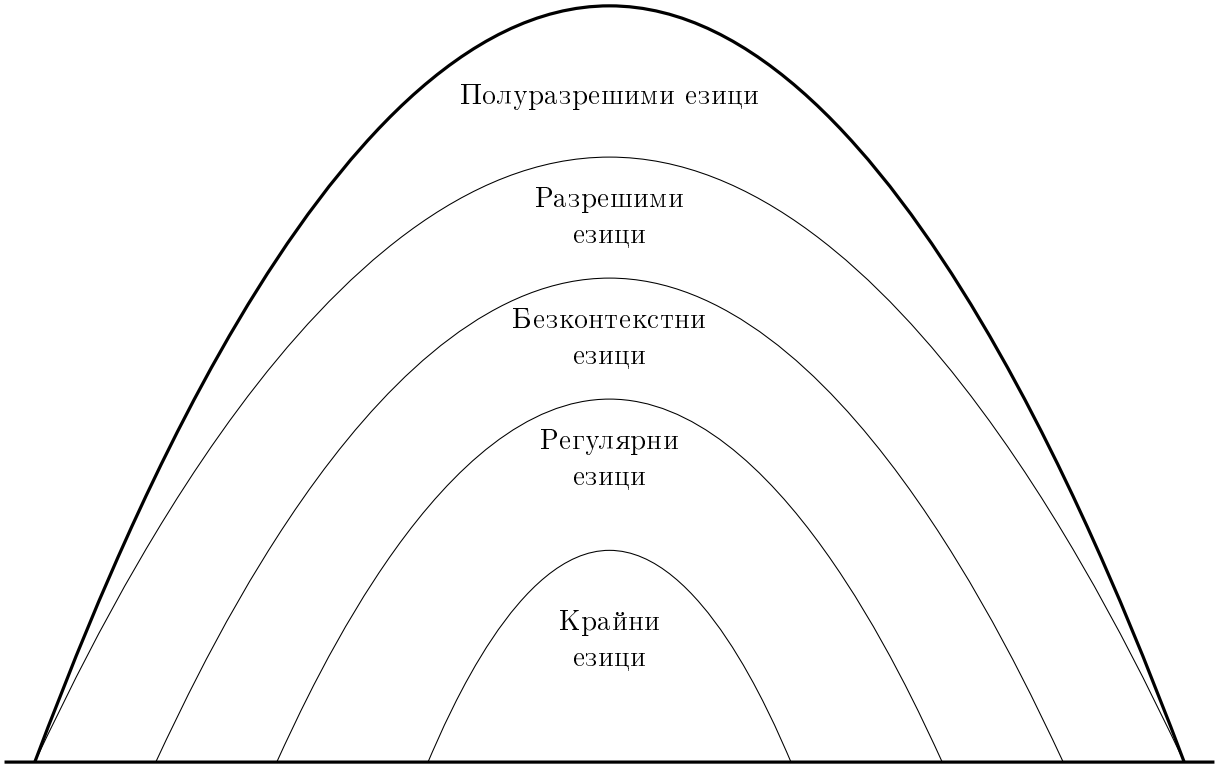
Записки по „Езици, автомати, изчислимост”

Стефан Вътев[1]

12 януари 2017 г.

[1]Това е чернова. Възможни са неточности и грешки, а също и несъответсвия с терминологията въведена на лекции. За забележки и коментари: stefanv@fmi.uni-sofia.bg


Полуразрешими езици
Разрешими
езици
Безконтекстни
езици
Регулярни
езици
Крайни
езици

Съдържание

Глава 1

Увод

1.1 Съждително смятане

На англ. Propositional calculus

Съждителното смятане наподобява аритметичното смятане, като вместо аритметичните операции $+, -, \cdot, /$, имаме съждителни операции като $\neg, \wedge, \vee$. Например, $(p \vee q) \wedge \neg r$ е съждителна формула. Освен това, докато аритметичните променливи приемат стойности числа, то съждителните променливи приемат само стойности **истина (1)** или **неистина (0)**.

Съждителна формула наричаме съвкупността от съждителни променливи $p, q, r, \ldots$, свързани със знаците за логически операции $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$ и скоби, определящи реда на операциите.

Съждителни операции

- Отрицание $\neg$
- Дизюнкция $\vee$
- Конюнкция $\wedge$
- Импликация $\rightarrow$
- Еквивалентност $\leftrightarrow$

Ще използваме таблица за истинност за да определим стойностите на основните съждителни операции при всички възможни набори на стойностите на променливите.

p	q	$\neg p$	$p \vee q$	$p \wedge q$	$p \rightarrow q$	$\neg p \vee q$	$p \leftrightarrow q$	$(\neg p \wedge q) \vee (p \wedge \neg q)$
0	0	1	0	0	1	1	1	1
0	1	1	1	0	1	1	0	0
1	0	0	1	0	0	0	0	0
1	1	0	1	1	1	1	1	1

Съждително верен (валиден) е този логически израз, който има верностна стойност **1** при всички възможни набори на стойностите на съждителните променливи в израза, т.е. стълбът на израза в таблицата за истинност трябва да съдържа само стойности **1**.

Два съждителни израза φ и ψ са **еквивалентни**, което означаваме $\varphi \equiv \psi$, ако са съставени от едни и същи съждителни променливи и двата израза имат едни и същи верностни стойности при всички комбинации от верностни стойности на променливите. С други думи, колоните на двата израза в съответните им таблици за истинност трябва да съвпадат. Така например, от горната таблица се вижда, че $p \to q \equiv \neg p \vee q$ и $p \leftrightarrow q \equiv (\neg p \wedge q) \vee (p \wedge \neg q)$.

Съждителни закони

I) **Комутативен закон**

$$p \vee q \equiv q \vee p$$
$$p \wedge q \equiv q \wedge p$$

II) **Асоциативен закон**

$$(p \vee q) \vee r \equiv p \vee (q \vee r)$$
$$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$$

III) **Дистрибутивен закон**

$$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$$
$$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$$

IV) **Закони на де Морган**

$$\neg(p \wedge q) \equiv (\neg p \vee \neg q)$$
$$\neg(p \vee q) \equiv (\neg p \wedge \neg q)$$

V) **Закон за контрапозицията**

$$p \to q \equiv \neg q \to \neg p$$

VI) **Обобщен закон за контрапозицията**

$$(p \wedge q) \to r \equiv (p \wedge \neg r) \to \neg q$$

VII) **Закон за изключеното трето**

$$p \vee \neg p \equiv \mathbf{1}$$

VIII) **Закон за силогизма (транзитивност)**

$$[(p \to q) \wedge (q \to r)] \to (p \to r) \equiv \mathbf{1}$$

Лесно се проверява с таблиците за истинност, че законите са валидни.

1.2 Предикати и квантори

Квантори

Свойствата или отношенията на елементите в произволно множество се наричат **предикати**. Нека да разгледаме един едноместен предикат $P(\cdot)$.

твърдение	Кога е истина?	Кога е неистина?
$\forall x P(x)$	$P(x)$ е вярно за всяко x	съществува x, за което $P(x)$ **не** е вярно
$\exists x P(x)$	съществува x, за което $P(x)$ е вярно	$P(x)$ **не** е вярно за всяко x

(I) **Квантор за общност** $\forall x$. Записът $(\forall x \in A)P(x)$ означава, че за всеки елемент a в A, твърдението $P(a)$ има стойност истина. Например, $(\forall x \in \mathbb{R})[(x+1)^2 = x^2 + 2x + 1]$.

(II) **Квантор за съществуване** $\exists x$. Записът $(\exists x \in A)P(x)$ означава, че съществува елемент a в A, за който твърдението $P(a)$ има стойност истина. Например, $(\exists x \in \mathbb{C})[x^2 = -1]$, но $(\forall x \in \mathbb{R})[x^2 \neq -1]$.

Закони на предикатното смятане

(I) $\neg\forall x P(x) \leftrightarrow \exists x \neg P(x)$

(II) $\neg\exists x P(x) \leftrightarrow \forall x \neg P(x)$

(III) $\forall x P(x) \leftrightarrow \neg\exists x \neg P(x)$

(IV) $\exists x P(x) \leftrightarrow \neg\forall x \neg P(x)$

(V) $\forall x \forall y P(x) \leftrightarrow \forall y \forall x P(x)$

(VI) $\exists x \exists y P(x,y) \leftrightarrow \exists y \exists x P(x)$

(VII) $\exists x \forall y P(x,y) \rightarrow \forall y \exists x P(x,y)$

Закони на Де Морган за квантори			
твърдение	Еквивалентно твърдение	Кога е истина?	Кога е неистина?
$\neg\exists x P(x)$	$\forall x \neg P(x)$	за всяко x $P(x)$ **не** е вярно	съществува x, за което $P(x)$ е вярно
$\neg\forall x P(x)$	$\exists x \neg P(x)$	съществува x, за което $P(x)$ **не** е вярно	$P(x)$ е вярно за всяко x

Задача 1.1. Да означим с $K(x,y)$ твърдението "x познава y". Изразете като предикатна формула следните твърдения.

1) Всеки познава някого. $\forall x \exists y K(x,y)$

2) Някой познава всеки. $\exists x \forall y K(x,y)$

3) Някой е познаван от всички. $\exists x \forall y K(y,x)$

4) Всеки знае някой, който не го познава. $\forall x \exists y (K(x,y) \wedge \neg K(y,x))$

5) Има такъв, който знае всеки, който го познава. $\exists x \forall y (K(y,x) \rightarrow K(x,y))$

6) Всеки двама познати имат общ познат. $(\forall x,y)(K(x,y)\ \&\ K(y,x) \rightarrow \exists z(K(x,z)\ \&\ K(y,z))$

Пример 1.1. Нека $D \subseteq \mathbb{R}$. Казваме, че $f : D \rightarrow \mathbb{R}$ е *непрекъсната* в точката $x_0 \in D$, ако

$$(\forall \varepsilon > 0)(\exists \delta > 0)(\forall x \in D)[|x_0 - x| < \delta \implies |f(x_0) - f(x)| < \varepsilon].$$

Да видим какво означава f да бъде *прекъсната* в точката $x_0 \in D$: f е прекъсната в x_0 ако f не е непрекъсната в x_0

$$\begin{aligned}
&\neg(\forall \varepsilon > 0)(\exists \delta > 0)(\forall x \in D)[|x_0 - x| < \delta \implies |f(x_0) - f(x)| < \varepsilon] \ \leftrightarrow \\
&(\exists \varepsilon > 0)\neg(\exists \delta > 0)(\forall x \in D)[|x_0 - x| < \delta \implies |f(x_0) - f(x)| < \varepsilon] \ \leftrightarrow \\
&(\exists \varepsilon > 0)(\forall \delta > 0)\neg(\forall x \in D)[|x_0 - x| < \delta \implies |f(x_0) - f(x)| < \varepsilon] \ \leftrightarrow \\
&(\exists \varepsilon > 0)(\forall \delta > 0)(\exists x \in D)\neg[|x_0 - x| < \delta \implies |f(x_0) - f(x)| < \varepsilon] \ \leftrightarrow \\
&(\exists \varepsilon > 0)(\forall \delta > 0)(\exists x \in D)\neg[\neg(|x_0 - x| < \delta) \vee |f(x_0) - f(x)| < \varepsilon] \ \leftrightarrow \\
&(\exists \varepsilon > 0)(\forall \delta > 0)(\exists x \in D)[\neg\neg(|x_0 - x| < \delta) \wedge \neg(|f(x_0) - f(x)| < \varepsilon)] \ \leftrightarrow \\
&(\exists \varepsilon > 0)(\forall \delta > 0)(\exists x \in D)[|x_0 - x| < \delta \ \wedge \ |f(x_0) - f(x)| \geq \varepsilon].
\end{aligned}$$

1.3 Доказателства на твърдения

Допускане на противното

Да приемем, че искаме да докажем, че свойството $P(x)$ е вярно за всяко естествено число. Един начин да направим това е следният:

- Допускаме, че съществува елемент n, за който $\neg P(n)$.
- Използвайки, че $\neg P(n)$ правим извод, от който следва факт, за който знаем, че винаги е лъжа. Това означава, че доказваме следното твърдение
$$\exists x \neg P(x) \rightarrow \mathbf{0}.$$
- Тогава можем да заключим, че $\forall x P(x)$, защото имаме следния извод:

$$\dfrac{\dfrac{\dfrac{\exists x \neg P(x) \to \mathbf{0}}{\mathbf{1} \to \neg\exists x \neg P(x)}}{\neg\exists x \neg P(x)}}{\forall x P(x)}$$

Ще илюстрираме този метод като решим няколко прости задачи.

Задача 1.2. За всяко $a \in \mathbb{Z}$, ако a^2 е четно, то a е четно.

Доказателство. Ние искаме да докажем твърдението P, където:

$$P \equiv (\forall a \in \mathbb{Z})[a^2 \text{ е четно } \to \ a \text{ е четно}].$$

Да допуснем противното, т.е. изпълнено е $\neg P$. Лесно се вижда, че

$$\neg P \ \leftrightarrow \ (\exists a \in \mathbb{Z})[a^2 \text{ е четно } \wedge \ a \text{ не е четно}].$$

$\neg(\forall x)(A(x) \to B(x))$ е еквивалентно на $(\exists x)(A(x) \wedge \neg B(x))$

Да вземем едно такова нечетно a, за което a^2 е четно. Това означава, че $a = 2k + 1$, за някое $k \in \mathbb{Z}$, и

$$a^2 = (2k+1)^2 = 4k^2 + 4k + 1,$$

което очевидно е нечетно число. Но ние допуснахме, че a^2 е четно. Така достигаме до противоречие, следователно нашето допускане е грешно и

$$(\forall a \in \mathbb{Z})[a^2 \text{ е четно } \to \ a \text{ е четно}].$$

□

Задача 1.3. Докажете, $\sqrt{2}$ **не** е рационално число.

Доказателство. Да допуснем, че $\sqrt{2}$ е рационално число. Тогава съществуват $a, b \in \mathbb{Z}$, такива че

$$\sqrt{2} = \frac{a}{b}.$$

Без ограничение, можем да приемем, че a и b са естествени числа, които нямат общи делители, т.е. не можем да съкратим дробта $\frac{a}{b}$. Получаваме, че

$$2b^2 = a^2.$$

Тогава a^2 е четно число и от Задача 1.2, a е четно число. Нека $a = 2k$. Получаваме, че

$$2b^2 = 4k^2,$$

от което следва, че

$$b^2 = 2k^2.$$

Това означава, че b също е четно число, $b = 2n$, за някое $n \in \mathbb{Z}$. Следователно, a и b са четни числа и имат общ делител 2, което е противоречие с нашето допускане, че a и b нямат общи делители. Така достигаме до противоречие. Накрая заключаваме, че $\sqrt{2}$ не е рационално число. □

Индукция върху естествените числа

Доказателството с индукция по $\mathbb{N}$ представлява следната схема:

Да напомним, че естествените числа са $\mathbb{N} = \{0, 1, 2, \dots\}$

$$\frac{P(0) \qquad (\forall x \in \mathbb{N})[P(x) \to P(x+1)]}{(\forall x \in \mathbb{N})P(x)}$$

Това означава, че ако искаме да докажем, че свойството $P(x)$ е вярно за всяко естествено число x, то трябва да докажем първо, че е изпълнено $P(0)$ и след това, за произволно естествено число x, ако $P(x)$ вярно, то също така е вярно $P(x+1)$.

Задача 1.4. Всяко естествено число $n \geq 2$ може да се запише като произведение на прости числа.

Доказателство. Доказателството протича с индукция по $n \geq 2$.

а) За $n = 2$ е ясно.

б) Ако $n+1$ е просто число, то всичко е ясно. Ако $n+1$ е съставно, то

$$n + 1 = n_1 \cdot n_2.$$

Тогава $n_1 = p_1^{n_1} \cdots p_k^{n_k}$ и $n_2 = q_1^{m_1} \cdots q_r^{m_r}$, където $p_1, \dots, p_k$ и $q_1, \dots, q_r$ са прости числа. Тогава е ясно, че $n+1$ също е произведение на прости числа.

□

Задача 1.5. Докажете, че за всяко n,

$$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1.$$

Доказателство. Доказателството протича с индукция по n.

- За $n = 0$, $\sum_{i=0}^{0} 2^i = 1 = 2^1 - 1$.
- Нека твърдението е вярно за n. Ще докажем, че твърдението е вярно за $n+1$.

$$\begin{aligned}
\sum_{i=0}^{n+1} 2^i &= \sum_{i=0}^{n} 2^i + 2^{n+1} \\
&= 2^{n+1} - 1 + 2^{n+1} && \text{(от И.П.)} \\
&= 2.2^{n+1} - 1 \\
&= 2^{(n+1)+1} - 1.
\end{aligned}$$

□

1.4 Множества, релации, функции

Основни операции върху множества

Ще разгледаме няколко операции върху произволни множества A и B.

- **Сечение**
$$A \cap B = \{x \mid x \in A \wedge x \in B\}.$$

- **Обединение**
$$A \cup B = \{x \mid x \in A \vee x \in B\}.$$

- **Разлика**
$$A \setminus B = \{x \mid x \in A \wedge x \notin B\}.$$

- **Степенно множество**
$$\mathscr{P}(A) = \{x \mid x \subseteq A\}.$$

  Примери:

  – $\mathscr{P}(\emptyset) = \{\emptyset\}$.
  – $\mathscr{P}(\{\emptyset\}) = \{\emptyset, \{\emptyset\}\}$.
  – $\mathscr{P}(\{\emptyset, \{\emptyset\}\}) = \{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}, \{\emptyset, \{\emptyset\}\}\}$.
  – $\mathscr{P}(\{1, 2\}) = \{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$.

Нека имаме редица от множества $\{A_1, A_2, \ldots, A_n\}$. Тогава имаме следните операции:

- **Обединение на редица от множества**
$$\bigcup_{i=1}^{n} A_i = \{x \mid \exists i(1 \leq i \leq n \;\&\; x \in A_i)\}.$$

- **Сечение на редица от множества**
$$\bigcap_{i=1}^{n} A_i = \{x \mid \forall i(1 \leq i \leq n \to x \in A_i)\}.$$

Задача 1.6. Проверете верни ли са свойствата:

а) $A \subseteq B \leftrightarrow A \setminus B = \emptyset \leftrightarrow A \cup B = B \leftrightarrow A \cap B = A$;

б) $A \setminus \emptyset = A$, $\emptyset \setminus A = \emptyset$, $A \setminus B = B \setminus A$.

в) $A \cap (B \cup A) = A \cap B$;

г) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ и $A \cap (B \cup C) = (A \cup B) \cap (A \cup C)$;

д) $A \setminus B = A \;\leftrightarrow\; A \cap B = \emptyset$;

е) $A \setminus B = A \setminus (A \cap B)$ и $A \setminus B = A \setminus (A \cup B)$;

ж) $(A \cup B) \setminus C = (A \setminus C) \cup (B \setminus C)$;

з) $C \setminus (A \cup B) = (C \setminus A) \cap (C \setminus B)$ и $C \setminus (A \cap B) = (C \setminus A) \cup (C \setminus B)$ — Закони на Де Морган

и) $C \setminus (\bigcup_{i=1}^{n} A_i) = \bigcap_{i=1}^{n} (C \setminus A_i)$ и $C \setminus (\bigcap_{i=1}^{n} A_i) = \bigcup_{i=1}^{n} (C \setminus A_i)$;

к) $(A \setminus B) \setminus C = (A \setminus C) \setminus (B \setminus C)$ и $A \setminus (B \setminus C) = (A \setminus B) \cup (A \cap C)$;

л) $A \subseteq B \Rightarrow \mathscr{P}(A) \subseteq \mathscr{P}(B)$;

м) $\mathscr{P}(A \cap B) = \mathscr{P}(A) \cap \mathscr{P}(B)$ и $\mathscr{P}(A \cup B) = \mathscr{P}(A) \cup \mathscr{P}(B)$; — $X \subseteq A \cup B \stackrel{?}{\Rightarrow} X \subseteq A \vee X \subseteq B$

За да дадем определение на понятието релация, трябва първо да въведем понятието декартово произведение на множества, което пък от своя страна се основава на понятието наредена двойка.

Наредена двойка

За два елемента a и b въвеждаме опрецията **наредена двойка** $\langle a, b \rangle$. Наредената двойка $\langle a, b \rangle$ има следното характеристичното свойство:

$$a_1 = a_2 \;\wedge\; b_1 = b_2 \quad \leftrightarrow \quad \langle a_1, b_1 \rangle = \langle a_2, b_2 \rangle.$$

Понятието наредена двойка може да се дефинира по много начини, стига да изпълнява харектеристичното свойство. Ето примери как това може да стане:

1) Първото теоретико-множествено определение на понятието наредена двойка е дадено от Норберт Винер: — Norbert Wiener (1914)

$$\langle a, b \rangle \stackrel{\text{деф}}{=} \{\{\{a\}, \emptyset\}, \{\{b\}\}\}.$$

2) Определението на Куратовски се приема за „стандартно” в наши дни: — Kazimierz Kuratowski (1921)

$$\langle a, b \rangle \stackrel{\text{деф}}{=} \{\{a\}, \{a, b\}\}.$$

Задача 1.7. Докажете, че горните дефиниции наистина изпълняват характеристичното свойство за наредени двойки.

Определение 1.1. Сега можем, за всяко естествено число $n \geq 1$, да въведем понятието наредена n-орка $\langle a_1, \ldots, a_n \rangle$: — Пример за индуктивна (рекурсивна) дефиниция

$$\langle a_1 \rangle \stackrel{\text{деф}}{=} a_1,$$
$$\langle a_1, a_2, \ldots, a_n \rangle \stackrel{\text{деф}}{=} \langle a_1, \langle a_2, \ldots, a_n \rangle \rangle$$

Оттук нататък ще считаме, че имаме операцията наредена n-орка, без да се интересуваме от нейната формална дефиниция.

Декартово произведение

На англ. cartesian product

Считаме, че $(A \times B) \times C = A \times (B \times C) = A \times B \times C$

За две множества A и B, определяме тяхното декартово произведение като

$$A \times B = \{\langle a, b\rangle \mid a \in A \ \& \ b \in B\}.$$

За краен брой множества $A_1, A_2, \ldots, A_n$, определяме

$$A_1 \times A_2 \times \cdots \times A_n = \{\langle a_1, a_2, \ldots, a_n\rangle \mid a_1 \in A_1 \ \& \ a_2 \in A_2 \ \& \ \ldots \ \& \ a_n \in A_n\}.$$

Задача 1.8. Проверете, че:

а) $A \times (B \cup C) = (A \times B) \cup (A \times C)$.

б) $(A \cup B) \times C = (A \times C) \cup (B \times C)$.

в) $A \times (B \cap C) = (A \times B) \cap (A \times C)$.

г) $(A \cap B) \times C = (A \times C) \cap (B \times C)$.

д) $A \times (B \setminus C) = (A \times B) \setminus (A \times C)$.

е) $(A \setminus B) \times C = (A \times C) \setminus (B \times C)$.

Основни видове бинарни релации

Подмножествата R от вида $R \subseteq A \times A \times \cdots \times A$ се наричат релации. Релациите от вида $R \subseteq A \times A$ са важен клас, който ще срещаме често. Да разгледаме няколко основни видове релации от този клас:

I) **рефликсивна**, ако

$$(\forall x \in A)[\langle x, x\rangle \in R].$$

Например, релацията $\leq \subseteq \mathbb{N} \times \mathbb{N}$ е рефлексивна, защото

$$(\forall x \in \mathbb{N})[x \leq x].$$

II) **транзитивна**, ако

$$(\forall x, y, z \in A)[\langle x, y\rangle \in R \ \& \ \langle y, z\rangle \in R \to \langle x, z\rangle \in R].$$

Например, релацията $\leq \subseteq \mathbb{N} \times \mathbb{N}$ е транзитивна, защото

$$(\forall x, y, z \in A)[x \leq y \ \& \ y \leq z \ \to \ x \leq z].$$

III) **симетрична**, ако

$$(\forall x, y \in A)[\langle x, y\rangle \in R \to \langle y, x\rangle \in R].$$

Например, релацията $= \subseteq \mathbb{N} \times \mathbb{N}$ е рефлексивна, защото

$$(\forall x, y \in \mathbb{N})[x = y \ \to \ y = x].$$

IV) **антисиметрична**, ако

$$(\forall x, y \in A)[\langle x, y\rangle \in R \ \& \ \langle y, x\rangle \in R \to x = y].$$

Например, релацията $\leq \subseteq \mathbb{N} \times \mathbb{N}$ е антисиметрична, защото

$$(\forall x, y, z \in A)[x \leq y \ \& \ y \leq x \ \to \ x = y].$$

- Една бинарна релация R над множеството A се нарича **релация на еквивалентност**, ако R е рефлексивна, транзитивна и симетрична.
- За всеки елемент $a \in A$, определяме неговия **клас на еквивалентност** относно релацията на еквивалентност R по следния начин:

$$[a]_R \stackrel{\text{деф}}{=} \{b \in A \mid \langle a, b\rangle \in R\}.$$

Забележка. Лесно се съобразява, че за всеки два елемента $a, b \in A$,

$$\langle a, b\rangle \in R \ \leftrightarrow \ [a]_R = [b]_R.$$

Пример 1.2. За всяко естествено число $n \geq 2$, дефинираме релацията R_n като

$$\langle x, y\rangle \in R_n \ \leftrightarrow \ x \equiv y \ (\mathrm{mod}\ n).$$

Ясно е, че R_n са релации на еквивалентност.

Операции върху бинарни релации

I) **Композиция** на две релации $R \subseteq B \times C$ и $P \subseteq A \times B$ е релацията $R \circ P \subseteq A \times C$, определена като:

$$R \circ P \stackrel{\text{деф}}{=} \{\langle a, c\rangle \in A \times C \mid (\exists b \in B)[\langle a, b\rangle \in P \ \& \ \langle b, c\rangle \in R]\}.$$

II) **Обръщане** на релацията $R \subseteq A \times B$ е релацията $R^{-1} \subseteq B \times A$, определена като:

$$R^{-1} \stackrel{\text{деф}}{=} \{\langle x, y\rangle \in B \times A \mid \langle y, x\rangle \in R\}.$$

III) **Рефлексивно затваряне** на релацията $R \subseteq A \times A$ е релацията

$$P \stackrel{\text{деф}}{=} R \cup \{\langle a, a\rangle \mid a \in A\}.$$

Очевидно е, че P е рефлексивна релация, дори ако R не е.

IV) **Итерация** на релацията $R \subseteq A \times A$ дефинираме като за всяко естествено число n, дефинираме релацията R^n по следния начин:

$$R^0 \stackrel{\text{деф}}{=} \{\langle a, a\rangle \mid a \in A\}$$
$$R^{n+1} \stackrel{\text{деф}}{=} R^n \circ R.$$

Лесно се вижда, че $R^1 = R$

V) **Транзитивно затваряне** на $R \subseteq A \times A$ е релацията

$$R^+ \stackrel{\text{деф}}{=} \bigcup_{n \geq 1} R^n.$$

✍ Проверете, че R^+ е транзитивна релация!

За дадена релация R, с $R^\star$ ще означаваме нейното *рефлексивно и транзитивно затваряне*. От дефинициите е ясно, че

$$R^\star = \bigcup_{n \geq 0} R^n.$$

Видове функции

Функцията $f : A \to B$ е:

- **инекция**, ако (или f е **обратима**)

$$(\forall a_1, a_2 \in A)[a_1 \neq a_2 \to f(a_1) \neq f(a_2)],$$

или еквивалентно,

$$(\forall a_1, a_2 \in A)[f(a_1) = f(a_2) \to a_1 = a_2].$$

- **сюрекция**, ако (или f е **върху** B)

$$(\forall b \in B)(\exists a \in A)[f(a) = b].$$

- **биекция**, ако е инекция и сюрекция.

Задача 1.9. Докажете, че $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ е биекция, където

$$f(x, y) = \frac{(x + y)(x + y + 1)}{2} + x.$$

Канторово кодиране. Най-добре се вижда като се нарисува таблица

1.5 Азбуки, думи, езици

Основни понятия

- **Азбука** ще наричаме всяко крайно множество, като обикновено ще я означаваме със Σ. Елементите на азбуката Σ ще наричаме **букви** или символи.

  Често ще използваме буквите a, b, c за да означаваме букви.

- **Дума** над азбуката Σ е произволна крайна редица от елементи на Σ. Например, за $\Sigma = \{a, b\}$, $aababba$ е дума над Σ с дължина 7. С $|\alpha|$ ще означаваме дължината на думата α.

  Обикновено ще означаваме думите с α, β, γ, ω.

- Обърнете внимание, че имаме единствена дума с дължина 0. Тази дума ще означаваме с ε и ще я наричаме **празната дума**, т.е. $|\varepsilon| = 0$.

- С a^n ще означаваме думата съставена от n a-та. Формалната индуктивна дефиниция е следната:

$$a^0 \stackrel{\text{деф}}{=} \varepsilon,$$
$$a^{n+1} \stackrel{\text{деф}}{=} a^n a.$$

- Множеството от всички думи над азбуката Σ ще означаваме със $\Sigma^\star$. Например, за $\Sigma = \{a, b\}$,

$$\Sigma^\star = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \dots\}.$$

  Обърнете внимание, че $\emptyset^\star = \{\varepsilon\}$.

Операции върху думи

- Операцията **конкатенация** взима две думи α и β и образува новата дума $\alpha \cdot \beta$ като слепва двете думи. Например $aba \cdot bb = ababb$. Обърнете внимание, че в общия случай $\alpha \cdot \beta \neq \beta \cdot \alpha$. Можем да дадем формална индуктивна дефиниция на операцията конкатенация по дължината на думата β.

  Често ще пишем $\alpha\beta$ вместо $\alpha \cdot \beta$

  – Ако $|\beta| = 0$, то $\beta = \varepsilon$. Тогава $\alpha \cdot \varepsilon \stackrel{\text{деф}}{=} \alpha$.

  – Ако $|\beta| = n + 1$, то $\beta = \gamma b$, $|\gamma| = n$. Тогава $\alpha \cdot \beta \stackrel{\text{деф}}{=} (\alpha \cdot \gamma)b$.

- Друга често срещана операция върху думи е **обръщането** на дума. Дефинираме думата α^R като обръщането на α по следния начин.

  – Ако $|\alpha| = 0$, то $\alpha = \varepsilon$ и $\alpha^R \stackrel{\text{деф}}{=} \varepsilon$.

  – Ако $|\alpha| = n + 1$, то $\alpha = a\beta$, където $|\beta| = n$. Тогава $\alpha^R \stackrel{\text{деф}}{=} (\beta^R)a$.

  Например, $reverse^R = esrever$.

- Казваме, че думата α е **префикс** на думата β, ако съществува дума γ, такава че $\beta = \alpha \cdot \gamma$. α е **суфикс** на β, ако $\beta = \gamma \cdot \alpha$, за някоя дума γ.

- Нека A и B са множества от думи. Дефинираме конкатенацията на A и B като
$$A \cdot B \stackrel{\text{деф}}{=} \{\alpha \cdot \beta \mid \alpha \in A \ \& \ \beta \in B\}.$$

Обърнете внимание, че $\emptyset \cdot A = A \cdot \emptyset = \emptyset$
Също така, $\{\varepsilon\} \cdot A = A \cdot \{\varepsilon\} = A$

- Сега за едно множество от думи A, дефинираме A^n индуктивно:
$$A^0 \stackrel{\text{деф}}{=} \{\varepsilon\},$$
$$A^{n+1} \stackrel{\text{деф}}{=} A^n \cdot A.$$

Ако $A = \{ab, ba\}$, то $A^0 = \{\varepsilon\}$, $A^1 = A$, $A^2 = \{abab, abba, baba, baab\}$.
Ако $A = \{a, b\}$, то $A^n = \{\alpha \in \{a, b\}^\star \mid |\alpha| = n\}$.

- За едно множеството от думи A, дефинираме:
$$A^\star \stackrel{\text{деф}}{=} \bigcup_{n \geq 0} A^n$$
$$= A^0 \cup A^1 \cup A^2 \cup A^3 \cup \ldots$$
$$A^+ \stackrel{\text{деф}}{=} A \cdot A^\star.$$

Операцията $\star$ е известна като звезда на Клини
Обърнете внимание, че $\emptyset^\star = \{\varepsilon\}$

Задача 1.10. Проверете:

а) операцията конкатенация е *асоциативна*, т.е. за всеки три думи α, β, γ,
$$(\alpha \cdot \beta) \cdot \gamma = \alpha \cdot (\beta \cdot \gamma);$$

б) за множествата от думи A, B и C,
$$(A \cdot B) \cdot C = A \cdot (B \cdot C);$$

в) $\{\varepsilon\}^\star = \varepsilon$;

г) за произволно множество от думи A, $A^\star = A^\star \cdot A^\star$ и $(A^\star)^\star = A^\star$;

д) за произволни букви a и b, $\{a, b\}^\star = \{a\}^\star \cdot (\{b\} \cdot \{a\}^\star)^\star$.

Задача 1.11. Докажете, че за всеки две думи α и β е изпълено:

а) $(\alpha \cdot \beta)^R = \beta^R \cdot \alpha^R$;

б) α е префикс на β точно тогава, когато α^R е суфикс на β^R;

в) $(\alpha^R)^R = \alpha$;

г) $(\alpha^n)^R = (\alpha^R)^n$, за всяко $n \geq 0$.

Задача 1.12. Нека $X, Y, Z \subseteq \Sigma^\star$ със свойството, че $Z = XZ \cup Y$.

С други думи, ако $\varepsilon \notin X$, то $Z = X^\star Y$ е най-малкото решение на уравнението $Z = XZ \cup Y$.

а) Докажете, че за всяко $n \in \mathbb{N}$, $X^n Y \subseteq Z$. Заключете, че $X^\star Y \subseteq Z$.

б) Да предположим, че $\varepsilon \notin X$. Докажете, че за всяка дума $\omega \in Z$ е изпълнено, че $\omega \in X^\star Y$.

Бележки

Повечето книги в тази област започват с уводна глава, в която въвеждат понятията множества, релации и езици.

- Глава 1 от [Ros12].
- Глава 1 от [PL98].
- За описанието на думи и азбуки следваме [Koz97, Глава 2].

Глава 2

Регулярни езици и автомати

2.1 Автоматни езици

Определение 2.1. Краен автомат е петорка $\mathcal{A} = \langle Q, \Sigma, q_{\texttt{start}}, \delta, F \rangle$, където

1) Q е крайно множество от състояния;

2) Σ е азбука;

3) $\delta : Q \times \Sigma \to Q$ е (частична) функция на преходите;

4) $q_{\texttt{start}} \in Q$ е начално състояние;

5) $F \subseteq Q$ е множеството от финални състояния, $F \neq \emptyset$.

Ако функцията на преходите δ е тотална функция, то казваме, че автоматът $\mathcal{A}$ е **тотален**. Това означава, че за всяка двойка $(a, q) \in \Sigma \times Q$, съществува $q' \in Q$, за което $\delta(a, q) = q'$.

Нека имаме една дума $\alpha \in \Sigma^\star$, $\alpha = a_1 a_2 \cdots a_n$. Казваме, че α се **разпознава** от автомата $\mathcal{A}$, ако съществува редица от състояния $q_0, q_1, q_2, \ldots, q_n$, такива че:

- $q_0 = q_{\texttt{start}}$, началното състояние на автомата;
- $\delta(q_i, a_{i+1}) = q_{i+1}$, за всяко $i = 0, \ldots, n-1$;
- $q_n \in F$.

Казваме, че $\mathcal{A}$ **разпознава** езика L, ако $\mathcal{A}$ разпознава точно думите от L, т.е. $L = \{\alpha \in \Sigma^\star \mid \mathcal{A} \text{ разпознава } \alpha\}$. Обикновено означаваме езика, който се разпознава от даден автомат $\mathcal{A}$ с $\mathcal{L}(\mathcal{A})$. В такъв случай ще казваме, че езикът L е **автоматен**.

При дадена (частична) функция на преходите δ, често е удобно да разглеждаме (частичната) функция $\delta^\star : Q \times \Sigma^\star \to Q$, която е дефинирана по следния начин:

- $\delta^\star(q, \varepsilon) = q$, за всяко $q \in Q$;
- $\delta^\star(q, a\beta) = \delta^\star(\delta(q, a), \beta)$, за всяко $q \in Q$, всяко $a \in \Sigma$ и $\beta \in \Sigma^\star$.

Тогава една дума α се *разпознава* от автомата $\mathcal{A}$ точно тогава, когато $\delta^\star(s, \alpha) \in F$. Оттук следва, че

$$\mathcal{L}(\mathcal{A}) = \{\alpha \in \Sigma^\star \mid \delta^\star(s, \alpha) \in F\}.$$

Твърдение 2.1. $(\forall q \in Q)(\forall \alpha, \beta \in \Sigma^\star)[\delta^\star(q, \alpha\beta) = \delta^\star(\delta^\star(q, \alpha), \beta)]$.

Доказателство. Индукция по дължината на α. □

Моментното описание на изчисление с краен автомат представлява двойка от вида $(q, \alpha) \in Q \times \Sigma^\star$, т.е. автоматът се намира в състояние q, а думата, която остава да се прочете е α. Удобно е да въведем бинарната релация $\vdash_{\mathcal{A}}$ над $Q \times \Sigma^\star$, която ще ни казва как моментното описание на автомата $\mathcal{A}$ се променя след изпълнение на една стъпка:

(На англ. *instantaneous description*). В случая въвеждането на това понятие не е напълно необходимо, но по-късно, когато въведем стекови автомати и машини на Тюринг, ще работим с такива моментни описания на изчисления и затова е добре още отначало да свикнем с него

$$(q, x\alpha) \vdash_{\mathcal{A}} (p, \alpha), \text{ ако } \delta(q, x) = p.$$

Рефлексивното и транзитивно затваряне на $\vdash_{\mathcal{A}}$ ще означаваме с $\vdash^\star_{\mathcal{A}}$. За да дадем точна дефиниция на $\vdash^\star_{\mathcal{A}}$, първо ще дефинираме релацията $\vdash^n_{\mathcal{A}}$, която определя работата на автомата $\mathcal{A}$ за n стъпки.

Рефл. и транз. затваряне на една релация е разгледано в Глава 1

- $(q, \alpha) \vdash^0_{\mathcal{A}} (q, \alpha)$, защото за 0 стъпки се случва нищо.
- Нека $\delta(q, x) = q'$ и $(q', \alpha) \vdash^n_{\mathcal{A}} (p, \beta)$. Тогава $(q, x\alpha) \vdash^{n+1}_{\mathcal{A}} (p, \beta)$, защото за $n + 1$ стъпки първо правим една стъпка и отиваме в моментното описание (q', α) и след това правим още n стъпки.

Сега можем да дефинираме $\vdash^\star_{\mathcal{A}}$ като:

$$(q, \alpha) \vdash^\star_{\mathcal{A}} (p, \beta) \overset{\text{деф}}{\leftrightarrow} (\exists n \in \mathbb{N})[(q, \alpha) \vdash^n_{\mathcal{A}} (p, \beta)].$$

Получаваме, че

$$\mathcal{L}(\mathcal{A}) = \{\alpha \in \Sigma^\star \mid (s, \alpha) \vdash^\star_{\mathcal{A}} (p, \varepsilon) \ \&\ p \in F\}.$$

Нашата дефиниция на автомат позволява δ да бъде частична функция, т.е. може да има $q \in Q$ и $a \in \Sigma$, за които $\delta(q, a)$ не е дефинирана. Следващото твърдение ни казва, че ние съвсем спокойно можем да разглеждаме автомати само с тотални функции на преходите δ.

Твърдение 2.2. За всеки краен автомат $\mathcal{A}$, съществува *тотален* краен автомат $\mathcal{A}'$, за който $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.

Доказателство. Нека $\mathcal{A} = \langle Q, \Sigma, q_{\texttt{start}}, \delta, F\rangle$. Дефинираме тоталния автомат

$$\mathcal{A}' = \langle Q \cup \{q_e\}, \Sigma, \delta', s, F\rangle,$$

като за всеки преход (q, a), за който δ не е дефинирана, дефинираме δ' да отива в новото състояние q_e. Ето и цялата дефиниция на новата функция на преходите δ':

- $\delta'(q_e, a) = q_e$, за всяко $a \in \Sigma$;
- За всяко $q \in Q$, $a \in \Sigma$, ако $\delta(q, a) = p$, то $\delta'(q, a) = p$;
- За всяко $q \in Q$, $a \in \Sigma$, ако $\delta(q, a)$ не е дефинирано, то $\delta'(q, a) = q_e$.

Сега лесно може да се докаже, че $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$. □

q_e - error състояние

$\mathcal{A}'$ симулира $\mathcal{A}$

✍ Довършете доказателството!

Твърдение 2.3. Класът на автоматните езици е затворен относно операцията *обединение*. Това означава, че ако L_1 и L_2 са два произволни автоматни езика над азбуката Σ, то $L_1 \cup L_2$ също е автоматен език.

Доказателство. Нека $L_1 = \mathcal{L}(\mathcal{A}_1)$ и $L_2 = \mathcal{L}(\mathcal{A}_2)$, където

$$\mathcal{A}_1 = \langle Q_1, \Sigma, s_1, \delta_1, F_1\rangle,\ \mathcal{A}_2 = \langle Q_2, \Sigma, s_2, \delta_2, F_2\rangle$$

са *тотални*. Определяме автомата $\mathcal{A} = \langle Q, \Sigma, q_{\texttt{start}}, \delta, F\rangle$, който разпознава $L_1 \cup L_2$ по следния начин:

Защо изискваме $\mathcal{A}_1$ и $\mathcal{A}_2$ да са тотални?

- $Q = Q_1 \times Q_2$;
- Определяме за всяко $\langle r_1, r_2\rangle \in Q$ и всяко $a \in \Sigma$,

$$\delta(\langle r_1, r_2\rangle, a) = \langle \delta_1(r_1, a), \delta_2(r_2, a)\rangle;$$

- $s = \langle s_1, s_2\rangle$;
- $F = \{\langle r_1, r_2\rangle \mid r_1 \in F_1 \vee r_2 \in F_2\} = (F_1 \times Q_2) \cup (Q_1 \times F_2)$.

□

Едновременно симулираме изчисление и по двата автомата

По-нататък ще дадем друга конструкция за обединение, която ще бъде по-ефективна относно броя на състоянията

✍ Проверете, че $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$

Следствие 2.1. Класът на автоматните езици е затворен относно операцията **сечение**. Това означава, че ако L_1 и L_2 са два произволни автоматни езика над азбуката Σ, то $L_1 \cap L_2$ също е автоматен език.

Доказателство. Използвайте конструкцията на автомата $\mathcal{A}$ от *Твърдение 2.3*, с единствената разлика, че тук избираме финалните състояния да бъдат елементите на множеството

$$F = \{\langle q_1, q_2\rangle \mid q_1 \in F_1 \ \&\ q_2 \in F_2\} = F_1 \times F_2.$$

□

✍ Докажете, че така построения автомат $\mathcal{A}$ разпознава $L_1 \cap L_2$!

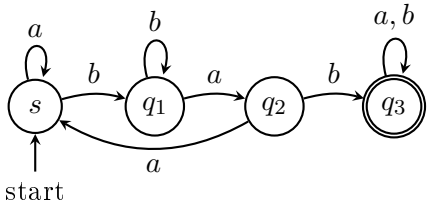
Твърдение 2.4. Нека L е автоматен език. Тогава $\Sigma^\star \setminus L$ също е автоматен език.

Доказателство. Нека $L = L(\mathcal{A})$, където $\mathcal{A} = \langle Q, \Sigma, q_{\texttt{start}}, \delta, F\rangle$ е **тотален**. Да вземем автомата $\mathcal{A}' = \langle Q, \Sigma, s, \delta, Q \setminus F\rangle$, т.е. $\mathcal{A}'$ е същия като $\mathcal{A}$, с единствената разлика, че финалните състояния на $\mathcal{A}'$ са тези състояния, които **не** са финални в $\mathcal{A}$. □

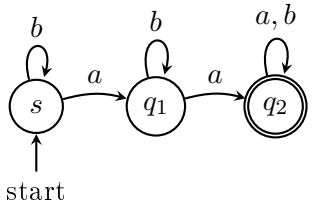
Защо искаме $\mathcal{A}$ да бъде тотален ?

✍ Проверете, че $\Sigma^\star \setminus L = \mathcal{L}(\mathcal{A}')$

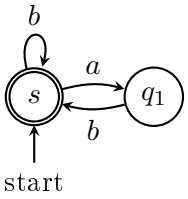
Пример 2.1. Да разгледаме няколко примера за автомати и езиците, които разпознават. Дефинирайте функцията на преходите δ за всеки автомат.



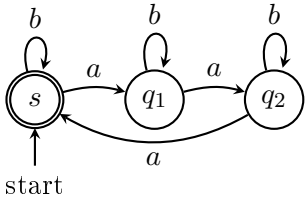
(а) $\{\omega \in \{a,b\}^\star \mid \omega$ съдържа $bab\}$



(б) $\{\omega \in \{a,b\}^\star \mid N_a(\omega) \geq 2\}$



(в) $\{\omega \in \{a,b\}^\star \mid$ всяко a в ω се следва от поне едно $b\}$



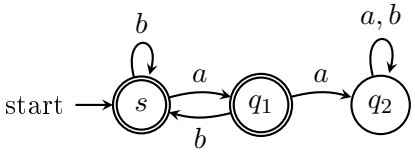
(г) $\{\omega \in \{a,b\}^\star \mid N_a(\omega) \equiv 0 \pmod 3\}$

В повечето от горните примери може сравнително лесно да се съобрази, че построения автомат разпознава желания език. При по-сложни задачи обаче, ще се наложи да дадем доказателство, като обикновено се прилага *метода на математическата индукция* върху дължината на думите. Ще разгледаме няколко такива примера.

Задача 2.1. Докажете, че езикът L е автоматен, където

$$L = \{\alpha \in \{a,b\}^\star \mid \alpha \text{ не съдържа две поредни срещания на } a\}.$$

Доказателство. Да разгледаме $\mathcal{A} = \langle Q, \Sigma, q_{\texttt{start}}, \delta, F\rangle$ с функция на преходите



Най-лесния начин да се сетим как да построим $\mathcal{A}$ е като първо построим тотален автомат за езика, който разпознава тези думи, в които се съдържат две поредни срещания на a и вземем неговото допълнение съгласно Твърдение 2.4

Ще докажем, че $L = \mathcal{L}(\mathcal{A})$. Първо ще се концентрираме върху доказателството на $\mathcal{L}(\mathcal{A}) \subseteq L$. Ще докажем с индукция по дължината на думата α, че:

Озн. $|\alpha|$ - дължината на думата α

(1) ако $\delta^\star(s, \alpha) = s$, то α не съдържа две поредни срещания на a и ако $|\alpha| > 0$, то α завършва на b;

(2) ако $\delta^\star(s, \alpha) = q_1$, то α не съдържа две поредни срещания на a и завършва на a.

За $|\alpha| = 0$, то твърденията (1) и (2) са ясни (Защо?). Да приемем, че твърденията (1) и (2) са верни за произволни думи α с дължина n. Нека $|\alpha| = n + 1$, т.е. $\alpha = \beta x$, където $|\beta| = n$ и $x \in \Sigma$. Ще докажем (1) и (2) за α.

- Нека $\delta^\star(s, \beta x) = s = \delta(\delta^\star(s, \beta), x)$. Според дефиницията на функцията δ, $x = b$ и $\delta^\star(s, \beta) \in \{s, q_1\}$. Тогава по **И.П.** за (1) и (2), β не съдържа две поредни срещания на a. Тогава е очевидно, че βx също не съдържа две поредни срещания на a.

- Нека $\delta^\star(s, \beta x) = q_1 = \delta(\delta^\star(s, \beta), x)$. Според дефиницията на δ, $x = a$ и $\delta^\star(s, \beta) = s$. Тогава по **И.П.** за (2), β не съдържа две поредни срещания на a и завършва на b. Тогава е очевидно, че βx също не съдържа две поредни срещания на a.

Така доказахме с индукция по дължината на думата, че за всяка дума α са изпълнени твърденията (1) и (2). По дефиниция, ако $\alpha \in \mathcal{L}(\mathcal{A})$, то $\delta^\star(s, \alpha) \in \{s, q_1\}$ и от (1) и (2) следва, че и в двата случа α не съдържа две поредни срещания на буквата a, т.е. $\alpha \in L$. С други думи, доказахме, че

$$\mathcal{L}(\mathcal{A}) \subseteq L.$$

Сега ще докажем другата посока, т.е. $L \subseteq \mathcal{L}(\mathcal{A})$. Това означава да докажем, че

$$(\forall \alpha \in \Sigma^\star)[\alpha \in L \ \Rightarrow \ \delta^\star(s, \alpha) \in F],$$

което е еквивалентно на

Да напомним, че $p \Rightarrow q \equiv \neg q \Rightarrow \neg p$

$$(\forall \alpha \in \Sigma^\star)[\delta^\star(s, \alpha) \notin F \ \Rightarrow \ \alpha \notin L]. \tag{2.1}$$

Това е лесно да се съобрази. Щом $\delta^\star(s, \alpha) \notin F$, то $\delta^\star(s, \alpha) = q_2$ и думата α може да се представи по следния начин:

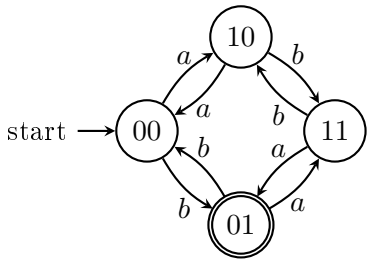
$$\alpha = \beta a \gamma \ \& \ \delta^\star(s, \beta) = q_1.$$

Използвайки свойство (2) от по-горе, понеже $\delta^\star(s, \beta) = q_1$, то β не съдържа две поредни срещания на a, но завършва на a. Сега е очевидно, че βa съдържа две поредни срещания на a и щом βa е префикс на α, то думата $\alpha \notin L$. С това доказахме Свойство 2.1, а следователно и посоката $L \subseteq \mathcal{L}(\mathcal{A})$. □

Задача 2.2. Докажете, че следния език е автоматен:

$L = \{w \in \{a,b\}^\star \mid a$ се среща четен брой, докато b нечетен брой пъти в $w\}$.

Упътване. Разгледайте автомата $\mathcal{A}$:



Фигура 2.2: $\mathcal{L}(\mathcal{A}) \stackrel{?}{=} L$

Докажете с индукция по дължината на думата ω, че:

а) $\delta^\star(q_{00}, \omega) = q_{00} \implies (\exists k, n \in \mathbb{N})[N_a(\omega) = 2k \ \&\ N_b(\omega) = 2n]$;

б) $\delta^\star(q_{00}, \omega) = q_{01} \implies (\exists k, n \in \mathbb{N})[N_a(\omega) = 2k \ \&\ N_b(\omega) = 2n+1]$;

в) $\delta^\star(q_{00}, \omega) = q_{10} \implies (\exists k, n \in \mathbb{N})[N_a(\omega) = 2k+1 \ \&\ N_b(\omega) = 2n]$;

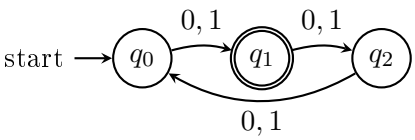
г) $\delta^\star(q_{00}, \omega) = q_{11} \implies (\exists k, n \in \mathbb{N})[N_a(\omega) = 2k+1 \ \&\ N_b(\omega) = 2n+1]$;

□

Задача 2.3. Докажете, че езикът $L = \{w \in \{0,1\}^\star \mid |w| \equiv 1 \bmod 3\}$ е автоматен.

Упътване. Разгледайте автомата $\mathcal{A}$:

$|w|$ = дължината на думата w



Фигура 2.3: $\mathcal{L}(\mathcal{A}) \stackrel{?}{=} \{w \in \{0,1\}^\star \mid |w| \equiv 1 \bmod 3\}$

Докажете с индукция по дължината на думата w едновременно следните три твърдения:

(1) $\delta^\star(q_0, w) = q_0 \implies |w| \equiv 0 \bmod 3$;

(2) $\delta^\star(q_0, w) = q_1 \implies |w| \equiv 1 \bmod 3$;

(3) $\delta^\star(q_0, w) = q_2 \implies |w| \equiv 2 \bmod 3$.

□

За една дума $\alpha \in \{0,1\}^\star$, нека с $\alpha_{(2)}$ да означим числото в десетична бройна система, което се представя в двоична бройна система като α. Например, $1101_{(2)} = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 13$. Тогава имаме следните свойства:

- $\varepsilon_{(2)} = 0$,
- $(\alpha 0)_{(2)} = 2 \cdot (\alpha)_{(2)}$,
- $(\alpha 1)_{(2)} = 2 \cdot (\alpha)_{(2)} + 1$.

Да отбележим, че за всяко число n има безкрайно много думи α, за които $\alpha_{(2)} = n$. Например, $10_{(2)} = 010_{(2)} = 0010_{(2)} = \cdots$

Задача 2.4. Докажете, че $L = \{\omega \in \{0,1\}^\star \mid \omega_{(2)} \equiv 2 \pmod 3\}$ е автоматен.

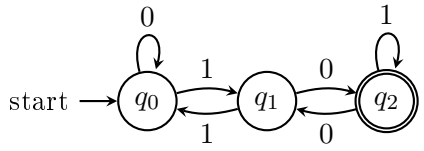
Доказателство. Нашият автомат ще има три състояния $\{q_0, q_1, q_2\}$, като началното състояние ще бъде q_0. Целта ни е да дефинираме така автомата, че да имаме следното свойство:

$$(\forall \alpha \in \Sigma^\star)(\forall i < 3)[\alpha_{(2)} \equiv i \pmod 3 \ \Leftrightarrow \ \delta^\star(q_0, \alpha) = q_i], \tag{2.2}$$

т.е. всяко състояние отговаря на определен остатък при деление на три. Понеже искаме нашия автомат да разпознава тези думи α, за които $\alpha_{(2)} \equiv 2 \mod 3$, финалното състояние ще бъде q_2. Дефинираме функцията δ следвайки свойствата:

- $\delta(q_0, 0) = q_0$, защото ако $\alpha_{(2)} \equiv 0 \bmod 3$, то $(\alpha 0)_{(2)} \equiv 0 \bmod 3$;
- $\delta(q_0, 1) = q_1$, защото ако $\alpha_{(2)} \equiv 0 \bmod 3$, то $(\alpha 1)_{(2)} \equiv 1 \bmod 3$;
- $\delta(q_1, 0) = q_2$, защото ако $\alpha_{(2)} \equiv 1 \bmod 3$, то $(\alpha 0)_{(2)} \equiv 2 \bmod 3$;
- $\delta(q_1, 1) = q_0$, защото ако $\alpha_{(2)} \equiv 1 \bmod 3$, то $(\alpha 1)_{(2)} \equiv 0 \bmod 3$;
- $\delta(q_2, 0) = q_1$, защото ако $\alpha_{(2)} \equiv 2 \bmod 3$, то $(\alpha 0)_{(2)} \equiv 1 \bmod 3$;
- $\delta(q_2, 1) = q_2$, защото ако $\alpha_{(2)} \equiv 2 \bmod 3$, то $(\alpha 1)_{(2)} \equiv 2 \bmod 3$.

Ето и картинка на автомата $\mathcal{A}$:



Фигура 2.4: $\mathcal{L}(\mathcal{A}) \stackrel{?}{=} \{\alpha \in \{0,1\}^\star \mid \alpha_{(2)} \equiv 2 \pmod 3\}$

Да разгледаме твърденията:

(1) $\delta^\star(q_0, \alpha) = q_0 \implies \alpha_{(2)} \equiv 0 \mod 3$;

(2) $\delta^\star(q_0, \alpha) = q_1 \implies \alpha_{(2)} \equiv 1 \mod 3$;

(3) $\delta^\star(q_0, \alpha) = q_2 \implies \alpha_{(2)} \equiv 2 \mod 3$.

Ще докажем (1), (2) и (3) *едновременно* с индукция по дължината на думата α. За $|\alpha| = 0$, всички условия са изпълнени. (Защо?) Да приемем, че (1), (2) и (3) са изпълнени за думи с дължина n. Нека $|\alpha| = n + 1$, т.е. $\alpha = \beta x$, $|\beta| = n$. За да приложим индукционното предположение, ще използваме следното свойство:

$$\delta^\star(q_0, \beta x) = \delta(\delta^\star(q_0, \beta), x).$$

Ще докажем подробно само (3) понеже другите твърдения се доказват по сходен начин. Нека $\delta^\star(q_0, \beta x) = q_2$. Имаме два случая:

Обърнете внимание, че в доказателството на (3) използваме И.П. не само за (3), но и за (2). Поради тази причина трябва да докажем трите свойства едновременно

- $x = 0$. Тогава, по дефиницията на δ, $\delta(q_1, 0) = q_2$ и следователно, $\delta^\star(q_0, \beta) = q_1$. По **И.П.** за (2) с β,

  $$\delta^\star(q_0, \beta) = q_1 \;\Rightarrow\; \beta_{(2)} \equiv 1 \bmod 3$$

  Тогава, $(\beta 0)_{(2)} \equiv 2 \mod 3$. Така доказахме, че

  $$\delta^\star(q_0, \beta 0) = q_2 \;\Rightarrow\; (\beta 0)_{(2)} \equiv 2 \bmod 3.$$

- $x = 1$. Тогава, по дефиницията на δ, $\delta(q_2, 1) = q_2$ и следователно, $\delta^\star(q_0, \beta) = q_2$. По **И.П.** за (3) с β,

  $$\delta^\star(q_0, \beta) = q_2 \;\Rightarrow\; \beta_{(2)} \equiv 2 \mod 3.$$

  Тогава, $(\beta 1)_{(2)} \equiv 2 \mod 3$. Така доказахме, че

  $$\delta^\star(q_0, \beta 1) = q_2 \;\Rightarrow\; (\beta 1)_{(2)} \equiv 2 \mod 3.$$

За да докажем (1), нека $\delta^\star(q_0, \beta x) = q_0$.

- $x = 0$. Разсъжденията са аналогични, като използваме **И.П.** за (1).
- $x = 1$. Разсъжденията са аналогични, като използваме **И.П.** за (2).

По същия начин доказваме и (2). Нека $\delta^\star(q_0, \beta x) = q_1$.

- При $x = 0$, използваме **И.П.** за (3).
- При $x = 1$, използваме **И.П.** за (1).

От (1), (2) и (3) следва директно, че $\mathcal{L}(\mathcal{A}) \subseteq L$.

За другата посока, нека $\alpha \in L$, т.е. $\alpha_{(2)} \equiv 2 \bmod 3$. Ако допуснем, че $\alpha \notin \mathcal{L}(\mathcal{A})$, то това означава, че $\delta^\star(q_0, \alpha) \in \{q_0, q_1\}$. Но в тези случаи получаваме от твърдения (1) и (2), че $\alpha_{(2)} \equiv 0 \bmod 3$ или $\alpha_{(2)} \equiv 1 \bmod 3$. Това е противоречие с избора на $\alpha \in L$. Следователно, ако $\alpha \in L$, то $\delta(q_0, \alpha) = q_2$. Така доказахме и посоката $L \subseteq \mathcal{L}(\mathcal{A})$. □

2.2 Регулярни изрази и езици

Да фиксираме една непразна азбука Σ. **Регулярните изрази r** могат да се опишат със следната абстрактна граматика

$$\mathbf{r} ::= \emptyset \mid \varepsilon \mid \mathbf{a} \mid \mathbf{r_1} \cdot \mathbf{r_2} \mid \mathbf{r_1} + \mathbf{r_2} \mid \mathbf{r_1^\star},$$

където ε означава празната дума и a е произволна буква от азбуката Σ.

Друг начин да се опишат регулярните изрази е по следния начин:

Това е пример за индуктивна дефиниция

- Символите $\emptyset$, ε и $\mathbf{a}$, всяка буква $a \in \Sigma$, са регулярен изрази;
- Ако $\mathbf{r_1}$ и $\mathbf{r_2}$ са регулярни изрази, то $\mathbf{r_1} \cdot \mathbf{r_2}$, $\mathbf{r_1} + \mathbf{r_2}$ и $\mathbf{r_1^\star}$ също са регулярни изрази;
- Всеки регулярен израз е получен по някое от горните правила.

Сега ще дефинираме езиците, които се описват с регулярни изрази. Тези езици се наричат **регулярни**. Това ще направим следвайки индуктивната дефиниция на регулярните изрази, т.е. за всеки регулярен израз $\mathbf{r}$ ще определим език $\mathcal{L}(\mathbf{r})$.

Това е друг пример за индуктивна (рекурсивна) дефиниция.

- $\{\varepsilon\}$ е регулярен език, който се разпознава от регулярния израз ε. Означаваме $\mathcal{L}(\varepsilon) = \{\varepsilon\}$;
- за всяка буква $a \in \Sigma$, $\{a\}$ е регулярен език, който се разпознава от регулярния израз $\mathbf{a}$. Означаваме $\mathcal{L}(\mathbf{a}) = \{a\}$;
- $\emptyset$ е регулярен език, който се разпознава от регулярния израз $\emptyset$. Означаваме $\mathcal{L}(\emptyset) = \emptyset$;

- Нека L_1 и L_2 са регулярни езици, т.е. съществуват регулярни изрази $\mathbf{r_1}$ и $\mathbf{r_2}$, за които $\mathcal{L}(\mathbf{r}_1) = L_1$ и $\mathcal{L}(\mathbf{r_2}) = L_2$. Тогава:
  - $L_1 \cup L_2$ е регулярен език, който се описва с регулярния израз $\mathbf{r_1} + \mathbf{r_2}$. Това означава, че $\mathcal{L}(\mathbf{r_1}) \cup \mathcal{L}(\mathbf{r_2}) = \mathcal{L}(\mathbf{r_1} + \mathbf{r_2})$.
  - $L_1 \cdot L_2$ е регулярен език, който се описва с регулярния израз $\mathbf{r_1} \cdot \mathbf{r_2}$. Това означава, че $\mathcal{L}(\mathbf{r_1}) \cdot \mathcal{L}(\mathbf{r_2}) = \mathcal{L}(\mathbf{r_1} \cdot \mathbf{r_2})$.

    Тази операция се наричка конкатенация. Обикновено изпускаме знака $\cdot$
  - $L_1^\star$ е регуларен език, който се описва с регулярния израз $\mathbf{r_1^\star}$. Това означава, че $L_1^\star = \mathcal{L}(\mathbf{r_1^\star})$.

    Звезда на Клини

Забележка. Ние знаем, че:

- Всеки регулярен израз представлява крайна дума над крайна азбука. Това означава, че множеството от всички регулярни изрази е изброимо безкрайно. Оттук следва, че всички регулярни езици образуват изброимо безкрайно множество.
- Понеже Σ е крайна азбука, то $\Sigma^\star$ е изброимо безкрайно множество;
- Един език над азбуката Σ представлява елемент на $\mathscr{P}(\Sigma^\star)$. Това означава, че всички езици над азбуката Σ представляват неизброимо безкрайно множество.

От всичко това следва, че има езици, които не са регулярни. По-нататък ще видим примери за такива езици.

Пример 2.2. Нека да разгледаме няколко примера какво точно представлява прилагането на операцията звезда на Клини върху един език.

- Нека $L = \{0, 11\}$. Тогава:
  - $L^0 = \{\varepsilon\}$, $L^1 = L$,
  - $L^2 = L^1 \cdot L^1 = \{00, 011, 110, 1111\}$,
  - $L^3 = L^1 \cdot L^2 = \{000, 0011, 0110, 01111, 1100, 11011, 11110, 111111\}$.
- Нека $L = \emptyset$. Тогава:
  - $L^0 = \{\varepsilon\}$,
  - $L^1 = \emptyset$,
  - $L^2 = L^1 \cdot L^1 = \emptyset$.

  Получаваме, че $L^\star = \{\varepsilon\}$, т.е. *краен* език
- Нека $L = \{0^i \mid i \in \mathbb{N}\} = \{\varepsilon, 0, 00, 000, \dots\}$. Тогава лесно може да се види, че $L = L^\star$.

Пример 2.3. Нека да построим регулярни изрази за всеки от езиците от *Пример 2.1*.

В [Sip97, стр. 73] е показан алгоритъм, за който по един автомат може да се получи регулярен израз описващ езика на автомата. Ние няма да разглеждаме този алгоритъм.

а) Нека $\mathbf{r} = (\mathbf{a}+\mathbf{b})^\star\mathbf{bab}(\mathbf{a}+\mathbf{b})^\star$. Тогава

$$\mathcal{L}(\mathbf{r}) = \{\omega \in \{a,b\}^\star \mid \omega \text{ съдържа } bab\}.$$

б) Нека $\mathbf{r} = \mathbf{b}^\star\mathbf{ab}^\star\mathbf{a}(\mathbf{a}+\mathbf{b})^\star$. Тогава

$$\mathcal{L}(\mathbf{r}) = \{\omega \in \{a,b\}^\star \mid N_a(\omega) \geq 2\}.$$

в) Нека $\mathbf{r} = (\mathbf{abb}^\star)^\star$. Тогава

$$\mathcal{L}(\mathbf{r}) = \{\omega \in \{a,b\}^\star \mid \text{ всяко } a \text{ в } \omega \text{ се следва от поне едно } b\}.$$

г) Нека $\mathbf{r} = (\mathbf{b}^\star\mathbf{ab}^\star\mathbf{ab}^\star\mathbf{ab}^\star)^\star$. Тогава

$$\mathcal{L}(\mathbf{r}) = \{\omega \in \{a,b\}^\star \mid N_a(\omega) \equiv 0 \bmod 3\}.$$

Задача 2.5. За произволни регулярни изрази r и s, проверете:

а) $r + s = s + r$;

б) $(\varepsilon + r)^\star = r^\star$;

в) $\emptyset^\star = \varepsilon$;

г) $(r^\star s^\star) = (r + s)^\star$;

д) $(r^\star)^\star = r^\star$;

е) $(rs + r)^\star r = r(sr + r)^\star$;

ж) $s(rs + s)^\star r = rr^\star s(rr^\star s)^\star$;

з) $(r + s)^\star = r^\star + s^\star$;

и) $\emptyset^\star = \varepsilon^\star$;

Теорема 2.1 (Клини)**.** Всеки автоматен език се описва с регулярен израз.

Доказателство. Нека $L = \mathcal{L}(\mathcal{A})$, за някой краен детерминиран автомат $\mathcal{A}$. Да фиксираме едно изброяване на състоянията $Q = \{q_1, \ldots, q_n\}$, като началното състояние е q_1. Ще означаваме с $L(i,j,k)$ множеството от тези думи, които могат да се разпознаят от автомата по път, който започва от q_i, завършва в q_j, и междинните състояния имат индекси $\leq k$. Например, за думата $\alpha = a_1a_2\cdots a_n$ имаме, че $\alpha \in L(i,j,k)$ точно

[PL98, стр. 79]; [HU79, стр. 33]

тогава, когато съществуват състояния $q_{l_1}, \ldots, q_{l_{n-1}}$, като $l_1, \ldots, l_{n-1} \leq k$ и

$$q_i \overset{a_1}{\to} q_{l_1} \overset{a_2}{\to} q_{l_2} \overset{a_3}{\to} \cdots \overset{a_{n-1}}{\to} q_{l_{n-1}} \overset{a_n}{\to} q_j.$$

Тогава за $n = |Q|$,

$$L(i, j, n) = \{\alpha \in \Sigma^\star \mid \delta^\star(q_i, \alpha) = q_j\}.$$

Така получаваме, че

$$\mathcal{L}(\mathcal{A}) = \bigcup\{L(1, j, n) \mid q_j \in F\} = \bigcup_{q_j \in F} L(1, j, n).$$

Ще докажем с *индукция по* k, че за всяко i, j, k, множествата от думи $L(i, j, k)$ се описват с регулярен израз $\mathbf{r^k_{i,j}}$

а) Нека $k = 0$. Ще докажем, че за всяко i, j, $L(i, j, 0)$ се описва с регулярен израз. Имаме да разгледаме два случая.

Ако $i = j$, то

$$L(i, j, 0) = \{\varepsilon\} \cup \{a \in \Sigma \mid \delta(q_i, a) = q_j\}. \tag{2.3}$$

Ако $i \neq j$, то

$$L(i, j, 0) = \{a \in \Sigma \mid \delta(q_i, a) = q_j\}.$$

И в двата случая, понеже $L(i, j, 0)$ е краен език, то е ясно, че той се описва с регулярен израз.

б) Да предположим, че $k > 0$ и за всяко i, j, можем да намерим регулярните изрази съответстващи на $L(i, j, k-1)$. Тогава

$$L(i, j, k) = L(i, j, k-1) \ \cup \ L(i, k, k-1) \cdot (L(k, k, k-1)^\star) \cdot L(k, j, k-1).$$

Тогава по **И.П.** следва, че $L(i, j, k)$ може да се опише с регулярен израз, който е

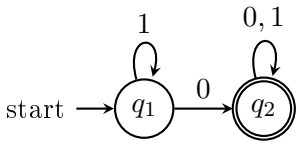
$$\mathbf{r^k_{i,j}} = \mathbf{r^{k-1}_{i,j}} + \mathbf{r^{k-1}_{i,k}} \cdot (\mathbf{r^{k-1}_{k,k}})^\star \cdot \mathbf{r^{k-1}_{k,j}}. \tag{2.4}$$

Заключаваме, че за всяко i, j, k, $L(i, j, k)$ може да се опише с регулярен израз $\mathbf{r^k_{i,j}}$. Тогава ако $F = \{q_{i_1}, \ldots, q_{i_k}\}$, то $\mathcal{L}(\mathcal{A})$ се описва с регулярния израз

$$\mathbf{r^n_{1,i_1}} + \mathbf{r^n_{1,i_2}} + \cdots + \mathbf{r^n_{1,i_k}}.$$

□

Пример 2.4. Да разгледаме следния автомат:



Фигура 2.5: Автомат разпознаващ $\mathcal{L}(\mathbf{1^\star 0(0+1)^\star})$

Лесно се съобразява, че езикът на автомата от Фигура ?? се описва с регулярния израз $\mathbf{1^\star 0(0+1)^\star}$. Следвайки конструкцията от доказателството на *Теорема 2.1*, езикът на този автомат се описва с регулярния израз $\mathbf{r^2_{1,2}}$, защото началното състояние е q_1, финалното е q_2 и броят на състоянията в автомата е 2.

$$
\begin{aligned}
\mathbf{r^2_{1,2}} &= \underbrace{\mathbf{r^1_{1,2}}}_{\mathbf{1^\star 0}} + \underbrace{\mathbf{r^1_{1,2}}}_{\mathbf{1^\star 0}} \cdot \underbrace{(\mathbf{r^1_{2,2}})^\star}_{(\varepsilon+\mathbf{0}+\mathbf{1})^\star} \cdot \underbrace{\mathbf{r^1_{2,2}}}_{\varepsilon+\mathbf{0}+\mathbf{1}} && \text{(според (2.4))} \\
&= \mathbf{1^\star 0} + \mathbf{1^\star 0}(\varepsilon + \mathbf{0} + \mathbf{1})^\star(\varepsilon + \mathbf{0} + \mathbf{1}) \\
&= \mathbf{1^\star 0} + \mathbf{1^\star 0}(\varepsilon + \mathbf{0} + \mathbf{1})^+ && (\mathbf{r}^+ = \mathbf{r^\star r}) \\
&= \mathbf{1^\star 0} + \mathbf{1^\star 0}(\mathbf{0} + \mathbf{1})^\star && (\mathbf{r^\star} = (\varepsilon + \mathbf{r})^+) \\
&= \mathbf{1^\star 0}(\varepsilon + (\mathbf{0} + \mathbf{1})^\star) && (\mathbf{r + rq} = \mathbf{r}(\varepsilon + \mathbf{q})) \\
&= \mathbf{1^\star 0(0+1)^\star} && (\mathbf{r^\star} = \varepsilon + \mathbf{r^\star})
\end{aligned}
$$

Тук използвахме, че:

$$
\begin{aligned}
\mathbf{r^1_{1,2}} &= \underbrace{\mathbf{r^0_{1,2}}}_{\mathbf{0}} + \underbrace{\mathbf{r^0_{1,1}}}_{\varepsilon+\mathbf{1}} \cdot \underbrace{(\mathbf{r^0_{1,1}})^\star}_{(\varepsilon+\mathbf{1})^\star} \cdot \underbrace{\mathbf{r^0_{1,2}}}_{\mathbf{0}} \\
&= \mathbf{0} + (\varepsilon + \mathbf{1})(\varepsilon + \mathbf{1})^\star \mathbf{0} \\
&= \mathbf{0} + \mathbf{1^\star 0} \\
&= \mathbf{1^\star 0}, \\
\mathbf{r^1_{2,2}} &= \underbrace{\mathbf{r^0_{2,2}}}_{\varepsilon+\mathbf{0}+\mathbf{1}} + \underbrace{\mathbf{r^0_{2,1}}}_{\emptyset} \cdot \underbrace{(\mathbf{r^0_{1,1}})^\star}_{\varepsilon+\mathbf{1}} \cdot \underbrace{\mathbf{r^0_{1,2}}}_{\mathbf{0}} \\
&= \varepsilon + \mathbf{0} + \mathbf{1} + \emptyset(\varepsilon + \mathbf{1})^\star \mathbf{0} \\
&= \varepsilon + 0 + 1 && \text{(защото } \emptyset \cdot \mathbf{r} = \emptyset)
\end{aligned}
$$

Следващата ни цел е да видим, че имаме и обратната посока на горната лема. Ще докажем, че всеки регулярен език е автоматен. За тази цел първо ще въведем едно обобщение на понятието краен детерминиран автомат.

2.3 Недетерминирани крайни автомати

Определение 2.2. Недетерминиран краен автомат представлява

$$\mathcal{N} = \langle Q, \Sigma, q_{\texttt{start}}, \Delta, F \rangle,$$

Въведени от Рабин и Скот [RS59]

За яснота, често ще означаваме с $\mathcal{N}$ недетерминирани автомати

- Q е крайно множество от състояния;
- Σ е крайна азбука;
- $\Delta : Q \times \Sigma \to \mathscr{P}(Q)$ е функцията на преходите. Обърнете внимание, че тя е тотална.

Да напомним, че $\mathscr{P}(Q) = \{R \mid R \subseteq Q\}$, $|\mathscr{P}(Q)| = 2^{|Q|}$

- $q_{\texttt{start}} \in Q$ е началното състояние;
- $F \subseteq Q$ е множеството от финални състояния.

В [Sip97] се позволяват ϵ-преходи

Удобно е да разширим функцията на преходите $\Delta : Q \times \Sigma \to \mathscr{P}(Q)$ до функцията $\Delta^\star : Q \times \Sigma^\star \to \mathscr{P}(Q)$ по следния начин:

- $\Delta^\star(q, \varepsilon) = \{q\}$;
- $\Delta^\star(q, b\alpha) = \bigcup_{p \in \Delta(q,b)} \Delta^\star(p, \alpha)$;

Съобразете, че $\Delta^\star(q, \alpha b) = \bigcup_{p \in \Delta^\star(q,\alpha)} \Delta(p, b)$

Теорема 2.2 (Рабин-Скот [RS59])**.** За всеки НКА $\mathcal{N}$ съществува еквивалентен на него ДКА $\mathcal{D}$, т.е. $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{D})$.

Упътване. Нека $\mathcal{N} = \langle Q, \Sigma, q_{\texttt{start}}, \Delta, F \rangle$. Ще построим детерминиран автомат

$$\mathcal{D} = (Q', \Sigma, \delta, q'_{\texttt{start}}, F'),$$

за който $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{D})$. Конструкцията е следната:

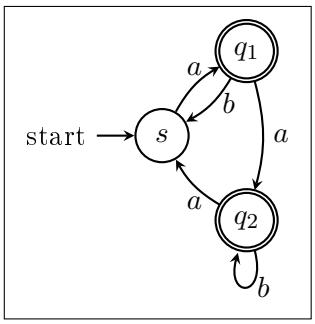
Да отбележим, че детерминираният автомат $\mathcal{D}$ има не повече от $2^{|Q|}$ на брой състояния Q'

- $Q' = \mathscr{P}(Q)$;
- $\delta(R, a) = \{q \in Q \mid (\exists r \in R)[q \in \Delta(r, a)]\} = \bigcup_{r \in R} \Delta(r, a)$;
- $q'_{\texttt{start}} = \{q_{\texttt{start}}\}$;
- $F' = \{R \subseteq Q \mid R \cap F \neq \emptyset\}$.

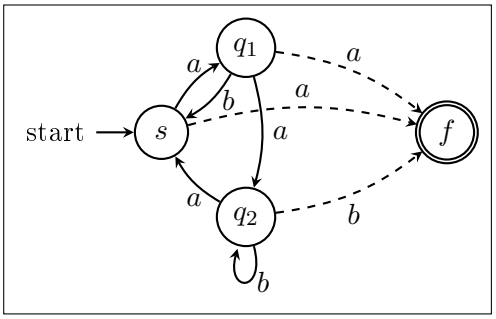
□

Задача 2.6. За всеки НКА $\mathcal{N}$ съществува НКА $\mathcal{N}'$ с едно финално състояние, за който $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{N}')$.

Упътване. Вместо формална конструкция, да разгледаме един пример, който илюстрира идеята.



(а) автомат $\mathcal{N}$



(б) автомат $\mathcal{N}'$, $\mathcal{L}(\mathcal{N}') = \mathcal{L}(\mathcal{N})$

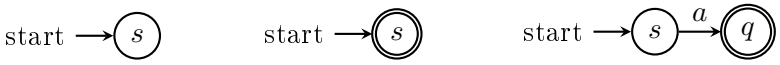
За произволен автомат $\mathcal{N}$, формулирайте точно конструкцията на $\mathcal{N}'$ с едно финално състояние и докажете, че наистина $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{N}')$. Обърнете внимание, че примера показва, че е възможно $\mathcal{N}$ да е детерминиран автомат, но полученият $\mathcal{N}'$ да бъде недетерминиран. □

Задача 2.7. Докажете, че ако L е автоматен език, то $L^R = \{\omega^R \mid \omega \in L\}$ също е автоматен.

Нека $\mathcal{A}$, $L = \mathcal{L}(\mathcal{A})$, е само с едно финално състояние.

Лема 2.1. Съществува НКА $\mathcal{N} = \langle Q, \Sigma, q_{\texttt{start}}, \Delta, F\rangle$, който разпознава езика $L(r)$, където $r = \emptyset$, $r = \varepsilon$ или $r = a$, за $a \in \Sigma$.

Упътване.



(а) $L(\emptyset)$ (б) $L(\varepsilon)$ (в) $L(a)$

□

Лема 2.2. Класът на автоматните езици е затворен относно операцията *конкатенация*. Това означава, че ако L_1 и L_2 са два произволни автоматни езика, то $L_1 \cdot L_2$ също е автоматен език.

Доказателство. Нека са дадени автоматите:

- $\mathcal{N}_1 = \langle Q_1, \Sigma, s_1, \Delta_1, F_1\rangle$, като $\mathcal{L}(\mathcal{N}_1) = L_1$;
- $\mathcal{N}_2 = \langle Q_2, \Sigma, s_2, \Delta_2, F_2\rangle$, като $\mathcal{L}(\mathcal{N}_2) = L_2$.

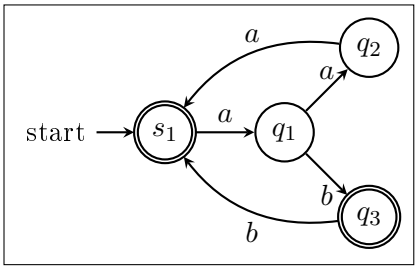
Ще дефинираме автомата $\mathcal{N} = \langle Q, \Sigma, q_{\texttt{start}}, \Delta, F\rangle$ като

$$\mathcal{L}(\mathcal{N}) = L_1 \cdot L_2 = \mathcal{L}(\mathcal{N}_1) \cdot \mathcal{L}(\mathcal{N}_2).$$

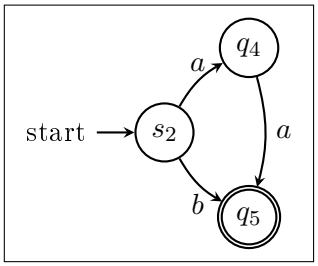
- $Q = Q_1 \cup Q_2$;

- $s = s_1$;
- $F = \begin{cases} F_1 \cup F_2, & \text{ако } s_2 \in F_2 \\ F_2, & \text{иначе.} \end{cases}$
- $\Delta(q,a) = \begin{cases} \Delta_1(q,a), & \text{ако } q \in Q_1 \setminus F_1 \ \&\ a \in \Sigma \\ \Delta_2(q,a), & \text{ако } q \in Q_2 \ \&\ a \in \Sigma \\ \Delta_1(q,a) \cup \Delta_2(s_2,a), & \text{ако } q \in F_1 \ \&\ a \in \Sigma. \end{cases}$

□

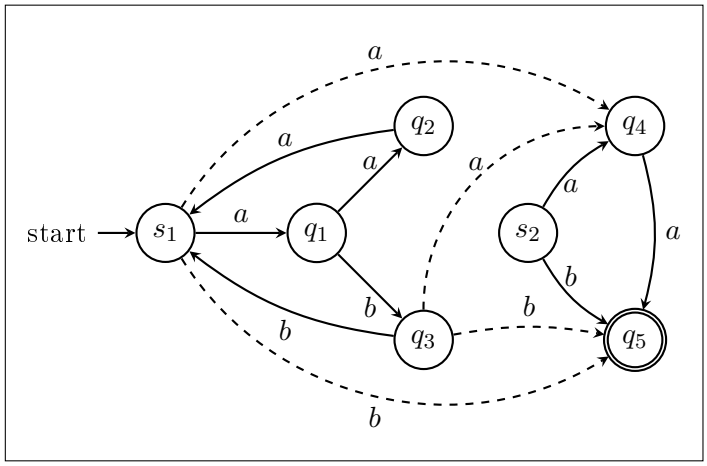


(а) автомат $\mathcal{N}_1$



(б) автомат $\mathcal{N}_2$

Пример 2.5. За да построим автомат, който разпознава конкатенацията на $\mathcal{L}(\mathcal{N}_1)$ и $\mathcal{L}(\mathcal{N}_2)$, трябва да свържем финалните състояния на $\mathcal{N}_1$ с изходящите от s_2 състояния на $\mathcal{N}_2$.



Фигура 2.9: $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{N}_1) \cdot \mathcal{L}(\mathcal{N}_2)$

Обърнете внимание, че $\mathcal{N}_1$ и $\mathcal{N}_2$ са детерминирани автомати, но $\mathcal{N}$ е недетерминиран. Също така, в този пример се оказва, че вече s_2 е недостижимо състояние, но в общия случай не можем да го премахнем, защото може да има преходи влизащи в s_2.

Лема 2.3. Класът от автоматните езици е затворен относно операцията *обединение*.

Упътване. Нека са дадени автоматите:

- $\mathcal{N}_1 = \langle Q_1, \Sigma, s_1, \Delta_1, F_1 \rangle$, като $L(\mathcal{N}_1) = L_1$;
- $\mathcal{N}_2 = \langle Q_2, \Sigma, s_2, \Delta_2, F_2 \rangle$, като $L(\mathcal{N}_2) = L_2$.

Ще дефинираме автомата $\mathcal{N} = \langle Q, \Sigma, q_{\texttt{start}}, \Delta, F \rangle$, така че

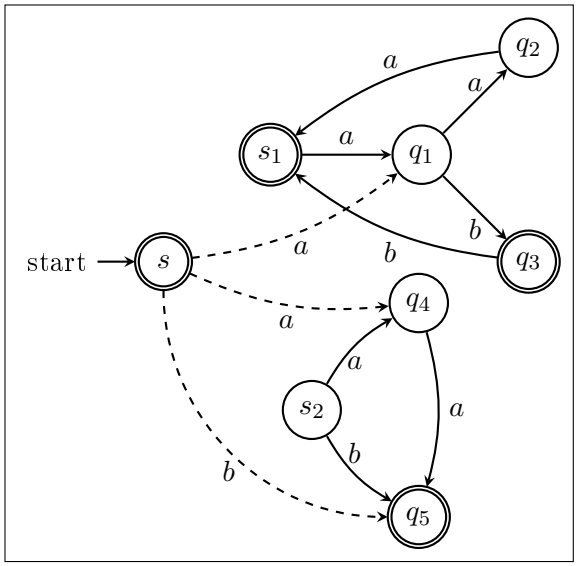
$$L(\mathcal{N}) = L(\mathcal{N}_1) \cup L(\mathcal{N}_2).$$

- $Q = Q_1 \cup Q_2 \cup \{s\}$;
- $F = \begin{cases} F_1 \cup F_2 \cup \{s\}, & \text{ако } s_1 \in F_1 \vee s_2 \in F_2 \\ F_1 \cup F_2, & \text{иначе} \end{cases}$
- $\Delta(q,a) = \begin{cases} \Delta_1(q,a), & \text{ако } q \in Q_1 \ \& \ a \in \Sigma \\ \Delta_2(q,a), & \text{ако } q \in Q_2 \ \& \ a \in \Sigma \\ \Delta_1(s_1,a) \cup \Delta_2(s_2,a), & \text{ако } q = s \ \& \ a \in \Sigma. \end{cases}$

□

Забележка. В началното състояние на новопостроения автомат $\mathcal{N}$ не влизат ребра.

Пример 2.6. За да построим автомат, който разпознава обединението на $\mathcal{L}(\mathcal{N}_1)$ и $\mathcal{L}(\mathcal{N}_2)$, трябва да добавим ново начално състояние, което да свържем с наследниците на началните състояния на $\mathcal{N}_1$ и $\mathcal{N}_2$.



Фигура 2.10: $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{N}_1) \cup \mathcal{L}(\mathcal{N}_2)$

Обърнете внимание, че $\mathcal{N}_1$ и $\mathcal{N}_2$ са детерминирани автомати, но $\mathcal{N}$ е недетерминиран. Освен това, новото състояние s трябва да бъде маркирано като финално, защото s_1 е финално.

Лема 2.4. Класът от автоматните езици е затворен относно операцията *звезда на Клини*, т.е. ако $\mathcal{L}(\mathcal{N}) = L$, то съществува $\hat{\mathcal{N}}$, за който $\mathcal{L}(\hat{\mathcal{N}}) = L^\star$.

Доказателство. Нека е даден автомата $\mathcal{N} = \langle Q, \Sigma, q_{\texttt{start}}, \Delta, F\rangle$, за който е изпънено, че $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathbf{r})$. Първата стъпка е да построим $\mathcal{N}_1 = \langle Q_1, \Sigma, s_1, \Delta_1, F_1\rangle$, такъв че

$$\mathcal{L}(\mathcal{N}_1) = \bigcup_{n\geq 1}(\mathcal{L}(\mathcal{N}))^n = \bigcup_{n\geq 1}(\mathcal{L}(\mathbf{r}))^n = \mathcal{L}(\mathbf{r}^+).$$

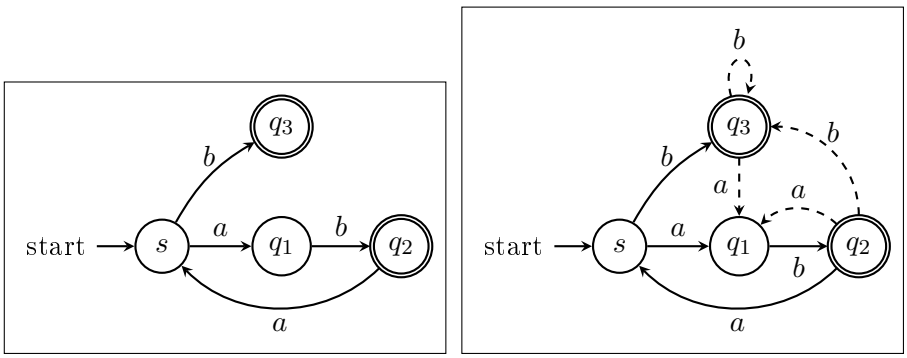
Това можем да направим по следния начин:

- $Q_1 = Q$;
- $s_1 = s$;
- $F_1 = F$;
- Функцията на преходите Δ_1 включва всичко от Δ като добавя нови преходи към наследниците на началното състояние s на $\mathcal{N}$. Дефиницията на Δ_1 може да се запише така:

$$\Delta_1(q,a) = \begin{cases} \Delta(q,a), & \text{ако } q \in Q \setminus F, a \in \Sigma \\ \Delta(q,a) \cup \Delta(s,a), & \text{ако } q \in F, a \in \Sigma. \end{cases}$$

Накрая строим желания автомат $\hat{\mathcal{N}}$, така че $\mathcal{L}(\hat{\mathcal{N}}) = \{\varepsilon\} \cup \mathcal{L}(\mathcal{N}_1)$. Ние знаем как да построим $\hat{\mathcal{N}}$, защото можем да посторим автомат $\mathcal{N}_0$ за езика $\{\varepsilon\}$ (*Лема 2.1*) и тогава $\hat{\mathcal{N}}$ се получава от конструкцията за обединение от *Лема 2.3* приложена върху $\mathcal{N}_0$ и $\mathcal{N}_1$. □

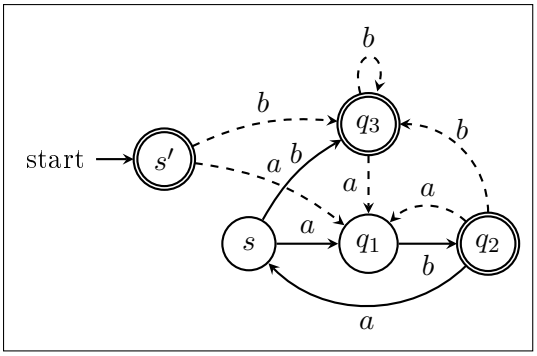
Пример 2.7. Нека да приложим конструкцията за да намерим автомат разпознаващ $\mathcal{L}(\mathcal{N})^\star$.



(а) автомат $\mathcal{N}$ (б) $\mathcal{L}(\mathcal{N}_1) = \mathcal{L}(\mathcal{N})^+$

След като построим автомат за езика $\mathcal{L}(\mathcal{N})^+$, трябва да приложим конструкцията за обединение на автомата за езика $\mathcal{L}(\mathcal{N})^+$ с автомата за езика $\{\varepsilon\}$. Защо трябва да добавим ново начално състояние s'? Да допуснем, че вместо това сме направили s финално. Тогава има опасност да разпознаем повече думи. Например, думата aba би се разпознала от този автомат, но $aba \notin \mathcal{L}(\mathcal{N})^\star$.

Лесно се вижда, че $\mathcal{L}(\mathcal{N}) = \{b\} \cup \{(aba)^n ab \mid n \in \mathbb{N}\}$



Фигура 2.12: $\mathcal{L}(\hat{\mathcal{N}}) = \mathcal{L}(\mathcal{N})^\star = \mathcal{L}(\mathcal{N})^+ \cup \{\varepsilon\}$

2.4 Лема за покачването

Лема 2.5 (за покачването)**.** Нека L да бъде *безкраен* регулярен език. Съществува число $p \geq 1$, зависещо само от L, за което за всяка дума $\alpha \in L, |\alpha| \geq p$ може да бъде записана във вида $\alpha = xyz$ и

На англ. се нарича Pumping Lemma

Има подобна лема и за безконтекстни езици

1) $|y| \geq 1$;

2) $|xy| \leq p$;

3) $(\forall i \in \mathbb{N})[xy^i z \in L]$.

Обърнете внимание, че $0 \in \mathbb{N}$ и $xy^0 z = xz$

Упътване. Понеже L е регулярен, то L е и автоматен език. Нека $\mathcal{A} = \langle Q, \Sigma, q_{\texttt{start}}, \delta, F\rangle$ е краен детерминиран автомат, за който $L = \mathcal{L}(\mathcal{A})$. Да положим $p = |Q|$ и нека $\alpha = a_1 a_2 \cdots a_k$ е дума, за която $k \geq p$. Да разгледаме първите p стъпки от изпълнението на α върху $\mathcal{A}$:

[PL98, стр. 88], [Sip97, стр. 78]

$$s \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots \xrightarrow{a_p} q_p.$$

Тъй като $|Q| = p$, а по този път участват $p+1$ състояния $q_0, q_1, \ldots, q_p$, то съществуват числа i, j, за които $0 \leq i < j \leq p$ и $q_i = q_j$. Нека разделим думата α на три части по следния начин:

$$x = a_1 \cdots a_i, \quad y = a_{i+1} \cdots a_j, \quad z = a_{j+1} \cdots a_k.$$

Ясно е, че $|y| \geq 1$ и $|xy| = j \leq p$. Освен това, лесно се съобразява,

✍ Докажете!

че за всяко $i \in \mathbb{N}$, $xy^iz \in L$. Да разгледаме случая за $i = 0$. Думата $xy^0z = xz \in L$, защото имаме следното изчисление:

$$s \underbrace{\xrightarrow{a_1} q_1 \cdots \xrightarrow{a_i}}_{x} q_i \underbrace{\xrightarrow{a_{j+1}} q_{j+1} \cdots \xrightarrow{a_k}}_{z} q_k \in F,$$

защото $q_i = q_j$. Да разгледаме и случая $i = 2$. Тогава думата $xy^2z \in L$, защото имаме следното изчисление:

$$s \underbrace{\xrightarrow{a_1} q_1 \cdots \xrightarrow{a_i}}_{x} q_i \underbrace{\xrightarrow{a_{i+1}} q_{i+1} \cdots \xrightarrow{a_j}}_{y} q_j \underbrace{\xrightarrow{a_{i+1}} q_{i+1} \cdots \xrightarrow{a_j}}_{y} q_j \underbrace{\xrightarrow{a_{j+1}} \cdots \xrightarrow{a_k}}_{z} q_k \in F.$$

□

Практически е по-полезно да разглеждаме следната еквивалентна формулировка на лемата за покачването.

Следствие 2.2 (Контрапозиция на лемата за покачването)**.** Нека L е произволен *безкраен* език. Нека също така е изпълнено, че за всяко естествено число $p \geq 1$ можем да намерим дума $\alpha \in L$, $|\alpha| \geq p$, такава че за всяко разбиване на думата на три части, $\alpha = xyz$, със свойствата $|y| \geq 1$ и $|xy| \leq p$, е изпълнено, че $(\exists i)[xy^iz \notin L]$. Тогава L **не** е регулярен език.

Ясно е, че всеки краен език е регулярен. Нали?

Доказателство. Да означим с (P) следната формула:

$$(\exists p \geq 1)(\forall \alpha \in L)[|\alpha| \geq p \Rightarrow (\exists x, y, z \in \Sigma^\star)[\alpha = xyz \ \wedge \ |y| \geq 1 \ \wedge \ |xy| \leq p \ \wedge \ (\forall i \in \mathbb{N})[xy^iz \in L]]].$$

Лемата за покачването представлява твърдението:

„Ако L е регулярен език, то е изпълнено свойството (P)."

Лемата може да се запише по следния еквивалентен начин:

Контрапозиция на твърдението $p \to q$ е твърдението $\neg q \to \neg p$

„Ако свойството (P) не е изпълнено, то L не е регулярен език."

Отрицанието на свойството (P) може да се запише по следния начин:

Използваме, че $\neg\exists\forall\exists\forall(\dots) \equiv \forall\exists\forall\exists\neg(\dots)$

$$(\forall p \geq 1)(\exists \alpha \in L)[|\alpha| \geq p \ \wedge (\forall x, y, z \in \Sigma^\star)[\alpha \neq xyz \ \vee \ |y| \not\geq 1 \ \vee \ |xy| \not\leq p \ \vee \ (\exists i \in \mathbb{N})[xy^iz \notin L]]].$$

Горната формула е еквивалентна на:

Използваме, че $\neg p \vee \neg q \vee r \equiv (p \wedge q) \to r$

$$(\forall p \geq 1)(\exists \alpha \in L)[|\alpha| \geq p \ \wedge \ (\forall x, y, z \in \Sigma^\star)[(\alpha = xyz \wedge |y| \geq 1 \wedge |xy| \leq p) \Rightarrow (\exists i \in \mathbb{N})[xy^iz \notin L]]].$$

Това означава, че ако

(∀) вземем произволна константа $p \geq 1$,

(∃) за нея намерим дума $\alpha \in L$, такава че $|\alpha| \geq p$ и

(∀) докажем, че за всяко нейно разбиване на три части x, y, z, със свойствата $|y| \geq 1$ и $|xy| \leq p$,

(∃) можем да намерим i, за което $xy^iz \notin L$,

то можем да заключим, че езикът L не е регулярен. □

Приложения на лемата за покачването

Задача 2.8. Докажете, че езикът $L = \{a^n b^n \mid n \in \mathbb{N}\}$ не е регулярен.

Доказателство. Ще докажем, че

$$(\forall p \geq 1)(\exists \alpha \in L)[|\alpha| \geq p \ \wedge \ (\forall x, y, z \in \Sigma^\star)[(\alpha = xyz \wedge |y| \geq 1 \wedge |xy| \leq p) \Rightarrow (\exists i \in \mathbb{N})[xy^i z \notin L]].$$

Това е важен пример. По-късно ще видим, че този език е безконтекстен

Доказателството следва стъпките:

(∀) Разглеждаме произволно число $p \geq 1$.

нямаме власт над избора на числото p

(∃) Избираме дума $\alpha \in L$, за която $|\alpha| \geq p$. Имаме свободата да изберем каквато дума α си харесаме, стига тя да принадлежи на L и да има дължина поне p. Щом имаме тази свобода, нека да изберем думата $\alpha = a^p b^p \in L$. Очевидно е, че $|\alpha| \geq p$.

Няма общо правило, което да ни казва как избираме думата α

Обърнете внимание, че думата α зависи от константата p

(∀) Разглеждаме произволно разбиване на α на три части, $\alpha = xyz$, за които изискваме свойствата $|xy| \leq p$ и $|y| \geq 1$.

не знаем нищо друго за x, y и z освен тези две свойства

(∃) Ще намерим $i \in \mathbb{N}$, за което $xy^i z \notin L$. Понеже $|xy| \leq p$, то $y = a^k$, за $1 \leq k \leq p$. Тогава ако вземем $i = 0$, получаваме $xy^0 z = a^{p-k} b^p$. Ясно е, че $xz \notin L$, защото $p - k < p$.

Тогава от *Следствие 2.2* следва, че L не е регулярен език. □

Забележка. Много често студентите правят следното разсъждение:

$$(\forall L, L' \subseteq \Sigma^\star)[L \text{ е регулярен } \& \ L' \subseteq L \implies L' \text{ е регулярен}].$$

Съобразете, че в общия случай това твърдение е невярно. За да видите това, достатъчно е да посочите регулярен език L, който има като подмножество нерегулярен език L'. Също лесно се вижда, че твърдението

$$(\forall L, L' \subseteq \Sigma^\star)[L \text{ е регулярен } \& \ L \subseteq L' \implies L' \text{ е регулярен}]$$

е невярно.

Задача 2.9. Докажете, че езикът $L = \{a^m b^n \mid m, n \in \mathbb{N} \ \& \ m < n\}$ не е регулярен.

Доказателство. Доказателството следва стъпките:

(∀) Разглеждаме произволно число $p \geq 1$.

(∃) Избираме дума $\alpha \in L$, за която $|\alpha| \geq p$. Имаме свободата да изберем каквато дума α си харесаме, стига тя да принадлежи на L и да има дължина поне p. Щом имаме тази свобода, нека да изберем думата $\alpha = a^p b^{p+1} \in L$. Очевидно е, че $|\alpha| \geq p$.

(∀) Разглеждаме произволно разбиване на α на три части, $\alpha = xyz$, за които изискваме свойствата $|xy| \leq p$ и $|y| \geq 1$.

(∃) Ще намерим $i \in \mathbb{N}$, за което $xy^iz \notin L$. Понеже $|xy| \leq p$, то $y = a^k$, за $1 \leq k \leq p$. Тогава ако вземем $i = 2$, получаваме

$$xy^2z = a^{p-k}a^{2k}b^{p+1} = a^{p+k}b^{p+1}.$$

Ясно е, че $xy^2z \notin L$, защото $p + k \geq p + 1$.

Тогава от *Следствие 2.2* следва, че L не е регулярен език. □

Задача 2.10. Докажете, че езикът $L = \{a^n \mid n \text{ е просто число}\}$ не е регулярен.

Доказателство. Доказателството следва стъпките:

(∀) Разглеждаме произволно число $p \geq 1$.

(∃) Избираме дума $w \in L$, за която $|w| \geq p$. Можем да изберем каквото w си харесаме, стига то да принадлежи на L и да има дължина поне p. Нека да изберем думата $w \in L$, такава че $|w| > p + 1$. Знаем, че такава дума съществува, защото L е безкраен език. По-долу ще видим защо този избор е важен за нашите разсъждения.

(∀) Разглеждаме произволно разбиване на w на три части, $w = xyz$, за които изискваме свойствата $|xy| \leq p$ и $|y| \geq 1$.

(∃) Ще намерим i, за което $xy^iz \notin L$, т.е. ще намерим i, за което $|xy^iz| = |xz| + i \cdot |y|$ е *съставно число*. Понеже $|xy| \leq p$ и $|xyz| > p+1$, то $|z| > 1$. Да изберем $i = |xz| > 1$. Тогава:

$$|xy^iz| = |xz| + i.|y| = |xz| + |xz|.|y| = (1 + |y|)|xz|$$

е съставно число, следователно $xy^iz \notin L$.

Тогава от *Следствие 2.2* следва, че L не е регулярен език. □

Задача 2.11. Докажете, че езикът $L = \{a^{n^2} \mid n \in \mathbb{N}\}$ не е регулярен.

Доказателство. В тази задача ще използваме следното свойство:

$$n \text{ не е точен квадрат } \leftrightarrow (\exists p \in \mathbb{N})[p^2 < n < (p+1)^2].$$

Доказателството следва стъпките:

(∀) Разглеждаме произволно число $p \geq 1$.

(∃) Избираме достатъчно дълга дума, която принадлежи на езика L. Например, нека $w = a^{p^2}$.

(∀) Разглеждаме произволно разбиване на w на три части, $w = xyz$, като $|xy| \leq p$ и $|y| \geq 1$.

(∃) Ще намерим i, за което $xy^iz \notin L$. В нашия случай това означава, че $|xz| + i \cdot |y|$ не е точен квадрат. Тогава за $i = 2$,

$$p^2 = |xyz| < |xy^2z| = |xyz| + |y| \leq p^2 + p < p^2 + 2p + 1 = (p+1)^2.$$

Получаваме, че $p^2 < |xy^2z| < (p+1)^2$, откъдето следва, че $|xy^2z|$ не е точен квадрат. Следователно, $xy^2z \notin L$.

Тогава от *Следствие 2.2* следва, че L не е регулярен език. □

Задача 2.12. Докажете, че езикът $L = \{a^{n!} \mid n \in \mathbb{N}\}$ не е регулярен.

Доказателство. Доказателството следва стъпките:

(∀) Разглеждаме произволно число $p \geq 1$.

(∃) Избираме достатъчно дълга дума, която принадлежи на езика L. Например, нека $\omega = a^{(p+1)!}$.

(∀) Разглеждаме произволно разбиване на ω на три части, $\omega = xyz$, като $|xy| \leq p$ и $|y| \geq 1$. Да обърнем внимание, че $1 \leq |y| \leq p$

(∃) Ще намерим i, за което $xy^iz \notin L$. Това означава да съществува n, за което $n! < |xy^iz| < (n+1)!$ Да разгледаме $i = 2$. Тогава:

$$(p+1)! < |xy^2z| = (p+1)! + |y| \leq (p+1)! + p < (p+1)! + (p+1)!(p+1) = (p+2)!$$

Тогава от *Следствие 2.2* следва, че L не е регулярен език. □

Следствия от лемата за покачването

Твърдение 2.5. Нека е даден автомата $\mathcal{A} = \langle Q, \Sigma, q_{\texttt{start}}, \delta, F \rangle$. Езикът $\mathcal{L}(\mathcal{A})$ е *непразен* е точно тогава, когато съдържа дума $\alpha, |\alpha| < |Q|$.

Доказателство. Ще разгледаме двете посоки на твърдението.

(⇒) Нека L е непразен език и нека $m = \min\{|\alpha| \mid \alpha \in L\}$. Ще докажем, че $m < |Q|$. За целта, да допуснем, че $m \geq |Q|$ и да изберем $\alpha \in L$, за която $|\alpha| = m$. Според *Лема 2.5*, съществува разбиване $xyz = \alpha$, такова че $xz \in L$. При положение, че $|y| \geq 1$, то $|xz| < m$, което е противоречие с минималността на m. Заключаваме, че нашето допускане е грешно. Тогава $m < |Q|$, откъдето следва, че съществува дума $\alpha \in L$ с $|\alpha| < |Q|$.

(⇐) Тази посока е тривиална. Ако L съдържа дума α, за която $|\alpha| < |Q|$, то е очевидно, че L е непразен език.

□

Следствие 2.3. Съществува алгоритъм, който проверява дали даден регулярен език е празен или не.

Следствие 2.4. Съществува алгоритъм, който определя дали два автомата $\mathcal{A}_1$ и $\mathcal{A}_2$ разпознават един и същ език.

$(L_1 \setminus L_2) \cup (L_2 \setminus L_1) = \emptyset$?

Твърдение 2.6. Регулярният език L, разпознаван от КДА $\mathcal{A}$, е *безкраен* точно тогава, когато съдържа дума $\alpha, |Q| \leq |\alpha| < 2|Q|$.

Доказателство. Да разгледаме двете посоки на твърдението.

($\Leftarrow$) Нека L е регулярен език, за който съществува дума α, такава че $|Q| \leq |\alpha| < 2|Q|$. Тогава от *Лема 2.5* следва, че съществува разбиване $\alpha = xyz$ със свойството, че за всяко $i \in \mathbb{N}$, $xy^iz \in L$. Следователно, L е безкраен, защото $|y| \geq 1$.

($\Rightarrow$) Нека L е безкраен език и да вземем *най-късата* дума $\alpha \in L$, за която $|\alpha| \geq 2|Q|$. Понеже L е безкраен, знаем, че такава дума съществува. Тогава отново по *Лема 2.5*, имаме следното разбиване на α:

$$\alpha = xyz,\ |xy| \leq |Q|,\ 1 \leq |y|,\ xz \in L.$$

Но понеже $|xyz| \geq 2|Q|$, а $1 \leq |y| \leq |Q|$, то $|xyz| > |xz| \geq |Q|$ и понеже избрахме $\alpha = xyz$ да бъде най-късата дума с дължина поне $2|Q|$, заключаваме, че $|Q| \leq |xz| < 2|Q|$ и $xz \in L$.

□

Следствие 2.5. Съществува алгоритъм, който проверява дали даден регулярен език е безкраен.

Примери, за които лемата не е приложима

Задача 2.13. Да се даде пример за език L, който **не** е регулярен, но удовлетворява условието на *Лема 2.5*.

Например,
$\{c\}^+ \cdot \{a^nb^n \mid n \in \mathbb{N}\} \cup \{a,b\}^\star$

Пример 2.8. Езикът $L = \{c^ka^nb^m \mid k,n,m \in \mathbb{N}\ \&\ k = 1 \implies m = n\}$ **не** е регулярен, но условието за покачване от *Лема 2.5* е изпълнено за него.

Доказателство. Да допуснем, че L е регулярен. Тогава ще следва, че

$$L_1 = L \cap ca^\star b^\star = \{ca^nb^n \mid n \in \mathbb{N}\}$$

е регулярен, но с лемата за разрастването лесно се вижда, че L_1 не е.

Сега да проверим, че условието за покачване от *Лема 2.5* е изпълнено за L. Да изберем константа $p = 2$. Сега трябва да разгледаме всички думи $\alpha \in L$, $|\alpha| \geq 2$ и за всяка α да посочим разбиване $xyz = \alpha$, за което са изпълнени трите свойства от лемата.

Условията за x, y, z са:

$|xy| \leq 2$
$|y| \geq 1$
$(\forall i \in \mathbb{N})(xy^iz \in L)$

- Ако $\alpha = a^n$ или $\alpha = b^n$, $n \geq 2$, то е очевидно, че можем да намерим такова разбиване.
- $\alpha = a^n b^m$ и $n + m \geq 2$, $n \geq 1$. Избираме $x = \varepsilon$, $y = a$, $z = a^{n-1}b^m$.
- $\alpha = ca^n b^n$, $n \geq 1$. Избираме $x = \varepsilon$, $y = c$, $z = a^n b^n$.
- $\alpha = c^2 a^n b^m$. Избираме $x = \varepsilon$, $y = c^2$, $z = a^n b^m$.
- $\alpha = c^k a^n b^m$, $k \geq 3$. Избираме $x = \varepsilon$, $y = c$, $z = c^{k-1} a^n b^m$.

□

2.5 Минимален автомат

Нека да започнем като въведем няколко означения. Нека α е дума над азбуката Σ и L е език. Означаваме

$$\alpha^{-1}L = \{\omega \in \Sigma^\star \mid \alpha\omega \in L\}.$$

Освен това, да означим

$$\alpha L = \{\alpha\omega \in \Sigma^\star \mid \omega \in L\}.$$

Имаме свойството, че

$$L = \{\omega \in \Sigma^\star \mid \varepsilon \in \omega^{-1}L\}.$$

Твърдение 2.7. За всеки две думи $\alpha, \beta \in \Sigma^\star$ е изпълнено, че

$$(\alpha \cdot \beta)^{-1}L = \beta^{-1}(\alpha^{-1}L).$$

2.5.1 Минимален автомат по даден регулярен език

Да разгледаме езика $L = \mathcal{L}(((\mathbf{a} + \mathbf{b})^+ \cdot \mathbf{a})^\star)$.

Удобно е да представим

$$L = \{\varepsilon\} \cup \{a, b\}^+ aL$$
$$= \{\varepsilon\} \cup a\{a, b\}^\star aL \cup b\{a, b\}^\star aL$$

Да видим как можем директно да построим минимален тотален детерминиран автомат $\mathcal{A}$ разпознаващ L, където

$$\mathcal{A} = \langle Q^{\mathcal{A}}, \Sigma, s, \delta_{\mathcal{A}}, F^{\mathcal{A}} \rangle.$$

Състоянията на автомата ще бъдат от вида q_B, където $B \subseteq \Sigma^\star$, така че накрая искаме да имаме свойството

$$B = \{\omega \in \Sigma^\star \mid \delta^\star_{\mathcal{A}}(q_B, \omega) \in F^{\mathcal{A}}\}.$$

Тогава началното състояние ще бъде q_L, защото

$$L = \mathcal{L}(\mathcal{A}) = \{\omega \in \Sigma^\star \mid \delta^\star_{\mathcal{A}}(q_L, \omega) \in F^{\mathcal{A}}\}.$$

Сега едновременно ще строим състоянията на автомата $Q^{\mathcal{A}}$ и функцията на преходите $\delta_{\mathcal{A}}$.

- $a^{-1}L = \{a,b\}^\star aL \stackrel{\text{деф}}{=} M$. Имаме, че $M \neq L$, защото $\varepsilon \in L$, но $\varepsilon \notin M$. Понеже $M \neq L$, имаме ново състояние q_M в автомата и $\delta_{\mathcal{A}}(q_L, a) = q_M$.
- $b^{-1}L = \{a,b\}^\star aL = M$; Следователно, $\delta_{\mathcal{A}}(q_L, b) = q_M$.
- $a^{-1}M = L \cup M \stackrel{\text{деф}}{=} N$. Имаме, че $N \neq L$, защото $a \in N$, но $a \notin L$. Освен това, $N \neq M$, защото $\varepsilon \in N$, но $\varepsilon \notin M$. Понеже $N \neq L, M$, имаме ново състояние q_N в автомата и $\delta_{\mathcal{A}}(q_M, a) = q_N$.
- $b^{-1}M = M$. Следователно, $\delta_{\mathcal{A}}(q_M, b) = q_M$.
- $a^{-1}N = L \cup M = N$. Следователно, $\delta_{\mathcal{A}}(q_N, a) = q_N$.
- $b^{-1}N = M$. Следователно, $\delta_{\mathcal{A}}(q_M, b) = q_M$.

Удобно е да представим

$$\begin{aligned} M &\stackrel{\text{деф}}{=} \{a,b\}^\star aL \\ &= aL \cup \{a,b\}^+ aL \\ &= aL \cup aM \cup bM \end{aligned}$$

Тогава

$$L = \{\varepsilon\} \cup aM \cup bM$$

Освен това,

$$\begin{aligned} N &\stackrel{\text{деф}}{=} L \cup M \\ &= \{\varepsilon\} \cup aM \cup bM \cup M \\ &= \{\varepsilon\} \cup aM \cup bM \cup aL \end{aligned}$$

От горните сметки следва, че

$$(\forall \omega \in \Sigma^\star)[\omega^{-1}L \in \{L, N, M\}].$$

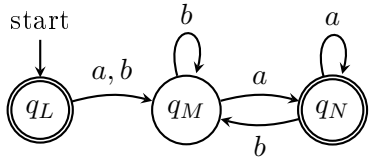
Така получихме, че

$$Q^{\mathcal{A}} = \{q_L, q_N, q_M\}.$$

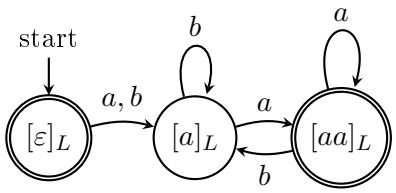
Нека сега да съобразим кои са финалните състояния. Понеже имаме свойството

$$L = \{\omega \in \Sigma^\star \mid \varepsilon \in \omega^{-1}L\},$$

то следва, че финалните състояния на автомата $\mathcal{A}$ са q_L и q_M, защото $\varepsilon \in L, M$. Сега вече сме готови да нарисуваме картинка на автомата.



(а) Минимален автомат за езика $\mathcal{L}(((\mathbf{a}+\mathbf{b})^+\mathbf{a})^\star)$



(б) Минимален автомат за езика $\mathcal{L}(((\mathbf{a}+\mathbf{b})^+\mathbf{a})^\star)$

Остава да се уверим, че нашата конструкция е коректна, т.е. наистина автоматът $\mathcal{A}$ е минимален за езика L. Според Теоремата на Майхил-Нероуд, състоянията на минималния автомат $\mathcal{M}$, разпознаващ L, са класовете на еквивалентност на релацията $\approx_L$. Да разгледаме изображението f с дефиниционна област $\Sigma^\star/_{\approx_L} = \{[\alpha]_L \mid \alpha \in \Sigma^\star\}$, дефинирано като:

$$f([\alpha]_L) = q_K, \text{ където } K = \alpha^{-1}L.$$

- Ако $[\alpha]_L \neq [\beta]_L$, то $\alpha^{-1}L \neq \beta^{-1}L$ и оттук следва, че $f([\alpha]_L) \neq f([\beta]_L)$. Това означава, че f е инективна.

- Да разгледаме $q_K \in Q^{\mathcal{A}}$. От конструкцията на автомата $\mathcal{A}$ следва, че съществува дума $\alpha \in \Sigma^\star$, за която $K = \alpha^{-1}L$. Това означава, че $f([\alpha]_L) = q_K$. Следователно, f е сюрективна.

- Имаме и свойството:

$$\begin{aligned} f(\delta_{\mathcal{M}}([\alpha]_L, x)) &= f([\alpha \cdot x]_L) && \text{(деф. на } \delta_{\mathcal{M}}) \\ &= q_N && (N = (\alpha \cdot x)^{-1}L = x^{-1}(\alpha^{-1}L)) \\ &= \delta_{\mathcal{A}}(q_M, x) && (M = \alpha^{-1}L) \\ &= \delta_{\mathcal{A}}(f([\alpha]_L), x), \end{aligned}$$

  от което следва, че f е биекция.

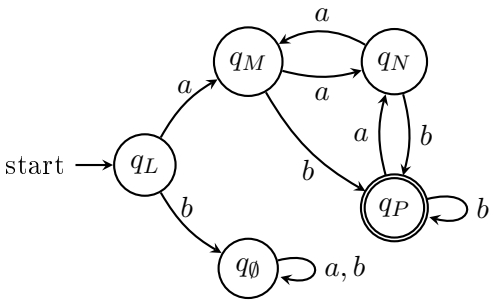
  Според горните разсъждения,

$$\begin{aligned} Q^{\mathcal{M}} &= \{f(q_L), f(q_M), f(q_N)\} = \{[\varepsilon]_L, [a]_L, [aa]_L\}, \\ F^{\mathcal{M}} &= \{f(q_L), f(q_N)\} = \{[\varepsilon]_L, [aa]_L\}. \end{aligned}$$

  Проверете, че наистина $L = [\varepsilon]_L \cup [aa]_L$.

Пример 2.9. Да разгледаме езика $L = \mathcal{L}(\mathbf{a} \cdot (\mathbf{a} + \mathbf{b})^\star \cdot \mathbf{b})$.

- $a^{-1}L = \{a, b\}^\star b = M$;
- $b^{-1}L = \emptyset$;
- $a^{-1}M = \{b\} \cup \{a, b\}^\star b = N$;
- $b^{-1}M = \{\varepsilon\} \cup \{b\} \cup \{a, b\}^\star b = P$;
- $a^{-1}N = \{b\} \cup \{a, b\}^\star b = M$;
- $b^{-1}N = \{\varepsilon\} \cup \{b\} \cup \{a, b\}^\star b = P$;
- $a^{-1}P = \{b\} \cup \{a, b\}^\star b = N$;
- $b^{-1}P = \{\varepsilon\} \cup \{b\} \cup \{a, b\}^\star b = P$.



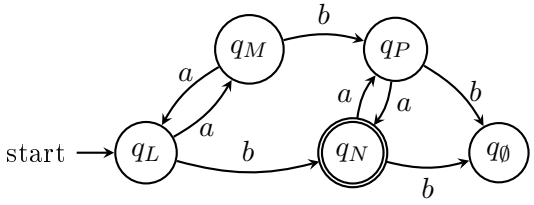
Фигура 2.14: Минимален автомат за $\mathcal{L}(\mathbf{a} \cdot (\mathbf{a} + \mathbf{b})^\star \cdot \mathbf{b})$

Пример 2.10. Да разгледаме езика

$$L = \{\omega \in \{a, b\}^\star \mid \omega \text{ съдържа четен брой } a \text{ и точно едно } b\}.$$

Нека да видим дали можем да построим автомат за този език.

- $a^{-1}L \stackrel{\text{деф}}{=} M$ е езика съставен от думите с нечетен брой a и точно едно b;
- $b^{-1}L \stackrel{\text{деф}}{=} N$ е езика съставен от думите с четен брой a и нито едно b;
- $a^{-1}M = L$;
- $b^{-1}M \stackrel{\text{деф}}{=} P$ е езика съставен от думите с нечетен брой a и нито едно b;
- $a^{-1}N = P$;
- $b^{-1}N = \emptyset$;
- $a^{-1}P = N$;
- $b^{-1}P = \emptyset$;



Фигура 2.15: Минимален автомат, който приема думи с четен брой a и точно едно b

Пример 2.11. Да разгледаме езика $L = \{a^n b^n \mid n \in \mathbb{N}\}$. Да се опитаме да построим автомат, който го разпознава. Нека

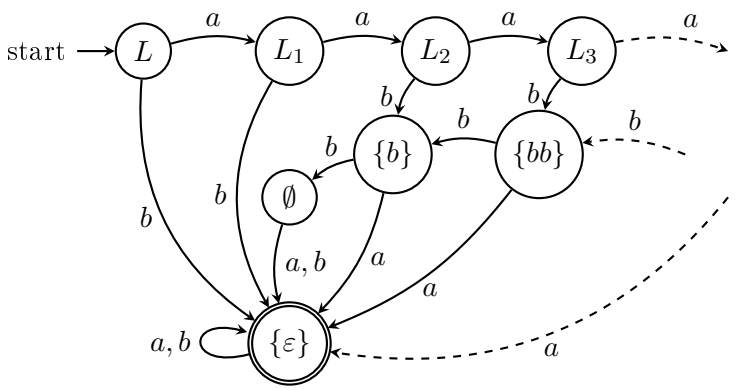
$$L_k \stackrel{\text{деф}}{=} \{a^n b^{n+k} \mid n \in \mathbb{N}\}.$$

Да видим какво се получава като приложим процедурата за строене на минимален автомат.

- $a^{-1}L = L_1$;
- $b^{-1}L = \emptyset$;
- $a^{-1}L_1 = L_2$;

- $b^{-1}L_1 = \{\varepsilon\}$;
- $a^{-1}\{\varepsilon\} = b^{-1}\{\varepsilon\} = \emptyset$;
- Вижда се, че $a^{-1}L_k = L_{k+1}$, за всяко k.
- Вижда се, че $b^{-1}L_{k+1} = \{b^k\}$, за всяко k. Освен това е ясно, че $b^{-1}\{b^k\} = \{b^{k-1}\}$, за всяко $k \geq 1$.

Получаваме, че езикът L се разпознава от автомат с *безкрайно много състояния*.



Фигура 2.16: Получаваме *безкраен* автомат за $\{a^n b^n \mid n \in \mathbb{N}\}$

2.5.2 Проверка за регулярност на език

Твърдение 2.8. Езикът L е регулярен точно тогава, когато релацията $\approx_L$ има *крайно много* класове на еквивалентност.

Доказателство. Ако L е регулярен, то той се разпознава от някой ДКА $\mathcal{A}$, който има крайно много състояния и следователно крайно много класове на еквивалентност относно $\sim_{\mathcal{A}}$. Релацията $\approx_L$ е по-груба от $\sim_{\mathcal{A}}$ и има по-малко класове на еквивалентност. Следователно, $\approx_L$ има крайно много класове на еквивалентност.

За другата посока, ако $\approx_L$ има крайно много класове на еквивалентност, то можем да построим ДКА $\mathcal{A}$ както в доказателството на Теоремата на Майхил-Нероуд, който разпознава L. □

Това следствие ни дава още един начин за проверка дали даден език е регулярен. За разлика от *Лема 2.5*, сега имаме **необходимо и достатъчно условие**. При даден език L, ние разглеждаме неговата релация $\approx_L$. Ако тя има крайно много класове, то езикът L е регулярен. В противен случай, езикът L не е регулярен.

Пример 2.12. За езика $L = \{a^n b^n \mid n \in \mathbb{N}\}$ имаме, че $|\approx_L| = \infty$, защото

$$(\forall k, j \in \mathbb{N})[k \neq j \implies [a^k b]_L \neq [a^j b]_L].$$

Проверете, че $[a^k b]_L = \{a^k b, a^{k+1}b^2, \ldots, a^{k+l}b^{l+1}, \ldots\}$. Така получаваме, че релацията $\approx_L$ има безкрайно много класове на еквивалентност. Заключаваме, че този език **не** е регулярен.

Пример 2.13. За езика $L = \{a^{n^2} \mid n \in \mathbb{N}\}$ имаме, че $|\approx_L| = \infty$, защото

$$(\forall m, n \in \mathbb{N})[m \neq n \implies [a^{n^2}]_L \neq [a^{m^2}]_L].$$

Без ограничение на общността, да разгледаме $n < m$ и думата $\gamma = a^{2n+1}$. Тогава $a^{n^2}\gamma = a^{(n+1)^2} \in L$, но $m^2 < m^2 + 2n + 1 < (m+1)^2$ и следователно $a^{m^2}\gamma = a^{m^2+2n+1} \notin L$.

Пример 2.14. За езика $L = \{a^{n!} \mid n \in \mathbb{N}\}$ имаме, че $|\approx_L| = \infty$, защото

$$(\forall m, n \in \mathbb{N})[m \neq n \implies [a^{n!}]_L \neq [a^{m!}]_L].$$

Без ограничение на общността, да разгледаме $n < m$ и думата $\gamma = a^{(n!)n}$. Тогава $a^{n!}\gamma = a^{(n+1)!} \in L$, но $m! < m! + (n!)n < m! + (m!)m = (m+1)!$ и следователно $a^{m!}\gamma = a^{m!+(n!)n} \notin L$.

Задача 2.14. Да разгледаме езика

$$L = \{a^{f_n} \mid f_0 = f_1 = 1 \ \&\ f_{n+2} = f_{n+1} + f_n\}.$$

Докажете, че $|\approx_L| = \infty$.

2.5.3 Релация на Майхил-Нероуд

- Нека $L \subseteq \Sigma^\star$ е език и нека $\alpha, \beta \in \Sigma^\star$. Казваме, че α и β са **еквивалентни относно** L, което записваме като $\alpha \approx_L \beta$, когато:

  $\approx_L$ е известна като релация на Майхил-Нероуд

  $$\alpha \approx_L \beta \overset{\text{деф}}{\leftrightarrow} \alpha^{-1}L = \beta^{-1}L.$$

  С други думи,

  $$\alpha \approx_L \beta \ \leftrightarrow\ (\forall \omega \in \Sigma^\star)[\alpha\omega \in L \ \leftrightarrow\ \beta\omega \in L].$$

- Нека $\mathcal{A} = \langle Q, \Sigma, q_{\texttt{start}}, \delta, F\rangle$ е ДКА. Казваме, че две думи $\alpha, \beta \in \Sigma^\star$ са **еквивалентни относно** $\mathcal{A}$, което означаваме с $\alpha \sim_{\mathcal{A}} \beta$, ако

  Трябва ли $\mathcal{A}$ да е тотален?

  $$\delta^\star(s, \alpha) = \delta^\star(s, \beta).$$

- Проверете, че $\approx_L$ и $\sim_{\mathcal{A}}$ са **релации на еквивалентност**, т.е. те са рефлексивни, транзитивни и симетрични.

- Класът на еквивалентност на думата α относно релацията $\approx_L$ означаваме като
$$[\alpha]_L \stackrel{\text{деф}}{=} \{\beta \in \Sigma^\star \mid \alpha \approx_L \beta\}.$$
Означаваме
$$\Sigma^\star/_{\approx_L} \stackrel{\text{деф}}{=} \{[\alpha]_L \mid \alpha \in \Sigma^\star\}.$$
Тогава с $|\Sigma^\star/_{\approx_L}|$ ще означаваме броя на класовете на еквивалентност на релацията $\approx_L$.

- Класът на еквивалентност на думата α относно релацията $\sim_{\mathcal{A}}$ означаваме като
$$[\alpha]_{\mathcal{A}} \stackrel{\text{деф}}{=} \{\beta \in \Sigma^\star \mid \alpha \sim_{\mathcal{A}} \beta\}.$$
Означаваме
$$\Sigma^\star/_{\sim_{\mathcal{A}}} \stackrel{\text{деф}}{=} \{[\alpha]_{\mathcal{A}} \mid \alpha \in \Sigma^\star\}.$$
С $|\Sigma^\star/_{\sim_{\mathcal{A}}}|$ ще означаваме броя на класовете на еквивалентност на релацията $\sim_{\mathcal{A}}$.

- Съобразете, че всяко състояние на $\mathcal{A}$, което е достижимо от началното състояние, определя клас на еквивалентност относно релацията $\sim_{\mathcal{A}}$. Това означава, че ако за всяка дума означим $q_\alpha = \delta^\star_{\mathcal{A}}(s, \alpha)$, то $\alpha \sim_{\mathcal{A}} \beta$ точно тогава, когато $q_\alpha = q_\beta$. Заключаваме, че броят на класовете на еквивалентност на $\sim_{\mathcal{A}}$ е равен на броя на достижимите от s състояния. Следователно,
$$|\Sigma^\star/_{\sim_{\mathcal{A}}}| \leq |Q^{\mathcal{A}}|.$$

- Релациите $\approx_{\mathcal{L}}$ и $\sim_{\mathcal{A}}$ са дясно-инвариантни, т.е. за всеки две думи α и β е изпълнено:
$$\begin{aligned} \alpha \sim_{\mathcal{A}} \beta &\implies (\forall \gamma \in \Sigma^\star)[\alpha\gamma \sim_{\mathcal{A}} \beta\gamma], \\ \alpha \approx_{\mathcal{L}} \beta &\implies (\forall \gamma \in \Sigma^\star)[\alpha\gamma \approx_{\mathcal{L}} \beta\gamma]. \end{aligned}$$

Твърдение 2.9. За всеки ДКА $\mathcal{A} = \langle Q, \Sigma, q_{\texttt{start}}, \delta, F\rangle$ е изпълнено:
$$(\forall \alpha, \beta \in \Sigma^\star)[\alpha \sim_{\mathcal{A}} \beta \implies \alpha \approx_{\mathcal{L}(\mathcal{A})} \beta].$$
С други думи, $[\alpha]_{\mathcal{A}} \subseteq [\alpha]_{\mathcal{L}(\mathcal{A})}$, за всяка дума $\alpha \in \Sigma^\star$.

Доказателство. Да означим за всяка дума α, $q_\alpha = \delta^\star_{\mathcal{A}}(s, \alpha)$. Лесно се съобразява, че за всеки две думи α и β имаме
$$\begin{aligned} \alpha \sim_{\mathcal{A}} \beta &\leftrightarrow \delta^\star(s, \alpha) = \delta^\star(s, \beta) \qquad (\text{по деф. на } \sim_{\mathcal{A}}) \\ &\leftrightarrow q_\alpha = q_\beta. \end{aligned}$$

Нека $\alpha \sim_{\mathcal{A}} \beta$. Ще проверим, че $\alpha \approx_{\mathcal{L}(\mathcal{A})} \beta$. За произволно $\gamma \in \Sigma^\star$ имаме:

$$\begin{aligned}
\alpha\gamma \in \mathcal{L}(\mathcal{A}) &\leftrightarrow \delta^\star(s, \alpha\gamma) \in F && \text{(по деф. на } \mathcal{L}(\mathcal{A})) \\
&\leftrightarrow \delta^\star(\delta^\star(s, \alpha), \gamma) \in F && \text{(по деф. на } \delta^\star) \\
&\leftrightarrow \delta^\star(q_\alpha, \gamma) \in F && (q_\alpha = \delta^\star(s, \alpha)) \\
&\leftrightarrow \delta^\star(q_\beta, \gamma) \in F && (q_\alpha = q_\beta, \text{ защото } \alpha \sim_{\mathcal{A}} \beta) \\
&\leftrightarrow \delta^\star(\delta^\star(s, \beta), \gamma) \in F && (q_\beta = \delta^\star(s, \beta)) \\
&\leftrightarrow \delta^\star(s, \beta\gamma) \in F && \text{(по деф. на } \delta^\star) \\
&\leftrightarrow \beta\gamma \in \mathcal{L}(\mathcal{A}) && \text{(по деф. на } \mathcal{L}(\mathcal{A})).
\end{aligned}$$

Заключаваме, че

$$(\forall \alpha, \beta \in \Sigma^\star)[\alpha \sim_{\mathcal{A}} \beta \implies \alpha \approx_{\mathcal{L}(\mathcal{A})} \beta].$$

□

Задача 2.15. Докажете, че за всеки тотален ДКА $\mathcal{A}$ и всяка дума α,

$$[\alpha]_{\mathcal{L}(\mathcal{A})} = \bigcup_{\beta \in [\alpha]_{\mathcal{L}(\mathcal{A})}} [\beta]_{\mathcal{A}}.$$

Следствие 2.6. За всеки тотален ДКА $\mathcal{A}$ е изпълнено, че

$$|\Sigma^\star/_{\approx_{\mathcal{L}(\mathcal{A})}}| \leq |\Sigma^\star/_{\sim_{\mathcal{A}}}|.$$

Упътване. Да означим $L = \mathcal{L}(\mathcal{A})$ и да разгледаме изображението

$$f([\alpha]_L) \stackrel{\text{деф}}{=} \{[\gamma]_{\mathcal{A}} \mid \gamma \approx_L \alpha\}.$$

- Ясно е, че за всяко α, $f([\alpha]_L) \neq \emptyset$.
- Съобразете, че f е **функция**, т.е.

$$(\forall \alpha, \beta \in \Sigma^\star)[\alpha \approx_L \beta \implies f([\alpha]_L) = f([\beta]_L)].$$

- Използвайте *Твърдение 2.9* за да съобразите, че

$$(\forall \alpha, \beta \in \Sigma^\star)[\alpha \not\approx_L \beta \implies f([\alpha]_L) \cap f([\beta]_L) = \emptyset].$$

- Заключете, че

$$|\Sigma^\star/_{\approx_{\mathcal{L}(\mathcal{A})}}| \leq |\Sigma^\star/_{\sim_{\mathcal{A}}}|.$$

□

Следствие 2.7. Нека L е произволен регулярен език L. Всеки тотален ДКА $\mathcal{A}$, който разпознава L има свойството

$$|\Sigma^\star/_{\approx_L}| \leq |Q|,$$

т.е. броят на класовете на еквивалентност на релацията $\approx_L$ не надвишава броя на състоянията на автомата.

Доказателство. Да изберем $\mathcal{A}$, който разпознава L, бъде такъв, че да *няма недостижими състояния*. Тъй като всяко достижимо състояние определя клас на еквивалентност относно $\sim_\mathcal{A}$, то получаваме, че $|Q| = |\sim_\mathcal{A}|$. Комбинирайки със *Следствие 2.6*,

$$|Q| = |\Sigma^\star/_{\sim_\mathcal{A}}| \geq |\Sigma^\star/_{\approx_L}|.$$

□

Така получаваме *долна граница* за броя на състоянията в тотален автомат разпознаващ езика L. Този брой е не по-малък от броя на класовете на еквивалентност на $\approx_L$. В следващия раздел ще видим, че тази долна граница може да бъде достигната.

2.5.4 Теорема за съществуване на минимален автомат

Теорема 2.3 (Майхил-Нероуд). Нека $L \subseteq \Sigma^\star$ е регулярен език. Тогава съществува ДКА $\mathcal{A} = \langle Q, \Sigma, q_{\texttt{start}}, \delta, F\rangle$, който разпознава L, с точно толкова състояния, колкото са класовете на еквивалентност на релацията $\approx_L$, т.е. $|Q| = |\Sigma^\star/_{\approx_L}|$.

на англ. Myhill-Nerode

Доказателство. Да фиксираме регулярния език L. Ще дефинираме тотален ДКА $\mathcal{A} = \langle Q, \Sigma, q_{\texttt{start}}, \delta, F\rangle$, разпознаващ L, като:

- $Q = \{[\alpha]_L \mid \alpha \in \Sigma^\star\}$;
- $s = [\varepsilon]_L$;
- $F = \{[\alpha]_L \mid \alpha \in L\} = \{[\alpha]_L \mid [\alpha]_L \cap L \neq \emptyset\}$;
- Определяме изображението δ като за всяка буква $x \in \Sigma$ и всяко състояние $[\alpha]_L \in Q$,
$$\delta([\alpha]_L, x) = [\alpha x]_L.$$

Първо, трябва да се уверим, че множеството от състояния Q е крайно, т.е. релацията $\approx_L$ има крайно много класове на еквивалентност. И така, тъй като L е регулярен език, то той се разпознава от някой тотален ДКА $\mathcal{A}'$. От *Следствие 2.7* имаме, че $|Q^{\mathcal{A}'}| \geq |\Sigma^\star/_{\approx_L}|$. Понеже $Q^{\mathcal{A}'}$ е крайно множество, то $\approx_L$ има крайно много класове и следователно Q също е крайно множество.

Второ, трябва да се уверим, че изображението δ задава функция, т.е. да проверим, че за всеки две думи α, β и всяка буква x,

$$[\alpha]_L = [\beta]_L \implies \delta([\alpha]_L, x) = \delta([\beta]_L, x).$$

Но това се вижда веднага, защото от определението на релацията $\approx_L$ следва, че ако $\alpha \approx_L \beta$, то за всяка буква x, $\alpha x \approx_L \beta x$, т.е. $[\alpha x]_L = [\beta x]_L$ и

$$\begin{aligned} [\alpha]_L = [\beta]_L &\implies [\alpha x]_L = [\beta x]_L && \text{(свойство на } \approx_L) \\ &\implies \delta([\alpha]_L, x) = [\alpha x]_L = [\beta x]_L = \delta([\beta]_L, x) && \text{(деф. на } \delta) \end{aligned}$$

Така вече сме показали, че $\mathcal{A}$ е коректно зададен тотален ДКА. Остава да покажем, че $\mathcal{A}$ разпознава езика L, т.е. $\mathcal{L}(\mathcal{A}) = L$. За целта, първо ще докажем едно помощно твърдение.

Твърдение 2.10. За всеки две думи α и β, $\delta^\star([\alpha]_L, \beta) = [\alpha\beta]_L$.

Доказателство. Ще докажем това свойство с индукция по дължината на β.

- За $\beta = \varepsilon$ свойството следва директно от дефиницията на $\delta^\star$ като рефлексивно и транзитивно затваряне на δ, защото $\delta^\star([\alpha]_L, \varepsilon) = [\alpha]_L$.
- Нека $|\beta| = n + 1$ и да приемем, че сме доказали твърдението за думи с дължина n. Тогава $\beta = \gamma a$, където $|\gamma| = n$. Свойството следва от следните равенства:

$$\begin{aligned} \delta^\star([\alpha]_L, \gamma a) &= \delta(\delta^\star([\alpha]_L, \gamma), a) && \text{(деф. на } \delta^\star) \\ &= \delta([\alpha\gamma]_L, a) && \text{(от } \textbf{И.П.} \text{ за } \gamma) \\ &= [\alpha\gamma a]_L && \text{(от деф. на } \delta) \\ &= [\alpha\beta]_L && (\beta = \gamma a). \end{aligned}$$

□

За да се убедим, че $L = \mathcal{L}(\mathcal{A})$ е достатъчно да проследим еквивалентностите:

$$\begin{aligned} \alpha \in \mathcal{L}(\mathcal{A}) &\leftrightarrow \delta^\star(s, \alpha) \in F && \text{(от деф. на } \mathcal{L}(\mathcal{A})) \\ &\leftrightarrow \delta^\star([\varepsilon]_L, \alpha) \in F && \text{(по деф. } s = [\varepsilon]_L) \\ &\leftrightarrow \delta^\star([\varepsilon]_L, \alpha) = [\alpha]_L \ \&\ \alpha \in L && \text{(от деф. на } F) \\ &\leftrightarrow \alpha \in L && \text{(от последното твърдение).} \end{aligned}$$

□

Определение 2.3. Нека $\mathcal{A}_1 = \langle Q_1, \Sigma, s_1, \delta_1, F_1 \rangle$ и $\mathcal{A}_2 = \langle Q_2, \Sigma, s_2, \delta_2, F_2 \rangle$. Казваме, че $\mathcal{A}_1$ и $\mathcal{A}_2$ са **изоморфни**, което означаваме с $\mathcal{A}_1 \cong \mathcal{A}_2$, ако съществува биекция $f : Q_1 \to Q_2$, за която:

- $f(s_1) = s_2$;
- $q \in F_1 \leftrightarrow f(q) \in F_2$;
- $(\forall a \in \Sigma)(\forall q \in Q_1)[f(\delta_1(q,a)) = \delta_2(f(q),a)]$.

Ще казваме, че f задава изоморфизъм на $\mathcal{A}_1$ върху $\mathcal{A}_2$.

Това означава, че два автомата $\mathcal{A}_1$ и $\mathcal{A}_2$ са изоморфни, ако можем да получим $\mathcal{A}_2$ като преименуваме състоянията на $\mathcal{A}_1$.

Следствие 2.8. Нека е даден регулярния език L. Всички минимални автомати за L са изоморфни на $\mathcal{A}_0$, автоматът построен в Теоремата на Майхил-Нероуд.

Доказателство. Нека $\mathcal{A} = \langle Q, \Sigma, q_{\texttt{start}}, \delta, F\rangle$ е произволен тотален автомат, за който $\mathcal{L}(\mathcal{A}) = L$ и $|Q| = |\Sigma^\star/_{\approx_L}|$. Съобразете, че $\mathcal{A}$ е *свързан*, т.е. всяко състояние на $\mathcal{A}$ е достижимо от началното. Искаме да докажем, че $\mathcal{A} \cong \mathcal{A}_0$. Понеже $\mathcal{A}$ е свързан, за всяко състояние q можем да намерим дума ω_q, за която $\delta^\star(s,\omega_q) = q$. Да дефинираме изображението $f : Q \to \Sigma^\star/_{\approx_L}$ като $f(q) = [\omega_q]_L$. Ще докажем, че f задава изоморфизъм на $\mathcal{A}$ върху $\mathcal{A}_0$.

- Първо да съобразим, че ако $\delta^\star_{\mathcal{A}}(s,\alpha) = q$, то $[\omega_q]_L = [\alpha]_L$. Понеже $\delta^\star_{\mathcal{A}}(s,\alpha) = q = \delta^\star_{\mathcal{A}}(s,\omega_q)$, то $\omega_q \sim_{\mathcal{A}} \alpha$. От *Твърдение 2.9* имаме, че

  $$\omega_q \sim_{\mathcal{A}} \alpha \implies \omega_q \approx_L \alpha.$$

  Това означава, $[\omega_q]_L = [\alpha]_L$ и следователно f е определена коректно, т.е. f е **функция**.

- Ще проверим, че f е **инективна**, т.е.

  $$(\forall q_1, q_2 \in Q)[q_1 \neq q_2 \implies f(q_1) \neq f(q_2)].$$

  Да допуснем, че има състояния $q_1 \neq q_2$, за които

  $$f(q_1) = [\omega_{q_1}]_L = [\omega_{q_2}]_L = f(q_2).$$

  Тогава $\omega_{q_1} \not\sim_{\mathcal{A}} \omega_{q_2}$ и $\omega_{q_1} \approx_L \omega_{q_2}$. Но тогава от *Следствие 2.7* получаваме, че $|\Sigma^\star/_{\sim_{\mathcal{A}}}| > |\Sigma^\star/_{\approx_L}|$, което противоречи с минималността на $\mathcal{A}$.

  ✍ Обяснете!

- За да бъде f **сюрективна** трябва за всеки клас $[\beta]_L$ да съществува състояние q, за което $f(q) = [\beta]_L$. Понеже $\mathcal{A}$ е свързан, съществува състояние q, за което $\delta^\star_{\mathcal{A}}(s,\beta) = q$. Вече се убедихме, че в този случай $\beta \approx_L \omega_q$, защото $\beta \sim_{\mathcal{A}} \omega_q$. Тогава $f(q) = [\omega_q]_L = [\beta]_L$.

- За последно оставихме проверката, че f наистина е **изоморфизъм**:

$$\begin{aligned}
f(\delta_{\mathcal{A}}(q,a)) &= f(\delta_{\mathcal{A}}(\delta^{\star}_{\mathcal{A}}(s,\omega_q),a)) && \text{(от избора на } \omega_q) \\
&= f(\delta^{\star}_{\mathcal{A}}(s,\omega_q a)) && \text{(от деф. на } \delta^{\star}_{\mathcal{A}}) \\
&= [\omega_q a]_L && \text{(от деф. на } f) \\
&= \delta^{\star}_{\mathcal{A}_0}([\varepsilon]_L,\omega_q a) && \text{(от деф. на } \mathcal{A}_0) \\
&= \delta_{\mathcal{A}_0}(\delta^{\star}_{\mathcal{A}_0}([\varepsilon]_L,\omega_q),a) && \text{(от деф. на } \delta^{\star}_{\mathcal{A}_0}) \\
&= \delta_{\mathcal{A}_0}([\omega_q]_L,a) && \text{(свойство на } \delta^{\star}_{\mathcal{A}_0}) \\
&= \delta_{\mathcal{A}_0}(f(q),a) && (f(q)=[\omega_q]_L).
\end{aligned}$$

□

2.5.5 Алгоритъм за строене на минимален автомат

- Да фиксираме произволен тотален ДКА $\mathcal{A} = \langle Q, \Sigma, q_{\texttt{start}}, \delta, F\rangle$.
- За състояние p в автомата $\mathcal{A}$, да означим с $\mathcal{L}_{\mathcal{A}}(p)$ езикът, който се разпознава от автомата $\mathcal{A}$, ако приемем, че p е началното състояние на автомата, т.е.

$$\mathcal{L}_{\mathcal{A}}(p) \stackrel{\text{деф}}{=} \{\omega \in \Sigma^{\star} \mid \delta^{\star}(p,\omega) \in F\}.$$

  В частност, $\mathcal{L}(\mathcal{A}) = \mathcal{L}_{\mathcal{A}}(s)$.
- Сега дефинираме следната релация между състояния на автомата $\mathcal{A}$:

$$p \equiv_{\mathcal{A}} q \stackrel{\text{деф}}{\leftrightarrow} \mathcal{L}_{\mathcal{A}}(p) = \mathcal{L}_{\mathcal{A}}(q).$$

  Това означава, че $p \equiv_{\mathcal{A}} q$ точно тогава, когато

$$(\forall \omega \in \Sigma^{\star})[\delta^{\star}(p,\omega) \in F \quad \leftrightarrow \quad \delta^{\star}(q,\omega) \in F]. \tag{2.5}$$

- Релацията $\equiv_{\mathcal{A}}$ между състояния на автомата $\mathcal{A}$ е релация на еквивалентност.
- Нека q_α е състоянието, което съответства на думата α в $\mathcal{A}$, т.е. $\delta^{\star}_{\mathcal{A}}(s,\alpha) = q_\alpha$. Тогава

$$\mathcal{L}_{\mathcal{A}}(q_\alpha) = \alpha^{-1}\mathcal{L}(\mathcal{A}).$$

  Оттук получаваме, че

$$\begin{aligned}
q_\alpha \equiv_{\mathcal{A}} q_\beta &\leftrightarrow \mathcal{L}_{\mathcal{A}}(q_\alpha) = \mathcal{L}_{\mathcal{A}}(q_\beta) \\
&\leftrightarrow \alpha^{-1}\mathcal{L}(\mathcal{A}) = \beta^{-1}\mathcal{L}(\mathcal{A}) \\
&\leftrightarrow \alpha \sim_{\mathcal{L}(\mathcal{A})} \beta.
\end{aligned}$$

  Това означава, че ако в $\mathcal{A}$ няма недостижими състояния от началното състояние s, то

$$|\Sigma^{\star}/_{\equiv_{\mathcal{A}}}| = |\Sigma^{\star}/_{\approx_{\mathcal{L}(\mathcal{A})}}|.$$

При даден език L и тотален ДКА $\mathcal{A} = \langle Q, \Sigma, q_{\texttt{start}}, \delta, F\rangle$, който го разпознава, нашата цел е да построим нов ДКА $\mathcal{A}_0$, който има толкова състояния колкото са класовете на еквивалентност на релацията $\approx_{\mathcal{L}}$. Това ще направим като "слеем" състоянията на $\mathcal{A}$, които са еквивалентни относно релацията $\equiv_{\mathcal{A}}$. Това означава, че всяко състояние на $\mathcal{A}_0$ ще отговаря на един клас на еквивалентност на релацията $\equiv_{\mathcal{A}}$. Проблемът с намирането на класовете на еквивалентност на релацията $\equiv_{\mathcal{A}}$ е кванторът $\forall\gamma \in \Sigma^\star$ във нейната дефиниция чрез (Формула 2.5), защото $\Sigma^\star$ е безкрайно множество от думи.

Да фиксираме автомата $\mathcal{A}$ и $L = \mathcal{L}(\mathcal{A})$. Да означим

$$\mathcal{L}^n_{\mathcal{A}}(p) \stackrel{\text{деф}}{=} \{\omega \in \Sigma^\star \mid |\omega| \leq n \ \&\ \delta^\star(p, \omega) \in F\}.$$

Според тази дефиниция, $L = \bigcup_{n\geq 0} \mathcal{L}^n_{\mathcal{A}}(s)$.

За всяко естествено число n, дефинираме бинарните релации $\equiv_n$ върху Q по следния начин:

$$p \equiv_n q \stackrel{\text{деф}}{\leftrightarrow} \mathcal{L}^n_{\mathcal{A}}(p) = \mathcal{L}^n_{\mathcal{A}}(q).$$

Релациите $\equiv_n$ представляват апроксимации на релацията $\equiv_{\mathcal{A}}$. Обърнете внимание, че за всяко n, $\equiv_n$ е *по-груба* релация от $\equiv_{n+1}$, която на свой ред е по-груба от $\equiv_{\mathcal{A}}$. Алгоритъмът строи $\equiv_n$ докато не срещнем n, за което $\equiv_n = \equiv_{n+1}$. Тъй като броят на класовете на еквивалентност на $\equiv_{\mathcal{A}}$ е краен (той е $\leq |Q|$), то със сигурност ще намерим такова n, за което $\equiv_n = \equiv_{n+1}$. Тогава заключаваме, че $\equiv_{\mathcal{A}} = \equiv_n$.

Понеже единствената дума с дължина 0 е ε и по определение $\delta^\star(p, \varepsilon) = p$, лесно се съобразява, че $\equiv_0$ има два класа на еквивалентност. Единият е F, а другият е $Q \setminus F$.

Твърдение 2.11. За всеки две състояния $p, q \in Q$, и всяко n, $p \equiv_{n+1} q$ точно тогава, когато

а) $p \equiv_n q$ и

б) $(\forall a \in \Sigma)[\delta(q, a) \equiv_n \delta(p, a)]$.

Упътване. [PL98, стр. 99]

$$\begin{aligned} p \equiv_{n+1} q &\leftrightarrow \mathcal{L}^{n+1}_{\mathcal{A}}(p) = \mathcal{L}^{n+1}_{\mathcal{A}}(q) \\ &\leftrightarrow \mathcal{L}^{n}_{\mathcal{A}}(p) = \mathcal{L}^{n}_{\mathcal{A}}(q) \ \&\ (\forall a \in \Sigma)[\mathcal{L}^n_{\mathcal{A}}(\delta(p, a)) = \mathcal{L}^n_{\mathcal{A}}(\delta(q, a))] \\ &\leftrightarrow p \equiv_n q \ \&\ (\forall a \in \Sigma)[\delta(p, a) \equiv_n \delta(q, a)]. \end{aligned}$$

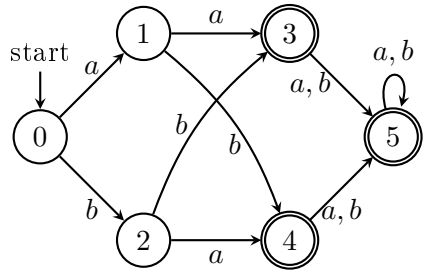
□

Нека е даден автомата $A = \langle Q, \Sigma, q_{\texttt{start}}, \delta, F\rangle$. След като сме намерили релацията $\equiv_{\mathcal{A}}$ за $\mathcal{A}$, строим автомата $\mathcal{A}' = (Q', \Sigma, s', \delta', F')$, където:

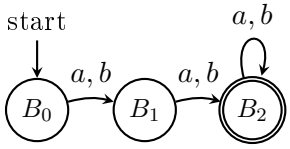
- $Q' = \{[q]_{\equiv_\mathcal{A}} \mid q \in Q\}$;
- $s' = [s]_{\equiv_\mathcal{A}}$;
- $\delta'([q]_{\equiv_\mathcal{A}}, a) = [\delta(q, a)]_{\equiv_\mathcal{A}}$;
- $F' = \{[q]_{\equiv_\mathcal{A}} \mid F \cap [q]_{\equiv_\mathcal{A}} \neq \emptyset\}$;

От всичко казано дотук знаем, че $\mathcal{A}'$ е минимален автомат разпознаващ езика $\mathcal{L}(\mathcal{A})$.

Пример 2.15. Да разгледаме следния краен детерминиран автомат $\mathcal{A}$.



(а) Ще построим минимален автомат разпознаващ $\mathcal{L}(\mathcal{A})$



(б) Получаваме следния минимален автомат $\mathcal{A}_0$, $\mathcal{L}(\mathcal{A}_0) = \mathcal{L}(\mathcal{A})$

Съобразете, че $\mathcal{L}(\mathcal{A}) = \{\alpha \in \{a, b\}^\star \mid |\alpha| \geq 2\}$.

Ще приложим алгоритъма за минимизация за да получим минималния автомат за езика L. За всяко $n = 0, 1, 2, \ldots$, ще намерим класовете на еквивалентност на $\equiv_n$, докато не намерим n, за което $\equiv_n = \equiv_{n+1}$.

- Класовете на еквивалентност на $\equiv_0$ са два. Те са $A_0 = Q \setminus F = \{0, 1, 2\}$ и $A_1 = F = \{3, 4, 5\}$.
- Сега да видим дали можем да разбием някои от класовете на еквивалентност на $\equiv_0$.

Q	0	1	2	$3^\star$	$4^\star$	$5^\star$
$\equiv_0$	A_0	A_0	A_0	A_1	A_1	A_1
a	A_0	A_1	A_1	A_1	A_1	A_1
b	A_0	A_1	A_1	A_1	A_1	A_1

  Виждаме, че $0 \not\equiv_1 1$ и $1 \equiv_1 2$. Класовете на еквивалентност на $\equiv_1$ са $B_0 = \{0\}$, $B_1 = \{1, 2\}$, $B_2 = \{3, 4, 5\}$.
- Сега да видим дали можем да разбием някои от класовете на еквивалентност на $\equiv_1$.

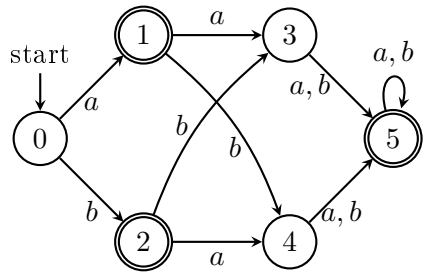
Q	0	1	2	$3^\star$	$4^\star$	$5^\star$
$\equiv_1$	B_0	B_1	B_1	B_2	B_2	B_2
a	B_1	B_2	B_2	B_2	B_2	B_2
b	B_1	B_2	B_2	B_2	B_2	B_2

Виждаме, че $\equiv_1 = \equiv_2$. Следователно, минималният автомат има три състояния. Той е изобразен на Фигура 2.17б. Минималният автомат може да се представи и таблично:

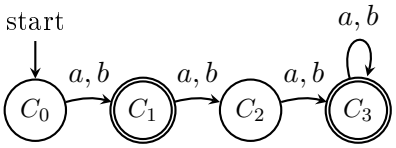
Получаваме, че $\equiv_{\mathcal{A}} = \equiv_1$

δ	B_0	B_1	B_2
a	B_1	B_2	B_2
b	B_1	B_2	B_2

Пример 2.16. Да разгледаме следния краен детерминиран автомат $\mathcal{A}$.



(а) Ще построим минимален автомат разпознаващ $\mathcal{L}(\mathcal{A})$



(б) Получаваме следния минимален автомат $\mathcal{A}_0$, $\mathcal{L}(\mathcal{A}_0) = \mathcal{L}(\mathcal{A})$

Съобразете, че $\mathcal{L}(\mathcal{A}) = \{a,b\} \cup \{\alpha \in \{a,b\}^\star \mid |\alpha| \geq 3\}$.

Отново следваме същата процедура за минимизация. Ще намерим класовете на еквивалентност на $\equiv_n$, докато не намерим n, за което $\equiv_n = \equiv_{n+1}$.

- Класовете на екиваленост на $\equiv_0$ са $A_0 = Q \setminus F = \{0,3,4\}$ и $A_1 = F = \{1,2,5\}$.
- Разбиваме класовете на еквивалентност на $\equiv_0$ като използваме *Твърдение 2.11*.

Q	0	$1^\star$	$2^\star$	3	4	$5^\star$
$\equiv_0$	A_0	A_1	A_1	A_0	A_0	A_1
a	A_1	A_0	A_0	A_1	A_1	A_1
b	A_1	A_0	A_0	A_1	A_1	A_1

Виждаме, че $1 \not\equiv_1 5$ и $1 \equiv_0 5$. Следователно, $\equiv_0 \neq \equiv_1$. Класовете на еквивалентност на $\equiv_1$ са $B_0 = \{0,3,4\}$, $B_1 = \{1,2\}$, $B_2 = \{5\}$.

- Сега се опитваме да разбием класовете на еквивалентност на $\equiv_1$.

Q	0	$1^\star$	$2^\star$	3	4	$5^\star$
$\equiv_1$	B_0	B_1	B_1	B_0	B_0	B_2
a	B_1	B_0	B_0	B_2	B_2	B_2
b	B_1	B_0	B_0	B_2	B_2	B_2

Имаме, че $0 \equiv_1 3$, но $0 \not\equiv_2 3$. Следователно $\equiv_1 \neq \equiv_2$. Класовете на еквивалентност на $\equiv_2$ са $C_0 = \{0\}$, $C_1 = \{1,2\}$, $C_2 = \{3,4\}$, $C_3 = \{5\}$.

- Отново опитваме да разбием класовете на $\equiv_2$.

Q	0	$1^\star$	$2^\star$	3	4	$5^\star$
$\equiv_2$	C_0	C_1	C_1	C_2	C_2	C_3
a	C_1	C_2	C_2	C_3	C_3	C_3
b	C_1	C_2	C_2	C_3	C_3	C_3

Виждаме, че не можем да разбием C_1 или C_2. Следователно, $\equiv_2 = \equiv_3$ и минималният автомат разпознаващ езика L има четири състояния. Вижте Фигура 2.18б за преходите на минималния автомат. Минималният автомат може да се представи и таблично:

Получаваме, че $\equiv_{\mathcal{A}} = \equiv_2$

δ	C_0	C_1	C_2	C_3
a	C_1	C_2	C_3	C_3
b	C_1	C_2	C_3	C_3

2.6 Регулярни граматики

Неограничена граматика е наредена четворка от вида

$$G = (V, \Sigma, R, S),$$

където

- V е крайно множество от *променливи* (нетерминали);
- Σ е крайно множество от *букви* (терминали), $\Sigma \cap V = \emptyset$;
- $R \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^\star$ е крайно множество от *правила*. Обикновено правилата $(\alpha, \beta) \in R$ ще означаваме като $\alpha \to_G \beta$, където $\alpha \in (V \cup \Sigma)^+, \beta \in (V \cup \Sigma)^\star$;

В [HU79] правилата се наричат *productions*

- $S \in V$ е началната променлива (нетерминал).

Казваме, че имаме **извод** $\omega \to_G \omega'$, ако $\omega = \alpha\beta\gamma \in (V \cup \Sigma)^\star$, $\omega' = \alpha\beta'\gamma \in (V \cup \Sigma)^\star$ и имаме правило $\beta \to \beta'$ в граматиката G. Нека $\to_G^\star$ е рефлексивното и транзитивно затваряне на релацията $\to_G$, т.е.

$$\begin{aligned} &\alpha \to_G^\star \alpha \\ &\alpha \to_G^\star \alpha' \ \&\ \alpha' \to_G \alpha'' \implies \alpha \to_G^\star \alpha''. \end{aligned}$$

Езикът, който се поражда от граматиката G е

$$\mathcal{L}(G) = \{\omega \in \Sigma^\star \mid S \to_G^\star \omega\}.$$

Граматиките се разделят на няколко вида в зависимост от това какви *ограничения* налагаме върху правилата R. В следващите няколко глави ще разгледаме различни ограничения. Сега ще разгледаме граматики с

такъв вид правила, които пораждат точно регулярните (или еквивалентно автоматни) езици.

Граматиката $G = (V, \Sigma, R, S)$ се нарича **регулярна граматика**, ако правилата са от вида

$$A \to \varepsilon,$$
$$A \to aB,$$

където $A, B \in V$ и $a \in \Sigma$.

Твърдение 2.12. Нека $G = \langle V, \Sigma, R, S \rangle$ е регулярна граматика и $L = \mathcal{L}(G)$. Съществува краен автомат $\mathcal{A}$, такъв че $L = \mathcal{L}(\mathcal{A})$.

Упътване. Нека $V = \{A_1, \ldots, A_k\}$. Тогава:

- $Q = \{q_1, \ldots, q_k\}$;
- $F = \{q_i \mid A_i \to \varepsilon\}$.
- $\delta(q_i, a) = q_j \quad \leftrightarrow \quad A_i \to aA_j$.

□

Твърдение 2.13. Нека $\mathcal{A}$ е краен автомат и $L = \mathcal{L}(\mathcal{A})$. Съществува регулярна граматика G, такава че $L = \mathcal{L}(G)$.

Упътване. Нека $Q = \{q_1, \ldots, q_k\}$. Тогава:

- $V = \{A_1, \ldots, A_k\}$;
- $A_i \to aA_j \quad \leftrightarrow \quad \delta(q_i, a) = q_j$;
- $A_i \to \varepsilon \quad \leftrightarrow \quad q_i \in F$.

□

2.7 Допълнителни задачи

2.7.1 Лесни задачи

Задача 2.16. Да разгледаме следната програма на езика си++

Библиотеката **regex** е част от C++11 стандарта на езика.

```
#include <iostream>
#include <regex>
#include <string>

using namespace std;

int main() {
```

```
    string input;
    regex reg("...................");
    while (true) {
      cout << "Input:" << endl;
      cin >> input;
      if (!cin || input=="q") break;
      if (regex_match(input, reg)) {
        cout << "Valid input" << endl;
      }
      else {
        cout << "Invalid input" << endl;
      }
    }
  }
```

Разучете как се създават обекти от тип **regex** и попълнете дефиницията на регулярния израз **reg** в горната програма, така че програмата приема за валиден вход:

а) само реални числа;

б) факултетни номера във ФМИ;

в) низове, които са съставени от поне 8 символа, измежду които се включват малки букви, големи букви, и цифри.

Задача 2.17. Да фиксираме една дума α над дадена азбука Σ. Опишете алгоритъм, който за вход произволен текстов файл, чието съдържание означаваме с β, отговаря дали думата α се среща в β. Каква е сложността на този алгоритъм относно дължините на α и β ?

Задача 2.18. Опишете алгоритъм, който при вход два регулярни израза $\mathbf{r}$ и $\mathbf{s}$, проверява дали $\mathcal{L}(\mathbf{r}) \subseteq \mathcal{L}(\mathbf{s})$.

Задача 2.19. За всеки от следните езици L, постройте автомат $\mathcal{A}$, който разпознава езика L.

а) $L = \{a^n b \mid n \geq 0\}$;

б) $L = \{a, b\}^\star \setminus \{\varepsilon\}$;

в) $L = \{w \in \{a, b\}^\star \mid a \text{ и } b \text{ се срещат четен брой пъти в } w\}$;

г) $L = \{w \in \{a, b\}^\star \mid a \text{ се среща четен брой, докато } b \text{ нечетен брой пъти в } w\}$;

д) $L = \{a^n b^m \mid n, m \geq 0\}$;

е) $L = \{a^n b^m \mid n, m \geq 1\}$;

ж) $L = \{a, b\}^\star \setminus \{a\}$;

з) $L = \{w \in \{a, b\}^\star \mid$ съдържа поне две a или не повече от три $b\}$;

и) $L = \{w \in \{a, b\}^\star \mid$ съдържа поне две a и поне едно $b\}$;

к) $L = \{w \in \{a, b\}^\star \mid$ на всяка нечетна позиция на w е буквата $a\}$;

л) $L = \{w \in \{a, b\}^\star \mid w$ съдържа четен брой a и най-много едно $b\}$;

м) $L = \{w \in \{a, b\}^\star \mid |w| \leq 3\}$; $|w|$ = дължина на думата w

н) $L = \{w \in \{a, b\}^\star \mid w$ не започва с $ab\}$;

о) $L = \{w \in \{a, b\}^\star \mid w$ завършва с ab или ba$\}$;

п) $L = \{w \in \{a, b\}^\star \mid w$ започва или завършва с буквата $a\}$;

р) $L = \{w \in \{a, b\}^\star \mid w$ започва с a точно тогава, когато завършва с $b\}$;

с) $L = \{\omega \in \{a, b\}^\star \mid |\omega| \equiv 0 \pmod 2$ & ω съдържа точно едно $a\}$;

т) $L = \{w \in \{a, b\}^\star \mid$ всяко a в w се следва от поне едно $b\}$;

у) $L = \{w \in \{a, b\}^\star \mid |w| \equiv 0 \bmod 3\}$;

ф) $L = \{w \in \{a, b\}^\star \mid N_a(w) \equiv 1 \bmod 3\}$; $N_a(w)$ - броят на срещанията на буквата a в думата w

х) $L = \{\omega \in \{a, b\}^\star \mid N_a(\omega) \equiv 0 \bmod 3$ & $N_b(\omega) \equiv 1 \bmod 2\}$;

ц) $L = \{\omega \in \{a, b\}^\star \mid N_a(\omega) \equiv 0 \bmod 2 \ \vee \ \omega$ съдържа точно две $b\}$;

ч) $L = \{\omega \in \{a, b\}^\star \mid \omega$ съдържа равен брой срещания на ab и на $ba\}$.

ш) $L = \{\omega_1 \sharp \omega_2 \sharp \omega_3 \mid \forall i \in [1, 3](\omega_i \in \{a, b\}^\star$ & $|\omega_i| \geq i + 1)\}$;

Задача 2.20. Докажете, че следните езици са регулярни:

а) $L = \{\alpha \in \{a, b\}^\star \mid |N_a(\omega) - N_b(\omega)| \leq 2$ за всяка представка ω на $\alpha\}$;

б) $L = \{\alpha \in \{a, b\}^\star \mid |N_a(\omega) - N_b(\omega)| > 2$ за някоя представка ω на $\alpha\}$;

в) $L = \{\alpha \in \{a, b\}^\star \mid |N_a(\omega) - N_b(\omega)| > 2$ за някоя наставка ω на $\alpha\}$.

Задача 2.21. Нека $\Sigma = \{a, b\}$. Проверете дали L е регулярен, където

а) $L = \{\alpha^R \mid \alpha \in L_0\}$, където L_0 е регулярен;

б) $L = \{a^i b^i \mid i \in \mathbb{N}\}$; $\alpha = a^p b^p$

в) $L = \{a^i b^i \mid i, j \in \mathbb{N}$ & $i \neq j\}$;

г) $L = \{a^i b^j \mid i > j\}$; $\alpha = a^{p+1} b^p$.

д) $L = \{a^n b^m \mid n \text{ дели } m\}$.

е) $L = \{a^{2n} \mid n \geq 1\}$;

ж) $L = \{a^m b^n a^{m+n} \mid m \geq 1 \ \& \ n \geq 1\}$;

з) $L = \{a^{n.m} \mid n, m \text{ са прости числа}\}$;

и) $L = \{\omega \in \{a,b\}^\star \mid N_a(\omega) = N_b(\omega)\}$;

$N_x(\omega)$ - брой срещания на буквата x в думата ω

к) $L = \{\omega\omega \mid \omega \in \{a,b\}^\star\}$;

$\alpha = a^p b a^p b$

л) $L = \{\omega\omega^R \mid \omega \in \{a,b\}^\star\}$;

м) $L = \{\alpha\beta\beta \in \{a,b\}^\star \mid \beta \neq \varepsilon\}$;

н) $L = \{a^n b^n c^n \mid n \geq 0\}$;

о) $L = \{\omega\omega\omega \mid \omega \in \Sigma^\star\}$;

п) $L = \{a^{2^n} \mid n \geq 0\}$;

р) $L = \{a^m b^n \mid n \neq m\}$;

с) $L = \{a^{n!} b^{n!} \mid n \neq 1\}$;

т) $L = \{a^{f_n} \mid f_0 = f_1 = 1 \ \& \ f_{n+2} = f_{n+1} + f_n\}$;

у) $L = \{\alpha \in \{a,b\}^\star \mid |N_a(\alpha) - N_b(\alpha)| \leq 2\}$;

ф) $L = \{\alpha \in \{a,b\}^\star \mid \alpha = \alpha\beta\alpha \ \& \ |\beta| \leq |\alpha|\}$;

х) $L = \{\alpha \in \{a,b\}^\star \mid \alpha = \beta\gamma\gamma^R \ \& \ |\beta| \leq |\gamma|\}$;

ц) $L = \{c^k a^n b^m \mid k, m, n > 0 \ \& \ n \neq m\}$;

ч) $L = \{c^k a^n b^n \mid k > 0 \ \& \ n \geq 0\} \cup \{a,b\}^\star$;

ш) $L = \{\omega \in \{a,b\}^\star \mid N_a(\omega) \text{ не дели } N_b(\omega)\}$;

щ) $L = \{\omega \in \{a,b\}^\star \mid N_a(\omega) < N_b(\omega)\}$;

ю) $L = \{\omega \in \{a,b\}^\star \mid N_a(\omega) = 2N_b(\omega)\}$;

я) $L = \{\omega \in \{a,b\}^\star \mid |N_a(\omega) - N_b(\omega)| \leq 3\}$.

Задача 2.22. Нека $\Sigma = \{a,b,c,d\}$. Да се докаже, че езикът

$$L = \{a_1 a_2 \cdots a_{2n} \in \Sigma^\star \mid (\forall j \in [1,n])[a_{2j-1} = a_{2j}] \ \& \ d \text{ се среща} \leq 3 \text{ пъти}\}$$

е регулярен.

Задача 2.23. Нека L_1 и L_2 са регулярни езици. Докажете, че L също е регулярен език, където

$$L = \{\alpha \mid (\exists \beta, \gamma)[\beta\alpha\gamma \in L_1] \ \& \ \alpha \in L_2 \vee \alpha^R \in L_2\}.$$

Определение 2.4. Да фиксираме две азбуки Σ_1 и Σ_2. Хомоморфизъм е изображение $h : \Sigma_1^\star \to \Sigma_2^\star$ със свойството, че за всеки две думи $\alpha, \beta \in \Sigma_1^\star$,

$$h(\alpha\beta) = h(\alpha) \cdot h(\beta).$$

Лесно се съобразява, че за всеки хомоморфизъм h, $h(\varepsilon) = \varepsilon$.

Задача 2.24. Нека $L \subseteq \Sigma_1^\star$ е регулярен език и $h : \Sigma_1^\star \to \Sigma_2^\star$ е хомоморфизъм. Тогава $h(L) = \{h(\alpha) \in \Sigma_2^\star \mid \alpha \in L\}$ е регулярен.

Упътване. Индукция по построението на регулярни езици. □

Задача 2.25. Нека $L \subseteq \Sigma_2^\star$ е регулярен език и $h : \Sigma_1^\star \to \Sigma_2^\star$ е хомоморфизъм. Тогава езикът $h^{-1}(L) = \{\alpha \in \Sigma_1^\star \mid h(\alpha) \in L\}$ е регулярен.

Упътване. Конструкция на автомат за $h^{-1}(L)$ при даден автомат за L. □

2.7.2 Не толкова лесни задачи

Задача 2.26. Докажете, че няма полиномиален алгоритъм за детерминизация на краен недетерминиран автомат.

Упътване. За произволно n, разгледайте недетерминирания автомат $\mathcal{A}_n$, за който $(\forall \alpha, \beta \in \{0,1\}^\star)[|\alpha| = |\beta| = n \implies (\alpha\beta \in \mathcal{L}(\mathcal{A}_n) \leftrightarrow \alpha \neq \beta)]$. Този автомат ще има $2n + 2$ състояния.

Допуснете, че за него съществува детерминиран автомат $\mathcal{D}_n$ с $< 2^n$ на брой състояния. Разгледайте всички думи с дължина n, $\omega_1, \omega_2, \ldots, \omega_{2^n}$. Приложете принципа на Дирихле и достигнете до противоречие. □

Задача 2.27. При дадени езици L, L' над азбуката Σ, да разгледаме: [PL98, стр. 84]

а) $\texttt{Pref}(L) = \{\alpha \in \Sigma^\star \mid (\exists \beta \in \Sigma^\star)[\alpha\beta \in L]\}$;

б) $\texttt{NoPref}(L) = \{\alpha \in L \mid$ не съществува префикс на α в $L\}$;

в) $\texttt{NoExtend}(L) = \{\alpha \in L \mid \ \alpha$ не е префикс на никоя дума от $L\}$;

г) $\mathrm{Suf}(L) = \{\beta \in \Sigma^\star \mid (\exists \alpha \in \Sigma^\star)[\alpha\beta \in L]\}$;

д) $\mathrm{Infix}(L) = \{\alpha \mid (\exists \beta, \gamma \in \Sigma^\star)[\beta\alpha\gamma \in L]\}$;

е) $\frac{1}{2}(L) = \{\omega \in \Sigma^\star \mid (\exists \alpha \in \Sigma^\star)[\omega\alpha \in L \ \& \ |\omega| = |\alpha|]\}$;

ж) $L/L' = \{\alpha \in \Sigma^\star \mid (\exists \beta \in L')[\alpha\beta \in L]\}$;

з) $\mathrm{Max}(L) = \{\alpha \in \Sigma^\star \mid (\forall \beta \in \Sigma^\star)[\beta \neq \varepsilon \implies \alpha\beta \notin L]\}$.

За всички тези езици, докажете, че са регулярни при условие, че L и L' са регулярни. Освен това, докажете, че L/L' е регулярен и при условието, че L е регулярен, но L' е произволен език над азбуката Σ.

Тази конструкция няма да бъде ефективна

Упътване.

а) Индукция по дефиницията на регулярен израз.

в) Най-лесно е да се построи автомат за $\mathrm{Infix}(L)$ като се използва автомата за L.

г) Конструкция с автомат за L и автомат за L^R.

□

Задача 2.28. Да фиксираме азбука само с един символ $\Sigma = \{a\}$. Да положим за всяко $p, q \in \mathbb{N}$,

[Koz97, стр. 75]; [PL98, стр. 89]

$$L(p,q) = \{a^k \mid (\exists n \in \mathbb{N})[k = p + q \cdot n]\}.$$

Ако за един език L съществуват константи $p_1, \ldots, p_k$ и $q_1, \ldots, q_k$, такива че

$$L = \bigcup_{1 \leq i \leq k} L(p_i, q_i),$$

то казваме, че L е *породен от аритметични прогресии*.

а) Докажете, че $L \subseteq \{a\}^\star$ е регулярен език точно тогава, когато L е породен от аритметична прогресия.

б) За произволна азбука Σ, докажете, че ако $L \subseteq \Sigma^\star$ е регулярен език, то езикът $\{a^{|\omega|} \mid \omega \in L\}$ е породен от аритметична прогресия.

Упътване.

а) За едната посока, разгледайте КДА за L.

б) За втората част, разгледайте $h : \Sigma \to \{a\}$ деф. като $(\forall b \in \Sigma)[h(b) = a]$. Докажете, че h е поражда хомоморфизъм между $\Sigma^\star$ и $\{a\}^\star$. Тогава $h(L) = \{a^{|\omega|} \mid \omega \in L\}$, а ние знаем, че регулярните езици са затворени относно хомоморфни образи.

□

Задача 2.29. Вярно ли е, че:

- $\{a^m \mid a^{m^2} \in L(p,q)\}$ е регулярен език ?
- $\{a^m \mid a^{2^m} \in L(p,q)\}$ е регулярен език ?

Задача 2.30. За даден език L над азбуката Σ, да разгледаме езиците:

а) $L' = \{\alpha \in \Sigma^\star \mid (\exists \beta \in \Sigma^\star)[|\alpha| = 2|\beta| \ \& \ \alpha\beta \in L]\}$;

б) $L'' = \{\alpha \in \Sigma^\star \mid (\exists \beta \in \Sigma^\star)[2|\alpha| = |\beta| \ \& \ \alpha\beta \in L]\}$;

в) $\frac{1}{3}(L) = \{\alpha \in \Sigma^\star \mid (\exists \beta, \gamma \in \Sigma^\star)[|\alpha| = |\beta| = |\gamma| \ \& \ \alpha\beta\gamma \in L]\}$;

г) $\frac{2}{3}(L) = \{\beta \in \Sigma^\star \mid (\exists \beta, \gamma \in \Sigma^\star)[|\alpha| = |\beta| = |\gamma| \ \& \ \alpha\beta\gamma \in L]\}$;

д) $\frac{3}{3}(L) = \{\gamma \in \Sigma^\star \mid (\exists \beta, \gamma \in \Sigma^\star)[|\alpha| = |\beta| = |\gamma| \ \& \ \alpha\beta\gamma \in L]\}$;

е) $\hat{L} = \{\alpha\gamma \in \Sigma^\star \mid (\exists \beta, \gamma \in \Sigma^\star)[|\alpha| = |\beta| = |\gamma| \ \& \ \alpha\beta\gamma \in L]\}$;

ж) $\sqrt{L} = \{\alpha \mid (\exists \beta \in \Sigma^\star)[|\beta| = |\alpha|^2 \ \& \ \alpha\beta \in L]\}$;

з) $\log(L) = \{\alpha \mid (\exists \beta \in \Sigma^\star)[|\beta| = 2^{|\alpha|} \ \& \ \alpha\beta \in L]\}$;

Проверете ако L е регулярен, то кои от горните езици също са регулярни.

Задача 2.31. Да разгледаме езика [Sip97, стр. 90]

$$L = \{\omega \in \{0,1\}^\star \mid \omega \text{ съдържа равен брой поднизове 01 и 10}\}.$$

Например, $101 \in L$, защото съдържа по веднъж 10 и 01. $1010 \notin L$, защото съдържа два пъти 10 и само веднъж 01. Докажете, че L е регулярен.

Задача 2.32. Нека L е регулярен език над азбуката $\{a, b\}$. Докажете, че следните езици са регулярни:

а) $\texttt{Diff}_1(L) \stackrel{\text{деф}}{=} \{\alpha \in L \mid (\exists \beta \in L)[|\alpha| = |\beta| \,\&\, \alpha \text{ се различава от } \beta \text{ в една позиция}]\}$;

б) $\texttt{Diff}_n(L) \stackrel{\text{деф}}{=} \{\alpha \in L \mid (\exists \beta \in L)[n \leq |\alpha| = |\beta| \,\&\, \alpha \text{ се различава от } \beta \text{ в } n \text{ позиции}]\}$;

в) $\texttt{Diff}(L) \stackrel{\text{деф}}{=} \{\alpha \in L \mid (\exists \beta \in L)[|\alpha| = |\beta| \,\&\, \alpha \text{ се различава от } \beta \text{ във всяка позиция}]\}$;

Упътване. Ако $L = \mathcal{L}(\mathcal{A})$, то правим декартово произведение на $\mathcal{A}$ плюс флаг дали сме направили грешка.

Не е ли по-лесно с индукция по построението на регулярните езици ? □

Задача 2.33. Да разгледаме азбуката:

$$\Sigma_3 = \left\{ \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \ldots, \begin{bmatrix}1\\1\\1\end{bmatrix} \right\}.$$

Докажете, че $L = \left\{ \begin{bmatrix}\alpha\\\beta\\\gamma\end{bmatrix} \in \Sigma_3^\star \mid \alpha_{(2)} + \beta_{(2)} = \gamma_{(2)} \right\}$ е автоматен език.

Упътване. По-удобно е да построим автомат $\mathcal{A}$, $\mathcal{L}(\mathcal{A}) = L^R$. Да започнем с състоянието $q_=$, за което искаме да имаме свойството, че за произволно състояние q,

$$\delta^\star(q, \begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix}) = q_= \ \leftrightarrow \ \alpha^R_{(2)} + \beta^R_{(2)} = \gamma^R_{(2)}.$$

Понеже за $\varepsilon + \varepsilon = \varepsilon$, състоянието $q_=$ ще бъде начално и финално за $\mathcal{A}$.

- Нека $\alpha_{(2)} + \beta_{(2)} = \gamma_{(2)}$. Тогава:
  - $0\alpha + 0\beta = 0\gamma$; $\delta(q_=, \begin{bmatrix}0\\0\\0\end{bmatrix}) = q_=$
  - $0\alpha + 1\beta = 1\gamma$; $\delta(q_=, \begin{bmatrix}0\\0\\0\end{bmatrix}) = q_=$
  - $1\alpha + 0\beta = 1\gamma$; $\delta(q_=, \begin{bmatrix}1\\0\\1\end{bmatrix}) = q_=$
  - $1\alpha + 1\beta = 10\gamma$. Този случай е по-специален и трябва да бъде разгледан отделно. Трябва да отидем в състояние q_1, в което ще помним, че третия ред трябва да започва с 1-ца. $\delta(q_=, \begin{bmatrix}1\\1\\0\end{bmatrix}) = q_1$
  - За останалите $x \in \Sigma_3$, $\delta(q_=, x) = q_{err}$, където q_{err} е състоянието, от което не можем да излезем.

- Горните разглеждания ни подсказват, че ще ни трябва и състояние q_1, за което искаме да е изпълнено свойството, че за произволно q,

$$\delta^\star(q, \begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix}) = q_1 \ \leftrightarrow \ \alpha^R_{(2)} + \beta^R_{(2)} = 1\gamma^R_{(2)}.$$

  Да разгледаме сега случая $\alpha + \beta = 1\gamma$. Тогава:
  - Очевидно е, че $0\alpha + 0\beta = 1\gamma$; $\delta(q_1, \begin{bmatrix}0\\0\\1\end{bmatrix}) = q_=$
  - $1\alpha + 1\beta = 11\gamma$; $\delta(q_1, \begin{bmatrix}1\\1\\1\end{bmatrix}) = q_1$
  - $1\alpha + 0\beta = 10\gamma$; $\delta(q_1, \begin{bmatrix}1\\0\\0\end{bmatrix}) = q_1$
  - Аналогично, $0\alpha + 1\beta = 10\gamma$; $\delta(q_1, \begin{bmatrix}0\\1\\0\end{bmatrix}) = q_1$
  - За останалите $x \in \Sigma_3$, $\delta(q_1, x) = q_{err}$.

□

Задача 2.34. Да разгледаме азбуката:

$$\Sigma_2 = \left\{ \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix} \right\}.$$

Една дума над азбуката Σ_2 ни дава два реда от 0-ли и 1-ци, които ще разглеждаме като числа в двоична бройна система. Да разгледаме езиците:

- $L_1 = \{\omega \in \Sigma_2^\star \mid$ долният ред на ω е по-голямо число от горния ред$\}$;
- $L_2 = \{\omega \in \Sigma_2^\star \mid$ долният ред на ω е три пъти по-голямо число от горния$\}$;
- $L_3 = \{\omega \in \Sigma_2^\star \mid$ долният ред на ω е обратния низ на горния ред$\}$.

Докажете, че L_1 и L_2 са автоматни, а L_3 не е автоматен.

Упътване. Ще построим автомат $\mathcal{A} = \langle Q, \Sigma, q_{\texttt{start}}, \delta, F\rangle$ за езика L_1^R. За улеснение, в рамките на тази задача ще пишем:

- $\alpha \equiv \beta$, ако $(\alpha^R)_{(2)} = (\beta^R)_{(2)}$,
- $\alpha < \beta$, ако $(\alpha^R)_{(2)} < (\beta^R)_{(2)}$,
- $\alpha > \beta$, ако $(\alpha^R)_{(2)} > (\beta^R)_{(2)}$.

Нека състоянията на автомата са $Q = \{q_=, q_<, q_>\}$. Искаме да е изпълнено свойствата:

- $\delta^\star(q_=, \begin{bmatrix}\alpha\\ \beta\end{bmatrix}) = q_=$ точно тогава, когато $\alpha \equiv \beta$;
- $\delta^\star(q_=, \begin{bmatrix}\alpha\\ \beta\end{bmatrix}) = q_<$ точно тогава, когато $\alpha < \beta$;
- $\delta^\star(q_=, \begin{bmatrix}\alpha\\ \beta\end{bmatrix}) = q_>$ точно тогава, когато $\alpha > \beta$.

Множеството от финални състояния ще бъде $F = \{q_<\}$, а началното състояние $s = q_=$. За да дефинираме функцията на преходите, трябва да разгледа няколко случая, в зависимост от това какво е отношението между α и β.

- Нека $\alpha \equiv \beta$. Тогава:
  - $\alpha 0 \equiv \beta 0$, $\alpha 1 \equiv \beta 1$;
  - $\alpha 0 < \beta 1$;
  - $\alpha 1 < \beta 0$;
- Нека $\alpha < \beta$. Тогава:
  - $\alpha 0 < \beta 0$, $\alpha 1 < \beta 1$, $\alpha 0 < \beta 1$;
  - $\alpha 1 > \beta 0$;
- Нека $\alpha > \beta$. Тогава:
  - $\alpha 0 > \beta 0$, $\alpha 1 > \beta 1$, $\alpha 1 > \beta 0$;
  - $\alpha 0 < \beta 1$;

Докажете, че за така дефинирания автомат $\mathcal{A}$, $\mathcal{L}(\mathcal{A}) = L_1^R$. □

$\delta(q_=, \begin{bmatrix}0\\0\end{bmatrix}) = \delta(q_=, \begin{bmatrix}1\\1\end{bmatrix}) = q_=$

$\delta(q_=, \begin{bmatrix}0\\1\end{bmatrix}) = q_>$

$\delta(q_=, \begin{bmatrix}1\\0\end{bmatrix}) = q_<$

$\delta(q_<, \begin{bmatrix}0\\0\end{bmatrix}) = \delta(q_<, \begin{bmatrix}1\\1\end{bmatrix}) =$
$\delta(q_<, \begin{bmatrix}0\\1\end{bmatrix}) = q_<$

$\delta(q_<, \begin{bmatrix}1\\0\end{bmatrix}) = q_>$

$\delta(q_>, \begin{bmatrix}0\\0\end{bmatrix}) = \delta(q_>, \begin{bmatrix}1\\1\end{bmatrix}) =$
$\delta(q_>, \begin{bmatrix}1\\0\end{bmatrix}) = q_>$

$\delta(q_>, \begin{bmatrix}0\\1\end{bmatrix}) = q_<$

Глава 3

Безконтекстни езици и стекови автомати

3.1 Безконтекстни граматики

В Раздел 2.6 въведохме понятието граматика. След това видяхме как можем да опишем регулярните езици със специален вид граматики, които нарекохме регулярни граматики. Сега ще разгледаме още един вид граматики, които описват по-широк клас от езици.

В [PL98] дефиницията е различна. Там $\Sigma \subseteq V$

На англ. *context-free grammar*

Други срещани наименования на български са *контекстно-свободна*, *контекстно-независима*

- Една граматика $G = (V, \Sigma, R, S)$ се нарича **безконтекстна**, ако имаме ограничението, че $R \subseteq V \times (V \cup \Sigma)^\star$.
- L се нарича **безконтекстен език**, ако съществува безконтекстна граматика G, за която $L = \mathcal{L}(G) = \{\omega \in \Sigma^\star \mid S \to_G^\star \omega\}$.

Забележка. Очевидно е, че всяка регулярна граматика е безконтекстна. Следователно, *всеки регулярен език е безконтекстен.*

Като първи пример нека да видим, че това включване е *строго*, т.е. съществува безконтекстен език, който не е регулярен. Да напомним, че вече видяхме, че езикът $L = \{a^n b^n \mid n \in \mathbb{N}\}$ не е регулярен.

Пример 3.1. Да разгледаме безконтекстната граматика G зададена със следните правила:

$$S \to aSb \mid \varepsilon.$$

Лесно се съобразява, че $\mathcal{L}(G) = \{a^n b^n \mid n \in \mathbb{N}\}$.

Пример 3.2. Да разгледаме безконтекстната граматика G зададена със следните правила:

$$S \to aSc \mid B$$
$$B \to bBc \mid \varepsilon.$$

Лесно се съобразява, че $\mathcal{L}(G) = \{a^n b^k c^{n+k} \mid n, k \in \mathbb{N}\}$.

Пример 3.3. Да разгледаме граматика с правила

$$S \to S + S \mid S * S \mid (S) \mid V$$
$$V \to x \mid y \mid z$$

Думата $x * y + z$ има две различни дървета на извод.
Да разгледаме граматика с правила

$$S \to E + S \mid E$$
$$E \to V * E \mid V \mid (S) * E \mid (S)$$
$$V \to x \mid y \mid z$$

Сега думата $x * y + z$ има само едно дърво на извод.

Пример 3.4.

$$S \to \textbf{if } S \textbf{ then } S \textbf{ else } S \mid \textbf{if } S \textbf{ then } S \mid V$$
$$V \to x \mid y \mid z$$

Ние искаме следната граматика:

$$S \to M \mid U$$
$$M \to \textbf{if } S \textbf{ then } M \textbf{ else } M \mid X$$
$$U \to \textbf{if } S \textbf{ then } S \mid \textbf{if } S \textbf{ then } M \textbf{ else } U$$

Пример 3.5. Да разгледаме граматика с правила

$$S \to E$$
$$E \to E + P \mid P$$
$$P \to P * N \mid N$$
$$N \to (E) \mid a.$$

Задача 3.1. Докажете, че езикът $L = \{a^m b^n c^k \mid m + n \geq k\}$ е безконтекстен.

Доказателство. Да разгледаме граматиката G с правила

$$S \to aSc \mid aS \mid B$$
$$B \to bBc \mid bB \mid \varepsilon.$$

Лесно се вижда с индукция по n, че за всяко n имаме свойствата: ✍ Докажете!

- $S \to^\star a^n S c^n$,

- $S \to^\star a^n S$,
- $B \to^\star a^n B c^n$,
- $B \to^\star b^n B$.

Комбинирайки горните свойства, можем да видим, че за всяко $n \geq k$,

- $S \to^\star a^n S c^k$,
- $B \to^\star b^n B c^k$.

За да докажем, че $L \subseteq L(G)$, да разгледаме една дума $\omega \in L$, т.е. $\omega = a^m b^n c^k$, където $m + n \geq k$. Имаме два случая:

- $k \leq m$, т.е. $m = k + l$ и $m + n = k + l + n$. Тогава имаме изводите:

$$S \to^\star a^k S c^k,\ S \to^\star a^l S,\ S \to B,\ B \to^\star b^n B,\ B \to \varepsilon.$$

  Обединявайки всичко това, получаваме:

$$S \to^\star a^m b^n c^k.$$

- $k > m$, т.е. $k = m + l$, за някое $l > 0$, и $m + n = k + r = m + l + r$, за някое r. Тогава имаме изводите:

$$S \to^\star a^m S c^m,\ S \to B,\ B \to^\star b^l B c^l,\ B \to b^r B,\ B \to \varepsilon,$$

  и отново получаваме $S \to^\star a^m b^n c^k$.

Така доказахме, че $\omega \in \mathcal{L}(G)$.

Сега ще докажем, че $\mathcal{L}(G) \subseteq L$. С индукция по дължината на извода l, ще докажем, че ако $S \overset{l}{\to} \omega$, то $\omega \in M$, където

$$M = \{a^n S c^k \mid n \geq k\} \cup \{a^n b^m B c^k \mid n + m \geq k\} \cup \{a^n b^m c^k \mid n + m \geq k\}.$$

Ако $l = 0$, то е ясно, че $S \overset{0}{\to} S$ и $S \in M$.

Нека $l > 0$ и $S \overset{l-1}{\to} \alpha \to \omega$. От **И.П.** имаме, че $\alpha \in M$. Нека ω се получава от α с прилагане на правилото $C \to \gamma$. Разглеждаме всички варианти за думата $\alpha \in M$ и за правилото $C \to \gamma$ в граматиката G за да докажем, че $\omega \in M$. Удобно е да представим всички случаи в таблица.

$\alpha \in M$	$C \to \gamma$	$\omega \in M$?
$a^n S c^k$	$S \to aSc$	$a^{n+1} S c^{k+1}$
$a^n S c^k$	$S \to aS$	$a^{n+1} S c^k$
$a^n S c^k$	$S \to B$	$a^n B c^k$
$a^n b^m B c^k$	$B \to bBc$	$a^n b^{m+1} B c^{k+1}$
$a^n b^m B c^k$	$B \to bB$	$a^n b^{m+1} B c^k$
$a^n b^m B c^k$	$B \to \varepsilon$	$a^n b^m c^k$

Във всички случаи се установява, че $\omega \in M$. Сега, за всяка дума $\omega \in L(G)$ следва, че

$$\omega \in \Sigma^\star \cap M = \{a^m b^n c^k \mid m + n \geq k\}.$$

□

Задача 3.2. Докажете, че езикът $L = \{a^m b^n c^k \mid m + n \geq k + 1\}$ е безконтекстен.

$S \to aS \mid aSc \mid aB \mid bB$
$B \to bB \mid bBc \mid \varepsilon$

Задача 3.3. Нека ω е произволна дума над азбуката $\{a, b\}$. Тогава:

а) ако $N_a(\omega) = N_b(\omega) + 1$, то съществуват думи ω_1, ω_2, за които $\omega = \omega_1 a \omega_2$, $N_a(\omega_1) = N_b(\omega_1)$ и $N_a(\omega_2) = N_b(\omega_2)$.

б) ако $N_b(\omega) = N_a(\omega) + 1$, то съществуват думи ω_1, ω_2, за които $\omega = \omega_1 b \omega_2$, $N_a(\omega_1) = N_b(\omega_1)$ и $N_a(\omega_2) = N_b(\omega_2)$.

Доказателство. Пълна индукция по дължината на думата ω, за които $N_a(\omega) = N_b(\omega) + 1$.

- $|\omega| = 1$. Тогава $\omega_1 = \omega_2 = \varepsilon$ и $\omega = a$.
- $|\omega| = n+1$. Ще разгледаме два случая, в зависимост от първия символ на ω.
  - Случаят $\omega = a\omega'$ е лесен. (Защо?)
  - Интересният случай е $\omega = b\omega'$. Тогава $\omega = b^{i+1}a\omega'$. Да разгледаме думата ω'', която се получава от ω като премахнем първото срещане на думата ba, т.е. $\omega'' = b^i\omega'$ и $|\omega''| = n - 1$. Понеже от ω сме премахнали равен брой a-та и b-та, $N_a(\omega'') = N_b(\omega'') + 1$. Според **И.П.** за ω'', можем да запишем думата като $\omega'' = \omega_1''a\omega_2''$ и $N_a(\omega_1'') = N_b(\omega_1'')$, $N_a(\omega_2'') = N_b(\omega_2'')$. Понеже b^i е префикс на ω_1'', за да получим обратно ω, трябва да прибавим премахнатата част ba веднага след b^i в ω_1''.

□

Задача 3.4. За произволна дума $\omega \in \{a, b\}^\star$, докажете, че ако $N_a(\omega) > N_b(\omega)$, то съществуват думи ω_1 и ω_2, за които $\omega = \omega_1 a \omega_2$ и $N_a(\omega_1) \geq N_b(\omega_1)$, $N_a(\omega_2) \geq N_b(\omega_2)$.

Задача 3.5. Да се докаже, че езикът $L = \{\alpha \in \{a, b\}^\star \mid N_a(\alpha) = N_b(\alpha)\}$ е безконтекстен.

Доказателство. Една възможна граматика G е следната:

$$S \to aSbS|bSaS|\varepsilon.$$

Алтернативна граматика за езика L е

$S \to \varepsilon \mid aSb \mid bSa \mid SS$.

Например, да разгледаме извода на думата $aabbba$ в тази граматика:

$$S \to aSbS \to aaSbSbS \to aa\varepsilon bSbS \to aab\varepsilon bS \to aabbbSaS$$
$$\to aabbb\varepsilon aS \to aabbba.$$

Като следствие от *Задача 3.3* може лесно да се изведе, че за думи ω, за които $N_a(\omega) = N_b(\omega)$, е изпълнено следното:

а) ако $\omega = a\omega'$, то $\omega = a\omega_1 b\omega_2$ и $N_a(\omega_1) = N_b(\omega_1)$, $N_a(\omega_2) = N_b(\omega_2)$;

б) ако $\omega = b\omega'$, то $\omega = b\omega_1 a\omega_2$ и $N_a(\omega_1) = N_b(\omega_1)$, $N_a(\omega_2) = N_b(\omega_2)$.

Сега първо ще проверим, че $L \subseteq L(G)$. За целта ще докажем с *пълна индукция* по дължината на думата ω, че за всяка дума ω със свойството $N_a(\omega) = N_b(\omega)$ е изпълнено $S \to^\star \omega$.

- Нека $|\omega| = 0$. Тогава $S \to \varepsilon$.
- Нека $|\omega| = k + 1$. Имаме два случая.
  - $\omega = a\omega'$, т.е. от свойство а), $\omega = a\omega_1 b\omega_2$ и $N_a(\omega_1) = N_b(\omega_1)$, $N_a(\omega_2) = N_b(\omega_2)$. Тогава $|\omega_1| \leq k$ и по И.П. $S \to^\star \omega_1$. Аналогично, $S \to^\star \omega_2$. Понеже имаме правило $S \to aSbS$, заключаваме че $S \to^\star a\omega_1 b\omega_2$.
  - $\omega = b\omega'$, т.е. свойство б), $\omega = b\omega_1 a\omega_2$ и $N_a(\omega_1) = N_b(\omega_1)$, $N_a(\omega_2) = N_b(\omega_2)$. Този случай се разглежда аналогично.

Преминаваме към доказателството на другата посока, т.е. $L(G) \subseteq L$. Тук с индукция по дължината на извода l ще докажем, че $S \overset{l}{\to} \omega$, то $\omega \in M$, където

$$M = \{\omega \in \{a, b, S\}^\star \mid N_a(\omega) = N_b(\omega)\}.$$

За $l = 0$ е ясно, че $S \overset{0}{\to}{}^\star S$. За $l = k+1$, то $S \overset{k}{\to}{}^\star \alpha \to \omega$. От **И.П.** имаме, че $\alpha \in M$. Нека ω се получава от α с прилагане на правилото $C \to \gamma$. Разглеждаме всички варианти за думата $\alpha \in M$ и за правилото $C \to \gamma$ в граматиката G за да докажем, че $\omega \in M$. Удобно е да представим всички случаи в таблица.

α	$C \to \gamma$	ω
$\in M$	$S \to aSbS$	$\in M$
$\in M$	$S \to bSaS$	$\in M$
$\in M$	$S \to \varepsilon$	$\in M$

Във всички случаи лесно се установява, че $\omega \in M$. Така за всяка дума $\omega \in L(G)$ следва, че

$$\omega \in \Sigma^\star \cap M = L.$$

□

Задача 3.6. Нека L_1 и L_2 са произволни безконтекстни езици. Докажете, че:

- $L_1 \cup L_2$ е безконтекстен език;
- $L_1^\star$ е безконтекстен език;

Задача 3.7. Да разгледаме граматиката G с правила

$$S \to AA \mid B,\ A \to B \mid bb,\ B \to aa \mid aB.$$

Да се намери езика на тази граматика и да се докаже, че граматиката разпознава точно този език.

3.2 Лема за покачването

- **Дължина** на път в дърво е броят на ребрата по този път.
- **Височина** на дърво е максималната дължина на път в дървото.

Твърдение 3.1. Нека G е в нормална форма на Чомски и думата $\omega \in \mathcal{L}(G)$ има дължина $|\omega| > 2^{n-1}$, за $n \geq 1$. Тогава съществува дърво на извод в G на думата ω с височина $\geq n+1$.

Помислете как трябва да модифицирате това твърдение, ако не изисквате граматиката да е в НФЧ

Упътване. По-лесно е да се докаже контрапозицията на горното твърдение, т.е. ако $\omega \in \mathcal{L}(G)$ има дърво на извод с височина $\leq n$, то $|\omega| \leq 2^{n-1}$. Докажете с индукция по $n \geq 1$. □

Лема 3.1 (за покачването (безконтекстни езици))**.** За всеки безконтекстен език L съществува $p > 0$, такова че ако $\alpha \in L, |\alpha| \geq p$, то съществува разбиване на думата на пет части, $\alpha = xyuvw$, за което е изпълнено:

[Sip97, стр. 123], [HU79, стр. 125]

Ще казваме, че p е константа на покачването

1) $|yv| \geq 1$,
2) $|yuv| \leq p$, и
3) $(\forall i \geq 0)[xy^iuv^iw \in L]$.

Доказателство. Нека G е граматиката за езика L в нормална форма на Чомски. Нека $p = 2^{|V|}$. Ще покажем, че p е константа на покачването за граматиката G.

Възлите във вътрешността на дървото са променливи, а листата са букви

Нека $|\alpha| \geq p = 2^{|V|}$. Тогава от *Твърдение 3.1* следва, че щом $|\alpha| > 2^{|V|-1}$, то α има дърво на извод T_α с дължина $\geq |V|+1$. Това означава, че има път в T_α с поне $|V|+2$ върха. Да фиксираме един такъв път π с максимална възможна дължина. Да разгледаме последните $|V|+1$ върха по пътя π, с които сме асоциирали променливи. От принципа на Дирихле следва, че измежду тези $|V|+1$ променливи има поне една повтаряща се. Нека да тръгнем от листото нагоре към върха по пътя π

и да отбележим първите два върха r_2 и r_1, с които е асоциирана една и съща променлива, която означаваме с X. Нека поддървото на T с връх r_2 да извежда думата u. Тогава поддървото на T с връх r_1 извежда думата yuv, за някои думи $y, v \in \Sigma^\star$. Това означава, че думата α може да се запише като $\alpha = xyuvw$, за някои $x, w \in \Sigma^\star$. Освен това имаме свойствата:

По пътя π е възможно да срещнем много различни двойки повтарящи се променливи, ние избрахме възможно най-долната.

1) $|yv| \geq 1$, защото граматиката е в нормална форма на Чомски. За да получим, че $|yv| = 0$, то трябва да имаме правила от вида $X \to Y$ или $X \to \varepsilon$.

2) $|yuv| \leq p$, защото сме избрали най-долната повтаряща се двойка от променливи.

3) $xy^iuv^iw \in L$, защото можем да заменим поддървото с корен r_2 с поддървото с корен r_1. В случая $i = 0$, заменяме поддървото с корен r_1 с това с корен r_2.

□

Следствие 3.1. Нека L е произволен **безкраен** език. Нека също така е изпълнено, че:

Ако L е краен език, то е ясно, че L е безконтекстен.

- за *всяко* естествено число $p \geq 1$,
- можем да намерим дума $\alpha \in L$, $|\alpha| \geq p$, такава че
- за *всяко* разбиване на думата на пет части, $\alpha = xyuvw$, със свойствата $|yv| \geq 1$ и $|yuv| \leq p$,
- можем да посочим $i \in \mathbb{N}$, за което е изпълнено, че $xy^iuv^iw \notin L$.

Тогава L **не** е безконтекстен език.

✍ Докажете! Аналогично е на *Следствие 2.2*

Следствие 3.2. Нека G е безконтекстна граматика и p е константата на покачването за G, $L = \mathcal{L}(G)$. Тогава $|L| = \infty$ точно тогава, когато съществува $\alpha \in L$, за която $p \leq |\alpha| < 2p$.

✍ Докажете!

Задача 3.8. Докажете, че езикът $L = \{a^nb^nc^n \mid n \in \mathbb{N}\}$ не е безконтекстен.

Доказателство.

(∀) Разглеждаме произволна константа $p \geq 1$.

(∃) Избираме дума $\alpha \in L$, $|\alpha| \geq p$. В случая, нека $\alpha = a^pb^pc^p$.

(∀) Разглеждаме произволно разбиване $xyuvw = \alpha$, за което $|xyv| \leq p$ и $1 \leq |yv|$.

(∃) Ще изберем i, за което $xy^iuv^iw \notin L$. Знаем, че поне едно от y и v не е празната дума. Имаме няколко случая за y и v.

- y и v са думи съставени от една буква. В този случай получаваме, че xy^2uv^2w има различен брой букви a, b и c.
- y или v е съставена от две букви. Тогава е възможно да се окаже, че xy^2uv^2w да има равен брой a, b и c, но тогава редът на буквите е нарушен.
- понеже $|yuv| \leq p$, то не е възможно в y или v да се срещат и трите букви.

Оказа се, че във всички възможни случаи за y и v, $xy^2uv^2w \notin L$.

Така от *Следствие 3.1* следва, че езикът L не е безконтекстен. □

Задача 3.9. Докажете, че $L = \{a^ib^jc^k \mid 0 \leq i \leq j \leq k\}$ не е безконтекстен език.

Доказателство.

(∀) Разглеждаме произволна константа $p \geq 1$.

(∃) Избираме дума $\alpha \in L$, $|\alpha| \geq p$. В случая, нека $\alpha = a^pb^pc^p$.

(∀) Разглеждаме произволно разбиване $xyuvw = \alpha$, за което $|xyv| \leq p$ и $1 \leq |yv|$. Знаем, че поне една от y и v не е празната дума.

(∃) Ще намерим $i \in \mathbb{N}$, за което $xy^iuv^iw \notin L$.

- y и v са съставени от една буква. Имаме три случая.
  - i) a не се среща в y и v. Тогава xy^0vu^0w съдържа повече a от b или c.
  - ii) b не се среща в y и v.
    - Ако a се среща в y или v, тогава xy^2uv^2w съдържа повече a от b.
    - Ако c се среща в y или v, тогава xy^0uv^0w съдържа по-малко c от b.
  - iii) c не се среща в y и v. Тогава xy^2uv^2w съдържа повече a или b от c.
- y или v е съставена от две букви. Тук разглеждаме xy^2uv^2w и съобразяваме, че редът на буквите е нарушен.

□

Задача 3.10. Докажете, че езикът $L = \{\beta\beta \mid \beta \in \{a,b\}^\star\}$ не е безконтекстен.

Упътване. Разгледайте $\alpha = a^p b^p a^p b^p$. □ Защо $\alpha = a^p b a^p b$ не става?

Задача 3.11. Докажете, че езикът

$$L = \{\alpha\beta\alpha^{rev} \mid \alpha, \beta \in \{a, b\}^\star \ \& \ |\alpha| = |\beta|\}$$

не е безконтекстен.

Упътване. $\alpha = a^p b^p a^p b^p b^p a^p$. Покаваме с повече от p би трябвало да свърши работа. □

Твърдение 3.2. Безконтекстните езици **не** са затворени относно сечение и допълнение.

Доказателство. Да разгледаме езика

$$L_0 = \{a^n b^n c^n \mid n \in \mathbb{N}\},$$

за който вече знаем от *Задача 3.8*, че не е безконтекстен. Да вземем също така и безконтекстните езици ✍ Защо са безконтекстни?

$$L_1 = \{a^n b^n c^m \mid n, m \in \mathbb{N}\}, \ L_2 = \{a^m b^n c^n \mid n, m \in \mathbb{N}\},$$

- Понеже $L_0 = L_1 \cap L_2$, то заключаваме, че безконтекстните езици не са затворени относно операцията сечение.
- Да допуснем, че безконтекстните езици са затворени относно операцията допълнение. Тогава $\overline{L_1}$ и $\overline{L_2}$ са безконтекстни. Знаем, че безконтекстните езици са затворени относно обединение. Следователно, езикът $L_3 = \overline{L_1} \cup \overline{L_2}$ също е безконтекстен. Ние допуснахме, че безконтекстните са затворени относно допълнение, следователно $\overline{L_3}$ също е безконтекстен. Но тогава получаваме, че езикът Озн. $\overline{L} = \Sigma^\star \setminus L$

$$L_0 = L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}} = \overline{L_3}$$

  е безконтекстен, което е противоречие.

□

3.3 Алгоритми

3.3.1 Опростяване на безконтекстни граматики

Премахване на безполезните променливи

Нека е дадена безконтекстната граматика $G = \langle V, \Sigma, R, S\rangle$. Една промен- [HU79, стр. 88]

лива A се нарича **полезна**, ако съществува извод от следния вид:

$$S \to^\star \alpha A \beta \to^\star \gamma,$$

където $\gamma \in \Sigma^\star$, а $\alpha, \beta \in (V \cup \Sigma)^\star$. Това означава, че една променлива е полезна, ако участва в извода на някоя дума в езика на граматиката. Една променлива се нарича **безполезна**, ако не е полезна. Целта ни е да получим еквивалентна граматика G' без безполезни променливи. Ще решим задачата като разгледаме две леми.

Лема 3.2. Нека е дадена безконтекстната граматика $G = \langle V, \Sigma, R, S\rangle$ и $\mathcal{L}(G) \neq \emptyset$. Съществува алгоритъм, който намира граматика $G' = \langle V', \Sigma, S, R'\rangle$, за която $\mathcal{L}(G) = \mathcal{L}(G')$, и за всяка променлива $A' \in V'$, съществува дума $\alpha \in \Sigma^\star$, за която $A' \to^\star \alpha$.

Упътване. Да разгледаме следната проста итеративна процедура.

Алгоритъм 1 Намираме $V' = \{A \in V \mid (\exists \alpha \in \Sigma^\star)[A \to^\star \alpha]\}$

```
1: V' := ∅
2: V'' := {A ∈ V | (∃α ∈ Σ*)[A → α]}
3: while V' ≠ V'' do
4:     V' := V''
5:     V'' := V' ∪ {A ∈ V | (∃α ∈ (Σ ∪ V')*)[A → α]}
6: return V'
```

Трябва да докажем, че във V' са точно полезните променливи за G. Очевидно е, че ако $A \in V'$, то A е полезна променлива. За другата посока, с индукция по дължината на извода се доказва, че ако $A \to^\star_G \omega$, то $A \in V'$. ✍ Докажете!

Правилата на G' са всички правила на G, в които участват променливи от V' и букви от Σ. □

Лема 3.3. Съществува алгоритъм, който по дадена безконтекстна граматика $G = \langle V, \Sigma, R, S\rangle$, намира $G' = \langle V', \Sigma', S, R'\rangle$, $\mathcal{L}(G') = \mathcal{L}(G)$, със свойството, че за всяко $x \in V' \cup \Sigma'$ съществуват $\alpha, \beta \in (V' \cup \Sigma')^\star$, за които $S \to^\star \alpha x \beta$, т.е. всяка променлива или буква в G' е достижима от началната променлива S.

Упътване. Намираме V' и Σ' итеративно, като в началото $V' = \{S\}$, $\Sigma' = \emptyset$. Ако $A \in V'$ и имаме правила в G:

$$A \to \alpha_0 \mid \alpha_1 \mid \cdots \mid \alpha_n,$$

то за всяко $i = 0, \ldots, n$ добавяме всички променливи на α_i към V' и всички нетерминали на α_i към Σ'. □

Теорема 3.1. За всяка безконтекстна граматика G, $\mathcal{L}(G) = \emptyset$ точно тогава, когато S е безполезно правило в граматиката.

Доказателство. Нека е дадена безконтекстна граматика G пораждаща L. Прилагаме върху G първо процедурата от *Лема 3.2* и след това върху резултата прилагаме процедурата от *Лема 3.3*. □

Защо е важна последователността на прилагане?

Пример 3.6. Да разгледаме следната граматика G:

$$S \to AB \mid aA$$
$$A \to a \mid aAa$$
$$B \to SB \mid BC$$
$$C \to \varepsilon \mid cC.$$

Първо да намерим променливите, от които се извеждат думи.

- $V_0 = \{A, C\}$, защото $A \to a$ и $C \to \varepsilon$;
- $V_1 = \{A, C, S\}$, защото $S \to aA$;
- не можем да добавим B към V_2, следователно $V_1 = V_2$.

Получаваме граматиката G':

$$S \to aA$$
$$A \to a \mid aAa$$
$$C \to \varepsilon \mid cC.$$

Сега премахваме променливите и буквите, които не са достижими от началната промелива S. Така получаваме граматиката G'':

$$S \to aA$$
$$A \to a \mid aAa.$$

Задача 3.12. Проверете дали $\mathcal{L}(G) = \emptyset$, където правилата на G са:

$$S \to AS \mid BC$$
$$A \to 0 \mid BA \mid SB$$
$$B \to 1 \mid BC \mid AB$$
$$C \to CB \mid SC \mid AS.$$

Премахване на ε-правила

За да премахнем правилата от вида $A \to \varepsilon$, следваме процедурата:

1) Намираме множеството $E = \{A \in V \mid A \to^\star \varepsilon\}$ по следния начин. Първо, $E := \{A \in V \mid A \to \varepsilon\}$. След това, за всяко правило от вида $B \to X_1 \cdots X_k$, ако всяко $X_i \in E$, то добавяме B към E.

Броят на правилата може да се увеличи експоненциално, защото в най-лошия случай извеждаме всички подмножества на дадено множество от променливи

2) Строим множеството от правила R', в което няма правила ε-правила по следния начин. За всяко правило $A \to x_1 \cdots x_k$, където $x_i \in V \cup \Sigma$, добавяме към R' всички правила от вида $A \to y_1 \cdots y_k$, където:

- ако $x_i \notin E$, то $y_i = x_i$;
- ако $x_i \in E$, то $y_i = x_i$ или $y_i = \varepsilon$;
- не всички y_i-та са ε.

Пример 3.7. Нека е дадена граматиката G с правила

$$\begin{aligned} &S \to D \\ &D \to AD \mid b \\ &A \to AB \mid BC \mid a \\ &B \to AA \mid UC \\ &C \to \varepsilon \mid CA \mid a \\ &U \to \varepsilon \mid aUb. \end{aligned}$$

Тогава $E = \{X \in V \mid X \to_G^\star \varepsilon\} = \{A, B, C, U\}$. Това означава, че $\varepsilon \notin \mathcal{L}(G)$. Граматиката G' без ε-правила, за която $\mathcal{L}(G') = \mathcal{L}(G)$ има следните правила

$$\begin{aligned} &S \to D \\ &D \to AD \mid D \mid b \\ &A \to A \mid B \mid C \mid AB \mid BC \mid a \\ &B \to A \mid E \mid C \mid AA \mid UC \\ &C \to C \mid A \mid CA \mid a \\ &U \to aUb \mid ab. \end{aligned}$$

Премахване на преименуващи правила

Преименуващите правила са от вида $A \to B$. Нека е дадена граматика $G = \langle V, \Sigma, R, S\rangle$, в която има преименуващи правила. Ще построим еквивалентна граматика G' без преименуващи правила. В началото нека в R' да добавим всички правила от R, които не са преименуващи. След това, за всяка променлива A, за която $A \to_G^\star B$, ако $B \to \alpha$ е правило в R, което не е преименуващо, то добавяме към R' правилото $A \to \alpha$.

Пример 3.8. Нека е дадена граматиката G с правила

$$\begin{aligned} &S \to B \mid CC \mid b \\ &A \to B \mid S \\ &B \to C \mid BC \\ &C \to AB \mid a \mid b. \end{aligned}$$

Първо добавяме към R' правилата $B \to BC, C \to AB \mid a \mid b, S \to CC \mid b$.

- Лесно се съобразява, че $A \to_G^\star B, S, C$. Добавяме правилата

$$A \to BC \mid AB \mid a \mid b \mid CC.$$

- Имаме $B \to_G^\star C$. Добавяме правилата $B \to AB \mid a \mid b$.
- Имаме $S \to_G^\star B, C$. Добавяме правилата $S \to BC \mid AB \mid a \mid b$.

Накрая получаваме, че граматиката G' има правила

$$\begin{aligned} S &\to BC \mid AB \mid CC \mid a \mid b \\ A &\to BC \mid AB \mid a \mid b \mid CC \\ B &\to AB \mid a \mid b \mid BC \\ C &\to AB \mid a \mid b. \end{aligned}$$

Задача 3.13. Премахнете преименуващите правила от граматиката G, като запазите езика, ако G има следните правила:

$$\begin{aligned} S &\to C \mid CC \mid b \\ A &\to B \\ B &\to S \mid C \mid BC \\ C &\to a \mid AB; \end{aligned}$$

Премахване на дългите правила

Едно правило се нарича дълго, ако е от вида $A \to \beta$, където $|\beta| \geq 3$. Да разгледаме едно дълго правило в граматиката от вида $A \to x_1 x_2 \cdots x_k$, където $k \geq 3$ и $x_i \in V \cup \Sigma$. За да получим еквивалентна граматика без това дълго правило, добавяме нови променливи $A_1, \ldots, A_{k-2}$, и правила

$$A \to x_1 A_1,\ A_1 \to x_2 A_2, \ldots,\ A_{k-2} \to x_{k-1} x_k.$$

Задача 3.14. Нека е дадена граматиката $G = \langle \{S, A, B, C\}, \{a, b\}, S, R \rangle$. Използвайте обща конструкция, за да премахнете „дългите" правила от G като при това получите безконтестна граматика G_1 с език $\mathcal{L}(G) = \mathcal{L}(G_1)$, където правилата на граматиката са:

$$\begin{aligned} S &\to CC \mid b \\ A &\to BSB \mid a \\ B &\to ba \mid BC \\ C &\to BaSA \mid a \mid b. \end{aligned}$$

3.3.2 Нормална Форма на Чомски

Една безконтекстна граматика е в **нормална форма на Чомски**, ако всяко правило е от вида

$$A \to BC \text{ и } A \to a,$$

като B, C *не могат* да бъдат променливата за начало S. Освен това, позволяваме правилото $S \to \varepsilon$. [1]

Теорема 3.2. Всеки безконтекстен език L се поражда от безконтекстна граматика в нормална форма на Чомски.

Доказателство. Нека имаме контекстно-свободна граматика G, за която $L = \mathcal{L}(G)$. Ще построим безконтекстна граматика G' в нормална форма на Чомски, $L = \mathcal{L}(G')$. Следваме следната процедура:

- Добавяме нов начален символ S_0 и правило $S_0 \to S$.

- Премахваме дългите правила. Заменяме правилата от вида $A \to x_1 x_2 \dots x_n$, $n \geq 3$, $x_i \in V \cup \Sigma$, с правилата — Време $O(n)$

$$A \to x_1 A_1,\ A_1 \to x_2 A_2,\ \dots,\ A_{n-2} \to x_{n-1} x_n.$$

  където A_i са нови променливи.

- Премахваме ε-правилата. За всяка променлива $A \neq S_0$ премахваме правилата от вида $A \to \varepsilon$. Това правим по следния начин. — Време $O(n^2)$

  Ако имаме правило от вида $R \to Au$ или $R \to uA$, $u \in V \cup \Sigma$, то добавяме правилото $R \to u$. Например,

  – ако имаме правило $R \to aA$, то добавяме правилото $R \to a$;
  – ако имаме правило $R \to AA$, то добавяме правилото $R \to A$.

  Ако имаме правило от вида $R \to A$, то добавяме правилото $R \to \varepsilon$ само ако променливата R още не е преминала през процедурата за премахване на ε.

- Премахваме преименуващите правила, т.е. правила от вида $A \to B$. Заменяме всяко правило от вида $B \to \beta$ с $A \to \beta$, освен ако $A \to \beta$ е вече премахнато преименуващо правило. — Време $O(n^2)$, Памет $O(n^2)$

- За правила от вида $A \to u_1 u_2$, където $u_1, u_2 \in V \cup \Sigma$, заменяме всяка — Време $O(n)$

[1] В [PL98, стр. 151] дефиницията е малко по-различна. Там дефинират G да бъде в нормална форма на Чомски ако $R \subseteq V \times (V \cup \Sigma)^2$. В този случай губим езиците $\{\varepsilon\}$ и $\{a\}$, за $a \in \Sigma$.

буква u_i с новата променлива U_i и добавяме правилото $U_i \to u_i$. Например, правилото $A \to aB$ се заменя с правилото $A \to XB$ и добавяме правилото $X \to a$, където X е нова променлива.

□

Теорема 3.3. При дадена безконтекстна граматика G с дължина n, можем да намерим еквивалентна на нея граматика G' в нормална форма на Чомски за време $O(n^2)$, като получената граматика е с дължина $O(n^2)$.

3.3.3 Проблемът за принадлежност

Теорема 3.4. Съществува *полиномиален* алгоритъм, който проверява дали дадена дума принадлежни на граматиката G.

За дума α, алгоритъмът работи за време $O(|\alpha|^3)$

Можем да приемем, че $G = \langle V, \Sigma, R, S\rangle$ е граматика в нормална форма на Чомски. Нека $\alpha = a_1 a_2 \dots a_n$ е дума, за която искаме да проверим дали $\alpha \in \mathcal{L}(G)$.

Това е алгоритъм на Cocke, Younger и Kasami (CYK), който е пример за динамично програмиране (стр. 195 от [Koz97])

Алгоритъм 2 Проверка дали $\alpha \in \mathcal{L}(G)$

```
1:  n := |α|                                          ▷ Вход дума α = a_1 ··· a_n
2:  for all i ∈ [1, n] do
3:      V[i, i] := {A ∈ V | A → a_i}
4:  for all i, j ∈ [1, n] & i ≠ j do
5:      V[i, j] := ∅
6:  for all s ∈ [1, n) do                             ▷ Дължина на интервала
7:      for all i ∈ [1, n − s] do                     ▷ Начало на интервала
8:          for all k ∈ [i, i + s) do                 ▷ Разделяне на интервала
9:              if ∃A → BC ∈ R & B ∈ V[i, k] & C ∈ V[k + 1, i + s] then
10:                 V[i, i + s] := V[i, i + s] ∪ {A}
11: if S ∈ V[1, n] then
12:     return True                                   ▷ Има извод на думата от S
13: else
14:     return False
```

Лема 3.4. За дадена граматика в нормална форма на Чомски и дума α, за всяко $0 \le s < |\alpha|$, след s-тата итерация на алгоритъма (редове 6 - 10), за всяка позиция $i = 1, \dots, n - s$,

$$V[i, i + s] = \{A \in V \mid A \to^\star_G a_i \dots a_{i+s}\}.$$

Доказателство. Пълна индукция по s. За $s = 0$ е ясно. (Защо?)

Нека твърдението е вярно за $s < n$. Ще докажем твърдението за $s+1$, т.е. за всяко $i = 1, \dots, n - s - 1$,

$$V[i, i + s + 1] = \{A \in V \mid A \to^\star_G a_i \dots a_{i+s+1}\}.$$

За едната посока, да разгледаме първото правило в извода $A \to^\star_G a_i \cdots a_{i+s+1}$. Понеже G е в НФЧ, то е от вида $A \to BC$ и тогава съществува някое t, за което $B \to^\star a_i \cdots a_{i+t}$ и $C \to^\star a_{i+t+1} \cdots a_{i+s+1}$. От И.П. получаваме, че $B \in V[i, i+t]$ и $C \in V[i+t+1, i+s+1]$. Тогава от ред 10 на алгоритъма е ясно, че $A \in V[i, i+s+1]$.

За другата посока, нека $A \in V[i, i+s+1]$. Единствената стъпка на алгоритъма, при която може да сме добавили A към множеството $V[i, i+s+1]$ е ред 10. Тогава имаме, че съществува k, за което $B \in V[i,k]$, $C \in V[k+1, i+s+1]$, и $A \to BC$ е правило в граматиката G. От И.П. имаме, че $B \to^\star_G a_i \cdots a_k$ и $C \to^\star_G a_{k+1} \cdots a_{i+s+1}$. Заключаваме веднага, че $A \to^\star_G a_i \cdots a_{i+s+1}$. □

Пример 3.9. Нека е дадена граматиката G с правила

$$\begin{aligned} S &\to a \mid AB \mid AC \\ A &\to a \\ B &\to b \\ C &\to SB \mid AS. \end{aligned}$$

Ще приложим CYK алгоритъма за да проверим дали думата $aaabb \in \mathcal{L}(G)$.

- $V[1,1] = V[2,2] = V[3,3] = \{S, A\}$; $V[4,4] = V[5,5] = \{B\}$.
- $V[1,2] = V[2,3] = \{C\}$; $V[3,4] = \{S, C\}$; $V[4,5] = \emptyset$.
- $V[1,3] = \{S\} \cup \emptyset$; $V[2,4] = \{S, C\} \cup \emptyset$; $V[3,5] = \emptyset \cup \{C\}$.
- $V[1,4] = \{S, C\} \cup \emptyset \cup \emptyset = \{S, C\}$; $V[2,5] = \{S\} \cup \emptyset \cup \{C\} = \{S, C\}$.
- $V[1,5] = \{S, C\} \cup \emptyset \cup \emptyset \cup \{C\} = \{S, C\}$.

Понеже $S \in V[1,5]$, то $aaabb \in \mathcal{L}(G)$.

Теорема 3.5. Съществуват алгоритми, които определят по дадена безконтекстна граматика G дали: [HU79, стр. 137]

а) $|\mathcal{L}(G)| = 0$;

б) $|\mathcal{L}(G)| < \infty$;

в) $|\mathcal{L}(G)| = \infty$.

Доказателство. Нека е дадена една безконтекстна граматика G.

($\mathcal{L}(G) = \emptyset$?) Прилагаме алгоритъма за премахване на безполезните променливи. Ако открием, че S е безполезна променлива, то $\mathcal{L}(G) = \emptyset$.

($|\mathcal{L}(G)| < \infty$? **или** $|\mathcal{L}(G)| = \infty$?) Нека да разгледаме граматиката G' в НФЧ без безполезни променливи, за която $\mathcal{L}(G) = \mathcal{L}(G')$. От граматиката $G' = \langle V', \Sigma, S, R' \rangle$ строим граф с възли променливите от V' като за $A, B \in V'$ имаме ребро $A \to B$ точно тогава, когато съществува $C \in V'$, за което $A \to BC$ или $A \to CB$ е правило в R'.

Ако в получения граф имаме цикъл, то $\mathcal{L}(G') = \infty$.

□

3.4 Недетерминирани стекови автомати

Недетерминиран стеков автомат е седморка от вида

На англ. *Push-down automaton*

$$P = \langle Q, \Sigma, \Gamma, \#, s, \Delta, F \rangle,$$

където:

- Q е крайно множество от състояния;
- Σ е крайна входна азбука;
- Γ е крайна стекова азбука;
- $\# \in \Gamma$ е символ за дъно на стека;
- $s \in Q$ е начално състояние;
- $\Delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathscr{P}_{fin}(Q \times \Gamma^\star)$ е функция на преходите;
- $F \subseteq Q$ е множество от заключителни състояния.

Озн. $\mathscr{P}_{fin}(A)$ - крайните подмножества на A

Моментно описание (или конфигурация) на изчислението със стеков автомат представлява тройка от вида $(q, \alpha, \gamma) \in Q \times \Sigma^\star \times \Gamma^\star$, т.е. автоматът се намира в състояние q, думата, която остава да се прочете е α, а съдържанието на стека е думата γ. Удобно е да въведем бинарната релация $\vdash_P$ над $Q \times \Sigma^\star \times \Gamma^\star$, която ще ни казва как моментното описание на автомата P се променя след изпълнение на една стъпка:

Instanteneous description

$$(q, x\alpha, Y\gamma) \vdash_P (p, \alpha, \beta\gamma), \text{ ако } \Delta(q, x, Y) \ni (p, \beta),$$

$$(q, \alpha, Y\gamma) \vdash_P (p, \alpha, \beta\gamma), \text{ ако } \Delta(q, \varepsilon, Y) \ni (p, \beta).$$

Рефлексивното и транзитивно затваряне на $\vdash_P$ ще означаваме с $\vdash_P^\star$. Сега вече можем да дадем дефиниция на език, разпознаван от стеков автомат P.

- $\mathcal{L}_F(P)$ е езика, който се разпознава от P **с финално състояние**,

$$\mathcal{L}_F(P) = \{\omega \mid (q_0, \omega, \#) \vdash_P^\star (q, \varepsilon, \alpha), \text{ за някои } q \in F, \alpha \in \Gamma^\star\}.$$

- $\mathcal{L}_S(P)$ е езика, който се разпознава от P **с празен стек**,

$$\mathcal{L}_S(P) = \{\omega \mid (q_0, \omega, \#) \vdash_P^\star (q, \varepsilon, \varepsilon), \text{ за някое } q \in Q\}.$$

Пример 3.10. За езика $L = \{a^n b^n \mid n \in \mathbb{N}\}$ съществува стеков автомат P, такъв че $L = \mathcal{L}_S(P)$. Да разгледаме $P = \langle Q, \Sigma, \Gamma, \#, s, \Delta, F\rangle$, където

- $Q = \{q, p\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{\#, A\}$, където символът $\#$ служи за дъно на стека, а броят на A-тата в стека ще показват колко букви a сме прочели от думата;
- $F = \emptyset$, защото разпознаваме с празен стек, а не с финално състояние;
- Функцията на преходите Δ има следната дефиниция:

  (1) $\Delta(q, a, \#) = \{(q, A\#), (p, A\#)\}$;
  (2) $\Delta(q, a, A) = \{(q, AA), (p, AA)\}$;
  (3) $\Delta(q, \varepsilon, \#) = \{(q, \varepsilon)\}$;
  (4) $\Delta(q, b, A) = \emptyset$;
  (5) $\Delta(q, b, \#) = \emptyset$;
  (6) $\Delta(p, b, A) = \{(p, \varepsilon)\}$;
  (7) $\Delta(p, b, \#) = \emptyset$;

Да видим как думата a^2b^2 се разпознава от автомата с празен стек:

✍ Докажете, че $L = \mathcal{L}_S(P)$!

$$\begin{aligned}
(q, a^2b^2, \#) &\vdash_P (q, ab^2, A\#) && \text{(правило (1))}\\
&\vdash_P (p, b^2, AA\#) && \text{(правило (2))}\\
&\vdash_P (p, b, A\#) && \text{(правило (6))}\\
&\vdash_P (p, \varepsilon, \#) && \text{(правило (6))}\\
&\vdash_P (p, \varepsilon, \varepsilon) && \text{(правило (7))}
\end{aligned}$$

Пример 3.11. За езика $L = \{\omega\omega^R \mid \omega \in \{a, b\}^\star\}$ съществува стеков автомат P, такъв че $L = \mathcal{L}_S(P)$. Нека $P = \langle Q, \Sigma, \Gamma, \#, s, \Delta, F\rangle$, където:

- $Q = \{q\}$; $F = \emptyset$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{A, B, \#\}$;
- началното състояние $s = q$;

- $F = \emptyset$, защото разпознаваме с празен стек;
- Функцията на преходите Δ има следната дефиниция:

(1) $\Delta(q, a, \#) = \{(q, A\#)\}$;
(2) $\Delta(q, b, \#) = \{(q, B\#)\}$;
(3) $\Delta(q, \varepsilon, \#) = \{(q, \varepsilon)\}$;
(4) $\Delta(q, a, A) = \{(q, AA), (p, \varepsilon)\}$;
(5) $\Delta(q, a, B) = \{(q, AB)\}$;
(6) $\Delta(q, b, A) = \{(q, BA)\}$;
(7) $\Delta(q, b, B) = \{(q, BB), (p, \varepsilon)\}$;
(8) $\Delta(p, a, A) = \{(p, \varepsilon)\}$;
(9) $\Delta(p, b, B) = \{(p, \varepsilon)\}$;
(10) $\Delta(p, \varepsilon, \#) = \{(p, \varepsilon)\}$;

Основното наблюдение, което трябва да направим за да разберем конструкцията на автомата е, че всяка дума от вида $\omega\omega^R$ може да се запише като $\omega_1 aa\omega_1^R$ или $\omega_1 bb\omega_1^R$. Да видим защо P разпознава думата $abaaba$ с празен стек. Започваме по следния начин:

$$\begin{aligned}(q, abaaba, \#) &\vdash_P (q, baaba, A\#) && \text{(правило (1))}\\ &\vdash_P (q, aaba, BA\#) && \text{(правило (6))}\\ &\vdash_P (q, aba, ABA\#) && \text{(правило (5)).}\end{aligned}$$

Сега можем да направим два избора как да продължим. Състоянието p служи за маркер, което ни казва, че вече сме започнали да четем ω^R. Поради тази причина, продължаваме така:

$$\begin{aligned}(q, aba, ABA\#) &\vdash_P (p, ba, BA\#) && \text{(правило (4))}\\ &\vdash_P (p, a, A\#) && \text{(правило (9))}\\ &\vdash_P (p, \varepsilon, \#) && \text{(правило (8))}\\ &\vdash_P (p, \varepsilon, \varepsilon) && \text{(правило (10)).}\end{aligned}$$

Да проиграем още един пример. Да видим защо думата aba не се извежда от автомата.

$$\begin{aligned}(q, aba, \#) &\vdash_P (q, ba, A\#) && \text{(правило (1))}\\ &\vdash_P (q, a, BA\#) && \text{(правило (6))}\\ &\vdash_P (q, \varepsilon, ABA\#) && \text{(правило (5)).}\end{aligned}$$

От последното моментно описание на автомата нямаме нито един преход, следователно думата aba не се разпознава от P с празен стек.

✍ Докажете, че $\mathcal{L}_S(P) = L$!

Пример 3.12. За езика $L = \{\omega \in \{a,b\}^\star \mid N_a(\omega) = N_b(\omega)\}$ съществува стеков автомат P, такъв че $L = \mathcal{L}_S(P)$. Нека $P = \langle Q, \Sigma, \Gamma, \#, s, \Delta, F\rangle$, където:

- $Q = \{q\}$; $F = \emptyset$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{A, B, \#\}$;
- началното състояние $s = q$;
- $F = \emptyset$, защото разпознаваме с празен стек;
- Функцията на преходите Δ има следната дефиниция:

  (1) $\Delta(q, \varepsilon, \#) = \{(q, \varepsilon)\}$;
  (2) $\Delta(q, a, \#) = \{(q, A\#)\}$;
  (3) $\Delta(q, b, \#) = \{(q, B\#)\}$;
  (4) $\Delta(q, a, A) = \{(q, AA)\}$;
  (5) $\Delta(q, a, B) = \{(q, \varepsilon)\}$;
  (6) $\Delta(q, b, A) = \{(q, \varepsilon)\}$;
  (7) $\Delta(q, b, B) = \{(q, BB)\}$.

Да видим защо думата $abbbaa \in \mathcal{L}_S(P)$.

$$
\begin{aligned}
(q, abbbaa, \#) &\vdash (q, bbbaa, A\#) && \text{(правило (2))}\\
&\vdash (q, bbaa, \#) && \text{(правило (6))}\\
&\vdash (q, baa, B\#) && \text{(правило (3))}\\
&\vdash (q, aa, BB\#) && \text{(правило (7))}\\
&\vdash (q, a, B\#) && \text{(правило (5))}\\
&\vdash (q, \varepsilon, \#) && \text{(правило (5))}\\
&\vdash (q, \varepsilon, \varepsilon) && \text{(правило (1)).}
\end{aligned}
$$

Теорема 3.6. Нека L е произволен език над азбука Σ. ([HU79], стр. 114)

1) Ако съществува НСА P, за който $L = \mathcal{L}_F(P)$, то съществува НСА P', за който $L = \mathcal{L}_S(P')$.

2) Ако съществува НСА P, за който $L = \mathcal{L}_S(P)$, то съществува НСА P', за който $L = \mathcal{L}_F(P')$.

С други думи, езиците разпознавани от НСА с празен стек са точно езиците разпознавани от НСА с финално състояние.

Доказателство.

1) Нека $L = \mathcal{L}_F(P)$, където $P = \langle Q, \Sigma, \Gamma, \#, s, \Delta, F\rangle$. Ще построим P', така че да симулира P и като отидем във финално състояние ще изпразним стека. Нека

$$P' = \langle Q \cup \{q_e, s'\}, \Sigma, \Gamma \cup \{\$\}, \$, s', \Delta', \emptyset\rangle,$$

където $\$ \notin \Gamma$. Важно е P' да има собствен нов символ за дъно на стека, защото е възможно за някоя дума $\alpha \notin \mathcal{L}_F(P)$ стековият автомат P да си изчисти стека и така да разпознаем повече думи.

- $\Delta'(s', \varepsilon, \$) = \{(s, \#\$)\}$; - започваме симулацията
- $\Delta'(q, a, X)$ включва множеството $\Delta(q, a, X)$, за всяко $q \in Q$, $a \in \Sigma_\varepsilon$, $X \in \Gamma$; - симулираме P
- $\Delta'(q, \varepsilon, X)$ съдържа също и елемента (q_e, ε), за всяко $q \in F$, $X \in \Gamma \cup \{\$\}$; - ако сме във финално, започваме да чистим стека
- $\Delta'(q_e, \varepsilon, X) = \{(q_e, \varepsilon)\}$, за всяко $X \in \Gamma \cup \{\$\}$; - изчистваме стека
- Δ' няма други правила.

2) Сега имаме $L = \mathcal{L}_S(P)$, където $P = \langle Q, \Sigma, \Gamma, \#, s, \Delta, \emptyset\rangle$. Да положим

$$P' = \langle Q \cup \{s', q_f\}, \Sigma, \Gamma \cup \{\$\}, \Delta', \$, \{q_f\}\rangle.$$

P' ще симулира P като ще внимаваме кога P изчиства символа $\#$. Тогава ще искаме да отидем във финалното състояние q_f.

- $\Delta'(s', \varepsilon, \$) = \{(s, \#\$)\}$; - започваме симулацията
- $\Delta'(q, a, X) = \Delta(q, a, X)$, за всяко $q \in Q$, $a \in \Sigma_\varepsilon$, $X \in \Gamma$; - симулираме P
- $\Delta'(q, \varepsilon, \$) = \{(q_f, \varepsilon)\}$. - щом сме стигнали до \$, значи P е изчистил стека си

□

Задача 3.15. Като използвате стековия автомат от Пример 3.10, дефинирайте автомат P', за който $\mathcal{L}_F(P') = \{a^n b^n \mid n \in \mathbb{N}\}$.

Теорема 3.7. Класът на езиците, които се разпознават от краен стеков автомат съвпада с класа на безконтекстните езици.

Доказателство. Ще разгледаме двете посоки на твърдението поотделно. [HU79, стр. 117]

1) Нека е дадена безконтекстна граматика $G = \langle V, \Sigma, R, S\rangle$. Нашата цел е да построим стеков автомат P, така че $\mathcal{L}_S(P) = \mathcal{L}(G)$. Нека

$$P = \langle \{q\}, \Sigma, \Sigma \cup V, S, q, \Delta, \emptyset\rangle,$$

където функцията на преходите е:

$$\Delta(q, \varepsilon, A) = \{(q, \alpha) \mid A \to \alpha \text{ е правило в граматиката } G\}$$
$$\Delta(q, a, a) = \{(q, \varepsilon)\}$$

2) Нека имаме $P = \langle Q, \Sigma, \Gamma, \Delta, s, \#, \emptyset \rangle$. Ще дефинираме безконтекстна граматика G, за която $\mathcal{L}_S(P) = \mathcal{L}(G)$. Променливите на граматика са

$$V = \{[q, A, p] \mid q, p \in Q, A \in \Gamma\}.$$

Правилата на G са следните:

- $S \to [s, \#, q]$, за всяко $q \in Q$;
- $[q, A, p] \to a[q_1, B_1, q_2][q_2, B_2, q_3] \dots [q_m, B_m, p]$, където

$$(q_1, B_1 \dots B_m) \in \Delta(q, a, A)$$

и произволни $q, q_1, \dots, q_m, p \in Q$, $a \in \Sigma \cup \{\varepsilon\}$.
Да обърнем внимание, че е възможно $m = 0$. Това означава, че $(p, \varepsilon) \in \Delta(q, a, A)$ и тогава имаме правилото $[q, A, p] \to a$, където $a \in \Sigma \cup \{\varepsilon\}$.

Трябва да докажем, че:

$$[q, A, p] \to_G^\star \alpha \;\Leftrightarrow\; (q, \alpha, A) \vdash_P^\star (p, \varepsilon, \varepsilon).$$

($\Rightarrow$) С пълна индукция по i, ще докажем, че

$$(q, \alpha, A) \vdash_P^i (p, \varepsilon, \varepsilon) \implies [q, A, p] \to_G^\star \alpha.$$

Ако $i = 1$, то е лесно, защото $\alpha \in \Sigma \cup \{\varepsilon\}$ и $m = 0$. Според конструкцията на граматиката G имаме правилото $[q, A, p] \to a$.
Ако $i > 1$, нека $\alpha = a\beta$. Тогава: Възможно е $a = \varepsilon$

$$(q, a\beta, A) \vdash_P (q_1, \beta, B_1 \dots B_n) \vdash_P^{i-1} (p, \varepsilon, \varepsilon).$$

Да разбием думата β на n части, $\beta = \beta_1 \cdots \beta_n$, със свойството, че след като прочетем β_i сме премахнали променливата B_i от върха на стека. Това означава, че съществуват състояния $q_2, \dots, q_n$:

$$(q_1, \beta_1, B_1) \vdash_P^{l_1} (q_2, \varepsilon, \varepsilon)$$
$$(q_2, \beta_2, B_2) \vdash_P^{l_2} (q_3, \varepsilon, \varepsilon)$$
$$\vdots$$
$$(q_n, \beta_n, B_n) \vdash_P^{l_n} (p, \varepsilon, \varepsilon),$$

където $l_1 + l_2 + \cdots + l_n = i - 1$. Сега по **И.П.** получаваме:

$$\begin{aligned}
&(q_1, \beta_1, B_1) \vdash_P^{l_1} (q_2, \varepsilon, \varepsilon) \implies [q_1, B_1, q_2] \to_G^\star \beta_1 \\
&(q_2, \beta_2, B_2) \vdash_P^{l_2} (q_3, \varepsilon, \varepsilon) \implies [q_2, B_2, q_3] \to_G^\star \beta_2 \\
&\vdots \\
&(q_n, \beta_n, B_n) \vdash_P^{l_n} (p, \varepsilon, \varepsilon) \implies [q_n, B_n, p] \to_G^\star \beta_n.
\end{aligned}$$

Понеже имаме правилото $\Delta(q, a, A) \ni (q_1, B_1 \cdots B_n)$, то в граматиката имаме правилото

$$[q, A, p] \to_G a[q_1, B_1, q_2] \ldots [q_n, B_n, p].$$

Обединявайки всичко, получаваме извода

$$[q, A, p] \to_G^\star a\beta.$$

($\Leftarrow$) Отново с пълна индукция по i ще докажем, че

$$[q, A, p] \to_G^i \alpha \implies (q, \alpha, A) \vdash_P^\star (p, \varepsilon, \varepsilon).$$

Ако $i = 1$, то имаме $[q, A, p] \to \alpha$, където $\alpha = a$ или $\alpha = \varepsilon$. Ако $i > 1$, то имаме, че $\alpha = a\beta$ и за някое n,

$$[q, A, p] \to_G a[q_1, B_1, q_2][q_2, B_2, q_3] \ldots [q_n, B_n, p] \to_G^{i-1} \beta.$$

Отново нека $\beta = \beta_1 \ldots \beta_n$, където

$$\begin{aligned}
&[q_1, B_1, q_2] \to_G^{i_1} \beta_1 \\
&[q_2, B_2, q_3] \to_G^{i_2} \beta_2 \\
&\vdots \\
&[q_n, B_n, p] \to_G^{i_n} \beta_n,
\end{aligned}$$

където $i_1 + i_2 + \cdots + i_n = i - 1$. От **И.П.** получаваме, че

$$\begin{aligned}
&[q_1, B_1, q_2] \to_G^{i_1} \beta_1 \implies (q_1, \beta_1, B_1) \vdash_P^\star (q_2, \varepsilon, \varepsilon) \\
&[q_2, B_2, q_3] \to_G^{i_2} \beta_2 \implies (q_2, \beta_2, B_2) \vdash_P^\star (q_3, \varepsilon, \varepsilon) \\
&\vdots \\
&[q_n, B_n, p] \to_G^{i_n} \beta_n \implies (q_n, \beta_n, B_n) \vdash_P^\star (p, \varepsilon, \varepsilon).
\end{aligned}$$

Правилото $[q, A, p] \to_G a[q_1, B_1, q_2][q_2, B_2, q_3] \ldots [q_n, B_n, p]$ е добавено в граматиката, защото $\Delta(q, a, A) \ni (q_1, B_1 B_2 \cdots B_n)$. Обединявайки всичко, което знаем, получаваме:

$$\begin{aligned}
(q, a\beta, A) &\vdash_P (q_1, \beta_1 \cdots \beta_n, B_1 \cdots B_n) \\
&\vdash_P^\star (q_2, \beta_2 \cdots \beta_n, B_2 \cdots B_n) \\
&\cdots \\
&\vdash_P^\star (q_n, \beta_n, B_n) \\
&\vdash_P^\star (p, \varepsilon, \varepsilon)
\end{aligned}$$

□

Пример 3.13. Нека е дадена граматиката G с правила

$$S \to ASB \mid \varepsilon$$
$$A \to aAa \mid a$$
$$B \to bBb \mid b.$$

Ще построим стеков автомат $P = \langle Q, \Sigma, \Gamma, \#, s, \Delta, F\rangle$, такъв че $\mathcal{L}_S(P) = \mathcal{L}(G)$.

- $\Sigma = \{a, b\}$;
- $\Gamma = \{A, S, B, a, b\}$;
- $\# = S$;
- $Q = \{q\}$;
- $F = \emptyset$;
- Дефинираме релацията на преходите, следвайки конструкцията от *Теорема 3.7*:
  - $\Delta(q, \varepsilon, S) = \{\langle q, ASB\rangle, \langle q, \varepsilon\rangle\}$;
  - $\Delta(q, \varepsilon, A) = \{\langle q, aAa\rangle, \langle q, a\rangle\}$;
  - $\Delta(q, \varepsilon, B) = \{\langle q, bBb\rangle, \langle q, b\rangle\}$;
  - $\Delta(q, a, a) = \{\langle q, \varepsilon\rangle\}$;
  - $\Delta(q, b, b) = \{\langle q, \varepsilon\rangle\}$.

Пример 3.14. Да видим как по стековия автомат за езика

$$L = \{\omega\omega^R \mid \omega \in \{a, b\}^\star\}$$

от *Пример 3.11* можем да построим граматика, следвайки конструкцията от *Теорема 3.7*.

- Променливите на граматиката са $2 \cdot 3 \cdot 2 + 1 = 13$.
- Започваме с правилата $S \to [q, \#, q] \mid [q, \#, p]$, защото началното състояние на автомата е q и символът за дъно на стек е $\#$.
- Понеже $\Delta(q, a, \#) = \{(q, A\#)\}$, то имаме правилата

$$[q, \#, q] \to a[q, A, q][q, \#, q] \mid a[q, A, p][p, \#, q]$$
$$[q, \#, p] \to a[q, A, q][q, \#, p] \mid a[q, A, p][p, \#, p].$$

- Понеже $\Delta(q, a, A) = \{(q, AA), (p, \varepsilon)\}$, то добавяме правилата:

$$[q, A, q] \to a[q, A, q][q, A, q] \mid a[q, A, p][p, A, q]$$
$$[q, A, p] \to a \mid a[q, A, q][q, A, p] \mid a[q, A, p][p, A, p].$$

- Понеже $\Delta(q, a, B) = \{(q, AB)\}$, то добавяме правилата:

✍ Довършете граматиката и след това я опростете!

$$[q, B, q] \to a[q, A, q][q, B, q] \mid a[q, A, p][p, B, q]$$
$$[q, B, p] \to a[q, A, q][q, B, p] \mid a[q, A, p][p, B, p].$$

Теорема 3.8. Нека L е безконтекстен език и R е регулярен език. Тогава тяхното сечение $L \cap R$ е безконтекстен език. [PL98, стр. 144]

Доказателство. Нека имаме стеков автомат

$$\mathcal{M}_1 = \langle Q_1, \Sigma, \Gamma, \#, s_1, \Delta_1, F_1 \rangle, \text{ където } \mathcal{L}_F(\mathcal{M}_1) = L,$$

и краен детерминиран автомат

всъщност няма нужда да е детерминиран

$$\mathcal{M}_2 = \langle Q_2, \Sigma, s_2, \delta_2, F_2 \rangle, \text{ където } \mathcal{L}(\mathcal{M}_2) = R.$$

Ще определим нов стеков автомат $\mathcal{M} = \langle Q, \Sigma, \Gamma, \#, s, \Delta, F \rangle$, където

- $Q = Q_1 \times Q_2$;
- $s = \langle s_1, s_2 \rangle$;
- $F = F_1 \times F_2$;
- Функцията на преходите Δ е дефинирана както следва:
  - Ако $\Delta_1(q_1, a, b) \ni \langle r_1, c \rangle$ и $\delta_2(q_2, a) = r_2$, то

симулираме едновременно изчислението и на двата автомата

$$\Delta(\langle q_1, q_2 \rangle, a, b) \ni \langle \langle r_1, r_2 \rangle, c \rangle.$$

  - Ако $\Delta_1(q_1, \varepsilon, b) \ni \langle r_1, c \rangle$, то за всяко $q_2 \in Q_2$,

празен ход на автомата M_2

$$\Delta(\langle q_1, q_2 \rangle, \varepsilon, b) \ni \langle \langle r_1, q_2 \rangle, c \rangle.$$

  - Δ не съдържа други преходи;

✍ Докажете, че $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{M}_1) \cap \mathcal{L}(\mathcal{M}_2)$!

□

Теорема 3.8 е удобна, когато искаме да докажем, че даден език не е безконтекстен. С нейна помощ можем да сведем езика до друг, за който вече знаем, че не е безконтекстен.

Пример 3.15. Езикът $L = \{\omega \in \{a,b,c\}^\star \mid N_a(\omega) = N_b(\omega) = N_c(\omega)\}$ не е безконтекстен. Да допуснем, че L е безконтекстен език. Тогава

$$L' = L \cap \mathcal{L}(a^\star b^\star c^\star)$$

също е безконтекстен език. Но $L' = \{a^n b^n c^n \mid n \in \mathbb{N}\}$, за който знаем от *Задача 3.8*, че *не* е безконтекстен. Достигнахме до противоречие. Следователно, L не е безконтекстен език.

Задача 3.16. Докажете, че езикът $L = \{\omega \in \{a,b\}^\star \mid N_a(\omega) = 2N_b(\omega)\}$ не е безконтекстен.

Упътване. Разгледайте езика $L' = L \cap \mathcal{L}(a^\star b^\star a^\star)$. □

3.5 Допълнителни задачи

Задача 3.17. Постройте регулярен израз за езика на следната граматика:

$$\begin{aligned} S &\to S + S \mid S * S \mid A \\ A &\to KL \mid LK \\ K &\to 0K \mid \varepsilon \\ L &\to 1K \mid \varepsilon. \end{aligned}$$

Задача 3.18. Докажете, че следните езици са безконтекстни.

а) $L = \{ww^R \mid w \in \{a,b\}^\star\}$; $S \to aSa \mid bSb \mid \varepsilon$

б) $L = \{w \in \{a,b\}^\star \mid w = w^R\}$; $S \to aSa \mid bSb \mid a \mid b \mid \varepsilon$

в) $L = \{a^n b^{2m} c^n \mid m,n \in \mathbb{N}\}$;

г) $L = \{a^n b^m c^m d^n \mid m,n \in \mathbb{N}\}$;

д) $L = \{a^n b^{2k} \mid n,k \in \mathbb{N} \ \&\ n \neq k\}$; Обединение на два езика

е) $L = \{a^n b^k \mid n > k\}$; $S \to aSb|aS|a$

ж) $L = \{a^n b^k \mid n \geq 2k\}$;

з) $L = \{a^n b^k c^m \mid n + k \geq m + 1\}$; $S \to aSc|aS|aB|bB$, $B \to bBc|bB|\varepsilon$

и) $L = \{a^n b^k c^m \mid n + k \geq m + 2\}$;

к) $L = \{a^n b^k c^m \mid n + k + 1 \geq m\}$; $S \to aSc|aS|B|Bc$, $B \to bBc|bB|\varepsilon$

л) $L = \{a^n b^m c^{2k} \mid n \neq 2m \ \&\ k \geq 1\}$;

м) $L = \{a^n b^k c^m \mid n + k \leq m\}$;

н) $L = \{a^n b^k c^m \mid n + k \leq m + 1\}$;

о) $L = \{a^n b^m c^k \mid n, m, k$ не са страни на триъгълник$\}$. Обединение на три езика

п) $L = \{a, b\}^\star \setminus \{a^{2n} b^n \mid n \in \mathbb{N}\}$;

р) $L = \{\alpha \in \{a, b\}^\star \mid N_a(\alpha) = N_b(\alpha) + 1\}$;

с) $L = \{\alpha \in \{a, b\}^\star \mid N_a(\alpha) \geq N_b(\alpha)\}$;

т) $L = \{\alpha \in \{a, b\}^\star \mid N_a(\alpha) > N_b(\alpha)\}$;

у) $L = \{\alpha \sharp \beta \mid \alpha, \beta \in \{a, b\}^\star \ \&\ \alpha \neq \beta\}$;

ф) $L = \{\alpha\beta \in \{a, b\}^\star \mid \ |\alpha| = |\beta| \ \&\ \alpha \neq \beta\}$;

х) $L = \{\alpha \in \{a, b\}^\star \mid$ във всеки префикс β на α, $N_b(\beta) \leq N_a(\beta)\}$;

ц) $L = \{\alpha \sharp \beta \mid \alpha, \beta \in \{a, b\}^\star \ \&\ \alpha^R$ е поддума на $\beta\}$.

ч) $L = \{\omega_1 \sharp \omega_2 \sharp \cdots \sharp \omega_n \mid n \geq 2 \ \&\ \omega_1, \omega_2, \ldots, \omega_n \in \{a, b\}^\star \ \&\ |\omega_1| = |\omega_2|\}$;

ш) $L = \{\omega_1 \sharp \omega_2 \sharp \cdots \sharp \omega_n \mid n \geq 2 \ \&\ \omega_1, \ldots, \omega_n \in \{a, b\}^\star \ \&\ (\exists i \neq j)[|\omega_i| = |\omega_j|]\}$;

щ) $L = \{\omega_1 \sharp \omega_2 \sharp \cdots \sharp \omega_n \mid n \geq 2 \ \&\ (\forall i \in [1, n])[\omega_i \in \{a, b\}^\star \ \&\ |\omega_i| = |\omega_{n+1-i}|]\}$.

Упътване.

р) $S \to EaE$, $E \to aEbE|bEaE|\varepsilon$.

с) $S \to E|SaS$, $E \to aEbE|bEaE|\varepsilon$.

у) Разгледайте граматиката:

$$\begin{aligned}
&S \to AaR \mid BbR \mid E\\
&A \to XAX \mid bR\sharp\\
&B \to XBX \mid aR\sharp\\
&E \to XEX \mid XR\sharp \mid \sharp XR\\
&R \to XR \mid \varepsilon\\
&X \to a \mid b.
\end{aligned}$$

Имаме, че за произволни думи $\alpha, \beta, \gamma, \delta \in \{a, b\}^\star$,

$$\begin{aligned}
&S \to^\star \alpha b\gamma\sharp\beta a\delta \ \&\ |\alpha| = |\beta|,\\
&S \to^\star \alpha a\gamma\sharp\beta b\delta \ \&\ |\alpha| = |\beta|, \text{ или}\\
&S \to^\star \alpha\sharp\beta \ \&\ |\alpha| \neq |\beta|
\end{aligned}$$

ф) Разгледайте граматиката:

$$\begin{aligned} S &\to AB \mid BA \\ A &\to XAX \mid a \\ B &\to XBX \mid b \\ X &\to a \mid b. \end{aligned}$$

х) $S \to aSbS \mid aS$.

□

Задача 3.19. Проверете дали следните езици са безконтекстни:

а) $\{a^n b^{2n} c^{3n} \mid n \in \mathbb{N}\}$;

б) $\{a^n b^k c^k a^n \mid k \leq n\}$;

в) $\{a^n b^m c^k \mid n < m < k\}$;

г) $\{a^n b^n c^k \mid n \leq k \leq 2n\}$;

д) $\{a^n b^m c^k \mid k = \min\{n, m\}\}$;

е) $\{a^n b^n c^m \mid m \leq n\}$;

ж) $\{a^n b^m c^k \mid k = n \cdot m\}$;

з) $L^\star$, където $L = \{\alpha\alpha^R \mid \alpha \in \{a, b\}^\star\}$;

и) $\{www \mid w \in \{a, b\}^\star\}$;

к) $\{a^{n^2} b^n \mid n \in \mathbb{N}\}$;

л) $\{a^p \mid p \text{ е просто }\}$;

м) $\{\omega \in \{a, b\}^\star \mid \omega = \omega^R\}$;

н) $\{\omega^n \mid \omega \in \{a, b\}^\star \ \&\ N_b(\omega) = 2 \ \&\ n \in \mathbb{N}\}$;

о) $\{\omega c^n \omega^R \mid \omega \in \{a, b\}^\star \ \&\ n = |\omega|\}$;

п) $\{wcx \mid w, x \in \{a, b\}^\star \ \&\ w \text{ е подниз на } x\}$;

р) $\{x_1 \sharp x_2 \sharp \ldots \sharp x_k \mid k \geq 2 \ \&\ x_i \in a^\star \ \&\ (\exists i, j)[i \neq j \ \&\ x_i = x_j]\}$;

с) $\{x_1 \sharp x_2 \sharp \ldots \sharp x_k \mid k \geq 2 \ \&\ x_i \in a^\star \ \&\ (\forall i, j \leq k)[i \neq j \ \leftrightarrow \ x_i \neq x_j]\}$;

т) $\{a^i b^j c^k \mid i, j, k \geq 0 \ \&\ (i = j \vee j = k)\}$;

у) $\{\alpha \in \{a, b, c\}^\star \mid N_a(\alpha) > N_b(\alpha) > N_c(\alpha)\}$;

ф) $\{a,b\}^\star \setminus \{a^n b^n \mid n \in \mathbb{N}\}$;

х) $\{a^n b^m c^k \mid m^2 = 2nk\}$;

ц) $L = \{a^n b^m c^m a^n \mid m,n \in \mathbb{N} \ \& \ n = m + 42\}$;

ч) $L = \{\sharp a \sharp aa \sharp aaa \sharp \cdots \sharp a^{n-1} \sharp a^n \sharp \mid n \geq 1\}$;

ш) $\{a^m b^n c^k \mid m = n \vee n = k \vee m = k\}$;

щ) $\{a^m b^n c^k \mid m \neq n \vee n \neq k \vee m \neq k\}$;

ю) $\{a^m b^n c^k \mid m = n \wedge n = k \wedge m = k\}$;

я) $\{w \in \{a,b,c\}^\star \mid N_a(w) \neq N_b(w) \vee N_a(w) \neq N_c(w) \vee N_b(w) \neq N_c(w)\}$.

Задача 3.20. Докажете, че езикът $L = \{a^n b^{kn} \mid k,n > 0\}$ не е безконтекстен.

Упътване. Да разгледаме ситуацията $\alpha \in L$ и $\alpha = xyuvw$, където $y = a^i$ и $v = b^j$ Интересният случай е когато $0 < i,j < p$.

- Нека $i = j$. Разглеждаме думата $xy^{p^2+1}uv^{p^2+1}w$. Това означава дали $p + p^2 i$ дели $p^2 + p^2 i$, т.е. дали $1 + pi$ дели $p + pi$.

$$p + pi = k(1 + pi), \text{ за някое } 1 \leq k < p$$
$$p = k + pi(k-1), \text{ за някое } 1 \leq k < p$$

  Достигаме до противоречие.

- Нека $i > j$, т.е. $i \geq j + 1$. Отново разглеждаме думата $xy^{p^2+1}uv^{p^2+1}w$. Това означава дали $p + p^2 i$ дели $p^2 + p^2 j$, т.е. дали $1 + pi$ дели $p + pj$, но $1 + pi \geq 1 + p(j+1) > p + pj$. Противоречие.

- Нека $i < j$ и тогава нека $j = mi + r$, където $p > i > r \geq 0$ Разглеждаме думата $xy^{mp^2+1}uv^{mp^2+1}w$. Това означава дали $p + mp^2 i$ дели $p^2 + mp^2 j$, т.е. дали $1 + mpi$ дели $p + mpj$.

$$p + pmj = k(1 + pmi), \text{ за някое } 1 \leq k.$$

  – Възможно ли е $k \geq p$? Тогава:

$$p + pmj = k(1 + pmi) \geq p(1 + pmi) \geq p(1 + pj)$$
$$p(1 + mj) \geq p(1 + pj)$$
$$m \geq p.$$

  Достигаме до противоречие. Следователно, $1 \leq k < p$.

– Възможно ли е $k \leq m$? Тогава:

$$p + pmj = k(1 + pmi) \leq m(1 + pmi) \leq m(1 + pj)$$
$$p(1 + mj) \leq m(1 + pj)$$
$$p + pmj \leq m + pmj$$
$$p \leq m.$$

Достигаме до противоречие, защото $m < p$.

– Заключаваме, че $1 \leq m < k < p$. Тогава:

$$p + pmj = k(1 + pmi)$$
$$p = k + pu(ki - j).$$

Понеже $m < k$, то $j < (m + 1)i \leq ki$, т.е. $ki - j > 0$. Достигаме до противоречие.

□

За всеки две думи с равна дължина, дефинираме функцията `diff` по следния начин:

$$\texttt{diff}(\varepsilon, \varepsilon) = 0$$
$$\texttt{diff}(a \cdot \alpha, b \cdot \beta) = \begin{cases} \texttt{diff}(\alpha, \beta), & \text{ако } a = b \\ 1 + \texttt{diff}(\alpha, \beta), & \text{ако } a \neq b \end{cases}$$

Задача 3.21. За всеки от следните езици, отговорете дали са безконтекстни, като се обосновете:

а) $L \stackrel{\text{деф}}{=} \{\alpha\sharp\beta \mid \alpha, \beta \in \{a, b\}^\star \ \&\ |\alpha| = |\beta| \ \&\ \texttt{diff}(\alpha, \beta^{rev}) = 1\}$; Да

б) $L \stackrel{\text{деф}}{=} \{\alpha\sharp\beta \mid \alpha, \beta \in \{a, b\}^\star \ \&\ |\alpha| = |\beta| \ \&\ \texttt{diff}(\alpha, \beta^{rev}) \geq 1\}$; Да

в) $L \stackrel{\text{деф}}{=} \{\alpha\sharp\beta \mid \alpha, \beta \in \{a, b\}^\star \ \&\ |\alpha| = |\beta| \ \&\ \texttt{diff}(\alpha, \beta) \geq 1\}$; Не

г) $L \stackrel{\text{деф}}{=} \{\alpha\beta \mid \alpha, \beta \in \{a, b\}^\star \ \&\ |\alpha| = |\beta| \ \&\ \texttt{diff}(\alpha, \beta) \geq 1\}$; Да

д) $L \stackrel{\text{деф}}{=} \{\alpha\beta \mid \alpha, \beta \in \{a, b\}^\star \ \&\ |\alpha| = |\beta| \ \&\ \texttt{diff}(\alpha, \beta) = 1\}$; Не

Задача 3.22. Да разгледаме езика

$$D_1 \stackrel{\text{деф}}{=} \{\alpha_1\sharp\alpha_2\sharp\cdots\sharp\alpha_n \mid n \geq 2 \ \&\ |\alpha_i| = |\alpha_{i+1}| \ \&\ \texttt{diff}(\alpha_i, \alpha_{i+1}^{rev}) = 1\}.$$

- Докажете, че D_1 не е безконтекстен.
- Докажете, че D_1 може да се представи като сечението на два безконтекстни езика.

Упътване. Да разгледаме безконтекстния език

$$L_1 = \{\alpha_1 \sharp \alpha_2 \mid |\alpha_1| = |\alpha_2| \ \&\ \texttt{diff}(\alpha_1, \alpha_2^{rev}) = 1\}.$$

Тогава

$$D_1 = L_1 \cdot (\sharp L_1)^\star \cdot (\{\varepsilon\} \cup \sharp\{a,b\}^\star) \cap \\ \{a,b\}^\star \cdot (\sharp L_1)^\star \cdot (\{\varepsilon\} \cup \sharp\{a,b\}^\star).$$

□

Задача 3.23. За произволен език L, дефинираме езика

$$\texttt{Diff}_n(L) = \{\alpha \in L \mid (\exists \beta \in L)[|\alpha| = |\beta| \ \&\ \texttt{diff}(\alpha, \beta) = n]\}.$$

Вярно ли е, че:

- ако L е регулярен, то $\texttt{Diff}_n(L)$ е регулярен?
- ако L е безконтекстен, то $\texttt{Diff}_n(L)$ е безконтекстен?

Упътване. Мисля, че най-лесно става като се разгледа съответно автомата или стековия автомат. □

Задача 3.24. Нека L_1 и L_2 са езици. Дефинираме [Sip12, стр. 158]

$$L_1 \triangle L_2 \stackrel{\text{деф}}{=} \{\alpha\beta \mid \alpha \in L_1 \ \&\ \beta \in L_2 \ \&\ |\alpha| = |\beta|\}.$$

Докажете, че

а) ако L_1 и L_2 са регулярни езици, то е възможно $L_1 \triangle L_2$ да не е регулярен;

б) ако L_1 и L_2 са регулярни езици, то $L_1 \triangle L_2$ е безконтекстен;

в) ако L_1 и L_2 са безконтекстни езици, то е възможно $L_1 \triangle L_2$ да не е безконтекстен.

Задача 3.25. Докажете, че ако L е безконтекстен език, то

$$L^{rev} = \{\omega^{rev} \mid \omega \in L\}$$

също е безконтекстен.

Задача 3.26. Нека $\Sigma = \{a, b, c, d, f, e\}$. Докажете, че езикът L е безконтекстен, където за думите $\omega \in L$ са изпълнени свойствата:

- за всяко $n \in \mathbb{N}$, след всяко срещане на n последователни a-та следват n последователни b-та, и b-та не се срещат по друг повод в ω, и

- за всяко $m \in \mathbb{N}$, след всяко срещане на m последователни c-та следват m последователни d-та, и d-та не се срещат по друг повод в ω, и

- за всяко $k \in \mathbb{N}$, след всяко срещане на k последователни f-а следват k последователни e-та, и e-та не се срещат по друг повод в ω.

Задача 3.27. Да разгледаме езиците:

$$P = \{\alpha \in \{a,b,c\}^* \mid \alpha \text{ е палиндром с четна дължина}\}$$
$$L = \{\beta b^n \mid n \in \mathbb{N}, \beta \in P^n\}.$$

Да се докаже, че:

а) L не е регулярен;

б) L е безконтекстен.

Задача 3.28. Нека L_1 е произволен регулярен език над азбуката Σ, а L_2 е езика от всички думи палиндроми над Σ. Докажете, че L е безконтекстен език, където:

$$L = \{\alpha_1\alpha_2\cdots\alpha_{3n}\beta_1\cdots\beta_m\gamma_1\cdots\gamma_n \mid \alpha_i, \gamma_j \in L_1, \beta_k \in L_2, m, n \in \mathbb{N}\}.$$

Задача 3.29. Нека $L = \{\omega \in \{a,b\}^\star \mid N_a(\omega) = 2\}$. Да се докаже, че езикът $L' = \{\alpha^n \mid \alpha \in L, n \geq 0\}$ не е безконтекстен.

Задача 3.30. Нека $\Sigma = \{a,b,c\}$ и $L \subseteq \Sigma^\star$ е безконтестен език. Ако имаме дума $\alpha \in \Sigma^\star$, тогава *L-вариант* на α ще наричаме думата, която се получава като в α всяко едно срещане на символа a заменим с (евентуално различна) дума от L. Тогава, ако $M \subseteq \Sigma^*$ е произволен безконтестен език, да се докаже че езикът

$$M' = \{\beta \in \Sigma^\star \mid \beta \text{ е } L\text{-вариант на } \alpha \in M\}$$

също е безконтекстен.

Задача 3.31. Докажете, че всеки безконтекстен език над азбуката $\Sigma = \{a\}$ е регулярен.

Упътване.

□

Задача 3.32. Да фиксираме азбуката Σ. Нека L е безконтекстен език, а R е регулярен език. Докажете, че езикът

$$L/R = \{\alpha \in \Sigma^\star \mid (\exists\beta \in R)[\alpha\beta \in L]\}$$

е безконтекстен.

Упътване.

□

Задача 3.33. Нека е дадена граматиката $G = \langle\{a, b\}, \{S, A, B, C\}, S, R\rangle$. Използвайте CYK-алгоритъма, за да проверите дали думата α принадлежи на $\mathcal{L}(G)$, където правилата на граматиката и думата α са зададени като:

а) $S \to BA \mid CA \mid a,\ C \to BS \mid SA,\ A \to a,\ B \to b,$
$\alpha = bbaaa$;

б) $S \to AB \mid BC,\ A \to BA \mid a,\ B \to CC \mid b,\ C \to AB \mid a,$
$\alpha = baaba$;

в) $S \to AB,\ A \to AC \mid a \mid b,\ B \to CB \mid a,\ C \to a,$
$\alpha = babaa$.

Задача 3.34. Нека L е безконтекстен език над азбуката Σ. Докажете, че следните езици са безконтекстни:

а) $\texttt{Pref}(L) = \{\alpha \in \Sigma^\star \mid (\exists\beta \in \Sigma^\star)[\alpha \cdot \beta \in L]\}$;

б) $\texttt{Suff}(L) = \{\beta \in \Sigma^\star \mid (\exists\alpha \in \Sigma^\star)[\alpha \cdot \beta \in L]\}$;

в) $\texttt{Infix}(L) = \{\beta \in \Sigma^\star \mid (\exists\alpha, \gamma \in \Sigma^\star)[\alpha \cdot \beta \cdot \gamma \in L]\}$;

Упътване.

□

Задача 3.35. Докажете, че езикът

$$L = \{\omega_1\sharp\omega_2\sharp\cdots\sharp\omega_{2n} \mid n \in \mathbb{N}\ \&\ \sum_{i=1}^{n}|\omega_{2i-1}| = \sum_{i=1}^{n}|\omega_{2i}|\}$$

е безконтекстен.

Упътване. Най-лесно става със стеков автомат. Интересно е също да се направи и безконтекстна граматика за L. Това дали е вярно:

$$S \to XSX \mid \sharp SS \mid SS\sharp \mid \sharp S\sharp$$
$$X \to a \mid b.$$

□

Задача 3.36. Нека с $\ulcorner\mathcal{A}\urcorner$ да означим думата над азбуката $\{0, 1\}$, която кодира крайния автомат $\mathcal{A}$. Посочете кои от следните езици са регулярни/безконтекстни/разрешими/полуразрешими, където:

а) $L = \{\ulcorner\mathcal{A}\urcorner \mid \mathcal{A}$ е краен автомат$\}$;

б) $L = \{\ulcorner \mathcal{A} \urcorner \mid \mathcal{A}$ е краен автомат с пет състояния$\}$;

в) $L = \{\ulcorner \mathcal{A} \urcorner \mid \mathcal{A}$ е краен детерминиран автомат$\}$;

г) $L = \{\ulcorner \mathcal{A} \urcorner \mid \mathcal{A}$ е краен детерминиран тотален автомат$\}$;

д) $L = \{\ulcorner \mathcal{A} \urcorner \cdot \omega \mid \omega \in \Sigma^\star \ \&\ \omega \in \mathcal{L}(\mathcal{A})\}$;

е) $L = \{\ulcorner \mathcal{A} \urcorner \cdot \omega \mid \omega \in \Sigma^\star \ \&\ \omega \notin \mathcal{L}(\mathcal{A})\}$;

ж) $L = \{\ulcorner \mathcal{A} \urcorner \cdot \ulcorner \mathcal{B} \urcorner \mid \mathcal{A}, \mathcal{B}$ са крайни автомати и $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})\}$;

Обосновете се!

Задача 3.37. Нека $G = \langle V, \Sigma, R, S\rangle$ е *регулярна* граматика, т.е. всички правила на G са от вида $A \to bC$ и $A \to \varepsilon$. Посочете кои от следните езици са безконтекстни, където:

а) $L = \{\alpha\sharp\beta^{rev} \mid \alpha, \beta \in (V \cup \Sigma)^\star \ \&\ \alpha \vdash_G \beta\}$;

б) $L = \{\alpha\sharp\beta^{rev} \mid \alpha, \beta \in (V \cup \Sigma)^\star \ \&\ \alpha \vdash_G^\star \beta\}$;

в) $L = \{\alpha\sharp\beta^{rev} \mid \alpha, \beta \in (V \cup \Sigma)^\star \ \&\ \alpha \nvdash_G \beta\}$;

г) $L = \{\alpha\sharp\beta^{rev} \mid \alpha, \beta \in (V \cup \Sigma)^\star \ \&\ \alpha \nvdash_G^\star \beta\}$.

Обосновете се!

Задача 3.38. Да разгледаме една *безконтекстна* граматика $G = \langle V, \Sigma, R, S\rangle$. Посочете кои от следните езици са безконтекстни, където:

а) $L = \{\alpha\sharp\beta^{rev} \mid \alpha, \beta \in (V \cup \Sigma)^\star \ \&\ \alpha \vdash_G \beta\}$;

б) $L = \{\alpha_1\sharp\alpha_2^{rev}\sharp\alpha_3\sharp\alpha_4^{rev}\sharp\cdots \mid \alpha_i \in (V \cup \Sigma)^\star \ \&\ \alpha_i \vdash_G \alpha_{i+1}$ за $i < n\}$;

в) $L = \{\alpha\sharp\beta^{rev} \mid \alpha, \beta \in (V \cup \Sigma)^\star \ \&\ \alpha \vdash_G^\star \beta\}$;

г) $L = \{\alpha\sharp\beta^{rev} \mid \alpha, \beta \in (V \cup \Sigma)^\star \ \&\ \alpha \nvdash_G \beta\}$;

д) $L = \{\alpha\sharp\beta^{rev} \mid \alpha, \beta \in (V \cup \Sigma)^\star \ \&\ \alpha \nvdash_G^\star \beta\}$.

Обосновете се!

Задача 3.39. Да разгледаме една *неограничена* граматика $G = \langle V, \Sigma, R, S\rangle$. Посочете кои от следните езици са безконтекстни, където:

а) $L = \{\alpha\sharp\beta^{rev} \mid \alpha, \beta \in (V \cup \Sigma)^\star \ \&\ \alpha \vdash_G \beta\}$;

б) $L = \{\alpha\sharp\beta^{rev} \mid \alpha, \beta \in (V \cup \Sigma)^\star \ \&\ \alpha \vdash_G^\star \beta\}$;

в) $L = \{\alpha\sharp\beta^{rev} \mid \alpha, \beta \in (V \cup \Sigma)^\star \ \&\ \alpha \nvdash_G \beta\}$;

г) $L = \{\alpha\sharp\beta^{rev} \mid \alpha, \beta \in (V \cup \Sigma)^\star \ \&\ \alpha \nvdash_G^\star \beta\}$.

Обосновете се!

Глава 4

Машини на Тюринг

Тук най-вече следваме [Sip12, Глава 3]

4.1 Основни понятия

Понятието за машина на Тюринг има много еквивалентни дефиниции

Детерминистична машина на Тюринг ще наричаме седморка от вида

$$\mathcal{M} = \langle Q, \Sigma, \Gamma, \delta, \sqcup, q_{\texttt{start}}, q_{\texttt{accept}}, q_{\texttt{reject}} \rangle,$$

където:

- Q - крайно множество от състояния;
- Σ - крайна азбука за входа;
- Γ - крайна азбука за лентата, $\Sigma \subseteq \Gamma$;
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\triangleleft, \triangleright, \square\}$ - частична функция на преходите; Изискваме $\delta(q_{\texttt{accept}}, x)$ и $\delta(q_{\texttt{reject}}, x)$ да не са дефинирани, за всяка $x \in \Gamma$.
- $q_{\texttt{start}}$ - начално състояние, $q_{\texttt{start}} \in Q$;
- $\sqcup$ - празен символ, $\sqcup \in \Gamma \setminus \Sigma$;
- $q_{\texttt{accept}} \in Q$ - приемащо състояние;
- $q_{\texttt{reject}} \in Q$ - отхвърлящо състояние.

Тези две състояния ще наричаме заключителни

$q_{\texttt{accept}} \neq q_{\texttt{reject}}$

Сега ще опишем как $\mathcal{M}$ работи върху вход думата $\alpha \in \Sigma^\star$. Първоначално, безкрайната лента съдържа само α. Останалите клетки на лентата съдържат $\sqcup$. Освен това, $\mathcal{M}$ се намира в началното състояние s и главата е върху най-левия символ на α. Работата $\mathcal{M}$ е описана от функцията на преходите.

- Формално, **моментната конфигурация** (или описание) на едно изчисление на машина на Тюринг е тройка от вида

$$(\alpha, q, x\beta) \in \Gamma^\star \times Q \times \Gamma^\star,$$

като интерпретацията на тази тройка е, че машината се намира в състояние q и лентата има вида

$$\ldots \sqcup\sqcup\sqcup \alpha\underline{x}\beta \sqcup\sqcup\sqcup \ldots,$$

като четящата глава на машината е поставена върху първия символ на β. Понякога за удобство ще означаваме моментната конфигурация като $(q, \alpha\underline{x}\beta)$.

- Ако $\beta = \varepsilon$, това означава, че главата на машината е върху $\sqcup$, т.е. лентата има вида

$$\ldots \sqcup\sqcup\sqcup \alpha\underline{\sqcup}\sqcup\sqcup\sqcup \ldots,$$

- **Началната конфигурация** за входа $\alpha \in \Sigma^\star$ представлява

$$(\varepsilon, q_{\texttt{start}}, \alpha).$$

Това означава, че лентата има вида

$$\cdots \sqcup\sqcup\underline{x}\alpha \sqcup\sqcup \cdots,$$

където $\alpha = x\alpha'$, ако $\alpha \in \Sigma^+$.

- Ако $\alpha = \varepsilon$, то началната конфигурация е $(\varepsilon, q_{\texttt{start}}, \varepsilon)$ и лентата има вида

$$\ldots \sqcup\sqcup\sqcup\underline{\sqcup}\sqcup\sqcup\sqcup \ldots.$$

- **Заключителна конфигурация** представлява тройка от вида

$$(\alpha, q_{\texttt{accept}}, \beta), \text{ или } (\alpha, q_{\texttt{reject}}, \beta).$$

Ако машината, която работи върху дадена входа дума, достигне до заключително състояние, ще казваме че машината *спира работа*.

Както за автомати, удобно е да дефинираме бинарна релация $\vdash_{\mathcal{M}}$ над $\Gamma^\star \times Q \times \Gamma^\star$, която ще казва как моментната конфигурация на машината $\mathcal{M}$ се променя при изпълнение на една стъпка.

- Ако $\delta_{\mathcal{M}}(q, z) = (p, y, \triangleright)$, то дефинираме $(\alpha, q, z\beta) \vdash_{\mathcal{M}} (\alpha y, p, \beta)$. Това означава, че ако лентата е имала вида

$$\ldots \sqcup\sqcup\sqcup \alpha\underline{z}\beta \sqcup\sqcup\sqcup \ldots,$$

след тази стъпка лентата има вида

$$\ldots \sqcup \sqcup \sqcup \, \alpha y \underline{b} \beta' \, \sqcup \sqcup \sqcup \ldots,$$

ако $\beta = b\beta'$, или

$$\ldots \sqcup \sqcup \sqcup \, \alpha y \underline{\sqcup} \sqcup \sqcup \sqcup \ldots,$$

ако $\beta = \varepsilon$.

- Ако $\delta_{\mathcal{M}}(q, z) = (p, y, \lhd)$, то дефинираме $(\alpha x, q, z\beta) \vdash_{\mathcal{M}} (\alpha, p, xy\beta)$. Това означава, че ако лентата е имала вида

  $$\ldots \sqcup \sqcup \sqcup \, \alpha x \underline{z} \beta \, \sqcup \sqcup \sqcup \ldots,$$

  след тази стъпка лентата има вида

  $$\ldots \sqcup \sqcup \sqcup \, \alpha \underline{x} y \beta \, \sqcup \sqcup \sqcup \ldots$$

- Ако $\delta_{\mathcal{M}}(q, z) = (p, y, \square)$, то дефинираме $(\alpha, q, z\beta) \vdash_{\mathcal{M}} (\alpha, p, y\beta)$. Това означава, че ако лентата е имала вида

  $$\ldots \sqcup \sqcup \sqcup \, \alpha \underline{z} \beta \, \sqcup \sqcup \sqcup \ldots,$$

  след тази стъпка лентата има вида

  $$\ldots \sqcup \sqcup \sqcup \, \alpha \underline{y} \beta \, \sqcup \sqcup \sqcup \ldots$$

С $\vdash^{\star}_{\mathcal{M}}$ ще означаваме рефлексивното и транзитивно затваряне на $\vdash_{\mathcal{M}}$.

- Машината на Тюринг $\mathcal{M}$ **приема** думата α, ако

  $$(\varepsilon, q_{\texttt{start}}, \alpha) \vdash^{\star}_{\mathcal{M}} (\gamma_1, q_{\texttt{accept}}, \gamma_2),$$

  за някои $\gamma_1, \gamma_2 \in \Gamma^{\star}$.

- Машината на Тюринг $\mathcal{M}$ **отхвърля** думата α, ако

  $$(\varepsilon, q_{\texttt{start}}, \alpha) \vdash^{\star}_{\mathcal{M}} (\gamma_1, q_{\texttt{reject}}, \gamma_2),$$

  за някои $\gamma_1, \gamma_2 \in \Gamma^{\star}$.

- Машината на Тюринг $\mathcal{M}$ **не приема** думата α, ако $\mathcal{M}$ отхвърля α или $\mathcal{M}$ никога не завършва при начална конфигурация $(\varepsilon, q_{\texttt{start}}, \alpha)$.

- Една машина на Тюринг се нарича **тотална**, ако при всеки вход достига до заключително състояние, т.е. достига до $q_{\texttt{accept}}$ или $q_{\texttt{reject}}$. Обърнете внимание, че това не е същото като да изискваме функцията на преходите δ да бъде тотална.

Това понятие не е стандартно

- Езикът, който се **разпознава** от машината $\mathcal{M}$ е:

$$\mathcal{L}(\mathcal{M}) = \{\alpha \in \Sigma^\star \mid (\varepsilon, q_{\texttt{start}}, \alpha) \vdash^\star_{\mathcal{M}} (\beta, q_{\texttt{accept}}, \gamma), \text{ за някои } \beta, \gamma \in \Gamma^\star\}.$$

- Езикът L се нарича **полуразрешим**, ако съществува машина на Тюринг $\mathcal{M}$, за която $L = \mathcal{L}(\mathcal{M})$. В този случай се казва, че $\mathcal{M}$ разпознава езика L.

- Един език L се нарича **разрешим**, ако за него съществува *тотална* машина на Тюринг $\mathcal{M}$, за която $L = \mathcal{L}(\mathcal{M})$. В този случай се казва, че $\mathcal{M}$ разрешава езика L.

Твърдение 4.1. Ако L е разрешим език над азбуката Σ, то $\Sigma^\star \setminus L$ също е разрешим език.

Забележка. По-късно, ще видим, че съществуват полуразрешими езици, чиито допълнения не са полуразрешими.

4.2 Примери за разрешими езици

Пример 4.1. Да разгледаме езика $L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$.

Знаем, че L не е безконтекстен

Нека да въведем нов символ x, с който ще маркираме обработените символи a, b, c. Идеята на алгоритъма, който ще разгледаме е да маркира на всяка итерация по едно a, b, и c. Той завършва успешно ако всички символи на думата са маркирани. Нека първоначално думата е копирана върху лентата и четящата глава е върху първия символ на думата.

(1) Чете x-та надясно по лентата докато срещне първото a и го замества с x. Отива на стъпка (2). Ако символите свършат (т.е. достигне се $\sqcup$) преди да се достигне a, то алгоритъмът завършва успешно.

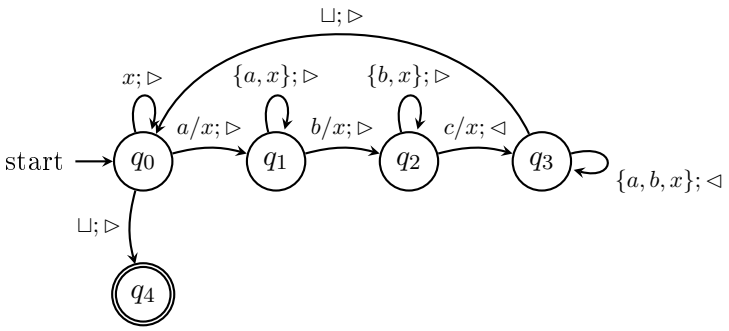
(2) Чете x-та надясно по лентата докато срещне първото b и го замества с x. Отива на стъпка (3).

(3) Чете x-та надясно по лентата докато срещне първото c и го замества с x.

(4) Връща четящата глава в началото на лентата, т.е. чете наляво докато не срещне символа $\sqcup$. Връща се в стъпка (1).

Нека сега да видим, че този алгоритъм може да се опише съвсем формално с машина на Тюринг. Ще построим машина на Тюринг $\mathcal{M}$, за която $L = \mathcal{L}(\mathcal{M})$, където

- $\Sigma = \{a, b, c\}$;
- $\Gamma = \{a, b, c, x, \sqcup\}$, за някой нов символ x;
- $Q = \{q_0, q_1, \ldots, q_4\}$;
- $q_{\texttt{start}} = q_0$;
- $q_{\texttt{accept}} = q_4$;
- частичната функция на преходите $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\triangleleft, \triangleright, \square\}$ е описана на схемата отдолу.

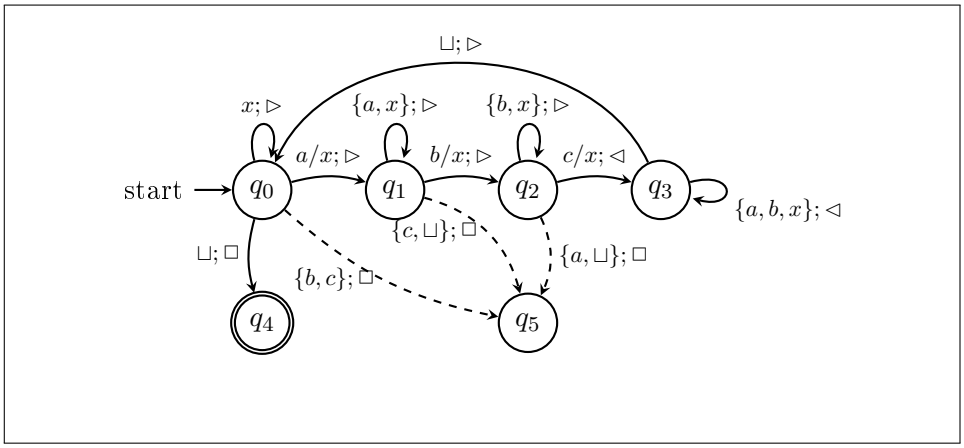


Фигура 4.1: детерминистична частична машина на Тюринг $\mathcal{M}$, за която $\mathcal{L}(\mathcal{M}) = \{a^n b^n c^n \mid n \in \mathbb{N}\}$

Например,

$$\delta(q_0, a) = (q_1, x, \triangleright)$$
$$\delta(q_3, \sqcup) = (q_0, \sqcup, \triangleright)$$
$$\delta(q_1, a) = (q_1, a, \triangleright).$$

Съобразете, че тази машина на Тюринг може да се направи тотална като се добави ново състояние $q_{\texttt{reject}}$ и за всяка двойка (q, z), за която функцията на преходите не е дефинирана, да сочи към $q_{\texttt{reject}} = q_6$. Така можем да получим *тотална* машина на Тюринг за езика L, което означава, че L е не само полуразрешим, но *разрешим* език.



Пример 4.2. Да разгледаме езика $L = \{\omega\sharp\omega \mid \omega \in \{a,b\}^\star\}$. Нека първо да видим, че можем неформално да опишем алгоритъм, който да разпознава думите на езика L. Нека една дума е копирана върху лентата и четящата глава е поставена върху първия символ от думата.

Да напомним, че този език не е безконтекстен

В [HU79, стр. 155] е дадено по-различно решение. Тук следваме [Sip12, стр. 173]. Там има малка грешка

(1) Чете x-ове надясно по лентата докато не срещне a или b и го замества с x. Запомня дали сме срещнали a или b. Ако вместо a или b срещне $\sharp$, то отива на стъпка (6).

(2) Чете a-та и b-та надясно по лентата докато не стигне $\sharp$.

(3) Чете c-то надясно по лентата и всички следващи x-ове докато не срещне символа a или b. Той трябва да е същия символ, който сме запаметили на стъпка (1). Заместваме този символ с x.

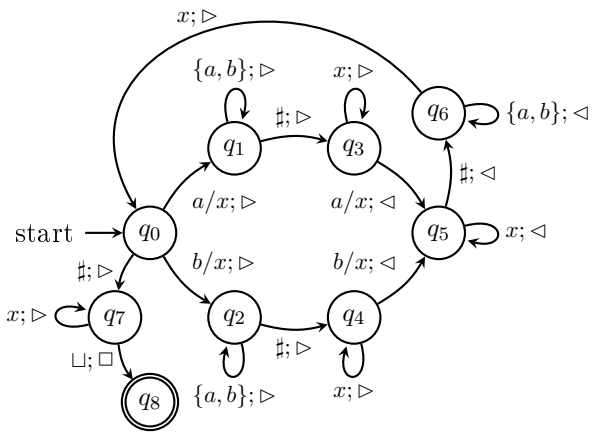
(4) Чете x-ове наляво по лентата докато не стигне $\sharp$.

(5) Чете a-та и b-та по лентата докато не стигне x. Поставя четящата глава върху символа точно след първия x. Отива на стъпка (1).

(6) Прочита $\sharp$ надясно по лентата и чете надясно x-ове докато не срещне $\sqcup$. Алгоритъмът завършва успешно.

Ще построим машина на Тюринг $\mathcal{M}$, за която $L = \mathcal{L}(\mathcal{M})$.

- $\Sigma = \{a, b, \sharp\}$;
- $\Gamma = \{a, b, \sharp, x, \sqcup\}$;
- $Q = \{q_0, q_1, \ldots, q_8\}$;
- $q_{\texttt{start}} = q_0$;
- $q_{\texttt{accept}} = q_8$;



Фигура 4.2: детерминистична частична машина на г-н Тюринг $\mathcal{M}$, за която $\mathcal{L}(\mathcal{M}) = \{\omega\sharp\omega \mid \omega \in \{a,b\}^\star\}$

Да проследим изчислението на думата $ab\sharp ab$.

$$\begin{aligned}(q_1, \underline{a}b\sharp ab) &\vdash (q_2, x\underline{b}\sharp ab) \vdash (q_2, xb\underline{\sharp}ab) \vdash (q_4, xb\sharp\underline{a}b) \vdash (q_6, xb\underline{\sharp}xb)\\ &\vdash (q_7, x\underline{b}\sharp xb) \vdash (q_7, \underline{x}b\sharp xb) \vdash (q_1, x\underline{b}\sharp xb) \vdash (q_3, xx\underline{\sharp}xb)\\ &\vdash (q_5, xx\sharp\underline{x}b) \vdash (q_5, xx\sharp x\underline{b}) \vdash (q_6, xx\sharp\underline{x}x) \vdash (q_6, xx\underline{\sharp}xx)\\ &\vdash (q_7, x\underline{x}\sharp xx) \vdash (q_1, xx\underline{\sharp}xx) \vdash (q_8, xx\sharp\underline{x}x) \vdash (q_8, xx\sharp x\underline{x})\\ &\vdash (q_8, xx\sharp xx\underline{\sqcup}) \vdash (q_9, xx\sharp xx\underline{\sqcup}).\end{aligned}$$

Може лесно да се съобрази, че тази машина на Тюринг може да се допълни до *тотална*.

4.3 Изчислими функции

Една *тотална* функция $f : \Sigma^\star \to \Sigma^\star$ се нарича изчислима с машина на Тюринг $\mathcal{M}$, ако за всяка дума $\alpha \in \Sigma^\star$,

$$(\varepsilon, q_{\texttt{start}}, \alpha) \vdash^\star_{\mathcal{M}} (\varepsilon, q_{\texttt{accept}}, f(\alpha)).$$

Това означава, че машината на Тюринг $\mathcal{M}$ е тотална.

Лесно може да се съобрази, че тогава езикът

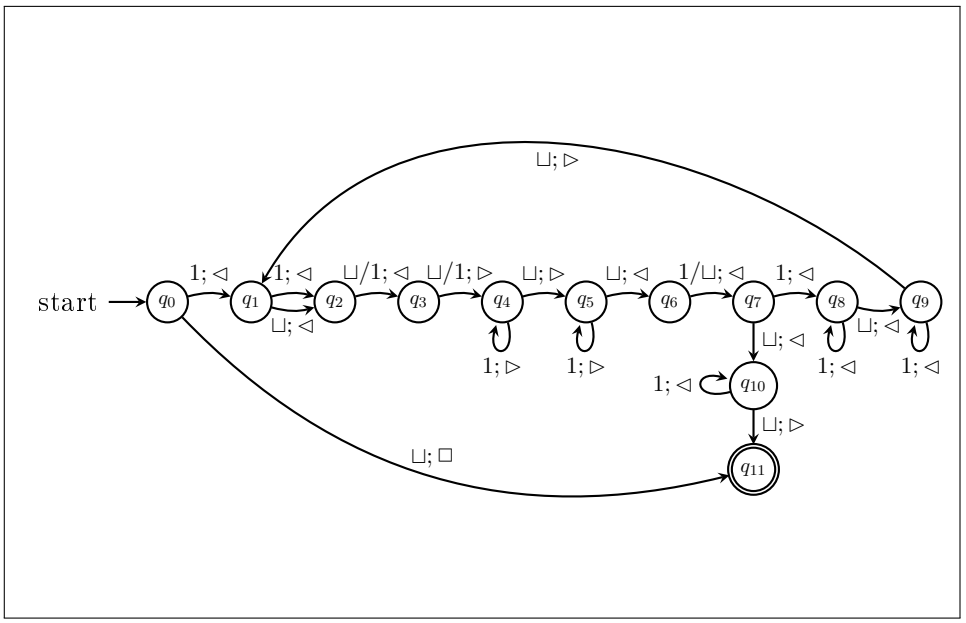
$$Graph(f) = \{\alpha\sharp f(\alpha) \mid \alpha \in \Sigma^\star\}$$

е разрешим.

Задача 4.1. Докажете, че съществуват функции от вида $f : \Sigma^\star \to \Sigma^\star$, които не са изчислими с машина на Тюринг.

Упътване. Всяка машина на Тюринг може да се кодира с естествено число. Това означава, че съществуват изброимо безкрайно много машини на Тюринг. От друга страна, съществуват неизброимо много функции от вида $f : \Sigma^\star \to \Sigma^\star$. □

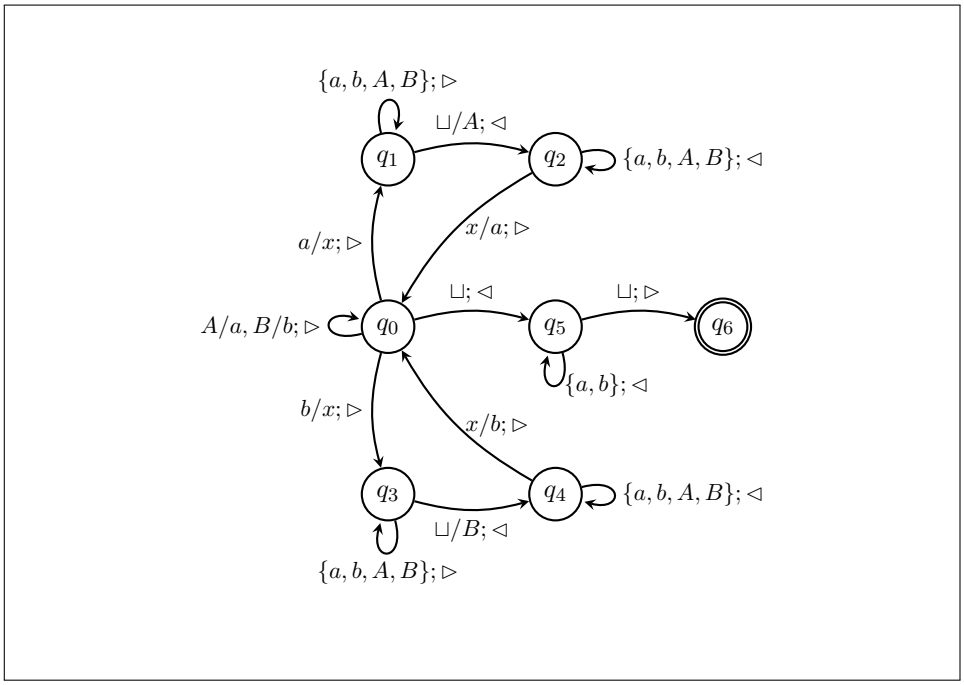
Пример 4.3. Да разгледаме функцията $f : \{1\}^\star \to \{1\}^\star$ дефинирана като $|f(\alpha)| = 2|\alpha|$.



$$(q_0, \underline{1}1) \vdash (q_1, \underline{\sqcup}11) \vdash (q_2, \underline{\sqcup} \sqcup 11) \vdash (q_3, \underline{\sqcup}1 \sqcup 11) \vdash (q_4, 1\underline{1} \sqcup 11) \vdash (q_4, 11\underline{\sqcup}11)$$
$$\vdash (q_5, 11 \sqcup \underline{1}1) \vdash \cdots \vdash (q_7, 11 \sqcup \underline{1}\sqcup) \vdash \cdots$$

Пример 4.4. Да видим защо тоталната функция $f : \{a, b\}^\star \to \{a, b\}^\star$, дефинирана като $f(\alpha) = \alpha \cdot \alpha$ е изчислима с машина на Тюринг.

- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, b, x, A, B\}$;
- $q_{\texttt{start}} = q_0$;
- $q_{\texttt{accept}} = q_6$



Да проследим работата на $\mathcal{M}$ върху думата ab:

$$\begin{aligned}(q_0, \underline{a}b) &\vdash (q_1, x\underline{b}) \vdash (q_1, xb\underline{\sqcup}) \vdash (q_2, x\underline{b}A) \vdash (q_2, \underline{x}bA)\\ &\vdash (q_0, a\underline{b}A) \vdash (q_3, ax\underline{A}) \vdash (q_3, axA\underline{\sqcup}) \vdash (q_4, ax\underline{A}B)\\ &\vdash (q_4, a\underline{x}AB) \vdash (q_0, ab\underline{A}B) \vdash (q_0, aba\underline{B}) \vdash (q_0, abab\underline{\sqcup})\\ &\vdash (q_5, aba\underline{b}) \vdash (q_5, ab\underline{a}b) \vdash (q_5, a\underline{b}ab) \vdash (q_5, \underline{a}bab)\\ &\vdash (q_5, \underline{\sqcup}abab) \vdash (q_6, \underline{a}bab).\end{aligned}$$

Да се каже нещо за графиката ? За момента може би е добре да се казва нищо за графиката. Да се разглеждат само тотални функции.

Пример 4.5. Да разгледаме тоталната функция

Изискваме $f(\alpha)$ да започва с 1 за да може f да бъде функция

$$f : \{0,1\}^\star \to 1 \cdot \{0,1\}^\star,$$

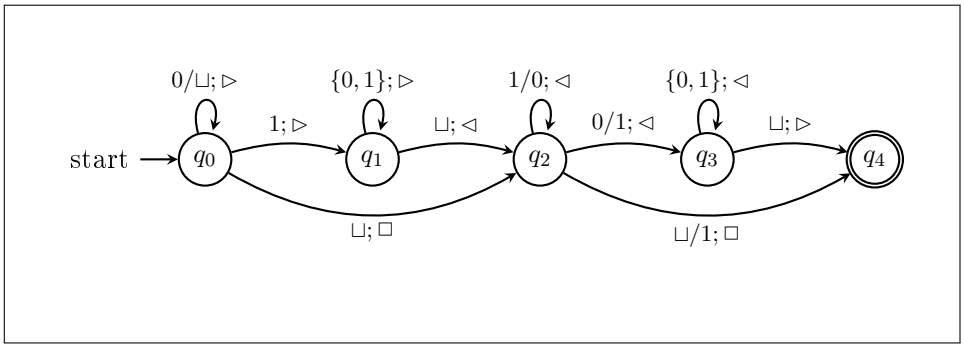
дефинирана като

$$(f(\alpha))_2 = (\alpha)_2 + 1.$$

Нека да видим, че тази функция е изчислима с машина на Тюринг.

- $\Sigma = \{0,1\}$;
- $\Gamma = \{0,1,\sqcup\}$;

- $q_{\texttt{start}} = q_0$;
- $q_{\texttt{accept}} = q_4$.



Да проследим изчислението на $\mathcal{M}$ върху вход 01011.

$$\begin{aligned}(q_0, 0\underline{1}011) &\vdash (q_0, \underline{1}011) \vdash (q_1, 1\underline{0}11) \vdash (q_1, 10\underline{1}1) \vdash (q_1, 101\underline{1}) \vdash (q_1, 1011\sqcup) \\ &\vdash (q_2, 101\underline{1}) \vdash (q_2, 10\underline{1}0) \vdash (q_2, 1\underline{0}00) \vdash (q_3, \underline{1}100) \vdash (q_3, \sqcup 1100) \\ &\vdash (q_4, \underline{1}100).\end{aligned}$$

Задача 4.2. Да разгледаме азбуката $\Sigma = \{0, 1, \ldots, k-1\}$, където $k > 2$. Да разгледаме тоталната функция

$$f : \Sigma^\star \to (\Sigma \setminus \{0\}) \cdot \Sigma^\star,$$

дефинирана като

$$(f(\alpha))_k = (\alpha)_k + 1.$$

Дефинирайте машина на Тюринг $\mathcal{M}$, която изчислява функцията f.

4.4 Многолентови машини на Тюринг

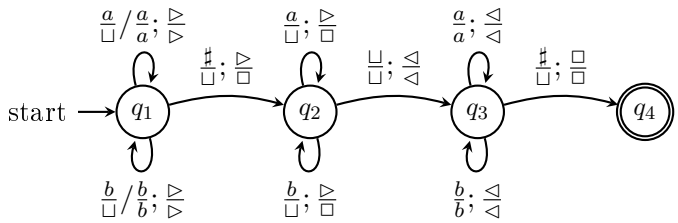
Машина на Тюринг с k ленти има същата дефиниция като еднолентова машина на Тюринг с единствената разлика, че

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{\lhd, \rhd, \square\}^k.$$

Пример 4.6. Да видим как двулентова машина на Тюринг разрешава езика $L = \{\omega\sharp\omega \mid \omega \in \{a, b\}^\star\}$.

- $\Sigma \stackrel{\text{деф}}{=} \{a, b, \sharp\}$;
- $\Gamma \stackrel{\text{деф}}{=} \{a, b, \sharp, \sqcup\}$

- $\delta : Q \times \Gamma^2 \to Q \times \Gamma^2 \times \{\triangleleft, \triangleright, \square\}^2$;



Фигура 4.3: двулентова детерминистична частична машина на Тюринг $\mathcal{M}$, за която $\mathcal{L}(\mathcal{M}) = \{\omega\sharp\omega \mid \omega \in \{a,b\}^\star\}$

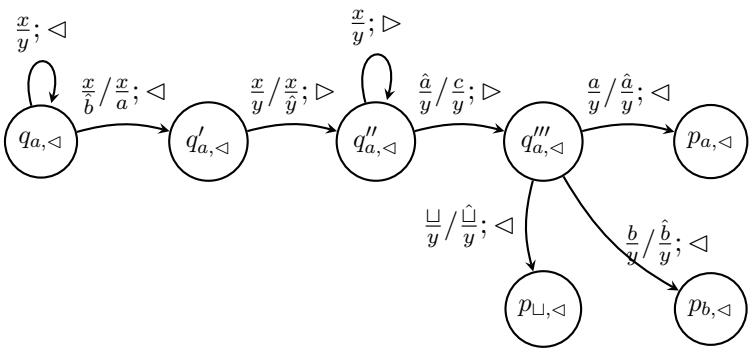
В началото втората лента е празна. Имаме две глави, които се движат независимо една от друга.

$$\begin{aligned}(q_1, \tfrac{\hat{a}}{\hat{\sqcup}}\tfrac{b}{\sqcup}\tfrac{\sharp}{\sqcup}\tfrac{a}{\sqcup}\tfrac{b}{\sqcup}) &\vdash (q_1, \tfrac{a}{a}\tfrac{\hat{b}}{\hat{\sqcup}}\tfrac{\sharp}{\sqcup}\tfrac{a}{\sqcup}\tfrac{b}{\sqcup}) \vdash (q_1, \tfrac{a}{a}\tfrac{b}{b}\tfrac{\hat{\sharp}}{\hat{\sqcup}}\tfrac{a}{\sqcup}\tfrac{b}{\sqcup}) \vdash (q_2, \tfrac{a}{a}\tfrac{b}{b}\tfrac{\sharp}{\hat{\sqcup}}\tfrac{\hat{a}}{\sqcup}\tfrac{b}{\sqcup}) \\ &\vdash (q_2, \tfrac{a}{a}\tfrac{b}{b}\tfrac{\sharp}{\hat{\sqcup}}\tfrac{a}{\sqcup}\tfrac{\hat{b}}{\sqcup}) \vdash (q_2, \tfrac{a}{a}\tfrac{b}{b}\tfrac{\sharp}{\hat{\sqcup}}\tfrac{a}{\sqcup}\tfrac{b}{\sqcup}\tfrac{\hat{\sqcup}}{\sqcup}) \\ &\vdash (q_3, \tfrac{a}{a}\tfrac{b}{\hat{b}}\tfrac{\sharp}{\sqcup}\tfrac{a}{\sqcup}\tfrac{b}{\sqcup}) \vdash (q_3, \tfrac{a}{\hat{a}}\tfrac{b}{b}\tfrac{\sharp}{\sqcup}\tfrac{a}{\sqcup}\tfrac{b}{\sqcup}) \vdash (q_3, \tfrac{\sqcup}{\hat{\sqcup}}\tfrac{a}{a}\tfrac{b}{b}\tfrac{\hat{\sharp}}{\sqcup}\tfrac{a}{\sqcup}\tfrac{b}{\sqcup}) \\ &\vdash (q_4, \tfrac{\sqcup}{\hat{\sqcup}}\tfrac{a}{a}\tfrac{b}{b}\tfrac{\hat{\sharp}}{\sqcup}\tfrac{a}{\sqcup}\tfrac{b}{\sqcup}).\end{aligned}$$

Твърдение 4.2. За всяка k-лентова машина на Тюринг $\mathcal{M}$ съществува еднолентова машина на Тюринг $\mathcal{M}'$, такава че $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{M}')$.

Доказателство. Нека $\mathcal{M}$ е k-лентова машина на Тюринг. Ще построим еднолентова машина на Тюринг $\mathcal{M}'$, за която $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{M}')$. Да означим $\hat{\Gamma} = \{\hat{x} \mid x \in \Gamma\}$. Тогава азбуката на лентата на $\mathcal{M}'$ ще бъде $\Gamma' = (\hat{\Gamma} \cup \Gamma)^k$. Сега вместо да имаме k ленти ще имаме една лента, която представлява k-орка. За да симулираме $\mathcal{M}$, използваме символите $\hat{x}$ за да маркират позицията на главите на $\mathcal{M}$, като във всяка компонента на лентата има точно по един символ от вида $\hat{x}$. За да определим следващия ход на машината $\mathcal{M}'$, трябва да сканираме лентата докато не открием разположението на всичките k на брой маркирани клетки. Тогава симулираме ход на $\mathcal{M}$ и отново трябва да променим маркираните клетки. □

В [Sip12, стр. 177] конструкцията е малко по-различна. Там съдържанието на всяка лента се поставя последователно върху една лента, като се разделят със специален символ. Тук следваме [HU79, стр. 162]



Фигура 4.4: Симулация на прехода $\delta_{\mathcal{M}}(q, \frac{a}{b}) = (p, \frac{b}{a}, \frac{\triangleright}{\triangleleft})$

- В еднолентовата машина на Тюринг, за удобство пишем $\frac{a}{b}$ вместо (a, b).
- Когато сме в състояние от вида $q_{a,\triangleleft}$ означава, че се намираме в състояние q на симулираната двулентова машина на Тюринг $\mathcal{M}$, главата на първата лента е върху символа a и главата на втората лента е разположена наляво от главата на първата лента.
- Аналалогично, когато сме в състояние от вида $q_{a,\square}$ означава, че двете глави са една върху друга в $\mathcal{M}$, а $q_{a,\triangleright}$ означава, че втората глава е отдясно на първата.
- $\frac{x}{y}$ е съкратен запис за $\{\frac{x}{y} \mid x, y \in \Gamma\}$;
- Възможно е на всяка симулирана стъпка, главите на двете ленти да се раздаличават. Това означава, че след симулацията на s стъпки, в най-лошия случай, двете глави са на разстояние $2s$. Това означава, че за да симулираме $(s+1)$-вата стъпка на $\mathcal{M}$, първо трябва да отидем $2s$ стъпки наляво, после $2s$ стъпки надясно. Следователно $(s+1)$-вата стъпка на $\mathcal{M}$ се симулира за $\approx 2s$ стъпки.
- Ако изчислението на $\mathcal{M}$ върху вход α отнема s стъпки, то симулираното изчисление върху α ще отнеме в най-лошия случай приблизително $\sum_{i=0}^{s} 4i = 2s^2 + 2s$ стъпки. Заключаваме, че симулацията на $\mathcal{M}$ е с времева сложност $\mathcal{O}(n^2)$.

4.5 Недетерминистични машини на Тюринг

Една машина на Тюринг $\mathcal{N}$ се нарича недетерминистична, ако функцията на преходите има вида

$$\Delta_{\mathcal{N}} : Q \times \Gamma \to \mathscr{P}(Q \times \Gamma \times \{\lhd, \rhd, \square\}).$$

Отново можем да дефинираме бинарна релация $\vdash_{\mathcal{N}}$ над $\Gamma^\star \times Q \times \Gamma^\star$, която ще казва как моментното описание на машината $\mathcal{N}$ се променя при изпълнение на една стъпка.

- Ако $\Delta_{\mathcal{N}}(q, z) \ni (p, y, \rhd)$, то дефинираме $(\alpha, q, z\beta) \vdash_{\mathcal{N}} (\alpha y, p, \beta)$.
- Ако $\Delta_{\mathcal{N}}(q, z) \ni (p, y, \lhd)$, то дефинираме $(\alpha x, q, z\beta) \vdash_{\mathcal{N}} (\alpha, p, xy\beta)$.
- Ако $\Delta_{\mathcal{N}}(q, z) \ni (p, y, \square)$, то дефинираме $(\alpha, q, z\beta) \vdash_{\mathcal{N}} (\alpha, p, y\beta)$.

С $\vdash^\star_{\mathcal{N}}$ ще означаваме рефлексивното и транзитивно затваряне на $\vdash_{\mathcal{N}}$.

Тогава за недетерминистична машина на Тюринг $\mathcal{N}$,

$$\mathcal{L}(\mathcal{N}) = \{\alpha \in \Sigma^\star \mid (\varepsilon, q_{\texttt{start}}, \alpha) \vdash^\star_{\mathcal{N}} (\beta, q_{\texttt{accept}}, \gamma), \text{ за някои } \beta, \gamma \in \Gamma^\star\}.$$

Забележка. Върху дадена дума ω, недетерминистичната машина на Тюринг $\mathcal{N}$ може да има много различни изчисления. Думата ω принадлежи на $\mathcal{L}(\mathcal{N})$ ако съществува *поне едно* изчисление, което завършва в състоянието $q_{\texttt{accept}}$. Възможно е много други изчисления за ω да завършват в $q_{\texttt{reject}}$ или да зациклят.

Аналогично, дефинираме една недетерминистична машина на Тюринг $\mathcal{N}$ да бъде **тотална**, ако за всяка дума и всяко изчисление на $\mathcal{N}$ върху ω завършва в $q_{\texttt{accept}}$ или $q_{\texttt{reject}}$.

Задача 4.3. $\mathcal{N} = (\{q_0, q_1, q_2, q_f\}, \{0, 1\}, \{0, 1, \sqcup\}, \Delta, q_0, \{q_f\})$, [HMU01]

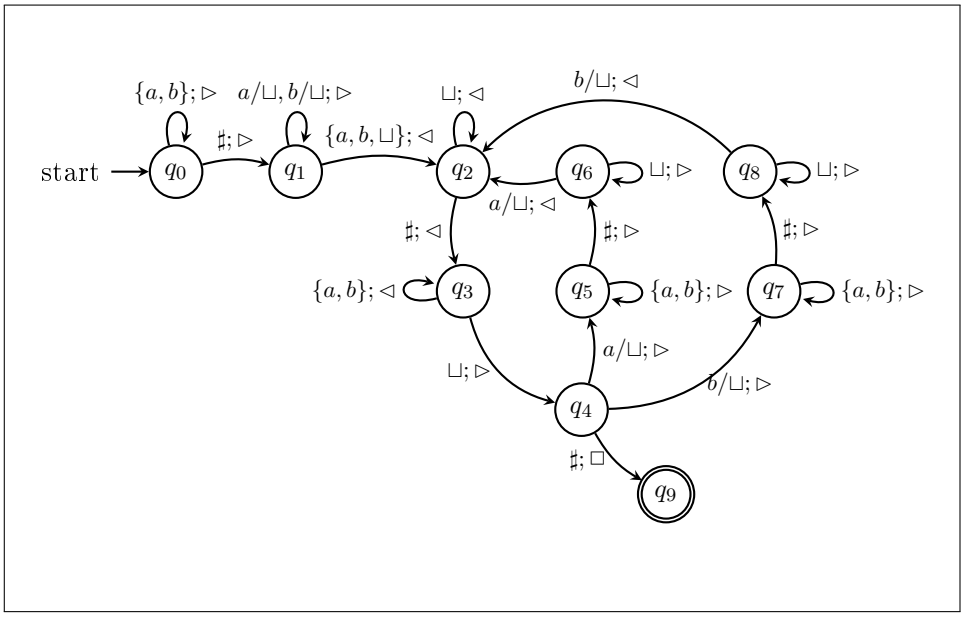
- $\Delta(q_0, 0) = \{(q_0, 1, \rhd), (q_1, 1, \rhd)\}$;
- $\Delta(q_1, 1) = \{(q_2, 0, \lhd)\}$;
- $\Delta(q_2, 1) = \{(q_0, 1, \rhd)\}$;
- $\Delta(q_1, \sqcup) = \{(q_f, \sqcup, \rhd)\}$.

Опишете $\mathcal{L}(\mathcal{N})$. $\{0^{n+1}1^k \mid n, k \in \mathbb{N}\}$

Пример 4.7. Да разгледаме езика

$$L = \{\alpha\sharp\beta \mid \alpha, \beta \in \{a, b\}^\star \ \&\ \alpha \text{ е подниз на } \beta\}.$$

Ще видим, че този език е разрешим като построим недетерминистична машина на Тюринг $\mathcal{N}$, която разрешава този език.



Да видим, че $\mathcal{M}$ успешно разпознава, че думата $ab\sharp aabb$ принадлежи на езика L.

$$
\begin{aligned}
(q_0, \underline{a}b\sharp aabb) &\vdash (q_0, a\underline{b}\sharp aabb) \vdash (q_0, ab\underline{\sharp}aabb) \vdash (q_1, ab\sharp\underline{a}abb) \vdash (q_1, ab\sharp \sqcup \underline{a}bb) \\
&\vdash (q_2, ab\sharp\underline{\sqcup}abb) \vdash (q_2, ab\underline{\sharp} \sqcup abb) \vdash (q_3, a\underline{b}\sharp \sqcup abb) \vdash (q_3, \underline{a}b\sharp \sqcup abb) \\
&\vdash (q_3, \underline{\sqcup}ab\sharp \sqcup abb) \vdash (q_4, \underline{a}b\sharp \sqcup abb) \vdash (q_5, \sqcup\underline{b}\sharp \sqcup abb) \vdash (q_5, \sqcup b\underline{\sharp} \sqcup abb) \\
&\vdash (q_6, \sqcup b\sharp\underline{\sqcup}abb) \vdash (q_6, \sqcup b\sharp \sqcup \underline{a}bb) \vdash (q_2, \sqcup b\sharp\underline{\sqcup} \sqcup bb) \vdash (q_2, \sqcup b\underline{\sharp} \sqcup \sqcup bb) \\
&\vdash (q_3, \sqcup\underline{b}\sharp \sqcup \sqcup bb) \vdash (q_3, \underline{\sqcup}b\sharp \sqcup \sqcup bb) \vdash (q_4, \sqcup\underline{b}\sharp \sqcup \sqcup bb) \vdash (q_7, \sqcup \sqcup \underline{\sharp} \sqcup \sqcup bb) \\
&\vdash (q_8, \sqcup \sqcup \sharp\underline{\sqcup} \sqcup bb) \vdash (q_8, \sqcup \sqcup \sharp \sqcup \underline{\sqcup}bb) \vdash (q_8, \sqcup \sqcup \sharp \sqcup \sqcup\underline{b}b) \\
&\vdash (q_2, \sqcup \sqcup \sharp \sqcup\underline{\sqcup} \sqcup b) \vdash \cdots \vdash (q_4, \sqcup \sqcup \underline{\sharp} \sqcup \sqcup \sqcup b) \vdash (q_9, \sqcup \sqcup \underline{\sharp} \sqcup \sqcup \sqcup b)
\end{aligned}
$$

Канонична подредба на $\Sigma^\star$

За доказателството, че всяка НМТ е еквивалентна на ДМТ, е необходимо да фиксираме канонична подредба на думите над дадена азбука

Нека $\Sigma = \{a_0, a_1, \ldots, a_{k-1}\}$. Подреждаме думите по ред на тяхната дължина. Думите с еднаква дължина подреждаме по техния числов ред, т.е. гледаме на буквите a_i като числото i в k-ична бройна система. Тогава думите с дължина n са числата от 0 до $k^n - 1$ записани в k-ична бройна система. Ще означаваме с ω_i i-тата дума в $\Sigma^\star$ при тази подредба.

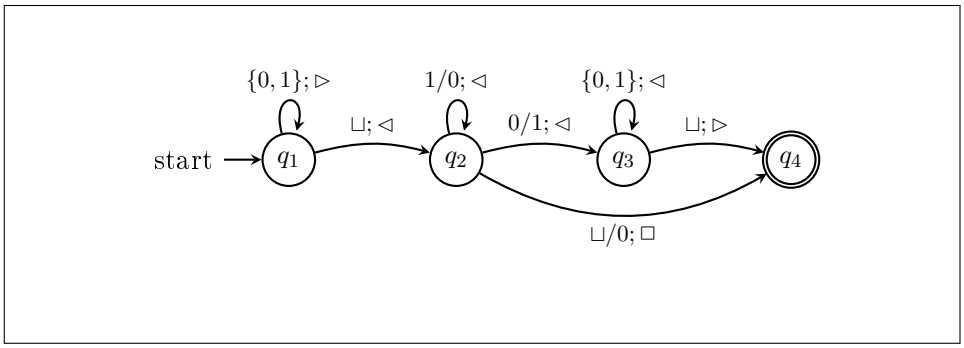
Пример 4.8. Ако $\Sigma = \{0, 1\}$, то наредбата започва така:

$$\varepsilon, 0, 1, \underbrace{00, 01, 10, 11}_{\text{от 0 до 3}}, \underbrace{000, 001, 010, 011, 100, 101, 110, 111}_{\text{от 0 до 7}}, 0000, 0001, \ldots$$

В този случай, $\omega_0 = \varepsilon$, $\omega_7 = 000$, $\omega_{13} = 110$.

Задача 4.4. Нека $\Sigma = \{a_0, \ldots, a_{k-1}\}$. Да разгледаме функцията $f : \Sigma^\star \to \Sigma^\star$, за която $f(\alpha)$ е думата веднага след α в каноничната подредба на $\Sigma^\star$. Докажете, че f е изчислима с машина на Тюринг.

Упътване. Ако $\Sigma = \{0, 1\}$, то машината на Тюринг има следния вид:



□

Теорема 4.1. Ако L се разпознава от *недетерминистична* машина на Тюринг $\mathcal{N}$, то L е разпознава и от *детерминистична* машина на Тюринг $\mathcal{D}$.

Доказателство. Нека имаме недетерминистичната машина на Тюринг $\mathcal{N}$, за която $L = \mathcal{L}(\mathcal{N})$. Една дума α принадлежи на $\mathcal{L}(\mathcal{N})$ точно тогава, когато съществува изчисление, което започва с думата α върху лентата и след краен брой стъпки, следвайки функцията на преходите $\Delta_\mathcal{N}$, достига до състоянието $q_{\texttt{accept}}$. Сложността идва от факта, че за думата α може да имаме много различни изчисления, като само някои от тях завършват в $q_{\texttt{accept}}$. Ще построим детерминистична машина на Тюринг, която последователно ще симулира всички възможни *крайни* изчисления за думата α, докато намери такова, което завършва в състоянието $q_{\texttt{accept}}$.

В [HU79, стр. 164] не е добре обяснено.

Лесно се съобразява, че всяко изчисление на $\mathcal{N}$ може да се представи като крайна редица от елементи на $Q \times \Gamma \times \{\triangleleft, \triangleright, \square\}$. Понеже това множество е крайно, то можем на всяка такава тройка да съпоставим число в интервала $[1, r]$, където

$$r = 3 \cdot |Q| \cdot |\Gamma|.$$

На практика това, което е правим е да представим всички възможни изчисления на $\mathcal{N}$ като r-разклонено дърво и да го обходим в широчина, докато не достигнем до $q_{\texttt{accept}}$

Оттук следва, че всяко изчисление на $\mathcal{N}$ може да се представи като крайна редица от числа, всяко принадлежащо на интервала $[1, r]$.

Детерминистичната машина на Тюринг $\mathcal{D}$ има три ленти.

- На първата лента съхраняваме входящия низ и *тя никога не се променя*.

- На втората лента ще записваме последователно низове следвайки каноничната подредба на думите над азбуката $\{1, 2, \ldots, r\}$.

- На третата лента симулираме изчислението на $\mathcal{N}$ върху думата от първата лента, използвайки изчислението, което е описано на втората лента. Например, ако съдържанието на втората лента е $4, 1, 2$, това означава, че симулираме изчисление от три стъпки като на първата стъпка избираме четвъртата възможна тройка, на втората стъпка избираме първата възможна тройка, на третата стъпка избираме втората възможна тройка.

  Ако симулацията завърши в състоянието $q_{\texttt{accept}}$ на $\mathcal{N}$, то машината $\mathcal{D}$ завършва успешно. В противен случай, на втората лента записваме следващия низ; изтриваме третата лента и започваме нова симулация.

□

Твърдение 4.3 (Лема на Кьониг)**.** Ако T е безкрайно дърво с крайно разклонение, то T съдържа безкраен път.

Упътване. Дефинираме безкрайния път на стъпки. На всяка стъпка избираме този наследник, който е корен на безкрайно дърво. Понеже T е безкрайно дърво с крайно разклонение, на всяка стъпка можем да изберем такъв наследник. □

Следствие 4.1. Ако L се разпознава от *тотална недетерминистична* машина на Тюринг $\mathcal{N}$, то L също се разпознава и от *тотална детерминистична* машина на Тюринг $\mathcal{D}$.

Доказателство. Да разгледаме дървото T, което представя всички изчисления на тоталната $\mathcal{N}$ при вход думата ω. От лемата на Кьониг следва, че T е крайно дърво, защото ако допуснем, че T е безкрайно, то ще има безкрайно дълго изчисление на $\mathcal{N}$, което е невъзможно, понеже $\mathcal{N}$ винаги достига до финално състояние.

- Ако $\mathcal{N}$ приема дадена дума ω, то детерминистичната ни симулация на $\mathcal{N}$ ще достигне до изчисление, кодирано като път в T, което завършва в състояние $q_{\texttt{accept}}$.

- Ако $\mathcal{N}$ не приема дадена дума ω, то детерминистичната ни симулация на $\mathcal{N}$ ще покаже, че всяко изчисление, кодирано като път в T, завършва в състояние $q_{\texttt{reject}}$.

□

4.6 Основни свойства

Твърдение 4.4. Ако L е разрешим език, то $\overline{L}$ е разрешим език.

Упътване. Нека $L = \mathcal{L}(\mathcal{M})$, където $\mathcal{M}$ е тотална машина на Тюринг. Нека $\mathcal{M}'$ е същата като $\mathcal{M}$, само със сменени $q_{\texttt{accept}}$ и $q_{\texttt{reject}}$ състояния. Тогава $\overline{L} = \mathcal{L}(\mathcal{M}')$. □

Твърдение 4.5. Ако L_1 и L_2 са разрешими езици, то $L_1 \cup L_2$ е разрешим език.

Упътване. Нека $L_1 = \mathcal{L}(\mathcal{M}_1)$ и $L_2 = \mathcal{L}(\mathcal{M}_2)$. Симулираме двете изчисления едновременно. Ако едната машина достигне accept състоянието си, то връщаме accept. □

Твърдение 4.6. Ако L_1 и L_2 са полуразрешими езици, то $L_1 \cup L_2$ е полуразрешим език.

Твърдение 4.7. Ако L_1 и L_2 са разрешими езици, то $L_1 \cap L_2$ е разрешим език.

Упътване. Нека $L_1 = \mathcal{L}(\mathcal{M}_1)$ и $L_2 = \mathcal{L}(\mathcal{M}_2)$. Симулираме двете изчисления едновременно. Ако и двете машини достигнат accept състоянията си, то връщаме accept. □

Теорема 4.2. L и $\overline{L}$ са полуразрешими езици точно тогава, когато L е разрешим език.

Упътване. Посоката ($\Rightarrow$) е ясна. За посоката ($\Leftarrow$), нека $L = \mathcal{L}(\mathcal{M}_1)$ и $\overline{L} = \mathcal{L}(\mathcal{M}_2)$. Симулираме едновременно и двете изчисления. Знаем със сигурност, че точно едно от тях ще завърши в accept състояние. Ако това е $\mathcal{M}_1$, връщаме accept. Ако това е $\mathcal{M}_2$, връщаме reject. □

4.6.1 Кодиране на машина на Тюринг

Кодиране на преход

Да разгледаме прехода $\delta(q_i, X_j) = (q_k, X_l, D_m)$. Кодираме този преход по следния начин:

$$0^i 1 0^j 1 0^k 1 0^l 1 0^m$$

Да обърнем внимание, че в този двоичен код няма последователни единици и той започва и завършва с нула.

За да кодираме една машина на Тюринг $\mathcal{M}$ е достатъчно да кодираме функцията на преходите δ. Понеже δ е крайна функция, нека с числото

r да означим броя на всички възможни преходи. По описания по-горе начин, нека $code_i$ е числото в двоичен запис, получено за i-тия преход на δ. Тогава кодът на $\mathcal{M}$ е следното число в двоичен запис:

$$\ulcorner \mathcal{M} \urcorner \stackrel{\text{деф}}{=} 111\ \mathtt{code}_1\ 11\ \mathtt{code}_2\ 11\ \cdots\ 11\ \mathtt{code}_r\ 111.$$

- Лесно се съобразява, че за две МТ $\mathcal{M}$ и $\mathcal{M}'$ с различни функции на преходите, имаме $\ulcorner \mathcal{M} \urcorner \neq \ulcorner \mathcal{M}' \urcorner$.

Пример 4.9. Да се даде пример за кода на конкретна машина на Тюринг.

Твърдение 4.8. Следните езици са разрешими:

- $L = \{\ulcorner \mathcal{M} \urcorner \mid \mathcal{M}$ е машина на Тюринг$\}$;
- $L = \{\ulcorner \mathcal{M} \urcorner \mid \mathcal{M}$ е детерминистична машина на Тюринг$\}$.

Забележка. Следният език **не** е разрешим:

$$L_{\mathtt{tot}} = \{\ulcorner \mathcal{M} \urcorner \mid \mathcal{M} \text{ е тотална машина на Тюринг}\}.$$

4.6.2 Диагоналният език $L_{\mathtt{diag}}$

Теорема 4.3. Езикът

$$L_{\mathtt{diag}} \stackrel{\text{деф}}{=} \{\ulcorner \mathcal{M} \urcorner \mid \mathcal{M} \text{ е М.Т. и } \ulcorner \mathcal{M} \urcorner \notin L(\mathcal{M})\}$$

не се разпознава от машина на Тюринг, т.е. $L_{\mathtt{diag}}$ **не** е полуразрешим език.

Доказателство. Да допуснем, че $L_{\mathtt{diag}}$ се разпознава от машина на Тюринг $\mathcal{M}$, т.е.

$$L_{\mathtt{diag}} = \mathcal{L}(\mathcal{M}).$$

Тогава:

$$\ulcorner \mathcal{M} \urcorner \in L_{\mathtt{diag}} \implies \ulcorner \mathcal{M} \urcorner \in \mathcal{L}(\mathcal{M}) \implies \ulcorner \mathcal{M} \urcorner \notin L_{\mathtt{diag}},$$
$$\ulcorner \mathcal{M} \urcorner \notin L_{\mathtt{diag}} \implies \ulcorner \mathcal{M} \urcorner \notin \mathcal{L}(\ulcorner \mathcal{M} \urcorner) \implies \ulcorner \mathcal{M} \urcorner \in L_{\mathtt{diag}}.$$

Достигаме до противоречие. □

Твърдение 4.9. Езикът

$$L_{\mathtt{halt}} \stackrel{\text{деф}}{=} \{\ulcorner \mathcal{M} \urcorner \mid \mathcal{M} \text{ е М.Т. и } \ulcorner \mathcal{M} \urcorner \in \mathcal{L}(\mathcal{M})\}$$

е полуразрешим, но не е разрешим.

Упътване. Лесно се съобразява, че $L_{\texttt{halt}}$ е полуразрешим. Дефинираме машина на Тюринг $\mathcal{M}'$, която работи по следния начин:

- вход дума α;
- $\mathcal{M}'$ проверява дали α има вида $\ulcorner\mathcal{M}\urcorner$, за някоя машина на Тюринг $\mathcal{M}$;
- Ако $\alpha = \ulcorner\mathcal{M}\urcorner$, то $\mathcal{M}'$ симулира работата на $\mathcal{M}$ върху α.
  - Ако $\mathcal{M}$ завърши след краен брой стъпки като приема α, то $\mathcal{M}'$ приема α.
  - Ако $\mathcal{M}$ завърши след краен брой стъпки като отхвърля α, то $\mathcal{M}'$ отхвърля α.
  - Ако $\mathcal{M}$ никога не завършва върху α, то $\mathcal{M}'$ също никога не завършва върху α.
- Ако α няма вида $\ulcorner\mathcal{M}\urcorner$, то $\mathcal{M}'$ завършва като отхвърля думата α.

Получаваме, че

$$\alpha \in L_{\texttt{halt}} \;\leftrightarrow\; \alpha \in \mathcal{L}(\mathcal{M}'),$$

откъдето следва, че $L_{\texttt{halt}}$ е полуразрешим език.

Ако допуснем, че $L_{\texttt{halt}}$ е разрешим, то

$$L_{\texttt{diag}} = (\{0,1\}^\star \setminus L_{\texttt{halt}}) \cap \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{M} \text{ е М.Т.}\}$$

е разрешим език, което е противоречие. □

4.6.3 Универсалният език $L_{\texttt{univ}}$

Теорема 4.4. Езикът

$$L_{\texttt{univ}} \stackrel{\text{деф}}{=} \{\ulcorner\mathcal{M}\urcorner \cdot \omega \mid \mathcal{M} \text{ е М.Т. и } \omega \in \mathcal{L}(\mathcal{M})\}$$

е полуразрешим, но **не** е разрешим.

Упътване. Първо да съобразим защо $L_{\texttt{univ}}$ е полуразрешим език. Дефинираме машина на Тюринг $\mathcal{M}'$, която работи по следния начин:

- вход дума α;
- $\mathcal{M}'$ проверява дали α има вида $\ulcorner\mathcal{M}\urcorner \cdot \omega$, за някоя машина на Тюринг $\mathcal{M}$ и дума ω;
- Ако $\alpha = \ulcorner\mathcal{M}\urcorner \cdot \omega$, то $\mathcal{M}'$ симулира работата на $\mathcal{M}$ върху ω.

– Ако $\mathcal{M}$ завърши след краен брой стъпки като приеме ω, то $\mathcal{M}'$ приема α.
– Ако $\mathcal{M}$ завърши след краен брой стъпки като отхвърли ω, то $\mathcal{M}'$ отхвърля α.
– Ако $\mathcal{M}$ никога не завършва върху ω, то очевидно $\mathcal{M}'$ също никога не завършва върху α.

• Ако α няма вида $\ulcorner\mathcal{M}\urcorner \cdot \omega$, то $\mathcal{M}'$ завършва като отхвърля думата α.

Получаваме, че

$$\alpha \in L_{\texttt{univ}} \;\leftrightarrow\; \alpha \in \mathcal{L}(\mathcal{M}').$$

Сега да съобразим защо $L_{\texttt{univ}}$ не е разрешим език. Имаме, че за произволна дума ω,

$$\begin{aligned}\omega \in L_{\texttt{halt}} &\leftrightarrow (\exists\mathcal{M})[\mathcal{M} \text{ е М.Т. } \& \ \omega = \ulcorner\mathcal{M}\urcorner \ \& \ \omega \in \mathcal{L}(\mathcal{M})] \\ &\leftrightarrow \omega\cdot\omega \in L_{\texttt{univ}}.\end{aligned}$$

Ако допуснем, че $L_{\texttt{univ}}$ е разрешим, то тогава $L_{\texttt{halt}}$ е разрешим език, което е противоречие. □

Следствие 4.2. Езикът

$$\overline{L}_{\texttt{univ}} \stackrel{\text{деф}}{=} \{\ulcorner\mathcal{M}\urcorner \cdot \omega \mid \ulcorner\mathcal{M}\urcorner \text{ е М.Т. и } \omega \notin \mathcal{L}(\mathcal{M})\}$$

не е полуразрешим.

4.7 Критерий за разрешимост

Твърдение 4.10. Докажете, че езикът

$$L_{\texttt{All}} \stackrel{\text{деф}}{=} \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{M} \text{ е М.Т. и } \mathcal{L}(\mathcal{M}) = \Sigma^\star\}$$

не е разрешим.

Доказателство. Ще дефинираме алгоритъм, за който по вход думата $\ulcorner\mathcal{M}\urcorner\cdot\omega$ връща код на машината на Тюринг $\mathcal{M}'_\omega$, която работи по следния начин:

За различни $\mathcal{M}$ и ω, получаваме различни $\mathcal{M}'_\omega$

• Вход дума α;

• Първоначално $\mathcal{M}'_\omega$ не обръща внимание на α.

• $\mathcal{M}'_\omega$ симулира $\mathcal{M}$ върху думата ω;

– Ако след краен брой стъпки $\mathcal{M}$ завърши като приеме думата ω, то $\mathcal{M}'_\omega$ приема думата α, т.е. $\mathcal{M}'_\omega$ завършва в състоянието $q_{\texttt{accept}}$.

– Ако след краен брой стъпки $\mathcal{M}$ завърши като отхърли думата ω, то $\mathcal{M}'_\omega$ отхвърля думата α, т.е. $\mathcal{M}'_\omega$ завършва в състоянието $q_{\texttt{reject}}$.
– В противен случай, $\mathcal{M}$ никога не завършва върху ω. Това означава, че $\mathcal{M}'_\omega$ никога не завършва върху входа α и следователно $\mathcal{M}'_\omega$ не приема думата α.

Получаваме, че:

$$\ulcorner\mathcal{M}\urcorner \cdot \omega \in L_{\texttt{univ}} \implies \mathcal{L}(\mathcal{M}'_\omega) = \Sigma^\star \implies \ulcorner\mathcal{M}'_\omega\urcorner \in L_{\texttt{All}},$$
$$\ulcorner\mathcal{M}\urcorner \cdot \omega \notin L_{\texttt{univ}} \implies \mathcal{L}(\mathcal{M}'_\omega) = \emptyset \implies \ulcorner\mathcal{M}'_\omega\urcorner \notin L_{\texttt{All}}.$$

Ако допуснем, че $L_{\texttt{All}}$ е разрешим език, то $L_{\texttt{univ}}$ също ще е разрешим, което е противоречие. □

Следствие 4.3. Езикът

$$\overline{L}_{\texttt{Empty}} \stackrel{\text{деф}}{=} \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{M} \text{ е М.Т. и } \mathcal{L}(\mathcal{M}) \neq \emptyset\}$$

не е разрешим.

Упътване. Същата конструкция както горе. □

Следствие 4.4. Езикът

$$L_{\texttt{Empty}} \stackrel{\text{деф}}{=} \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{M} \text{ е М.Т. и } \mathcal{L}(\mathcal{M}) = \emptyset\}$$

не е разрешим.

Упътване. Ако $L_{\texttt{Empty}}$ беше разрешим, то

$$\overline{L}_{\texttt{Empty}} = \{0,1\}^\star \setminus L_{\texttt{Empty}} \cap \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{M} \text{ е М.Т.}\}$$

щеше да е разрешим език, което е противоречие. □

Твърдение 4.11. Докажете, че езикът

$$L_{\texttt{reg}} \stackrel{\text{деф}}{=} \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{M} \text{ е М.Т. и } \mathcal{L}(\mathcal{M}) \text{ е регулярен език}\}$$

не е разрешим.

Доказателство. Да фиксираме един език, за който знаем, че не е регулярен, например, $\{0^n1^n \mid n \in \mathbb{N}\}$.

Дефинираме алгоритъм, за който по вход $\ulcorner\mathcal{M}\urcorner \cdot \omega$ връща код на машината на Тюринг $\mathcal{M}'_\omega$. Сега ще опишем как работи $\mathcal{M}'_\omega$.

- Вход думата α;
- Ако $\alpha = 0^n1^n$, за някое n, то $\mathcal{M}'_\omega$ приема думата α.

- Ако α не е от вида 0^n1^n, тогава $\mathcal{M}'_\omega$ симулира $\mathcal{M}$ върху думата ω.
  - Ако след краен брой стъпки $\mathcal{M}$ завърши като приеме думата ω, то $\mathcal{M}'_\omega$ приема α.
  - Ако след краен брой стъпки $\mathcal{M}$ завърши като отхвърли думата ω, то $\mathcal{M}'_\omega$ отхвърля думата α.
  - В противен случай, $\mathcal{M}$ никога не завършва върху ω. Това означава, че $\mathcal{M}'_\omega$ никога не завършва върху входа α и следователно $\mathcal{M}'_\omega$ не приема думата α.

Получаваме, че:

$$\ulcorner\mathcal{M}\urcorner \cdot \omega \in L_{\texttt{univ}} \implies \mathcal{L}(\mathcal{M}'_\omega) = \Sigma^\star \implies \ulcorner\mathcal{M}'_\omega\urcorner \in L_{\texttt{reg}},$$
$$\ulcorner\mathcal{M}\urcorner \cdot \omega \notin L_{\texttt{univ}} \implies \mathcal{L}(\mathcal{M}'_\omega) = \{0^n1^n \mid n \in \mathbb{N}\} \implies \ulcorner\mathcal{M}'_\omega\urcorner \notin L_{\texttt{reg}}.$$

Ако допуснем, че $L_{\texttt{reg}}$ е разрешим език, то $L_{\texttt{univ}}$ също ще е разрешим, което е противоречие. □

Твърдение 4.12. Докажете, че езикът

$$\overline{L}_{\texttt{reg}} \stackrel{\text{деф}}{=} \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{M} \text{ е М.Т. и } \mathcal{L}(\mathcal{M}) \text{ не е регулярен език}\}$$

не е разрешим.

Доказателство. Да фиксираме машина на Тюринг $\mathcal{M}_L$, за което $\mathcal{L}(\mathcal{M}_L) = L$ не е регулярен език.

Дефинираме алгоритъм, за който по вход $\ulcorner\mathcal{M}\urcorner \cdot \omega$ връща код на машината на Тюринг $\mathcal{M}'_\omega$. Сега ще опишем как работи $\mathcal{M}'_\omega$.

За различни $\mathcal{M}$ и ω, получаваме различни $\mathcal{M}'_\omega$

- Вход думата α;
- Първоначално $\mathcal{M}'_\omega$ не обръща внимание на α.
- $\mathcal{M}'_\omega$ симулира $\mathcal{M}$ върху думата ω;
  - Ако след краен брой стъпки $\mathcal{M}$ завърши като приеме думата ω, то $\mathcal{M}'_\omega$ симулира $\mathcal{M}_L$ върху α.
    - Ако след краен брой стъпки $\mathcal{M}_L$ завърши като приеме α, то $\mathcal{M}'_\omega$ приема α;
    - Ако след краен брой стъпки $\mathcal{M}_L$ завърши като отхърли α, то $\mathcal{M}'_\omega$ отхвърля α;
    - Ако $\mathcal{M}_L$ никога не свършва върху α, то $\mathcal{M}'_\omega$ никога няма да свирши върху α, което означава, че $\mathcal{M}'_\omega$ не приема α.
  - Ако след краен брой стъпки $\mathcal{M}$ завърши като отхвърли думата ω, то $\mathcal{M}'_\omega$ отхвърля думата α.

– В противен случай, $\mathcal{M}$ работи безкрайно много стъпки върху ω без да завърши. Това означава, че $\mathcal{M}'_\omega$ ще работи безкрайно много стъпки върху думата α и следователно $\mathcal{M}'_\omega$ не приема думата α.

Получаваме, че:

$$\ulcorner\mathcal{M}\urcorner \cdot \omega \in L_{\mathtt{univ}} \implies \mathcal{L}(\mathcal{M}'_\omega) = L \implies \ulcorner\mathcal{M}'_\omega\urcorner \in \overline{L}_{\mathtt{reg}},$$
$$\ulcorner\mathcal{M}\urcorner \cdot \omega \notin L_{\mathtt{univ}} \implies \mathcal{L}(\mathcal{M}'_\omega) = \emptyset \implies \ulcorner\mathcal{M}'_\omega\urcorner \notin \overline{L}_{\mathtt{reg}}.$$

Ако допуснем, че $\overline{L}_{\mathtt{reg}}$ е разрешим език, то $L_{\mathtt{univ}}$ също ще е разрешим, което е противоречие. □

Сега ще видим, че идеята, която следваме в горните доказателства може да се обобщи. Нека $\mathscr{S}$ е множество от полуразрешими езици над фиксирана азбука Σ. Например,

$\mathscr{S} = \{L \mid$
L се разпознава от машина на Тюри
Това защо не върши
работа?

$$\mathscr{S} = \{L \subseteq \Sigma^\star \mid L \text{ е регулярен език}\}.$$

Ще казваме, че $\mathscr{S}$ е свойство на полуразрешимите езици. $\mathscr{S}$ е **тривиално свойство**, ако $\mathscr{S} = \emptyset$ или $\mathscr{S}$ съдържа точно всички полуразрешими езици. Нека означим езика, който съответсва на свойството $\mathscr{S}$:

$$L_{\mathscr{S}} \overset{\text{деф}}{=} \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{M} \text{ е машина на Тюринг и } \mathcal{L}(\mathcal{M}) \in \mathscr{S}\}.$$

Теорема 4.5 (Райс)**.** Всяко нетривиално свойство $\mathscr{S}$ на полуразрешимите езици е неразрешимо.

Доказателство. Без ограничение на общността, нека $\emptyset \notin \mathscr{S}$. Понеже $\mathscr{S}$ е нетривиално свойство, да разгледаме езика $L \in \mathscr{S}$, като $\mathcal{M}_L$ е машина на Тюринг, за която $\mathcal{L}(\mathcal{M}_L) = L$. Да разгледаме алгоритъм, който по дадена дума $\ulcorner\mathcal{M}\urcorner \cdot \omega$ връща код на машина на Тюринг $\mathcal{M}'_\omega$, която зависи от $\mathcal{M}$, ω и от $\mathcal{M}_L$. Тя работи по следния начин:

[HU79, стр. 188]
Цел: да сведем ефективно $L_{\mathtt{univ}}$ към $L_{\mathscr{S}}$

- вход думата α;
- първоначално $\mathcal{M}'_\omega$ не обръщаме внимание на α;
- $\mathcal{M}'_\omega$ симулира $\mathcal{M}$ върху думата ω.
  - ако след краен брой стъпки $\mathcal{M}$ завърши като приеме думата ω, то $\mathcal{M}'_\omega$ симулира $\mathcal{M}_L$ върху входната дума α;
    - ако след краен брой стъпки $\mathcal{M}_L$ завърши като приеме думата α, то $\mathcal{M}'_\omega$ приема α;
    - ако след краен брой стъпки $\mathcal{M}_L$ завърши като отхвърли думата α, то $\mathcal{M}'_\omega$ отхвърля α;

Неформално описваме функцията δ за $\mathcal{M}'$

в този случай ще получим, че $\mathcal{L}(\mathcal{M}') = L$

- ако $\mathcal{M}_L$ никога не завършва върху α, то $\mathcal{M}'_\omega$ никога няма да завърши върху α и следователно $\mathcal{M}'_\omega$ не приема α.

– ако след краен брой стъпки $\mathcal{M}$ завърши като отхвърли думата ω, то $\mathcal{M}'_\omega$ отхвърля α;

при тези два случая ще получим, че $\mathcal{L}(\mathcal{M}') = \emptyset$

– Ако $\mathcal{M}$ никога не свършва върху ω, то $\mathcal{M}'_\omega$ никога няма да свирши върху α, което означава, че $\mathcal{M}'_\omega$ не приема α.

От всичко това следва, че така описаната машина на Тюринг $\mathcal{M}'$ има свойствата:

$$\ulcorner\mathcal{M}\urcorner \cdot \omega \in L_{\mathtt{univ}} \implies \mathcal{L}(\mathcal{M}') = L \implies \mathcal{L}(\mathcal{M}') \in \mathscr{S},$$
$$\ulcorner\mathcal{M}\urcorner \cdot \omega \notin L_{\mathtt{univ}} \implies \mathcal{L}(\mathcal{M}') = \emptyset \implies \mathcal{L}(\mathcal{M}') \notin \mathscr{S}.$$

Да допуснем, че $L_{\mathscr{S}}$ е разрешимо множество от полуразрешими езици. Тогава от еквивалентността,

$$\ulcorner\mathcal{M}\urcorner \cdot \omega \in L_{\mathtt{univ}} \leftrightarrow \ulcorner\mathcal{M}'\urcorner \in L_{\mathscr{S}},$$

получаваме, че $L_{\mathtt{univ}}$ е разрешимо множество, което е противоречие.

Ако $\emptyset \in \mathscr{S}$, то правим горните разсъждения за класа

$$\overline{\mathscr{S}} = \{\mathcal{L}(\mathcal{M}) \mid \mathcal{M} \text{ е М.Т. и } \mathcal{L}(\mathcal{M}) \notin \mathscr{S}\}.$$

По аналогичен начин доказваме, че $L_{\overline{\mathscr{S}}}$ не е разрешим език. Понеже

$$L_{\overline{\mathscr{S}}} = (\{0,1\}^\star \setminus L_{\mathscr{S}}) \cap \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{M} \text{ е машина на Тюринг}\},$$

то $L_{\mathscr{S}}$ също не е разрешим език. □

Следствие 4.5. За всяко от следните свойства $\mathscr{S}$ на полуразрешимите множества, $L_{\mathscr{S}}$ **не** е разрешим език, където:

а) $\mathscr{S}$ е свойството празнота, т.е. езикът

$$L_{\mathscr{S}} = \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{M} \text{ е М.Т. и } \mathcal{L}(\mathcal{M}) = \emptyset\}$$

не е разрешим;

б) $\mathscr{S}$ е свойството за пълнота, т.е. езикът

$$L_{\mathscr{S}} = \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{M} \text{ е М.Т. и } \mathcal{L}(\mathcal{M}) = \Sigma^\star\}$$

не е разрешим;

в) $\mathscr{S}$ е свойството крайност, т.е. езикът

$$L_{\mathscr{S}} = \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{M} \text{ е М.Т. и } |\mathcal{L}(\mathcal{M})| < \infty\}$$

не е разрешим;

г) $\mathscr{S}$ е свойството безкрайност, т.е. езикът

$$L_{\mathscr{S}} = \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{M} \text{ е М.Т. и } |\mathcal{L}(\mathcal{M})| = \infty\}$$

не е разрешим;

д) $\mathscr{S}$ е свойството регулярност, т.е. езикът

$$L_{\mathscr{S}} = \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{M} \text{ е М.Т. и } \mathcal{L}(\mathcal{M}) \text{ е регулярен език}\}$$

не е разрешим;

е) $\mathscr{S}$ е свойството безконтекстност, т.е. езикът

$$L_{\mathscr{S}} = \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{M} \text{ е М.Т. и } \mathcal{L}(\mathcal{M}) \text{ е безконтекстен}\}$$

не е разрешим;

ж) $\mathscr{S}$ е свойството разрешимост, т.е. езикът

$$L_{\mathscr{S}} = \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{M} \text{ е М.Т. и } \mathcal{L}(\mathcal{M}) \text{ е разрешим}\}$$

не е разрешим.

4.8 Критерии за полуразрешимост

Лема 4.1. Нека $\mathscr{S}$ е свойство на полуразрешимите езици. Ако съществува безкраен език $L_0 \in \mathscr{S}$, който няма крайно подмножество в $\mathscr{S}$, то $L_{\mathscr{S}}$ не е полуразрешим език.

Упътване. Нека $L_0 = \mathcal{L}(\mathcal{M}_0)$. Ще опишем алгоритъм, който при вход дума $\ulcorner\mathcal{M}\urcorner\cdot\omega$, извежда код на машина на Тюринг $\mathcal{M}'_\omega$, която работи така:

- вход думата α;
- $\mathcal{M}'_\omega$ симулира $\mathcal{M}$ върху думата ω:
  - ако $\mathcal{M}$ завърши за по-малко от $|\alpha|$ на брой стъпки като *отхвърли* ω, то $\mathcal{M}'_\omega$ симулира $\mathcal{M}_0$ върху α;
  - ако $\mathcal{M}$ завърши за по-малко от $|\alpha|$ на брой стъпки като приеме ω, то $\mathcal{M}'_\omega$ завършва като отхвърля α.
  - ако $\mathcal{M}$ не завърши за по-малко от $|\alpha|$ на брой стъпки върху ω, то $\mathcal{M}'_\omega$ завършва като отхвърля α.

Така получаваме, че

$$\mathcal{L}(\mathcal{M}') = \begin{cases} \{\alpha \in L_0 \mid |\alpha| < s\}, & \text{ако } \mathcal{M} \text{ приема } \omega \\ L, & \text{ако } \mathcal{M} \text{ не приема } \omega, \end{cases}$$

където s е минималното число, такова че $\mathcal{M}$ завършва за s на брой стъпки като приема думата ω.

Заключаваме, че

$$\ulcorner\mathcal{M}\urcorner \cdot \omega \in L_{\mathtt{univ}} \implies \ulcorner\mathcal{M}'_\omega\urcorner \notin L_{\mathscr{S}}$$
$$\ulcorner\mathcal{M}\urcorner \cdot \omega \notin L_{\mathtt{univ}} \implies \ulcorner\mathcal{M}'_\omega\urcorner \in L_{\mathscr{S}}.$$

Това означава, че ефективно можем да сведем въпрос за принадлежност в $\overline{L}_{\mathtt{univ}}$ към въпрос за принадлежност в $L_{\mathscr{S}}$. Следователно, ако $L_{\mathscr{S}}$ е полуразрешим език, то $\overline{L}_{\mathtt{univ}}$ е полуразрешим език, което е противоречие. □

Следствие 4.6. Следните езици **не** са полуразрешими:

- $L = \{\ulcorner\mathcal{M}\urcorner \mid |\mathcal{L}(\mathcal{M})| = \infty\}$;
- $L = \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{L}(\mathcal{M}) = \Sigma^\star\}$;
- $L = \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{L}(\mathcal{M})$ не е разрешим$\}$;
- $L = \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{L}(\mathcal{M})$ не е полуразрешим$\}$;
- $L = \{\ulcorner\mathcal{M}\urcorner \mid \mathcal{L}(\mathcal{M})$ не е регулярен$\}$.

Лема 4.2. Нека L_1 е език в $\mathscr{S}$ и нека L_2 е полуразрешим език, като $L_1 \subset L_2$ и $L_2 \notin \mathscr{S}$. Тогава $L_{\mathscr{S}}$ не е полуразрешим език.

Упътване. Нека $L_1 = \mathcal{L}(\mathcal{M}_1)$ и $L_2 = \mathcal{L}(\mathcal{M}_2)$. Ще опишем алгоритъм, който при вход дума $\ulcorner\mathcal{M}\urcorner \cdot \omega$, извежда код на машина на Тюринг $\mathcal{M}'$, която работи така:

- вход думата α;
- $\mathcal{M}'_\omega$ симулира едновременно две изчисления - $\mathcal{M}_1$ върху α и $\mathcal{M}$ върху ω докато намери стъпка s, такава че:
  - ако $\mathcal{M}_1$ завършва за s на брой стъпки като приема думата α, то $\mathcal{M}'_\omega$ завършва като приеме думата α;
  - ако $\mathcal{M}$ за завършва за s на брой стъпки като приема думата ω, то $\mathcal{M}'_\omega$ започваме да симулира $\mathcal{M}_2$ върху α;
    - Ако $\mathcal{M}_2$ завърши като приеме α, то $\mathcal{M}'_\omega$ завършва като приема α;
    - Ако $\mathcal{M}_2$ завърши като отхвърли α, то $\mathcal{M}'_\omega$ завършва като отхвърля α;
    - Ако $\mathcal{M}_2$ не завършва никога върху α, то $\mathcal{M}'_\omega$ никога не завършва върху α.

– ако $\mathcal{M}'_\omega$ не намери такава стъпка, то $\mathcal{M}'_\omega$ никога не завършва върху α.

Получаваме, че:

$$\mathcal{L}(\mathcal{M}'_\omega) = \begin{cases} L_2, & \text{ако } \mathcal{M} \text{ приема } \omega \\ L_1, & \text{ако } \mathcal{M} \text{ не приема } \omega. \end{cases}$$

Заключаваме, че:

$$\ulcorner \mathcal{M} \urcorner \cdot \omega \in \overline{L}_{\mathtt{univ}} \;\leftrightarrow\; \ulcorner \mathcal{M}'_\omega \urcorner \in L_{\mathscr{S}},$$

защото $L_2 \notin \mathscr{S}$, а $L_1 \in \mathscr{S}$. Това означава, че ефективно можем да сведем въпрос за принадлежност в $\overline{L}_{\mathtt{univ}}$ към въпрос за принадлежност в $L_{\mathscr{S}}$. Следователно, ако $L_{\mathscr{S}}$ е полуразрешим език, то $\overline{L}_{\mathtt{univ}}$ е полуразрешим език, което е противоречие. □

Следствие 4.7. Следните езици **не** са полуразрешими:

- $L = \{\ulcorner \mathcal{M} \urcorner \mid \mathcal{L}(\mathcal{M}) \text{ е регулярен}\}$;
- $L = \{\ulcorner \mathcal{M} \urcorner \mid \mathcal{L}(\mathcal{M}) \text{ е безконтекстен}\}$;
- $L = \{\ulcorner \mathcal{M} \urcorner \mid \mathcal{L}(\mathcal{M}) \text{ е разрешим}\}$;
- $L = \{\ulcorner \mathcal{M} \urcorner \mid |\mathcal{L}(\mathcal{M})| = 42\}$;

4.9 Сложност

- Детерминистичната машината на Тюринг $\mathcal{M}$ е **полиномиално ограничена**, ако съществува полином $p(x)$, такъв че за всеки вход ω, машината $\mathcal{M}$ завършва след най-много $p(|\omega|)$ стъпки.
- Езикът L се нарича **полиномиално разрешим**, ако съществува полиномиално ограничена тотална детерминистична машина на Тюринг $\mathcal{M}$, за която $L = \mathcal{L}(\mathcal{M})$.
- $\mathcal{P} \stackrel{\text{деф}}{=} \{L \subseteq \Sigma^\star \mid L \text{ е детерминистично полиномиално разрешим}\}$.
- Недетерминистичната машината на Тюринг $\mathcal{M}$ е **полиномиално ограничена**, ако съществува полином $p(x)$, такъв че за всеки вход ω, съществува изчисление на машината $\mathcal{M}$ върху думата ω, което завършва след най-много $p(|\omega|)$ стъпки.
- Езикът L се нарича **недетерминистично полиномиално разрешим**, ако съществува полиномиално ограничена тотална недетерминистична машина на Тюринг $\mathcal{M}$, за която $L = \mathcal{L}(\mathcal{M})$.

- $\mathcal{NP} \stackrel{\text{деф}}{=} \{L \subseteq \Sigma^\star \mid L \text{ е недетерминистично полиномиално разрешим}\}$.

Твърдение 4.13. Можем да обобщим някои от резултатите от предишните глави:

$$\texttt{REG} \subsetneq \texttt{CFG} \subsetneq \mathcal{P}.$$

Упътване. Езикът $\{a^n b^n c^n \mid n \in \mathbb{N}\} \in \mathcal{P}$, но не е безконтекстен. □

4.10 Задачи

Задача 4.5. Вярно ли е, че следните езици са разрешими?

а) $\{\ulcorner \mathcal{A} \urcorner \mid \mathcal{A} \text{ е ДКА и } \mathcal{L}(\mathcal{A}) = \{0,1\}^\star\}$;

б) $\{\ulcorner \mathcal{A} \urcorner \mid \mathcal{A} \text{ е ДКА и } \mathcal{L}(\mathcal{A}) \text{ съдържа поне една дума с равен брой нули и единици}\}$;

в) $\{\ulcorner \mathcal{A} \urcorner \mid \mathcal{A} \text{ е ДКА и } \mathcal{L}(\mathcal{A}) \text{ съдържа поне една дума палиндром}\}$;

г) $\{\ulcorner \mathcal{A} \urcorner \mid \mathcal{A} \text{ е ДКА и } \mathcal{L}(\mathcal{A}) \text{ не съдържа дума с нечетен брой единици}\}$;

д) $\{\ulcorner \mathcal{A} \urcorner \cdot \ulcorner \mathcal{B} \urcorner \mid \mathcal{A} \text{ и } \mathcal{B} \text{ са ДКА и } \mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})\}$;

Задача 4.6. Вярно ли е, че следните езици са разрешими?

а) $\{\ulcorner G \urcorner \mid G \text{ е безконтекстна граматика над } \{0,1\}^\star \text{ и } \mathcal{L}(1^\star) \subseteq \mathcal{L}(G)\}$;

б) $\{\ulcorner G \urcorner \mid G \text{ е безконтекстна граматика над } \{0,1\}^\star \text{ и } \varepsilon \in \mathcal{L}(G)\}$;

в) $\{\ulcorner G \urcorner \cdot 0^k \mid G \text{ е безконтекстна граматика над } \{0,1\}^\star \text{ и } |\mathcal{L}(G)| \leq k\}$;

г) $\{\ulcorner G \urcorner \mid G \text{ е безконтекстна граматика над } \{0,1\}^\star \text{ и } |\mathcal{L}(G)| = \infty\}$;

Бележки

- За основните дефиниции следваме основно Глава 3 от [Sip12].
- За въпросите за неразрешимост следваме основно Глава 8 от [HU79].

Библиография

[HMU01] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman, *Introduction to automata theory, languages, and computation*, second ed., Addison-Wesley, 2001.

[HU79] John E. Hopcroft and Jeffrey D. Ullman, *Introduction to automata theory, languages, and computation*, first ed., Addison-Wesley, 1979.

[Koz97] Dexter Kozen, *Automata and computability*, Springer, 1997.

[PL98] Christos Papadimitriu and Harry Lewis, *Elements of the theory of computation*, Prentice-Hall, 1998.

[Ros12] Kenneth H. Rosen, *Discrete Mathematics and Its applications*, seventh ed., McGraw Hill, 2012.

[RS59] M. O. Rabin and D. Scott, *Finite automata and their decision problems*, IBM Journal of Research and Development **3** (1959), pp. 114 – 125.

[Sip97] Michael Sipser, *Introduction to the theory of computation*, 1 ed., PWS Publishing Company, 1997.

[Sip12] ———, *Introduction to the theory of computation*, 3 ed., Cengage Learning, 2012.

Азбучен указател