &= Q_1 \cup Q_2 \cup \{q_0\} \\ \Delta &= \Delta_1 \cup \Delta_2 \cup \{(q_0, a, q) \mid (q_{01}, a, q) \in \Delta_1 \text{ или } (q_{02}, a, q) \in \Delta_2\} \\ F &= \begin{cases} F_1 \cup F_2 \cup \{q_0\}, & \text{ако } q_{01} \in F_1 \text{ или } q_{02} \in F_2 \\ F_1 \cup F_2, & \text{иначе} \end{cases} \end{aligned}$$

Така, състоянията и преходите на $\mathcal{A}_1$ и $\mathcal{A}_2$ са състояния и преходи на $\mathcal{A}$. При това всеки преход на $\mathcal{A}$, с изключение на тези, излизащи от новото начално състояние q_0, е преход или на $\mathcal{A}_1$ или на $\mathcal{A}_2$. Оттук за всяко състояние q на $\mathcal{A}_i$, $i = 1, 2$, е в сила

$$q \xmapsto[\mathcal{A}]{w} q' \iff q \xmapsto[\mathcal{A}_i]{w} q'. \qquad (*)$$

Нека първо докажем, че $L_1 \cup L_2 \subseteq L(\mathcal{A})$. За целта нека $w \in L_1 \cup L_2$. Ако $w = \epsilon$, то тогава $q_{01} \in F_1$ или $q_{02} \in F_2$, откъдето $q_0 \in F$ и значи $w = \epsilon \in L(\mathcal{A})$. Нека сега $w \neq \epsilon$. Тогава $w = aw'$ за някое $a \in \Sigma$ и $w' \in \Sigma^*$. Нека за определеност $w \in L_1$. Тогава

$$q_{01} \xmapsto[\mathcal{A}_1]{w} q$$

за някое $q \in F_1$, откъдето

$$q_{01} \xmapsto[\mathcal{A}_1]{a} q' \text{ и } q' \xmapsto[\mathcal{A}_1]{w'} q$$

за някое $q' \in Q_1$. Оттук

$$q_0 \xmapsto[\mathcal{A}]{a} q' \text{ и } q' \xmapsto[\mathcal{A}]{w'} q$$

и следователно

$$q_0 \xmapsto[\mathcal{A}]{w} q,$$

откъдето $w \in L(\mathcal{A})$.

За обратното включване нека $w \in L(\mathcal{A})$. Ако $w = \epsilon$, то $q_0 \in F$, откъдето $q_{01} \in F_1$ или $q_{02} \in F_2$ и следователно $w = \epsilon \in L_1 \cup L_2$. Нека сега $w \neq \epsilon$. Тогава $w = aw'$ за някое $a \in \Sigma$ и $w' \in \Sigma^*$. Тъй като $w \in L(\mathcal{A})$, в сила е

$$q_0 \xmapsto[\mathcal{A}]{w} q$$

за някое $q \in F$, откъдето

$$q_0 \xmapsto[\mathcal{A}]{a} q' \text{ и } q' \xmapsto[\mathcal{A}]{w'} q$$

за някое $q' \in Q$. Предвид това, дефиницията на преходите на $\mathcal{A}$, $q' \in Q_1$ или $q' \cup Q_2$. Нека за определеност $q' \in Q_1$. Тогава $(*)$ ни дава

$$(q', w') \xmapsto[\mathcal{A}_1]{w'} q.$$

От друга страна от $q_0 \overset{a}{\underset{\mathcal{A}}{\longmapsto}} q'$ имаме

$$q_{01} \overset{a}{\underset{\mathcal{A}_1}{\longmapsto}} q'$$

и следователно

$$q_{01} \overset{w}{\underset{\mathcal{A}_1}{\longmapsto}} q.$$

От $q \in F$ получаваме $q \in F_1$ и следователно $w \in L_1$.

□

Теорема 6.7. Конкатенацията на автоматни езици е автоматен език.

Доказателство. Нека $L_1, L_2 \subseteq \Sigma^*$ са автоматни езици и нека $\mathcal{A}_1 = (Q_1, q_{01}, \Delta_1, F_1)$ и $\mathcal{A}_2 = (Q_2, q_{02}, \Delta_2, F_2)$ са крайни автомати над Σ, такива че $L_1 = L(\mathcal{A}_1)$ и $L_2 = L(\mathcal{A}_2)$. Нека още $Q_1 \cap Q_2 = \emptyset$. Да разгледаме крайния автомат $\mathcal{A} = (Q, q_{01}, \Delta, F)$, където

$$\begin{aligned} Q &= Q_1 \cup Q_2 \\ \Delta &= \Delta_1 \cup \Delta_2 \cup \{(q, a, q') \mid q \in F_1 \text{ и } (q_{02}, a, q') \in \Delta_2\} \\ F &= \begin{cases} F_1 \cup F_2, & \text{ако } q_{02} \in F_2 \\ F_2, & \text{иначе} \end{cases} \end{aligned}$$

Отново, както в предното доказателство, $\mathcal{A}$ съдържа състоянията и преходите на $\mathcal{A}_1$ и $\mathcal{A}_2$. При това, новите преходи на $\mathcal{A}$ имат за начало заключително състояние на $\mathcal{A}_1$ и край състояние на $\mathcal{A}_2$. Тъй като началното състояние на $\mathcal{A}$ и $\mathcal{A}_1$ съвпадат, всяко изпълнение на $\mathcal{A}$ върху дума w започва като изпълнение в $\mathcal{A}_1$. Ако в даден момент от изпълнението на $\mathcal{A}$ върху w попаднем в състояние от $\mathcal{A}_2$, то от там нататък изпълнението ще се извършва изцяло в рамките на $\mathcal{A}_2$.

Нека първо докажем, че $L_1L_2 \subseteq L(\mathcal{A})$. За целта нека $w \in L_1L_2$. Тогава $w = w_1w_2$ за някои $w_1 \in L_1$ и $w_2 \in L_2$, откъдето

$$q_{01} \overset{w_1}{\underset{\mathcal{A}_1}{\longmapsto}} q_1 \text{ и } q_{02} \overset{w_2}{\underset{\mathcal{A}_2}{\longmapsto}} q_2$$

за някои $q_1 \in F_1$ и $q_2 \in F_2$. Тогава

$$q_{01} \overset{w_1}{\underset{\mathcal{A}}{\longmapsto}} q_1 \text{ и } q_{02} \overset{w_2}{\underset{\mathcal{A}}{\longmapsto}} q_2$$

Ако $w_2 = \epsilon$, то $q_{02} \in F_2$, откъдето $q_1 \in F_1 \subseteq F$ и значи $w = w_1 \in L(\mathcal{A})$. В противен случай $w_2 = aw_2'$ и следователно

$$q_{02} \overset{a}{\underset{\mathcal{A}_2}{\longmapsto}} q' \text{ и } q' \overset{w_2'}{\underset{\mathcal{A}_2}{\longmapsto}} q_2.$$

Съгласно дефиницията на Δ в сила е

$$q_1 \overset{a}{\underset{\mathcal{A}}{\longmapsto}} q'$$

и следователно

$$q_{01} \overset{w}{\underset{\mathcal{A}}{\longmapsto}} q_2.$$

Оттук $w \in L(\mathcal{A})$ и следователно $L_1L_2 \subseteq L(\mathcal{A})$.

За обратното включване нека $w = a_1a_2 \ldots a_n \in L(\mathcal{A})$, където $a_1, a_2, \ldots, a_n \in \Sigma$ $(n \geq 0)$. Тогава в G_A съществува път

$$q_{01} = q_0, a_1, q_1, \ldots, a_n, q_n$$

където $q_n \in F$. Нека първо предположим, че за всяко $1 \leq i \leq n$ в сила е $q_i \in Q_1$. Тогава $q_{01} \overset{w}{\underset{\mathcal{A}_1}{\longmapsto}} q_n$. Освен това $q_n \in Q_1 \cap F$, откъдето $q_n \in F_1$ и $F_1 \subseteq F$. Тогава $w \in L_1$, $\epsilon \in L_2$ и следователно $w = w\epsilon \in L_1L_2$. Нека сега не е вярно, че $q_i \in Q_1$ за всяко $1 \leq i \leq n$ и нека j е най-малкото, за което $q_j \in Q_2$. Тогава

$$q_{01} \overset{a_1a_2\ldots a_{j-1}}{\underset{\mathcal{A}_1}{\longmapsto}} q_{j-1}$$

и тъй като в $\mathcal{A}$ няма преходи от Q_2 към Q_1, в сила е

$$q_j \overset{a_{j+1}a_{j+2}\ldots a_n}{\underset{\mathcal{A}_2}{\longmapsto}} q_n.$$

От дефиницията на Δ следва, че $q_{j-1} \in F_1$ и $(q_{02}, a_j, q_j) \in \Delta_2$. Оттук

$$a_1 a_2 \dots a_{j-1} \in L_1 \text{ и } a_j a_{j+1} \dots a_n \in L_2$$

и следователно $w \in L_1 L_2$.

□

Теорема 6.8. Звездата на Клини на автоматен език е автоматен език.

Доказателство. Нека $L \subseteq \Sigma^*$ е автоматен език и нека $\mathcal{A} = (Q, q_0, \Delta, F)$ е краен автомат над Σ, такъв че $L = L(\mathcal{A})$. Да разгледаме крайния автомат $\mathcal{A}' = (Q', q_0', \Delta', F')$, където $q_0' \notin Q$ и

$$\begin{aligned} Q' &= Q \cup \{q_0'\} \\ \Delta &= \Delta \cup \{(q, a, q') \mid q \in F' \text{ и } (q_0, a, q') \in \Delta\} \\ F' &= F \cup \{q_0'\}. \end{aligned}$$

Нека първо докажем, че $L^* \subseteq L(\mathcal{A}')$. За целта нека $w \in L^*$. Ако $w = \epsilon$, то $w \in L(\mathcal{A}')$, тъй като $q_0' \in F'$ В противен случай $w = w_1 w_2 \dots w_n$ за някое $n \geq 1$ и непразни $w_1, \dots, w_n \in L$. Нека $w_i = a_i w_i'$ за $i = 1, 2, \dots, n$, където $a_i \in \Sigma$, а $w_i' \in \Sigma^*$. Тогава за $1 \leq i \leq n$

$$q_0 \xmapsto[\mathcal{A}]{a_i} q_i \text{ и } q_i \xmapsto[\mathcal{A}]{w_i'} q_i'$$

за някои $q_i \in Q$ и $q_i' \in F$. От дефиницията на Δ в сила е

$$q_{i-1}' \xmapsto[\mathcal{A}']{a_i} q_i \text{ и } q_i \xmapsto[\mathcal{A}']{w_i'} q_i'$$

за $0 \leq i \leq n$, откъдето

$$q_0' \overset{w_1 w_2 \dots w_n}{\underset{\mathcal{A}'}{\longmapsto}} q_n$$

и следователно $w \in L(\mathcal{A})$.

За обратното включване, нека $w \in L(\mathcal{A})$. Ако $w = \epsilon$, няма какво да доказваме и за това нека предположим, че $w = a_1 a_2 \dots a_n$, където $n \geq 1$ и $a_1, a_2, \dots, a_n \in \Sigma$. Нека

$$q_0', a_1, q_1, \dots, a_n, q_n$$

е път в $G_{\mathcal{A}'}$. Нека $0 = i_0 < i_1 < \dots < i_k = n$ са всички индекси, за които $(q_{i_s}, a_{i_s+1}, q_{i_s+1}) \notin \Delta$. Тогава, съгласно дефиницията на Δ', имаме $(q_0, a_{i_s+1}, q_{i_s+1}) \in \Delta$ и $q_{i_s} \in F$ за $s \geq 1$. Следователно за всяко $0 \leq s \leq k-1$

$$q_0, a_{i_s+1}, q_{i_s+1}, \dots, a_{i_{s+1}}, q_{i_{s+1}}$$

е път в $G_{\mathcal{A}}$, откъдето

$$a_{i_s+1} a_{i_s+2} \dots a_{i_{s+1}} \in L \text{ за } s = 0, 1, \dots, n-1.$$

Така $w \in L^*$ и следователно $L(\mathcal{A}) \subseteq L^*$.

□

Теорема 6.9. Сечението на два автоматни езика е автоматен език.

Доказателство. Нека $L_1, L_2 \subseteq \Sigma^*$ са автоматни езици. Тъй като автоматните езици са затворени относно обединение и допълнение, имаме, че

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

е автоматен език.

Сега ще дадем и директно доказателство на теоремата. Нека $\mathcal{A}_1 = (Q_1, q_{01}, \Delta_1, F_1)$ и $\mathcal{A}_2 = (Q_2, q_{02}, \Delta_2, F_2)$ са крайни автомати над Σ, такива че $L_1 = L(\mathcal{A}_1)$ и $L_2 = L(\mathcal{A}_2)$. Да разгледаме крайния автомат $\mathcal{A} = (Q, q_0, \Delta, F)$, където $q_0 = (q_{01}, q_{02})$

$$\begin{aligned} Q &= Q_1 \times Q_2 \\ \Delta &= \{((q_1, q_2), a, (q_1', q_2')) \mid (q_1, a, q_1') \in \Delta_1 \text{ и } (q_2, a, q_2') \in \Delta_2\} \\ F &= F_1 \times F_2 \end{aligned}$$

Първо с индукция по $|w|$ ще докажем, че

$$(q_1, q_2) \xmapsto[\mathcal{A}]{w} (q_1', q_2') \iff q_1 \xmapsto[\mathcal{A}_1]{w} q_1' \text{ и } q_2 \xmapsto[\mathcal{A}_2]{w} q_2'$$

За $|w| \leq 1$ еквивалентността е очевидна. Нека сега еквивалентността е вярна за $n \geq 1$ и нека $|w| = n + 1$, т.е. $w = w'a$ за някое $w' \in \Sigma^*$ с $|w'| = n$ и $a \in \Sigma$. Тогава

$$\begin{aligned}
(q_1, q_2 \xmapsto[\mathcal{A}]{w} (q_1', q_2') \quad &\iff \quad (q_1, q_2) \xmapsto[\mathcal{A}]{w'} (q_1'', q_2'') \text{ и } (q_1'', q_2'') \xmapsto[\mathcal{A}]{a} (q_1', q_2') \\
&\iff \quad q_1 \xmapsto[\mathcal{A}_1]{w'} q_1'',\ q_2 \vdash_{\mathcal{A}_2} w'q_2'',\ q_1'' \xmapsto[\mathcal{A}_1]{a} q_1' \text{ и } q_2'' \xmapsto[\mathcal{A}_2]{a} q_2' \\
&\iff \quad q_1 \xmapsto[\mathcal{A}_1]{w} q_1' \text{ и } q_2 \xmapsto[\mathcal{A}_2]{w} q_2'
\end{aligned}$$

Тогава за всяка дума $w \in \Sigma^*$ имаме

$$\begin{aligned}
w \in L(\mathcal{A}) \quad &\iff \quad (q_{01}, q_{02}) \xmapsto[\mathcal{A}]{w} (q_1', q_2') \text{ за някои } q_1' \in F_1, q_2' \in F_2 \\
&\iff \quad q_{01} \xmapsto[\mathcal{A}_1]{w} q_1' \text{ и } q_{02} \xmapsto[\mathcal{A}_2]{w} q_2' \text{ за някои } q_1' \in F_1, q_2' \in F_2 \\
&\iff \quad w \in L_1 \text{ и } w \in L_2.
\end{aligned}$$

□

6.5 Регулярни езици. Теорема на Клини

Нека Σ е азбука, която не съдържа символите $+$, $*$, $\emptyset$, както и лява и дясна скоба. *Регулярните изрази* над Σ са специални думи над азбуката $\Sigma \cup \{+, *, \emptyset, (,)\}$, които се дефинират чрез следната рекурсия:

(i) Символът $\emptyset$, както и всяка буква $a \in \Sigma$ са регулярни изрази;

(ii) Ако r_1 и r_2 са регулярни изрази, то $(r_1 r_2)$ и $(r_1 + r_2)$ са регулярни изрази;

(iii) Ако r е регулярен израз то r^* е регулярен израз.

Пример. Нека $\Sigma = \{0, 1\}$. Регулярни изрази над Σ са

$$\emptyset,\ 0,\ (01),\ ((((01) + (11)) + (10)^*)(1(11)))$$

Както се виждда от последния пример, скобите (които формално са крайно необходими) утежняват много записа на регулярните изрази. За да го улесним, ще спазваме обичайните правила при работа със скоби, а именно - ще изпускаме скобите, когато те групират една и съща операция, а така също ще считаме, че степенуването (със звезда) има приоритет пред умножението (конкатенацията), което на свой ред има приоритет пред умножението. Така, регулярният израз

$$((((01) + (11)) + (10)^*)(1(11)))$$

ще записваме като

$$(01 + 11 + (10)^*)111$$

.

Езикът $L(r)$ на един регулярен израз r над Σ дефинираме чрез следната рекурсия (съгласно рекурсивната дефиниция на регулярните изрази):

(i) $L(\emptyset) = \emptyset$, $L(a) = \{a\}$ за $a \in \Sigma$;

(ii) $L(r_1 + r_2) = L(r_1) \cup L(r_2)$ и $L(r_1 r_2) = L(r_1)L(r_2)$;

(iii) $L(r^*) = L(r)^*$.

Ще казваме, че езикът L над Σ е *регулярен*, ако $L = L(r)$ за някой регулярен израз r над Σ.

Пример. Нека $\Sigma = \{0, 1\}$. Тогава

- $L = \{w \in \Sigma^* \mid w \text{ завършва на } 011\}$ е регулярен, тъй като
$$L = L((0 + 1)^*011)$$
- $L = \{w \in \Sigma^* \mid w \text{ започва с } 011\}$ е регулярен, тъй като
$$L = L(011(0 + 1)^*)$$
- $L = \{w \in \Sigma^* \mid w \text{ съдържа } 011\}$ е регулярен, тъй като
$$L = L((0 + 1)^*011(0 + 1)^*)$$
- $L = \{w \in \Sigma^* \mid w \text{ съдържа четен брой } 1\}$ е регулярен, тъй като
$$L = L(0^* + (0^*10^*1)^*0^*)$$
- $L = \{w \in \Sigma^* \mid \text{в } w \text{ всяка буква 1 е последвана от (поне една) буква } 0\}$ е регулярен, тъй като
$$L = L((0 + 10)^*).$$

Класът на *регулярните* езици над Σ, ще означаваме с Reg_Σ. Очевидно, Reg_Σ съдържа празния език, езиците, състоящи се от точно една еднобуквена дума, и е затворен относно операциите обединение, конкатенация и звезда на Клини. Следващото твърдение показва, че Reg_Σ е най-малкият клас с това свойство.

Твърдение 6.10. Нека $\mathcal{C} \subseteq \mathcal{P}(\Sigma^*)$ е клас от езици. Тогава ако $\emptyset \in \mathcal{C}$, $\{a\} \in \mathcal{C}$ за всяко $a \in \Sigma$ и $\mathcal{C}$ е затворен относно обединение, конкатенация и звезда на Клини, то $\mathrm{Reg}_\Sigma \subseteq \mathcal{C}$, т.е. $\mathcal{C}$ съдържа всички регулярни езици.

Доказателство. Да допуснем, че $\mathrm{Reg}_\Sigma \not\subseteq \mathcal{C}$ и нека r е регулярен израз над Σ с възможно най-къса дължина, за който $L(r) \notin \mathcal{C}$. Тъй като $\emptyset \in \mathcal{C}$, и $\{a\} \in \mathcal{C}$ за всяко $a \in \Sigma$, то $r \neq \emptyset$ и $r \neq a$ за никое $a \in \Sigma$. Тогава $r = (r_1 + r_2)$, $r = (r_1 r_2)$ или $r = r_1^*$. Нека например $r = (r_1 + r_2)$, за някои регулярни изрази r_1 и r_2 над Σ. Тогава r_1 и r_2 са по-кратки регулярни изрази от r и значи $L(r_1) \in \mathcal{C}$ и $L(r_2) \in \mathcal{C}$, откъдето $L(r) \in \mathcal{C}$ (защото $L(r) = L(r_1) \cup L(r_2)$) — противоречие. В останалите два случая разсъждавайки аналогично, отново стигаме до противоречие. Следователно допускането ни $\mathrm{Reg}_\Sigma \not\subseteq \mathcal{C}$ е грешно и значи $\mathrm{Reg}_\Sigma \subseteq \mathcal{C}$.

□

Следствие 6.11. Регулярните езици са автоматни.

Доказателство. Нека Σ е азбука. От една страна езикът на $\mathcal{A}_\emptyset = (\{A\}, A, \emptyset, \emptyset)$ е $\emptyset$, а от друга за всяко $a \in \Sigma$ за крайния автомат

$$\mathcal{A}_a = (\{A, B\}, A, \{(A, a, B)\}, \{B\})$$

над Σ е валидно $L(\mathcal{A}_a) = \{a\}$. Освен това автоматните езици са затворени относно операциите обединение, конкатенация и звезда на Клини и следователно, съгласно предното твърдение, регулярните езици са автоматни.

□

Теорема 6.12 (Клини)**.** Автоматните езици са регулярни.

Доказателство. Нека $L \subseteq \Sigma^*$ е автоматен език и нека $\mathcal{A} = (Q, q_0, \Delta, F)$ е краен автомат над Σ с $L(\mathcal{A}) = L$. Без ограничение

$$Q = \{0, 1, 2, \ldots, n-1\}$$

и $q_0 = 0$. Нека за всяко $0 \leq i, j \leq n-1$ и всяко $0 \leq k \leq n$ с R_{ij}^k означим множеството от всички думи над Σ, чрез които можем да се придвижим от състоянието i до състоянието j без да преминаваме през състояние $s \geq k$, т.е.

$$R_{ij}^k = \{w \in \Sigma^* \mid i \xmapsto[\mathcal{A}]{w} j, \text{ минавайки само през състояния } s < k\}$$

Тогава

$$R_{ii}^0 = \{\epsilon\} \cup \{a \mid (i, a, i) \in \Delta\} \text{ и } R_{ij}^0 = \{a \mid (i, a, j) \in \Delta\} \text{за } i \neq j$$

и следователно за всяко $0 \leq i, j \leq n-1$ езикът R_{ij}^0 е регулярен. От друга страна за всяко $0 \leq i, j \leq n-1$ и всяко $0 \leq k \leq n-1$ в сила е

$$R_{ij}^{k+1} = R_{ij}^k \cup R_{ik}^k (R_{kk}^k)^* R_{kj}^k$$

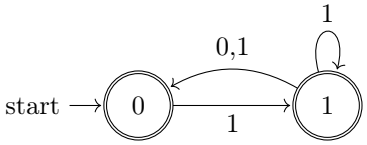
Оттук езиците R_{ij}^k са регулярни за всяко $0 \leq i, j \leq n-1$ и всяко $0 \leq k \leq n$. От друга страна

$$L(\mathcal{A}) = \bigcup_{i \in F} R_{0i}^n$$

и следователно $L(\mathcal{A})$ е регулярен.

□

Пример. Да разгледаме автомата



Следвайки доказателството на теоремата на Клини намираме регулярни изрази за R_{ij}^k за $0 \leq i, j \leq 1$ и $0 \leq k \leq 1$.

$r_{00}^0 = \emptyset^*$	$r_{00}^1 = \emptyset^* + \emptyset^*(\emptyset^*)^*\emptyset^*$
$r_{01}^0 = 1$	$r_{01}^1 = 1 + \emptyset^*(\emptyset^*)^*1$
$r_{10}^0 = 0 + 1$	$r_{10}^1 = 0 + 1 + (0+1)(\emptyset^*)^*\emptyset^*$
$r_{11}^0 = \emptyset^* + 1$	$r_{11}^1 = (0+1) + (0+1)(\emptyset^*)^*1$

и следователно

$$\begin{aligned} r_{00}^2 &= (\emptyset^* + \emptyset^*(\emptyset^*)^*\emptyset^*) + \\ &\quad (1 + \emptyset^*(\emptyset^*)^*1)((0+1) + (0+1)(\emptyset^*)^*1)^*(0 + 1 + (0+1)(\emptyset^*)^*\emptyset^*) \\ r_{01}^2 &= (1 + \emptyset^*(\emptyset^*)^*1) + \\ &\quad (1 + \emptyset^*(\emptyset^*)^*1)((0+1) + (0+1)(\emptyset^*)^*1)^*((0+1) + (0+1)(\emptyset^*)^*1) \end{aligned}$$

и регулярният израз за езика, разпознаван от автомата, е

$$r = r_{00}^2 + r_{01}^2.$$

6.6 Лема за разрастването на регулярните езици

Теорема 6.13 (Лема за разрастването)**.** Нека $L \subseteq \Sigma^*$ е регулярен език. Тогава съществува естетвено число $n_0 \geq 1$, такова че за всяко $w \in L$ с $|w| > n_0$ съществуват $x, y, z \in \Sigma^*$, такива че $w = xyz$, $|xy| \leq n_0$, $|y| \geq 1$ и за всяко $i \geq 0$ в сила е $xy^iz \in L$.

Доказателство. Нека $\mathcal{A} = (Q, q_0, \Delta, F)$ е краен автомат над Σ, такъв че $L(\mathcal{A}) = L$. Да положим $n_0 = |Q|$. Нека сега $w \in L$ с $|w| = m > n_0$ и нека $w = a_1a_2 \ldots a_m$, където $a_1, a_2, \ldots, a_m \in \Sigma$. Тогава съществува път

$$q_0, a_1, q_1, \ldots, a_m, q_m$$

в $G_\mathcal{A}$ с $q_m \in F$. Тъй като $m > n_0$, то поне две от състоянията в горната редица съвпадат. Нека s е най-голямото, за което $q_i \neq q_j$ за всяко $0 \leq i < j \leq s$. Ясно е, че $s \leq n_0$ и освен това $q_k = q_s$ за някое $k < s$. Полагаме

$$x = a_1 \ldots a_k,\ y = a_{k+1} \ldots a_s,\ z = a_{s+1} \ldots a_m.$$

Тогава $|xy| = s \leq n_0$ и $|y| = s - k \geq 1$. При това

$$q_0 \overset{x}{\underset{\mathcal{A}}{\longmapsto}} q_k,\ q_k \overset{y}{\underset{\mathcal{A}}{\longmapsto}} q_k \text{ и } q_k \overset{z}{\underset{\mathcal{A}}{\longmapsto}} q_m,$$

откъдето, за всяко $i \geq 0$ в сила е $xy^iz \in L$.

□

Лемата за разрастването дава необходимо условие един език L да бъде регулярен. Обичайно, тя се използва в контрапозиция, а именно:

Нека $L \subseteq \Sigma^*$ е такъв, че за всяко $n \geq 1$, съществува $w \in L$, такъв че $|w| > n$ и за всеки избор на $x, y, z \in \Sigma^*$, за който $w = xyz$, $|xy| \leq n$ и $|y| \geq 1$, съществува $i \geq 0$, такова че $xy^iz \notin L$. Тогава L не е регулярен.

Пример. Ще докажем, че езикът $L = \{a^kb^k | k \geq 0\}$ не е регулярен. За целта нека $n \geq 1$ и да разгледаме думата

$$w = a^nb^n.$$

Имаме $w \in L$ и $|w| = 2n > n$. Нека сега x, y и z са думи, такива че

$$w = xyz,\ |xy| \leq n \text{ и } |y| \geq 1.$$

От $xyz = a^nb^n$ и $|xy| \leq n$ следва, че $xy = a^m$, откъдето $y = a^k$ за някое $k \geq 1$ (защото $|y| \geq 1$). Тогава при $i = 0$ имаме

$$xy^iz = xz = a^{n-k}b^n \notin L.$$

Пример. Ще докажем, че езикът $L = \{ab^k c^k | k \geq 0\}$ не е регулярен. За целта нека $n \geq 1$ и да разгледаме думата

$$w = ab^n c^n.$$

Имаме $w \in L$ и $|w| = 2n + 1 > n$. Нека сега x, y и z са думи, такива че

$$w = xyz,\ |xy| \leq n \text{ и } |y| \geq 1.$$

Тогава xy не съдържа буквата c, откъдето думата $xy^0z = xz$ съдържа n на брой c, но дължината ѝ е по-малка от $2n + 1$ и значи не принадлежи на L.

Забележка. Условието на лемата за разрастването не е достатъчно условие един език да бъде регулярен. Наистина, нека $\Sigma = \{a, b, c\}$,

$$\begin{aligned} L_1 &= \{ab^k c^k \mid k \geq 0\} \\ L_2 &= \{a^k w \mid k \neq 1 \text{ и } w \in \{b, c\}^*\} \end{aligned}$$

и нека

$$L = L_1 \cup L_2.$$

Първо ще докажем, че L изпълнява условието за разрастване. Наистина, нека $n_0 = 2$. Да разгледаме произволна дума $w \in L$ с $|w| > n_0$. Тогава $w \in L_1$ или $w \in L_2$. Нека първо $w \in L_1$, т.е.

$$w = ab^k c^k \text{ за някое } k \geq 0.$$

Избираме

$$x = \varepsilon,\ y = a \text{ и } z = b^k c^k.$$

В сила е

$$xyz = w,\ |xy| = 1 \leq n_0 \text{ и } |y| = 1.$$

При това, за всяко $i \geq 0$ е в сила

$$xy^i z = a^i b^k c^k$$

и следователно

$$xy^1 z \in L_1 \text{ и } xy^i z \in L_2 \text{ при } i \neq 1.$$

Нека сега $w \in L_2$, т.е.

$$w = a^k w' \text{ за някои } k \neq 1 \text{ и } w' \in \{b, c\}^*$$

Ако $k = 0$, избираме

$$x = \varepsilon,\ y = \alpha \text{ и } z = w'',\ \text{където } \alpha \in \{b, c\} \text{ и } w' = \alpha w''\ (|w'| = |w| > 2).$$

В сила е

$$xyz = w,\ |xy| = 1 \leq n_0 \text{ и } |y| = 1.$$

При това, за всяко $i \geq 0$ е изпълнено, че

$$xy^i z \in \{b, c\}^* \text{ и значи } xy^i z \in L_2.$$

Ако $k = 2$, избираме

$$x = \varepsilon,\ y = aa \text{ и } z = w'.$$

В сила е

$$xyz = w,\ |xy| = 2 \leq n_0 \text{ и } |y| = 2.$$

При това, за всяко $i \geq 0$ е изпълнено, че

$$xy^i z = a^{2i} w' \text{ и значи } xy^i z \in L_2.$$

Накрая, ако $k \geq 3$, избираме

$$x = \varepsilon,\ y = a \text{ и } z = a^{k-1} w'.$$

В сила е

$$xyz = w,\ |xy| = 1 \leq n_0 \text{ и } |y| = 1.$$

При това, за всяко $i \geq 0$ е изпълнено, че

$$xy^i z = a^{k+i-1} w',\ \text{като } k + i - 1 \geq 2 \text{ и значи } xy^i z \in L_2.$$

Следователно L удовлетворява условието на лемата за разрастването. Въпреки това L *не е* регулярен. Наистина, да допуснем, че L е регулярен. Тогава и езикът $L \cap \{a\}\{b, c\}^*$ също е регулярен. Но $L \cap \{a\}\{b, c\}^* = L_1$, който съгласно предния пример не е регулярен — противоречие. Следователно L не е регулярен.

6.7 Обобщени автомати — ε-преходи и повече от едно начални състояния

ε-преходи.

Краен автомат с ε-преходи над Σ наричаме всеки краен автомат, в който освен преходи с букви от Σ допускаме и преходи с празната дума, т.е. всяка четворка

$$\mathcal{A} = (Q, q_0, \Delta, F),$$

за която Q е крайно множество, $q_0 \in Q$, $F \subseteq Q$ и

$$\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q.$$

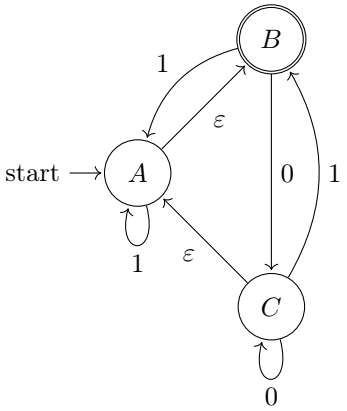
Дефинираме релацията $q \overset{w}{\underset{\mathcal{A}}{\longmapsto}} q'$, аналогично на дефиницията на случая без ϵ-преходи, отчитайки това, че ϵ е неутрален елемент относно конкатенацията, а именно: $q \overset{w}{\underset{\mathcal{A}}{\longmapsto}} q'$ тогава и само тогава, когато в $G_{\mathcal{A}}$ има път

$$s_0, a_1, s_1, \ldots, a_{n-1}, s_{n-1}, a_n, s_n, \text{ (т.е. } (s_i, a_{i+1}, s_{i+1}) \in \Delta),$$

такъв че $s_0 = q$, $s_n = q'$ и $w = a_1 a_2 \ldots a_n$.[1] Съответно, дефинираме езика на $\mathcal{A}$ чрез

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid q_0 \overset{w}{\underset{\mathcal{A}}{\longmapsto}} q \text{ за някое } q \in F\}.$$

Пример. Да разгледаме автомата



Тогава думата 0 принадлежи на езика на автомата, т.к. $A \overset{0}{\underset{}{\longmapsto}} B$, защото в автомата имаме път

$$A, \varepsilon, B, 0, C, \varepsilon, A, \varepsilon, B.$$

Твърдение 6.14. За всеки краен автомат $\mathcal{A}$ с ε-преходи, съществува краен автомат $\mathcal{A}'$ (без ε-преходи), такъв че $L(\mathcal{A}) = L(\mathcal{A}')$.

Доказателство. Ще покажем, как да елиминираме ε-преходите. Нека $\mathcal{A}$ е автомат с ε-преходи над Σ. Образуваме краен автомат $\mathcal{A}'$ (без ε-преходи), премахвайки ε-преходи от $\mathcal{A}$ и добавяйки преходи с $a \in \Sigma$ от състоянието q към q' винаги, когато $q \overset{a}{\underset{\mathcal{A}}{\longmapsto}} q'$. [2] По-формално, ако

$$\mathcal{A} = (Q, q_0, \Delta, F),$$

то образуваме

$$\mathcal{A}' = (Q, q_0, \Delta', F'),$$

където за всяко $a \in \Sigma$ и всяко $q, q' \in Q$ имаме

$$(q, a, q') \in \Delta' \iff q \overset{a}{\underset{\mathcal{A}}{\longmapsto}} q'.$$

[1] Т.к. може $a_i = \varepsilon$, то $|w| \leq n$

[2] Например, в горния автомат трябва да добавим преходи с буквата 0 от състоянието A към състоянията A, B и C

Тогава за всяка буква $a \in \Sigma$ и всеки две състояния $q, q' \in Q$ имаме

$$q \xmapsto[\mathcal{A}]{a} q' \iff q \xmapsto[\mathcal{A}']{a} q',$$

откъдето за всяка непразна дума $w \in \Sigma^*$ е в сила

$$q \xmapsto[\mathcal{A}]{w} q' \iff q \xmapsto[\mathcal{A}']{w} q'.$$

Освен това, дефинираме

$$F' = \begin{cases} F \cup \{q_0\}, & q_0 \xmapsto[\mathcal{A}]{\varepsilon} q \text{ за някое } q \in F \\ F, & \text{иначе} \end{cases}.$$

По този начин

$$\varepsilon \in L(\mathcal{A}) \iff \varepsilon \in L(\mathcal{A}'),$$

а за непразни думи w имаме

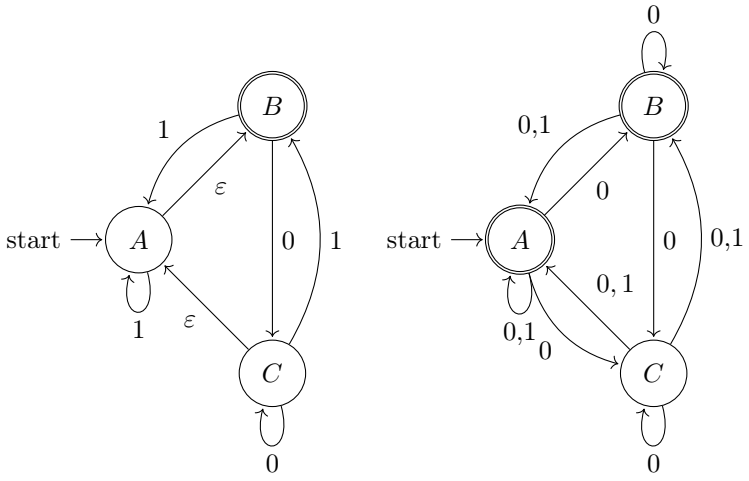
$$\begin{aligned} w \in L(\mathcal{A}) &\iff q_0 \xmapsto[\mathcal{A}]{w} q \text{ за някое } q \in F \\ &\iff q_0 \xmapsto[\mathcal{A}]{w} q \text{ за някое } q \in F \iff w \in L(\mathcal{A}') \end{aligned}$$

и значи

$$L(\mathcal{A}) = L(\mathcal{A}').$$

□

Прилагайки конструкцията от доказатлеството върху автомата от предния пример, получаваме



Основната трудност при елиминирането на ε-преходите е установяването, кои са състоянията q', за които $q \xmapsto[\mathcal{A}]{a} q'$ за $a \in \Sigma$. Един систематичен подход към този проблем е да образуваме *ε-затварянето*

$$\mathcal{E}(q) = \{q' \mid q \xmapsto[\mathcal{A}]{\varepsilon} q'\}$$

за всяко състояние q на автомата. [3] Намирането на $\mathcal{E}(q)$ можем да извършим чрез следната рекурсивна схема. Образуваме $\mathcal{E}_0(q) = \{q\}$ и $\mathcal{E}_{k+1}$ (за $k \geq 0$), добавяйки към $\mathcal{E}_k$ състоянията, към които има ε-преход от състояние в $\mathcal{E}_k$. Така получаваме редица

$$\mathcal{E}_0(q) \subseteq \mathcal{E}_1(q) \subseteq \cdots \subseteq Q.$$

Тъй като Q е крайно, то горната редица се стабилизира от дадено място нататък. По-точно, съществува s_0, такова че

$$\mathcal{E}_i(q) \subsetneq \mathcal{E}_{i+1}(q) \text{ за } i < s_0 \text{ и } \mathcal{E}_i(q) = \mathcal{E}_{i+1}(q) \text{ за } i \geq s_0.$$

Тогава

$$\mathcal{E}(q) = \mathcal{E}_{s_0}(q).$$

След като сме определили ε-затварянето на всяко едно състояние, вече можем да определим състоянията q', за които $q \xmapsto[\mathcal{A}]{a} q'$, за $a \in \Sigma$, чрез

$$q \xmapsto[\mathcal{A}]{a} q' \iff \exists s \exists s' (s \in \mathcal{E}(q) \;\&\; q' \in \mathcal{E}(s') \;\&\; (s, a, s') \in \Delta).$$

[3] За горния пример е в сила $\mathcal{E}(A) = \{A, B\}$, $\mathcal{E}(B) = \{B\}$, $\mathcal{E}(C) = \{C, A, B\}$

Използвайки ε-преходите, можем да използваме по-прости алгоритми за обединение, конкатенация и звезда на Клини за автоматни езици.

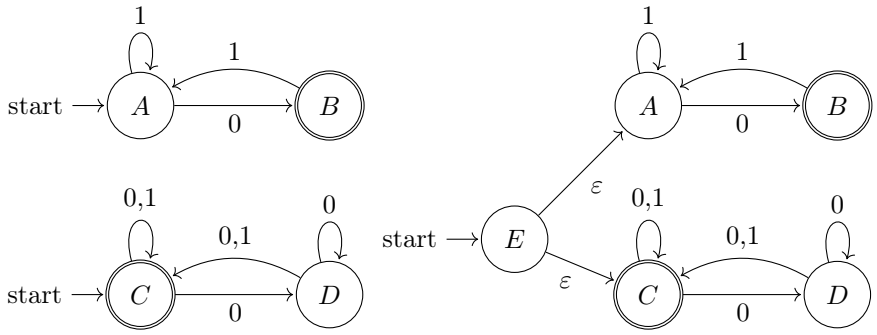
Обединение. При дадени автомати $\mathcal{A}_1$ и $\mathcal{A}_2$, без общи състояния, за да образуваме автомат $\mathcal{A}$, разпознаващ $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$, достатъчно е да добавим ново начално състояние, от което да излизат ε-преходи към началните състояния на двата автомата, т.е. при

$$\mathcal{A}_1 = (Q_1, q_{01}, \Delta_1, F_1) \text{ и } \mathcal{A}_2 = (Q_2, q_{02}, \Delta_2, F_2) \text{ с } Q_1 \cap Q_2 = \emptyset$$

образуваме $\mathcal{A} = (Q, q_0, \Delta, F)$, където

$$\begin{aligned} Q &= Q_1 \cup Q_2 \cup \{q_0\}, \\ q_0 &\notin Q_1 \cup Q_2, \\ \Delta &= \Delta_1 \cup \Delta_2 \cup \{(q_0, \varepsilon, q_{01}), (q_0, \varepsilon, q_{02})\}, \\ F &= F_1 \cup F_2. \end{aligned}$$

Например,



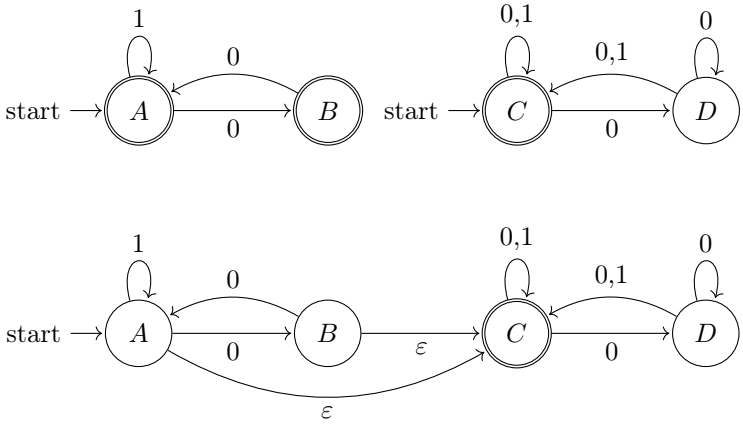
Конкатенация. При дадени автомати $\mathcal{A}_1$ и $\mathcal{A}_2$, без общи състояния, за да образуваме автомат $\mathcal{A}$, разпознаващ $L(\mathcal{A}_1)L(\mathcal{A}_2)$, достатъчно е да запазим за начално началното състояние на първия автомат, да отменим финалността на заключителните състояния на първия автомат и да добавим ε-преходи от заключителните състояния на първия автомат към началното състояние на втория автомат, т.е. при

$$\mathcal{A}_1 = (Q_1, q_{01}, \Delta_1, F_1) \text{ и } \mathcal{A}_2 = (Q_2, q_{02}, \Delta_2, F_2) \text{ с } Q_1 \cap Q_2 = \emptyset$$

образуваме $\mathcal{A} = (Q, q_0, \Delta, F)$, където

$$\begin{aligned} Q &= Q_1 \cup Q_2, \\ q_0 &= q_{01}, \\ \Delta &= \Delta_1 \cup \Delta_2 \cup \{(q, \varepsilon, q_{02}) \mid q \in F_1\}, \\ F &= F_2. \end{aligned}$$

Например,



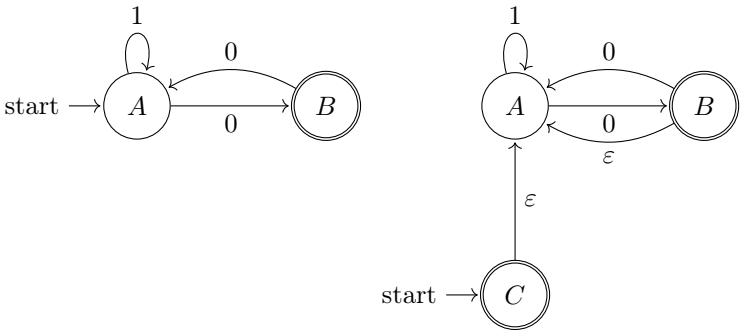
Звезда на Клини. При даден автомат $\mathcal{A}$ за да образуваме автомат $\mathcal{A}'$, разпознаващ $L(\mathcal{A})^*$, достатъчно е да добавим ново начално състояние, което е финално, и от всяко финално състояние да добавим ε-преход към началното състояние на оригиналния автомат, т.е. при

$$\mathcal{A} = (Q, q_0, \Delta, F)$$

образуваме $\mathcal{A}' = (Q', q_0', \Delta', F')$, където

$$\begin{array}{rcl} Q' & = & Q \cup q_0', \\ q_0' & \notin & Q, \\ \Delta & = & \Delta \cup \{(q, \varepsilon, q_0) \mid q \in F'\}, \\ F' & = & F \cup \{q_0'\}. \end{array}$$

Например,



Множество от начални състояния.

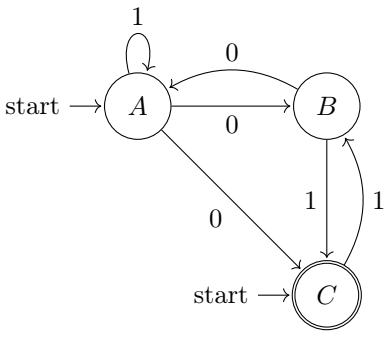
Освен автомати с едно начално състояние се разглеждат и автомати с повече от едно начално състояние. По-точно, краен автомат над азбуката Σ с повече от едно начални състояния наричаме четворката

$$\mathcal{A} = (Q, I, \Delta, F),$$

където Q, Δ и F са както при дефиницията на обикновения краен автомат, а $I \subseteq Q$ е поне двуелементно множество (от начални състояния). Релацията $q \xmapsto[\mathcal{A}]{w} q'$ се дефинира по същия начин както при обикновенните крайни автомати. Разликата е при дефиницията на език на автомата, като в този случай допускаме успешното изпълнение (изпълнение, завършващо във финално състояние) да започва от кое да е от началните състояния. По-точно, дефинираме

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid q \xmapsto[\mathcal{A}]{w} q' \text{ за някои } q \in I \text{ и } q' \in F\}.$$

Пример. Да разгледаме автомата



Тогава

$$01 \in L(\mathcal{A}) \text{ и } 11 \in L(\mathcal{A}),$$

защото имаме съответно успешни изпълнения

$$A, 0, B, 1, C \text{ и } C, 1, B, 1, C.$$

Отново, лесно се вижда, че наличието на повече от едно начално състояние не увеличава разпознавателната способност на автоматите.

Твърдение 6.15. За всеки краен автомат $\mathcal{A}$ с повече от едно начално състояние съществува краен автомат $\mathcal{A}'$ с ε преходи, такъв че $L(\mathcal{A}) = L(\mathcal{A}')$.

Доказателство. Нека $\mathcal{A}$ е краен автомат с повече от едно начално състояние. Свеждаме този автомат до автомат $\mathcal{A}'$ с едно начално състояние, добавяйки ново начално състояние (което е единствено за $\mathcal{A}'$), от което излизат ε-преходи към началните състояния на $\mathcal{A}$. По-точно, ако

$$\mathcal{A} = (Q, I, \Delta, F),$$

то $\mathcal{A}' = (Q', q_0, \Delta', F')$, където

$$\begin{aligned} Q' &= Q \cup \{q_0\}, \\ q_0 &\notin Q, \\ \Delta' &= \Delta \cup \{(q_0, \varepsilon, q) \mid q \in I\}, \\ F' &= F. \end{aligned}$$

Тъй като единствените преходи, свързващи началното състояние q_0 със състоянията на $\mathcal{A}$ са ε-преходите към състоянията от I, то за всяка буква $a \in \Sigma$

$$q_0 \xmapsto[\mathcal{A}']{a} q' \iff q \xmapsto[\mathcal{A}]{a} q' \text{ за някое } q \in I,$$

откъдето за всяка непразна дума $w \in \Sigma^*$ е в сила

$$q_0 \xmapsto[\mathcal{A}']{w} q' \iff q \xmapsto[\mathcal{A}]{w} q' \text{ за някое } q \in I.$$

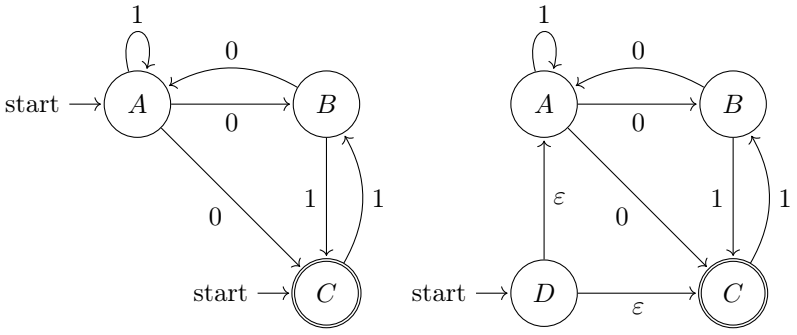
Оттук и очевидното $\varepsilon \in L(\mathcal{A}') \iff \varepsilon \in L(\mathcal{A})$ следва

$$w \in L(\mathcal{A}') \iff w \in L(\mathcal{A}),$$

за всяка дума $w \in \Sigma^*$.

□

За примера по-горе, прилагайки алгоритъма от доказателството, получаваме



Обединение. С помощтта на автомати с повече от едно начално състояние можем лесно да реализираме алгоритъм за обединение. Наистина, нека $\mathcal{A}_1$ и $\mathcal{A}_2$ са два крайни автомата (с едно или повече начални състояния), които нямат общи състояния. Тогава автомата $\mathcal{A}$, който представлява „механичното“ обединение на двата автомата, е такъв, че $L(\mathcal{A}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$. По-точно, ако

$$\mathcal{A}_1 = (Q_1, I_1, \Delta_1, F_1) \text{ и } \mathcal{A}_2 = (Q_2, I_2, \Delta_2, F_2) \text{ с } Q_1 \cap Q_2 = \emptyset,$$

то $\mathcal{A} = (Q, I, \Delta, F)$, където

$$\begin{aligned} Q &= Q_1 \cup Q_2, \\ I &= I_1 \cup I_2, \\ \Delta &= \Delta_1 \cup \Delta_2, \\ F &= F_1 \cup F_2. \end{aligned}$$

6.8 Минимални тотални детерминирани крайни автомати

Нека L е език над азбуката Σ. Релацията $\sim_L$ на Майхил-Нероуд дефинираме по следния начин. Казваме, че две думи $w, w' \in \Sigma^*$ *са еквивалентни над* L и ще пишем $w \sim_L w'$, ако

$$\forall u \in \Sigma^* (wu \in L \iff w'u \in L)$$

В случай, че езикът L е произволен би било твърде сложно, а в някои случаи и невъзможно да установим, дали две думи са в еквивалентни над L. Не така стоят нещата, когато L е регулярен.

Лема 6.16. Нека $\mathcal{A} = (Q, q_0, \Delta, F)$ е тотален и детерминиран автомат над азбуката Σ. Нека $q \in Q$ и $w, w' \in \Sigma^*$ са такива, че $q_0 \overset{w}{\underset{\mathcal{A}}{\longmapsto}} q$ и $q_0 \overset{w'}{\underset{\mathcal{A}}{\longmapsto}} q$. Тогава $w \sim_{L(\mathcal{A})} w'$.

Доказателство. Тъй като $\mathcal{A}$ е тотален и детерминиран, за всяка дума $u \in \Sigma^*$ и всяко $q' \in Q$

$$q_0 \overset{wu}{\underset{\mathcal{A}}{\longmapsto}} q' \iff q \overset{u}{\underset{\mathcal{A}}{\longmapsto}} q' \iff q_0 \overset{w'u}{\underset{\mathcal{A}}{\longmapsto}} q',$$

откъдето

$$\begin{aligned} wu \in L(\mathcal{A}) &\iff q_0 \overset{wu}{\underset{\mathcal{A}}{\longmapsto}} q' \text{ за някое } q' \in F \\ &\iff q_0 \overset{w'u}{\underset{\mathcal{A}}{\longmapsto}} q' \text{ за някое } q' \in F \\ &\iff w'u \in L(\mathcal{A}). \end{aligned}$$

Следователно

$$w \sim_{L(\mathcal{A})} w'.$$

□

Ясно е, че релацията $\sim_L$ е рефлексивна, симетрична и транзитивна и следователно $\sim_L$ е релация на еквивалентност. Така всеки език L над Σ разбива множеството Σ^* на непресичащи се класове на еквивалентност по релацията $\sim_L$. По-точно, всеки език $L \subseteq \Sigma^*$ дефинира фактормножеството

$$\Sigma^* / \sim_L = \{[w]_L \mid w \in \Sigma^*\},$$

където

$$[w]_L = \{w' \in \Sigma^* \mid w \sim_L w'\}.$$

Първата ни цел е да установим връзка между броя на елемнетите на $\Sigma^* / \sim_L$ и регулярността на L. Ще започнем с това, че ако един език е регулярен, то множеството $\Sigma^* / \sim_L$ е крайно. За да формулираме, максимално точно това твърдение въвеждаме следното понятие: Казваме, че състоянието q на автомата $\mathcal{A} = (Q, q_0, \Delta, F)$ е *достижимо*, ако $q_0 \overset{w}{\underset{\mathcal{A}}{\longmapsto}} q$ за някое $w \in \Sigma^*$.

Твърдение 6.17. Нека L е регулярен език над Σ. Тогава множеството $\Sigma^* / \sim_L$ е крайно, като при това броят на елементите му не надминава броя на достижимите състояния на кой да е детерминиран и тотален автомат, разпознаващ L.

Доказателство. Нека $\mathcal{A} = (Q, q_0, \Delta, F)$ е тотален и детерминиран автомат над Σ с $L(\mathcal{A}) = L$. Нека в $\mathcal{A}$ има n достижими състояния и нека те са $q_0, q_1, \ldots, q_{n-1}$. Да фиксираме $w_0, w_1, \ldots, w_{n-1}$, такива че

$$q_0 \overset{w_i}{\underset{\mathcal{A}}{\longmapsto}} q_i \text{ за } 0 \leq i \leq n-1.$$

Тогава за произволна дума $w \in \Sigma^*$ в сила е

$$q_0 \overset{w}{\underset{\mathcal{A}}{\longmapsto}} q_i \text{ за някое } 0 \leq i \leq n-1,$$

откъдето предвид горната лема имаме

$$w \in [w_i] \text{ за някое } 0 \leq i \leq n-1.$$

Така

$$\Sigma^*/\sim_L = \{[w_0]_L, [w_1]_L, \ldots, [w_{n-1}]_L\}$$

и следонателно $\Sigma^*/\sim_L$ има най-много n елемента.

□

Нека сега предположим, че множството $\Sigma^*/\sim_L$ е крайно. Да разгледаме крайния автомат $\mathcal{A}_L = (Q_L, q_{0L}, \Delta_L, F_L)$, където

$$\begin{aligned} Q_L &= \Sigma^*/\sim_L, \\ q_{0L} &= [\varepsilon]_L, \\ \Delta_L &= \{([w]_L, a, [wa]_L) \mid w \in \Sigma^* \text{ и } a \in \Sigma\}, \\ F_L &= \{[w]_L \mid w \in L\}. \end{aligned}$$

Ясно, че $\mathcal{A}_L$ е тотален. Твърдим, че $\mathcal{A}_L$ е детерминиран. Наистина нека $q, q', q'' \in Q_L$ и $a \in \Sigma$ са такива, че

$$(q, a, q') \in \Delta_L \text{ и } (q, a, q'') \in \Delta_L.$$

Тогава

$$q' = [w'a]_L \text{ и } q'' = [w''a]_L$$

за някои $w', w'' \in \Sigma^*$ такива че

$$[w']_L = q = [w'']_L.$$

Но тогава $w' \sim_L w''$ и значи $w'a \sim_L w''a$. Оттук $q' = q''$ и следователно $\mathcal{A}_L$ е детерминиран.

Твърдение 6.18. Нека $L \subseteq \Sigma^*$ е такъв, че множеството $\Sigma^*/\sim_L$ е крайно. Тогава $L(\mathcal{A}_L) = L$. В частност L е регулярен.

Доказателство. Първо ще докажем, че за произволни $u, w \in \Sigma^*$

$$[u]_L \xmapsto[\mathcal{A}_L]{w} [uw]_L.$$

Ще проведем индукция по $|w|$. Ако $|w| \leq 1$, твърдението е очевидно. Нека сега $|w| = n > 1$ и твърдението е вярно за думи с дължина $n-1$. Нека $w = w'a$, където $w' \in \Sigma^*$ и $a \in \Sigma$. Тогава $|w'| = n-1$ и съгласно индукционното предположение

$$[u]_L \xmapsto[\mathcal{A}_L]{w'} [uw']_L.$$

От друга страна $[uw']_L \xmapsto[\mathcal{A}_L]{a} [uw'a]_L$ и следователно

$$[u]_L \xmapsto[\mathcal{A}_L]{w'a} [uw'a]_L,$$

което трябваше да докажем.

Оттук за произволна дума $w \in \Sigma^*$ в сила е

$$q_{0L} \xmapsto[\mathcal{A}_L]{w} [w]_L$$

и следователно

$$w \in L(\mathcal{A}_L) \iff [w]_L \in F_L.$$

От друга страна $[w]_L \in F_L$ тогава и само тогава, когато $w \sim_L w'$ за някое $w' \in L$. Но от $w \sim_L w'$ следва

$$w' \in L \iff w'\epsilon \in L \iff w\epsilon \in L \iff w \in L,$$

откъдето

$$w \in L(\mathcal{A}_L) \iff w \in L.$$

□

Ще казваме, че един тотален и детерминиран автомат е *минимален*, ако всеки друг тотален и детерминиран автомат, разпознаващ същия език, има не по-малко състояния. Сега от предните две твърдения непосредствено следва следната теорема.

Теорема 6.19 (Майхил-Нероуд)**.** Езикът $L \subseteq \Sigma^*$ е регулярен тогава и само тогава, когато множеството $\Sigma^*/\sim_L$ е крайно. При това, ако L е регулярен, то $\mathcal{A}_L$ е минимален тотален и детерминиран автомат.

Дотук видяхме, че за всеки регулярен език има минимален тотален и детерминиран краен автомат, който го разпознава. Сега ще докажем, че този минимален автомат е единствен. За целта първо ще дадем следната дефиниция.

Определение 6.20. Нека $\mathcal{A} = (Q, q_0, \Delta, F)$ и $\mathcal{A}' = (Q', q_0', \Delta', F')$ са крайни автомати над Σ. Ще казваме, че изображението $\varphi : Q \to Q'$ е изоморфизъм, ако са изпълнени следните четири свойства:

(i) φ е биекция;

(ii) $\varphi(q_0) = q_0'$;

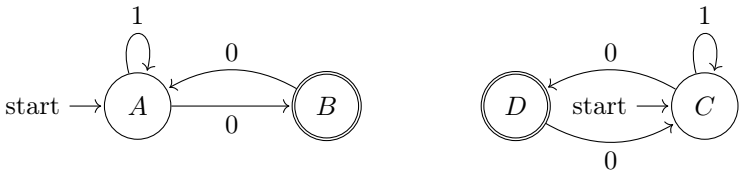
(iii) За всяко $q_1, q_2 \in Q$ и всяко $a \in \Sigma$

$$(q_1, a, q_2) \in \Delta \iff (\varphi(q_1), a, \varphi(q_2)) \in \Delta';$$

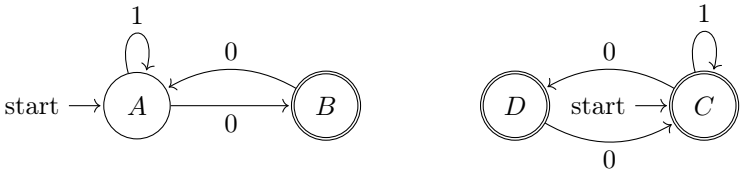
(iv) $\varphi(F) = F'$.

В случай, че съществува изоморфизъм между $\mathcal{A}$ и $\mathcal{A}'$, ще казваме, че $\mathcal{A}$ и $\mathcal{A}'$ са изоморфни и ще пишем $\mathcal{A} \cong \mathcal{A}'$

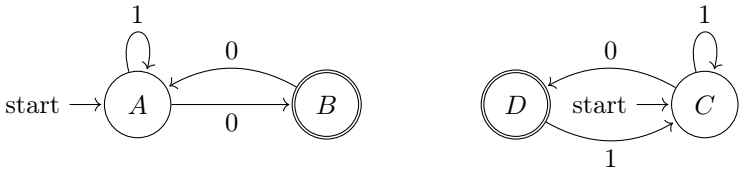
Интуитивно, два автомата са изоморфни тогава и само тогава, когато те се различават единствено и само по имената на състоянията си. Така например автоматите



са изоморфни, а автоматите



както и автоматите



не са изоморфни.

В този смисъл е ясно, че ако $\mathcal{A} \cong \mathcal{A}'$, то $L(\mathcal{A}) = L(\mathcal{A}')$. Освен това

1. $\mathcal{A} \cong \mathcal{A}$ (посредством изображението Id_Q)
2. Ако $\mathcal{A} \cong \mathcal{A}'$ (посредством φ), то $\mathcal{A}' \cong \mathcal{A}$ (посредством φ^{-1})
3. Ако $\mathcal{A} \cong \mathcal{A}'$ (посредством φ) и $\mathcal{A}' \cong \mathcal{A}''$ (посредством ψ), то $\mathcal{A} \cong \mathcal{A}''$ (посредством $\psi \circ \varphi$)

Теорема 6.21. За всеки регулярен език $L \subseteq \Sigma^*$ с точност до изоморфизъм съществува единствен минимален тотален и детерминиран автомат, разпознаващ L.

Доказателство. Нека L е регулярен език над Σ и нека $\mathcal{A} = (Q, q_0, \Delta, F)$ е минимален тотален и детерминиран автомат, разпознаващ L. Ще докажем, че $\mathcal{A} \cong \mathcal{A}_L$. Да разгледаме $\varphi : Q \to Q_L$, действащо по правилото

$$\varphi(q) = [w]_L \iff q_0 \xmapsto[\mathcal{A}]{w} q.$$

Първо трябва да докажем, че φ е коректно дефинирано. Нека първо да докажем, че φ е тотално. Тъй като $\mathcal{A}$ и $\mathcal{A}_L$ са минимални за един и същи език, то множествата Q и Q_L имат равен брой елементи и следователно, съгласно първото твърдение, всяко състояние на $\mathcal{A}$ е достижимо. Тогава за всяко $q \in Q$ съществува $w \in \Sigma^*$, такова че $q_0 \xmapsto[\mathcal{A}]{w} q$ и следователно φ е тотална.

Остава да докажем, че $\varphi(q)$ е еднозначно определено. За целта нека $w, w' \in \Sigma^*$ са такива, че

$$q_0 \xmapsto[\mathcal{A}]{w} q \text{ и } q_0 \xmapsto[\mathcal{A}]{w'} q.$$

Тогава, съгласно лемата,

$$w \sim_L w',$$

откъдето

$$[w]_L = [w']_L.$$

Следователно $\varphi(q)$ е еднозначно определено и значи φ е коректно дефинирана.

Да забележим, че тъй като $\mathcal{A}$ е тотален автомат, то за всяко $w \in \Sigma^*$ съществува $q \in Q$, такова че $q_0 \xmapsto[\mathcal{A}]{w} q$. Оттук φ е сюрективно и следователно φ е биективно, защото Q и Q' са крайни множества с равен брой елементи. Така доказахме (i).

За (ii) да забележим, че $q_0 \xmapsto[\mathcal{A}]{\varepsilon} q_0$ и следователно

$$\varphi(q_0) = [\varepsilon]_L = q_{0L}.$$

Нека сега докажем (iii). Нека $q, q' \in Q$ и $a \in \Sigma$. Нека още

$$\varphi(q) = [w]_L \text{ и } \varphi(q') = [w']_L,$$

т.е.

$$q_0 \xmapsto[\mathcal{A}]{w} q \text{ и } q_0 \xmapsto[\mathcal{A}]{w'} q'.$$

Тогава

$$\begin{aligned}
(q, a, q') \in \Delta &\iff q_0 \xmapsto[\mathcal{A}]{wa} q' \\
&\iff [wa]_L = \varphi(q') = [w']_L \\
&\iff ([w]_L, a, [w']) \in \Delta_L \\
&\iff (\varphi(q), a, \varphi(q')) \in \Delta_L.
\end{aligned}$$

Остава да докажем (iv). Нека $q \in Q$ и нека $\varphi(q) = [w]_L$, т.е. $q_0 \xmapsto[\mathcal{A}]{w} q$. Тогава

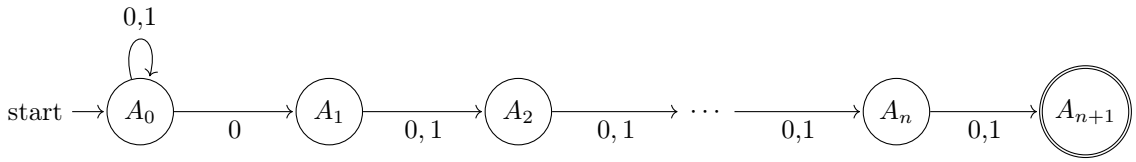
$$q \in F \iff w \in L \iff [w]_L \in F_L \iff \varphi(q) \in L.$$

□

Пример. Нека $n > 0$ е фиксирано естествено число и нека L е езикът, състоящ се от всички думи над азбуката $\{0, 1\}$, които имат дължина поне $n + 1$, и $n + 1$-вата буква отзад напред в w е 0. С други думи

$$L = \{u0v \mid u, v \in \{0, 1\}^* \text{ и } |v| = n\}.$$

Ясно е, че езикът L е регулярен. При това той се разпознава от недетрминирания автомат



Нека сега разгледаме релацията $\sim_L$. Нека $v = a_0a_1 \dots a_n$ и $w = b_0b_1 \dots b_n$ са две различни думи с дължина $n+1$ и нека i е такова, че $a_i \neq b_i$, като например $a_i = 0$, $b_i = 1$. Тогава $v0^i \in L$, но $w0^i \notin L$ и значи $v \not\sim_L w$. Следователно думите с дължина $n+1$ пораждат различни класове на еквивалентност по отношение на $\sim$.

От друга страна, ако $v = a_1a_2 \dots a_k$ е дума с дължина $k < n+1$, то $v \sim_L 1^{n+1-k}v$, и следователно всяка дума с дължина, по-малка от $n+1$ попада в клас на еквивалентност породен от дума с дължина $n+1$.

Накрая, ако $v = a_1a_2 \dots a_k$ е дума с дължина $k > n+1$, то $v \sim_L a_{k-n}a_{k-n+1} \dots a_k$, и следователно всяка дума с дължина, по-малка от $n+1$ попада в клас на еквивалентност породен от дума с дължина $n+1$.

Така

$$\{0,1\}^*/\sim_L = \{[w]_L \mid |w| = n+1\},$$

като при това, ако $|u| = |v| = n+1$ и $u \neq v$, то $[u]_L \neq [v]_l$. Следователно минималниият тотален и детерминиран автомат, разпознаващ L, има 2^{n+1} състояния. Освен това имаме

$$[a_0a_1 \dots a_n] \xmapsto[\mathcal{A}_L]{a} [a_1 \dots a_na]$$

и

$$F_L = \{[0a_1 \dots a_n] \mid a_1, \dots, a_n \in \{0,1\}^*\}.$$

Алгоритъм за минимизация на тотални и детерминирани автомати Нека $\mathcal{A} = (Q, q_0, \Delta, F)$ е тотален, детерминиран краен автомат над Σ, в който всяко състояние е достижимо. Нека $L = L(\mathcal{A})$. Да разгледаме изображението $\varphi : Q \to Q_L$, действащо по правилото

$$\varphi(q) = [w]_L \iff q_0 \xmapsto[\mathcal{A}]{w} q.$$

Тогава съгласно доказателството на Теорема 6.21 φ е сюрективно, като

$$\begin{array}{rcl} \varphi(q_0) & = & q_{0L}, \\ q \xmapsto[\mathcal{A}]{a} q' & \iff & \varphi(q) \xmapsto[\mathcal{A}_L]{a} \varphi(q') \\ q \in F & \iff & \varphi(q) \in F_L \end{array}$$

Дефинираме релацията $\equiv$ в Q чрез

$$q \equiv q' \iff \varphi(q) = \varphi(q').$$

Ясно е, че $\equiv$ е релация на еквивалетност. Нека $\mathcal{A}/\equiv = (Q', q_0', \Delta', F')$, където

$$\begin{array}{rcl} Q' & = & Q/\equiv \\ q_0' & = & [q_0]_\equiv \\ \Delta' & = & \{([q]_\equiv, a, [s]_\equiv) \mid q \xmapsto[\mathcal{A}]{a} s\} \\ F' & = & \{[q]_\equiv \mid q \in F\} \end{array}$$

Да разгледаме изображението $\psi : Q' \to Q_L$, действащо по правилото

$$\psi([q]_\equiv) = \varphi(q).$$

Тогава ψ е коректно дефинирано и инективно изображение, тъй като

$$[q] = [q'] \iff \varphi(q) = \varphi(q') \iff \psi([q]_\equiv) = \psi([q']_\equiv).$$

Ясно е, че ψ е сюрективно и следователно ψ е биекция. При това

$$\begin{array}{rcl} [q]_\equiv \xmapsto[\mathcal{A}_\equiv]{a} [s]_\equiv & \iff & q' \xmapsto[\mathcal{A}]{a} s' \text{ за някои } q' \equiv q,\ s' \equiv s \\ & \iff & \varphi(q') \xmapsto[\mathcal{A}_L]{a} \varphi(s') \text{ за някои } q', s', \text{ такива че } \varphi(q') = \varphi(q),\ \varphi(s') = \varphi(s) \\ & \iff & \psi([q]_\equiv) \xmapsto[\mathcal{A}_L]{a} \psi([s]_\equiv). \end{array}$$

и

$$\begin{aligned}[q]_{\equiv} \in F' &\iff q \equiv q' \text{ за някое } q' \in F \\ &\iff \varphi(q) = \varphi(q') \in F_L \text{ за някое } q' \\ &\iff \psi([q]_{\equiv}) \in F_L.\end{aligned}$$

Следователно ψ е изоморфизъм между $\mathcal{A}/\equiv$ и $\mathcal{A}_L$ и значи $\mathcal{A}/\equiv$ е минимален тотален детерминиран автомат, разпознаващ L.

За да построим $\mathcal{A}/\equiv$ достатъчно е да определим релацията $\equiv$. Да забележим, че $q \equiv q'$ тогава и само тогава, когато

$$\forall w \in \Sigma^*(q \xmapsto[\mathcal{A}]{w} f \text{ за някое } f \in F \iff q' \xmapsto[\mathcal{A}]{w} f' \text{ за някое } f' \in F)$$

Да означим с $\equiv_n$, $n \in \mathbb{N}$ релацията в Q, дефинирана чрез $q \equiv_n q'$ тогава и само тогава, когато

$$\forall w \in \Sigma^{\leq n}(q \xmapsto[\mathcal{A}]{w} f \text{ за някое } f \in F \iff q' \xmapsto[\mathcal{A}]{w} f' \text{ за някое } f' \in F)$$

Тогава

$$\equiv_0 \supseteq \equiv_1 \supseteq \cdots \supseteq \equiv_n \supseteq \cdots \supseteq \equiv .$$

Тъй като релациите са крайни, то в горната редица има само краен брой строги включвания, т.е. съществува n_0, такова че

$$\equiv_{n_0} = \equiv_{n_0+1} = \cdots = \equiv .$$

От друга страна

$$q \equiv_{i+1} q' \iff q \equiv_i q' \ \& \ \forall a \in \Sigma \forall s \forall s'(q \xmapsto[\mathcal{A}]{a} s,\ q' \xmapsto[\mathcal{A}]{a} s' \Longrightarrow s \equiv_i s'),$$

което дава рекурсивна дефиниция на $\equiv_n$ зависеща само от преходите на автомата $\mathcal{A}$.

ГЛАВА 7

Безконтекстни граматики

7.1 Безконтекстни граматики и езици

Безконтекстна (или контекстносвободна) *граматика* ще наричаме всяка четворка $G = (\Gamma, \Sigma, R, S)$, където $\Sigma \subseteq \Gamma$ са крайни азбуки, $S \in \Gamma \setminus \Sigma$, а $R \subseteq (\Gamma \setminus \Sigma) \times \Gamma^*$ е крайно множество. Γ наричаме азбука на граматиката, Σ — терминални символи на граматиката, $\Gamma \setminus \Sigma$ — нетерминални символи на граматиката, S — начален символ на граматиката, а R — правила на граматиката. Ако $(A, \alpha) \in R$, ще пишем $A \to_G \alpha$ или $A \to \alpha \in R$ и ще казваме, че можем да заместим символа A с думата α.

За всеки избор на думи $w_1, w_2 \in \Gamma^*$ и правило $A \to_G \alpha$, ще пишем

$$w_1 A w_2 \Rightarrow_G w_1 \alpha w_2.$$

Ще казваме, че G преобразува (за $n \geq 0$ стъпки) думата $u \in \Gamma^*$ до дума $u' \in \Gamma^*$ и ще пишем

$$u \Rightarrow_G^* u'$$

ако

$$u \Rightarrow_G u_1 \Rightarrow_G u_2 \Rightarrow_G \cdots \Rightarrow_G u'$$

за някои думи $u_1, \ldots, u_{n-1} \in \Gamma^*$ (при $n = 0$ получаваме $u \Rightarrow_G^* u$).

Под *език* на G ще разбираме множеството

$$L(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}.$$

Пример. Да разгледаме граматиката G с азбука $\{S, a, b\}$, терминални символи $\{a, b\}$, начален символ S и правила

$$S \to \varepsilon \mid aSb.$$

Тогава G за $n \geq 1$ стъпки преобразува S до

$$a^n S b^n \text{ или } a^{n-1} b^{n-1}.$$

Действително, твърдението е очевидно за $n = 1$, а ако твърдението е вярно за $n \geq 1$, то от $a^n S b^n$ за една стъпка можеж да получим

$$a^{n+1} S b^{n+1} \text{ или } a^n b^n,$$

а от $a^{n-1} b^{n-1}$ нищо не можем да получим, тъй като не съдържа нетерминални символи.

Следователно

$$L(G) = \{a^n b^n \mid n \geq 0\}.$$

Пример. Безконтекстните граматики се използват основно за генерирането на структуриран текст като математически изрази и компютърни програми. Да разгледаме, например, граматиката G, която има азбука $\{T, \mathrm{id}, +, ., (,)\}$, терминални символи $\{\mathrm{id}, +, ., (,)\}$, начален символ T и правила

$$T \to \mathrm{id} \mid (T) \mid T + T \mid T.T$$

С нейна помощ можем да генерираме общи аритметични изрази, като например

$$\begin{aligned} T &\Rightarrow_G T.T \\ &\Rightarrow_G T.(T) \\ &\Rightarrow_G \mathrm{id}.(T) \\ &\Rightarrow_G \mathrm{id}.(T + T) \\ &\Rightarrow_G \mathrm{id}.(\mathrm{id} + T) \\ &\Rightarrow_G \mathrm{id}.(\mathrm{id} + \mathrm{id}) \end{aligned}$$

и

$$\begin{aligned} T &\Rightarrow_G T+T \\ &\Rightarrow_G T+T.T \\ &\Rightarrow_G \mathrm{id}+T.T \\ &\Rightarrow_G \mathrm{id}+T.\mathrm{id} \\ &\Rightarrow_G \mathrm{id}+\mathrm{id}.\mathrm{id} \end{aligned}$$

По общо, граматика отговаря на следната рекурсивна дефиниция на аритметични изрази:

(i) id е аритметичен израз;

(ii) ако α е аритметичен израз, то (α) е аритметичен израз;

(iii) ако α и β са аритметични изрази, то $\alpha+\beta$ и $\alpha.\beta$ също са аритметични изрази.

В тази граматика терминалният символ id има смисъл на „алгебричен индентификатор“ (като например число в десетичен запис, променлива, функция или друго). Ако, например, искаме да генерираме аритметичните изрази над естествените числа в десетичен запис, то достатъчно е да „направим“ символа id нетерминален, да добавим термиални символи $\{0,1,\ldots,9\}$, нетерминален сивмол M както и правилата

$$\begin{aligned} \mathrm{id} &\to 0 \mid 1M \mid 2M \mid \cdots \mid 9M \\ M &\to \varepsilon \mid 0M \mid 1M \mid \cdots \mid 9M \end{aligned}$$

Тогава числото 25072 се извежда по следния начин

$$\begin{aligned} \mathrm{id} &\Rightarrow_G 2M \\ &\Rightarrow_G 25M \\ &\Rightarrow_G 250M \\ &\Rightarrow_G 2507M \\ &\Rightarrow_G 25072M \\ &\Rightarrow_G 25072 \end{aligned}$$

Ако искаме да разполагаме с изрази за целите числа в десетичен запис, трябва да добавим нов нетерминален символ N, терминалния символ „$-$“ и да заменим горните правила за id с

$$\begin{aligned} \mathrm{id} &\to 0 \mid N \mid -N \\ N &\to 0 \mid 1M \mid 2M \mid \cdots \mid 9M \\ M &\to \varepsilon \mid 0M \mid 1M \mid \cdots \mid 9M \end{aligned}$$

Чрез тези правила можем да изведем -3780 по следния начин

$$\begin{aligned} \mathrm{id} &\Rightarrow_G -N \\ &\Rightarrow_G -3M \\ &\Rightarrow_G -37M \\ &\Rightarrow_G -378M \\ &\Rightarrow_G -3780M \\ &\Rightarrow_G -3780 \end{aligned}$$

По-горе видяхме, че езикът $\{a^n b^n \mid n \geq 0\}$ е безконтекстен. Както знаем, този език не е регулярен и следователно класовете на регулярните и на безконтекстните езици за различни. Сега ще видим, че класът на регулярните езици се съдържа в този на безконтекстните езици. За целта, ще казваме, че една граматика $G=(\Gamma,\Sigma,S,R)$ е *регулярна* (или *дясно линейна*), ако правилата на G са от вида

$$A \to wB \text{ или } A \to w$$

където $w \in \Sigma^*$, а B е нетерминален символ. В сила е следната теорема

Твърдение 7.1. Езиците на регулярните граматики са точно регулярните езици.

Доказателство. Нека първо $L \subseteq \Sigma^*$ е регулярен език и нека $\mathcal{A} = (Q, q_0, \Delta, F)$ е краен автомат с ε-преходи над Σ, за който $Q \cap \Sigma = \emptyset$ и $L(\mathcal{A}) = L$. Да разгелдаме граматиката G_L с азбука $Q \cup \Sigma$, терминални символи Σ, начален символ q_0 и правила

$$\begin{aligned} q &\to xq' \quad \text{за} \quad (q, x, q') \in \Delta, \\ q &\to \varepsilon \quad \text{за} \quad q \in F. \end{aligned}$$

Ясно е, че G_L е регулярна граматика. От вида на правилата директно следва, че ако

$$q \Rightarrow_G^* u,$$

където $q \in Q$, то

$$u \in \Sigma^* \text{ или } u = wq' \text{ за някое } w \in \Sigma^*.$$

Освен това, за $x \in \Sigma \cup \{\varepsilon\}$ в сила е

$$q \Rightarrow_G^* xq' \iff q \xmapsto[\mathcal{A}]{x} q'.$$

Оттук, за всяка непразна дума $w \in \Sigma^*$

$$q \Rightarrow_G^* wq' \iff q \xmapsto[\mathcal{A}]{w} q'.$$

Действително, за думи с дължина 0 или 1 твърдението е вярно. От друга страна, ако твърдението е вярно за думи с дължина n и $|w| = n + 1$, то $w = w'a$ за някое $w \in \Sigma^*$ с $|w'| = n$ и $a \in \Sigma$ и тогава

$$\begin{aligned} q \Rightarrow_G^* wq' \quad &\iff \quad q \Rightarrow_G^* w'q'' \text{ и } q'' \Rightarrow_G^* aq' \text{ за някое } q'' \in Q \\ &\iff \quad q \xmapsto[\mathcal{A}]{w'} q'' \text{ и } q'' \xmapsto[\mathcal{A}]{a} q' \text{ за някое } q'' \in Q \\ &\iff \quad q \xmapsto[\mathcal{A}]{w} q'. \end{aligned}$$

Оттук за произволна $w \in \Sigma^*$

$$\begin{aligned} w \in L(G_L) \quad &\iff \quad q_0 \Rightarrow_G^* w \\ &\iff \quad q_0 \Rightarrow_G^* wq' \text{ и } q' \Rightarrow_G \varepsilon \text{ за някое } q' \in Q \\ &\iff \quad q \xmapsto[\mathcal{A}]{w} q' \text{ за някое } q' \in F \\ &\iff \quad w \in L(\mathcal{A}) \end{aligned}$$

и следователно

$$L(G_L) = L.$$

Така всеки регулярен език е език на регулярна граматика. Нека сега G е регулярна граматика. Модифицираме G (ако е необходимо) по следния начин:

- Заместваме всяко правило r от вида
$$A \to a_1 a_2 \dots a_n B,$$
където $a_1, a_2, \dots, a_n$ са терминални, $n \geq 0$, а B е нетерминален символ, с правилата
$$A \to a_1 A_1^r,\ A_1^r \to a_2 A_2^r, \dots, A_{n-1}^r \to a_n B,$$
където $A_1^r, \dots, A_{n-1}^r$ са нови нетерминални символи.

- Заместваме всяко правило r от вида
$$A \to a_1 a_2 \dots a_n,$$
където $a_1, a_2, \dots, a_n$ са терминални с правилата
$$A \to a_1 A_1^r,\ A_1^r \to a_2 A_2^r, \dots, A_{n-1}^r \to a_n A_n^r, A_n^r \to \varepsilon,$$
където $A_1^r, \dots, A_n^r$ са нови нетерминални символи.

Нека граматиката $G' = (\Gamma, \Sigma, S, R)$ е резултатът от горните преобразувания. Тогава всяко правило r на G от вида $A \to a_1 a_2 \dots a_n B$, където $a_1, a_2, \dots, a_n$ са терминални, $n \geq 0$, а B е нетерминален символ може да бъде симулирано от извода

$$A \Rightarrow_{G'} a_1 A_1^r \Rightarrow_{G'} a_1 a_2 A_2^r \Rightarrow_{G'} \dots \Rightarrow_{G'} a_1 \dots a_{n-1} A_{n-1}^r \Rightarrow_{G'} a_1 a_2 \dots a_n B.$$

а правилата r от вида $A \to a_1 a_2 \dots a_n$, където $a_1, a_2, \dots, a_n$ са терминални, $n \geq 0$, могат да бъдат симулирани чрез извода

$$A \Rightarrow_{G'} a_1 A_1^r \Rightarrow_{G'} a_1 a_2 A_2^r \Rightarrow_{G'} \dots \Rightarrow_{G'} a_1 a_2 \dots a_n A_n^r \Rightarrow_{G'} a_1 a_2 \dots a_n.$$

Освен това, всеки нов нетерминален символ, може да участва единствено и само в изводи от горния вид. Следователно $L(G) = L(G')$. Нека сега $\mathcal{A}_G = (Q, q_0, \Delta, F)$ е краен автомат с ε-преходи, за който

$$\begin{aligned} Q &= \Gamma \setminus \Sigma \\ q_0 &= S \\ \Delta &= \{(A, x, B) \mid A \to_{G'} xB\} \\ F &= \{A \mid A \to_G \varepsilon\}. \end{aligned}$$

Нека $G_{L(\mathcal{A})}$ е граматиката, съответвстваща на $\mathcal{A}_G$, получена по начина, описан по-горе. Тогава $L(G_{L(\mathcal{A})}) = L(\mathcal{A}_G)$. Но $G_{L(\mathcal{A})}$ е точно граматиката G' и значи

$$L(G) = L(\mathcal{A}_G).$$

□

7.2 Най-ляв и най-десен извод. Дърво на извода

Граматиките, които разглеждаме, се наричат безконтекстни, тъй като правилата са от вида $A \to_G \alpha$ и значи G позволява заместването на символа A с думата α независимо от това, какво пише отляво и отдясно на A (т.е. какъв е левия и десния контекст, в който се среща A). Интуитивно, това означава, че правилото може да бъде приложено по всяко време на извода, като избора на точния момент, в който да направим това не влияе върху цялостния резултат на извода. По-формално, нека G е безконтекстна граматика. Тогава изводите

$$uAvBw \underset{B\to\beta}{\Rightarrow_G} uAv\beta w \underset{A\to\alpha}{\Rightarrow_G} u\alpha v\beta w$$

и

$$uAvBw \underset{A\to\alpha}{\Rightarrow_G} u\alpha vBw \underset{B\to\beta}{\Rightarrow_G} u\alpha v\beta w$$

са взаимнозаменяеми.

Ще казваме, че изводите

$$v \Rightarrow_G u_1 \Rightarrow_G \cdots \Rightarrow_G u_n \Rightarrow_G v'$$

и

$$v \Rightarrow_G u'_1 \Rightarrow_G \cdots \Rightarrow_G u'_n \Rightarrow_G v'$$

са *еквиваленти*, ако вторият може да се получи от първия (съответно, първият от втория) чрез последователни размени от по-горния вид.

В случай, че началната дума е нетерминален символ, а последната дума е съставена само от терминални символи, то даден извод

$$A \Rightarrow_G u_1 \Rightarrow_G \cdots \Rightarrow_G u_n \Rightarrow_G w$$

е еквивалентен на извод

$$A \Rightarrow_G u'_1 \Rightarrow_G \cdots \Rightarrow_G u'_n \Rightarrow_G w,$$

в който всяка дума се получава от предходната, заместваики, съгласно някое от правилата, най-левият нетерминален символ. Такъв извод наричаме най-ляв извод на w от A. Аналогично имаме и извод

$$A \Rightarrow_G u''_1 \Rightarrow_G \cdots \Rightarrow_G u''_n \Rightarrow_G w,$$

в който всяка дума се получава от предходната, заместваики, съгласно правилата някое от правилата, най-десният нетерминален символ. Подобен извод наричаме най-десен извод на w от A.

Пример. Да разгледаме граматиката G, която има азбука $\{T, \mathrm{id}, +, ., (,)\}$, терминални символи $\{\mathrm{id}, +, ., (,)\}$, начален символ T и правила

$$T \to \mathrm{id} \mid (T) \mid T + T \mid T.T$$

Тогава можем да преобразуваме извода

$$\begin{aligned}
T &\Rightarrow_G T + T && // \ T \to T + T \\
&\Rightarrow_G T + T.T && // \ T \to T.T \\
&\Rightarrow_G T + \mathrm{id}.T && // \ T \to \mathrm{id} \\
&\Rightarrow_G \mathrm{id} + \mathrm{id}.T && // \ T \to \mathrm{id} \\
&\Rightarrow_G \mathrm{id} + \mathrm{id}.\mathrm{id} && // \ T \to \mathrm{id}
\end{aligned}$$

до най-ляв извод

$$\begin{aligned}
T &\Rightarrow_G T + T && // \ T \to T + T \\
&\Rightarrow_G \mathrm{id} + T && // \ T \to \mathrm{id} \\
&\Rightarrow_G \mathrm{id} + T.T && // \ T \to T.T \\
&\Rightarrow_G \mathrm{id} + \mathrm{id}.T && // \ T \to \mathrm{id} \\
&\Rightarrow_G \mathrm{id} + \mathrm{id}.\mathrm{id} && // \ T \to \mathrm{id}
\end{aligned}$$

и до най-десен извод

$$\begin{aligned}
T &\Rightarrow_G T + T && // \ T \to T + T \\
&\Rightarrow_G T + T.T && // \ T \to T.T \\
&\Rightarrow_G T + T.\mathrm{id} && // \ T \to \mathrm{id} \\
&\Rightarrow_G T + \mathrm{id}.\mathrm{id} && // \ T \to \mathrm{id} \\
&\Rightarrow_G \mathrm{id} + \mathrm{id}.\mathrm{id} && // \ T \to \mathrm{id}
\end{aligned}$$

Трите извода в горния пример са еквивалентни. За да уловим по-добре тази еквивалентност въвеждаме понятието дърво на извод по следния (рекурсивен) начин:

(i) За всеки терминален символ a

$$a$$

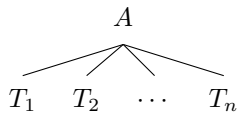
е дърво на извода с корен a, височина 0 и извод a.

(ii) За всяко правило $A \to_G \varepsilon$

$$\begin{array}{c} A \\ | \\ \varepsilon \end{array}$$

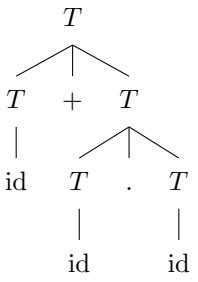
е дърво с корен A, височина 1 и извод ε.

(iii) За всяко правило $A \to_G X_1 X_2 \dots X_n$, където $X_1, \dots, X_n$ са символи на граматиката, и всеки избор на дървета $T_1, \dots, T_n$ съответно с корени $X_1, \dots, X_n$, височини $h_1, \dots, h_n$ и изводи $w_1, \dots, w_n$

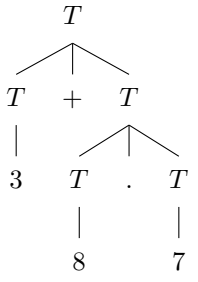


е дърво с корен A, височина $1 + \max\{h_1, \dots, h_n\}$ и извод $w_1 w_2 \dots w_n$.

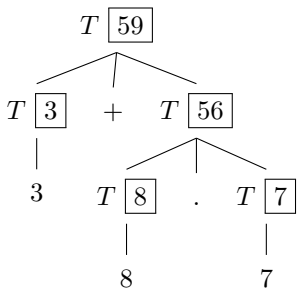
Пример. В предния пример на трите извода на $\mathrm{id} + \mathrm{id.id}$ съответства дървото



като най-левият (най-десният) извод съответства на обхождане в дълбочина отляво (отдясно) на дървото. Дървото на извода ясно показва структурата на израза $\mathrm{id} + \mathrm{id.id}$, като той може да бъде използван за да намираме неговата стойност, когато даваме различни стойности на id. Така, например, на израза $3 + 8.7$ съответства дървото



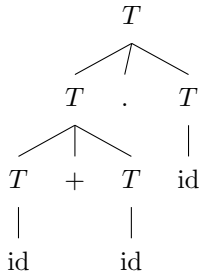
което, остойностявайки отдолу нагоре, получаваме



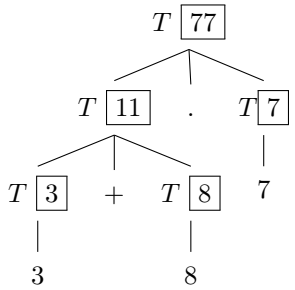
Конкретната граматика

$$T \to \text{id} \mid (T) \mid T + T \mid T.T$$

не е подходяща за генериране на аритметични изрази, тъй като някой изрази допускат повече от едно дърво на извод. Например, изразът id + id.id има и дърво на извод



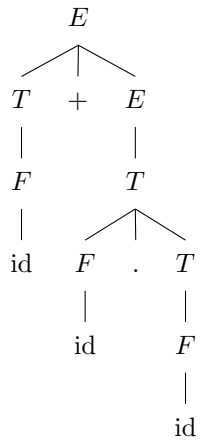
Сега, ако искаме да намерим стойността на $3 + 8.7$, използвайки това дърво, получаваме



Така получаваме две различни интерпретации на израза $3 + 8.7$ (една правилна и една грешна), което прави дадената граматика неприложима за генериране и остойностяване на аритметични изрази. Основният проблем е, че граматиката не отчита приоритета на операцията умножение пред операцията събиране. Бихме могли да поправим този недостатък, въвеждайки още два терминални символ E и F, като начален е символа E, и използвайки правилата

$$\begin{aligned} E &\to T + E \mid T \\ T &\to F.T \mid F \\ F &\to \text{id} \mid (E) \end{aligned}$$

Тази граматика (както и предната) генерира точно аритметичните изрази изградени от id, +, . и скоби, като този път за всеки аритметичен израз съществува единствено дърво на извода. За израза id + id.id дървото на извода е



Граматики, за които всяка дума от езика има единствено дърво на извод, наричаме *еднозначни*, а другите (т.е. за които поне една дума от езика им има поне две различни дървета на извод) наричаме *многозначни*.

7.3 Обединение, конкатенация, звезда на Клини

Твърдение 7.2. Безконтекстните езици за затворение относно операцията обединение.

Доказателство. Нека L_1 и L_2 са безконтекстни езици над азбуката Σ и нека

$$G_1 = (\Gamma_1, \Sigma, R_1, S_1) \text{ и } G_2 = (\Gamma_2, \Sigma, R_2, S_2)$$

са безконтекстни граматики с

$$L(G_1) = L_1 \text{ и } L(G_2) = L_2,$$

и такива, че

$$\Gamma_1 \cap \Gamma_2 = \Sigma.$$

Да разгледаме безконтекстната граматика

$$G = (\Gamma, \Sigma, R, S),$$

където

$$\begin{aligned} \Gamma &= \Gamma_1 \cup \Gamma_2 \cup \{S\} \\ S &\notin \Gamma_1 \cup \Gamma_2 \\ R &= R_1 \cup R_2 \cup \{S \to S_1 | S_2\} \end{aligned}$$

Имаме $L_1 \cup L_2 \subseteq L(G)$. Действително, ако $w \in L_1$, то

$$S_1 \Rightarrow_{G_1}^* w,$$

и тъй като $R_1 \subseteq R$, то

$$S_1 \Rightarrow_G^* w.$$

От друга страна

$$S \Rightarrow_G S_1,$$

откъдето

$$S \Rightarrow_G^* w$$

и значи $w \in L(R)$.

За обратното включване, да забележим, че от дефиницията на R и това, че G_1 и G_2 нямат общи нетерминални символи, следва че ако

$$u \Rightarrow_G v$$

за някое $u \in \Gamma_1^*$, то $v \in \Gamma_1^*$ и

$$u \Rightarrow_{G_1} v.$$

Оттук, ако

$$u \Rightarrow_G^* v$$

за някое $u \in \Gamma_1^*$, то $v \in \Gamma_1^*$ и

$$u \Rightarrow_{G_1}^* v.$$

Аналогично, ако

$$u \Rightarrow_G^* v$$

за някое $u \in \Gamma_2^*$, то $v \in \Gamma_2^*$ и

$$u \Rightarrow_{G_2}^* v.$$

Нека сега $w \in L(G)$. Тогава $S \Rightarrow_G^* w$ и значи

$$S \Rightarrow_G S_1 \text{ и } S_1 \Rightarrow_G^* w$$

или

$$S \Rightarrow_G S_2 \text{ и } S_2 \Rightarrow_G^* w$$

Но $S_1 \in \Gamma_1^*$, а $S_2 \in \Gamma_2^*$ и значи

$$S_1 \Rightarrow_{G_1}^* w \text{ или } S_2 \Rightarrow_{G_2}^* w,$$

т.е.

$$w \in L_1 \text{ или } w \in L_2.$$

□

Твърдение 7.3. Безконтекстните езици за затворение относно операцията конкатенация.

Доказателство. Нека L_1 и L_2 са безконтекстни езици над азбуката Σ и нека

$$G_1 = (\Gamma_1, \Sigma, R_1, S_1) \text{ и } G_2 = (\Gamma_2, \Sigma, R_2, S_2)$$

са безконтекстни граматики с

$$L(G_1) = L_1 \text{ и } L(G_2) = L_2,$$

и такива, че

$$\Gamma_1 \cap \Gamma_2 = \Sigma.$$

Да разгледаме безконтекстната граматика

$$G = (\Gamma, \Sigma, R, S),$$

където

$$\begin{aligned} \Gamma &= \Gamma_1 \cup \Gamma_2 \cup \{S\} \\ S &\notin \Gamma_1 \cup \Gamma_2 \\ R &= R_1 \cup R_2 \cup \{S \to S_1 S_2\} \end{aligned}$$

Имаме $L_1 L_2 \subseteq L(G)$. Действително, нека $w_1 \in L_1$ и $w_2 \in L_2$. Тогава

$$S_1 \Rightarrow_{G_1}^* w_1 \text{ и } S_2 \Rightarrow_{G_2}^* w_2.$$

Оттук и $R_1 \cup R_2 \subseteq R$ следва

$$S_1 \Rightarrow_G^* w_1 \text{ и } S_2 \Rightarrow_G^* w_2,$$

откъдето

$$S \Rightarrow_G S_1 S_2 \Rightarrow_G^* w_1 S_2 \Rightarrow_G^* w_1 w_2$$

и значи $w_1 w_2 \in L$.

За обратното включване, отново както в доказателството на предното твърдение, имаме, че ако

$$u \Rightarrow_G^* v$$

за някое $u \in \Gamma_i^*$, то $v \in \Gamma_i^*$ и

$$u \Rightarrow_{G_i}^* v.$$

Нека сега $w \in L(G)$ и да разгледаме един най-ляв извод на w от S. Той има вида

$$S \Rightarrow_G S_1 S_2 \Rightarrow_G^* w_1 S_2 \Rightarrow_G^* w_1 w_2,$$

където $w_1 w_2 = w$ и

$$S_1 \Rightarrow_G^* w_1 \text{ и } S_2 \Rightarrow_G^* w_2.$$

Но $S_1 \in \Gamma_1^*$, а $S_2 \in \Gamma_2^*$ и значи

$$S_1 \Rightarrow_{G_1}^* w_1 \text{ и } S_2 \Rightarrow_{G_2}^* w_2,$$

откъдето

$$w_1 \in L_1 \text{ и } w_2 \in L_2$$

и следователно $w \in L_1 L_2$.

□

Твърдение 7.4. Безконтекстните езици за затворение относно операцията звезда на Клини.

Доказателство. Нека L е безконтекстен език над азбуката Σ и нека

$$G = (\Gamma, \Sigma, R, S)$$

е безконтекстна граматика с

$$L(G) = L.$$

Да разгледаме безконтекстната граматика

$$G' = (\Gamma', \Sigma, R', S'),$$

където

$$\begin{aligned} \Gamma' &= \Gamma \cup \{S'\} \\ S' &\notin \Gamma \\ R' &= R \cup \{S' \to \varepsilon \mid S'S\} \end{aligned}$$

Имаме $L^* \subseteq L(G)$. Действително,

$$S' \Rightarrow_{G'} \varepsilon$$

и значи $\varepsilon \in L(G)$. От друга страна, ако $w_1, \dots w_n \in L$, то

$$S \Rightarrow_G^* w_i \text{ за } 1 \leq i \leq n,$$

откъдето, благодарение на $R \subseteq R'$, имаме

$$S \Rightarrow_{G'}^* w_i \text{ за } 1 \leq i \leq n,$$

откъдето

$$\begin{aligned} S' &\Rightarrow_{G'}^* S' \underbrace{S\ S\ \dots\ S}_{n} \\ &\Rightarrow_{G'} \underbrace{S\ S\ \dots\ S}_{n} \\ &\Rightarrow_{G'}^* w_1 \underbrace{S\ \dots\ S}_{n-1} \\ &\ \ \vdots \\ &\Rightarrow_{G'}^* w_1 w_2 \dots w_n \end{aligned}$$

и значи $w_1 w_2 \dots w_n \in L(G')$.

За обратното включване, отново както в доказателството на предните две твърдения, имаме, че ако

$$u \Rightarrow_{G'}^* v$$

за някое $u \in \Gamma^*$, то $v \in \Gamma^*$ и

$$u \Rightarrow_G^* v.$$

Нека сега $w \in L(G)$ и да разгледаме един най-ляв извод на w от S. Той има вида

$$S' \Rightarrow_{G'}^* S' \underbrace{SS \dots S}_{n} \Rightarrow_{G'} \underbrace{SS \dots S}_{n} \Rightarrow_{G'}^* w_1 \underbrace{S \dots S}_{n-1} \cdots \Rightarrow_{G'}^* w_1 w_2 \dots w_n$$

където $n \geq 0$, $w_1 w_2 \dots w_n = w$ и

$$S \Rightarrow_{G'}^* w_i \text{ за } 1 \leq i \leq n.$$

Но $S \in \Gamma^*$ и значи

$$S \Rightarrow_G^* w_i \text{ за } 1 \leq i \leq n.$$

откъдето

$$w_i \in L \text{ и за } 1 \leq i \leq n$$

и следователно $w \in L^*$.

□

7.4 Нормална форма на Чомски

Ще казваме, че безконтекстната граматика G е *в нормална форма на Чомски*, ако за всяко правило $A \to_G \alpha$ е в сила $|\alpha| = 2$.

Теорема 7.5. За всяка безконтекстна граматика G съществува безконтекстна граматика G' в нормална форма на Чомски, такава че за всяка дума w с

$$|w| \geq 2$$

е изпълнено

$$w \in L(G) \iff w \in L(G').$$

Доказателство. Ще покажем, как да трансформираме произволна безконтекстна граматика до граматика в нормална форма на Чомски. Нека $G = (\Gamma, \Sigma, R, S)$ е безконтекстна граматика. G може да нарушава условието за нормална форма по две причини: в G има правила $A \to_G \alpha$ с $|\alpha| > 2$ (дълги правила) и в G има правила $A \to_G \alpha$ с $|\alpha| < 2$ (къси правила). Първо елиминираме дългите правила по следната схема:

- Заместваме правилата от вида $A \to X_1 X_2 \dots X_n$, където $n \geq 3$ и $X_1, X_2, \dots, X_n \in \Gamma$, с

$$\begin{aligned} A &\to X_1 A_1 \\ A_1 &\to X_2 A_2 \\ &\vdots \\ A_{n-2} &\to X_{n-1} X_n \end{aligned}$$

  където $A_1, \dots, A_n$ са нови нетерминални символи.

Означаваме новополучената граматика с G_1. Ясно е, че G_1 може да симулира всяко правило $A \to_G X_1 X_2 \dots X_n$ чрез извода

$$A \Rightarrow_{G_1} X_1 A_1 \Rightarrow_G \dots \Rightarrow_{G_1} X_1 \dots X_{n-2} A_{n-2} \Rightarrow_{G_1} X_1 \dots X_n.$$

При това, всяко едно от „новите“ правила може да участва единствено и само в извод еквивалентен на този от горния вид и следователно

$$L(G) = L(G_1).$$

За всяко правило $A \to_{G_1} \alpha$ е в сила $|\alpha| \leq 2$. На следващата стъпка се освобождаваме от ε-правилата (т.е. от правилата от вида $A \to \varepsilon$) по следната схема:

- Образуваме множеството

$$\mathcal{E} = \{B \mid B \Rightarrow^*_{G_1} \varepsilon\}.$$

- Премахваме всички правила от вида $A \to \varepsilon$.
- За всяко $B \in \mathcal{E}$ и всяко правило $A \to BC$ или $A \to CB$ добавяме правило

$$A \to C.$$

Означамаме новополучената граматика с G_2. За всяко правило $A \to_{G_2} C$ е вярно, че или

$$A \to_{G_1} C,$$

или

$$A \Rightarrow_{G_1} BC \Rightarrow^*_{G_1} C,$$

или

$$A \Rightarrow_{G_1} CB \Rightarrow^*_{G_1} C.$$

Следователно (тъй като всички други правила на G_2 са правила на G_1) всеки извод в G_2 може да бъде симулиран в G_1 и значи

$$L(G_2) \subseteq L(G_1).$$

Обратно, нека имаме извод в G_1, започващ от началния символ S, който извежда непразна дума, и в който е използван точно едно ε-правило. Нека това правило е приложено към символа B. Този символ се е появил или

чрез правило $A \to_{G_1} BC$, или чрез правило $A \to_{G_1} CB$, или чрез правило $B' \to_{G_1} B$. Ако е в сила последното, то символът B' се е появил или чрез правило $A \to_{G_1} B'C$, или чрез правило $A \to_{G_1} CB$, или чрез правило $B'' \to_{G_1} B'$. Прилагайки тези разсъждения краен брой пъти и преместваики прилагането на правилата, така че те да се извършват непосредствено едно след друго, получаваме, че първоначалният извод е еквивалентен на

$$\begin{aligned} S &\Rightarrow^*_{G_1} wAw' \\ &\Rightarrow_{G_1} wB^{(n)}Cw' \\ &\Rightarrow_{G_1} wB^{(n-1)}Cw' \\ &\quad\vdots \\ &\Rightarrow_{G_1} wBCw' \\ &\Rightarrow_{G_1} wCw' \\ &\quad\vdots \end{aligned} \qquad \text{или} \qquad \begin{aligned} S &\Rightarrow^*_{G_1} wAw' \\ &\Rightarrow_{G_1} wCB^{(n)}w' \\ &\Rightarrow_{G_1} wCB^{(n-1)}w' \\ &\quad\vdots \\ &\Rightarrow_{G_1} wCBw' \\ &\Rightarrow_{G_1} wCw' \\ &\quad\vdots \end{aligned}$$

който в G_2 може да бъде симулиран чрез

$$S \Rightarrow^*_{G_2} wAw' \Rightarrow_{G_2} wCw' \Rightarrow_{G_2} \dots$$

тъй като G_2 съдържа всички правила на G_1 с изключение на ε-правилата.

По този начин можем да „елиминараме“ употребата на всички ε-правила от произволен извод в G_1, започващ от S, който извежда непразна дума. Следователно за непразни думи w

$$w \in L(G_2) \iff w \in L(G_1).$$

От друга страна, очевидно

$$\varepsilon \notin L(G_2)$$

и следователно

$$L(G_2) = L(G) \setminus \{\varepsilon\}.$$

За всяко правило $A \to_{G_2} \alpha$ е в сила $1 \leq |\alpha| \leq 2$. Остава да се освободим от *преименуващите* правила, т.е. правилата, за които $|\alpha| = 1$. Правим това по следната схема

- За всеки символ X от азбуката на G_2 образуваме

  $$D(X) = \{Z \in \Gamma_2 \mid Z \text{ е символ на } G_2 \text{ и } X \Rightarrow^*_{G_2} Z\},$$

  т.е. $D(X)$ се състои от всички символи, в които G_2 може да преименува X. Да забележим, че

  $$X \in D(X)$$

  и за всеки терминален сивмол a е изпълнено

  $$D(a) = \{a\}.$$

- Премахваме всички преименуващи правила.

- Заместваме всяко правило $A \to XY$ с правила

  $$A \to X'Y' \text{ за } X' \in D(X) \text{ и } Y' \in D(Y).$$

- За началния символ S на граматиката добавяме

  $$S \to X'Y'$$

  за всяко правило $A \to X'Y'$, където $A \in D(S)$.

Означаваме новополучената граматика с G', която вече е в нормална форма на Чомски. Правилата

$$A \to_{G'} X'Y'$$

могат да бъдат симулирани от G_2 чрез изводи от вида

$$\begin{aligned} A &\Rightarrow_{G_2} XY \\ &\Rightarrow_{G_2} X_1Y \\ &\Rightarrow_{G_2} X_2Y \\ &\vdots \\ &\Rightarrow_{G_2} X'Y \\ &\Rightarrow_{G_2} X'Y_1 \\ &\vdots \\ &\Rightarrow_{G_2} X'Y' \end{aligned}$$

а правилата

$$S \to_{G'} X'Y'$$

чрез изводи от вида

$$\begin{aligned} S &\Rightarrow_{G_2} A_1 \\ &\Rightarrow_{G_2} A_2 \\ &\vdots \\ &\Rightarrow_{G_2} A \\ &\Rightarrow_{G_2} XY \\ &\Rightarrow_{G_2} X_1Y \\ &\vdots \\ &\Rightarrow_{G_2} X'Y \\ &\Rightarrow_{G_2} X'Y_1 \\ &\vdots \\ &\Rightarrow_{G_2} X'Y' \end{aligned}$$

Оттук

$$L(G') \subseteq L(G_2).$$

Обратно, нека имаме извод в G_2 на дума с дължина поне 2, започващ от началния символ S. Преобразуваме този извод до еквивалентен на него, в който приложението на правилата са групирани, така че преименуващите правила да се прилагат непосредствено след появяването на символа, който преименуват. Ако преименуващите праивла са приложено още в самото начало то тогава извода започва с

$$\begin{aligned} S &\Rightarrow_{G_2} A_1 \\ &\Rightarrow_{G_2} A_2 \\ &\vdots \\ &\Rightarrow_{G_2} A \\ &\Rightarrow_{G_2} XY \\ &\Rightarrow_{G_2} X_1Y \\ &\vdots \\ &\Rightarrow_{G_2} X'Y \\ &\Rightarrow_{G_2} X'Y_1 \\ &\vdots \\ &\Rightarrow_{G_2} X'Y' \\ &\vdots \end{aligned}$$

като тази част може да бъде заместена с правилото

$$S \to_{G'} X'Y'.$$

Ако преименуващите правила са приложени в средата на извода, то тогава имаме

$$\begin{aligned} S &\Rightarrow^*_{G_2} wAw' \\ &\Rightarrow_{G_2} wX_1Yw' \\ &\Rightarrow_{G_2} wX_2Yw' \\ &\vdots \\ &\Rightarrow_{G_2} wX'Yw' \\ &\Rightarrow_{G_2} wX'Y_1w' \\ &\vdots \\ &\Rightarrow_{G_2} wX'Y'w' \\ &\vdots \end{aligned}$$

и тогава тази част от извода може да бъде заместено с правилото

$$A \rightarrow_{G'} X'Y'.$$

Следователно, за думи с дължина поне 2

$$w \in L(G) \iff w \in L(G').$$

□

Пример. Нека G е граматика със символи $\{S, A, B, 0, 1\}$, терминални символи $\{0, 1\}$, начален символ S и правила

$$\begin{aligned} S &\rightarrow A \mid B \\ A &\rightarrow \varepsilon \mid A01 \\ B &\rightarrow S \mid 0B1 \end{aligned}$$

Първо премахваме дългите правила и получаваме граматика с правила

$$\begin{aligned} S &\rightarrow A \mid B \\ A &\rightarrow \varepsilon \mid AA_1 \\ A_1 &\rightarrow 01 \\ B &\rightarrow S \mid 0B_1 \\ B_1 &\rightarrow B1 \end{aligned}$$

и нови нетерминални символи A_1 и B_1. За тази граматика

$$\mathcal{E} = \{A, S, B\}.$$

Образуваме нова граматика с правила

$$\begin{aligned} S &\rightarrow A \mid B \\ A &\rightarrow AA_1 \mid A_1 \\ A_1 &\rightarrow 01 \\ B &\rightarrow S \mid 0B_1 \\ B_1 &\rightarrow B1 \mid 1 \end{aligned}$$

За всеки символ на тази граматика пресмятаме множеството от достижимите символи и получаваме

$$\begin{aligned} D(0) &= \{0\} \\ D(1) &= \{1\} \\ D(S) &= \{S, A, B, A_1\} \\ D(A) &= \{A, A_1\} \\ D(A_1) &= \{A_1\} \\ D(B) &= \{B, S, A, A_1\} \\ D(B_1) &= \{B_1, 1\} \end{aligned}$$

Окончателно граматиката в нормална форма на Чомски, еквивалентна на оригиналната за думи с дължина поне 2, е

$$\begin{aligned} S &\to AA_1 \mid A_1A_1 \mid 0B_1 \mid 01 \\ A &\to AA_1 \mid A_1A_1 \\ A_1 &\to 01 \\ B &\to 0B_1 \mid 01 \\ B_1 &\to B1 \mid S1 \mid A1 \mid A_11 \end{aligned}$$

Ще използваме нормалната форма на Чомски за да построим алгоритъм, който разпознава дали една дума принадлежи на езика на граматиката. Нека G е граматика в нормална форма на Чомски. Нека w е дума с $|w| = n \geq 2$, съставена само от терминални символи и нека $w = a_1a_2 \dots a_n$. За всяко $1 \leq i \leq j \leq n$ образуваме множеството

$$N[i,j] = \{A \in \Gamma \mid A \Rightarrow_G^* a_ia_{i+1} \dots a_j\}.$$

Тогава

$$w \in L(G) \iff S \in N[1,n],$$

където S е началният символ на граматиката.

Да забележим, че

$$N[i,i] = \{a_i\}.$$

Тъй като G е в нормална форма на Чомски, то за $i < j$

$$A \Rightarrow_G^* a_ia_{i+1} \dots a_j \iff X \Rightarrow_G^* a_ia_{i+1} \dots a_k \text{ и } Y \Rightarrow_G^* a_{k+1} \dots a_j$$

за някое правило $A \to_G XY$ и някое $i \leq k \leq j$. Следователно

$$N[i,j] = \bigcup_{k=i}^{j} \{A \mid A \to_G XY \text{ за някои } X \in N[i,k],\ Y \in N[k,j]\}.$$

Пример. Да разгледаме граматиката с правила

$$\begin{aligned} S &\to AA_1 \mid A_1A_1 \mid 0B_1 \mid 01 \\ A &\to AA_1 \mid A_1A_1 \\ A_1 &\to 01 \\ B &\to 0B_1 \mid 01 \\ B_1 &\to B1 \mid S1 \mid A1 \mid A_11 \end{aligned}$$

терминални символи $\{0,1\}$ и начален символ S. Искаме да разберем дали думата 0011 принадлежи на езика на граматиката. За целта пресмятаме всички множества $N[i,j]$ за тази дума, започвайки от тези за които $j - i = 0$, продължавайки с тези, за които $j - i = 1$, след което тези, за които $j - i = 2$ и накрая $N[1,4]$. За улеснение попълваме резултатите в следната таблица

0			
$\emptyset$	0		
$\emptyset$	S, A_1, B	1	
S, B	B_1	$\emptyset$	1

и следователно $0011 \in L(G)$.

7.5 Лема за разрастването

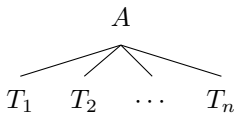
Ще казваме, че дървото на извода T' е *поддърво* на дървото на извода T (и ще пишем $T' \preceq T$), ако $T' \equiv T$ или T' участва в изграждането на T. По-точно, ако T е

$$a$$

или

$$\begin{array}{c} A \\ | \\ \varepsilon \end{array}$$

то единственото поддърво на T е самото T. Ако T има вида



то поддърветата на T са T и поддърветата на $T_1, \ldots, T_n$.

Лема 7.6. Нека G е безконтекстна граматика в нормална форма на Чомски. Нека T е дърво на извода в G с връх A и извод w. Нека T' е поддърво на T с връх B и извод α. Тогава съществуват думи w' и w'', такива че

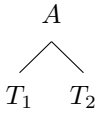
$$w = w'\alpha w''$$

и

$$A \Rightarrow_G^* w'Bw''$$

При това, ако $T \not\equiv T'$, то $|w'w''| \geq 1$.

Доказателство. Ще проведем индукция по височината на T. Ако височината на T е 0, то твърдението е очевидно. Нека сега твърдението е вярно за дървета с височина най-много h и нека височината на T е $h+1$. Ако $T' \equiv T$, то $B = A$ и твърдението е в сила при $w' = w'' = \varepsilon$. Нека сега $T' \not\equiv T$. Тъй като G е в нормална форма на Чомски, то T има вида



където T_1 и T_2 са дърветa с върхове съответно A_1 и A_2 и изводи съответно w_1 и w_2. Тъй като G е в нормална форма на Чомски, то $w_1 \neq \varepsilon$ и $w_2 \neq \varepsilon$. Дървото T' е поддърво на T_1 или на T_2. Нека за определеност $T' \preceq T_1$. Съгласно индукционното предположение, съществуват думи w_1' и w_1'', такива че

$$w_1 = w_1'\alpha w_1''$$

и

$$A_1 \Rightarrow_G^* w_1'Bw_1''$$

Оттук

$$A \Rightarrow_G A_1A_2 \Rightarrow_G^* w_1'Bw_1''A_2 \Rightarrow_G^* w_1'Bw_1''w_2$$

и следователно търсените думи w' и w'' са

$$w' = w_1' \text{ и } w'' = w_1''w_2.$$

□

Теорема 7.7 (Лема за разрастването)**.** Нека L е безконтекстен език. Тогава съществува число $n_0 > 0$, такова че за всяка $w \in L$ с $|w| > n_0$, съществуват думи x, y, z, u и v, такива че $w = xyzuv$, $|yzu| \leq n_0$, $|yu| \geq 1$ и за всяко $i \geq 0$ е изпълнено $xy^izu^iv \in L$.

Доказателство. Нека G е безконтекстна граматика в нормална форма на Чомски, която генерира точно думите от L с дължина поне 2. Нека броят на нетерминалните символи на G е m. Полагаме $n_0 = 2^m$. Нека сега $w \in L$ и $|w| > n_0$. Тогава $|w| > 2$ (защото в G има поне един нетерминален символ) и следователно в G има дърво на извода T с връх S — началния символ на граматиката, и извод w. Тъй като G е в нормлна форма на Чомски, то ако едно дърво на извода има височина q, то извода на това дърво е с дължина най-много 2^q. Оттук и $|w| > 2^m$ получаваме, че височината на дървото T е $h > m$. Образуваме редица от поддървета на T

$$T = T_0 \succ T_1 \succ T_2 \cdots \succ T_h$$

и нека T_i е с връх A_i и извод w_i за $0 \leq i \leq h$. Да забележим, че височината на $T_i = h - i$. Редицата

$$A_{h-m-1}, A_{h-m} \ldots, A_{h-1}$$

съдържа $m+1$ нетерминални символа и значи поне един от тях се повтаря два пъти. Нека $h-m-1 \leq j < k \leq h-1$ са такива, че $A_j = A_k$. От $T \succ T_j$ и предната лема, следва че съществуват думи x и v, такива че

$$w = xw_jv$$

и

$$S \Rightarrow_G^* xA_jv$$

От друга страна, от $T_j \succ T_k$ и лемата следва, че съществуват думи y и u, такива че

$$w_j = yw_ku$$

и

$$A_j \Rightarrow_G^* yA_ku$$

като при това $|yu| \leq 1$. Оттук, полагайки $z = w_k$ и използвайки $A_j = A_k$ получаваме

$$S \Rightarrow_G^* xA_jv \Rightarrow_G^* xyA_juv \Rightarrow_G^* xyyA_juuv \Rightarrow_G^* \cdots \Rightarrow_G^* xy^iA_jv^iz \Rightarrow_G^* xy^izu^iv$$

Освен това $yzu = w_j$ и тъй като височината на T_j е най-много m, то $|yzu| \leq 2^m = n_0$.

□

Пример. С помощта на лемата за разрастването ще докажем, че езикът

$$L = \{0^k1^k2^k \mid n \geq 0\}$$

не е безконтекстен. За целта нека $n > 0$. Да разгледаме думата

$$w = 0^n1^n2^n.$$

Тогава $w \in L$ и $|w| = 3n > n$. Нека x, y, z, u и v са думи, такива че $w = xyzuv$, $|yzu| \leq n$ и $|yu| \geq 1$. Тогава yzu не съдържа нули или не съдържа двойки (или и двети). Нека за определеност yzu не съдържа двойки (в другия случай разсъждаваме аналогично). Да разгледаме

$$w_0 = xy^0zu^0v = xzv.$$

Тогава w_0 съдържа точно n на брой двойки, но тъй като $|yu| \geq 1$, то $|w_0| < 3n$ и следователно

$$w_0 \notin L.$$

Оттук и лемата за разрастването следва, че L не е безконтекстен език.

Теорема 7.8. Безконтекстните езици не са затворени относно операцията сечение.

Доказателство. Нека

$$\begin{aligned} L_1 &= \{0^k1^k2^s \mid k, s \geq 0\} \\ L_2 &= \{0^k1^s2^s \mid k, s \geq 0\} \end{aligned}$$

Езикът L_1 е безконтекстен, тъй като той е конкатенацията на безконтекстния език $\{0^k1^k \mid k \geq 0\}$ и регулярния език $\{2^s \mid s \geq 0\}$. Езикът L_2 също е безконтекстен, тъй като той е конкатенацията на регулярния език $\{0^k \mid k \geq 0\}$ и безконтекстния език $\{1^s2^s \mid s \geq 0\}$. От друга страна

$$L_1 \cap L_2 = \{0^k1^k2^k \mid k \geq 0\},$$

който не е безконтекстен език.

□

Теорема 7.9. Безконтестните езици не са затворени относно операцията допълнение.

Доказателство. Съгласно правилото на Де Морган

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}.$$

Оттук, затвореността на безконтекстните езици относно операцията обединение и това, че те не са затворени относно операцията сечение, следва, че безконтексните езици не са затворени относно операцията допълнение.

□

7.6 Стекови автомати

Нека Σ е крайна азбука. *Стеков автомат над* Σ наричаме всяка наредена петорка $\mathcal{A} = (Q, \Gamma, q_0, \Delta, F)$, където Q е крайно множество (множество от състояния на автомата), Γ е крайна азбука (азбука на стека),

$$\Delta \subseteq (Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^*) \times (Q \times \Gamma^*)$$

е крайно множество (релация на преходите), $q_0 \in Q$ (начално състояние), а $F \subseteq Q$ (множество от финалните състояния).

Конфигурация на $\mathcal{A}$ ще наричаме всяка наредена тройка от вида

$$(q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*.$$

Конфигурации от вида

$$(q_0, w, \varepsilon)$$

наричаме начални, а конфигурации от вида

$$(f, \varepsilon, \varepsilon),$$

където $f \in F$, наричаме заключителни. Стековият автомат $\mathcal{A}$ дефинира релация $\vdash_{\mathcal{A}}$ между конфигурации (преобразуване на конфигурации) чрез

$$(q, aw, \alpha\gamma) \vdash_{\mathcal{A}} (q', w, \alpha'\gamma) \iff ((q, a, \alpha), (q', \alpha')) \in \Delta.$$

С $\vdash^*_{\mathcal{A}}$ ще означаваме рефлексивното и транзитивно затваряне на релацията $\vdash_{\mathcal{A}}$, т.е.

$$(q, w, \gamma) \vdash^*_{\mathcal{A}} (q', w', \gamma')$$

тогава и само тогава когато съществува редица

$$(q, w, \gamma) \vdash_{\mathcal{A}} (q_1, w_1, \gamma_1) \vdash_{\mathcal{A}} \cdots \vdash_{\mathcal{A}} (q_{n-1}, w_{n-1}, \gamma_{n-1}) \vdash_{\mathcal{A}} (q', w', \gamma')$$

(при $n = 0$ имаме $(q, w, \gamma) \vdash^*_{\mathcal{A}} (q, w, \gamma)$).

Език на стековия автомат $\mathcal{A}$ наричаме

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid (q_0, w, \varepsilon) \vdash^*_{\mathcal{A}} (f, \varepsilon, \varepsilon) \text{ за някое } f \in F\},$$

т.е. езикът на $\mathcal{A}$ се състои от думите w, за които $\mathcal{A}$ може да преобразува началната конфигурация (q_0, w, ε) до заключителна конфигурация.

Пример. Да разгледаме езикът L над $\{0, 1\}$, състоящ се от онези думи, които имат равен брой нули и единици. Естественият начин да разпознаем дали една дума w принадлежи на езика е да въведем един брояч S, който в началото е нула, след което да прочетем думата отляво надясно, като при всяко срещане на 0 прибавяме единица към S, а при всяко срещане на 1 вадим единица от S. Една дума w принадлежи на L тогава и само тогава, когато след прочитането на w, по описания по-горен начин, $S = 0$ (при $S > 0$ имаме повече нули, а при $S < 0$ имаме повече единици). За да реализираме горната процедура чрез стеков автомат, ще записваме S в специален унарен запис. По-точно, ако в даден момент $S \geq 0$ в стека ще пише $\underbrace{PP\ldots P}_{S}Z$, а когато $S \leq 0$, то в стека ще пише $\underbrace{NN\ldots N}_{-S}Z$.

Да разгледаме стековия автомат $\mathcal{A}$ над $\{0, 1\}$ със състояния A, B и C, от които A е начално, а C е финално, азбука на стека $\{Z, P, N\}$ и преходи

$((A, \varepsilon, \varepsilon), (B, Z))$	// отбелязваме дъното на стека със Z
$((B, 0, Z), (B, PZ))$	// ако сме на дъното на стека и четем 0, добавяме P
$((B, 1, Z), (B, NZ))$	// ако сме на дъното на стека и четем 1, добавяме N
$((B, 0, P), (B, PP))$	// ако най-горният елемента на стека е P и четем 0, добавяме P
$((B, 1, N), (B, NN))$	// ако най-горният елемента на стека е N и четем N, добавяме N
$((B, 0, N), (B, \varepsilon))$	// ако най-горният елемента на стека е N и четем 0, махаме N
$((B, 1, P), (B, \varepsilon))$	// ако най-горният елемента на стека е N и четем 1, махаме P
$((B, \varepsilon, Z), (C, \varepsilon)$	// ако сме на дъното на стека, можем да преминем към финалното състояние

Нека $w \in \{0, 1\}^*$ и разликата между нулите и единиците в w е m. Тъй като правилата на $\mathcal{A}$ са такива, че от състоянието B със всяка входна буква и всеки връх на стека можем отново да достигнем до състоянието B, то конфигурацията (B, w, Z) може да бъде преобразувана до конфигурация, започваща отново със състоянието B. Нека

$$(B, w, Z) \vdash^*_{\mathcal{A}} (B, \varepsilon, \gamma).$$

Тогава

$$m \geq 0 \Longrightarrow \gamma = P^m Z,$$

и

$$m \leq 0 \Longrightarrow \gamma = N^m Z.$$

В частност γ (съдържанието на стека) еднозначно се определя от w.

Ще докажем двете твърдения едновременно с индукция по дължината на w. Ако $w = \varepsilon$, то твърдението е очевидно. Нека сега твърдението е вярно за думи с дължина n и нека $|w| = n + 1$. Тогава $w = w'a$ за някоя дума w' с дължина n и $a \in \{0, 1\}$, като разликата между броя на нулите и единиците в w' е m'. От вида на преходите на $\mathcal{A}$ следва, че

$$(B, w', Z) \vdash_{\mathcal{A}}^{*} (B, \varepsilon, \gamma') \text{ и } (B, a, \gamma') \vdash_{\mathcal{A}} (B, \varepsilon, \gamma).$$

Нека първо $m > 0$ и $a = 0$. Тогава $m' \geq 0$ и следователно $\gamma' = P^{m'} Z$. Следователно към конфигурацията (B, a, γ') може да се приложи или второто правило от програмата (ако $m = 0$) или четвъртото правило от програмата (ако $m > 0$). И в двата случая получаваме, че $\gamma = P^{m'+1} Z = P^m Z$. Нека сега $a = 1$. Тогава $m' > 0$ и следователно $\gamma' = P^{m'} Z$. Единственото правило приложимо към конфигурацията (B, a, γ') е седмото и значи $\gamma = P^{m'-1} Z = P^m Z$.

Ако $m < 0$ разсъждаваме аналогично. Нека накрая $m = 0$. Тогава или $m' = 1$ и $a = 1$, или $m' = -1$ и $a = 0$. Да разгледаме първия случай. Тогава $\gamma' = PZ$. Единственото правило приложимо към конфигурацията (B, a, γ') е седмото и значи $\gamma = Z = P^m Z$. Вторият случай се разглежда аналогично.

Следователно, ако $w \in L$, то $m = 0$ и тогава имаме изпълнение

$$(A, w, \varepsilon) \vdash_{\mathcal{A}} (B, w, Z) \vdash_{\mathcal{A}}^{*} (B, \varepsilon, Z) \vdash_{\mathcal{A}} (C, \varepsilon, \varepsilon)$$

и следователно $w \in L(\mathcal{A})$.

Обратно, ако $w \in L(\mathcal{A})$, то има изпълнение на $\mathcal{A}$ трансформиращо началната конфигурация (A, w, ε) във финалната конфигурация $(C, \varepsilon, \varepsilon)$. Тъй като има само един преход, излизащ от A, и само един преход, влизащ в C, то тогава изпълнението на автомата има вида

$$(A, w, \varepsilon) \vdash_{\mathcal{A}} (B, w, Z) \vdash_{\mathcal{A}}^{*} (B, \varepsilon, Z) \vdash_{\mathcal{A}} (C, \varepsilon, \varepsilon)$$

и следователно $m = 0$, т.е. $w \in L$.

Теорема 7.10. Всеки безконтекстен език се разпознава от стеков автомат.

Доказателство. Нека L е безконтекстен език и нека $G = (\Gamma, \Sigma, R, S)$ е безконтекстна граматика с $L(G) = L$. Ще докажем, че L се разопзнава от стеков автомат, симулиращ най-левите изводи в G. Да разгледаме стековия автомат $\mathcal{A}$ над Σ, който има състояния $\{p, f\}$, като p е начално, а f — финално, азбука на стека Γ и преходи

$$\begin{array}{ll} ((p, \varepsilon, \varepsilon), (f, S)) & \\ ((f, \varepsilon, A), (f, \alpha)) & \text{// за всяко правило } A \to_G \alpha \\ ((f, a, a), (f, \varepsilon)) & \text{// за всяка буква } a \in \Sigma \end{array}$$

Единствената роля на началното състояние p е да постави началния символ S в дъното на стека. Ще докажем, че

$$(f, w, S) \vdash_{\mathcal{A}}^{*} (f, \varepsilon, \gamma) \iff S \Rightarrow_G^{*} w\gamma \text{ чрез най-ляв извод}$$

откъдето за всяка дума $w \in \Sigma^*$ ще имаме

$$(p, w, \varepsilon) \vdash_{\mathcal{A}}^{*} (f, \varepsilon, \varepsilon) \iff S \Rightarrow_G^{*} w,$$

т.е.

$$L(\mathcal{A}) = L(G).$$

Нека първо $(f, w, S) \vdash_{\mathcal{A}}^{*} (f, \varepsilon, \gamma)$. Ще докажем, че $S \Rightarrow_G^{*} w\gamma$ с индукция по броя на преходите, участващи в трансформацията на конфигурациите. Ако броят на преходите е 0, то $w = \varepsilon$, $\gamma = S$ и имаме $S \Rightarrow_G^{*} \varepsilon S$. Нека сега твърдението е вярно за трансформации, използващи n прехода и нека $(f, w, S) \vdash_{\mathcal{A}}^{*} (f, \varepsilon, \gamma)$ използва $n + 1$ прехода. Нека първо предположим, че последният преход е $((f, a, a), (f, \varepsilon))$. Тогава $w = w'a$, където a е буква и

$$(f, w, S) \vdash_{\mathcal{A}}^{*} (f, a, a\gamma) \vdash_{\mathcal{A}} (f, \varepsilon, \gamma).$$

Тогава

$$(f, w', S) \vdash_{\mathcal{A}}^{*} (f, \varepsilon, a\gamma)$$

чрез n прехода. Оттук и индукционното предположение $S \Rightarrow_G^{*} w'a\gamma$.

Нека сега последният преход е $((f, \varepsilon, A), (f, \alpha))$. Тогава $A \to_G \alpha$ и

$$(f, w, S) \vdash_{\mathcal{A}}^{*} (f, \varepsilon, A\gamma')$$

чрез n стъпки като $\gamma = \alpha\gamma'$. Оттук и индукционното предположение имаме

$$S \Rightarrow_G^{*} wA\gamma' \Rightarrow_G w\gamma.$$

За обратната посока, нека предположим, че $S \Rightarrow_G^{*} w\gamma$ чрез най-ляв извод, където $w \in \Sigma^*$. Ще докажем, че $(f, w, S) \vdash_{\mathcal{A}}^{*} (f, \varepsilon, \gamma)$ чрез индукция по дължината на извода в G. Ако дължината на извода е 0, то $w = \varepsilon$, $\gamma = S$ и имаме $(f, \varepsilon, S) \vdash_{\mathcal{A}}^{*} (f, \varepsilon, S)$. Нека сега твърдението е вярно за изводи с дължина n и нека $S \Rightarrow_G^{*} w\gamma$ чрез най-ляв извод с дължина $n + 1$. Тогава

$$S \Rightarrow_G^{*} w'A\gamma' \Rightarrow_G w'w''\gamma''\gamma',$$

където $w'w'' = w$, $\gamma''\gamma' = \gamma$, $A \to_G w''\gamma''$ и $S \Rightarrow_G^{*} w'A\gamma'$ чрез най-ляв извод члез n стъпки. Тогава, използвайки индукционното предположение, получаваме

$$\begin{aligned} (f, w, S) &\vdash_{\mathcal{A}}^{*} (f, w'', A\gamma') \\ &\vdash_{\mathcal{A}} (f, w'', w''\gamma''\gamma') \\ &\vdash_{\mathcal{A}}^{*} (f, \varepsilon, \gamma). \end{aligned}$$

□

Теорема 7.11. Езикът на всеки стеков автомат е безконтекстен.

Доказателство. Нека $\mathcal{A} = (Q, \Gamma, q_0, \Delta, F)$ е стеков автомат над Σ. Ще докажжем, че $L(\mathcal{A})$ е безконтекстен. Преди да започнем да строим граматика G с $L(G) = L(\mathcal{A})$ ще модифицираме $\mathcal{A}$ (без да променяме езика му), така че при всеки преход на автомата да четем точно един символ от стека, т.е. всеки преход да има вида

$$((q, x, A), (q', \gamma)), \text{ където } x \in \Sigma \cup \{\varepsilon\},\ A \in \Gamma,\ \gamma \in \Gamma^*.$$

Трансформираме $\mathcal{A}$ по следната схема:

1. Добавяме ново начално състояние q_0' и преход

   $$((q_0', \varepsilon, \varepsilon), (q_0, Z)),$$

   където Z е нов стеков символ, чиято единствена цел ще бъде да отбелязва дъното на стека. Тази стъпка се налага, защото във всяка начална конфитурация няма символ от стека, който да бъде прочетен.

2. Всички състояния от F престават да бъдат финални. Добавяме ново финално състояние f' и преходи

   $$((f, \varepsilon, Z), (f', \varepsilon)), \text{ за } f \in F.$$

   По този начин, всеки път когато стигнем до състояние от F (финално състояние за $\mathcal{A}$) и сме на дъното на стека, можем да премахнем символа Z от дъното на стека, да преминем във финалното състояние f' и да приключим изчислението. Ясно е, че тези първи две стъпки не променят езика на автомата.

3. Заместваме всеки преход на $\mathcal{A}$ от вида

   $$((q, x, A_1A_2 \dots A_k), (q', \gamma)),$$

   където $k \geq 2$ и $A_1, \dots, A_k \in \Gamma$, с преходите

   $$\begin{aligned} &((q, x, A_1), (q_1, \gamma)) \\ &((q_1, \varepsilon, A_2), (q_2, \varepsilon)) \\ &\vdots \\ &((q_{k-1}, \varepsilon, A_k), (q', \varepsilon)) \end{aligned}$$

   където $q_1, \dots, q_{k-1}$ са нови състояния. Ясно е, че и тази стъпка не променя езика на автомата.

4. Заместваме всеки преход на $\mathcal{A}$ от вида
$$((q, x, \varepsilon), (q', \gamma)),$$
с преходите
$$((q, x, A), (q', \gamma A)), \text{ за всеки символ } A \in \Gamma \cup \{Z\}.$$
Благодарение на символа Z, който стои на дъното на стека до края на изчислението, тези замени не променят езика на автомата.

За автомата $\mathcal{A}'$ е в сила $L(\mathcal{A}') = L(\mathcal{A})$. Освен, това за всяко състояние q, различно от новото начално състояние q_0', преходите, излизащи от q са от вида

$$((q, x, A), (q', \gamma)), \text{ където } x \in \Sigma \cup \{\varepsilon\},\ A \in \Gamma,\ \gamma \in \Gamma^*.$$

Сега се заемаме с конструирането на безконтекстна граматика G с $L(G) = L(\mathcal{A}')$. Терминалните символи на G са, разбира се, символите от Σ. Нетерминалните символи на G са символите

$$[q, A, p],$$

където q и p са състояния на $\mathcal{A}'$, различни от q_0', а $A \in \Gamma \cup \{Z\}$, като началният символ е $[q_0, Z, f']$. Целта ни е да направим граматиката G, така че за всеки нетерминален символ $[q, A, p]$ и всяка дума от нетерминални символи w да бъде в сила

$$[q, A, p] \Rightarrow_G^* w \iff (q, w, A) \vdash_{\mathcal{A}'}^* (p, \varepsilon). \tag{7.1}$$

За целта за всеки преход на $\mathcal{A}'$
$$((q, x, A), (q', B_1 B_2 \dots B_k)),$$
където
$$\begin{aligned} q &\neq q_0' \\ x &\in \Sigma \cup \{\varepsilon\} \\ k &\geq 0 \\ B_1, \dots, B_k &\in \Gamma \cup \{Z\} \end{aligned}$$
и всяка редица $p_1, \dots, p_{k-1}$ от състояния $\mathcal{A}'$, различни от q_0', въвеждаме в G правилото
$$[q, A, p] \to x[q', B_1, p_1][p_1, B_2, p_2] \dots [p_{k-1}, B_k, p]$$

Да отбележим, че при $k = 0$ правилото има вида
$$[q, A, p] \to x,$$

Сега ще докажем (7.1). Нека първо $[q, A, p] \Rightarrow_G^* w$. Ще докажем, че $(q, w, A) \vdash_{\mathcal{A}'}^* (q', \varepsilon)$ с индукция по дължината на извода в G. Дължината на извода не е нула, тъй като $[q, A, q']$ е нетерминален символ, а w се състои само от терминали. Ако дължината на извода е едно, то дефиницията на правилата на G ни дава, че $w = x$ за някое $x \in \Sigma \cup \{\varepsilon\}$, за което има преход
$$((q, x, A), (q', \varepsilon))$$
и следователно
$$(q, x, A) \vdash_{\mathcal{A}'} (q', \varepsilon, \varepsilon).$$

Нека сега $[q, A, p] \Rightarrow_G^* w$ за $n \geq 2$ стъпки и нека твърдението е вярно за изводи с дължини по-малки от n. Тогава извода има вида
$$[q, A, p] \Rightarrow_G x[q', B_1, p_1][p_1, B_2, p_2] \dots [p_{k-1}, B_k, p]$$
като
$$\begin{aligned} [q', B_1, p_1] &\Rightarrow_G^* w_1 \\ [p_1, B_2, p_2] &\Rightarrow_G^* w_2 \\ &\vdots \\ [p_{k-1}, B_k, p] &\Rightarrow_G^* w_k \end{aligned}$$

за по-малко от n стъпки и
$$w = x w_1 w_2 \dots w_k.$$

Така от една страна

$$(q, x, A) \vdash_{\mathcal{A}'} (q', \varepsilon, B_1 B_2 \dots B_k),$$

а от друга, съгласно индукционното предоложение

$$\begin{aligned} (q', w_1, B_1) &\vdash^*_{\mathcal{A}'} (p_1, \varepsilon, \varepsilon) \\ (p_1, w_2, B_2) &\vdash^*_{\mathcal{A}'} (p_2, \varepsilon, \varepsilon) \\ &\vdots \\ (p_{k-1}, w_k, B_k) &\vdash^*_{\mathcal{A}'} (p, \varepsilon, \varepsilon) \end{aligned}$$

и следователно

$$(q, \underbrace{x w_1 w_2 \dots w_k}_{w}, A) \vdash^*_{\mathcal{A}'} (p, \varepsilon, \varepsilon).$$

За обратното, нека $(q, w, A) \vdash^*_{\mathcal{A}'} (p, \varepsilon, \varepsilon)$. Ще докажем, че $[q, A, p] \Rightarrow^*_G w$ с индукция по броя на използваните преходи в трансформацията на конфигурациите (който е поне един). Ако е използван само един преход, то тогава $w = x$ за някое $x \in \Sigma \cup \{\varepsilon\}$ и има преход $((q, x, A), (p, \varepsilon))$, откъдето

$$[q, A, p] \Rightarrow_G x$$

Нека сега $(q, w, A) \vdash^*_{\mathcal{A}'} (p, \varepsilon, \varepsilon)$, използвайки $n \geq 2$ прехода и твърдението е вярно за трансформации, използващи по-малко от n прехода. Тогава трансформацията има вида

$$(q, \underbrace{x w'}_{w}, A) \vdash_{\mathcal{A}'} (q', w', B_1 B_2 \dots B_k) \vdash^*_{\mathcal{A}'} (p, \varepsilon, \varepsilon)$$

и значи имаме преход $((q, x, A), (q', B_1 B_2 \dots B_k))$ в $\mathcal{A}'$. Разделяме трансформацията $(q', w', B_1 B_2 \dots B_k) \vdash^*_{\mathcal{A}'} (p, \varepsilon, \varepsilon)$ на следните етапи

$$(q', \underbrace{w_1 w_2 \dots w_k}_{w'}, B_1 B_2 \dots B_k) \vdash^*_{\mathcal{A}'} (p_1, w_2 w_3 \dots w_k, B_2 B_3 \dots B_k) \tag{1}$$

$$(p_1, w_2 w_3 \dots w_k, B_2 \dots B_k) \vdash^*_{\mathcal{A}'} (p_3, w_3 \dots w_k, B_3 \dots B_k) \tag{2}$$

$$\vdots$$

$$(p_{k-1}, w_k, B_k) \vdash^*_{\mathcal{A}'} (p, \varepsilon, \varepsilon) \tag{k}$$

като в етапа (i) символът B_i е изваден от стека единствено и само посредством последния преход. Тогава

$$\begin{aligned} (q', w_1, B_1) &\vdash^*_{\mathcal{A}'} (p_1, \varepsilon, \varepsilon) \\ (p_1, w_2, B_2) &\vdash^*_{\mathcal{A}'} (p_2, \varepsilon, \varepsilon) \\ &\vdots \\ (p_{k-1}, w_k, B_k) &\vdash^*_{\mathcal{A}'} (p, \varepsilon, \varepsilon) \end{aligned}$$

като всяка от тези трансформации се извършва за по-малко от n стъпки. Оттук и индукционното предположение

$$\begin{aligned} [q', B_1, p_1] &\Rightarrow^*_G w_1 \\ [p_1, B_2, p_2] &\Rightarrow^*_G w_2 \\ &\vdots \\ [p_{k-1}, B_k, p] &\Rightarrow^*_G w_k \end{aligned}$$

От друга страна, тъй като имаме преход в $((q, x, A), (q', B_1 B_2 \dots B_k))$ в $\mathcal{A}'$, то в G имаме правило

$$[q, A, p] \to_G x[q' B_1 p_1][p_1 B_2 p_2] \dots [p_{k-1} B_k p].$$

Оттук и горните изводи в G получамаве

$$[q, A, P] \Rightarrow^*_G x w_1 \dots w_k = x w' = w,$$

с което еквивалентността (7.1) е доказана.

Оттук за произволна дума $w \in \Sigma^*$ имаме

$$\begin{aligned} w \in L(G) &\iff [q_0, Z, f'] \Rightarrow_G^* w \\ &\iff (q_0, w, Z) \vdash_{\mathcal{A}'}^* (f', \varepsilon, \varepsilon) \\ &\iff (q_0', w, \varepsilon) \vdash_{\mathcal{A}'}^* (f', \varepsilon, \varepsilon) \\ &\iff w \in L(\mathcal{A}'). \end{aligned}$$

□

7.7 Детерминирани стекови автомати

Казваме, че един стеков автомат $\mathcal{A}$ е *детерминиран*, ако за за всеки три иконфигурации (q, w, γ), (q', w', γ') и (q'', w'', γ'') от

$$\begin{aligned}(q, w, \gamma) &\vdash_{\mathcal{A}} (q', w', \gamma') \\ (q, w, \gamma) &\vdash_{\mathcal{A}} (q'', w'', \gamma'')\end{aligned}$$

следва, че

$$(q', w', \gamma') = (q'', w'', \gamma'').$$

Нека отбележим, че ако $\mathcal{A}$ е детерминиран, то за всяка конфигурация (q, w, γ) и всяко $n \geq 0$ съществува най-много една конфигурация (q', w', γ'), такава че

$$(q, w, \gamma) \vdash^*_{\mathcal{A}} (q', w', \gamma') \text{ за } n \text{ стъпки.}$$

Ще казваме, че две думи са *сравними*, ако едната е префикс на другата. Казваме, че два различни прехода $((q, x, \gamma), (q_1, \alpha))$ и $((q, x', \gamma'), (q'_1, \alpha'))$ са *съвместими*, ако x и x' сравними, и γ и γ' са сравними, т.е. ако двата преход са приложими към една и съща конфигурация. Тогава един стеков автомат $\mathcal{A}$ е детерминиран тогава и само тогава, когато в $\mathcal{A}$ *няма* съвместими преходи.

Пример. Да разгледаме езика $L = \{u2u^R \mid w \in \{0,1\}^*\}$. За да разпознаем дали една дума w е в L, достатъчно е да изпълним следното:

1. четем думата w отляво надясно;
2. преди да прочетем 2, всяка прочетен буква се слага в стека. По този начин в стека стои, прочетената до момента дума, като във върха на стека стои последната прочетена буква;
3. когато прочетем 2, продължаваме да четем думата w, но напредваме само, ако върха на стека (който махаме) съвпада с буквата, която четем. Така ще прочетем цялата дума след 2, само при положение, че преди да прочетем 2 сме прочели огледалния образ на това, което ни остава да четем след 2.

Формално, горният алгоритъм съответства на стеков автомат $\mathcal{A}$ над $\{0,1,2\}$ със състояния $\{s, f\}$, от които s е начално, а f е финално, азбука на стека $\{0,1,2\}$ и преходи

$$\begin{aligned}((s,0,\varepsilon),(s,0)) &\quad // \text{ четем 0 и слагаме 0 в стека} \\ ((s,1,\varepsilon),(s,1)) &\quad // \text{ четем 1 и слагаме 1 в стека} \\ ((s,2,\varepsilon),(f,\varepsilon)) &\quad // \text{ четем 2 и сменяме състоянието} \\ ((f,0,0),(f,\varepsilon)) &\quad // \text{ четем 0 и махаме 0 от стека} \\ ((f,1,1),(f,\varepsilon)) &\quad // \text{ четем 1 и махаме 1 от стека}\end{aligned}$$

Автоматът $\mathcal{A}$ е детерминиран. При това за всяка дума $u \in \{0,1\}^*$ имаме

$$\begin{aligned}(s, u, \varepsilon) &\vdash^*_{\mathcal{A}} (s, \varepsilon, u^R) \\ (s, u2, \varepsilon) &\vdash^*_{\mathcal{A}} (f, \varepsilon, u^R) \\ (f, u^R, u^R) &\vdash^*_{\mathcal{A}} (f, \varepsilon, \varepsilon)\end{aligned}$$

и следователно $L(\mathcal{A}) = L$.

Нека сега разгледаме $L' = \{uu^R \mid u \in \{0,1\}^*\}$. Той се разпознава от модифициран вариант $\mathcal{A}'$ на автомата $\mathcal{A}$, различаващ се само в преходите, а именно:

$$\begin{aligned}((s,0,\varepsilon),(s,0)) &\quad // \text{ четем 0 и слагаме 0 в стека} \\ ((s,1,\varepsilon),(s,1)) &\quad // \text{ четем 1 и слагаме 1 в стека} \\ ((s,\varepsilon,\varepsilon),(f,\varepsilon)) &\quad //\text{променен преход} \\ ((f,0,0),(f,\varepsilon)) &\quad // \text{ четем 0 и махаме 0 от стека} \\ ((f,1,1),(f,\varepsilon)) &\quad // \text{ четем 1 и махаме 1 от стека}\end{aligned}$$

Чрез промяната на третия преход позволяваме автомата недетерминистично да реши, че е достигната средата на думата, като след това стартира процеса по проверка, дали оставащата част от думата е обърната дума на тази прочетена до момента.

Ще казваме, че безконтекстният език $L \subseteq \Sigma^*$ е *детерминиран*, ако езикът $L\#$, където $\# \notin \Sigma$ е нов символ, отбелязващ края на думата, се разпознава от детерминиран стеков автомат. Причината да въведем допълнителен символ, отбелязващ края на думата е заради изискването ни за празен стек при разпознаване на думата. Ако не отбелязваме края на думата, то ще ограничим значително класът на детерминираните безконтекстни езици. Така например езикът $L = \{0^m \mid m \geq 0\} \cup \{0^n 1^n \mid n \geq 0\}$ не се разпознава от детерминиран стеков автомат, докато езикът $L\#$ се разпознава.

Теорема 7.12. Детерминираните безконтекстни езици са затворени относно операцията допълнение.

Доказателство. Нека $L \subseteq \Sigma^*$ е детерминиран безконтекстен език и нека $\mathcal{A}$ е детерминиран автомат, разпознаващ $L\#$. Използвайки конструкцията от доказателството на теоремата относно това, че езикът на един стеков автомат е безконтекстен, можем да считаме, че единствената работа на началното състояние е да отбележи дъното на стека със символа Z и всеки преход на автомата, който не излиза от началното състояние, има вида

$$((q, x, A), (q', \alpha)),$$

където x е буква или ε, а A е стеков символ.

Ще казваме, че една конфигурация (q, w, γ) е неперспективна, ако от

$$(q, u, \gamma) \vdash_{\mathcal{A}}^* (q', u', \gamma')$$

следва, че

$$u = u' \text{ и } |\gamma| \leq |\gamma'|,$$

т.е. една конфигурация е неперспективна, ако започвайки от нея $\mathcal{A}$ не може нито да напредне в четенето на входната дума, нито да намали дължината на думата съхранявана в стека. Тъй като $\mathcal{A}$ е детерминиран, ако $\mathcal{A}$ започне от началната конфигурация за дума $w\#$ и достигне до неперспективна конфигурация, то $\mathcal{A}$ не разпознава $w\#$. От друга страна, ако $\mathcal{A}$ няма нетривиални неперспективни конфигурации, то за всеки вход $w\#$ ще съществува състояние q, такова че

$$(q_0, w\#, \varepsilon) \vdash_{\mathcal{A}}^* (q, \varepsilon, \varepsilon).$$

Наистина в случай, че $\mathcal{A}$ няма неперспективни конфигурации, то за всяка конфигурация (q, u, γ) има конфигурация (q', u', γ'), такива че

$$(q, u, \gamma) \vdash_{\mathcal{A}}^* (q', u', \gamma')$$

и

$$|u| > |u'| \text{ или } |\gamma| > |\gamma'|$$

Тъй като със сигурност $|u| \geq |u'|$, то

$$(|u'|, |\gamma'|) \prec (|u|, |\gamma|),$$

където $\prec$ е лексикографската наредба в $\mathbb{N} \times \mathbb{N}$, т.е.

$$(n', m') \prec (n, m) \iff n' < n, \text{ или } n' = n \text{ и } m' < m.$$

Тъй като $\prec$ е добра наредба (т.е. в нея има най-малък елемент, а именно $(0, 0)$, и няма безкрайна намаляваща редица), то в крайна сметка дължините на втората и третата компонента на конфигурацията ще станат 0, т.е. ще прочетем целия вход и ще изпразним стека.

От специалния вид на автомата следва, че една нетривиална конфигурация (q, u, γ) е неперспективна тогава и само тогава, когато конфигурацията (q, x, A), където x е първата буква на u, а A е върхът на стека, е неперспективна.

Модифицираме $\mathcal{A}$ по следния начин:

1. Добавяме нови състояния r и r'.
2. За всяка неперспективна конфигурация (q, a, A), където a и A са символи, премахваме преходите, съвместими с нея (т.е. преходите от вида $((q, a, A), (q', \alpha))$ и $(q, \varepsilon, A), (q', \alpha))$), и добавяме прехода $((q, a, A), (r, \varepsilon))$. След тази манипулация езикът на автомата остава непроменен.

3. Добавяме преходи $((r, a, \varepsilon), (r, \varepsilon))$ за всяка буква $a \in \Sigma$ и прехода $(r, \#, \varepsilon), (r', \varepsilon))$. По този начин, в случай, че автоматът достигне състоянието r, то той ще прочете целия вход, след което ще премине в състоянието r'. По този начин за всеки вход или автомата приема входната дума, или завършва с празен стек в състояние, което не е финално, или изчита целия вход и попада в състоянието r'

4. Добавяме преходите $((r', \varepsilon, A), (r', \varepsilon))$ за всеки стеков символ A включително символа, който бележи дъното на стека. По този начин, след като автомата влезе в състоянието r' единственото нещо, което може да направи е да изпразни стека.

Нека означим получения автомат с $\mathcal{A}'$. Ясно е, че $L(\mathcal{A}) = L(\mathcal{A}')$, като при това за всеки вход автоматът $\mathcal{A}'$ изчита целия вход и изпразва стека. Сега, ако разменим финалностите на състоянията получаваме стеков автомат, разпознаващ езика $(\Sigma^* - L)\#$.

□

Следствие 7.13. Не всеки безконтекстен език е детерминиран.

Освен за бързото разпознаване на това дали една дума е от даден детерминиран безконтекстен език детерминираните стекови автомати се използват и като генератори на дървета на извод.

Пример. Да разгледаме езика $\{0^k 1^k \mid k \geq 0\}$. Този език се генерира от граматиката с правила

$$S \to \varepsilon \mid 0S1.$$

Следователно той се разпознава от автомата със състояния $\{s, f\}$ (s —начално; f — финално) и правила

$$\begin{array}{l}((s, \varepsilon, \varepsilon), (f, S))\\ ((f, 0, 0), (f, \varepsilon))\\ ((f, 1, 1), (f, \varepsilon))\\ ((f, \varepsilon, S), (f, \varepsilon))\\ ((f, \varepsilon, S), (f, 0S1))\end{array}$$

Този автомат е недетерминиран, като единствения недетерминизъм е в предпоследния и последния преход: при положение, че най-горната буква в стека е S, не е ясно дали трябва да изтрием S от стека, или да го заместим с $0S1$. Първия случай съответства на това, че сме прочели всичкки нули и трябва вече да четем 1, а втория — че има още нули за четене. Въпреки това, езикът е детерминиран. За да намерим детерминиран автомат, разпознаващ думите от езика с отбелязан край на думата, достатъчно е преди да предприемем действие със стека да „погледнем“ още една стъпка напред във входната дума. Това поглеждане напред можем да осъществим, отлагайки работата със стека, четейки само от входната лента (фиксиран брой символи — в случая един) и „запаметявайки“ прочетеното в състоянията на автомата. По-точно, детерминиран автомат разпознаващ думите от езика с отбелязан край на думата е

$$\begin{array}{rl}((s, \varepsilon, \varepsilon), (f, S)) & \\ ((f, 0, \varepsilon), (f_0, \varepsilon)) & \text{// прочитаме 0 и запаметяваме в } q_0\\ ((f_0, \varepsilon, 0), (f, \varepsilon)) & \\ ((f, 1, \varepsilon), (f_1, \varepsilon)) & \text{// прочитаме 1 и запаметяваме в } q_1\\ ((f_1, \varepsilon, 1), (f, \varepsilon)) & \\ ((f_0, \varepsilon, S), (f_0, 0S1)) & \text{//съответства на правилото } S \to 0S1\\ ((f_1, \varepsilon, S), (f_1, \varepsilon)) & \text{//съответства на правилото } S \to \varepsilon\\ ((f, \#, \varepsilon), (f_\#, \varepsilon)) & \end{array}$$

където началното състояние е s, а финално е само състоянието $f_\#$. Следейки преходите, които използваме за да трансформираме начална конфигурация в заключителна, ние установяваме кои са правилата от първоначалната граматика, прилагани в най-левия извод на думата. Да разгледаме например думата 0011. Изпълнението на

автомата върху нея е

$$\begin{aligned}
(s, 0011\#, \varepsilon) &\vdash (f, 0011\#, S) \\
&\vdash (f_0, 011\#, S) \\
&\vdash (f_0, 011\#, 0S1) && //S \to 0S1 \\
&\vdash (f_0, 011\#, S1) \\
&\vdash (f_0, 11\#, S1) \\
&\vdash (f_0, 011\#, 0S11) && //S \to 0S1 \\
&\vdash (f, 11\#, S11) \\
&\vdash (f_1, 1\#, S1) \\
&\vdash (f_1, 1\#, 11) && //S \to \varepsilon \\
&\vdash (f, 1\#, 1) \\
&\vdash (f_1, \#, 1) \\
&\vdash (f, \#, \varepsilon) \\
&\vdash (f_\#, \varepsilon, \varepsilon)
\end{aligned}$$

Пример. Нека сега разгледаме еднозначната граматика, генерираща аритмитични изрази, с правила

$$\begin{aligned}
E &\to E + T \mid T \\
T &\to T.F \mid F \\
F &\to \mathrm{id} \mid (E)
\end{aligned}$$

Правилата за F пораждат недетерминизъм, който би могъл да се избегне чрез метода, показан в предния пример (а именно, чрез предварително четене на фиксирана част от входната дума и запаметяване на прочетеното с помощта на състоянията на автомата). От друга страна правилата за E и T пораждат друг тип недетерминизъм, който няма как да бъде преодолян по тази схема.

Да разгледаме правилата за E. Изводите, използващи тези две правила, имат вида

$$E \Rightarrow E + T \to E + T + T \to \cdots \Rightarrow E + T + \cdots + T \Rightarrow T + T + \cdots + T$$

За да разберем, в кой момент трябва да приложим правилото $E \to T$ е необходимо да знаем предварително, точно колко символа $+$ имаме в израза, който искаме да генерираме. Това обаче ние няма как да разберем при положение, че имаме право само един прочит на думата. За да избегнем тази недетерминираност заместваме правилата $E \to E + T \mid T$ с правилата

$$\begin{aligned}
E &\to TE' \\
E' &\to \varepsilon \mid +TE'
\end{aligned}$$

Ясно е, че оригиналните правила както и тези генерират едни и същи думи. От друга страна вече нямаме недетерминираност за символа E, а недетерминираността за символа E' се разрешава, чрез предварително четене и запаметяване на входа. Прилагайки същото преобразуване за правилата за T, получаваме следната граматика, еквивалентна на оригиналната:

$$\begin{aligned}
E &\to TE' \\
E' &\to \varepsilon \mid +TE' \\
T &\to FT' \\
T' &\to \varepsilon \mid .FT' \\
F &\to \mathrm{id} \mid (E)
\end{aligned}$$

От тази граматика получаваме детерминирания стеков автомат

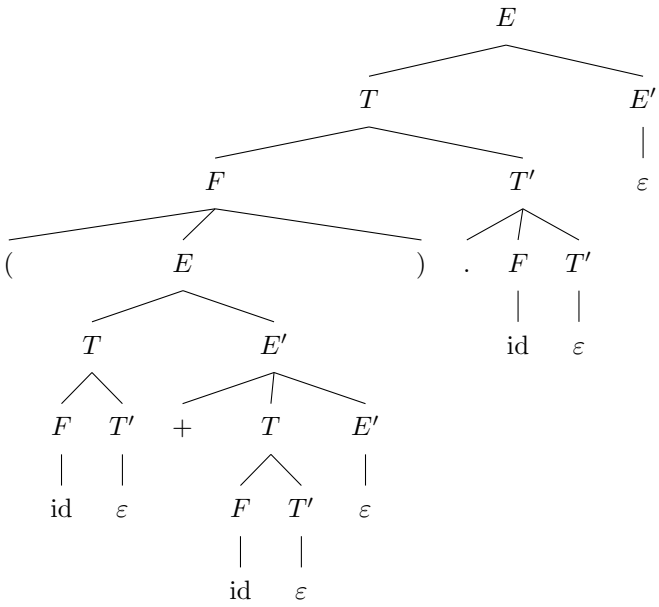
$$
\begin{aligned}
&((s,\varepsilon,\varepsilon),(f,E))\\
&((f,a,\varepsilon),(f_a,\varepsilon)) && \text{за } a\in\{\mathrm{id},+,.,(,),\#\}\\
&((f_a,\varepsilon,a),(f,\varepsilon)) && \text{за } a\neq\#\\
&((f_a,\varepsilon,E),(f_a,TE')) && \text{за всяко } a\ //E\to TE'\\
&((f_+,\varepsilon,E'),(f_+,+TE')) && //E'\to +TE'\\
&((f_a,\varepsilon,E'),(f_a,\varepsilon) && \text{за } a\in\{),\#\}//E'\to\varepsilon\\
&((f_a,\varepsilon,T),(f_a,FT')) && \text{за всяко } a//\ T\to FT'\\
&((f_.,\varepsilon,T'),(f_.,.FT')) && //T'\to .FT'\\
&(f_a,\varepsilon,T'),(f_a,\varepsilon)) && \text{за } a\in\{+,),\#\}\ //T'\to\varepsilon\\
&((f_(,\varepsilon,F),(f_(,(E))) && //F\to(E)\\
&((f_{\mathrm{id}},\varepsilon,F),(f_{\mathrm{id}},\mathrm{id}) && //F\to\mathrm{id}
\end{aligned}
$$

Нека видим как работи автоматът върху входа $(\mathrm{id}+\mathrm{id}).\mathrm{id}\#$

$$
\begin{aligned}
(s,(\mathrm{id}+\mathrm{id})\mathrm{id}\#,\varepsilon) &\vdash (f,(\mathrm{id}+\mathrm{id}).\mathrm{id}\#,E)\\
&\vdash (f_(,\mathrm{id}+\mathrm{id}).\mathrm{id}\#,E)\\
&\vdash (f_(,\mathrm{id}+\mathrm{id}).\mathrm{id}\#,TE') && //E\to TE'\\
&\vdash (f_(,\mathrm{id}+\mathrm{id}).\mathrm{id}\#,FT'E') && //T\to FT'\\
&\vdash (f_(,\mathrm{id}+\mathrm{id}).\mathrm{id}\#,(E)T'E') && //F\to(E)\\
&\vdash (f,\mathrm{id}+\mathrm{id}).\mathrm{id}\#,E)T'E'))\\
&\vdash (f_{\mathrm{id}},+\mathrm{id}).\mathrm{id}\#,E)T'E'))\\
&\vdash (f_{\mathrm{id}},+\mathrm{id}).\mathrm{id}\#,TE')T'E')) && //E\to TE'\\
&\vdash (f_{\mathrm{id}},+\mathrm{id}).\mathrm{id}\#,FT'E')T'E')) && //T\to FT'\\
&\vdash (f_{\mathrm{id}},+\mathrm{id}).\mathrm{id}\#,\mathrm{id}T'E')T'E')) && //F\to\mathrm{id}\\
&\vdash (f,+\mathrm{id}).\mathrm{id}\#,T'E')T'E'))\\
&\vdash (f_+,\mathrm{id}).\mathrm{id}\#,T'E')T'E'))\\
&\vdash (f_+,\mathrm{id}).\mathrm{id}\#,E')T'E')) && //T'\to\varepsilon\\
&\vdash (f_+,\mathrm{id}).\mathrm{id}\#,+TE')T'E')) && //E'\to +TE'\\
&\vdash (f,\mathrm{id}).\mathrm{id}\#,TE')T'E'))\\
&\vdash (f_{\mathrm{id}},).\mathrm{id}\#,TE')T'E'))\\
&\vdash (f_{\mathrm{id}},).\mathrm{id}\#,FT'E')T'E')) && //T\to FT'\\
&\vdash (f_{\mathrm{id}},).\mathrm{id}\#,\mathrm{id}T'E')T'E')) && //F\to\mathrm{id}\\
&\vdash (f,).\mathrm{id}\#,T'E')T'E'))\\
&\vdash (f_),.\mathrm{id}\#,T'E')T'E'))\\
&\vdash (f_),.\mathrm{id}\#,E')T'E')) && //T'\to\varepsilon\\
&\vdash (f_),.\mathrm{id}\#,)T'E')) && //E'\to\varepsilon\\
&\vdash (f,.\mathrm{id}\#,T'E'))\\
&\vdash (f_.,\mathrm{id}\#,T'E'))\\
&\vdash (f_.,\mathrm{id}\#,.FT'E')) && //T'\to .FT'\\
&\vdash (f,\mathrm{id}\#,FT'E'))\\
&\vdash (f_{\mathrm{id}},\#,FT'E'))\\
&\vdash (f_{\mathrm{id}},\#,\mathrm{id}T'E')) && //F\to\mathrm{id}\\
&\vdash (f,\#,T'E'))
\end{aligned}
$$

$$
\begin{aligned}
&\vdash (f_\#,\varepsilon,T'E'))\\
&\vdash (f_\#,\varepsilon,E')) && //T'\to\varepsilon\\
&\vdash (f_\#,\varepsilon,\varepsilon)) && //E'\to\varepsilon
\end{aligned}
$$

Оттук получаваме следното дърво на извода


E
T
E′
F
T′
ε
(
E
)
.
F
T′
T
E′
id
ε
F
T′
+
T
E′
id
ε
F
T′
ε
id
ε

глава 8

Машини на Тюринг

8.1 Еднолентови машини на Тюринг

Машините на Тюринг са абстрактни изчислителни устройства, чиято цел е да могат да симулират алгоритмичните формални изчисления, извършвани от човек. Всяка машина на Тюринг разполага с неограничено количество памет във формата на безкрайна лента (разделена на клетки), четяща глава и крайна програма. Всяка клетка на лентата е или празна или съдържа един символ от от фиксирана азбука. Машината работи на тактове (равни интервали от време). На всеки такт от работата на машината само краен брой клетки от лентата не са празни. Четящата глава на машината разглежда точно една клетка от лентата, като разпознава дали клетка е празна или в нея има разположен символ от азбуката. За да опростим разглежданията ще считаме, че цялата лента е запълнена със символи от крайна азбука Γ, съдържаща съмивол $\sqcup$, маркиращ празна клетка, като във всеки един момент от изпълнението само краен брой клетки съдържат символ, различен от $\sqcup$. *Програмата* на машината на Тюринг е организирана с помощта на *състояния*, като на всеки такт действието на машината се определя еднозначно от състоянието, в което се намира, и това което вижда четящата глава. На всеки такт машината извършва точно едно от следните три действия:

- записва дадена буква в клетката, над която се намира четящата глава;
- мести четящата глава наляво;
- мести четящата глава надясно.

За допълнително улеснение ще считаме, че лентата е неограничена само надясно, т.е. има ляв край. Този ляв край на лентата ще бележим със символа $\triangleright$, като считаме, че той стои в най-лявата клетка на лентата.

По-формално, *машина на Тюринг* наричаме всяка наредена петорка $\mathcal{M} = (K, \Gamma, \delta, s, H)$, където

- K е крайно множество (състояния на машината);
- Γ е крайна азбука, съдържаща $\sqcup$, но *не* съдържаща символите $\triangleright$, $\rightarrow$ и $\leftarrow$;
- δ е изображение от $(K - H) \times (\Gamma \cup \{\triangleright\})$ в $K \times (\Gamma \cup \{\rightarrow, \leftarrow\})$;
- за всеки две състояния q и q', ако $\delta(q, \triangleright) = (q', y)$, то y е $\rightarrow$;
- $s \in K$ — начално състояние на $\mathcal{M}$;
- $H \subseteq K$ — множество от „стоп“ състояния.

Изображението δ е дефинирано за всяка двойка (q, x), дъдето $q \in K$ не е „стоп“ състояние, а x и символ, който може да бъде разположен на върху лентата на машината, като то изобразява (q, x) в (q', x'), където q' може да бъде произволно състояние, а смисълът на x' е следния: ако x е от Γ, то машината записва x в клетката, над която се намира четящата глава; ако x е $\rightarrow$, то машината премества четящата глава надясно; ако x е $\leftarrow$, то машината премества четящата глава наляво. При това, ако четящата глава чете символа $\triangleright$, единственото, което може да направи машината, е да премести четящата глава надясно. В случай, че машината достигне до състояние от H, то машината спира изпълнението.

Всяко изпълнение на машината се определя еднозначно от състоянието, в което машината се намира, символите, записани на лентата, и позицията на четящата глава. Тъй като нашата лента е ограничена отляво чрез символа $\triangleright$ и тя може да съдържа само краен брой символи, различни от $\sqcup$, то достатъчно е да знаем символите, които са записани между символа $\triangleright$ и последния символ, различен от $\sqcup$, или позицията на четящата глава, в случай че тя

се намира вдясно от последната непразна клетка. Тази информация представлява крайна дума σ, започваща с $\triangleright$. Позицията на четящата глава ще отбелязваме, подчертавайки символа от σ, който тя вижда в момента.

Конфигурация ще наричаме всяка двойка (q, σ), където q е състояние, а σ е дума от вида $\triangleright u$, $u \in \Gamma^*$, в която точно един символ е подчертан, като в случай че последния символ е $\sqcup$, то именно той е подчертаният. където $u \in \Gamma^*$, а v е или празна, или завършва на символ различен от $\sqcup$. Ако $q \notin H$, ще казваме че конфигурацията е нефинална, а ако $q \in H$, че конфигурацията е финална.

Всяка машина на тюринг $\mathcal{M}$ дефинира релацията $\vdash_{\mathcal{M}}$ (трансформация на конфигурации) между конфигурации по следния начин: Нека имаме нефинална конфигурация $(q, u\underline{x}v)$. Нека $\delta(q, x) = (q', x')$. Тогава

1. Ако $x' \in \Gamma$, то
$$(q, u\underline{x}v) \vdash_{\mathcal{M}} (q', u\underline{x'}v).$$
2. Ако x' е $\leftarrow$ и u е с най-десен символ y, т.е. $u = u'y$, то
$$(q, u'y\underline{x}v) \vdash_{\mathcal{M}} (q', u'\underline{y}xv).$$
3. Ако x' е $\rightarrow$ и v е непразна дума с най-ляв символ y, т.е. $v = yv'$, то
$$(q, u\underline{x}yv') \vdash_{\mathcal{M}} (q', ux\underline{y}v').$$
4. Ако $x' = \rightarrow$ и v е празната дума, то
$$(q, u\underline{x}) \vdash_{\mathcal{M}} (q', ux\underline{\sqcup}).$$

Тъй като δ е функция, то за всяка нефинална конфигурация (q, σ) съществува единствена конфигурация (q', σ'), такава че
$$(q, \sigma) \vdash_{\mathcal{M}} (q, \sigma').$$

С $\vdash^n_{\mathcal{M}}$ ще означаваме резултата от $n \geq 0$ последователни прилагания на релацията $\vdash_{\mathcal{M}}$. С други думи за две конфигурации C и C' е в сила $C \vdash^n_{\mathcal{M}} C'$ тогава и само тогава,
$$C \vdash_{\mathcal{M}} C_1 \vdash_{\mathcal{M}} \cdots \vdash_{\mathcal{M}} C_{n-1} \vdash_{\mathcal{M}} C'$$
като при $n = 0$ имаме
$$C \vdash^0_{\mathcal{M}} C' \iff C = C'.$$

С $\vdash^*_{\mathcal{M}}$ ще означаваме рефлексивното и транзитивно затваряне на релацията $\vdash_{\mathcal{M}}$. С други думи
$$C \vdash^*_{\mathcal{M}} C' \iff C \vdash^n_{\mathcal{M}} C' \text{ за някое } n \geq 0.$$

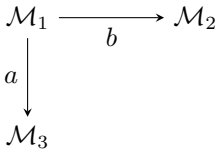
Пример. Да разгледаме машината $\mathcal{M}$, която има азбука $\{0, 1, \sqcup\}$, състояния $\{s, q, f\}$, като $H = \{f\}$, и преходи

$$\begin{aligned}
\delta(s, 0) &= (q, 1)\\
\delta(s, 1) &= (q, 0)\\
\delta(s, \sqcup) &= (f, \sqcup)\\
\delta(s, \triangleright) &= (s, \rightarrow)\\
\delta(q, x) &= (s, \rightarrow) \text{ за } x \in \{0, 1, \sqcup, \triangleright\}
\end{aligned}$$

Машината $\mathcal{M}$ преборазува конфигурацията $\triangleright 0s110$ по следния начин

$$\begin{aligned}
(s, \triangleright\underline{0}110) \quad &\vdash_{\mathcal{M}} \quad (q \triangleright \underline{1}110)\\
&\vdash_{\mathcal{M}} \quad (s, \triangleright 1\underline{1}10)\\
&\vdash_M c \quad (q, \triangleright 1\underline{0}10)\\
&\vdash_M c \quad (s, \triangleright 10\underline{1}0)\\
&\vdash_M c \quad (q, \triangleright 100\underline{0})\\
&\vdash_M c \quad (s, \triangleright 1000\underline{0})\\
&\vdash_M c \quad (q, \triangleright 1001\underline{1})\\
&\vdash_M c \quad (s, \triangleright 1001\underline{\sqcup})\\
&\vdash_M c \quad (f, \triangleright 1001\underline{\sqcup})
\end{aligned}$$

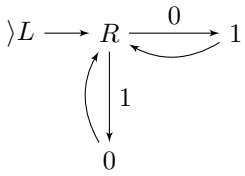
Пример. Елементарните машини на Тюринг над дадена фиксирана азбука Γ дефинираме по следния начин: за всяко $z \in \Gamma \cup \{\leftarrow, \rightarrow\}$ с $\mathcal{M}_z$ ще означаваме машината със две състояния, да речем s и h, като s е начално, а h е „стоп“ състояние, а преходите са $\delta(s,x) = (h,z)$ за всяко $x \in \Gamma$ и $\delta(s, \triangleright) = (s, \rightarrow)$. Машината $\mathcal{M}_a$ за $a \in \Gamma$ записва символа a върху клетката, която разглежда четящата глава. Тези машини ще означаваме с буквата, която записват. Машината $\mathcal{M}_\leftarrow$ премества четящата глава наляво и спира, а машината $\mathcal{M}_\rightarrow$ премества четящата глава надясно и спира. Тези машини ще означаваме съответно с L и R. Последователното изпълнение на елементарни машини ще записваме като дума над азбуката $\Gamma \cup \{L, R\}$. Така например с $RaRR \sqcup L$ ще означаваме машината, която премества четящата надясно, пише буквата a, премества четящата глава две клетки надясно, изтрива съдържанието на клетката и премества главата наляво. Освен чрез последователно изпълнение можем да свързваме и машини и с условни преходи зависещи от символа, който вижда четящата глава. Така например



изпълнява първо $\mathcal{M}_1$, като след като тя приключи, ако четящата глава вижда буквата a, то започваме да изпълняваме $\mathcal{M}_3$, а ако вижда b изпълняваме $\mathcal{M}_2$. Тази схема можем да реализираме по следния естествен начин: Нека $\mathcal{M}_i = (K_i, \Gamma, \delta_i, s_i, H_i)$ за $i = 1,2,3$, като K_1, K_2 и K_3 нямат общи елементи. На схемата описана по-горе съответвства машина $\mathcal{M} = (K, \Gamma, \delta, s, H)$, където

$$\begin{aligned}
K &= K_1 \cup K_2 \cup K_3 \cup \{h\}, \text{ където } h \notin K_1 \cup K_2 \cup K_3 \\
\delta &= \delta_1 \cup \delta_2 \cup \delta_3 \cup \\
&\quad \{((h_1, a), (s_3, a) \mid h_1 \in H_1\} \cup \\
&\quad \{((h_1, b), (s_2, b) \mid h_1 \in H_1\} \cup \\
&\quad \{((h_1, x), (h, x) \mid h_1 \in H_1,\ x \neq a, b\} \\
s &= s_1 \\
H &= H_2 \cup H_3 \cup \{h\}.
\end{aligned}$$

Машината от предния пример можем да опишем по следния начин



където в случай, че дадена стрелка няма етикет, то това означава, че правим този преход независимо от символа, който вижда четящата глава.

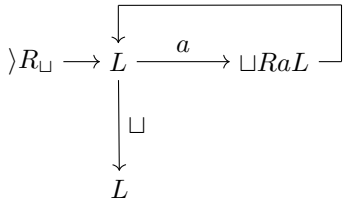
Пример. Машините

където преход с етикет $\overline{a}$ се извършват по всички символи, различни от a, преместват четящата глава съответно надясно и наляво, докато не срещнат символа a. Ще отбелязваме тези машини с R_a и L_a.

Машините

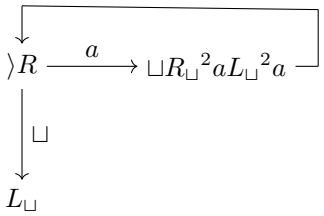
предвижват четящата глава съответно надясно и наляво, докато не достигнат до символ, различен от a.

Пример. Машината $S_{\rightarrow}$



преобразува $\sqcup w\sqcup$, където w е непразна дума, несъдържаща $\sqcup$, в $\sqcup\sqcup w\sqcup$, т.е. премества w с една позиция надясно.

Пример. Машината C



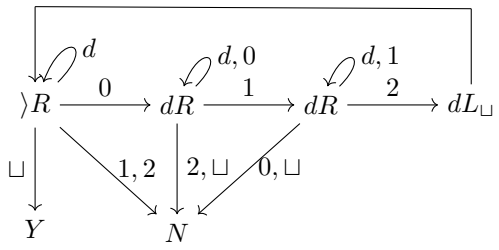
преобразува $\sqcup w\sqcup$, където w е непразна дума, несъдържаща $\sqcup$, в $\sqcup w\sqcup w\sqcup$, т.е. прави копие на w отдясно на w.

8.2 Разрешими езици. Изчислими функции.

Ще казваме, че машината на Тюринг $\mathcal{M} = (K, \Gamma, \delta, s, H)$ е *разпознавател*, ако $H = \{y, n\}$. Финална конфигурация от вида (y, σ) ще наричаме *приемаща*, а такава от вида (n, σ) — *отхвърляща*.

Нека Σ е крайна азбука, несъдържаща $\sqcup$. Ще казваме, че $L \subseteq \Sigma^*$ е *разрешим*, ако съществува разпознавател $\mathcal{M}$, такъв че за всяко $w \in \Sigma^*$, $\mathcal{M}$ преобразува $(s, \triangleright\underline{\sqcup}w)$ до приемаща конфигурация, в случай че $w \in L$, и до неприемаща конфигурация, в случай че $w \notin L$ (тук s е началното състояние на $\mathcal{M}$). С други думи, L е разрешим, ако съществува машина на Тюринг, която започвайки от конфигурацията $(s, \triangleright\underline{\sqcup}w)$ задължително в даден момент спира изпълнението си и или приема думата w в случай, че $w \in L$, или отхвърля w в случай, че $w \notin L$.

Пример. Нека с Y и N означим машините на Тюринг, чиято единствена работа да преминат съответно в заключително състояние y и заключително състояние n. Езикът $L = \{0^n1^n2^n \mid n \geq 0\}$ се разрешава от следния разпознавател

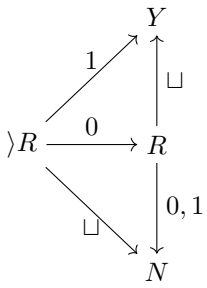


За да илюстрираме, как работи разпознавателят нека разгледаме думата 001122.

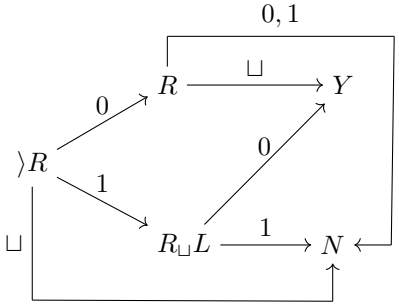
$$\begin{aligned}
\triangleright\underline{\sqcup}001122 &\vdash \triangleright\sqcup\underline{0}01122 \vdash \triangleright\sqcup\underline{d}01122 \vdash \triangleright\sqcup d\underline{0}1122 \vdash \triangleright\sqcup d0\underline{1}122 \vdash \triangleright\sqcup d0\underline{d}122\\
&\vdash \triangleright\sqcup d0d\underline{1}22 \vdash \triangleright\sqcup d0d1\underline{2}2 \vdash \triangleright\sqcup d0d1\underline{d}2 \vdash \triangleright\sqcup d0d1\underline{d}2 \vdash \triangleright\sqcup d0\underline{d}1d2\\
&\vdash \triangleright\sqcup d\underline{0}d1d2 \vdash \triangleright\sqcup\underline{d}0d1d2 \vdash \triangleright\underline{\sqcup}d0d1d2\\
&\vdash \triangleright\sqcup\underline{d}0d1d2 \vdash \triangleright\sqcup d\underline{0}d1d2 \vdash \triangleright\sqcup d\underline{d}d1d2 \vdash \triangleright\sqcup dd\underline{d}1d2 \vdash \triangleright\sqcup ddd\underline{1}d2\\
&\vdash \triangleright\sqcup dddd\underline{d}2 \vdash \triangleright\sqcup ddddd\underline{2} \vdash \triangleright\sqcup dddddd\underline{d} \vdash \triangleright\sqcup ddddd\underline{d}d\\
&\vdash \triangleright\sqcup dddddd \vdash \triangleright\sqcup ddddddd \vdash \triangleright\sqcup dddddd \vdash \triangleright\underline{\sqcup}dddddd\\
&\vdash \triangleright\sqcup\underline{d}dddddd \vdash \triangleright\sqcup d\underline{d}ddddd \vdash \triangleright\sqcup dd\underline{d}dddd \vdash \triangleright\sqcup ddd\underline{d}ddd \vdash \triangleright\sqcup dddd\underline{d}dd\\
&\vdash \triangleright\sqcup ddddd\underline{d} \vdash \triangleright\sqcup dddddd\underline{\sqcup} \vdash y
\end{aligned}$$

За всяко естествено число n с $B(n)$ ще означаваме двоичния запис на n. Да отбележим, че $B(n)$ е непразна дума над азбуката $\{0, 1\}^*$. Ще казваме, че множеството $A \subseteq \mathbb{N}$ е *изчислимо*, ако езикът $\{B(n) \mid n \in A\}$ е разрешим.

Пример. Множеството $\mathbb{N}$ е изчислимо, като то се разпознава от машината $\mathcal{M}_{\mathbb{N}}$



Множеството от четните числа също е разрешимо, като то се разпознава от машината



Теорема 8.1. Разрешимите езици са затворени относно операциите обединение, сечение и допълнение.

Доказателство. Нека за всяка машина $\mathcal{M} = (K, \Gamma, s, H)$, която не използва символа $\#$, с $\mathcal{M}^{\#}$ означим машината $\mathcal{M}^{\#} = (K', \Gamma \cup \{\#\}\delta', H\}$, където

$$K' = K \cup \{p'_q \mid q \in K\} \cup \{p''_q \mid q \in K\}$$

и

$$\begin{aligned}
\delta'(q,x) &= \delta(q,a) \text{ за } q \in K \text{ и } x \neq \# \\
\delta'(q,\#) &= (p'_q, \sqcup) \text{ за } q \in K \\
\delta'(p'_q, \sqcup) &= (p''_q, \rightarrow) \\
\delta'(p'_q, x) &= (p'_q, x) \text{ за } x \neq \sqcup \\
\delta'(p''_q, \sqcup) &= (p''_q, \#) \\
\delta'(p''_q, \#) &= (q, \leftarrow) \\
\delta'(p''_q, x) &= (p''_q, x) \text{ за } x \neq \sqcup, \#
\end{aligned}$$

Ролята на състоянията p'_q и p''_q е в случай, че машината достигне символа $\#$ в състояние $q \in K$, то тя да премести $\#$ една клетка надясно и да върне машината обратно към клетката, в която е бил $\#$ и да бъде отново в състояние q. По точно, за всяко $q \in K$ имаме

$$(q, \sigma\underline{\#}) \vdash_{\mathcal{M}^{\#}} (p'_q, \sigma\underline{\sqcup}) \vdash_{\mathcal{M}^{\#}} (p''_q, \sigma \sqcup \underline{\sqcup}) \vdash_{\mathcal{M}^{\#}} (p''_q, \sigma \sqcup \underline{\#}) \vdash_{\mathcal{M}^{\#}} (q, \sigma\underline{\sqcup}\#)$$

От друга страна, за всяка конфигурация (q, σ) на $\mathcal{M}$, ако $(q, \sigma) \vdash_{\mathcal{M}} (q', \sigma')$ и $\sigma' \neq \sigma\underline{\sqcup}$, то

$$(q, \sigma\#) \vdash_{\mathcal{M}} (q', \sigma'\#),$$

а ако $\sigma' = \sigma\underline{\sqcup}$, то

$$(q, \sigma\#) \vdash_{\mathcal{M}^{\#}} (q', \sigma\underline{\#}),$$

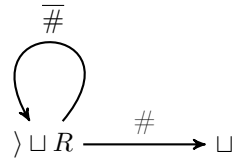
Оттук, за всяко дума w имаме

$$(q, \sigma) \vdash^*_{\mathcal{M}} (q', \sigma') \iff (q, \sigma\#) \vdash^*_{\mathcal{M}^{\#}} (q', \sigma' \sqcup^n \#) \text{ за някое } n \geq 0.$$

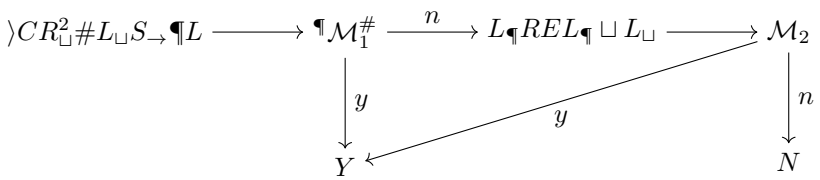
Следователно, ако $\mathcal{M}$ е разпознавател за езика L, то $\mathcal{M}^{\#}$ е разпознавател за $L\#$, като по време на изпълнението на $\mathcal{M}^{\#}$, започващо от начална конфигурация, символът $\#$ отбелязва най-дясната клетка на паметта, използвана от машината.

За всяка машина $\mathcal{M}$ с $\P\mathcal{M}$ означаваме машината, в която ролята на символа $\triangleright$ е подменена от символа $\P$.

Накрая, нека с E означим машината, която изтрива лентата надясно, започвайки от текущата позиция на четящата глава и свършвайки със символа $\#$, т.е.



Нека сега L_1 и L_2 са разрешими езици над Σ, като L_1 се разпознава от $\mathcal{M}_1$, а L_2 — от $\mathcal{M}_2$. Машина, разпознаваща $L_1 \cup L_2$ е следната



За всяка дума $w \in \Sigma^*$ частта $CR^2_{\sqcup}\#L_{\sqcup}S_{\rightarrow}\P L$ трансформира

$$\triangleright\underline{\sqcup}w$$

в

$$\triangleright \sqcup w\P\underline{\sqcup}w\#$$

След това ${}^{\P}\mathcal{M}_1^{\#}$ се изпълнява само върху частта $\P\underline{\sqcup}w\#$, като в случай че завърши с y, то $\mathcal{M}_1$ би разпознало w и тогава завършваме цялото изпълнение с y, или ${}^{\P}\mathcal{M}_1^{\#}$ завършва с n и лентата има вида

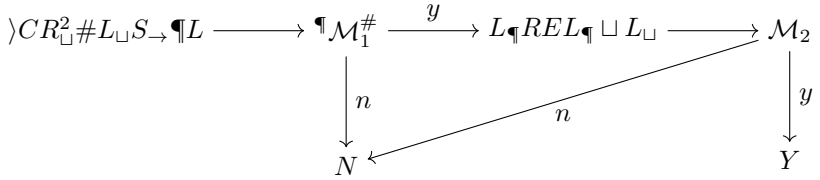
$$\triangleright \sqcup w\P\sigma\#,$$

като четящата глава се намира някъде в σ. Частта $L_{\P}REL_{\P} \sqcup L_{\sqcup}$ премества четящата глава над $\P$, след което изтрива всичко до $\#$ включително, връща се обратно над $\P$ и се позиционира отляво на w, т.е. достигаме до

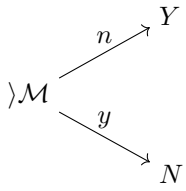
$$\triangleright\underline{\sqcup}w$$

Накрая пускаме $\mathcal{M}_2$, като ако $\mathcal{M}_2$ разпознае w, то завършваме с Y, а ако $\mathcal{M}_2$ не разпознае w, завършваме с N.

Аналогично, машина, разпознаваща $L_1 \cap L_2$ е



Накрая, ако L е език над Σ, който се разпознава от машината $\mathcal{M}$, то $\Sigma^* - L$ се разпознава от



□

Казваме, че машина на Тюринг $\mathcal{M}$ изчислява $f : \underbrace{\Sigma^* \times \cdots \times \Sigma^*}_{n} \to \Sigma^*$, $n \geq 1$ ако за всяко $w_1, \ldots, w_n \in \Sigma^*$ е изпълнено

$$(s, \triangleright\underline{\sqcup}w_1\$\ldots\$w_n) \vdash^*_{\mathcal{M}} (h, \triangleright\underline{\sqcup}f(w_1, \ldots, w_n)),$$

където s е началното състояние, h е стопсъстояние, а $\$ \notin \Sigma$. С други думи, $\mathcal{M}$ изчислява f, ако при всеки избор на думи w_1, …, w_n над Σ, ако разположим върху лентата на Машината думата $w_1\$w_2\$\ldots\$w_n$ (т.е. използваме $ като сепаратор) вдясно от левия край $\triangleright$, оставяйки една празна клетка и разположим четящата глава върху тази празна клетка, то след изпълнението на $\mathcal{M}$ главата отново ще бъде позиционирана върху клетката непосредствено до левия край $\triangleright$, като тази клетка отново ще бъде празна и непосредствено вдясно от нея ще бъде записана думата $f(w_1, w_2, \ldots, w_n)$.

Ще казваме, че една функция е *изчислима*, ако тя се изчислява от някоя машина на Тюринг.

Теорема 8.2. Изчислимите функции са затворени относно операцията суперпозиция.

Доказателство. Нека $k, n \geq 1$ и

$$g_i : (\Sigma^*)^n \to \Sigma^* \text{ за } 1 \leq i \leq k$$

и

$$h : (\Sigma^*)^k \to \Sigma^*$$

са изчислими функции. Нека $\mathcal{M}_{g_i}$ изчислява g_i за $1 \leq i \leq k$ и нека $\mathcal{M}_h$ изчислява h. Нека

$$f : (\Sigma^*)^n \to \Sigma^*$$

действа по правилото

$$f(w_1, \ldots, w_n) = h(g_1(w_1, \ldots, w_n), \ldots, g_n(w_1, \ldots, w_n)).$$

Ще опишем неформално действието на машина $\mathcal{M}$ изчисляваща f. Започваме от конфигурация

$$\triangleright\underline{\sqcup}w_1\$\ldots\$w_n.$$

1. $\mathcal{M}$ преобразува началалнта конфигурация до конфигурация

$$\triangleright\underbrace{\P\underline{\sqcup}w_1\$\ldots\$w_n\P\sqcup w_1\$\ldots\$w_n\P\ldots\P\sqcup w_1\$\ldots\$w_n\P}_{k}\#,$$

където ¶ и # не се използват от машините $\mathcal{M}_{g_i}$ за $1 \leq i \leq k$ и $\mathcal{M}_f$. Това можем да направим прилагайки многократно копиращата машина C и машината, преместваща надясно, $S_{\rightarrow}$, като слагаме символите ¶и # на подходящите места.

2. За $1 \leq i \leq k$ машината $\mathcal{M}'_{g_i}$ е модификация на машината $\mathcal{M}_{g_i}$, която работи точно както $\mathcal{M}_{g_i}$ с изключение на:

   (а) $\mathcal{M}'_{g_i}$ третира ¶ като $\triangleright$, когато го достигне при движение наляво;

   (б) когато $\mathcal{M}'_{g_i}$ достигне ¶ при движение надясно, премества всичко, което пише вдясно отнего с една клетка вдясно;

   (в) кагота четящата глава е в празна клетката непосредствено вляво от ¶ и $\mathcal{M}_{g_i}$ иска да премести главата наляво, то $\mathcal{M}'_{g_i}$ премества ¶ и всичко вдясно от ¶ с една клетка наляво, след което продължава с изпълнението на $\mathcal{M}_{g_i}$.

   Тогава $\mathcal{M}_{g_i}$ преобразува конфигурация $\triangleright\sigma$ до $\triangleright\sigma'$ тогава и само тогава, когато за всяка дума u машината $\mathcal{M}'_{g_i}$ преобразува $\P\sigma\P u\#$ до $\P\sigma'\P u\#$

   Върху конфигурацията, получена в предната точка, първо пускаме $\mathcal{M}_{g_1}$. След приключване на нейната работа получаваме конфигурацията

$$\triangleright\underbrace{\P\underline{\sqcup}g_1(w_1,\ldots,w_n)\P\sqcup w_1\$\ldots\$w_n\P\ldots\P\sqcup w_1\$\ldots\$w_n\P}_{k}\#$$

   Преместваме четящата глава на следващата празна клетка и получаваме

$$\triangleright\underbrace{\P\sqcup g_1(w_1,\ldots,w_n)\P\underline{\sqcup}w_1\$\ldots\$w_n\P\ldots\P\sqcup w_1\$\ldots\$w_n\P}_{k}\#$$

   Пускаме $\mathcal{M}_{g_2}$ и получаваме

$$\triangleright\underbrace{\P\sqcup g_1(w_1,\ldots,w_n)\P\underline{\sqcup}g_2(w_1,\ldots,w_n)\P\ldots\P\sqcup w_1\$\ldots\$w_n\P}_{k}\#$$

   Продължавайки аналогично, получаваме

$$\triangleright\P\sqcup g_1(w_1,\ldots,w_n)\P\sqcup g_2(w_1,\ldots,w_n)\P\ldots\P\underline{\sqcup}g_k(w_1,\ldots,w_n)\P\#$$

3. Изтриваме първото и последно ¶. Премахваме излишните интервали използвайки машина $S_{\leftarrow}$, аналогична на $S_{\rightarrow}$. Вътрешните ¶ заместваме с \$. Премахваме # и преместваме главата непосредствено до $\triangleright$. Така получаваме

$$\triangleright\underline{\sqcup}g_1(w_1,\ldots,w_n)\$g_2(w_1,\ldots,w_n)\$\ldots\$g_k(w_1,\ldots,w_n)$$

4. Пускаме машината $\mathcal{M}_f$ и получаваме

$$\triangleright\underline{\sqcup}h(g_1(w_1,\ldots,w_n),g_2(w_1,\ldots,w_n),\ldots,g_k(w_1,\ldots,w_n))$$

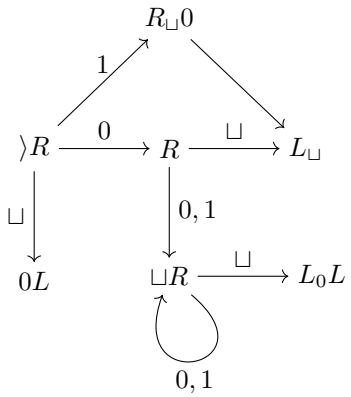
□

Ще казваме, че функцията $f:\mathbb{N}^n\to\mathbb{N}$ е изчислима, ако функцията $f_B:(\{0,1\}^*)^n\to\{0,1\}^*$, за която

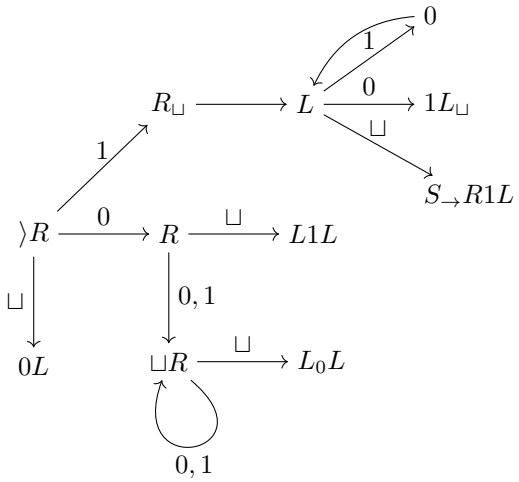
$$f_B(w_1,\ldots,w_n)=\begin{cases}f(m_1,\ldots,m_n), & \text{ако } w_i=B(m_i) \text{ за } 1\leq i\leq n\\ 0, & \text{иначе.}\end{cases}$$

е изчислима.

Пример. Функцията $f:\mathbb{N}\to\mathbb{N}$, действаща по правилото $f(n)=2n$ е изчислима. Машина, изчисляваща f_B, е


$R_{\sqcup}0$
1
$\rangle R$
0
R
$\sqcup$
$L_{\sqcup}$
$\sqcup$
$0, 1$
$0L$
$\sqcup R$
$\sqcup$
L_0L
$0, 1$

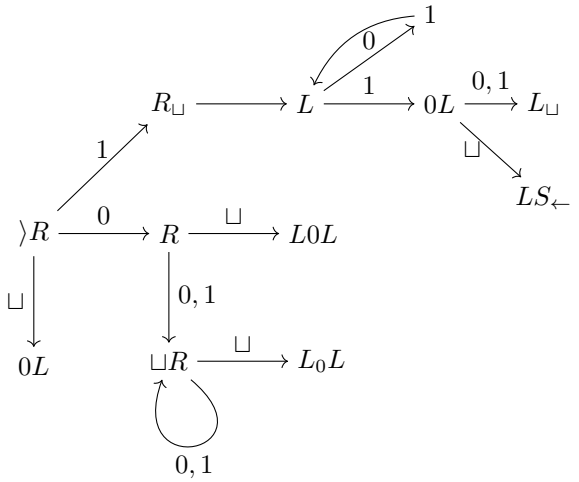
Пример. Функцията $f : \mathbb{N} \to \mathbb{N}$, действаща по правилото $f(n) = n + 1$ е изчислима. Машина, изчисляваща f_B, е $\mathcal{M}_{+1}$



Пример. Функцията $f : \mathbb{N} \to \mathbb{N}$, действаща по правилото

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

е изчислима. Машина, изчисляваща f_B, е $\mathcal{M}_{\dot{-}1}$



8.3 Многолентови машини на Тюринг

Основният проблем при използването на стандартни машини на Тюринг е работата с паметта. Така например алгоритъмът за разпознаване на обединението на два разрешими езика изисква непрекъснато да се следи най-дясната клетка посетена някога от четящата глава, а алгоритъмът за изчисляването на суперпозиция на изчислими функции изисква многократно преместване надясно и наляво на части от записа върху лентата. Някои от тези проблеми бихме могли да преодолеем лесно, ако машината на Тюринг имаше повече от една лента. Така например, ако можехме да използваме две ленти, за да разпознаем дали една дума принадлежи на обединението $L_1 \cup L_2$ на разршимите езици L_1 и L_2, щеше да е достатъчно да копираме думата от първата лента върху втората, след което да пуснем разпознавателя за L_1 да работи върху думата от първата лента, като в случай, че той даде отговор y, то думата е от обединението, а в случай, че даде отговор не, то пускаме разпознавателя за L_2 да работи върху втората лента. В този случай, ако той даде отговор y, то думата е от обединението, а ако отговорът е n, то думата не е от обединението. Аналогично, ако трябва да изчислим $h(g_1(w_1, \ldots, w_n), \ldots, g_k(w_1, \ldots, w_n))$, където h, g_1, $\ldots$, g_n са изичслими, достатъчно е да имаме $k+1$ ленти, като след получаване на входа $w_1\$\ldots\$w_n$ на първата лента, той се копира върху останалите k ленти, като първо пускаме машината, изчисляваща g_1 да работи върху втората лента, след което пускаме машината, изчисляваща g_2 да работи върху третата лента и т.н. След приключване на работата и последната маина, върху лентите с номера $2, \ldots k+1$ ще бъдат разположение съответно $g_1(w_1, \ldots, w_n)$, $\ldots$, $g_k(w_1, \ldots, w_n)$. Прехвърляме тази информация върху първата лента във формата $g_1(w_1, \ldots, w_n)\$\ldots\$g_k(w_1, \ldots, w_n)$, след което пускаме върху този вход да работи машината, изчисляваща h.

За да можем да работим с k ленти едновременно ще ни бъде необходима по една четяща глава за всяка лента, като всички глави трябва да могат да се движат независимо една от друга. Отново ще считаме, че всяка от лентите има ляв край, отбелязан с $\triangleright$. На всяка стъпка решението, какво да бъде предприето от четящите глави, ще зависи от това, което виждат всичките заедно. По-формално, под *k-лентова* машина на Тюринг ($k \geq 1$) ще разбираме наредена петорка $\mathcal{M} = (K, \Gamma, \delta, s, H)$, където K, Γ и H са крайни множества, представляващи съответно стоянията, символите (азбуката) и стопсъстоянията на машината ($H \subseteq K$), а δ е програмата на машината, като

$$\delta : (K - H) \times (\Gamma \cup \{\triangleright\})^k \longrightarrow K \times (\Gamma \cup \{\rightarrow, \leftarrow\})^k,$$

като ако $\delta(q, (x_1, \ldots, x_n)) = (q', (x'_1, \ldots, x'_n))$ и $x_i = \triangleright$ за някое $1 \leq i \leq k$, то $x'_i = \rightarrow$.

Конфигурация на k-лентова машина е наредена двойка $(q, (\sigma_1, \ldots, \sigma_k))$, където σ_i (за $1 \leq i \leq k$) е дума, в която точно един от символите е подчертан, като тя започва с $\triangleright$ и продължава със символи от Γ и в случай, че последния символ е $\sqcup$, то именно той е подчертаният.

Аналогично на случая на еднолентови машини, всяка k-лентова машина $\mathcal{M}$ дефинира релации $\vdash_{\mathcal{M}}$, $\vdash^n_{\mathcal{M}}$ и $\vdash^*_{\mathcal{M}}$, трансформиращи еднозначно k-лентови конфигурации съответно за една, n и неопределен брой стъпки.

Една k-лентова машина е разпознавател, ако стопсъстоянията ѝ са n и y. Казваме, че един разпознавател $\mathcal{M}$ приема (съответно отхвърля) дума w, ако $\mathcal{M}$ трансформира началтата конфигурация

$$(s, (\triangleright\underline{\sqcup}w, \triangleright\underline{\sqcup}, \ldots, \triangleright\underline{\sqcup}))$$

във финална конфигурация $(y, (\sigma_1, \ldots, \sigma_k))$ (съответно $(n, (\sigma_1, \ldots, \sigma_k))$). Един разпознавател разрешава езика $L \subseteq \Sigma^*$, ако приема всички думи от L и отхвърля всички други думи от Σ^*.

Ще казваме, че една k-лентова машина на Тюринг изчислява функцията $f : (\Sigma^*)^n \to \Sigma^*$, ако за всяко $w_1, \ldots, w_n \in \Sigma^*$ машината преобразува началната конфигурация

$$(s, (\triangleright\underline{\sqcup}w_1\$\ldots\$w_n, \triangleright\underline{\sqcup}, \ldots, \triangleright\underline{\sqcup}))$$

където \$ не символ от Σ, във финална конфигурация

$$(h, (\triangleright\underline{\sqcup}f(w_1, \ldots, w_n), \sigma_2, \ldots, \sigma_k)).$$

Теорема 8.3. Работата на всяка многолентова машина та Тюринг може да бъде симулирана от еднолентова машина на Тюринг.

Доказателство. Нека $\mathcal{M} = (K, \Gamma, \delta, s, H)$ е k-лентова машина на Тюринг, $k \geq 2$. Ще симулираме работата на $\mathcal{M}$ чрез еднолентова машина $\mathcal{M}^O$, която използва азбуката

$$\Gamma' = \Gamma \cup \{\underline{x} \mid x \in \Gamma\} \cup \{\P, \underline{\P}, \#\},$$

като символите $\underline{x}$ ще имат ролята на маркери на позицията на четящите глави на $\mathcal{M}$. Тъй като подчертаните символи са част от $\mathcal{M}^O$, за отбелязване на позицията на четящата ѝ глава ще използваме квадратче, с което ще обграждаме съответния символ.

$\mathcal{M}^O$ ще съдържа състоянията на $\mathcal{M}$. Ще казваме, че една конфигурацията на $\mathcal{M}^O$ е *точна*, ако тя има вида

$$(q, \triangleright\P\underline{\ \ x_1\ \ }\P\underline{\ \ x_2\ \ }\dots\P\underline{\ \ x_k\ \ }\boxed{\#}), \tag{8.1}$$

където q е състояние на $\mathcal{M}$, всички символи, които не са явно споменати не са нито ¶, нито са подчертани. Допуска се x_i да съвпада с ¶ вляво от него. Точната конфигурация (8.1) съответства на конфигурацията

$$(q, (\triangleright\underline{\ \ x_1\ \ }, \triangleright\underline{\ \ x_2\ \ }, \dots, \triangleright\underline{\ \ x_k\ \ }))$$

на $\mathcal{M}$.

В началото $\mathcal{M}^O$ трансформира входа

$$\triangleright\boxed{\sqcup}w$$

в

$$\triangleright\P\sqcup w\underbrace{\P\sqcup\P\dots\P\sqcup}_{k-1}\boxed{\#}$$

който съответства на конфигурацията

$$(s, (\triangleright\sqcup w, \triangleright\sqcup, \dots, \triangleright\sqcup))$$

на $\mathcal{M}$. $\mathcal{M}^O$ преминава към изпълнение на състоянието s на $\mathcal{M}$.

Оттук нататък работата на $\mathcal{M}^O$ ще бъде разделена на блокове, като всеки блок ще бъде посветен на изпълнението на дадено състояние на $\mathcal{M}$, като в случай, че блока започне симулация на изпълнение на състояние q на $\mathcal{M}$ от точна конфигурация

$$(q, \triangleright\P\underline{\ \ x_1\ \ }\P\underline{\ \ x_2\ \ }\dots\P\underline{\ \ x_k\ \ }\boxed{\#})$$

то той ще завърши в точна конфигурация

$$(q', \triangleright\P\underline{\ \_\_x'_1\_\_\ }\P\underline{\ \_\_x'_2\_\_\ }\dots\P\underline{\ \_\_x'_k\_\_\ }\boxed{\#})$$

и ще бъде в сила

$$(q, (\triangleright\underline{\ \ x_1\ \ }, \triangleright\underline{\ \ x_2\ \ }, \dots, \triangleright\underline{\ \ x_k\ \ })) \vdash_{\mathcal{M}} (q', \triangleright\underline{\ \_\_x'_1\_\_\ }, \triangleright\underline{\ \_\_x'_2\_\_\ }, \dots, \triangleright\underline{\ \_\_x'_k\_\_\ })$$

Нека $\mathcal{M}^O$ е завършила поредния блок от изпълнение на някое състояние на $\mathcal{M}$ в точна конфигурация

$$\triangleright\P\underline{\ \ x_1\ \ }\P\underline{\ \ x_2\ \ }\dots\P\underline{\ \ x_k\ \ }\boxed{\#}$$

и сега трябва да се изпълни състоянието $q \in K - H$ на $\mathcal{M}$.

Тъй като конфигурацията е точна, то между $\triangleright$ и $\#$ има точно k символа ¶ (някои от които евентуално подчертани). Освен това между $\triangleright$ и $\#$ има точно k подчертани символа, като между всеки два символа ¶ има точно един подчертан символ.

$\mathcal{M}^O$ премества главата вляво върху символа $\triangleright$

$$\boxed{\triangleright}\P\underline{\ \ x_1\ \ }\P\underline{\ \ x_2\ \ }\dots\P\underline{\ \ x_k\ \ }\#$$

като по пътя си запаметява последователно (с помощта на състоянията си) подчертаните символи $x_k, \dots, x_1$ (като ако символът е бил ¶, то вместо него запаметяваме $\triangleright$). Символите $x_1, \dots, x_k$ съответстват на символите, които виждат четящите глави на $\mathcal{M}$ и значи сега $\mathcal{M}$ трябва да изпълни състоянието q върху $(x_1, \dots, x_k)$. Нека

$$\delta(q, (x_1, \dots, x_k)) = (q', (y_1, \dots, y_k)).$$

Сега $\mathcal{M}^O$ премества четящата глава надясно, докато не достигне до първия подчертан символ. Това е символът $\underline{x_1}$.

$$\triangleright\P\underline{\ \ \boxed{x_1}\ \ }\P\underline{\ \ x_2\ \ }\dots\P\underline{\ \ x_k\ \ }\#$$

В този момент $\mathcal{M}^O$ изпълнява y_1 по следната схема:

1. Ако $y_1 \in \Gamma$ (т.е. y_1 не е $\rightarrow$ или $\leftarrow$), то $\mathcal{M}^O$ замества $\underline{x_1}$ с $\underline{y_1}$ и оставя четящата глава върху него.

$$\triangleright\P\underline{\ \ \boxed{y_1}\ \ }\P\underline{\ \ x_2\ \ }\dots\P\underline{\ \ x_k\ \ }\#$$

2. Ако y_1 е $\rightarrow$, то $\mathcal{M}^O$ проверява, дали следващият символ, е ¶ (или $\underline{¶}$). Ако отговорът е не, то този символ е $z_1 \in \Gamma$

$$\triangleright ¶ \_\_ \boxed{\underline{x_1}} z_1 \_\_ ¶ \_\_ \underline{x_2} \_\_ \ldots ¶ \_\_ \underline{x_k} \_\_ \#$$

Тогава $\mathcal{M}^O$ замества $\underline{x}_1$ с x_1, z_1 с $\underline{z}_1$ и оставя главата върху $\underline{z}_1$.

$$\triangleright ¶ \_\_ x_1 \boxed{\underline{z_1}} \_\_ ¶ \_\_ \underline{x_2} \_\_ \ldots ¶ \_\_ \underline{x_k} \_\_ \#$$

Ако отговорът е да, т.е. имаме

$$\triangleright ¶ \_\_ \boxed{\underline{x_1}} ¶ \_\_ \underline{x_2} \_\_ \ldots ¶ \_\_ \underline{x_k} \_\_ \#$$

то $\mathcal{M}^O$ премества цялата информация, записана вдясно от $\underline{x}_1$ с една клетка надясно (т.е. изпълнява S_L)

$$\triangleright ¶ \_\_ \boxed{\underline{x_1}} \sqcup ¶ \_\_ \underline{x_2} \_\_ \ldots ¶ \_\_ \underline{x_k} \_\_ \#$$

след което замества $\underline{x}_1$ с x_1, а в празната клетка непосредствено вдясно от x_1 записва $\underline{\sqcup}$ и оставя четящата глава там

$$\triangleright ¶ \_\_ x_1 \boxed{\underline{\sqcup}} ¶ \_\_ \underline{x_2} \_\_ \ldots ¶ \_\_ \underline{x_k} \_\_ \#$$

3. Ако y_1 е $\leftarrow$, x_1 е $\sqcup$, z_1 е символът непосрествено вляво от x_1, а символът непосредствено вдясно от x_1 е ¶

$$\triangleright ¶ \_\_ z_1 \boxed{\underline{\sqcup}} ¶ \_\_ \underline{x_2} \_\_ \ldots ¶ \_\_ \underline{x_k} \_\_ \#$$

то $\mathcal{M}^O$ то $\mathcal{M}^O$ изтрива $\underline{x}_1$

$$\triangleright ¶ \_\_ z_1 \boxed{\sqcup} ¶ \_\_ \underline{x_2} \_\_ \ldots ¶ \_\_ \underline{x_k} \_\_ \#$$

премества една клетка наляво цялата информация, записана вдясно от x_1

$$\triangleright ¶ \_\_ \boxed{z_1} ¶ \_\_ \underline{x_2} \_\_ \ldots ¶ \_\_ \underline{x_k} \_\_ \#$$

след което замества символа z_1 с $\underline{z}_1$ и оставя четящата глава върху него

$$\triangleright ¶ \_\_ \boxed{\underline{z_1}} ¶ \_\_ \underline{x_2} \_\_ \ldots ¶ \_\_ \underline{x_k} \_\_ \#$$

Ако отговорът е не, то $\mathcal{M}^O$ замества $\underline{x}_1$ с x_1 и z_1 с $\underline{z}_1$ и оставя четящата глава там.

След като изпълни y_1, $\mathcal{M}^O$ премества четящата глава вдясно, докато не открие следващия подчертан символ, който е точно x_2. След като открие x_2, $\mathcal{M}^O$ изпълнява y_2 по същата схема, по която изпълнява y_1, след което мести четящата глава надясно, докато не открие следващия подчертан символ (който е x_3) и т.н. След като изпълни y_k, $\mathcal{M}^O$ премества четящата глава (надясно) върху # и преминава към изпълнение на състоянието q' на $\mathcal{M}^O$.

Ако $\mathcal{M}$ е разпознавател (т.е. $H = \{y, n\}$), когато $\mathcal{M}^O$ достигне до стопсъстояние на $\mathcal{M}$, то тя спира изпълнението в това състояние. В противен случай, когато достигне до стопсътояние $\mathcal{M}^O$ придвижва четящата глава (наляво, броейки с помощтта на състоянията) до първото ¶и проверява дали символа непосредствено вдясно от него е $\underline{\sqcup}$. Ако не, $\mathcal{M}^0$ прекратява изпълнението. Ако да, то $\mathcal{M}^0$ премества четящата глава надясно до следващото ¶, изтрива всичко от това ¶ до #, включително, след което се връща наляво до ¶, изтрива го, премества се наляво, докато не намери $\underline{x}_1$, замества го x_1, продължава наляво до първото ¶, изтрива го, премества информацията вдясно една клетка вляво и преминава във стопсъстояние.

□

Пример. Функцията $f : \mathbb{N}^2 \rightarrow \mathbb{N}$, действаща по правилото $f(m, n) = m + n$ е изчислима. Наистина можем да изчислим f_B чрез трилентова машина на Тюринг, действаща по следната схема. Да напомним, че машината тррябва да работи правилно при начална конфигурация

$$\triangleright \underline{\sqcup} w_1 \$ w_2$$
$$\triangleright \underline{\sqcup}$$
$$\triangleright \underline{\sqcup}$$

където $w_1, w_2 \in \{0, 1\}^*$, като в случай, че една от тях не е двоичен запис на естествено число, машината трябва да върне 0:

1. Преместваме главата на втора лента една клетка вдясно. Преместваме четящата глава на първата лента надясно, докато не достигнем до \$, като същевременно прочетеното се изтрива от първата лента и се копира върху втората лента. Връщаме главата на втората лента до празната клетка вдясно от $\triangleright$. Така достигаме до конфигурация

$$\begin{aligned}&\triangleright \sqcup \underbrace{\underline{\sqcup} \ldots \sqcup}_{|w_1|} \underline{\$} w_2\\ &\triangleright \underline{\sqcup} w_1\\ &\triangleright \underline{\sqcup}\end{aligned}$$

2. Изтриваме \$ от първата лента. Преместваме четящата глава на първата лента надясно, докато не достигнем до $\sqcup$, като същевременно прочетеното се изтрива от първата лента и се копира върху третата лента. Връщаме главите на първата и третата лента до празната клетка вдясно от $\triangleright$. Така достигаме до конфигурация

$$\begin{aligned}&\triangleright \underline{\sqcup}\\ &\triangleright \underline{\sqcup} w_1\\ &\triangleright \underline{\sqcup} w_2\end{aligned}$$

3. Проверяваме, дали w_1 е двоичен запис на естествено число, т.е. дали w_1 е 0 или е дума, започваща с 1. Ако не е, записваме 0 във втората клетка отдясно на $\triangleright$ в първа лента, оставяме главата вляво и спираме изпълнението.

4. Проверяваме, дали w_2 е двоичен запис на естествено число. Ако не е, записваме 0 във втората клетка отдясно на $\triangleright$ в първа лента, оставяме главата вляво и спираме изпълнението.

5. Ако и двете проверки са успешни, т.е. $w_1 = B(m)$ и $w_2 = B(n)$ изпълняваме следния цикъл:

   (а) Проверяваме дали думата на трета лента е 0. Ако да, то спираме цикъла и преминаваме към следващата стъпка от алгоритъма.

   (б) Ако думата на трета лента не е 0, то върху трета лента изпълняваме $\mathcal{M}_{\dot{-}1}$, след което върху втора изпълняваме $\mathcal{M}_{+1}$

   Така на всяка итерация от цикъла, ако тя започне в конфигурация

$$\begin{aligned}&\triangleright \underline{\sqcup}\\ &\triangleright \underline{\sqcup} B(m')\\ &\triangleright \underline{\sqcup} B(n')\end{aligned}$$

   където $n' > 0$, то итерацията завършва в

$$\begin{aligned}&\triangleright \underline{\sqcup}\\ &\triangleright \underline{\sqcup} B(m'+1)\\ &\triangleright \underline{\sqcup} B(n'-1)\end{aligned}$$

   Тъй като цикълът започва от

$$\begin{aligned}&\triangleright \underline{\sqcup}\\ &\triangleright \underline{\sqcup} B(m)\\ &\triangleright \underline{\sqcup} B(n)\end{aligned}$$

   то той окончателно завършва при конфигурация

$$\begin{aligned}&\triangleright \underline{\sqcup}\\ &\triangleright \underline{\sqcup} B(m+n)\\ &\triangleright \underline{\sqcup} B(0)\end{aligned}$$

6. Преместваме главата на първа лента една клетка вдясно, преписваме думата от втора лента върху първа лента, връщаме главата на първа лента върху празната клетка до $\triangleright$

$$\begin{aligned}&\triangleright \underline{\sqcup} B(m+n)\\ &\triangleright \underline{\sqcup} B(m+n)\\ &\triangleright \underline{\sqcup} B(0)\end{aligned}$$

   и спираме изпълнението.

Пример. Функцията $f : \mathbb{N}^2 \to \mathbb{N}$, действаща по правилото $f(m, n) = mn$ е изчислима. Наистина можем да изчислим f_B чрез четирилентова машина на Тюринг, действаща по следната схема. Започваме изпълнението, аналогично на машината от предния пример докато не достигнем до конфигурацията

$$\begin{array}{l} \triangleright\underline{\sqcup} \\ \triangleright\underline{\sqcup}B(0) \\ \triangleright\underline{\sqcup}B(m) \\ \triangleright\underline{\sqcup}B(n) \end{array}$$

където $B(m) = w_1$ и $B(n) = w_2$ (в случай, че проверката, дали w_1 и w_2 са двоични записи на естествени числа, даде отрицателен резултат, то машината връща 0 и спира).

Оттук нататък изпълняваме следния цикъл:

1. Проверяваме дали думата на четвърта лента е 0. Ако да, то спираме цикъла и преминаваме към следващата стъпка от алгоритъма.
2. Ако думата на четвърта лента не е 0, то върху четвърта лента изпълняваме $\mathcal{M}_{\dot{-}1}$. Преместваме главата на втора лента до първата празна вдясно от написаната дума, записваме символа \$, след което преписваме отдясно на него думата на трета лента и връщаме главите на втора и трета лента върху празните клетки непосредствено вдясно от $\triangleright$. Накрая, върху втора изпълняваме $\mathcal{M}_+$, която е еднолентов вариант на машината от предния пример.

Така на всяка итерация от цикъла, ако тя започне в конфигурация

$$\begin{array}{l} \triangleright\underline{\sqcup} \\ \triangleright\underline{\sqcup}B(m') \\ \triangleright\underline{\sqcup}B(m) \\ \triangleright\underline{\sqcup}B(n') \end{array}$$

където $n' > 0$, то итерацията минава последователно през

$$\begin{array}{l} \triangleright\,\underline{\sqcup} \\ \triangleright\,\underline{\sqcup}B(m') \\ \triangleright\,\underline{\sqcup}B(m) \\ \triangleright\,\underline{\sqcup}B(n'-1) \end{array} \qquad \begin{array}{l} \triangleright\,\underline{\sqcup} \\ \triangleright\,\sqcup B(m')\underline{\sqcup} \\ \triangleright\,\underline{\sqcup}B(m) \\ \triangleright\,\underline{\sqcup}B(n'-1) \end{array}$$

$$\begin{array}{l} \triangleright\,\underline{\sqcup} \\ \triangleright\,\sqcup B(m')\underline{\$} \\ \triangleright\,\underline{\sqcup}B(m) \\ \triangleright\,\underline{\sqcup}B(n'-1) \end{array} \qquad \begin{array}{l} \triangleright\,\underline{\sqcup} \\ \triangleright\,\underline{\sqcup}B(m')\$B(m) \\ \triangleright\,\underline{\sqcup}B(m) \\ \triangleright\,\underline{\sqcup}B(n'-1) \end{array}$$

и завършва в

$$\begin{array}{l} \triangleright\underline{\sqcup} \\ \triangleright\underline{\sqcup}B(m'+m) \\ \triangleright\underline{\sqcup}B(m) \\ \triangleright\underline{\sqcup}B(n'-1) \end{array}$$

като е изпълнено $(m' + m) + m(n' - 1) = m' + mn$. Така окончателно той завършва в конфигурацията

$$\begin{array}{l} \triangleright\underline{\sqcup} \\ \triangleright\underline{\sqcup}B(mn) \\ \triangleright\underline{\sqcup}B(m) \\ \triangleright\underline{\sqcup}B(0) \end{array}$$

Накрая копираме съдържанието на втора лента върху първа лента

$$\begin{array}{l} \triangleright\underline{\sqcup}B(mn) \\ \triangleright\underline{\sqcup}B(mn) \\ \triangleright\underline{\sqcup}B(m) \\ \triangleright\underline{\sqcup}B(0) \end{array}$$

и спираме изпълнението.

Забележка. Машината от първия пример пресмята $m + n$ приблизително за $m + n$ стъпки, а втора — mn приблизително за mn стъпки. Важна особеност е, използваме за вход на двете машини думите $B(m)$ и $B(n)$, които имат приблизитени дължини $\log_2 m$ и $\log_2 n$. Това означава, че спрямо дължината на входа си двете машини работят експоненциално дълго време. Далеч по-ефективни са алгоритмите за събиране и умножение, изучавани в училище. При тях, алгоритъмът за събиране е линеен спрямо дължината на входната дума, а този за умножение — квадратичен спрямо входа.

Пример. Нека функцията $g : \mathbb{N}^{k+1} \to \mathbb{N}$, $k \geq 1$ е изчислима и за всяко $\overline{x} \in \mathbb{N}^k$ съществува $y \in \mathbb{N}$, такова че $g(\overline{x}, y) = 0$. Нека $f : \mathbb{N}^k \to \mathbb{N}$ е дефинирана, така че

$$\begin{aligned} g(\overline{x}, f(\overline{x})) &= 0 \\ g(\overline{x}, y) &\neq 0 \text{ за } y < f(\overline{x}) \end{aligned}$$

т.е. $f(\overline{x})$ е най-малкото y, за което $g(\overline{x}, y) = 0$. Ще означаваме

$$f(\overline{x}) = \mu y[g(\overline{x}, y) = 0]$$

и ще казваме, че f се получава от g чрез *минимизация*.

Функцията f е изчислима с помощта на следната четирилентова машина на Тюринг. В началото машината трансформира началната конфигурация

$$\begin{aligned} &\triangleright\underline{\sqcup} w_1\$ \ldots \$w_k \\ &\triangleright\underline{\sqcup} \\ &\triangleright\underline{\sqcup} \\ &\triangleright\underline{\sqcup} \end{aligned}$$

в

$$\begin{aligned} &\triangleright\underline{\sqcup} \\ &\triangleright\underline{\sqcup} \\ &\triangleright\underline{\sqcup} w_1\$ \ldots \$w_k \\ &\triangleright\underline{\sqcup} 0 \end{aligned}$$

след което проверява, че w_1, w_2, …, w_k са двоични записи на естествени числа. Ако някоя от думите не е двоичен запис на естествено число, то връщаме 0 и прекратяваме изчислението. В противен случай, конфигурацията има вида

$$\begin{aligned} &\triangleright\underline{\sqcup} \\ &\triangleright\underline{\sqcup} \\ &\triangleright\underline{\sqcup} B(n_1)\$ \ldots \$B(n_k) \\ &\triangleright\underline{\sqcup} 0 \end{aligned}$$

където n_1, …, n_k са естествени числа.

Оттук нататък изпълняваме следния цикъл:

1. Преписваме третата лента върху втората лента, като оставяме главата на втората лента вдясно от думата, а главата на третата лента вдясно от думата.
2. Върху втората лента записваме \$, след което преписваме съдържанието на четвъртата лента непосредствено вдясно от \$ и преместваме четящата глава непосредствено вдясно от $\triangleright$.
3. Пускаме върху втората лента еднолентов вариант на машина изчисляваща g.
4. Ако върху втората лента пише 0, преписваме четвъртата лента върху първата лента и прекратяваме изпълнението на машината. Ако не, то изтриваме съдържанието на втора лента и пускаме върху четвърта лента машината $\mathcal{M}_{+1}$

5. Започваме цикъла отначало.

Така всяка итерация от цикъла започва в конфигурация

$$\begin{array}{l}\triangleright\underline{\sqcup}\\ \triangleright\underline{\sqcup}\\ \triangleright\underline{\sqcup}B(n_1)\$\ldots\$B(n_k)\\ \triangleright\underline{\sqcup}B(n)\end{array}$$

минава последователно през

$$\begin{array}{ll}\triangleright\underline{\sqcup} & \triangleright\underline{\sqcup}\\ \triangleright\underline{\sqcup} & \triangleright\sqcup B(n_1)\$\ldots\$B(n_k)\underline{\sqcup}\\ \triangleright\underline{\sqcup}B(n_1)\$\ldots\$B(n_k) & \triangleright\underline{\sqcup}B(n_1)\$\ldots\$B(n_k)\\ \triangleright\underline{\sqcup}B(n) & \triangleright\underline{\sqcup}B(n)\end{array}$$

$$\begin{array}{ll}\triangleright\underline{\sqcup} & \triangleright\underline{\sqcup}\\ \triangleright\sqcup B(n_1)\$\ldots\$B(n_k)\underline{\$} & \triangleright\underline{\sqcup}B(n_1)\$\ldots\$B(n_k)\$B(n)\\ \triangleright\underline{\sqcup}B(n_1)\$\ldots\$B(n_k) & \triangleright\underline{\sqcup}B(n_1)\$\ldots\$B(n_k)\\ \triangleright\underline{\sqcup}B(n) & \triangleright\underline{\sqcup}B(n)\end{array}$$

$$\begin{array}{l}\triangleright\underline{\sqcup}\\ \triangleright\underline{\sqcup}B(g(n_1,\ldots,n_k,n)\\ \triangleright\underline{\sqcup}B(n_1)\$\ldots\$B(n_k)\\ \triangleright\underline{\sqcup}B(n)\end{array}$$

В случай, че на втора линия пише 0, то $g(n_1,\ldots,n_k,n)=0$ и това е първото n, за което това се случва и за това машината извежда $B(n)$ на първата лента. Ако на втора лента пише нещо различно от 0, то $g(n_1,\ldots,n_k,n)\neq 0$, при което машината трансформира конфигурацията в

$$\begin{array}{l}\triangleright\underline{\sqcup}\\ \triangleright\underline{\sqcup}\\ \triangleright\underline{\sqcup}B(n_1)\$\ldots\$B(n_k)\\ \triangleright\underline{\sqcup}B(n+1)\end{array}$$

и започва цикъла отначало.

Пример. Лесно може да се съобрази, че всяка функция $I_n^k:\mathbb{N}^n\to\mathbb{N}$, където $n\geq 1$ и $1\leq k\leq n$, действаща по правилото

$$I_n^k(x_1,\ldots,x_n)=x_k$$

е изчислима.

Изчислимите функции над естествените числа имат следното алгебрично описание

Теорема 8.4. Класът на изчислимите функции над $\mathbb{N}$ е най-малкият клас, съдържащ функциите събиране, умножение, както и функциите I_n^k за всяко $n\geq 1$ и всяко $1\leq k\leq n$, и затворен относно суперпозиция и минимизация (както е дефинирана в предпоследния пример).

8.4 Полуразрешими и изчислимономеруеми множества. Частични изчислими функции

Да напомним, че една машина на Тюринг е разпознавател, ако има точно две стопсъстояния, а именно y и n, като при спиране в състояние y машината приема входа, а при спиране в състояние n отхвърля входа. Един разпознавател разрешава език L, ако приема думите от L и отхвърля думите, които не са от L.

Така в случай, че един разпознавател разрешава един език, то той трябва да спира върху всеки вход. От друга страна, машините на Тюринг, дори и когато са разпознаватели, не са длъжни да спират работата си върху всеки вход. Да разгледаме например разпознавателя със състояние $\{s, y, n\}$, символи $\{0, 1\}$ и програма δ, действаща по правилото

$$\delta(s, x) = (s, \rightarrow), \text{ за всеки символ } x.$$

Действието на тази машина е неограничено движение надясно, като входа на машината няма никакво отношение към нейното поведение. Тази машина няма да завърши работа върху нито един вход (в частност не разрешава нито един език).

За да можем да дадем по-точно описание на отношението между езици и разпознаватели въвеждаме понятието полуразрешимост. Казваме, че един език L е полуразрешим, ако съществува разпознавател, който приема думите от L, а думите, които не са от L, или се отхвърлят, или разпознавателят никога не спира върху тях. Отново за да можем да работим не само с езици (т.е. множества от думи), а и с релации от думи, като на релацията

$$R \subseteq (\Sigma^*)^n$$

съпоставяме езика

$$L_R = \{w_1\$ \ldots \$w_n \mid (w_1, \ldots, w_n) \in R)\}$$

(където \$ е специален разделител, който не се съдържа в Σ), като релацията R е полуразрешима тогава и само тогава, когато L_R е полуразрешим. Както и в случай на разрешимост на множества от естествени числа, казваме че едно множество

$$A \subseteq \mathbb{N}^k$$

е полуразрешимо, ако езикът

$$\mathbb{B}(A) = \{B(n_1)\$ \ldots \$B(n_k) \mid (n_1, \ldots, n_k) \in A\}$$

е полуразрешим.

Пример. Да разгледаме множеството $A \subseteq \mathbb{N}^2$, състоящо се от онези двойки (a, d), за които аритметичната прогресия с първи член a и разлика d съдържа поне два последователни члена, които са прости числа. Не е ясно, дали множеството A е разрешимо. За сметка на това, можем да покажем, че A е полуразрешимо, реализирайки следната проста схема: изчисляваме последователно a, $a + d$, $a + 2d$, $\ldots$, като същевременно проверяваме дали всяко едно от тези числа е просто. В случай, че за някое k се окаже, че $a + kd$ и $a + (k + 1)d$ са прости то спираме и разпознаваме двойката (a, d). Тази схема спира работа тогава и само тогава, когато двойката $(a, d) \in A$ и значи A е полуразрешимо.

Теорема 8.5. Езикът $L \subseteq \Sigma^*$ е разрешим тогава и само тогава, кагото L и $\Sigma^* - L$ са полуразрешими.

Доказателство. Ясно е, че ако един език е разрешим, то тогава той е полуразрешим (при това с един и същи разпознавател). Следователно, ако L е разрешим, то имаме, че $\Sigma^* - L$ също е разрешим, откъдето L и $\Sigma^* - L$ са полуразрешими.

Нека сега L и $\Sigma^* - L$ са полуразрешими и нека $\mathcal{M}_+$ и $\mathcal{M}_-$ са два еднолентови разпознавателя, които полуразрешават съответно L и $\Sigma^* - L$. Да разгледаме двулентов разпознавател $\mathcal{M}$, който работи по следната схема:

1. В началото преписва входа (разположен върху първата лента) върху втората лента.

2. Докато една от машините $\mathcal{M}_+$ и $\mathcal{M}_-$ не разпознае входа, пуска $\mathcal{M}_+$ да работи една стъпка върху първата лента, след което $\mathcal{M}_-$ една стъпка върху втората лента, след което отново $\mathcal{M}_+$ една стъпка върху първа лента, след което отново $\mathcal{M}_-$ една стъпка върху втората лента и т.н. В случай, че $\mathcal{M}_+$ приеме входа, то $\mathcal{M}$ спира работа и приема входа, а ако $\mathcal{M}_-$ приеме входа, то $\mathcal{M}$ спира работа и отхвърля входа.

□

Казваме, че изображението f е *номерация* на множеството A, ако е сюрективно изображение от $\mathbb{N}$ върху A. Ще казваме, че езикът L е *изчислимономеруем*, ако съществува изчислима номерация на L. Ще считаме, че празното множество е изчислимономеруемо.

С други думи, един непразен език L е изчисимономеруем, ако съществува изчислима редица

$$a_0, a_1, \ldots, a_n, \ldots,$$

такава, че

$$L = \{a_n \mid n \in \mathbb{N}\}.$$

Да отбележим, че нямаме никакви ограничения върху реда, в който се изреждат елементите на L. Освен това, един и същи елемент на L може да участва повече от един път в редицата. Нещо повече, ако L е краен, то поне един от неговите елементи трябва да участва безброй много пъти в редицата.

Пример. Нека Σ е $n+1$ буквена азбука, $n \geq 0$. Фиксираме едно изреждане

$$a_0, \ldots, a_n$$

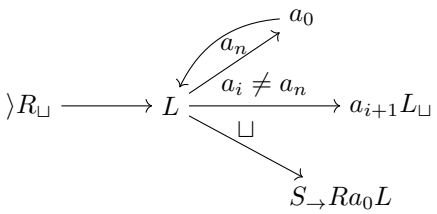
на буквите на Σ. Лексикографската наредбда $<_L$ върху думите на Σ се дефинира по следния начин

$$w <_L w' \quad \stackrel{\text{def}}{\iff} \quad \begin{array}{l} |w| < |w'| \text{ или} \\ |w| = |w'|,\ w = ua_iv,\ w' = ua_jv',\ \text{като } i < j \end{array}$$

Да отбележим, че лексикографската наредба е добра наредба, т.е. има най-малък елемент и няма безкрайно намаляваща редица. В частност, тъй като Σ^* е изброимо, то съществува единствена биекция $\lambda : \mathbb{N} \to \Sigma^*$, такава че

$$m < n \iff \lambda(m) <_L \lambda(n).$$

Ще докажем, че λ е изчислима номерация. За целта първо разглеждаме машината $\mathcal{M}_S$



която при вход $\lambda(n)$ дава изход $\lambda(n+1)$. Сега можем да разгледаме трилентовата машина $\mathcal{M}_\lambda$, изчисляваща λ по следната схема:

1. Проверява дали входа е двоичен запис на естествено число. Ако не е, то изписва 0 върху първата лента и спира работа. В противен случай:

2. Записва 0 в третата лента, като по този начин получаваме конфигурацията

$$\begin{array}{l} \triangleright \underline{\sqcup} B(n) \\ \triangleright \underline{\sqcup} \\ \triangleright \underline{\sqcup} 0 \end{array}$$

и започва изпълнение на следния цикъл:

(а) Проверява дали числата върху първа и трета лента съвпадат. Ако това е така, то изтрива съдържанието на първа лента, копира думата от втора лента върху първа лента и спира работата на машината. Ако не, то:

(б) Изпълнява машината $\mathcal{M}_S$ върху втора лента.

(в) Изпълнява машината $\mathcal{M}_{+1}$ върху трета лента.

(г) Започва цикъла отначало.

По този начин всяка итерация на цикъла започва при конфигурация

$$\triangleright\underline{\sqcup}B(n)$$
$$\triangleright\underline{\sqcup}\lambda(m)$$
$$\triangleright\underline{\sqcup}B(m)$$

където $m \leq n$, като в случай че $n = m$ машината преминава в конфигурация

$$\triangleright\underline{\sqcup}\lambda(n)$$
$$\triangleright\underline{\sqcup}\lambda(n)$$
$$\triangleright\underline{\sqcup}B(n)$$

и спира работа, а в случай че $n < m$ изпълнява пълната итерация и завършва в конфигурация

$$\triangleright\underline{\sqcup}B(n)$$
$$\triangleright\underline{\sqcup}\lambda(m+1)$$
$$\triangleright\underline{\sqcup}B(m+1)$$

Теорема 8.6. Един език е полуразрешим тогава и само тогава, когато той е изчислимономеруем.

Доказателство. Нека $L \subseteq \Sigma^*$ е полуразрешим. Ако $L = \emptyset$, то L е изчислимономеруем. Нека сега L е непразен и нека $\mathcal{M}_L$ е машина, полуразрешаваща L. Ясно е, че множеството

$$S_L = \{(w, n) \mid \mathcal{M}_L \text{ разпознава } w \text{ за по-малко от } n \text{ стъпки}\}$$

е разрешимо. Номерираме L по следната схема: Нека

$$a_0, a_1, \ldots, a_m, \ldots$$

е лексикографското изреждане на думите от Σ^*. Ще използваме брояч C, който в началото е равен на 0. Изпълняваме следния цикъл:

Проверяваме последователно, дали двойките

$$(a_0, C), (a_1, C), \ldots, (a_C, C)$$

принадлежат на S_L. В случай, че (a_i, C) принадлежи, то номерираме a_i. Наркрая увеличаваме брояча C с 1 и започваме цикъла отначало. Така например, ако a_{10} се приема от $\mathcal{M}_L$ за 5 стъпки, a_3 за 7 стъпки, a_{23} за 8 стъпки, и няма друга дума, която се приема за по-малко от 10 стъпки, то след 10 итерации на цикъла бихме имали редицата

$$\underbrace{a_{10}}_{\text{6-та}}, \underbrace{a_{10}}_{\text{7-ма}}, \underbrace{a_3, a_{10}}_{\text{8-а}}, \underbrace{a_3, a_{10}, a_{23}}_{\text{9-а}}, \underbrace{a_3, a_{10}, a_{23}}_{\text{10-а}}$$

За всяка дума a_k на Σ^*, ако $a_k \notin L$, то $(a_k, m) \notin S_L$ за всяко m и значи a_k никога няма да бъде номерирано. От друга страна, ако $a_k \in L$, то $\mathcal{M}_L$ разпознава a_k за някакъв брой стъпки n и следователно a_k ще бъде номерирана на всяка итерация на цикъла след n-тата.

Нека сега L е непразен и изчислимономеруем и нека f е изчислима номерация на L. Можем да полуразрешим L по следната схема: При даден вход w генерираме последователно

$$f(0), f(1), f(2), \ldots$$

В случай, че $f(i) = w$ за някое i, то прекратяваме изчислението и приемаме w. Така (тъй като f е номерация) ще приемем всяка дума от L, а за всяка дума, която не е от L изчислението никога няма да спре.

□

Казваме, че f е *частична функция* от A в B, ако f е функция с $\mathrm{dom}f \subseteq A$ и $\mathrm{range}f \subseteq B$. Казваме, че f е *частична изчислима функция* от $(\Sigma^*)^n$ в Σ'^*, ако f е частична функция от $(\Sigma^*)^n$ в Σ'^* и съществува машина на Тюринг $\mathcal{M}$, така че за всяка n-орка $(w_1, \ldots, w_n)$, ако $(w_1, \ldots, w_n) \in \mathrm{dom}f$, то

$$(s, \triangleright\underline{\sqcup}w_1\$\ldots\$w_n) \vdash^*_{\mathcal{M}} (h, \triangleright\underline{\sqcup}f(w_1, \ldots, w_n) \text{ за някое стопсъстояние } h$$

а ако $(w_1, \ldots, w_n) \notin \mathrm{dom} f$, то

$$(s, \triangleright \underline{\sqcup} w_1 \$ \ldots \$ w_n) \not\vdash^*_{\mathcal{M}} (h, \sigma) \text{ за всяко стопсъстояние } h,$$

т.е. $\mathcal{M}$ никога не спира.

Теорема 8.7. Една функция е частична изчислима тогава и само тогава, когато графиката ѝ е полуразрешимо множество.

Доказателство. Нека първо f е частична изчислима функция от $(\Sigma^*)^n$ в Σ'^* и нека $\mathcal{M}_f$ е машина на Тюринг, изчисляваща f. Можем да полуразрешим

$$G_f = \{(w_1, \ldots, w_n, f(w_1, \ldots, w_n)) \mid (w_1, \ldots, w_n) \in \mathrm{dom} f\}$$

чрез разпознавателя $\mathcal{M}$, действащ по следната схема: При вход

$$\triangleright \underline{\sqcup} w_1 \$ \ldots \$ w_n \$ w$$

преписваме

$$\underline{\sqcup} w_1, \ldots, w_n$$

върху втора лента и пускаме върху нея да работи $\mathcal{M}_f$. В случай, че $\mathcal{M}_f$ спре работа, то втората лента има вида

$$\triangleright \underline{\sqcup} f(w_1, \ldots, w_n)$$

Проверяваме дали $f(w_1, \ldots, w_n) = w$. Ако да, спираме работа и приемаме думата, а ако не, спираме работа и отхвърляме думата.

Нека сега G_f е полуразрешимо множество. Нека g е изчислима номерация на G_f. Можем да изчислим f чрез трилентова машина на Тюринг $\mathcal{M}$, действаща по следната схема: Нека е даден вход

$$\triangleright \underline{\sqcup} w$$

С помощта на третата лента, която служи за брояч, генерираме последвователно

$$g(0), g(1), \ldots g(n), \ldots$$

върху втората лента. Всяко $g(i)$ има вида

$$u^i_1 \$ \ldots \$ u^i_n \$ f(u^i_1, \ldots, u^i_n)$$

След като то бъде изписано върху втората лента проверяваме, дали

$$u^i_1 \$ \ldots \$ u^i_n = w$$

Ако да, то изтриваме първата лента, изписваме върху нея $f(u^i_1, \ldots, u^i_n)$ и спираме работа. Ако не, то генерираме $g(i+1)$ и повтаряме проверката.

□

Теорема 8.8. Един език е полуразрешим тогава и само тогава, котаго той е дефиниционна област на частична изчислима функция.

Доказателство. Нека $L \subseteq \Sigma^*$ е полуразрешим език и нека $\mathcal{M}_L$ полуразрешава L. Тогава L е дефиниционната област на частичната функция f, действща по правилото

$$f(w) = \begin{cases} \varepsilon, & w \in L \\ \neg!, & w \notin L \end{cases}$$

където $\neg!$ означава, че функцията не е дефинирана. Частичната функция може да се изчисли по следния начин: При вход

$$\triangleright \underline{\sqcup} w$$

пускаме $\mathcal{M}_L$ да работи върху този вход. В случай, че $\mathcal{M}_L$ спре и приеме думата, то връщаме ε и спираме, а в случай, че $\mathcal{M}_L$ спре и не приеме думата, то започваме процедура, която никога не завършва (например, на всяка стъпка местим четящата глава надясно).

Обратно, нека L е дефиниционната област на частичната изчислима функция g и нека $\mathcal{M}_g$ изчислява g. Можем да полуразрешим L по следния начин: При вход

$$\triangleright \underline{\sqcup} w$$

пускаме $\mathcal{M}_g$ да работи върху входа. В случай, че тя спре и даде резултат, спираме изпълнението и приемаме думата.

□

8.5 Код на машина на Тюринг. Универсална машина

Нека Γ е фиксирана азбука, съдържаща $\sqcup$, 0 и 1, но не и $\triangleright$, $\leftarrow$, $\rightarrow$. Нека

$$a_1, a_2, \ldots, a_m$$

е фиксирано изреждане на символите от

$$\Gamma \cup \{\triangleright, \leftarrow, \rightarrow\},$$

като

$$a_1 = \triangleright,\ a_2 = \rightarrow .$$

Нека $\mathcal{M} = (K, \Gamma, \delta, s, H)$ е машина на Тюринг. Ще казваме, че четворката

$$(q, x, q', y)$$

е преход на $\mathcal{M}$, ако

$$\delta(q, x) = (q', y)$$

Тъй като функцията δ е крайна (има крайна дефиниционна област $(K - H) \times (\Gamma \cup \{\triangleright\})$), то тя може да бъде описана чрез крайния списък от всички преходите на $\mathcal{M}$. Да отбележим, че от този списък можем да възстановим, множеството $K - H$, азбуката Γ, както и онези стопсъстояния h, за които същуствува състояние q, буква x и команда y, такива че

$$\delta(q, x) = (h, y).$$

Без ограничения можем да считаме, че H се състои само от такива състояния. Освен това можем да считаме, че състоянията на $\mathcal{M}$ са

$$q_1, q_2, \ldots, q_m,$$

като $s = q_1$, а състояния от H са последните няколко в списъка, като в случай, че $\mathcal{M}$ е разпознавател, то $q_{m-1} = y$ и $q_m = n$. С тези уговорки, машината $\mathcal{M}$ се описва изцяло от списъка с преходите ѝ.

Дефинираме кода $\ulcorner\mathcal{M}\urcorner$ на $\mathcal{M}$ по следната схема. Ще кодираме състоянието q_i чрез думата 10^i (за $1 \leq i \leq n$), а символа a_j чрез думата 10^j (за $1 \leq j \leq m$). Използвайки кодовете на състоянията и символите, кодираме прехода (q_i, a_j, q_k, a_l) чрез думата

$$10^i 10^j 10^k 10^l$$

Нека накрая

$$p_1, p_2, \ldots, p_t$$

е фиксирано изреждане на преходите на $\mathcal{M}$ и нека κ_i е кодът на прехода p_i. Тогава код $\ulcorner\mathcal{M}\urcorner$ на $\mathcal{M}$ е естественото число, което има двоичен запис

$$\kappa_1 \kappa_2 \ldots \kappa_t,$$

ако $\mathcal{M}$ не е разпознавател и

$$1\kappa_1 \kappa_2 \ldots \kappa_t,$$

ако $\mathcal{M}$ е разпознавател.

Пример. Нека $\mathcal{M}$ е машина на Тюринг с $K = \{s, h\}$, $\Gamma = \{0, 1, \sqcup\}$, $H = \{h\}$ и δ се състои от преходите

$$(s, \triangleright, s, \rightarrow), (s, \sqcup, h, 0), (s, 0, h, 1), (s, 1, s, \leftarrow).$$

Нека фиксираното изреждане на символите от $\Gamma \cup \{\triangleright, \leftarrow, \rightarrow\}$ е

$$\triangleright, \rightarrow, \leftarrow, \sqcup, 0, 1.$$

Тогава код на $\mathcal{M}$ е числото с двоичен запис

$$\underbrace{10101010010100001001000001010000010010000001010000001010000}_{(s,\triangleright,s,\rightarrow)\quad(s,\sqcup,h,0)\quad(s,0,h,1)\quad(s,1,s,\leftarrow)}$$

Нека отбележим, че $\ulcorner\mathcal{M}\urcorner$ зависи от изреждането на преходите на $\mathcal{M}$. Така всяка машина има $t!$ кода, където t е броят на преходите на $\mathcal{M}$.

Нека още отбележим, че по даден двоичен запис на естествено число, можем да определим, дали то е код на машина на Тюринг. За целта, ако двоичния запис започва с две единици, изтриваме първата и извършваме следните проверки:

1. Разбиваме думата на думи всяка, от които започва с 1, съдържа 4 единици и всяка 1 е последвана от поне една 0. Ако не можем да го направим, то тогава двоичният запис не е код на машина на Тюринг.

2. Намираме най-големия брой нули n, които следват след първата единица на всяка една от получените думи, както и най-големия брой на нули m, които следват след трета единица на всяка една от получените думи.

   Забележка. Числото n е евентуалният брой на състоянията, които не са стопсъстояния, а m е броят на всички състояния.

3. За всяко $1 \leq i \leq n$ и всяко j, такова че $a_j \neq \leftarrow, \rightarrow$, проверяваме, дали точно една от думите започва с 10^i10^j. Ако не, то числото не е код на машина на Тюринг.

   Забележка. Това е условието δ да бъде функция, дефинирана в $(K - H) \times (\Gamma \cup \{\triangleright\})$.

4. За всяко $1 \leq i \leq n$ проверяваме, дали думата, започваща с 10^i101 е от вида 10^i1010^k100 за някое k. Ако не, то числото не е код на машина на Тюринг.

   Забележка. Това е условието за всяко състояние $q \in K - H$ да бъде изпълнено $\delta(q, \triangleright) = (q', \rightarrow)$.

5. За всяка дума $10^i10^j10^k10^l$ проверяваме дали $a_l \neq \triangleright$. Ако не, то числото не е код на машина на Тюринг.

   Забележка. Това е условието δ да бъде функция в $K \times (\Gamma \cup \{\rightarrow, \leftarrow\})$.

6. За всяко $n < k < m$ проверяваме дали има дума от вида $10^i1^0j10^k10^l$. Ако не, то числото не е код на Машина на Тюринг.

   Забележка. Това е допълнителното условие всяко стопсъстояние да бъде трета координата на някой преход.

7. Ако числото е започвало с 11, т.е. то е евентуално код на разпознавател, проверяваме дали $m = n + 2$. Ако не, то числото не е код на Машина на Тюринг.

   Забележка. Ако машината е разпознавател, то тя трябва да има точно две стопсъстояния.

Ако всички проверки са успешни то числото е код на машина на Тюринг. Такива числа ще наричаме истински кодове. Ще считаме, че останалите числа са кодове на машина на Тюринг, която мести безкрайно дълго четящата глава надясно. По този начин всяко естествено число е код на машина на Тюринг.

Теорема 8.9. Съществува машина на Тюринг $\mathcal{M}_U^\Gamma$, която при вход

$$B(\ulcorner\mathcal{M}\urcorner)\$w$$

където $\mathcal{M}$ е машина с азбука Γ, симулира действието на $\mathcal{M}$ върху вход w.

Доказателство. Отново ще опишем неформално поведението на $\mathcal{M}_U^\Gamma$. При вход $u\$w$ първо местим думата u върху втора лента, като на първа лента остава $\triangleright\sqcup w$. След това проверяваме дали u е естествено число в двоичен запис. В случай, че не е, то спираме действието на $\mathcal{M}_U^\Gamma$. Ако $u = B(n)$ за някое n проверяваме, дали $B(n)$ е истински код. Ако не е, пускаме четящата глава на първа лента да се движи безкрайно дълго надясно. В противен случай $n = \ulcorner\mathcal{M}\urcorner$ за някоя машина $\mathcal{M}$. Преобразуваме $B(n)$, замествайки всяка част

$$10^i10^j10^k10^l$$

съответстваща на даден преход, с

$$10^i1a_j10^ka_l$$

Накрая записваме

$$101\sqcup$$

на трета лента и започваме симулацията на $\mathcal{M}$ чрез следния цикъл:

1. Нека върху трета лента е изписана думата 10^i1a_j, като главата на първа лента се намира точно върху символ a_j

2. Търсим върху втора лента преход започващ с 10^i1a_j. Ако такъв няма, то това означава, че сме достигнали до стопсъстояние на $\mathcal{M}$. В този случай, ако $\mathcal{M}$ е разпознавател ($B(n)$ започва с 11), то изчисляваме дали това стопсъстояние е първото или второто по големина на кода, като в случай, че то е първото, прекратяваме изчислението на $\mathcal{M}_U^\Gamma$ и приемаме входа, а във вслучай, че е второто, прекратяваме изчислението на $\mathcal{M}_U^\Gamma$ и отхвърляме входа. В случай, че $\mathcal{M}$ не е разпознавател, просто прекратяваме изчислението на $\mathcal{M}_U^\Gamma$.

3. Нека сме намерили преход $10^i1a_j10^k1a_l$ (тъй като $B(n)$ е истински код, то този преход е единствен).

   (а) Ако $a_l \neq \leftarrow, \rightarrow$, то заместваме символа a_i, който се вижда от главата на първа лента, със символа a_l (без да местим главата), и заместваме думата 10^i1a_j с 10^k1a_l.

   (б) Ако $a_l = \rightarrow$, то преместваме четящата глава на първа лента с една клетка надясно. Нека там е записан символа a_s. Заместваме думата 10^i1a_j с 10^k1a_s.

   (в) Ако $a_l = \leftarrow$, то преместваме четящата глава на първа лента с една клетка наляво. Нека там е записан символа a_s. Заместваме думата 10^i1a_j с 10^k1a_s.

4. Повтаряме цикъла отначало.

□

8.6 Стоппроблем

Неформално стоппроблемът може да се изкаже по следния начин: *Вярно ли е, че машината на Тюринг $\mathcal{M}$ спира върху вход w?* Можем да формализираме стоппроблема, използвайки кодовете на машина на Тюринг. По-точно, нека при фиксирана азбука Γ (съдържаща 0, 1 и $\sqcup$ и несъдържаща $\triangleright$, $\rightarrow$ и $\leftarrow$) и непразна $\Sigma \subseteq \Gamma - \{\sqcup\}$ с $\mathcal{H}$ означим множеството

$$\mathcal{H} = \{(n, w) \mid \text{машината } \mathcal{M} \text{ с код } n \text{ спира при вход } w \in \Sigma^*\},$$

т.е. $(n, w) \in \mathcal{H}$ тогава и само тогава, когато за машината $\mathcal{M}$ с код n е в сила

$$(s, \triangleright\underline{\sqcup}) \vdash^*_{\mathcal{M}} (h, \sigma)$$

където s е началното състояние, а h е някое стопсъстояние на $\mathcal{M}$.

Така неформалния проблем се свежда до разрешимостта на множеството $\mathcal{H}$. Ясно, че $\mathcal{H}$ е полуразрешимо чрез универсалната машина $\mathcal{M}^\Gamma_U$. Наистина при вход (n, w) машината $\mathcal{M}^\Gamma_U$ симулира работата на машината $\mathcal{M}$ с код n върху входа w и значи $\mathcal{M}^\Gamma_U$ спира върху (n, w) тогава и само тогава, когато $\mathcal{M}$ спира върху входа w.

Теорема 8.10. Множестово $\mathcal{H}$ не е разрешимо.

Доказателство. Нека фиксираме $a \in \Sigma$. Да допуснем, че $\mathcal{H}$ е разрешимо и нека $\mathcal{M}_\mathcal{H}$ е машина, която го разрешава. Тъй като Γ съдържа 0 и 1, без ограничение можем да считаме, че азбуката на $\mathcal{M}_\mathcal{H}$ е Γ. [1] Да разгледаме множеството

$$\mathcal{H}_1 = \{a^n \mid (n, a^n) \in \mathcal{H}\}.$$

Тъй като $\mathcal{H}$ е разрешимо, то $\mathcal{H}_1$ също е разрешимо чрез машина, използваща Γ (от a^n можем да възстановим двоичен запис на n, четейки думата отляво надясно и прибавяйки 1 при всяко движение надясно) и следователно множеството

$$\mathcal{H}_1^\partial = \{a^n \mid a^n \notin \mathcal{H}_1\}.$$

също е разрешимо посредством машина, използваща азбуката Γ. Оттук $\mathcal{H}_1$ е полуразрешимо чрез машина с код n_0, която спира точно върху думите от множеството. Тогава

$$a^{n_0} \in \mathcal{H}_1^\partial \iff (n_0, a^{n_0}) \in \mathcal{H} \iff a^{n_0} \in \mathcal{H}_1 \iff a^{n_0} \notin \mathcal{H}_1^\partial,$$

което е невъзможно. Следователно $\mathcal{H}$ не е разрешимо.

Следствие 8.11. Не всяко полуразрешимо множество е разрешимо.

[1] Ако една машина $\mathcal{M}$ работи с азбука $\Gamma' = \{a_1, a_2, \ldots, a_n\}$, то тя може да бъде симулирана от машина, която работи с азбука, съдържаща 0 и 1, като всяка буква $a_i \in \Gamma'$ се замества с думата 10^i.

8.7 Машини с неограничени регистри

Машините с неограничени регистри спадат към клас машини с непоследователен достъп до паметта (RAM-памет). Всяка машина с неограничени регистри се състои от памет и програма. Паметта на машината е разпределена в краен брой регистри, всеки от които има адрес, представляващ естествено число, и в който във всеки един момент (стъпка от изпълнението) се съхранява естествено число. В началото на изпълнението всеки регистър съдържа числото 0. Програмата на машината е крайна редица от редове, индексирани с последователни естествени числа (номера на редовете), съдържащи една от следните команди:

$Z(n)$ //Анулирай регистъра с адрес n. Изпълни следващия ред на програмата.

$S(n)$ //Прибави единица към регистъра с адрес n. Изпълни следващия ред на програмата.

$T(n,m)$ //Копирай съдържанието на регистъра с адрес n в регистъра с адрес m. Изпълни следващия ред на програмата.

$J(n,m,l)$ //Ако съдържанието на регистъра с адрес n съвпада със съдържанието на регистъра с адрес m, изпълни ред с номер l от програмата. В противен случай изпълни следващия ред на програмата.

В случай, че машината стигне до ред от програмата без съдържание, то тя спира изпълнението си. В началото на изпълнението машината получава вход наредена k-орка от естествени числа $(n_1, \ldots, n_k)$, като числата n_1, n_2, $\ldots$, n_k се поставят съответно в регистрите с адреси 1, 2, $\ldots$, k, след което машината започва да изпълнява първия ред на програмата. В случай, че в даден момент машината стигне до празен ред (ред без команда), машината спира и изходът ѝ е съдържанието на регистъра с адрес 0.

Така всяка машина с неограничени регистри изчислява частична функция от $\mathbb{N}^k$ в $\mathbb{N}$.

Пример. Машината с неограничени регистри с програма

1. $Z(3)$
2. $Z(4)$
3. $Z(5)$
4. $T(1,0)$
5. $J(2,3,9)$
6. $S(0)$
7. $S(3)$
8. $J(4,5,5)$

изчислява функцията $f_+ : \mathbb{N}^2 \to \mathbb{N}$, действаща по правилото

$$f_+(m,n) = m + n.$$

Всяка машина с неограничени регистри може да се използва като подпрограма на друга машина с неограничени регистри. За целта е достатъчно да преномерираме редовете на програмата, така че да няма повторение в номерата на редовете, и да транслираме с подходяща константа адресите на използваните регистри, така че подпрограмата да не използва регистри, които се използват от основната програма. Така например горната програма може да се използва като подпрограма в друга машина, модифицирайки я чрез две контстанти l и r, като l е транслацията на редовете, а r е транслацията на регистрите, а именно:

$l+1.$ $Z(r+3)$
$l+2.$ $Z(r+4)$
$l+3.$ $Z(r+5)$
$l+4.$ $T(r+1, r+0)$
$l+5.$ $J(r+2, r+3, l+9)$
$l+6.$ $S(r+0)$
$l+7.$ $S(r+3)$
$l+8.$ $J(r+4, r+5, l+5)$

Тази подпрограма при вход n и m, разположен в регистри $r+1$ и $r+2$, връща изход $n+m$ в регистър $r+0$ и предава управлението, пращайки към изпълнението на ред $l+9$. Нека означим тази подпрограма с $N_+(l,r)$.

Пример. Машината с програма

1. $Z(3)$
2. $Z(4)$
3. $Z(5)$
4. $Z(6)$
5. $T(2,7)$
6. $T(6,8)$
7. $J(2,3,11)$
8. $N_+(7,6)$
9. $S(3)$
10. $J(4,5,6)$
11. $T(4,0)$

изчислява функцията $f_\times : \mathbb{N}^2 \to \mathbb{N}$, действаща по правилото

$$f_\times(m,n) = mn.$$

Считаме, че една машина с неограничени регистри изчислява множеството $A \subseteq \mathbb{N}^k$, ако тя изчислява характеристичната функция χ_A на A, където $\chi_A : \mathbb{N}^k \to \{0,1\}$ действа по правилото

$$\chi_A(n_1,\ldots,n_k) = \begin{cases} 1, & (n_1,\ldots,n_k) \in A; \\ 0, & (n_1,\ldots,n_k) \notin A. \end{cases}$$

По този начин, освен функции, машините с неограничени регистри изчисляват и релации между естествени числа.

Пример. Релацията $\leq$ се изчислява от машината

1. $Z(3)$
2. $Z(4)$
3. $Z(5)$
4. $Z(6)$
5. $J(1,3,9)$
6. $J(2,4,10)$
7. $S(3)$
8. $S(4)$
9. $S(0)$

Пример. Функцията $f_{\dot{-}} : \mathbb{N}^2 \to \mathbb{N}$, действаща по правилото $f_{\dot{-}}(m,n) = m \dot{-} n$, където

$$m \dot{-} n = \begin{cases} m-n, & m \geq n; \\ 0, & \text{иначе} \end{cases}$$

се изчислява от машина с неограничени регистри със следното неформално описание:

1. Машината получава входа си в регистри 1 и 2.
2. Проверяваме, дали съдържанието на регистър 2 е по-голяма от съдържанието на регистъра 1. Ако да прекратяваме изпълнението.
3. Използваме регистър 3 за брояч. В началото той е нула.
4. Използваме регистър 4 за съхранение на сумата на регистри 2 и 3. Преписваме съдържанието на регистър 2 в регистър 4.

5. Започваме следния цикъл:
   (а) Проверяваме, дали съдържанието на регистър 4 е равно на съдържанието на регистър 1. Ако да прекратяваме изпълнението на цикъла.
   (б) Увеличаваме съдържанието на регистър 3 с 1.
   (в) Увеличаваме съдържанието на регистър 4 с 1.
   (г) Започваме цикъла отначало.
6. Прехвърляме съдържанието на регистър 3 в регистър 0 и спираме работа.

Пример. Функцията $Div : \mathbb{N}^2 \to \mathbb{N}$, която по дадени (m, n) изчислява цялата част при деленето на m на n (считаме, че цялата част при делене на 0 е 0) се изчислява от машина с неограничени регистри със следното неформално описание:

1. Машината получава входа си в регистри 1 и 2.
2. Проверяваме, дали съдържанието на регистър 2 е 0. Ако да прекратяваме изпълнението.
3. Използваме регистър 3 за брояч. В началото той е нула.
4. Използваме регистър 4 за съхранение на произведението на регистри 2 и 3. В началото той е нула.
5. Започваме следния цикъл:
   (а) Проверяваме, дали съдържанието на регистър 4 е по-голямо от съдържанието на регистър 1. Ако да прекратяваме изпълнението на цикъла.
   (б) Увеличаваме съдържанието на регистър 3 с 1.
   (в) В регистър 4 записваме сумата на съдържанията на регистър 2 и регистър 4.
   (г) Започваме цикъла отначало.
6. Вадим 1 от съдържанието на регистър 3.
7. Прехвърляме съдържанието на регистър 3 в регистър 0 и спираме работа.

Пример. Функцията $Rem : \mathbb{N}^2 \to \mathbb{N}$, която по дадени (m, n) изчислява остатъка при деленето на m на n (считаме, че цялата част при делене на 0 е делимото), защото

$$Rem(m, n) = m \dot{-} nDiv(m, n)$$

Теорема 8.12. Всяка машина с неограничени регистри може да бъде симулирана от машина на Тюринг

Доказателство. Нека е дадена машина с неограничени регистри $\mathcal{N}$, която използва r регистъра. Ще симулираме работата на $\mathcal{N}$ чрез машина на Тюринг $\mathcal{M}$ с r ленти, работеща над азбука $\Gamma = \{1, \sqcup\}$. Всяка лента на $\mathcal{M}$ съответства на един от регистрите на $\mathcal{N}$. При това, съдържание на k-ти регистър равно на n съответства на съдържание $\triangleright \sqcup 1^n$ на k-та лента. Машината $\mathcal{M}$ разполага с главни състояния $Q_s, \ldots, Q_{s+l}$ съответстващи на редовете на програмата на $\mathcal{N}$ (включително първия празен ред след края ѝ), като всяко главно състояние има свои собствени спомагателни състояния. Началното състояние на машината е Q_s, а стопсъстояния са всички главни състояния, съответстващи на празни редове.

Машината $\mathcal{M}$ се държи така, че при изпълнение на всяко главно състояние, четящата глава на всяка лента да се намира върху празната клетка непосредствено до левия ѝ край. Главното състоянието Q_i действа по следния начин (чрез помощните си състояния):

1. Ако i-тият ред на $\mathcal{N}$ е $Z(n)$, то Q_i пуска машина, която изтрива n-тата лента. Предава управлението на състояние Q_{i+1}.
2. Ако i-тият ред на $\mathcal{N}$ е $S(n)$, то Q_i пуска машина, която добавя буквата 1 в десния край на думата, изписана на n-та лента. Предава управлението на състояние Q_{i+1}.

3. Ако i-тият ред на $\mathcal{N}$ е $T(m,n)$, то Q_i пуска машина, която изтрива n-тата лента, след което пуска втора машина, която копира съдържанието на m-та лента върху n-та лента. Предава управлението на състояние Q_{i+1}.

4. Ако i-тият ред на $\mathcal{N}$ е $J(m,n,s)$, то Q_i пуска машина, която проверява дали съдържанието на m-тата лента съвпада с това на n-тата лента. Ако проверката е успешна, то предава управлението на състояние Q_s, а в противен случай — на състояние Q_{i+1}.

□

Теорема 8.13. Работата на всяка машина на Тюринг може да бъде симулирана от машина с неограничени регистри.

Доказателство. Нека $\mathcal{M} = (K, \Gamma, \delta, s, H)$ е еднолентова машина на Тюринг. Нека фиксираме изреждане

$$a_0, a_1, \ldots, a_{p-1}$$

на елементите на $\Gamma \cup \{\triangleright\}$, като $a_0 = \sqcup$. Тъй като машините с неограничени регистри работят с естествени числа, а не с произволна азбука, трябва първо да кажем, как ще кодираме думите над азбуката Γ. Възползваме се от това, че за всяко $p \geq 2$ и всяко естествено n съществува единствено $s \geq 0$ и числа $0 \leq k_0, k_1, \ldots, k_s < p$ такива, че

$$n = k_0 p^s + k_1 p^{s-1} + \ldots + k_{s-1} p + k_s \text{ (p-ичен запис на n)}$$

На буквата a_i съпоставяме числото $\overline{a_i} = i$. Тогава функцията, която на думата $x_1 x_2 \ldots x_t \in (\Gamma \cup \{\triangleright\})^*$ съпоставя числото

$$\overline{x_1 x_2 \ldots x_t} = \overline{x_0} p^t + \overline{x_1} p^{t-1} + \ldots + \overline{x_{t-1}} p + \overline{x_t}$$

е биективна.

Машината с неограничени регистъра $\mathcal{N}$, симулираща $\mathcal{M}$ ще използва регистри 1 и 2 за да съхранява, записаното върху лентата на $\mathcal{M}$, както и позицията на четящата глава, по следната схема: нека върху лентата е записана думата

$$x_1 x_2 \ldots x_{m-1} \underline{x_m} x_{m+1} \ldots x_n$$

където x_i са букви, като $x_1 = \triangleright$, а в случай, че $x_n = \sqcup$, то $n = m+1$. Тогава в регистър 1 ще съхраняваме числото

$$\overline{x_1 x_2 \ldots x_{m-1} x_m},$$

а в регистър 2 числото

$$\overline{x_n x_{n-1} \ldots x_{m+1}}.$$

Да забележим, че

$$\begin{aligned} Div\,(\overline{x_1 x_2 \ldots x_{m-1} x_m}, p) &= \overline{x_1 x_2 \ldots x_{m-1}} \\ Rem\,(\overline{x_1 x_2 \ldots x_{m-1} x_m}, p) &= \overline{x_m} \\ \overline{x_1 x_2 \ldots x_{m-1} x_m}.p &= \overline{x_1 x_2 \ldots x_{m-1} x_m a_0} \end{aligned}$$

Нека състоянията на $\mathcal{M}$ са $q_0, q_1, \ldots, q_l$, където $q_0 = s$ и за някое k е в сила $K - H = \{q_0, q_1, \ldots, q_k\}$. Програмата на $\mathcal{N}$ е разделена на $k+1$ блока, като i-тия блок съответства на състоянието q_i. Всеки блок е разделен на p подблока, които съответстват на p-те символа на Γ. Описваме действието на i-тия блок по следния неформален начин:

1. Записваме остатъкът от деленето на съдържанието на регистър 1 на p (т.е. $\overline{x_m}$ — това което се вижда от четящата глава на $\mathcal{M}$) в регистър 3.

2. Анулираме регистър 4. Ако съдържанието му съвпада с това на регистър 3, предаваме изпълнението на подблок 0. Иначе добавяме 1 към регистър 4. Ако съдържанието на регистри 3 и 4 съвпада, то предаваме управлението на подблок 1. Иначе добавяме 1 към регистър 4 и така нататък (общо p пъти). По този начин, коректно разпознаваме поредността на буквата x_m в списъка

$$a_0, a_1, \ldots, a_{p-1}.$$

3. Подблокът j, отговаря за това да бъде изпълнен прехода $\delta(q_i, a_j)$ на $\mathcal{M}$. Нека $\delta(q_i, a_j) = (q_{i'}, y)$. Възможни са три случая:

- $y = a_{j'}$ за някое $0 \leq j' \leq p-1$.
  (а) Записваме в регистър 1 частното при делене на p на съдържанието на регистър 1 (така регистър 1 съдържа $\overline{x_1 x_2 \dots x_{m-1}}$).
  (б) Записваме в регистър 1 произведението на съдържанието на регистър 1 и p (така регистър 1 съдържа $\overline{x_1 x_2 \dots x_{m-1} a_0}$).
  (в) Записваме в регистър 1 сумата на съдържанието на регистър 1 и $\overline{a_{j'}}$ (така регистър 1 съдържа $\overline{x_1 x_2 \dots x_{m-1} a_{j'}}$).
  (г) Предаваме изпълнението на блок i'.
- $y = \rightarrow$.
  (а) Записваме в регистър 4 остатъкът при деленето на съдържанието на регистър 2 на p (т.е. $\overline{x_{m+1}}$).
  (б) Записваме в регистър 2 частното при делене на p на съдържанието на регистър 2 (така регистър 2 съдържа $\overline{x_n x_{n-1} \dots x_{m+2}}$).[2]
  (в) Записваме в регистър 1 произведението на съдържанието на регистър 1 и p (така регистър 1 съдържа $\overline{x_1 x_2 \dots x_{m-1} x_m a_0}$).
  (г) Записваме в регистър 1 сумата на съдържанието на регистър 1 и регистър 4 (така регистър 1 съдържа $\overline{x_1 x_2 \dots x_m x_{m+1}}$).
  (д) Предаваме изпълнението на блок i'.
- $y = \leftarrow$.
  (а) Записваме в регистър 1 частното при делене на p на съдържанието на регистър 1 (така регистър 1 съдържа $\overline{x_1 x_2 \dots x_{m-1}}$).
  (б) Записваме в регистър 2 произведението на съдържанието на регистър 2 и p (така регистър 2 съдържа $\overline{x_n x_{n-1} \dots x_{m+1} a_0}$).
  (в) Записваме в регистър 2 сумата на съдържанието на регистър 2 и регистър 3 (така регистър 2 съдържа $\overline{x_n x_{n-1} \dots x_{m+1} x_m}$).
  (г) Предаваме изпълнението на блок i'.

□

Тезис на Чърч. Машините на Тюринг са най-общото ефективно алгоритмично средство.

[2] В случай, че $n = m+1$, то при делене на p се получава частно $0 = \overline{a_0}$

ГЛАВА 9

Апендикс

9.1 Добри наредби. Начални отрези

Да отбележим, че пример за добре наредено множество е празното множество. Добрите наредби имат следните забележителни свойства:

Лема 9.1. Всяко добре наредено множесто е линейно наредено.

Доказателство. Нека A е добре наредено и $a, b \in A$. Тогава в множеството $\{a, b\}$ има най-малък елемент и значи $a \leq b$ или $b \leq a$.

□

Лема 9.2. Нека X е собствен начален отрез на добре нареденото множество $(A, \leq)$. Тогава $X = \mathbb{I}(a)$ за някое $a \in A$.

Доказателство. Тъй като X е собствен начален отрез, множеството $A \setminus X$ е непразно и нека a е най-малкият му елемент. Тогава, ако $x < a$ то $x \in X$ и значи $\mathbb{I}(a) \subseteq X$. От друга страна, ако $x \in X$, то $a \not\leq x$, откъдето $x < a$. Следователно $X \subseteq \mathbb{I}(a)$. Оттук $X = \mathbb{I}(a)$.

□

Определение 9.3. Нека $<_A$ и $<_B$ са частични наредби съответно в A и B. Ще казваме, че изображението $f : A_1 \to A_2$ е *монотонно*, ако

$$x <_A y \Rightarrow f(x) <_B f(y)$$

за всяко $x, y \in A$.

Лема 9.4. Ако $<$ е добра наредба в A и $f : A \to A$ е монотонно, то $x \leq f(x)$ за всяко $x \in X$.

Доказателство. Да допуснем, че множеството $X = \{x \in A \mid f(x) < x\}$ е непразно и нека m е най-малкият му елемент. Тогава $f(m) < m$ (тъй като $m \in X$), откъдето $f(f(m)) < f(m)$ съгласно монотоността на f. От друга страна $f(m) \notin X$ и значи $f(m) \leq f(f(m))$, което е противоречие. Така $X = \emptyset$ и следоветлно $x \leq f(x)$ за всяко $x \in A$.

□

Определение 9.5. Нека $<_A$ и $<_B$ са частични наредби съответно в A и B. Казваме, че $f : A \to B$ е изоморфизъм (на наредбите), ако f е биекция и

$$x <_A y \iff f(x) <_B f(y).$$

Ще казваме, че частично наредените множества A и B са изоморфни (спрямо наредбите си) и ще пишем $A \cong B$, ако съществува изоморфизъм (на наредбите) между A и B.

Лема 9.6. Никое добре наредено множество не е изоморфно на свой собствен начален отрез.

Доказателство. Нека $<$ е добра наредба в A и да допуснем, че $A \simeq \mathbb{I}(a)$ за някое $a \in A$. Нека $f : A \to \mathbb{I}(a)$ е изоморфизъм. Тогава f е монотонно и значи $a \leq f(a)$ съгласно предната лема. Но $f(a) \in \mathbb{I}(a)$ и следователно $f(a) < a$, което е противоречие. Следователно нашето допускане е погрешно и значи A не е изоморфно на свой собствен начален отрез.

□

Теорема 9.7. За всеки две добре наредени множества едното от тях е изоморфно на начален отрез на другото.

Доказателство. Нека $<_A$ и $<_B$ са добри наредби съответно в A и B. Да разгледаме релацията $f \subseteq A \times B$, дефинирана чрез

$$xfy \iff \mathbb{I}(x) \simeq \mathbb{I}(y).$$

Първо ще докажем, че f е *функция*. Наистина, нека xfy и xfy', като без ограничение можем да считаме, че $y <_B y'$. От дефиницията на f имаме $\mathbb{I}(y) \cong \mathbb{I}(x) \cong \mathbb{I}(y')$ и следователно $\mathbb{I}(y) \cong \mathbb{I}(y')$. Но $\mathbb{I}(y)$ е начален отрез на $\mathbb{I}(y')$ (тъй като $y <_B y$ и $y = y'$. Аналогично, ако xfy и $x'fy$, то $x = x'$ и следователно f е *инективна*.

Нека сега xfy (т.е. $\mathbb{I}(x) \simeq \mathbb{I}(y)$) и нека $g : \mathbb{I}(x) \to \mathbb{I}(y)$ е изоморфизъм. Тогава за всяко $x' <_A x$ (т.е. $x' \in \mathbb{I}(x)$) е в сила $\mathbb{I}(x') \simeq \mathbb{I}(g(x'))$, т.е. $x' f\, g(y)$. Следователно $\mathrm{dom}(f)$ е *начален отрез* на A. Аналогично $\mathrm{range}(f)$ е *начален отрез* на B.

Оттук нататък ще пишем обичайното $f(x) = y$ вместо xfy. Сега твърдим, че

$$x <_A x' \iff f(x) <_B f(x')$$

за всяко $x, x' \in \mathrm{dom}(f)$. Наистина, нека $g : \mathbb{I}(x_1) \to \mathbb{I}(R(x_1))$ е изоморфизъм. Нека първо $x_1' <_1 x_1$. Тогава $\mathbb{I}(x_1') \cong \mathbb{I}(f(x_1'))$ и следователно $g(x_1') = R(x_1')$. Но $g(x_1') < f(x_1)$ и значи $f(x_1') < f(x_1)$. Симетрично, ако $R(x_1') <_2 R(x_1)$, имаме $\mathbb{I}\left(f^{-1}(R(x_1'))\right) \cong \mathbb{I}(x_1')$, откъдето $x_1' = f^{-1}(R(x_1')) <_1 x_1$.

Тъй като f е инективна, то f е биекция между $\mathrm{dom}(f)$ и $\mathrm{range}(f)$ Така R е изоморфизъм между началните отрези R_1 и R_2. Остава само да покажем, че $R_1 = A_1$ или $R_2 = A_2$. Да допуснем, че $R_1 \neq A_1$ и $R_2 \neq A_2$. Тогава $R_1 = \mathbb{I}(x_1)$ и $R_2 = \mathbb{I}(x_2)$ за някои $x_1 \in A_1$ и $x_2 \in A_2$. Но R е изоморфизъм между R_1 и R_2 и следователно $R(x_1) = x_2$, откъдето $x_1 \in R_1$ ($x_2 \in R_2$), което е невъзможно и следователно нашето допускане е погрешно. Следователно $A_1 = R_1$ или $A_2 = R_2$, с което теоремата е доказана.

□

9.2 Аксиома за избора. Лема на Цорн

Аксиомата за избора (AC): *Нека $\mathcal{A}$ е множество от непразни множества. Тогава съществува функция $f : \mathcal{A} \to \bigcup \mathcal{A}$, такава че $f(A) \in A$ за всяко $A \in A$.*

Еквивалентно аксиомата за избора може да бъде формулирана и по следния начин:

Нека $\mathcal{S}$ е фамилия от непразни множества индексирана по множеството I, т.е. $\mathcal{S} = \{A_i \mid i \in I\}$, като $A_i \neq \emptyset$ за всяко $i \in I$. Тогава съществува функция $f : I \to \bigcup_{i \in I} A_i$, такава че $f(i) \in A_i$ за всяко $i \in I$.

Определение 9.8. Нека $(A, \leq)$ е частично наредено множество. Казваме, че $L \subseteq A$ е верига в A, ако L е линейно наредено спрямо наредбата $\leq$.

Определение 9.9. Нека $(A, \leq)$ е частично наредено множество и нека $X \subseteq A$. Казваме, че $m \in A$ е горна граница за X в A, ако $x \leq m$ за всяко $x \in X$.

Теорема 9.10 (Лема на Цорн)**.** Нека $(A, \leq)$ е частично наредено множество, такова че всяка верига в A има горна граница в A. Тогава в A има максимален елемент.

Доказателство. Да допуснем, че в A няма максимален елемент, т.е. за всяко $a \in A$ съществува $a' \in A$, такова че $a < a'$. Нека с $\mathcal{W}$ означим съвкупността от всички подмножества на A, които са добре наредени относно $\leq$. Тъй като всяка добра наредба е линейна, $\mathcal{W}$ е множество от вериги в A и следователно всеки елемент на $\mathcal{W}$ има горна граница в A. Нещо повече, съгласно нашето допускане в A няма максимален елемент и следователно за всяко $D \in \mathcal{W}$ множеството $M(D) = \{m \in A \mid x < m \text{ за всяко } x \in D\}$ е непразно. Нека g е функция на избора за множеството

$$\{M(D) \mid D \in \mathcal{W}\}.$$

Ще казваме, че $D \in \mathcal{W}$ е g-добро, ако за всяко $x \in D$ в сила е

$$a = g\left(\{x \in D \mid x < a\}\right).$$

Да означим с $\mathcal{W}_g$ множеството от всички g-добри множества. Първо ще докажем, че за всяко $D_1, D_2 \in \mathcal{W}_g$ или D_1 е начален отрез на D_2, или D_2 е начален отрез на D_1. Ако $D_1 = D_2$ няма какво да доказваме. За това нека $D_1 \neq D_2$ и нека означим с H множеството

$$H = \{x \in D_1 \cap D_2 \mid \forall y \leq x (y \in D_1 \iff y \in D_2)\}.$$

Очевидно $H \subseteq D_1 \cap D_2$. Твърдим, че H е начален отрез както на D_1, така и на D_2. Наистина, нека $a \in H$ и нека $x \in D_1$ е такова, че $x \leq a$. Тогава, $x \in D_2$ и значи $x \in D_1 \cap D_2$. От друга страна, ако $y \leq x$, то $y \leq a$ и значи $y \in D_1 \iff y \in D_2$. Така $x \in H$ и следователно H е начален отрез на D_1. Аналогично, H е начален отрез на D_2.

Сега да допуснем, че $H \neq D_1$ и $H \neq D_2$. Тъй като D_1 и D_2 са добри наредби то съществуват $m_1 \in D_1$ и $m_2 \in D_2$, такива че

$$H = \{x \in D_i \mid x < m_i\}, \text{ за } i = 1, 2.$$

Тъй като D_1 и D_2 са g-добри, в сила е $m_1 = g(H) = m_2$. Но тогава $m_1 \in D_1 \cap D_2$, от където следва, че $m_1 \in H$. Това противоречи с избора на m_1 и следователно или $H = D_1$, или $H = D_2$. Така или D_1 е начален отрез на D_2, или D_2 е начален отрез на D_1.

Нека сега с W означим множеството

$$W = \bigcup_{D \in \mathcal{W}_g} D.$$

Твърдим, че W е g-добра. Нека първо да докажем, че W е добре наредено. За целта нека $X \subseteq W$ е непразно. Тогава X съдържа елемент на някое $D \in \mathcal{W}_g$. Тъй като D е добре наредено, в $X \cap D$ има най-малък елемент m. Да разгледаме $x \in X$. Тогава $x \in D'$ за някое $D' \in \mathcal{W}_g$. Тогава или D е начален отрез на D' или D' е начален отрез на D, като и в двата случая имаме, че най-малкият елемент на $D \cap X$ и $D' \cap X$ е един и същ. Оттук $m \leq x$.

Остава да докажем, че W е g-добра. Нека $a \in W$ тогава $a \in D$ за някое $D \in \mathcal{W}_g$. Да разгледаме $x \in W$, такова че $x < a$. Тогава $x \in D'$ за някое $D' \in \mathcal{W}_g$. Тъй като или D е начален отрез на D' или D' е начален отрез на D, в сила е $x \in D$ и следователно $\{x \in W \mid x < a\} = \{x \in D \mid x < a\}$. Оттук и това, че D е g-добра получаваме

$$a = g(\{x \in D \mid x < a\}) = g(\{x \in W \mid x < a\})$$

и следователно W е g-добра. Да разгледаме множеството $W \cup \{g(W)\}$. Очевидно $W \cup \{g(W)\} \in \mathcal{W}_g$ и следователно $g(W) \in W$, което е абсурдно. Следователно в A има максимален елемент.

□