

Міністерство освіти і науки України
ДВНЗ “Прикарпатський національний університет
імені Василя Стефаника”

Володимир Гаврилків

Формальні мови та скінченні автомати

Навчальний посібник

Хрестоматія

Івано-Франківськ

2020

УДК 510.5:004.423.24

ББК 22.123

Г 12

Гаврилків В.М. *Формальні мови та скінченні автомати: навчальний посібник, хрестоматія*, Івано-Франківськ, 2020. — 95 с.

У хрестоматії у вигляді курсу з 10 параграфів викладено основи теорії формальних мов і скінченних автоматів. Кожен параграф супроводжується питаннями та завданнями для самостійного розв'язування.

© Володимир Гаврилків, 2020

Зміст

Розділ І. Формальні мови	4
§ 1. Вільні напівгрупи і формальні мови	4
§ 2. Системи числення	10
§ 3. Регулярні мови і регулярні вирази	23
§ 4. Формальні породжуючі граматики	30
Розділ ІІ. Скінченні автомати	40
§ 1. Автомати. Їх типи та задання	40
§ 2. Детерміновані скінченні автомати без виходу	50
§ 3. Недетерміновані скінченні автомати без виходу	60
§ 4. Перетворення НСА до ДСА	67
§ 5. Скінченні автомати і регулярні мови	76
§ 6. Автомати з магазинною пам'яттю	85
Список літератури	92
Показчик	94

Розділ I

Формальні мови

§ 1. Вільні напівгрупи і формальні мови

В цьому параграфі вводиться поняття формальної мови та вивчаються деякі властивості формальних мов. Вивчення формальних мов спричинено, в першу чергу, їх застосуванням в комп'ютерних науках. Формальні мови використовуються для опису штучних мов, наприклад, мов програмування, мов команд операційної системи. Будь-яке програмне забезпечення створюється і експлуатується з допомогою тієї чи іншої формальної мови.

Розпочнемо параграф з необхідних допоміжних означень. Нехай X – довільна множина. *Бінарною операцією на множині X* називається відображення $* : X \times X \rightarrow X$, де $X \times X$ – множина всіх впорядкованих пар елементів з X . Образ в X елемента $(a, b) \in X \times X$ позначатимемо через $a * b$ або просто через ab . Непорожня множина X , наділена бінарною операцією $* : X \times X \rightarrow X$, називається *групоїдом*. Бінарна операція $*$ на множині X називається *асоціативною*, якщо $a(bc) = (ab)c$ для всіх $a, b, c \in X$. Якщо операція $*$ є асоціативною на X , то пара $(X, *)$ називається *напівгрупою*. Нехай $(X, *)$ і (Y, \circ) – напівгрупи. *Гомоморфізмом з X в Y* називається таке відображення $\varphi : X \rightarrow Y$, що $\varphi(a * b) = \varphi(a) \circ \varphi(b)$ для всіх $a, b \in X$.

Нехай X – непорожня множина, яку називатимемо (*формальним*) *алфавітом*, а її елементи – *буквами (символами, знаками)*. Якщо $|X| = 2$, то алфавіт називається *бінарним, або двійковим*. Найчастіше бінарний алфавіт позначається через $B = \{0, 1\}$.

1.1. ПРИКЛАД. Алфавіт X формул алгебри висловлень є об'єднанням $X = X_1 \cup X_2 \cup X_3$, де множина $X_1 = \{p, q, r, \dots\}$ містить пропозиційні змінні, $X_2 = \{\wedge, \vee, \rightarrow, \leftrightarrow, \neg\}$ – логічні зв'язки, а множина $X_3 = \{(\,,\,)\}$ складається з допоміжних символів (дужок).

Визначимо *слово (ланцюг)* в алфавіті X як непорожню скінченну послідовність $x_1x_2 \dots x_m$ елементів з X . Наприклад, 010111 – слово в бінарному алфавіті B , а $p \wedge q \rightarrow r$ – слово в алфавіті X з прикладу 1.1. Таким чином, два слова $x_1x_2 \dots x_m$ і $y_1y_2 \dots y_n$ дорівнюють тоді і тільки тоді, коли вони співпадають як послідовності, тобто коли $m = n$ і $x_1 = y_1, \dots, x_m = y_m$.

Алфавіти позначатимемо великими буквами латинського алфавіту, наприклад A, B, C, X і т.д. Букви формального алфавіту будемо позначати малими буквами латинського алфавіту: a, b, c і т.д., а слова – малими буквами грецького алфавіту, наприклад, α, β, ω .

Через X^+ позначимо множину всіх слів в алфавіті X . Наприклад, якщо X – бінарний алфавіт $\{0, 1\}$, то

$$X^+ = \{0, 1, 00, 01, 10, 11, 000, 001, \dots\}.$$

На множині X^+ визначимо бінарну операцію:

$$x_1x_2 \dots x_m \circ y_1y_2 \dots y_n = x_1x_2 \dots x_my_1y_2 \dots y_n.$$

Операція \circ називається *конкатенацією*. Вона, очевидно, асоціативна, і (X^+, \circ) називається *вільною напівгрупою на множині X* . Вільну напівгрупу на множині X позначають часто також через $F(X)$ (від англійського слова „free” – вільний).

Поряд з вільною напівгрупою X^+ над алфавітом X часто розглядають вільний моноїд $X^* = X^+ \cup \{e\}$ над X , в якому одиниця e є порожнім словом, яке не містить жодної букви.

Слово, яке містить i букв (слів) a позначатимемо через a^i . Наприклад, $a^1 = a$ (таким чином, ми ототожнюємо букву a з словом, що містить єдину букву a), $a^2 = aa$, $a^0 = e$ – порожнє слово e . *Оберненням слова x* (позначається через x^R) називається слово, записане в оберненому порядку, тобто якщо $x = a_1 \dots a_n$, де всі a_i – букви, то $x^R = a_n \dots a_1$. Крім того, $e^R = e$. Легко перевірити, що $(xy)^R = y^R x^R$.

Нехай α, β і γ – довільні слова в деякому алфавіті X . Назвемо α *префіксом слова $\alpha\beta$* , а β – *суфіксом слова $\alpha\beta$* . Слово β називатимемо *підсловом слова $\alpha\beta\gamma$* . Префікс і суфікс слова є його підсловами. Наприклад, ba – префікс і підслово слова bac . Зауважимо, що порожнє слово є префіксом, суфіксом і підсловом довільного слова.

Якщо $\alpha \neq \beta$ і α – префікс (суфікс) слова β , то α називається *власним префіксом (суфіксом) слова β* . *Довжина слова* – це

кількість букв у ньому, тобто якщо $\omega = a_1 \dots a_n$, де всі a_i – букви, то довжина слова ω дорівнює n . Довжину слова ω позначатимемо через $|\omega|$. Наприклад, $|aab| = 3$ і $|e| = 0$. Очевидно, що $|\omega_1 \circ \omega_2| = |\omega_1| + |\omega_2|$.

Формальною мовою в алфавіті X називається довільна множина слів в X , тобто довільна підмножина вільного моноїда X^* . Наприклад, множина \emptyset – це формальна мова. Множина $\{e\}$, яка містить тільки порожнє слово, також є мовою. Зауважимо, що \emptyset і $\{e\}$ – дві різні формальні мови. Іншим прикладом формальної мови є мова L , що містить всі слова, які складаються з нуля і більше букв a . Її можна позначити через $\{a^i : i \geq 0\}$. Ясно, що $L = \{a\}^*$ (для спрощення $\{a\}^*$ позначатимемо надалі через a^*).

Якщо мова L така, що жодне слово в L не є власним префіксом (суфіксом) жодного іншого слова в L , то кажуть, що L має *префіксну (суфіксну) властивість*. Наприклад, a^* не має префіксної властивості, а $\{a^i b : i \geq 0\}$ має.

Оскільки формальна мова L – це множина (підмножина X^*), то до формальних мов застосовні теоретико-множинні операції об'єднання, перетину, знаходження різниці і доповнення. Операцію конкатенації можна застосувати до мов таким же чином, як і до слів.

Нехай L_1 та L_2 – формальні мови в алфавітах X_1 та X_2 відповідно. *Конкатенацією (добутком) мов L_1 і L_2* називається формальна мова $L_1 L_2 = \{\omega_1 \circ \omega_2 : \omega_1 \in L_1, \omega_2 \in L_2\}$ в алфавіті $X_1 \cup X_2$, яка складається зі слів, які утворюються шляхом дописування до будь-якого слова мови L_1 довільного слова мови L_2 .

Ітерація (замикання Кліні) мови L , яка позначається через L^* , визначається наступним чином:

- 1) $L^0 = \{e\}$,
- 2) $L^n = L L^{n-1}$ для $n \geq 1$,
- 3) $L^* = \bigcup_{n \geq 0} L^n$.

Розглянемо деякі приклади.

1.2. ПРИКЛАД. Знайдемо ітерацію мови $L = \{00, 111\}$. Мова $L^0 = \{e\}$ містить одне порожнє слово незалежно від самої мови L . $L^1 = L$ містить всі слова мови L . Таким чином, перші два члени в розкладі L^* утворюють мову $\{e, 00, 111\}$. Далі знайдемо мову $L^2 = L L = \{0000, 00111, 11100, 111111\}$. Аналогічно $L^3 = L L^2$ є множиною слів,

утворених трикратним вибором із двох слів мови L . Для обчислення L^* необхідно знайти L^i для кожного $i \in \mathbb{N} \cup \{0\}$ і об'єднати всі ці мови. Хоча кожна множина L^i є скінченною, об'єднання нескінченної кількості множин, як правило, утворює нескінченну мову. Це, зокрема, стосується і мови L , яка містить нескінченну кількість слів, в яких максимальні підслова лише з нулів мають парну довжину, а довжина максимальних слів з одиниць кратна три.

1.3. ПРИКЛАД. Покажемо, що формальна мова $\{0, 10\}^*$ складається з усіх слів, які не містять підслова 11 і закінчуються на 0.

Очевидно, що конкатенація довільної кількості 0 та 10 закінчується на 0. Вона не може породити підслово 11, оскільки останні букви обох слів 0 та 10 розділяють довільні дві 1 в їхній конкатенації.

Нехай ω – слово в алфавіті $\{0, 1\}$, яке не містить підслів 11 і закінчується на 0. Якщо ω не містить входжень 1, то $\omega \in \{0, 10\}^{|\omega|} \subset \{0, 10\}^*$. Нехай слово ω містить $n \geq 1$ входжень букв 1. Тоді після кожного входження букви 1 в ω мусить слідувати буква 0, бо інакше буква 1 або слідує за 1, або є останньою буквою слова ω , що суперечить вибору слова ω . Таким чином, слово ω запишеться як

$$0 \dots 0(10)0 \dots 0(10)0 \dots 0(10)0 \dots 0,$$

де вираз $0 \dots 0$ означає нуль або більше входжень букви 0. Отже, $\omega \in \{0, 10\}^*$.

Для мови L визначимо *обернену мову* як $L^R = \{\omega^R : \omega \in L\}$.

1.4. ПРИКЛАД. Доведемо, що для формальних мов A і B виконуються рівності $(AB)^R = B^R A^R$ та $(A \cup B)^R = A^R \cup B^R$.

Дійсно, $(AB)^R = \{\omega^R \mid \omega \in AB\} = \{(\alpha\beta)^R \mid \alpha \in A, \beta \in B\} = \{\beta^R \alpha^R \mid \alpha \in A, \beta \in B\} = \{\beta^R \mid \beta \in B\} \cdot \{\alpha^R \mid \alpha \in A\} = B^R A^R$.

Аналогічно $(A \cup B)^R = \{\omega^R \mid \omega \in A \cup B\} = \{\omega^R \mid \omega \in A\} \cup \{\omega^R \mid \omega \in B\} = A^R \cup B^R$.

1.5. ТВЕРДЖЕННЯ. (Лема Ардена) *Нехай A і B – дві формальні мови, причому $e \notin A$, і мова X задовольняє рівність $X = AX \cup B$. Тоді $X = A^*B$.*

ДОВЕДЕННЯ. Доведемо індукцією за довжиною $|\omega|$ слова ω , що $X \subset A^*B$. Спершу розглянемо випадок $|\omega| = 0$, тобто $\omega = e$. Якщо $e \in X$, то $e \in AX \cup B$. Оскільки $e \notin A$, то необхідно $e \in B$ і, отже, $e \in A^*B$.

Далі припустимо, що для всіх слів ω довжини менше n з того, що $\omega \in X$ випливає $\omega \in A^*B$, і розглянемо слово α довжини n . Якщо $\alpha \in X = AX \cup B$, то або $\alpha \in B \subset A^*B$, або $\alpha = \beta\gamma$ для деяких $\beta \in A$ і $\gamma \in X$. В другому випадку $\beta \neq e$ і, отже, $|\gamma| < |\alpha|$. Тому за індуктивним припущенням $\gamma \in A^*B$ і $\alpha = \beta\gamma \in AA^*B \subset A^*B$. Цим завершується індуктивний крок і, отже, $X \subset A^*B$.

Щоб довести протилежне включення, використаємо індукцію, щоб показати, що $A^nB \subset X$ для всіх $n \geq 0$. Для $n = 0$ маємо, що $A^0B = B \subset AX \cup B = X$. Для $n > 0$ за індуктивним припущенням виконується $A^nB = A(A^{n-1}B) \subset AX$. Таким чином, $A^nB \subset AX \subset AX \cup B = X$ і $A^*B = (\cup_{n \geq 0} A^n)B = \cup_{n \geq 0} (A^nB) \subset X$. \square

1.6. ПРИКЛАД. Нехай формальні мови $A, B \subset \{a, b\}^*$ задовольняють рівності:

$$\begin{aligned} A &= \{e\} \cup \{a\}A \cup \{b\}B, \\ B &= \{e\} \cup \{b\}B. \end{aligned}$$

Знайти мови A і B .

Застосуємо лему Ардена до другого рівняння і знайдемо мову $B = \{b\}^*\{e\} = \{b\}^*$. Аналогічно знаходимо мову $A = \{a\}^*(\{e\} \cup \{b\}B)$.

Насамкінець підставимо $\{b\}^*$ замість B і одержуємо:

$$A = \{a\}^*(\{e\} \cup \{b\}\{b\}^*) = \{a\}^*\{b\}^*.$$

Нехай X_1 і X_2 – алфавіти і $h : X_1 \rightarrow X_2^*$ – довільне відображення. Відображення h можна продовжити до гомоморфізму $\bar{h} : X_1^* \rightarrow X_2^*$, покладаючи $\bar{h}(e) = e$ і $\bar{h}(\omega a) = \bar{h}(\omega)h(a)$ для всіх $\omega \in X_1^*$ і $a \in X_1$. Застосовуючи гомоморфізм до мови L , одержимо іншу мову $\bar{h}(L)$, яка є множиною слів $\{\bar{h}(\omega) : \omega \in L\}$.

1.7. ПРИКЛАД. Припустимо, що потрібно замінити кожне входження в слово букви 0 на букву a , а кожне входження 1 на bb . Тоді можна визначити гомоморфізм h так, що $h(0) = a$ і $h(1) = bb$. Якщо $L = \{0^n 1^n : n \geq 1\}$, то $\bar{h}(L) = \{a^n b^{2n} : n \geq 1\}$.

Рекомендована література : [2, с. 26–31], [4, с. 460–471], [5, с. 338–341], [24, с. 6–14].

Питання та вправи до параграфу 1.

- 1.1. Дайте означення формального алфавіту та формальної мови.
- 1.2. Де застосовуються формальні мови?
- 1.3. Які операції над формальними мовами ви знаєте?
- 1.4. Визначте формальні алфавіти для запису:
 - а) арифметичних виразів;
 - б) азбуки Морзе.
- 1.5. Визначте довжину слова $\alpha =$ „математика” в українському алфавіті.
- 1.6. Які з наведених слів є підсловами слова $\omega =$ „індустріалізація”: $\alpha_1 =$ „індус”, $\alpha_2 =$ „індустрія”, $\alpha_3 =$ „ліза”, $\alpha_4 =$ „акція”?
- 1.7. Випишіть всі (власні) префікси, суфікси і підслова слова $abca$.
- 1.8. Скільки формальних мов можна побудувати на алфавіті $X = \{0, 1, 2, 3\}$, в яких довжина кожного слова не перевищує 3?
- 1.9. Скільки слів ω довжини $|\omega| \leq n$ можна побудувати в алфавіті, що містить k букв? Скільки формальних мов можна утворити з цих слів?
- 1.10. Знайдіть всі слова ω в алфавіті $\{0, 1\}$, які задовольняють рівність $\omega 011 = 011\omega$.
- 1.11. Чи правда, що якщо формальна мова A містить n слів, а мова B – m слів, то AB мусить містити nm слів?
- 1.12. Нехай $A = \{(01)^n : n \geq 0\}$ і $B = \{01, 010\}$. Знайти $A \cup B$, $A \cap B$, $A \setminus B$, $B \setminus A$, $A \Delta B$, AB і ABA .

1.13. Нехай $A = \{\text{пра}, e\}$, $B = \{\text{дід}, \text{баба}\}$. Знайти AB і A^*B .

1.14. Чи може мова L^* або L^+ бути порожньою? За яких умов мови L^* і L^+ є скінченними?

1.15. Знайти слово найменшої довжини в алфавіті $\{0\}$, яке не належить формальній мові $\{e, 0, 0^2, 0^5\}^3$.

1.16. Довести рівність: $(A \cup B)^* = A^*(BA^*)^*$.

1.17. Нехай A і B – формальні мови. Довести або спростувати рівності:

- 1) $(A^R)^* = (A^*)^R$;
- 2) $(A^+)^* = A^*$;
- 3) $(A \cup A^R)^* = A^* \cup (A^*)^R$;
- 4) $A^2 \cup B^2 = (A \cup B)^2$;
- 5) $A^* \cap B^* = (A \cap B)^*$.

1.18. Які з наступних мов мають префіксну (суфіксну) властивість?

- 1) \emptyset ;
- 2) $\{e\}$;
- 3) $\{a^n b^n : n \geq 1\}$;
- 4) L^* , якщо L має префіксну властивість;
- 5) $\{\omega : \omega \in \{a, b\}^* \text{ і кількість букв } a \text{ в } \omega \text{ дорівнює кількості букв } b\}$.

1.19. Нехай h – гомоморфізм, визначений рівностями $h(0) = a$, $h(1) = bb$ і $h(2) = e$. Опишіть формальну мову $\bar{h}(L)$, де $L = \{012\}^*$.

§ 2. Системи числення

В цифрових обчислювальних машинах будь-яка інформація, задана певною формальною мовою, зображується в вигляді строго визначеної послідовності цифр. Кожній такій послідовності можна поставити в відповідність певне число і звести, таким чином, будь-яке перетворення інформації до операцій над числами, заданих у деякій системі числення. В зв'язку з цим у даному параграфі ми детально зупинимося на різних числових системах.

Система числення (англ. number (numeration) system, notation) – це сукупність способів і засобів запису чисел для проведення підрахунків.

Найпростішою системою числення є *унарна система числення*, в якій кожне натуральне число зображується відповідною кількістю деяких символів. Наприклад, якщо вибрати символ „|”, то натуральне число сім запишеться у вигляді ||||| |. Унарна система числення використовується для запису малих чисел і має застосування в теорії алгоритмів.

2.1. Типи систем числення. Розрізняють такі типи систем числення: *позиційні*, *непозиційні* та *змішані*. Найбільш поширеними є позиційні системи числення. У них одна і та ж цифра (числовий знак) у записі числа набуває різних значень залежно від своєї позиції. Кожна позиція з присвоєним їй порядковим номером називається *розрядом числа*. Домовимося про наступну нумерацію розрядів: якщо число має n розрядів цілої і m розрядів дробової частини, то старшому розряду цілої частини присвоюється номер $n-1$, молодшому розряду цілої частини – номер 0, старшому розряду дробової частини – номер -1, а молодшому розряду – номер $-m$. В позиційних системах числення кожному розряду присвоюється відповідна *вага*. Найчастіше використовуються позиційні системи числення, в яких i -му розряду присвоюється вага g^i , де $g \in \mathbb{N} \setminus \{1\}$. Натуральне число g при цьому називають *основною системою числення*. Значення числа за його зображенням визначається за формулою:

$$(1) \quad (a_{n-1}a_{n-2} \dots a_i \dots a_1a_0, a_{-1}a_{-2} \dots a_{-m})_g = \sum_{i=-m}^{n-1} a_i g^i =$$

$$a_{n-1}g^{n-1} + a_{n-2}g^{n-2} + \dots + a_i g^i + \dots + a_1 g + a_0 g^0 + a_{-1} g^{-1} + \dots + a_{-m} g^{-m}$$

В лівій частині формули (1) записано символічне зображення числа. Тут a_i ($i = n-1, n-2, \dots, -m$) – символи, які позначають деякі цілі числа. Права частина формули (1) визначає правило обчислення числового значення символічного запису лівої частини цієї формули. Можна вважати, що ліва частина формули (1) є шифром

числа, а права частина визначає алгоритм дешифрування даного шифру.

2.1. ПРИКЛАД. Обчислимо значення числа $10(-1)(-1),01$ в системі числення з основою 3 і символами -1 , 0 та 1 . В цьому випадку кількість цілих розрядів $n = 4$, а кількість дробових розрядів $m = 2$. З допомогою формули (1) одержуємо:

$$10(-1)(-1),01 = 1 \cdot 3^3 + 0 \cdot 3^2 + (-1) \cdot 3 + (-1) + 0 \cdot 3^{-1} + 1 \cdot 3^{-2} = 23\frac{1}{9}.$$

Найчастіше (хоча і не завжди) символи a_i позначають цілі числа від 0 до $g - 1$ і називаються *цифрами* даної системи числення, а сама система числення називається *g-ковою системою числення*. В *g*-ковій системі числення кількість різних цифр дорівнює основі *g* системи числення. Наприклад, в звичайній арабській (десятковій) системі числення використовуються цифри $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$.

У *непозиційних системах числення* величина, яку позначає цифра, не залежить від її позиції у числі. При цьому система може накладати обмеження на позиції цифр, наприклад, щоб вони були розташовані по спаданню чи згруповані за значенням. Проте це не є принциповою умовою для розуміння записаних такими системами чисел.

Типовим прикладом непозиційної системи числення є *римська система числення*. У ній в якості цифр використовуються латинські букви:

Римська цифра	Десяткове значення
<i>I</i>	1
<i>V</i>	5
<i>X</i>	10
<i>L</i>	50
<i>C</i>	100
<i>D</i>	500
<i>M</i>	1000

Натуральні числа записуються за допомогою повторення цих цифр. При цьому, якщо більша цифра стоїть перед меншою, то вони додаються (принцип додавання), якщо ж менша — перед більшою, то менша віднімається від більшої (принцип віднімання). Останнє

правило застосовується тільки для уникнення чотириразового повторення однієї цифри. Римські цифри I , X , C ставляться відповідно перед X , C , M для позначення 9, 90, 900 або перед V , L , D для позначення 4, 40, 400. Наприклад, $VI = 5 + 1 = 6$, $IV = 5 - 1 = 4$, $XIX = 10 + 10 - 1 = 19$ (замість $XVIII$), $XL = 50 - 10 = 40$, $XXXIII = 10 + 10 + 10 + 1 + 1 + 1 = 33$ тощо.

Римські числа використовувалися стародавніми римлянами. Виконання арифметичних дій над багатозначними числами в цій системі дуже незручне. Дана система числення на сьогодні майже не застосовується. З допомогою таких чисел позначають століття (XV ст.), роки н.е. ($MCMXXVII$) та місяці в датах ($1.V.1975$), порядкові числівники, а також похідні невеликих порядків (y^{IV} , y^V). Часто римські числа використовують також із естетичною метою.

Змішана система числення є узагальненням системи числення з основою g і її часто відносять до позиційних систем числення. Основою змішаної системи є послідовність чисел, що зростає, $\{b_k\}_{k=0}^n$ і кожне число x записується як лінійна комбінація: $x = \sum_{k=0}^n a_k b_k$, де на коефіцієнти a_k (цифри) накладаються деякі обмеження. Якщо $b_k = g^k$ для деякого g , а $a_k \in \{0, 1, \dots, g-1\}$, то змішана система співпадає з g -ковою системою числення.

Найвідомішим прикладом змішаної системи числення є зображення часу у вигляді кількості діб, годин, хвилин і секунд. При цьому величина d днів h годин m хвилин s секунд відповідає значенню $d \cdot 24 \cdot 60 \cdot 60 + h \cdot 60 \cdot 60 + m \cdot 60 + s$ секунд.

Цікавим прикладом є *система числення майя*. Майя використовували двадцяткову систему числення за одним винятком: у другому розряді було не 20, а 18 ступенів, тобто після числа (17)(19) відразу йшло число (1)(0)(0). Це було зроблено для полегшення розрахунків календарного циклу, оскільки (1)(0)(0) дорівнювало 360, що приблизно дорівнює кількості днів у сонячному році.

У нумізматиці особливо велику вагу мають десяткова, дванадцяткова (дуодецимальна), четвіркова та шісткова системи числення. У інформаційних технологіях застосовуються двійкова, десяткова, вісімкова та шістнадцяткова системи числення.

2.2. Позиційні g -кові системи числення. Зупинимось детальніше на g -кових позиційних системах числення. Спершу доведемо наступну лему.

2.2. ЛЕМА. *Якщо $a, g \in \mathbb{Z}$ і $g > 0$, то існують такі єдині числа $s, r \in \mathbb{Z}$, що $a = sg + r$, де $0 \leq r < g$.*

ДОВЕДЕННЯ. Розглянемо множину всіх цілих чисел виду $a - xg$, де $x \in \mathbb{Z}$. Нехай $r = a - sg$ – найменший невід’ємний елемент цієї множини. Ми стверджуємо, що $0 \leq r < g$. Дійсно, в протилежному випадку $a - sg \geq g$, а тому $0 \leq a - (s + 1)g < r$, що суперечить мінімальності r . Доведемо єдиність даного зображення. Припустимо, що існує інший запис $a = \tilde{s}g + \tilde{r}$, де $0 \leq \tilde{r} < g$. Тоді $a = sg + r = \tilde{s}g + \tilde{r}$, звідки $(s - \tilde{s})g = \tilde{r} - r$ і $-g < \tilde{r} - r < g$. Оскільки $\tilde{r} - r$ ділиться на g , то $\tilde{r} - r = 0$, а отже, $\tilde{r} = r$ і $\tilde{s} = s$. \square

Нехай $g \in \mathbb{N} \setminus \{1\}$ – основа g -кової системи числення.

2.3. ТЕОРЕМА. *Кожне натуральне число m можна єдиним чином подати у вигляді*

$$m = a_n g^n + a_{n-1} g^{n-1} + \dots + a_1 g + a_0,$$

де a_i ($0 \leq i \leq n$) – деякі цілі невід’ємні числа, менші від g , причому $a_n \neq 0$.

ДОВЕДЕННЯ. Згідно з лемою 2.2 для натуральних чисел m та g знайдуться такі єдині цілі числа s_0 і a_0 , що $m = s_0 g + a_0$ і $0 \leq a_0 < g$. Оскільки $g \geq 2$, то $0 \leq s_0 < m$. Якщо $s_0 = 0$, то теорема доведена, інакше існують такі єдині цілі числа s_1 і a_1 , що $s_0 = s_1 g + a_1$, $0 \leq a_1 < g$ і $0 \leq s_1 < s_0$. Продовжуючи даний процес аналогічно, одержимо строго спадну послідовність (s_k) натуральних чисел. Дана послідовність не може бути нескінченною, а тому існує таке n , що $s_{n-1} \neq 0$, але $s_n = 0$. Звідси одержуємо, що $s_{n-2} = s_{n-1} g + a_{n-1}$ і $s_{n-1} = s_n g + a_n = a_n$, де $0 \leq a_i < g$. Тоді шуканий єдиний запис числа m матиме вигляд $m = s_0 g + a_0 = (s_1 g + a_1) g + a_0 = s_1 g^2 + a_1 g + a_0 = \dots = s_{n-2} g^{n-1} + a_{n-2} g^{n-2} + \dots + a_1 g + a_0 = (s_{n-1} g + a_{n-1}) g^{n-1} + a_{n-2} g^{n-2} + \dots + a_1 g + a_0 = a_n g^n + a_{n-1} g^{n-1} + a_{n-2} g^{n-2} + \dots + a_1 g + a_0$. \square

Можна довести, що для кожного додатнього дробового числа a так само, як і для кожного натурального числа, в системі числення з будь-якою основою g існує тільки єдиний запис виду

$$a = \sum_{i=-m}^n a_i g^i = a_n g^n + \dots + a_i g^i + \dots + a_0 g^0 + a_{-1} g^{-1} + \dots + a_{-m} g^{-m}.$$

Отже, *запис числа a в системі числення з основою g* – це зображення числа у вигляді суми степенів основи g з цілими невід’ємними коефіцієнтами, меншими ніж основа g .

Вираз

$$a = a_n g^n + \dots + a_i g^i + \dots + a_0 g^0 + a_{-1} g^{-1} + \dots + a_{-m} g^{-m}.$$

скорочено записують так

$$a = (a_n a_{n-1} \dots a_i \dots a_1 a_0, a_{-1} a_{-2} \dots a_{-m})_g.$$

Введення знака мінус дає змогу записувати у системі числення з основою g також і від’ємні числа.

2.3. Арифметичні операції в різних системах числення.

При виконанні арифметичних операцій над числами, записаними в десятковій системі числення, ми користуємось правилами додавання, віднімання і множення чисел „стовпцем” і ділення „кутом”. За цими ж правилами виконують операції й над числами, записаними в будь-якій іншій системі числення. Доведемо, наприклад, правило додавання в системі числення з основою g . Нехай $a = (a_k \dots a_1 a_0)_g$ і $b = (b_s \dots b_1 b_0)_g$. Не втрачаючи загальності, можна вважати, що $k \geq s$. Знайдемо суму $a + b$.

$$\begin{aligned} a + b &= (a_k \dots a_1 a_0)_g + (b_s \dots b_1 b_0)_g = (a_k g^k + \dots + a_{s+1} g^{s+1} + a_s g^s + \\ &\dots + a_1 g + a_0) + (b_s g^s + \dots + b_1 g + b_0) = a_k g^k + \dots + a_{s+1} g^{s+1} + (a_s + \\ &b_s) g^s + \dots + (a_1 + b_1) g + (a_0 + b_0). \end{aligned}$$

У цьому записі деякі з чисел $a_0 + b_0, a_1 + b_1, \dots, a_s + b_s$ можуть виявитися більшими або дорівнювати основі числення g . Якщо $a_m + b_m \geq g$ ($0 \leq m \leq s$), то $a_m + b_m = g + r_m$, де $0 \leq r_m < g$. Тому $(a_{m+1} + b_{m+1}) g^{m+1} + (a_m + b_m) g^m = (a_{m+1} + b_{m+1} + 1) g^{m+1} + r_m g^m$. Замінивши кожен вираз $(a_m + b_m) \geq g$, починаючи з $a_0 + b_0$ і закінчуючи $a_s + b_s$, рівнозначним йому виразом $g + r_m$ і перенісши там, де це потрібно, одиницю в наступний розряд, дістанемо запис суми $a + b$ в системі числення з основою g :

$$a + b = c_t g^t + \dots + c_1 g + c_0,$$

або скорочено $a + b = (c_t \dots c_1 c_0)_g$.

Таким чином, щоб додати два цілі додатні числа, записані в системі числення з основою g , потрібно додати їхні цифри першого розряду, потім цифри другого розряду і т.д. При цьому кожного разу, коли при додаванні цифр даного розряду дістанемо суму більшу або рівну основі системи числення g , потрібно перенести одиницю в наступний розряд. При додаванні чисел доданки доцільно підписувати один під одним (у стовпчик) так, щоб цифри, які відповідають однаковим розрядам, були одна під одною. Як бачимо, додавання багатоцифрових чисел зводиться до додавання одноцифрових чисел. Таким чином, основою додавання системних чисел є таблиця додавання одноцифрових чисел, що визначає суму двох чисел, менших, ніж основа g .

Можна показати, що віднімання, множення і ділення чисел виконується цілком аналогічно правилам звичайної шкільної десятичної арифметики.

Побудуємо, наприклад, таблиці додавання і множення одноцифрових чисел в четвірковій системі числення:

+	0	1	2	3
0	0	1	2	3
1	1	2	3	10
2	2	3	10	11
3	3	10	11	12

·	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	10	12
3	0	3	12	21

У кожній клітинці цих таблиць записано у четвірковій системі числення суму (добуток) цифр, що є номерами рядка і стовпця, на перетині яких стоїть дана клітинка. Користуючись цими таблицями і правилами, аналогічними правилам десятичної арифметики, виконаємо арифметичні операції над числами в четвірковій системі числення:

$$\begin{array}{r}
 +203, 2_4 \\
 132, 1_4 \\
 \hline
 1001, 3_4
 \end{array}
 \quad
 \begin{array}{r}
 -203, 2_4 \\
 132, 1_4 \\
 \hline
 11, 1_4
 \end{array}
 \quad
 \begin{array}{r}
 102_4 \\
 21_4 \\
 \hline
 102_4 \\
 +210_4 \\
 \hline
 2202_4
 \end{array}$$

2.4. Переведення цілих чисел з однієї позиційної системи числення в іншу. У процесі розв'язування задач доводиться переводити цілі числа з однієї позиційної системи в іншу. Як же перевести число a , записане в системі числення з основою s , в систему числення з основою g ? Як відомо, записати число a в системі числення з основою g – це зобразити його у вигляді суми $a = a_k g^k + \dots + a_1 g + a_0$. Знайдемо коефіцієнти a_0, a_1, \dots, a_k . Поділимо в системі числення з основою s число a на g , дістанемо $a = b_0 g + a_0$, де $a_0 < g$. Далі поділимо b_0 на g : $b_0 = b_1 g + a_1$, де $a_1 < g$. Звідси

$$a = b_0 g + a_0 = (b_1 g + a_1) g + a_0 = b_1 g^2 + a_1 g + a_0.$$

Потім поділимо b_1 на g і т.д. Цей процес продовжуватимемо доти, доки не отримаємо частку, яка дорівнює нулю. Внаслідок цього матимемо:

$$a = a_k g^k + \dots + a_1 g + a_0.$$

Оскільки за теоремою 2.3 число a можна подати у такому вигляді єдиним чином, то a_0, a_1, \dots, a_k є цифрами числа a в системі числення з основою g .

Таким чином, для переведення натурального числа a , заданого в s -ковій системі числення, в g -кову систему числення користуємось наступним правилом: *спочатку записуємо число a в s -ковій системі, а потім виконуємо (в s -ковій системі числення!) кілька ділень числа a_s і послідовно утворюваних часток на число g_s доти, доки не отримаємо частку, яка дорівнює нулю. Здобуті остачі спочатку виражаємо цифрами g -кової системи числення і записуємо в зворотньому порядку. Записані таким чином остачі (це вже цифри в g -ковій системі числення) і є g -ковим записом числа a_s .*

2.4. ПРИКЛАД. Запишемо число $a = 47_{10}$ в трійковій системі числення:

$$\begin{aligned} 47 &= 15 \cdot 3 + \underline{2} \\ 15 &= 5 \cdot 3 + \underline{0} \\ 5 &= 1 \cdot 3 + \underline{2} \\ 1 &= \underline{0} \cdot 3 + \underline{1} \end{aligned}$$

Звідси випливає, що число a в трійковій системі числення запишеться $a = 1202_3$.

2.5. ПРИКЛАД. Переведемо число $a = 12210_3$ з трійкової системи числення у вісімкову. Нова основа у трійковій системі числення дорівнює 22. Поділимо послідовно у трійковій системі числення число $a = 12210_3$ на 22_3

$$\begin{aligned} 12210_3 &= 201_3 \cdot 22_3 + \underline{11_3} \\ 201_3 &= 2_3 \cdot 22_3 + \underline{10_3} \\ 2_3 &= 0_3 \cdot 22_3 + \underline{2_3} \end{aligned}$$

Результати послідовного ділення записані у трійковій системі числення, але ми знаємо, що $11_3 = 4_8$ і $10_3 = 3_8$. Отже, $a = 234_8$.

Перехід від g -кової системи числення до десяткової виконують ще й так. Число a_g записують у вигляді

$$a_g = a_n g^n + \dots + a_1 g + a_0.$$

Потім замість чисел a_n, \dots, a_1, a_0 і g підставляють їхні десяткові записи і роблять відповідні обчислення. Десятковий запис результату є шуканим числом.

У випадку, коли $g = s^k$ перехід від однієї системи числення до іншої значно спрощується. Дійсно, нехай маємо два записи числа a в g -ковій і s -ковій системах числення:

$$a = (a_n \dots a_1 a_0)_g = (b_l \dots b_1 b_0)_s, \text{ тобто}$$

$$a = a_n g^n + \dots + a_1 g + a_0 = b_l s^l + \dots + b_1 s + b_0.$$

Покажемо спочатку, що для довільних c_0, c_1, \dots, c_{k-1} , $0 \leq c_i < s$,

$$(2) \quad c_{k-1} s^{k-1} + \dots + c_1 s + c_0 < g.$$

Дійсно,

$$c_{k-1} s^{k-1} + \dots + c_1 s + c_0 \leq (s-1) s^{k-1} + \dots + (s-1) s + (s-1) = s^k - 1 < s^k.$$

З огляду на (2) остача a_0 від ділення числа $a = b_l s^l + \dots + b_k s^k + b_{k-1} s^{k-1} + \dots + b_1 s + b_0 = (b_l s^{l-k} + \dots + b_k) s^k + (b_{k-1} s^{k-1} + \dots + b_1 s + b_0)$ на $g = s^k$ дорівнює $b_{k-1} s^{k-1} + \dots + b_1 s + b_0$. Отже, $(a_0)_g = (b_{k-1} \dots b_1 b_0)_s$. Далі з рівності

$$a_n g^{n-1} + \dots + a_2 g + a_1 = b_l s^{l-k} + \dots + b_{k+1} s + b_k.$$

неповних часток від ділення a на $g = s^k$ випливає, що

$$(a_1)_g = (b_{2k-1} \dots b_{k+1} b_k)_s.$$

Аналогічно можна одержати зображення решти g -кових цифр в вигляді k -розрядних s -кових чисел. Таким чином, для переведення числа, записаного в g -ковій системі числення, в s -кову систему числення достатньо кожному g -кову цифру замінити рівним їй k -розрядним s -ковим числом.

2.6. ПРИКЛАД. Переведемо число $a = 765,163_8$ в двійкову систему числення:

$$a = 765,163_8 = \underbrace{111}_7 \underbrace{110}_6 \underbrace{101}_5, \underbrace{001}_1 \underbrace{110}_6 \underbrace{011}_3 = 111110101,001110011_2.$$

Для переведення числа з двійкової в вісімкову систему числення достатньо розбити його праворуч і ліворуч від коми на тріади і замінити кожен тріаду відповідною їй вісімковою цифрою. Якщо при розбитті крайні тріади виявляться неповними, то їх слід доповнити нулями.

2.7. ПРИКЛАД. Переведемо число $a = 1011,1_2$ у вісімкову систему числення:

$$a = 1011,1_2 = \underbrace{001}_1 \underbrace{011}_3, \underbrace{100}_4 = 13,4_8.$$

Розглянемо метод переведення правильних дробів з однієї системи числення в іншу. Нехай правильний дріб a , заданий в s -ковій системі числення, потрібно записати в системі числення з основою g . Припустимо, що запис числа a в g -ковій системі числення знайдений:

$$a = (0, a_{-1}a_{-2} \dots a_{-m})_g = a_{-1}g^{-1} + a_{-2}g^{-2} + \dots + a_{-m}g^{-m}.$$

Якщо помножити число a на основу g нової системи числення, то ціла частина добутку буде дорівнювати a_{-1} , а дробова –

$$a_1 = a_{-2}g^{-1} + \dots + a_{-m}g^{-m+1}.$$

Число a_1 є правильним дробом, оскільки всі цифри $a_i < g$. Таким чином, в результаті множення числа a на g отримуємо цілу частину, яка дорівнює значенню цифри старшого розряду дробового g -кового числа. Продовжуючи множення дробових частин a_i на основу g (цілі частини при кожному множенні відкидаються), можна одержати решту цифр шуканого дробу: ними є цілі частини одержаних добутків. При цьому, якщо $g < s$, то цілі частини добутку є цифрами g -кової системи числення, а якщо $g > s$, то цілі

частини є числами s -кової системи числення, які необхідно замінити цифрами g -кової системи.

Сформулюємо правило переходу від однієї системи числення до іншої методом множення на основу: *щоб перетворити правильний дріб з однієї системи числення в іншу, необхідно послідовно множити це число і проміжні значення на основу нової системи числення (записану в старій системі), відкидаючи кожен раз цілі частини добутків; ці цілі частини є записом цифр шуканого числа g -кової системи числення.*

2.8. ПРИКЛАД. Переведемо десяткове число $a = 0,6875_{10}$ в двійкову систему числення:

$$\begin{array}{r}
 \times \underline{0},6875 \\
 2 \\
 \hline
 \times \underline{1},3750 \\
 2 \\
 \hline
 \times \underline{0},7500 \\
 2 \\
 \hline
 \times \underline{1},5000 \\
 2 \\
 \hline
 \underline{1},0000
 \end{array}$$

Отже, $a = 0,1011_2$

Слід зауважити, що в загальному випадку дробові числа переводяться з однієї системи числення в іншу наближено, тобто дробова частина добутку при послідовному множенні на основу ніколи не дорівнюватиме нулю. В цьому випадку процес множення продовжують доти, доки не буде одержана необхідна для досягнення заданої точності запису числа кількість розрядів g -кового числа.

2.9. ПРИКЛАД. Число $\frac{1}{5}$ в десятковій системі числення записують скінченним дробом, а в дванадцятковій – нескінченним:

$$\frac{1}{5} = 0,2_{10} = 0,249724972497 \dots_{12};$$

число $\frac{1}{6}$, навпаки, в дванадцятковій системі числення записують скінченним дробом, а в десятковій – нескінченним:

$$\frac{1}{6} = 0,2_{12}, \frac{1}{6} = 0,1666 \dots_{10}.$$

В обох цих системах число $\frac{1}{4}$ запишеться скінченним дробом, а число $\frac{1}{7}$ – нескінченним:

$$\frac{1}{4} = 0,25_{10} = 0,3_{12}, \frac{1}{7} = 0,142857142857..._{10} = 0,186035186035..._{12}.$$

Для переведення неправильного дробу з однієї системи числення в іншу, слід цілу частину отримати методом ділення, а дробову – методом множення на основу нової системи числення.

Рекомендована література : [3, с. 58–80], [9, с. 4–11], [10, с. 79–88], [19, с. 70–89].

Питання та вправи до параграфа 2.

- 2.1.** На які типи поділяються системи числення?
- 2.2.** Які цифри використовуються для запису чисел в римській системі числення?
- 2.3.** Опишіть процес переведення чисел з двійкової системи числення в четвіркову і навпаки.
- 2.4.** Записати в десятковій системі числення такі числа римської нумерації: а) *CXVI*; б) *CXIV*; в) *DXCVI*; г) *MCCII*; д) *MCDXXIX*; е) *MDCCCLXXIV*.
- 2.5.** Записати в римській системі числення числа: а) 39; б) 93; в) 2011; г) 1999.
- 2.6.** Обчислити:
- $(3333_4 + 2222_4) \cdot 12_4$;
 - $2011_8 + 2012_8 - 4321_8$;
 - $20671_8 / 131_8 - 137_8$;
 - $120111_3 / 102_3 + 201_3 \cdot 12_3 - 11220_3$;
 - $232011_5 / 104_5 + 1234_5 \cdot 322_5 - 1022131_5$;
 - $3215_7 \cdot 24_7 - 11461_7 / 25_7 + 1532_7 - 115044_7$.
- 2.7.** Записати в десятковій системі числення такі числа:
- | | |
|------------------------|----------------------|
| а) 100001101_2 ; | г) $0,221_3$; |
| б) 7563401_8 ; | д) $11001,11001_2$; |
| в) $(10)7(11)9_{12}$; | е) $437,3201_8$. |

2.8. Перевести з однієї системи числення в іншу. Зробити перевірку.

- | | |
|-------------------------------------|---|
| а) $124_{10} \rightarrow x_4$; | г) $32014_5 \rightarrow x_8$; |
| б) $1340_{10} \rightarrow x_{14}$; | д) $33311_7 \rightarrow x_3$; |
| в) $101000_3 \rightarrow x_4$; | е) $34(11)57_{12} \rightarrow x_{11}$. |

2.9. Перевести з однієї системи числення в іншу.

- | | |
|--------------------------------------|--|
| а) $10111001101_2 \rightarrow x_4$; | г) $1201222011, 2111_3 \rightarrow x_9$; |
| б) $1330_4 \rightarrow x_2$; | д) $3387656, 3455_9 \rightarrow x_3$; |
| в) $101765_8 \rightarrow x_2$; | е) $1011100010101, 1_2 \rightarrow x_{16}$. |

2.10. Перевести число $2012, 2012_{10}$ з десятикової системи числення в двійкову, трійкову та четвіркову системи числення.

2.11. Знайти x , якщо:

- | | |
|------------------------|------------------------|
| а) $201_x = 41_8$; | д) $324_x = 10022_3$; |
| б) $203_x = 53_{10}$; | е) $364_x = 3001_4$; |
| в) $106_x = 153_7$; | є) $401_x = 265_7$; |
| г) $236_x = 1240_5$; | ж) $541_x = 2014_6$. |

2.12. Знайти частку від ділення числа $(62xy427)_{10}$ на 99_{10} , а також невідомі цифри x і y , якщо ділення виконується без остачі.

2.13. Знайти таке найменше натуральне число m , яке в десятичній системі числення закінчується цифрою 6, причому, якщо цю цифру записати на початку числа, то воно збільшиться в 4 рази.

2.14. Число $(42x4y)_{10}$ ділиться на 72_{10} . Знайти цифри x і y .

2.15. Нехай $2 \cdot (xyz)_{10} = (xyz)_g$. Знайти $(xyz)_{10}$ і g .

2.16. Знайти число n , якщо $n = (abc)_7 = (cba)_9$.

2.17. Знайти 8-цифрове число $n = (abcdefgh)_{10}$, якщо числа $(abcd)_{10}$ та $(efgh)_{10}$ відрізняються на одиницю.

§ 3. Регулярні мови і регулярні вирази

В цьому параграфі опишемо важливий клас формальних мов – регулярні мови і їх задання з допомогою регулярних виразів, а також розглянемо алгебраїчні закони для регулярних виразів. Ці закони багато в чому подібні до алгебраїчних законів арифметики, проте між арифметичними і регулярними виразами є і суттєві відмінності. Регулярні вирази тісно пов'язані зі скінченими автоматами, які ми розглядатимемо в наступному розділі. Їх можна також розглядати як „мову програмування” для опису деяких важливих додатків, наприклад, програм текстового пошуку чи компонентів компілятора.

Нехай X – довільний алфавіт. Визначимо *регулярну мову (множину) в алфавіті X* рекурсивно наступним чином:

- 1) порожня множина \emptyset є регулярною мовою;
- 2) для кожної букви $a \in X$ множина $\{a\}$ є регулярною мовою;
- 3) якщо A і B є регулярними мовами, то множини $A \cup B$, AB і A^* також є регулярними мовами;
- 4) інших регулярних мов не існує.

3.1. ПРИКЛАД. Безпосередньо з означення випливає:

- 1) множина $\{e\}$ є регулярною мовою, бо $\{e\} = \emptyset^*$;
- 2) множина $\{001, 110\}$ – регулярна мова над бінарним алфавітом, бо $\{001, 110\} = (\{0\}\{0\}\{1\}) \cup (\{1\}\{1\}\{0\})$;
- 3) аналогічно, як і в пункті 2), можна показати, що будь-яка скінченна мова є регулярною.

Щоб спростити запис регулярних мов, визначимо поняття *регулярного виразу над алфавітом X* наступним чином:

- 1) порожня множина \emptyset є регулярним виразом, який позначає порожню множину;
- 2) e є регулярним виразом, який позначає мову $\{e\}$;
- 3) a – регулярний вираз, що позначає мову $\{a\}$ для кожної букви $a \in X$;
- 4) якщо r_A і r_B є регулярними виразами, що позначають мови A і B відповідно, то $(r_A) + (r_B)$, $(r_A)(r_B)$ і $(r_A)^*$ є регулярними виразами, що позначають мови $A \cup B$, AB і A^* відповідно;
- 5) інших регулярних виразів над алфавітом X не існує.

Якщо r – регулярний вираз, то через $L(r)$ позначатимемо відповідну йому регулярну мову. Щоб зменшити кількість дужок в регулярних виразах, будемо вважати, що найвищий пріоритет має операція ітерації $*$, потім виконується конкатенація і останньою – операція $+$. Наприклад, запис регулярного виразу $((0)(1))|((1)(0))$, який позначає мову $\{01, 10\}$, спрощується до $01|10$, вираз 0^* позначає $\{0\}^*$, $(0|1)^* = \{0, 1\}^*$, а $(0|1)^*001$ позначає множину всіх слів, які складаються з нулів і одиниць та закінчуються словом 001 .

Зрозуміло, що для кожної регулярної мови можна знайти принаймні один регулярний вираз, який її позначає. І навпаки, для кожного регулярного виразу можна побудувати регулярну мову, що задається цим виразом. На жаль, для регулярної мови існує багато виразів. Наприклад, вирази $0^*1 + \emptyset$ і 0^*1 позначають одну й ту ж мову $\{0\}^*\{1\}$. Кажемо, що два регулярних вирази рівні ($=$), якщо вони позначають одну і ту ж регулярну мову.

3.2. ТВЕРДЖЕННЯ. *Нехай α , β і γ – регулярні вирази. Тоді:*

- $\alpha + \beta = \beta + \alpha$;
- $\emptyset^* = e$;
- $\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$;
- $\alpha(\beta\gamma) = (\alpha\beta)\gamma$;
- $\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma$;
- $(\alpha + \beta)\gamma = \alpha\gamma + \beta\gamma$;
- $\alpha e = e\alpha = \alpha$;
- $\emptyset\alpha = \alpha\emptyset = \emptyset$;
- $\alpha^* = \alpha + \alpha^*$;
- $(\alpha^*)^* = \alpha^*$;
- $\alpha + \alpha = \alpha$;
- $\alpha + \emptyset = \alpha$.

ДОВЕДЕННЯ. Нехай α і β позначають мови A і B відповідно. Тоді $\alpha + \beta$ позначає $A \cup B$, а $\beta + \alpha$ позначає $B \cup A$. Але $A \cup B = B \cup A$ за означенням об'єднання, тому $\alpha + \beta = \beta + \alpha$. Решту рівностей доводяться аналогічно. \square

Для зручності запису введемо позначення: $r^+ = rr^* = r^*r$.

Розглянемо деякі приклади.

3.3. ПРИКЛАД. Довести рівності:

- а) $a^*(a+b)^* = (a+ba^*)^*$;
 б) $(ba)^+(a^*b^* + a^*) = (ba)^*ba^+b^*$.

а) Покажемо, що обидві частини рівності дорівнюють $(a+b)^*$. Дійсно, обидві частини рівності є підмножинами $(a+b)^*$, бо $(a+b)^*$ позначає множину всіх слів в алфавіті $\{a, b\}$. Таким чином, достатньо показати, що обидві частини містять $(a+b)^*$. Оскільки $e \in a^*$, то $a^*(a+b)^* \supset (a+b)^*$. З того, що $b \in ba^*$ випливає потрібне включення $(a+b)^* \subset (a+ba^*)^*$.

$$\text{б) } (ba)^+(a^*b^* + a^*) = (ba)^*(ba)a^*(b^* + e) = (ba)^*ba^+b^*.$$

3.4. ПРИКЛАД. Регулярний вираз

$1((0+1+2+3+4)(0+1+2+3+4+5+6+7+8+9)+5(0+1+2))$
 позначає множину натуральних чисел відрізка $[100, 152]$.

3.5. ПРИКЛАД. Знайти регулярний вираз для множини всіх слів в алфавіті $\{0, 1, 2\}$, які є записами в трійковій системі чисел невід'ємних цілих степенів числа 9.

В трійковій системі числення степені 9^n запишуться як $1 \underbrace{00 \dots 0}_{2n}$.

Таким чином, шуканий регулярний вираз має вигляд $1(00)^*$.

3.6. ПРИКЛАД. Записати регулярний вираз для множини всіх слів у двійковому алфавіті, які містять підслово 001.

Кожне таке слово запишеться у вигляді $\alpha 001\beta$, де α і β – довільні слова в двійковому алфавіті. Таким чином, одержуємо шуканий регулярний вираз:

$$(0+1)^*001(0+1)^*.$$

3.7. ПРИКЛАД. Записати регулярний вираз для множини всіх слів у двійковому алфавіті, які не містять підслова 001.

Якщо слово ω належить шуканій множині, то воно не містить підслова 00, за винятком того випадку, коли воно має суфікс 0^k для $k \geq 2$. Аналогічно, як і у прикладі 1.3, можна показати, що множина слів, які не містять підслова 00, записується з допомогою регулярного виразу $(01+1)^*(e+0)$. Таким чином, множина всіх

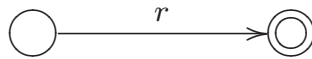
слів, які не містять підслова 001, запишеться наступним регулярним виразом:

$$(01 + 1)^*(e + 0 + 000^*) = (01 + 1)^*0^*.$$

Розглянемо рівняння $\chi = \alpha\chi + \beta$, де α і β регулярні вирази в алфавіті X і $\chi \notin X$. З твердження 1.5 випливає, що $\chi = \alpha^*\beta$ – єдиний розв’язок рівняння, якщо $e \notin \alpha$. У випадку $e \in \alpha$ рівняння має безліч розв’язків, кожен з яких, очевидно, має вигляд $\alpha^*(\beta \cup \gamma)$, де γ – довільна (не обов’язково регулярна) мова. Використовуючи твердження 1.5, можна розв’язувати системи рівнянь з регулярними коефіцієнтами (методом, аналогічним методу Гауса для розв’язування систем лінійних рівнянь).

Регулярні вирази часто зображують *поміченими орієнтованими графами*, стрілки яких позначають буквами з алфавіту $X \cup \{e\}$. Для довільного регулярного виразу r його граф утворюється наступним чином:

Спочатку малюємо дві спеціальні вершини, які називаються початковою і кінцевою вершинами, і сполучаємо їх стрілкою, позначеною r :



Далі повторюємо наступні кроки доти, доки кожна позначка довільної стрілки не міститиме символів $+$, \cdot і $*$:

1) замінюємо кожну стрілку з позначкою $f + g$ на дві паралельні стрілки з позначками f і g :



2) замінюємо кожну стрілку з позначкою fg на додаткову вершину і дві стрілки з позначками f і g :



3) замінюємо кожну стрілку з позначкою f^* на додаткову вершину і на три стрілки з позначками e , f і e :

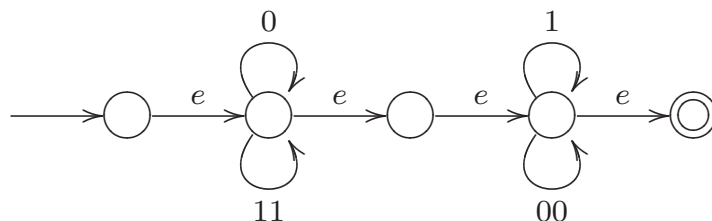
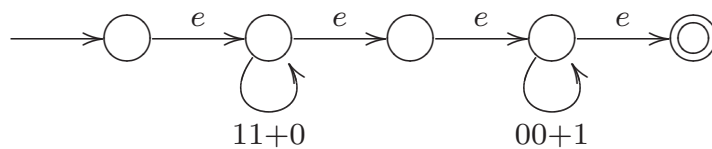
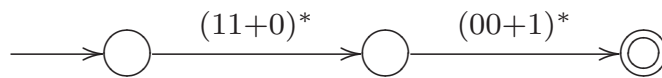
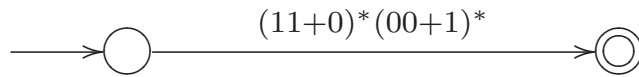


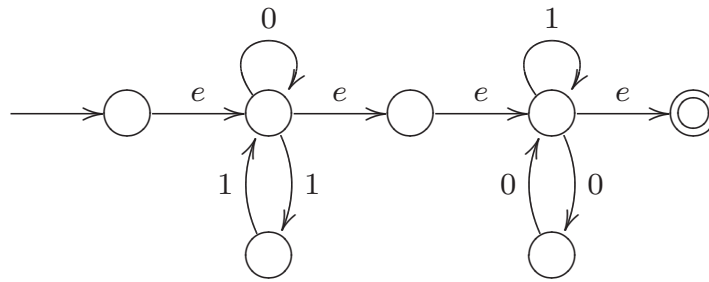
Через $G(r)$ позначимо помічений орієнтований граф регулярного виразу r . Очевидно, що кожна стрілка в $G(r)$ має позначку з множини $X \cup \{e\}$. Кожному шляху в $G(r)$ відповідає слово, яке одержується конкатенацією всіх букв, які позначають стрілки шляху. Слово x належить мові $L(r)$ тоді і тільки тоді, коли існує шлях в $G(r)$ з початкової вершини в кінцеву вершину, якому відповідає слово x . Кожна e -стрілка в $G(r)$, яка є єдиною вихідною стрілкою з некінцевої вершини чи єдиною вхідною стрілкою в непочаткову вершину, може бути стягнута в одну вершину.

3.8. ПРИКЛАД. Побудуємо граф $G(r)$ регулярного виразу

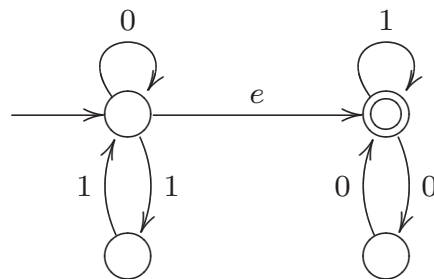
$$r = (11 + 0)^*(00 + 1)^*.$$

Зобразимо крок за кроком побудову даного графу:





Стягнувши, де це можливо, ϵ -стрілки, остаточно одержуємо помічений граф $G(r)$ регулярного виразу $r = (11 + 0)^*(00 + 1)^*$:



Рекомендована література : [2, с. 124–131], [4, с. 490–494], [5, с. 351–353], [7], [16, с. 13–31].

Питання та вправи до параграфа 3.

- 3.1.** Яка формальна мова називається регулярною?
- 3.2.** Де застосовуються регулярні вирази?
- 3.3.** Опишіть процес побудови графа регулярного виразу.
- 3.4.** Знайти слово найменшої довжини в кожній з наступних регулярних мов, заданих регулярним виразом. Які непорожні слова є найкоротшими в даних мовах?

- а) $10 + (0 + 11)0^*1$;
- б) $(00 + 11 + (01 + 10)(00 + 11)^*(01 + 10))^*$;
- в) $((00 + 11)^* + (001 + 110)^*)^*$.

- 3.5.** Визначити, чи належить рядок 1011 регулярним мовам, заданим регулярними виразами:

- | | |
|-------------------|----------------------------|
| а) 10^*1 ; | г) $(10)^+1011$; |
| б) $1(01)^*1^*$; | д) $1(00)^*(11)^*$; |
| в) $(10)^+11$; | е) $(1 + 00)(01 + 0)1^*$. |

3.6. Спростити регулярні вирази:

- а) $(00)^*0 + (00)^*$;
- б) $(0 + 1)(e + 00)^+ + (0 + 1)$;
- в) $(0 + e)0^*1$.

3.7. Доведіть рівність $(0^2 + 0^3)^* = (0^20^*)^*$.

3.8. Чи є мова $\{0^{3m+4n} \mid m, n \geq 0\}$ регулярною? Відповідь обґрунтуйте.

3.9. Визначити алфавіт та побудувати регулярний вираз для наступних мов:

- а) множина всіх непарних натуральних чисел, записаних в четвірковій системі числення;
- б) множина всіх непарних натуральних чисел, записаних в трійковій системі числення.

3.10. Побудуйте регулярний вираз для множини всіх цілих чисел відрізка $[10, 2012]$.

3.11. Розв'язати системи рівнянь з регулярними коефіцієнтами.

$$\begin{array}{ll}
 \text{а) } \begin{cases} X_1 = e + 0X_1 + 1X_2 \\ X_2 = e + 1X_2 \end{cases} & \text{в) } \begin{cases} X_1 = (01^* + 1)X_1 + X_2 \\ X_2 = 11 + 1X_1 + 00X_3 \\ X_3 = e + X_1 + X_2 \end{cases} \\
 \text{б) } \begin{cases} X_1 = 0X_2 + 1X_1 + e \\ X_2 = 0X_3 + 1X_2 \\ X_3 = 0X_1 + 1X_3 \end{cases} & \text{г) } \begin{cases} X_1 = 0X_3 + 1X_2 \\ X_2 = e + 1X_1 + 0X_3 \\ X_3 = e + 0X_1 \end{cases}
 \end{array}$$

3.12. Побудувати помічений граф для наступних регулярних виразів:

- а) $01 + 11^*$;
- б) $(00 + 10)(101)^* + 01$;
- в) $((00+11)^*+(001+110)^*)^*$;
- г) $a^*b(c + da^*b)^*$;
- д) $(a + bc^*d)^*bc^*$.

3.13. Побудувати регулярні вирази для мови всіх слів в алфавіті $\{0, 1\}$, для яких виконується наступна умова:

- а) містять підслово 000 або 111;
- б) в яких п'ятою буквою справа є 0;
- в) містять не більше однієї пари послідовних одиниць;
- г) кількість нулів кратна п'яти;
- ґ) містять непарну кількість букв 0;
- д) не містять підслова 000;
- е) не містять ні підслова 000, ні 111;
- є) не містять підслова 011;
- ж) містять непарну кількість входжень підслова 011;
- з) кількість одиниць ділиться на п'ять, а кількість нулів парна.

3.14. Нехай L – мова над алфавітом $\{0\}$. Довести, що мова L^* є регулярною. [Підказка: доведіть і використайте той факт, що якщо a та b є взаємно простими натуральними числами, то для кожного натурального $n \geq ab$ існують такі невід'ємні цілі числа u та v , що виконується рівність $n = ua + vb$.]

§ 4. Формальні породжуючі граматики

Ми визначили формальну мову L як множину слів скінченної довжини в алфавіті X . Якщо L містить скінченну кількість слів, то найбільш очевидний спосіб описати мову – скласти перелік всіх її слів. Однак для багатьох мов не можна (або, можливо, не бажано) обмежувати довжини слів. Отже, доводиться розглядати мови, які містять як завгодно багато слів. Відразу виникає важливе питання: як описати мову L в тому випадку, коли вона нескінченна. Очевидно, що такі формальні мови неможливо визначити вичерпним переліком слів, які їм належать, і тому потрібно шукати інші способи їх опису. Як і раніше, ми хочемо, щоб опис мови був скінченним, хоча сама мова може бути й нескінченною. Відомо декілька методів опису мов, які задовольняють цю вимогу. Один з них показаний в попередньому параграфі для регулярних мов. Розглянемо метод, який полягає в використанні структури, яка називається *граматикою*.

Формальною породжуючою граматикою називається четвірка виду $G = (N, T, S, P)$, де N і T – скінченні непорожні неперетинні множини, елементи яких будемо називати *нетермінальними та термінальними символами* відповідно, $S \in N$ – *початковий нетермінальний символ*, P – скінченна множина *продукцій (правил перетворення)* вигляду $\xi \rightarrow \eta$, де ξ та η – слова в алфавіті $N \cup T$. Букви термінального алфавіту позначатимемо малими латинськими буквами чи цифрами, а нетермінального алфавіту – великими латинськими буквами.

Нас цікавитимуть слова, які можуть бути породжені продукціями граматики. Нехай $G = (N, T, S, P)$ – граматика і $\alpha_0 = \sigma\xi\tau$ (тобто α_0 – конкатенація слів σ , ξ і τ), $\alpha_1 = \sigma\eta\tau$ – слова в алфавіті $N \cup T$. Якщо $\xi \rightarrow \eta$ – продукція граматики G , то кажуть, що α_1 *безпосередньо виводиться* з α_0 і записують $\alpha_0 \Rightarrow \alpha_1$. Якщо ж $\alpha_0, \alpha_1, \dots, \alpha_n$ – слова в алфавіті $N \cup T$ такі, що $\alpha_0 \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \dots \Rightarrow \alpha_{n-1} \Rightarrow \alpha_n$, то говорять, що α_n *виводиться* з α_0 і записують $\alpha_0 \Rightarrow^* \alpha_n$.

Мовою, породженою граматикою $G = (N, T, S, P)$, називають множину всіх слів, які виводяться з початкового символу S і містять тільки термінальні символи. Її позначають $L(G)$. Таким чином,

$$L(G) = \{\omega \in T^* \mid S \Rightarrow^* \omega\}.$$

4.1. ПРИКЛАД. Граматика G , яка породжує речення (слово з чотирьох букв формальної мови $L(G)$) „хороший студент одержав п’ятірку” матиме вигляд $G = (N, T, S, P)$, де

$N = \{\langle \text{речення} \rangle, \langle \text{група підмета} \rangle, \langle \text{група присудка} \rangle, \langle \text{означення} \rangle, \langle \text{підмет} \rangle, \langle \text{присудок} \rangle, \langle \text{додаток} \rangle\};$
 $T = \{\text{хороший, студент, одержав, п’ятірку}\};$
 $S = \langle \text{речення} \rangle.$

Множина продукцій P містить продукції:

$\langle \text{речення} \rangle \rightarrow \langle \text{група підмета} \rangle \langle \text{група присудка} \rangle,$
 $\langle \text{група підмета} \rangle \rightarrow \langle \text{означення} \rangle \langle \text{підмет} \rangle,$
 $\langle \text{група присудка} \rangle \rightarrow \langle \text{присудок} \rangle \langle \text{додаток} \rangle,$
 $\langle \text{означення} \rangle \rightarrow \text{хороший},$
 $\langle \text{підмет} \rangle \rightarrow \text{студент},$
 $\langle \text{присудок} \rangle \rightarrow \text{одержав},$
 $\langle \text{додаток} \rangle \rightarrow \text{п’ятірку}.$

Розглянемо виведення даної граматики з початкового нетермінального символу:

$\langle \text{речення} \rangle \Rightarrow \langle \text{група підмета} \rangle \langle \text{група присудка} \rangle \Rightarrow$
 $\langle \text{означення} \rangle \langle \text{підмет} \rangle \langle \text{група присудка} \rangle \Rightarrow$
 $\langle \text{означення} \rangle \langle \text{підмет} \rangle \langle \text{присудок} \rangle \langle \text{додаток} \rangle \Rightarrow$
 $\text{хороший} \langle \text{підмет} \rangle \langle \text{присудок} \rangle \langle \text{додаток} \rangle \Rightarrow$
 $\text{хороший студент} \langle \text{присудок} \rangle \langle \text{додаток} \rangle \Rightarrow$
 $\text{хороший студент одержав} \langle \text{додаток} \rangle \Rightarrow$
 $\text{хороший студент одержав п'ятірку}.$

Таким чином, $L(G)$ складається з єдиного слова (речення української мови), яке містить чотири термінали.

$$L(G) = \{ \text{„хороший студент одержав п'ятірку"} \}.$$

4.2. ПРИКЛАД. Нехай G – граматика з нетермінальним алфавітом $N = \{S, A\}$, множиною терміналів $T = \{a, b\}$, початковим символом S і множиною продукцій $P = \{S \rightarrow bA, S \rightarrow a, A \rightarrow bb\}$. Знайдемо мову $L(G)$, що породжується цією граматикою.

Із початкового символу S можна вивести слово bA за допомогою продукції $S \rightarrow bA$ чи застосувати продукцію $S \rightarrow a$, щоб вивести a . З bA , скориставшись продукцією $A \rightarrow bb$, можна вивести слово bbb . Оскільки інших виведень не існує, то $L(G) = \{a, bbb\}$.

4.3. ПРИКЛАД. Розглянемо граматичу $G = (N = \{S\}, T = \{0, 1\}, S, P = \{S \rightarrow e, S \rightarrow 0S1\})$ і знайдемо мову, що породжується нею.

Граматика G містить єдиний нетермінальний символ і єдине правило $S \rightarrow 0S1$, права частина якого містить цей символ. Таким чином, єдиними виведеннями граматичи G є виведення форми:

$$S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow \dots \Rightarrow 0^n S 1^n \Rightarrow 0^n e 1^n = 0^n 1^n$$

Отже,

$$L(G) = \{0^n 1^n \mid n \geq 0\}.$$

Якщо є багато правил граматичи з однаковими лівими частинами

$$A \rightarrow x_1, A \rightarrow x_2, \dots, A \rightarrow x_m,$$

то їх об'єднують в одне правило наступного вигляду:

$$A \rightarrow x_1 \mid x_2 \mid \dots \mid x_m.$$

Наприклад, правила з попереднього прикладу можуть бути записані як $S \rightarrow e \mid 0S1$.

4.4. ПРИКЛАД. Розглянемо граматику $G = (N = \{S, A, B\}, T = \{a, b\}, S, P)$, де множина продукцій P містить наступні правила:

$$S \rightarrow ABA, A \rightarrow a \mid bb, B \rightarrow bS \mid e,$$

і знайдемо мову $L(G)$.

Оскільки нетермінальний символ A може тільки продукувати термінальні символи, то його заміщення слід відкласти наостанок. Таким чином, виведення граматики G мають наступну загальну форму:

$$S \Rightarrow ABA \Rightarrow AbSA \Rightarrow AbABAA \Rightarrow AbAbSAA \Rightarrow A(bA)^2BA^3 \Rightarrow \dots \Rightarrow A(bA)^nBA^{n+1} \Rightarrow A(bA)^nA^{n+1}.$$

Насамкінець замінюємо нетермінал A терміналами a та bb , і одержуємо слова вигляду:

$$(a + bb)(ba + bbb)^n(a + bb)^{n+1}, n \geq 0.$$

Таким чином, $L(G)$ складається з усіх побудованих вище слів.

Граматики класифікують за типами продукцій. Розглянемо класифікацію, яку запропонував американський математик і лінгвіст Н. Хомський. Принцип цієї класифікації полягає в тому, що на продукції накладено певні обмеження. Він поділив граматики на чотири типи.

У граматиці *типу 0* немає жодних обмежень на продукції. Граматика *типу 1* може мати лише продукції вигляду $\xi \rightarrow \eta$, де довжина слова η не менша, ніж довжина слова ξ , або у вигляді $\xi \rightarrow e$. У граматиці *типу 2* можуть бути лише продукції $A \rightarrow \omega$, де A – нетермінальний символ, $\omega \in (N \cup T)^*$. Граматика *типу 3* може мати такі продукції: $A \rightarrow \omega B$, $A \rightarrow \omega$ і $A \rightarrow e$, де A, B – нетермінали, ω – слово з терміналів. Із цих означень випливає, що кожна граматика типу n , де $n = 1, 2, 3$, є граматикою типу $n - 1$. Граматики типу 2 називають *контекстно вільними*, бо нетермінал A в лівій частині продукції $A \rightarrow \eta$ можна замінити словом η в довільному оточенні щоразу, коли він зустрічається, тобто незалежно від контексту. Мову, яку породжує граматика типу 2, називають *контекстно вільною*. Якщо в множині продукцій P є продукція виду $\gamma\mu\delta \rightarrow \gamma\nu\delta$, $|\mu| \leq |\nu|$, то граматику називають *контекстно залежною*, оскільки μ можна замінити на ν лише в оточенні слів $\gamma \dots \delta$, тобто у відповідному контексті. Мову, яку породжує граматика типу 1, називають *контекстно залежною*. Граматику типу 3 називають *праволінійною*.

Двоїсто визначають *ліволінійну граматику*; вона може мати такі продукції: $A \rightarrow B\omega$, $A \rightarrow \omega$ і $A \rightarrow e$, де A, B – нетермінали, ω – слово з терміналів. Ліволінійні і праволінійні граматики називаються *регулярними*. В параграфі 5 наступного розділу ми доведемо, що регулярні граматики породжують регулярні мови і тільки їх.

4.5. ПРИКЛАД. Побудуємо контекстно вільну граматику, яка породжує мову $L = \{0^n 1^{2n} \mid n \geq 0\}$.

Ця мова нагадує мову, породжену граматиною з прикладу 4.2, в тому сенсі, що замінивши кожну 1 на 11, одержимо граматику для L . Отже, шукана граматика має вигляд

$$G = (\{S\}, \{0, 1\}, S, \{S \rightarrow e \mid 0S11\}).$$

Іншим методом побудови граматики є відшукування рекурсивної функції, яка пов'язує довші слова мови L з коротшими словами. Наприклад, слово $0^{n+1}1^{2(n+1)}$ можна записати як $0(0^n 1^{2n})11$. Таким чином, якщо існує виведення $S \Rightarrow^* 0^n 1^{2n}$, то потрібна продукція $S \rightarrow 0S11$, щоб одержати виведення $S \Rightarrow^* 0^{n+1}1^{2(n+1)}$. Продемонструємо цю ідею в наступному прикладі.

4.6. ПРИКЛАД. Побудуємо контекстно вільну граматику, що породжує мову $L = \{x \in \{0, 1\}^* \mid x = x^R\}$.

Оскільки k -та зліва буква кожного слова має дорівнювати k -тій справа букві, то слово $x \in L$ довжини $2n + 2$ може бути записане або як $0y0$, або $1y1$ для деякого слова $y \in L$ довжини $2n$. Отже, нам потрібні правила $S \rightarrow 0S0$ і $S \rightarrow 1S1$, щоб породити слово x рекурсивно. Зі сказаного вище випливає, що мова L породжується граматиною $G = (\{S\}, \{0, 1\}, S, P)$, де множина P містить наступні продукції:

$$S \rightarrow e \mid 0 \mid 1 \mid 0S0 \mid 1S1.$$

З чотирьох класів граматик ієрархії Хомського клас контекстно вільних граматик є найбільш важливим з точки зору застосувань до мов програмування і компіляції. З допомогою контекстно вільної граматики можна визначити більшу частину синтаксичної структури мов програмування. Крім того, вона є основою різних схем задання перекладів на машинну мову комп'ютера.

Слово ω належить мові $L(G)$ контекстно вільної граматики G тоді і лише тоді, коли існує виведення $S \Rightarrow^* \omega$ в G . Воно показує,

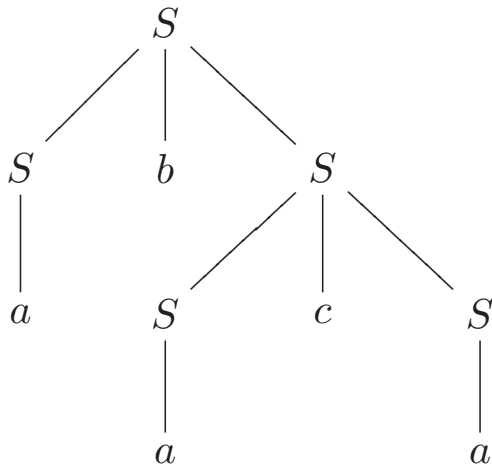
як крок за кроком застосовуються продукції граматики, щоб одержати слово ω з початкового нетермінального символу S . Це виведення можна також продемонструвати наочно з допомогою дерева, яке називатимемо *деревом виведення*. Наприклад, розглянемо контекстно вільну граматику $G = (\{S\}, \{a, b, c\}, S, P)$, де $P = \{S \rightarrow SbS \mid ScS \mid a\}$ і слово $abaca \in L(G)$. Виведення

$$S \Rightarrow SbS \Rightarrow SbScS \Rightarrow abScS \Rightarrow abSca \Rightarrow abaca$$

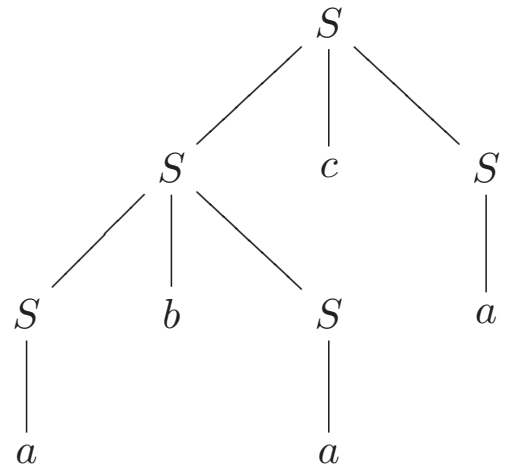
показано на рисунку (а), а виведення

$$S \Rightarrow ScS \Rightarrow SbScS \Rightarrow abScS \Rightarrow abSca \Rightarrow abaca$$

– на рисунку (б).



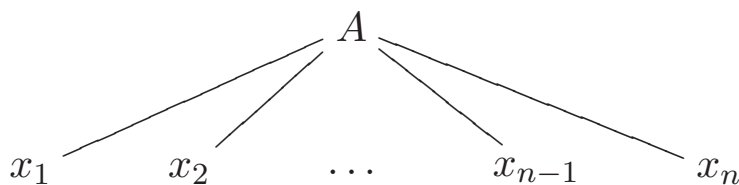
(а)



(б)

В загальному випадку для контекстно вільної граматики $G = (N, T, S, P)$ дерево виведення будується наступним чином:

1. Вершиною дерева є початковий нетермінальний символ S .
2. Повторюємо наступні кроки доти, доки кожен листок дерева буде або порожнім символом ϵ , або термінальним символом: для кожного листка $A \in N$ дерева обираємо продукцію $A \rightarrow x_1 x_2 \dots x_n$ граматики G , де $x_i \in N \cup T$, і замінюємо вершину $A \in N$ деревом:



Якщо конкатенація листків дерева виведення є словом $\omega \in T^*$, то кажемо, що дане дерево є *деревом виведення для слова ω* .

Дерево виведення слова ω відповідає виведенню ω в граматиці G . Часто дерево виведення відповідає більш ніж одному виведенню слова ω . Наприклад, для граматики G , визначеної вище, є 16 різних виведень слова $\omega = abaca$ в граматиці G . Серед них 8 виведень, які починаються з $S \Rightarrow SbS$, зображуються одним і тим же деревом виведення, показаним на рисунку (а). Решту 8 виведень, що починаються з $S \Rightarrow ScS$, зображені деревом виведення на рисунку (б).

Щоб зробити відповідність між деревами виведення і виведеннями в граматиці G взаємно однозначною, назвемо виведення слова $\omega \in L(G)$ *лівим виведенням*, якщо завжди застосовується відповідна продукція граматики G для заміни найлівішого нетермінального символу в проміжному слові при виведенні слова ω . Очевидно, що кожне дерево виведення слова ω відповідає єдиному лівому виведенню слова ω . Наприклад, серед восьми виведень для $\omega = abaca$, які починаються з $S \Rightarrow SbS$, єдиним лівим виведенням є

$$S \Rightarrow SbS \Rightarrow abS \Rightarrow abScS \Rightarrow abacS \Rightarrow abaca.$$

Аналогічно дається означення правого виведення.

Контекстно вільна граматика $G = (N, T, S, P)$ називається *неоднозначною*, якщо існує слово $\omega \in L(G)$, яке має два або більше різних дерев виведень. Якщо граматика використовується для визначення мови програмування, то бажано, щоб вона була однозначною. В іншому випадку програміст і компілятор можуть по-різному зрозуміти смисл деяких програм.

4.7. ПРИКЛАД. Найвідомішим прикладом неоднозначності в мовах програмування є „плаваюче” **else**.

Розглянемо граматику G з правилами

$$S \rightarrow \text{if } b \text{ then } S \text{ else } S \mid \text{if } b \text{ then } S \mid a.$$

Ця граматика неоднозначна, оскільки слово

$$\text{if } b \text{ then if } b \text{ then } a \text{ else } a$$

має два виводи

$$\text{if } b \text{ then (if } b \text{ then } a) \text{ else } a \text{ та}$$

$\text{if } b \text{ then } (\text{if } b \text{ then } a \text{ else } a),$

яким відповідають два різних дерева виведення.

Визначена нами неоднозначність – це властивість граматики, а не мови. Для деяких неоднозначних граматик можна побудувати рівносильні їм однозначні граматики.

4.8. ПРИКЛАД. Розглянемо граматику і мову з попереднього прикладу. Граматика G неоднозначна, бо **else** можна асоціювати з двома різними **then**. З тієї ж причини мови програмування, в яких дозволяються як оператори виду **if-then**, так і оператори виду **if-then-else**, можуть бути неоднозначними. Невизначеність можна виправити, якщо домовитися, що **else** повинно асоціюватися з останнім з записаних перед ним **then**.

Можна поправити граматику з попереднього прикладу, ввівши два нетермінали S_1 і S_2 і вимагаючи, щоб S_2 породжував оператори виду **if-then-else**, тоді як S_1 може породжувати оператори обох видів. Продукції нової граматики є такими:

$$S_1 \rightarrow \text{if } b \text{ then } S_1 \mid \text{if } b \text{ then } S_2 \text{ else } S_1 \mid a$$

$$S_2 \rightarrow \text{if } b \text{ then } S_2 \text{ else } S_2 \mid a.$$

Той факт, що перед **else** знаходиться тільки S_2 , гарантує появу всередині конструкції **then-else** або букви a , або іншого **else**. Таким чином, структура **if** b **then** (**if** b **then** a) **else** a не виникає.

Рекомендована література : [1, с. 92–150], [2, с. 105–123, 163–192], [5, с. 341–351, 354–359], [17, с. 278–285], [24, с. 14–24].

Питання та вправи до параграфа 4.

- 4.1.** Дайте означення граматики.
- 4.2.** Які типи граматик ви знаєте?
- 4.3.** Як будується дерево виведення для слова ω ?
- 4.4.** Чи може для деякого слова ω існувати декілька дерев виведення?

4.5. Нехай $N = \{S, A, B\}$, $T = \{a, b\}$. Визначити мову, породжену граматикою $G = (N, T, S, P)$ з вказаною множиною продукцій P :

- а) $P = \{S \rightarrow AB \mid aA, A \rightarrow a, B \rightarrow ba\};$
- б) $P = \{S \rightarrow AB \mid AA, A \rightarrow aB \mid ab, B \rightarrow b\};$
- в) $P = \{S \rightarrow AA \mid B, A \rightarrow aaA \mid aa, B \rightarrow bB \mid b\};$
- г) $P = \{S \rightarrow aS \mid Sb \mid a\};$
- д) $P = \{S \rightarrow SS \mid a \mid b\};$
- е) $P = \{S \rightarrow SS \mid aSb \mid e\};$
- є) $P = \{S \rightarrow e \mid aS \mid aSbS\};$
- ж) $P = \{e \mid aSbS \mid bSaS\}.$

4.6. Знайти мову, породжену граматикою $G = (N, T, S, P)$, де $N = \{S, A, Q, R\}$, $T = \{a\}$, а множина продукцій $P = \{S \rightarrow QAQ, QA \rightarrow QR, RA \rightarrow AAR, RQ \rightarrow AAQ, A \rightarrow a, Q \rightarrow e\}$.

4.7. Нехай задана граматики $G = (N, T, S, P)$, де $N = \{S, A\}$, $T = \{0, 1\}$, множина продукцій

$$P = \{S \rightarrow 0S \mid 1A \mid 1 \mid e, A \rightarrow 1A \mid 1\}.$$

- а) Побудувати вивід для слова $0^3 1^6$;
- б) Показати, що дана граматики породжує мову $\{0^m 1^n \mid m, n = 0, 1, 2, \dots\}$.

4.8. Побудувати граматики, які породжують вказані мови:

- а) $\{01^{2n} \mid n = 0, 1, \dots\};$
- б) $\{0^{3n} 1^n \mid n = 0, 1, \dots\};$
- в) $\{0^n 1^m 0^n \mid m, n = 0, 1, \dots\}.$

4.9. З'ясувати, до яких типів за класифікацією Хомського належить граматики G з множиною продукцій P :

- а) $P = \{S \rightarrow SAB, SA \rightarrow a, B \rightarrow b\};$
- б) $P = \{S \rightarrow ABS, AB \rightarrow ab, S \rightarrow c\};$
- в) $P = \{S \rightarrow aAB, A \rightarrow Bb, B \rightarrow e\};$
- г) $P = \{S \rightarrow aA, A \rightarrow bB, B \rightarrow b \mid e\};$
- д) $P = \{S \rightarrow A, A \rightarrow B, B \rightarrow e\}.$

4.10. Побудувати ліволінійну граматики для формальної мови всіх слів довжини менше шести в алфавіті $\{0, 1, m, n\}$, які починаються з букв m або n .

4.11. Побудувати праволінійну граматику, яка породжує множину цілих чисел.

4.12. Довести, що жодне слово мови $L(G)$ не містить підслова ba , де контекстно-вільна граMATика G задана продукціями

$$S \rightarrow aS \mid bA \mid a, \quad A \rightarrow bA \mid b.$$

4.13. Показати, що кожне слово мови $L(G)$ містить більше букв a , ніж букв b , де контекстно-вільна граMATика G задана продукціями

$$S \rightarrow Sa \mid bSS \mid SSb \mid SbS \mid a.$$

4.14. Показати, що мова

$$\{\omega \in \{0, 1\}^* \mid \omega \text{ містить однакову кількість букв } 0 \text{ та } 1\}$$

не породжується граMATикою з продукціями

$$S \rightarrow 0S1 \mid 01S \mid 1S0 \mid 10S \mid S01 \mid S10 \mid e.$$

4.15. Довести, що мова

$$\{\omega \in \{0, 1\}^* \mid \omega \text{ містить однакову кількість букв } 0 \text{ та } 1\}$$

породжується граMATикою з продукціями

$$S \rightarrow SS \mid 0S1 \mid 1S0 \mid e.$$

4.16. Яку формальну мову породжує граMATика з продукціями

$$S \rightarrow aSBa \mid aba, \quad aB \rightarrow Ba, \quad bB \rightarrow bb \quad ?$$

4.17. Нехай G – граMATика з $N = \{S\}$, $T = \{a, b, c\}$, початковий символ S і множина продукцій

$$P = \{S \rightarrow abS \mid bcS \mid bbS \mid a \mid cb\}.$$

Побудувати дерево виведення для слів:

а) bbbcbba

б) bcabbbbbbcb.

4.18. Показати, що граMATика

$$G = \{\{S\}, \{+, x\}, S, \{S \rightarrow S + S \mid x\}\}$$

є неоднозначною. Побудувати всі дерева виведення для слів $x + x + x$ та $x + x + x + x$.

Розділ II

Скінченні автомати

§ 1. Автомати. Їх типи та задання

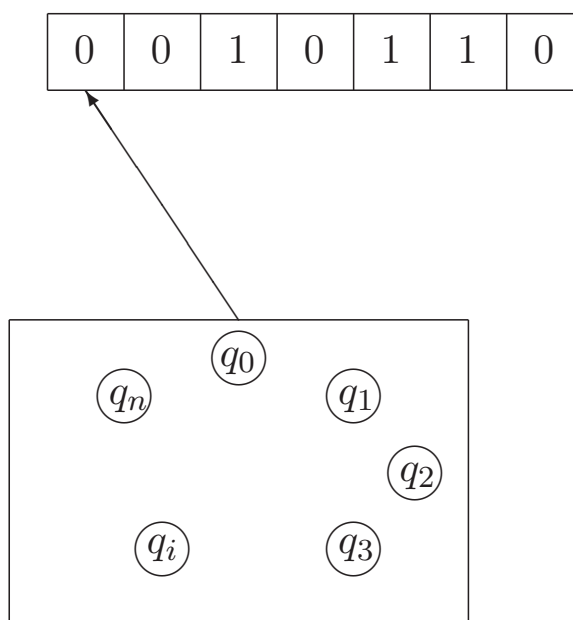
Існує багато різноманітних методів перетворення інформації. Алгоритм роботи перетворювачів інформації можна інтуїтивно визначити як деяку сукупність правил, за якими даний пристрій перетворює вхідну інформацію у вихідну. В даному параграфі ми розглянемо перетворювач інформації, який називається автоматом.

Щоб перейти до формальних методів опису процесу перетворення інформації з допомогою автоматів, розглянемо деякий пристрій, який має n входів і m виходів. Будемо вважати, що на входи інформація подається в дискретні моменти часу t_0, t_1, \dots, t_n . Інтервал часу $t_{i+1} - t_i$ називається *тактом*. Аналогічно і з виходів інформація одержується дискретно в часі. На кожному із входів (виходів) сигнал може приймати дискретні значення з деякого набору u_0, u_1, \dots, u_q . Сукупність значень вхідних сигналів у момент часу t_i називатимемо *вхідною буквою*, а послідовність вхідних букв – *вхідним словом*. Множину всіх вхідних букв назвемо *вхідним алфавітом*. Аналогічні означення відносяться і до вихідних сигналів.

Давши це означення, можна уявляти будь-який перетворювач інформації як пристрій з одним входом, на який надходять слова, складені з букв вхідного алфавіту, і з одним виходом, з якого отримуються слова в вихідному алфавіті. Процес перетворення інформації в такому пристрої зводиться до встановлення деякої відповідності між словами вхідного та вихідного алфавітів.

Автоматом називають перетворювач алфавітної інформації, який має один вхід і один вихід. Для задання умов функціонування автомата фіксуються три множини, елементами яких є букви трьох алфавітів: вхідного $X = \{x_1, \dots, x_m\}$, вихідного $Y = \{y_1, \dots, y_l\}$ і множини внутрішніх станів автомата $Q = \{q_0, \dots, q_n\}$.

Автомат складається з трьох частин: *стрічки, головки і керуючого пристрою*. Стрічка використовується для запису вхідних даних (вхідного слова). Вона поділена на скінченну кількість комірок.



Головка переглядає стрічку, читає букву з неї і передає одержану інформацію керуючому пристрою. Кожного разу головка переглядає тільки одну комірку стрічки, читає записану в ній букву і переходить до наступної комірки праворуч. Керуючий пристрій складається з елементів множини станів Q .

Автомат функціонує в дискретні моменти часу, які прийнято позначати цілими невід'ємними числами $t = 0, 1, \dots, k$. В початковий момент часу $t = 0$ керуючий пристрій завжди знаходиться в початковому стані $q_0 = q(0)$, в якому автомат починає роботу над вхідним словом. Потім за активним станом і зчитаною головою буквою він визначає, в який стан потрібно перейти, і яку букву треба написати на місці прочитаної вхідної букви. На згаданих вище множинах задаються дві функції. *Функція переходів* δ , яка визначає стан автомата $q(t+1) \in Q$ в момент часу $t+1$ в залежності від його стану $q(t) \in Q$ і значення вхідного сигналу $x(t) \in X$ в момент часу t : $q(t+1) = \delta(q(t), x(t))$. Тобто зміна станів визначається за допомогою функції двох змінних $\delta : Q \times X \rightarrow Q$. *Функція виходів* $f : Q \times X \rightarrow Y$ визначає значення вихідного сигналу $y(t) \in Y$ в залежності від стану автомата $q(t) \in Q$ і значення вхідного сигналу $x(t) \in X$ в момент часу t . Таким чином, якщо керуючий пристрій

перебуває в стані $q \in Q$ і головка зчитує з стрічки букву $a \in X$, то керуючий пристрій переходить в новий стан $p = \delta(q, a)$ і на місці букви a записується буква $b = f(q, a)$.

Коли на вхід автомата подавати слово (послідовність вхідних букв), на виході також з'являється слово (послідовність вихідних букв). У цьому випадку автомат A можна розглядати як алфавітний перетворювач інформації, який перетворює слова вільної напівгрупи $F(X)$ в слова вільної напівгрупи $F(Y)$. Визначена таким чином словесна функція $g_A : F(X) \rightarrow F(Y)$ називається *функцією, визначеною автоматом A* , або *автоматною A -функцією*.

Для повного опису автомата потрібно виділити множину $F \subset Q$ *кінцевих станів*, в яких він підтверджує, що вхідне слово належить шуканій мові. Таким чином, ми приходимо до такого визначення автомата.

Сімка $A = (Q, X, Y, \delta, f, q_0, F)$ називається *автоматом Мілі*, якщо вона складається із множини станів автомата Q , множини вхідних букв X , множини вихідних букв Y , функції переходів $\delta : Q \times X \rightarrow Q$, функції виходів $f : Q \times X \rightarrow Y$, початкового стану q_0 та множини кінцевих станів F . Множини X і Y називають відповідно *вхідним і вихідним алфавітами автомата*. Якщо $\delta(a, x) = a'$, то кажуть, що автомат Q під дією вхідного сигналу $x \in X$ переходить в стан a' , або сигнал x переводить автомат Q із стану a в стан a' . Якщо $f(a, x) = y$, то кажуть, що автомат Q перетворює в стані a вхідний сигнал $x \in X$ у вихідний сигнал $y \in Y$.

Як саме працює автомат на вхідному слові? На початку роботи керуючий пристрій перебуває в початковому стані q_0 , на стрічці записане вхідне слово і головка зчитує букву з першої зліва комірки. Потім автомат працює крок за кроком відповідно до функцій переходів δ та виходів f . Він зупиняє роботу тоді, коли головка прочитає букву, записану в останній справа комірці стрічки. Коли обробка автоматом вхідного слова завершилася, то кажуть, що *автомат розпізнає вхідне слово*, якщо він зупиняється в одному з кінцевих станів множини F . В протилежному випадку, кажуть, що *вхідне слово не розпізнається автоматом*. Якщо ж множина F кінцевих станів не задана, то вважається, що автомат розпізнає всі слова вхідного алфавіту. Формальна мова всіх слів вхідного алфавіту X , які розпізнаються автоматом A , позначається $L(A)$. В цьому випадку кажуть, що мова $L(A)$ *розпізнається автоматом*.

А. Множина всіх вихідних слів автомата A , які є результатом роботи автомата на мові $L(A)$, називається *мовою, що породжується автоматом A* .

Автомат $A = (Q, X, Y, \delta, f, q_0, F)$ називається *скінченним*, якщо всі три множини – Q , X і Y – скінченні, і *нескінченним*, якщо хоч одна з них нескінченна.

Автомат називається *повним* або *повністю визначеним*, якщо функції переходів і виходів всюди визначені, і *частковим*, якщо хоч одна з них є частково визначеною функцією.

Іноді трапляються автомати, в яких компонент δ – не функція, а деяке відношення, тобто в таких автоматах не виконується умова однозначності переходу. Автомати такого типу називаються *недетермінованими*. Якщо ж відношення δ є функцією, то автомат називається *детермінованим*. Отже, для детермінованого автомата Q з початковим станом a однозначно знаходиться стан b , в який автомат перейде під дією слова $\omega \in F(X)$. А в недетермінованому автоматі таких станів може бути не один, а декілька. Ясно, що клас детермінованих автоматів є підкласом класу недетермінованих автоматів.

Крім класів детермінованих і недетермінованих автоматів існують і інші спеціальні класи. Серед них виділяються деякі вироджені класи автоматів, в яких одна з множин (Q , X або Y) одноелементна. У таких випадках розглядаються спрощені моделі автоматів. Наведемо деякі приклади.

Автомат без пам'яті – це трійка (X, Y, f) , де $f : X \rightarrow Y$. Цей автомат виконує побуквену трансформацію букв вхідного алфавіту X у букви вихідного алфавіту Y , при якій на одну і ту ж вхідну букву $x \in X$ він реагує одною і тою ж вихідною буквою $y \in Y$. Поняття *автомат без пам'яті* – основне в теорії перемикальних схем, яка вивчає способи подання таких автоматів мережами із логічних елементів.

Автономний автомат – це четвірка (Q, Y, δ, f) , де $\delta : Q \rightarrow Q$, $f : Q \rightarrow Y$. Для такого автомата поданням початкового стану визначається весь подальший процес його функціонування.

Автомат без виходів – це п'ятірка (Q, X, δ, q_0, F) , де F – множина кінцевих станів автомата, а $q_0 \in Q$ – початковий стан автомата. Такі автомати будуть детальніше розглянуті в наступних параграфах.

Із перелічених класів автоматів тільки автомати без виходів, з точки зору теорії, викликають інтерес, бо в двох попередніх випадках поведінка автомата є наперед визначеною.

Автомати Мура є окремим випадком автоматів Мілі. Автомат $A = (Q, X, Y, \delta, f, q_0, F)$ називається *автоматом Мура*, якщо $f(q, x_1) = f(q, x_2)$ для кожних $q \in Q$ та $x_1, x_2 \in X$. В автоматі Мура вихідний сигнал у момент часу t однозначно визначається станом автомата в той же момент часу, тобто $y(t) = h(q(t))$, де $h : Q \rightarrow Y$. Функція h називається *функцією відміток автомата*, а її значення $h(a)$ на стані a – *відміткою цього стану*.

Хоча автомати Мура і є окремим випадком автоматів Мілі, в теорії автоматів вони заслуговують на особливу увагу, оскільки в ряді випадків їх специфічні властивості дають можливість будувати більш змістовну і глибоку теорію, ніж теорія автоматів Мілі.

Перейдемо до розгляду скінченних автоматів і способів їх задання.

У загальному випадку для задання скінченних автоматів користуються двома стандартними способами, які називаються *універсальними* – це таблиці переходів і виходів та графи переходів і виходів.

Таблиці переходів і виходів автомата – це дві матриці однакової розмірності, стовпчики яких відмічені різними буквами вхідного алфавіту, а рядки – різними символами станів автомата. Для функції переходів (перша матриця) на перетині i -го рядка, відміченого станом $a \in Q$, і j -го стовпчика, відміченого буквою $x \in X$, знаходиться значення $\delta(a, x) = b$. Аналогічно для функції виходів (друга матриця) на перетині i -го рядка і j -го стовпчика знаходиться значення $f(a, x) = y \in Y$. Вважають, що початковий стан автомата відмічає перший рядок як в таблиці переходів, так і в таблиці виходів.

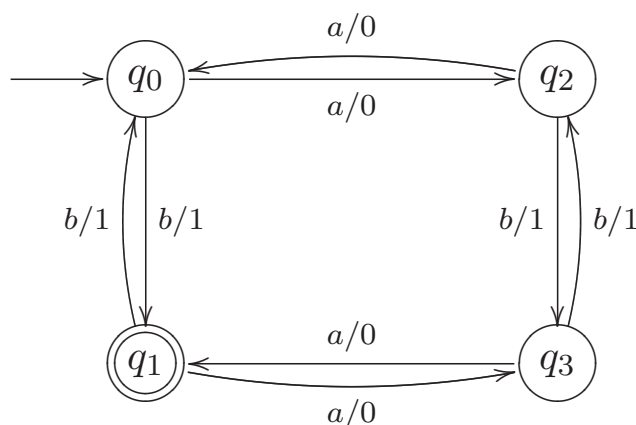
1.1. ПРИКЛАД. Автомат $A = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \{0, 1\}, \delta, f, q_0, \{q_1\})$ задається такими таблицями переходів і виходів:

δ	a	b	f	a	b
q_0	q_2	q_1	q_0	0	1
q_1	q_3	q_0	q_1	0	1
q_2	q_0	q_3	q_2	0	1
q_3	q_1	q_2	q_3	0	1

В якому б стані даний автомат не перебував, він замінює букви a і b на 0 і 1 відповідно. Таким чином, скінченний автомат A переписує будь-яке слово в алфавіті $\{a, b\}$ в слово в алфавіті $\{0, 1\}$. Автомат A не є автоматом Мура, оскільки перебуваючи в стані q_0 він замінює букву a на 0, а букву b – на 1.

Граф переходів і виходів скінченного автомата A є альтернативним шляхом для зображення A . Він являє собою помічений орієнтований граф, вершини якого взаємно однозначно відповідають станам автомата. Стрілки графа відмічені парами букв: перша – буква вхідного алфавіту, друга – буква вихідного алфавіту. У даному випадку відповідність між графом переходів і виходів та функціями переходів δ і виходів f така, що коли $\delta(a, x) = a'$ і $f(a, x) = y$, то в графі з вершини a у вершину a' веде стрілка, відмічена парою (x/y) , і навпаки, коли в графі переходів і виходів автомата існує стрілка (a, a') , відмічена парою (x/y) , то для функцій переходу δ і виходу f виконуються рівності $\delta(a, x) = a'$ і $f(a, x) = y$. Крім того, щоб підкреслити початковий стан q_0 , проводиться стрілка без початкової вершини, кінцевою вершиною якої є стан q_0 . Кінцевий стан позначається двома концентричними колами.

1.2. ПРИКЛАД. Граф переходів і виходів для автомата з прикладу 1.1 має вигляд:



З наведеної відповідності між графом автомата та його функціями переходів і виходів випливає, що не кожен граф, стрілки якого відмічені парами символів із деяких алфавітів X і Y , буде графом переходів і виходів детермінованого автомата. Для того, щоб відмічений граф був графом детермінованого автомата, необхідно і достатньо, щоб виконувались дві умови, які називаються умовами *автоматності графа*:

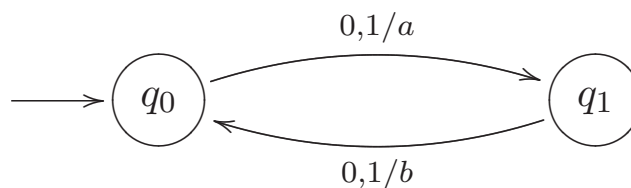
1) не існує двох стрілок з однаковими вхідними відмітками, що виходять з однієї і тієї ж вершини (умова однозначності);

2) для будь-якої вершини і вхідного символу існує стрілка, що виходить із цієї вершини і відмічена цим вхідним символом (умова повноти).

Отже, для недетермінованих автоматів не виконується умова 1), а для часткових – умова 2). Незважаючи на це, як недетерміновані, так і часткові автомати теж зображують графами.

1.3. ПРИКЛАД. Побудувати детермінований скінченний автомат Мура, який всі букви бінарного слова $\omega \in \{0, 1\}^*$, що стоять на непарних позиціях, замінює на букву a , а на парних – на букву b .

Граф переходів і виходів шуканого автомата A має вигляд:



Перебуваючи в стані q_0 , даний автомат замінює кожну букву вхідного слова на букву a , а в стані q_1 – на букву b .

Таким чином, шуканий автомат A задається як шістка

$$A = (Q, X, Y, \delta, f, q_0),$$

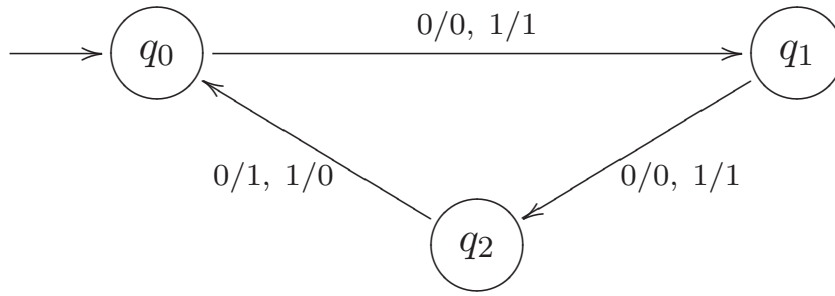
де $Q = \{q_0, q_1\}$, $X = \{0, 1\}$, $Y = \{a, b\}$, а функції переходів δ і виходів f задані таблицями:

δ	0	1
q_0	q_1	q_1
q_1	q_0	q_0

f	0	1
q_0	a	a
q_1	b	b

1.4. ПРИКЛАД. Нехай $X = Y = \{0, 1\}$. Побудуємо детермінований скінченний автомат, який „шифрує” всі бітові рядки з 0 та 1, замінюючи всі біти, які стоять на позиціях кратних 3, на протилежні.

Граф переходів і виходів шуканого автомата A має вигляд:



Зауважимо, що якщо „зашифровану” послідовність (вихідне слово) з 0 та 1 подати на вхід цього автомата, то одержимо початкову послідовність (вхідне слово), тобто цей автомат одночасно буде й „дешифруючим пристроєм”.

Функції переходів і виходів можна продовжити на вхідну і вихідну напівгрупи $F(X)$ і $F(Y)$. Для цього необхідно покласти

$$\begin{aligned}\delta(q, e) &= q, \delta(q, a\omega) = \delta(\delta(q, a), \omega), \\ f(q, e) &= e, f(q, a\omega) = f(q, a)f(\delta(q, a), \omega).\end{aligned}$$

Використовуючи індукцію за довжиною слова ω , неважко довести таке співвідношення:

$$\begin{aligned}\delta(q, a\omega) &= \delta(\delta(q, a), \omega), \\ f(q, a\omega) &= f(q, a)f(\delta(q, a), \omega).\end{aligned}$$

Дійсно, якщо $\omega = x$, тобто $l(\omega) = 1$, то база індукції має місце за означенням.

Якщо ж $\omega = \nu x$ і для ν співвідношення асоціативності виконується, то

$$\delta(q, a\omega) = \delta(\delta(q, a\nu), x) = \delta(\delta(q, a), \nu x),$$

$$f(q, a\nu x) = f(q, a\omega)f(\delta(q, a\nu), x) =$$

$$= f(q, a)f(\delta(q, a), \alpha)f(\delta(q, a\nu), x) = f(q, a)f(\delta(q, a), \nu x).$$

Якщо $\delta(p, \omega) = q$, то кажемо, що *автомат A переходить із стану p в стан q під дією слова ω* , або, що *слово ω переводить автомат A із стану p в стан q* .

Стан q називається *досяжним* із стану p ($p \rightarrow q$), коли існує таке вхідне слово ω , яке переводить автомат A із стану p в стан q . Для того, щоб встановити досяжність q з p , достатньо в графі переходів і виходів знайти шлях із вершини p у вершину q . Очевидно, що коли такий шлях існує, то послідовність перших компонентів відміток стрілок цього шляху і буде складати слово ω .

Зауважимо, що для встановлення досяжності q з p достатньо розглядати лише такі шляхи в графі автомата (вони називаються *простими*), які не проходять двічі через одну і ту ж вершину графа. Це зауваження дає можливість обмежити пошук необхідного вхідного слова лише такими словами, довжина яких не перевищує загальної кількості станів автомата. Ясно, що у випадку скінченних автоматів перевірка істинності відношення $p \rightarrow q$ завжди можлива.

Рекомендована література : [4, с. 494–509, 552–564], [5, с. 385–398], [12, с. 286–309], [13, с. 410–426], [17, с. 285–289].

Питання та вправи до параграфа 1.

- 1.1. Дайте означення скінченного автомата.
- 1.2. Які типи автоматів вам відомі?
- 1.3. В чому полягає різниця між автоматом Мура та автоматом Мілі?
- 1.4. Які способи задання автоматів ви знаєте?
- 1.5. Коли ми кажемо, що стан q скінченного автомата A є досяжним із стану p ?
- 1.6. Побудувати граф скінченного автомата $A = (Q, X, Y, \delta, f, q_0)$, де $Q = \{q_0, q_1, q_2\}$, $X = \{0, 1, 2\}$, $Y = \{a, b, c\}$, а функції переходів δ і виходів f задані таблицями:

δ	0	1	2
q_0	q_1	q_1	q_2
q_1	q_1	q_2	q_0
q_2	q_1	q_0	q_2

f	0	1	2
q_0	a	b	a
q_1	b	c	b
q_2	c	a	b

Визначте, в яке слово перетворює автомат A кожне з наступних вхідних слів:

а) 012;

в) 100000;

б) 001022;

г) 000111222.

1.7. Побудувати граф скінченного автомата $A = (Q, X, Y, \delta, f, q_0)$, де $Q = \{q_0, q_1\}$, $X = \{00, 01, 10, 11\}$, $Y = \{0, 1\}$, а функції переходів δ і виходів f задані таблицями:

δ	00	01	10	11
q_0	q_0	q_0	q_0	q_1
q_1	q_0	q_1	q_1	q_1

f	00	01	10	11
q_0	0	1	1	0
q_1	1	0	0	1

В чому полягає робота даного автомата?

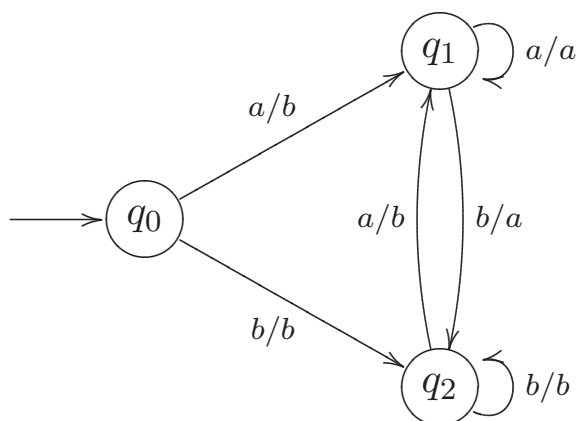
1.8. В чому полягає робота автомата $A = (Q, X, Y, \delta, f, q_0)$, де $Q = \{q_0, q_1\}$, $X = \{0, 1, *\}$, $Y = \{0, 1, +, -\}$, а функції переходів δ і виходів f задані таблицями:

δ	0	1	*
q_0	q_0	q_1	q_0
q_1	q_1	q_0	q_0

f	0	1	*
q_0	0	1	+
q_1	0	1	-

?

1.9. Скінченний автомат A заданий графом:



Задати його як шістку $A = (Q, X, Y, \delta, f, q_0)$ і з'ясувати у чому полягає робота автомата.

1.10. Нехай $X = \{a, b, c\}$, $Y = \{*, +, \diamond\}$. Побудувати детермінований скінченний автомат, який реалізує алгоритм шифрування вхідного слова шифром простої заміни, міняючи букву a на $*$, b на $+$ і c на \diamond .

1.11. Нехай $X = Y = \{a, b, c, d\}$. Побудувати скінченний автомат, який шифрує букви вхідного слова, які знаходяться на парних позиціях, шифром зсуву на дві букви праворуч, а на непарних – на три букви ліворуч.

1.12. Нехай $X = Y = \{0, 1, 2\}$. Побудувати скінченний автомат, який шифрує вхідне слово, замінюючи цифру x на позиції $3n + k \geq 1$, де $n \geq 0$, $k \in \{0, 1, 2\}$, остачею від ділення $x + k$ на 3.

1.13. Побудувати „дешифруючі автомати” для автоматів з двох попередніх прикладів.

1.14. Побудувати скінченний автомат з виходом, у якого вхідний алфавіт $X = \{0, 1, a\}$ і вихідний $Y = \{0, 1, p, n\}$. Вихідна послідовність з 0 та 1 збігається з вхідною, а на кожний символ запиту a друкується p , якщо кількість 0 від початку роботи парна, і n – якщо непарна.

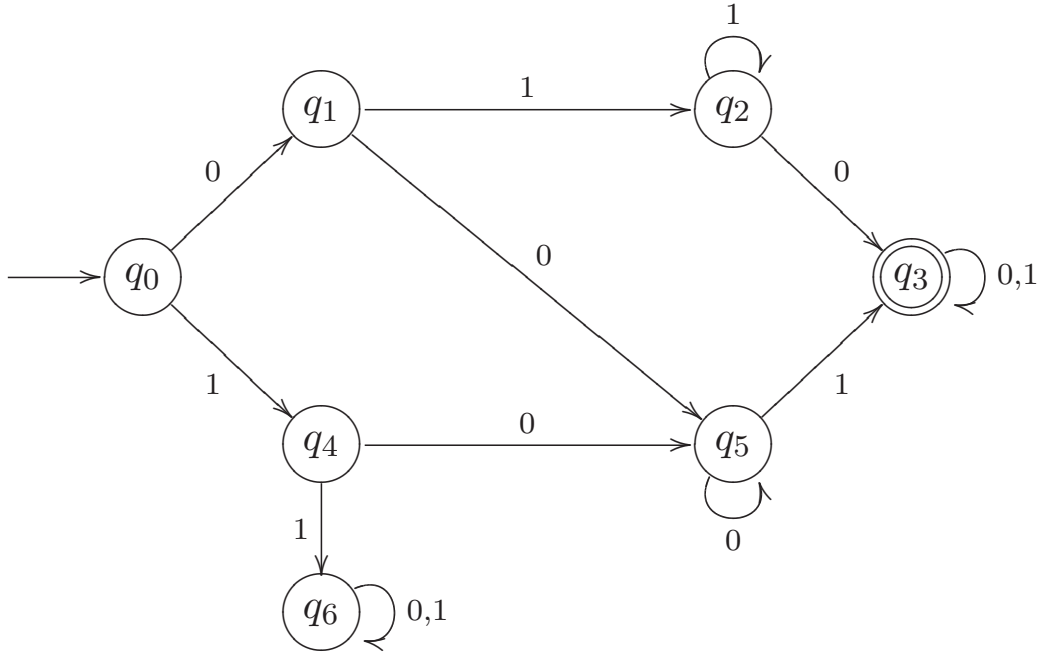
1.15. Нехай $X = Y = \{a, b\}$. Побудувати скінченний автомат, який дає на виході слово, що складається тільки з букв a , тоді і лише тоді, коли вхідне слово містить префікс ab .

1.16. Нехай $X = Y = \{a, b\}$. Побудувати скінченний автомат, який дає на виході слово, що закінчується буквою b , тоді і лише тоді, коли вхідне слово не містить суфікса bbb .

§ 2. Детерміновані скінченні автомати без виходу

В цьому параграфі ми на прикладах розглянемо методи побудови детермінованих скінченних автоматів (ДСА) без виходу. Зокрема покажемо, що клас формальних мов, які розпізнаються детермінованими скінченними автоматами, є замкненим відносно об'єднання, перетину, доповнення, різниці та симетричної різниці.

2.1. ПРИКЛАД. Знайти формальну мову, що розпізнається ДСА $A = (Q, X, \delta, q_0, F)$, який заданий поміченим графом:



Нагадаємо, що слово $\omega \in L(A)$ тоді і тільки тоді, коли автомат A зупиняє аналіз ω в одному з кінцевих станів множини F . Якщо при аналізі слова ω автомат переходить в стан q_6 , то дане слово не розпізнається A , і навпаки, перейшовши в кінцевий стан q_3 автомат розпізнає слово ω . Отже, нам потрібно проаналізувати, як зі стану q_0 можна перейти в стан q_3 . Всього є три типи шляхів з q_0 в q_3 :

$$(q_0, q_1, q_2, \dots, q_2, q_3); \quad (q_0, q_1, q_5, \dots, q_5, q_3); \quad (q_0, q_4, q_5, \dots, q_5, q_3).$$

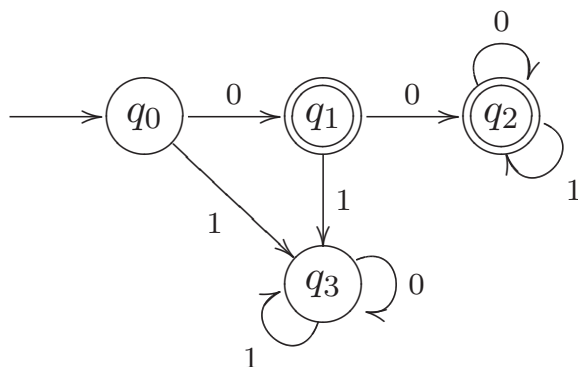
Вони відповідають словам, префікси яких належать регулярним мовам 011^*0 , 000^*1 і 100^*1 відповідно. Таким чином,

$$L(A) = (011^*0 + 000^*1 + 100^*1)(0 + 1)^*.$$

2.2. ПРИКЛАД. Побудувати граф ДСА $A = (Q, X, \delta, q_0, F)$ і знайти мову, яка розпізнається A , якщо $Q = \{q_0, q_1, q_2, q_3\}$, $X = \{0, 1\}$, $F = \{q_1, q_2\}$, а функція переходів δ задана таблицею:

δ	0	1
q_0	q_1	q_3
q_1	q_2	q_3
q_2	q_2	q_2
q_3	q_3	q_3

Помічений граф автомата A має вигляд:

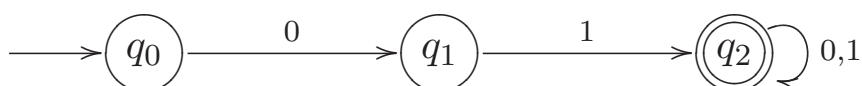


Знайдемо мову $L(A)$, яка розпізнається автоматом A .

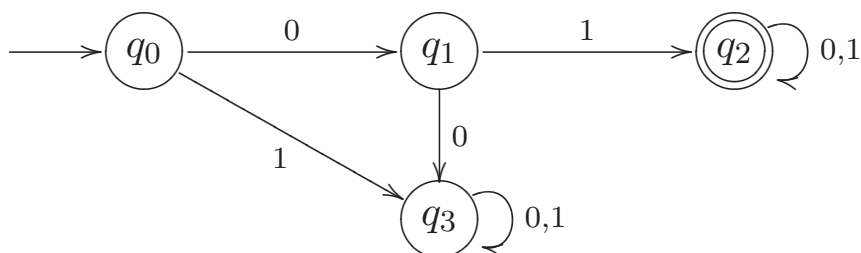
З графа ДСА видно: перейшовши в стан q_3 , автомат залишається в ньому до кінця роботи і $\omega \notin L(A)$. З іншого боку, перейшовши в кінцевий стан q_2 , автомат ніколи не покине даного стану, і такі слова розпізнаються ДСА. З вищесказаного випливає, що мова $L(A)$ містить слово 0 (аналіз якого зупиняється в стані q_1) і всі слова, які мають префікс 00 . Таким чином, $L(A)$ записується регулярним виразом $0 + 00(0 + 1)^*$.

2.3. ПРИКЛАД. Побудувати ДСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, які мають префікс 01 .

Легко бачити, що регулярний вираз шуканої формальної мови має вигляд $01(0 + 1)^*$. Побудуємо граф цього регулярного виразу:



Даний граф не є графом ДСА, оскільки в графі ДСА з кожної вершини q має виходити єдина стрілка з позначкою a для кожної пари $(q, a) \in Q \times X$. Щоб виконати цю умову, побудуємо додаткову тупикову вершину q_3 і стрілки $q_0 \rightarrow q_3$ і $q_1 \rightarrow q_3$ з позначками 1 і 0 відповідно. Шуканий граф ДСА має вигляд:



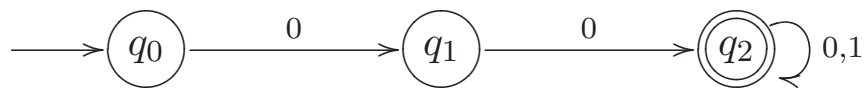
2.4. ПРИКЛАД. Побудувати ДСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, які містять підслово 00 .

Спершу зауважимо, що з регулярного виразу $(0 + 1)^*00(0 + 1)^*$ не можна визначити перше входження пари 00 у вхідному слові.

З другого боку, шуканий ДСА має виявляти 00 вже при першому його входженні. Проаналізуємо дану проблему детальніше.

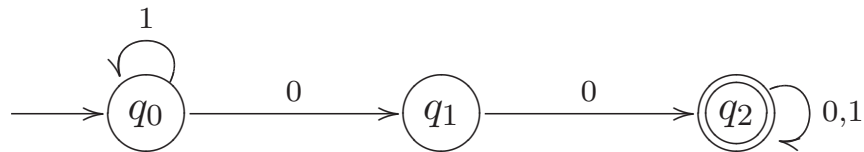
Припустимо, що на стрічці записане слово $\omega = x_1x_2\dots x_n$, де $x_i \in \{0,1\}$. Ми маємо перевіряти кожне підслово $x_1x_2, x_2x_3, \dots, x_{n-1}x_n$, і як тільки виявиться, що деяка пара рівна 00, робити висновок, що слово w розпізнається ДСА. Звідси випливає наступний алгоритм побудови графа автомата:

Крок 1. Будуємо граф

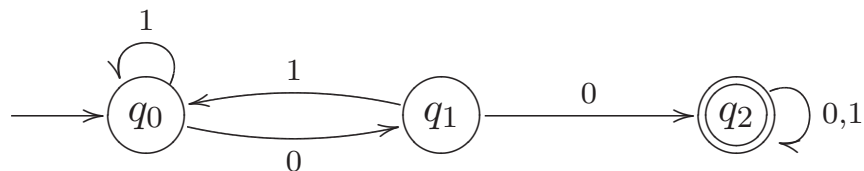


з кінцевою вершиною q_2 , з допомогою якої розпізнаються слова, що містять 00. Зокрема, якщо $x_1x_2 = 00$, то слово ω розпізнається.

Крок 2. Якщо $x_1 = 1$, то залишаємо підслово x_1x_2 і переходимо до перевірки x_2x_3 . Таким чином, автомату потрібно перейти в стан q_0 , тобто $\delta(q_0, 1) = q_0$.

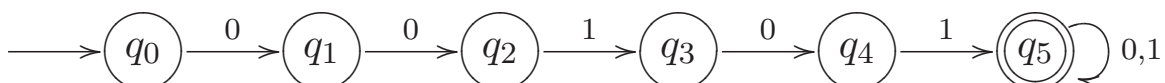


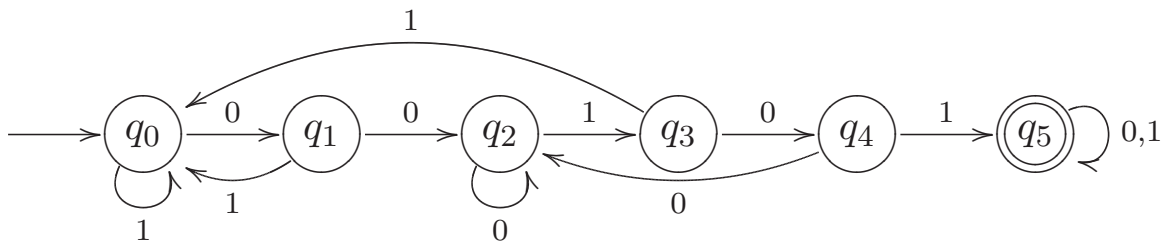
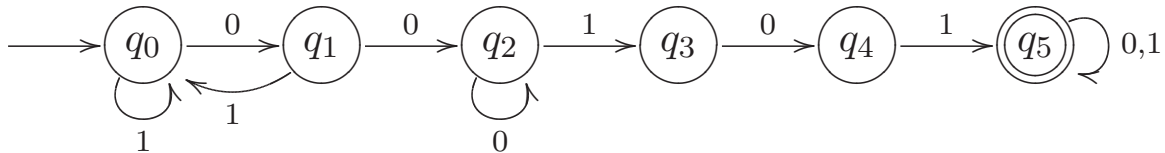
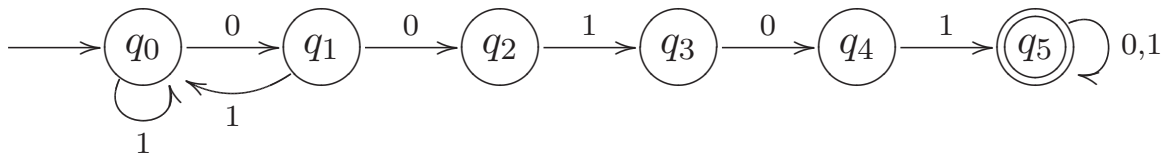
Крок 3. Якщо $x_1 = 0$ і $x_2 = 1$, то ні підслово x_1x_2 , ні x_2x_3 не дорівнює 00. Отже, керуючий пристрій ДСА має перейти в стан q_0 для аналізу підслова x_3x_4 . Таким чином, покладемо $\delta(q_1, 1) = q_0$. Граф шуканого ДСА має вигляд:



2.5. ПРИКЛАД. Побудувати ДСА, який розпізнає всі слова в бінарному алфавіті $X = \{0,1\}$, які містять підслово 00101.

Аналогічно, як і в попередньому прикладі, крок за кроком будемо граф ДСА:



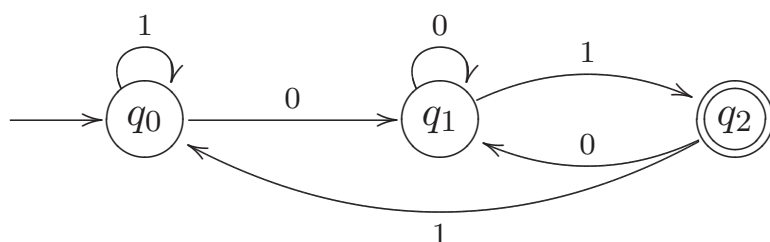


2.6. ПРИКЛАД. Побудувати ДСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, які мають суфікс 01.

Спершу побудуємо наступний граф:



Стани q_0 та q_1 вказують, що „не знайдено префікса слова 01” та „знайдений префікс 0 слова 01” відповідно. Аналогічно, як і в попередньому прикладі, покладемо $\delta(q_0, 1) = q_0$ і $\delta(q_1, 0) = q_1$. Якщо перейшовши в стан q_2 автомат не закінчив аналіз вхідного слова, то необхідно покласти $\delta(q_2, 0) = q_1$ і $\delta(q_2, 1) = q_0$. Таким чином, шуканий граф ДСА має вигляд:



2.7. ПРИКЛАД. Побудувати ДСА, який розпізнає всі двійкові натуральні числа, які конгруентні нулю за модулем 5.

Ідея побудови даного ДСА подібна до ідеї побудови автоматів з попередніх двох прикладів. Побудуємо п'ять станів q_0, q_1, \dots, q_4 і вважаємо, що кожен стан q_i має значення „префікс α вхідного слова має властивість $\alpha \equiv i(mod\ 5)$ ”. Тобто потрібно визначити $\delta(q_0, x_1x_2 \dots x_k) = q_i$, якщо $x_1x_2 \dots x_k \equiv i(mod\ 5)$.

Як же побудувати стрілки між станами автомата, використовуючи цю ідею? Нагадаємо, що для функції переходів δ виконується наступна рівність:

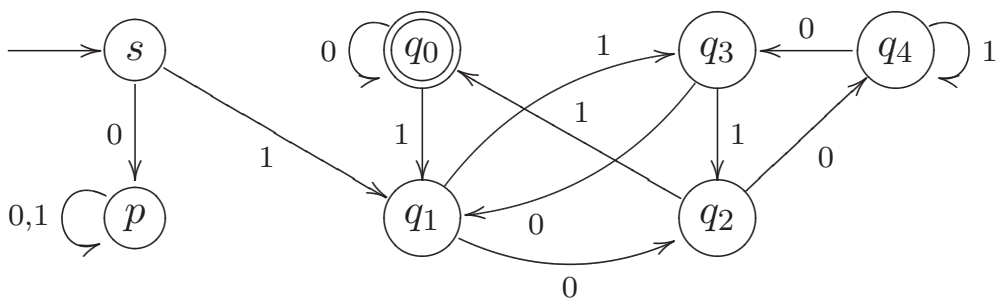
$$\delta(\delta(q_0, \omega), a) = \delta(q_0, \omega a)$$

для будь-якого бінарного слова ω і довільного $a \in \{0, 1\}$. Припустимо, що $\delta(q_0, \omega) = q_i$ і $\delta(q_0, \omega a) = q_j$. Тоді мають виконуватися конгруенції: $\omega \equiv i(mod\ 5)$ і $\omega a \equiv j(mod\ 5)$. Таким чином,

$$j \equiv \omega a(mod\ 5) \equiv 2 \cdot \omega + a(mod\ 5) \equiv 2 \cdot i + a(mod\ 5).$$

Отже, покладемо $\delta(q_i, a) = q_j$, якщо $j \equiv 2 \cdot i + a(mod\ 5)$. Наприклад, $\delta(q_2, 0) = q_4$ і $\delta(q_2, 1) = q_0$. Крім того, стан q_0 є єдиним кінцевим станом, оскільки $\delta(q_0, \omega) = q_0$ означає, що $\omega \equiv 0(mod\ 5)$.

Насамкінець зауважимо, що двійковий запис натурального числа завжди починається з 1. Таким чином, потрібно додати новий початковий стан s і тупиковий стан p як показано на графі:



2.8. ПРИКЛАД. Побудувати ДСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, які містять підслово 00 або закінчуються на 01.

Шукана мова є об'єднанням двох мов, записаних регулярними виразами $(0+1)^*00(0+1)^*$ і $(0+1)^*01$. В прикладах 2.4 та 2.6 ми побудували ДСА, які розпізнають ці дві мови. Для перевірки того, чи вхідне слово ω належить об'єднанню цих мов, можна аналізувати слово ω двома автоматами паралельно. Наприклад, нехай $\omega = 0101$.

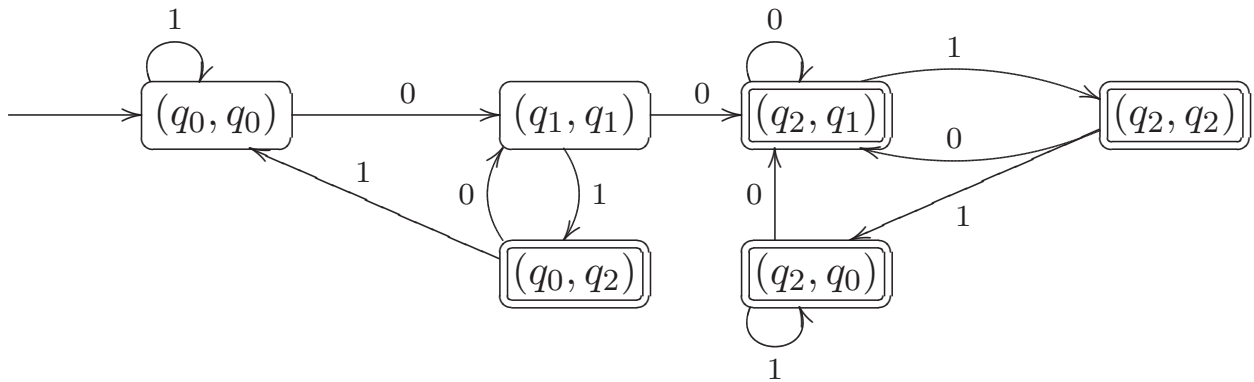
На першому ДСА обчислювальний шлях для слова ω має вигляд $(q_0, q_1, q_0, q_1, q_0)$, а на другому – $(q_0, q_1, q_2, q_1, q_2)$. Оскільки аналіз ω на другому ДСА закінчується в кінцевому стані, то слово належить об'єднанню мов.

Ідея побудови ДСА для об'єднання цих мов полягає в розгляді так званого добутку автоматів. Нехай $A_1 = (Q_1, X, \delta_1, q_0, F_1)$ і $A_2 = (Q_2, X, \delta_2, q_0, F_2)$ – два автомати (зауважимо, що Q_1 і Q_2 можуть мати стани з однаковими назвами, які виконують різні функції в двох ДСА). Визначимо *добуток* $A = A_1 \times A_2$ *автоматів* A_1 і A_2 наступним чином. Нехай $A = (Q, X, \delta, \tilde{q}_0, F)$. Покладемо

$$Q = Q_1 \times Q_2 = \{(q_i, q_j) \mid q_i \in Q_1, q_j \in Q_2\},$$

$$\delta((q_i, q_j), a) = (\delta_1(q_i, a), \delta_2(q_j, a)) \quad \text{і} \quad \tilde{q}_0 = (q_0, q_0).$$

Наприклад, обчислювальний шлях слова $\omega = 0101$ в добутку автоматів A має вигляд: $(q_0, q_0); (q_1, q_1); (q_0, q_2); (q_1, q_1); (q_0, q_2)$. Якщо $q_i \in F_1$ або $q_j \in F_2$, то покладаємо $(q_i, q_j) \in F$, тобто $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$. Очевидно, що даний автомат A розпізнає об'єднання мов, які розпізнаються автоматами A_1 та A_2 . Граф шуканого ДСА має вигляд:



Звернемо увагу на два важливі факти, пов'язані з автоматом A . По-перше, оскільки стани (q_0, q_1) , (q_1, q_0) та (q_1, q_2) є недостижними з початкового стану (q_0, q_0) , то ми їх опускаємо. По-друге, стани (q_2, q_1) , (q_2, q_0) та (q_2, q_2) можуть бути об'єднані в єдиний стан, оскільки всі вони є кінцевими і неможливо їх покинути, якщо автомат перейшов хоча б в один із них.

2.9. ПРИКЛАД. Побудувати ДСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, які містять підслово 00 і закінчуються на 01.

Дана мова є перетином двох мов, записаних регулярними виразами $(0+1)^*00(0+1)^*$ і $(0+1)^*01$. В прикладі 2.8 побудовано добуток автоматів, який розпізнає об'єднання заданих мов. Тут ми розглянемо той самий добуток ДСА, в якому лише поміняємо множину кінцевих станів, поклавши $F = F_1 \times F_2$. Граф шуканого автомата такий же, як і автомата з прикладу 2.8, з тією різницею, що множина F кінцевих станів містить тільки стан (q_2, q_2) .

2.10. ПРИКЛАД. Побудувати ДСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, які містять підслово 00 і не закінчуються на 01.

Дана мова є різницею мов, записаних регулярними виразами $(0+1)^*00(0+1)^*$ і $(0+1)^*01$. Тому ми можемо використати той самий добуток автоматів, що і в прикладах 2.8 та 2.9, включивши в множину кінцевих станів ті пари, перша компонента яких є кінцевим станом першого ДСА, а друга компонента не є кінцевим станом другого автомата, тобто поклавши $F = F_1 \times (Q_2 \setminus F_2)$. Граф шуканого автомата такий же, як і автомата з прикладу 2.8, з тією різницею, що множина F кінцевих станів містить стани (q_2, q_0) і (q_2, q_1) .

Частковим випадком різниці формальних мов є доповнення $\bar{L} = X^* \setminus L$. В цьому випадку використовуємо наступну конструкцію. Зауважимо, що для автомата $A = (Q, X, \delta, q_0, F)$ вхідне слово $\omega \in L(A)$ тоді і тільки тоді, коли $\delta(q_0, \omega) \in F$. Аналогічно $\omega \notin L(A)$ тоді і лише тоді, коли $\delta(q_0, \omega) \notin F$. Звідси випливає, що ДСА $(Q, X, \delta, q_0, Q \setminus F)$ розпізнає доповнення мови $L(A)$.

Таким чином, ми довели наступну теорему:

2.11. ТЕОРЕМА. *Клас формальних мов, які розпізнаються детермінованими скінченними автоматами, є замкненим відносно скінченних об'єднань, перетинів, доповнення, різниці та симетричної різниці.*

Рекомендована література : [16, с. 33–45], [17, с. 289–294], [23, с. 23–38].

Питання та вправи до параграфа 2.

2.1. Як будується добуток $A = A_1 \times A_2$ автоматів A_1 і A_2 ?

2.2. Сформулюйте теорему про замкненість класу формальних мов, які розпізнаються детермінованими скінченними автоматами.

2.3. Визначити, який з бітових рядків

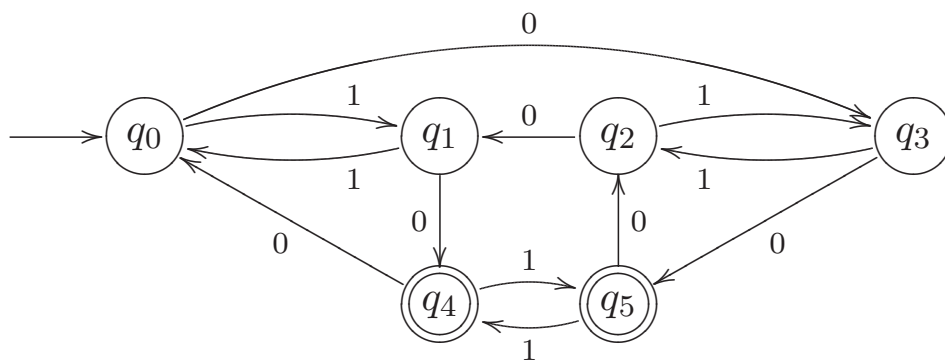
а) 1110

в) 110001010010

б) 101010

г) 000000000111

розпізнається детермінованим скінченним автоматом без виходу A , що заданий графом:



2.4. Серед слів регулярної мови $(10)^*$ визначити ті, які належать мові $L(A)$ автомата A з попереднього прикладу.

2.5. Розглянемо ДСА без виходу $A_{m,d} = (Q, X, \delta, q_0, F)$, де $m, d \in \mathbb{N}$, $Q = \{q_0, q_1, \dots, q_{m-1}\}$, $X = \{0, 1, \dots, d-1\}$, $F = \{q_1\}$ і $\delta(q_i, k) = q_{(di+k) \bmod m}$. Побудувати граф автомата $A_{7,2}$ і визначити, який з бітових рядків розпізнається ДСА:

а) 0101

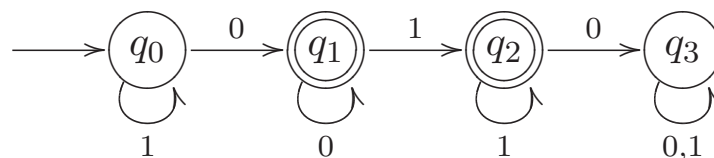
в) 1101101010

б) 11010

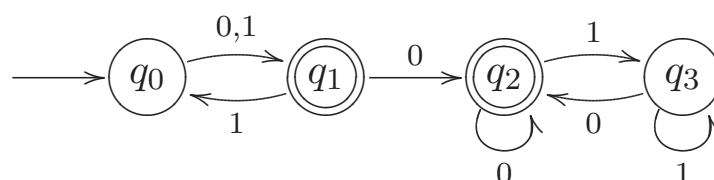
г) 11110000.

2.6. Знайти мову, що розпізнається ДСА без виходу:

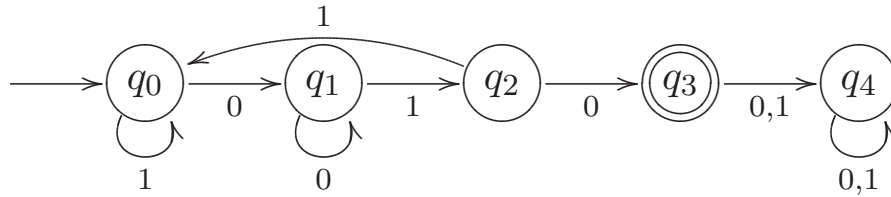
а)



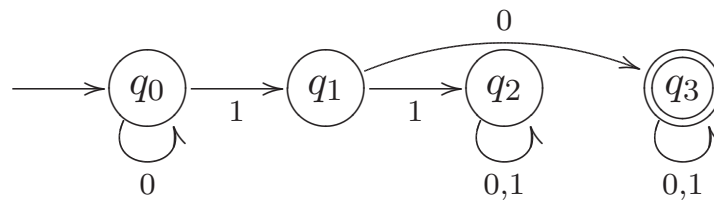
б)



в)



г)



2.7. Побудувати ДСА без виходу, які розпізнають формальні мови:

а) $\{0, 1\}$;б) $\{10, 101\}$;в) $\{0^n \mid n = 2, 3, \dots\}$;г) $\{0^n 1^m \mid n, m \in \mathbb{N}\}$.

2.8. Побудувати ДСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, що містять підслово 100101.

2.9. Побудувати ДСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, що мають префікс 010.

2.10. Побудувати ДСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, що закінчуються на 101.

2.11. Побудувати ДСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, що починаються на 010 або закінчуються на 101.

2.12. Побудувати ДСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, що містять підслово 11 і не закінчуються на 101.

2.13. Побудувати ДСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, що мають префікс 010, суфікс 101 і містять підслово 0000.

§ 3. Недетерміновані скінченні автомати без виходу

Недетермінований скінченний автомат (НСА) без виходу $A = (Q, X, \delta, q_0, F)$ визначається так само, як і ДСА, за винятком того, що δ не обов'язково є всюди визначеною функцією, а може бути й відношенням. Це означає, що для кожного стану q і вхідної букви a значення $\delta(q, a)$ є підмножиною множини станів Q , $\delta(q, a) = \{p_1, p_2, \dots, p_k\}$, тобто проаналізувавши букву a в стані q автомат може перейти в довільний зі станів p_1, p_2, \dots, p_k . Якщо $\delta(q, a) = \emptyset$, то автомат не переходить в жоден стан, і вважається, що вхідне слово не розпізнається НСА, незважаючи на те що деякі букви слова ще не проаналізовані автоматом. В цьому випадку кажуть, що НСА перейшов в тупиковий стан. Крім переходів в декілька станів, в НСА дозволяються також e -переходи (e -такти). При e -переході головка автомата нічого не виконує (не читає і не рухається), але стан при цьому може змінитися на довільний із заданих цим переходом станів.

З сказаного вище випливає, що δ можна визначити як функцію, значення якої є підмножинами множини станів Q , тобто

$$\delta : Q \times (X \cup \{e\}) \rightarrow 2^Q,$$

де через 2^Q позначається сім'я всіх підмножин множини Q .

Функцію δ зручно задавати за допомогою таблиці, стовпчики якої відмічені різними буквами вхідного алфавіту, а рядки – станами автомата. Для функції переходів на перетині i -го рядка, відміченого станом $q \in Q$, і j -го стовпчика, відміченого буквою $a \in X \cup \{e\}$, знаходиться значення $\delta(q, a) = \{p_1, p_2, \dots, p_k\}$.

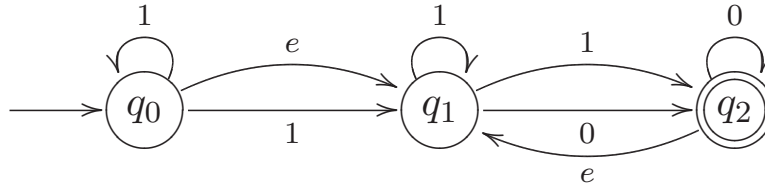
3.1. ПРИКЛАД. Функція δ автомата

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

задається таблицею:

δ	0	1	e
q_0	\emptyset	$\{q_0, q_1\}$	$\{q_1\}$
q_1	$\{q_2\}$	$\{q_1, q_2\}$	\emptyset
q_2	$\{q_2\}$	\emptyset	$\{q_1\}$

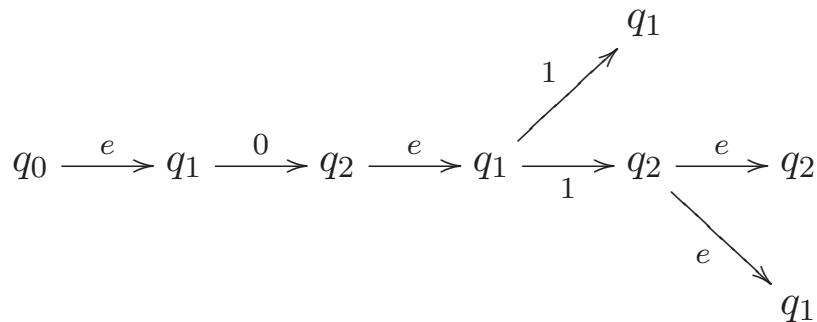
НСА, так як і ДСА, задаються графами, вершинами яких є стани автомата, а за допомогою стрілок зображуються переходи. Якщо $\delta(q, a) = \{p_1, p_2, \dots, p_k\}$, то проводиться k стрілок з вершини q до кожної з вершин p_1, p_2, \dots, p_k , і всі стрілки помічаються буквою a . Наприклад, граф автомата A з прикладу 3.1 має вигляд:



Вхідне слово ω може мати більше одного обчислювального шляху. Наприклад, слово $\omega = 01$ має три обчислювальні шляхи:

$$\begin{aligned} q_0 &\xrightarrow{e} q_1 \xrightarrow{0} q_2 \xrightarrow{e} q_1 \xrightarrow{1} q_1, \\ q_0 &\xrightarrow{e} q_1 \xrightarrow{0} q_2 \xrightarrow{e} q_1 \xrightarrow{1} q_2, \\ q_0 &\xrightarrow{e} q_1 \xrightarrow{0} q_2 \xrightarrow{e} q_1 \xrightarrow{1} q_2 \xrightarrow{e} q_1. \end{aligned}$$

Обчислювальні шляхи вхідного слова ω утворюють кореневе дерево виведення, бо всі вони виходять з однієї і тієї ж вершини q_0 та їх гілки не утворюють циклів. Зобразимо дерево виведення слова $\omega = 01$ (ми додаємо стрілку $q_2 \xrightarrow{e} q_2$, щоб наголосити, що другий шлях закінчується в вершині q_2):



Деякі з цих обчислювальних шляхів закінчуються в кінцевому стані, а інші – ні. Як же в цьому випадку визначити, чи розпізнається вхідне слово НСА? Вважають, що автомат розпізнає слово ω , якщо принаймні один з обчислювальних шляхів закінчується в кінцевому стані. Наприклад, другий обчислювальний шлях слова $\omega = 01$ закінчується в q_2 , і тому воно розпізнається НСА A з прикладу 3.1.

Щоб визначити строго, коли НСА розпізнає вхідне слово ω , потрібно ввести поняття *e-замикання* множини. Назвемо *e-замиканням* підмножини $P \subset Q$ множину станів, які досягаються з станів $q \in P$ *e*-переходами (включаючи переходи з q в q). Тобто

$$cl_e(P) = \{p \in Q \mid (p \in P) \vee (\exists q_0, \dots, q_m)[q_0 \in P, q_m = p, q_{i+1} \in \delta(q_i, e)]\}.$$

Наприклад, $cl_e(\{q_0\}) = \{q_0, q_1\}$, а $cl_e(\{q_2\}) = \{q_1, q_2\}$ для автомата з прикладу 3.1.

Продовжимо функцію переходів δ на множину $2^Q \times (X \cup \{e\})$, поклавши

$$\delta(P, a) = cl_e\left(\bigcup_{q \in cl_e(P)} \delta(q, a)\right),$$

а далі продовжимо її на $2^Q \times X^*$ наступним чином:

$$\delta(P, e) = cl_e(P),$$

$$\delta(P, \omega a) = \delta(\delta(P, \omega), a), \text{ якщо } \omega \in X^* \text{ і } a \in X.$$

Так, у прикладі 3.1 маємо, що $\delta(\{q_0\}, 0) = \{q_1, q_2\}$, а $\delta(\{q_1, q_2\}, 1) = \{q_1, q_2\}$, тому $\delta(\{q_0\}, 01) = \{q_1, q_2\}$.

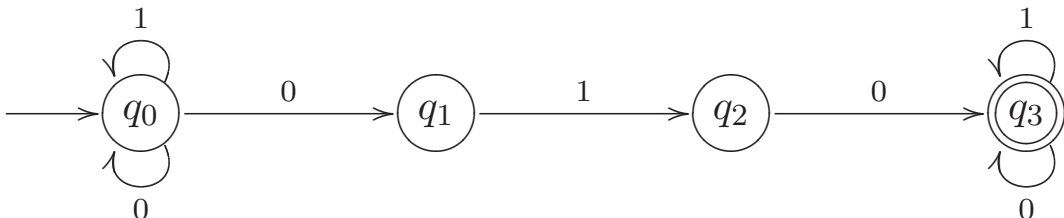
Зауважимо, що $\delta(\{q_0\}, \omega)$ є множиною всіх останніх станів (не обов'язково кінцевих!) обчислювальних шляхів слова ω .

Тепер можна строго визначити, що *НСА* (Q, X, δ, q_0, F) *розпізнає слово* ω , якщо $\delta(\{q_0\}, \omega) \cap F \neq \emptyset$. Через $L(A)$ позначаємо мову, яка розпізнається НСА A , тобто

$$L(A) = \{\omega \in X^* \mid \delta(\{q_0\}, \omega) \cap F \neq \emptyset\}.$$

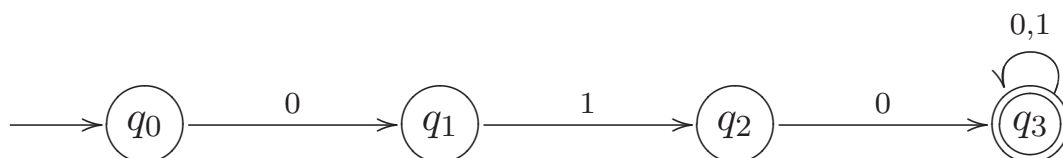
3.2. ПРИКЛАД. Побудувати НСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, що містять підслово 010.

Будуємо даний автомат так само, як ми будували ДСА, тільки в початковому стані додаємо дві петлі з позначками 0 і 1. З їх допомогою автомат чекає доти, доки не знайде підслово 010. Маючи ці дві петлі, не потрібно покладати $\delta(q_1, 0) = q_1$ і $\delta(q_2, 1) = q_0$, як це робилося для ДСА. Справді, достатньо просто покласти $\delta(q_1, 0) = \delta(q_2, 1) = \emptyset$. Граф НСА має вигляд:

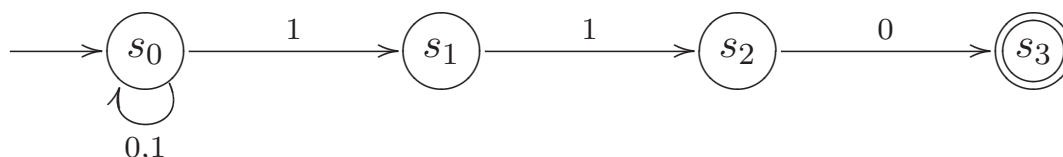


3.3. ПРИКЛАД. Побудуємо НСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, що починаються з 010 або закінчуються на 110.

Множина всіх бінарних слів, які починаються з 010, розпізнається НСА, заданим графом:

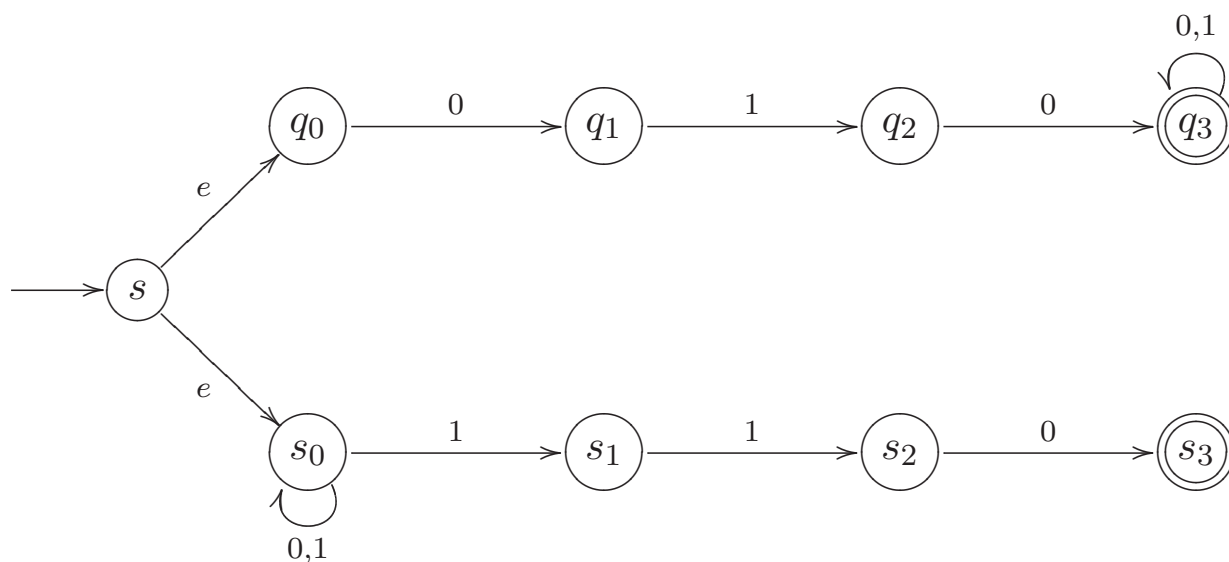


Множина всіх бінарних слів, які закінчуються на 110, розпізнається автоматом:



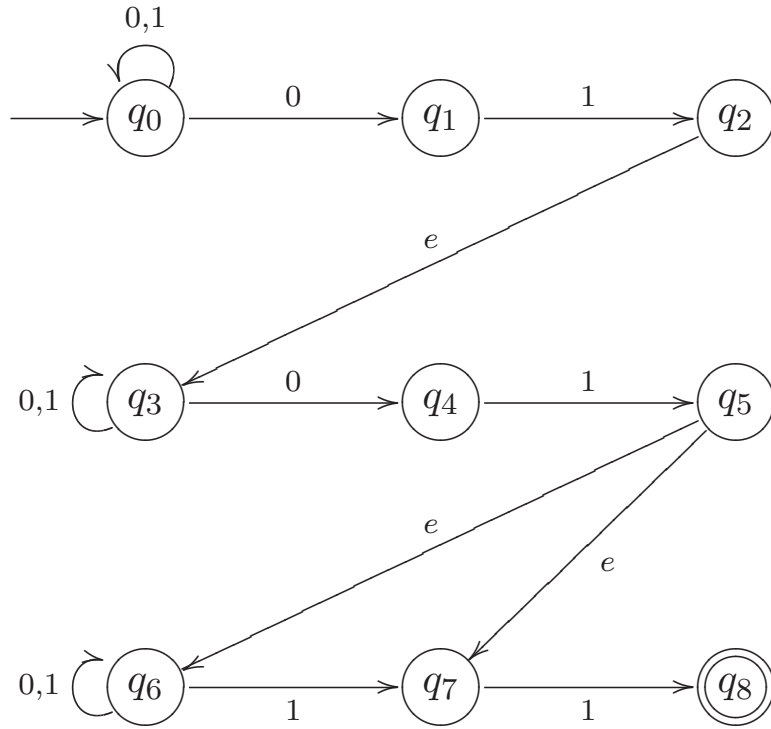
Зауважимо, що в цьому автоматі кінцевий стан не має вихідних стрілок, тобто $\delta(s_3, 0) = \delta(s_3, 1) = \emptyset$. Таким чином, якщо аналізуючи вхідне слово ω , автомат переходить в тупиковий кінцевий стан s_3 раніше, ніж слово ω повністю проаналізується НСА, то з допомогою цього шляху ω не розпізнається (але це не означає, що ω не розпізнається НСА).

Насамкінець об'єднаємо ці два графи в один, додавши новий початковий стан і дві ϵ -стрілки, які виходять з нього в старі початкові стани. В результаті одержимо наступний граф шуканого НСА:



3.4. ПРИКЛАД. Побудуємо НСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, що містять принаймні два входження підслова 01 і закінчуються на 11.

Використаємо e -переходи, щоб об'єднати три простіших НСА в один. В результаті отримаємо:



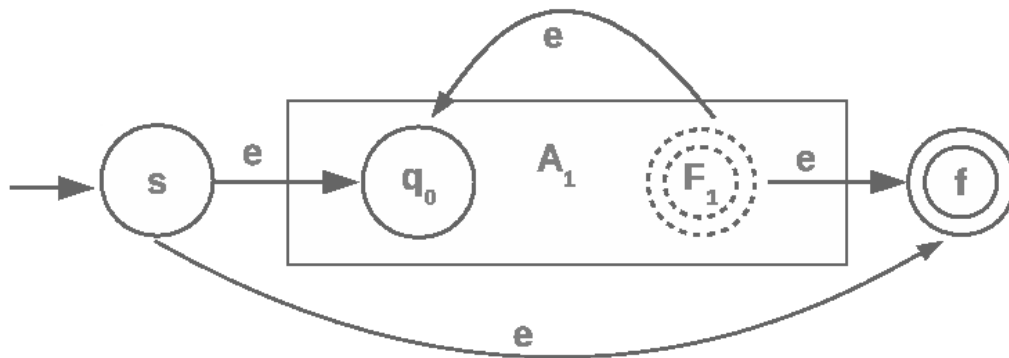
Зауважимо, що буква 1 другого входження підслова 01 може співпасти з першою буквою суфікса 11, але це не є проблемою у випадку НСА: слід просто додати ще одну e -стрілку з стану q_5 в стан q_7 .

3.5. ПРИКЛАД. Нехай A_1 і A_2 – два НСА. Побудуємо НСА A , для якого $L(A) = L(A_1) \circ L(A_2)$.

Нехай $A_1 = (Q_1, X, \delta_1, q_0^1, F_1)$ і $A_2 = (Q_2, X, \delta_2, q_0^2, F_2)$, де $Q_1 \cap Q_2 = \emptyset$. Побудуємо автомат $A = (Q, X, \delta, q_0, F)$ наступним чином. Покладемо $Q = Q_1 \cup Q_2$. Початковий стан q_0^1 автомата A_1 є початковим станом автомата A , а множина F кінцевих станів A дорівнює F_2 . Також для кожного $q \in F_1$ довізначимо $\delta(q, e) = q_0^2$, тобто проведемо e -стрілки з кожного стану $q \in F_1$ в початковий стан q_0^2 автомата A_2 . На решту наборах функція δ визначається так само, як функції δ_1 і δ_2 . Легко бачити, що автомат A розпізнає мову $L(A) = L(A_1) \circ L(A_2)$.

3.6. ПРИКЛАД. Нехай A_1 – НСА. Побудуємо НСА A , для якого $L(A) = L(A_1)^*$.

Нехай $A_1 = (Q_1, X, \delta_1, q_0, F_1)$. Побудуємо граф автомата A , додавши новий початковий стан s і єдиний кінцевий стан f . Проведемо e -стрілки з стану s в старий початковий стан $q_0 \in Q_1$ та з кожного $q_i \in F_1$ в новий кінцевий стан f . Далі відкладемо з кожного стану $q_i \in F_1$ e -стрілку в початковий стан q_0 автомата A_1 . Насамкінець додамо e -стрілку з початкового стану s в новий кінцевий стан f (при цьому порожнє слово ϵ розпізнається автоматом). Шуканий НСА показано на рисунку:



З прикладів 3.5 та 3.6 випливає наступна теорема:

3.7. ТЕОРЕМА. *Клас формальних мов, які розпізнаються недетермінованими скінченними автоматами, замкнений відносно конкатенації та ітерації.*

Рекомендована література : [2, с. 134–138], [12, с. 309–312], [16, с. 55–60], [23, с. 38–45].

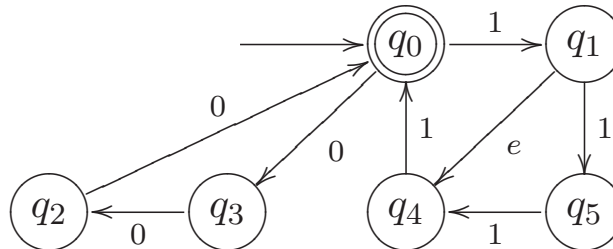
Питання та вправи до параграфа 3.

- 3.1. У чому полягає різниця між ДСА і НСА?
- 3.2. Як за деревом виведення вхідного слова з'ясувати, чи розпізнається воно автоматом?
- 3.3. Сформулюйте теорему про замкненість класу формальних мов, які розпізнаються недетермінованими скінченними автоматами.
- 3.4. Визначити, який з бітових рядків

а) 00010
б) 11000

в) 110001111000
г) 000000000111

розпізнається недетермінованим скінченним автоматом без виходу A , що заданий графом:

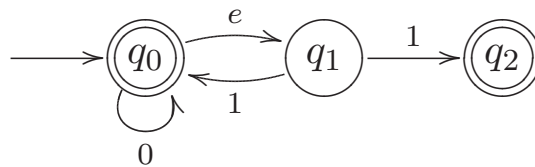


Знайти e -замикання множини $\{q_0, q_1\}$.

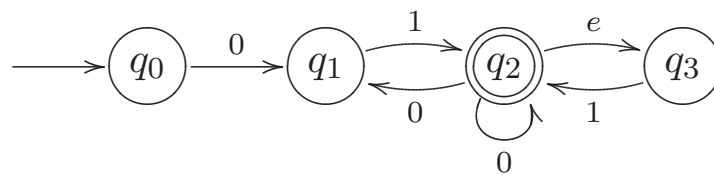
3.5. Серед слів регулярної мови 0^*1^* визначити ті, які належать мові $L(A)$ НСА без виходу A з попереднього прикладу. Задати його як п'ятірку $A = (Q, X, \delta, q_0, F)$.

3.6. Знайти мову, що розпізнається НСА без виходу:

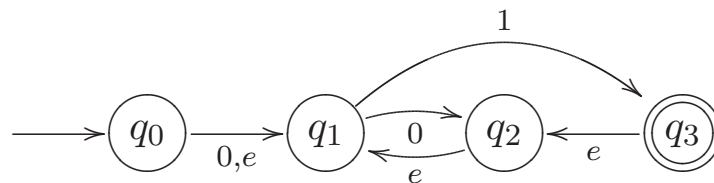
а)



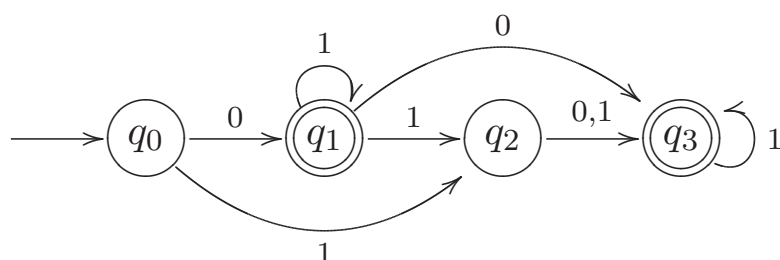
б)



в)



г)



3.7. Побудувати дерева виведення слова 010011 для кожного автомата з попереднього прикладу і з їх допомогою з'ясувати, які з автоматів розпізнають дане слово.

3.8. Побудувати НСА без виходу, які розпізнають формальні мови:

а) $\{1, e\}$;

б) $\{01, 101\}$;

в) $\{0^n \mid n = 2, 3, \dots\}$;

г) $\{0^n 1^m \mid n, m \in \mathbb{N}\}$.

3.9. Побудувати НСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, що містять підслово 00101.

3.10. Побудувати НСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, що мають префікс 111.

3.11. Побудувати НСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, що закінчуються на 001.

3.12. Побудувати НСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, що починаються на 111 або закінчуються на 001.

3.13. Побудувати НСА, який розпізнає всі слова в алфавіті $X = \{0, 1\}$, що містять щонайменше три входження підслова 010.

3.14. Користуючись розглянутою в прикладі 3.5 параграфа 3 конструкцією побудувати автомат, який розпізнає всі слова в алфавіті $X = \{0, 1\}$, що починаються на 11 і закінчуються на 01.

3.15. Користуючись розглянутою в прикладі 3.6 параграфа 3 конструкцією побудувати НСА, який розпізнає регулярну мову $\{1, 01\}^*$ в алфавіті $X = \{0, 1\}$.

§ 4. Перетворення НСА до ДСА

Недетерміновані скінченні автомати будуються простіше, ніж детерміновані скінченні автомати, але вони є ідеалізованими машинами і не можуть бути ефективно застосовані на практиці, бо реальна машина може працювати тільки за однозначно визначеним алгоритмом. Виявляється, що є простий метод для перетворення НСА в рівносильний ДСА в тому розумінні, що обидва автомати розпізнають одну і ту ж формальну мову.

Розглянемо НСА $A = (Q, X, \delta, q_0, F)$. Для кожного $\omega \in X^*$ позначимо через Q_ω множину останніх (не обов'язково кінцевих!) станів обчислювальних шляхів слова ω , тобто $Q_\omega = \delta(\{q_0\}, \omega)$, де δ – функція переходів, визначена на множині $2^Q \times X^*$. Зокрема, $Q_e = cl_e(\{q_0\})$. Таким чином, слово ω розпізнається НСА тоді і тільки тоді, коли $Q_\omega \cap F \neq \emptyset$. Отже, можна взяти підмножини $Q_\omega \subset Q$ в якості станів нового рівносильного ДСА. Іншими словами, побудуємо ДСА $A' = (Q', X, \delta', Q_e, F')$ з наступними компонентами:

$$Q' = \{Q_\omega \mid \omega \in X^*\}, \quad F' = \{Q_\omega \mid Q_\omega \cap F \neq \emptyset\},$$

$$\delta'(Q_\omega, a) = Q_{\omega a}, \text{ де } \omega \in X^*, a \in X.$$

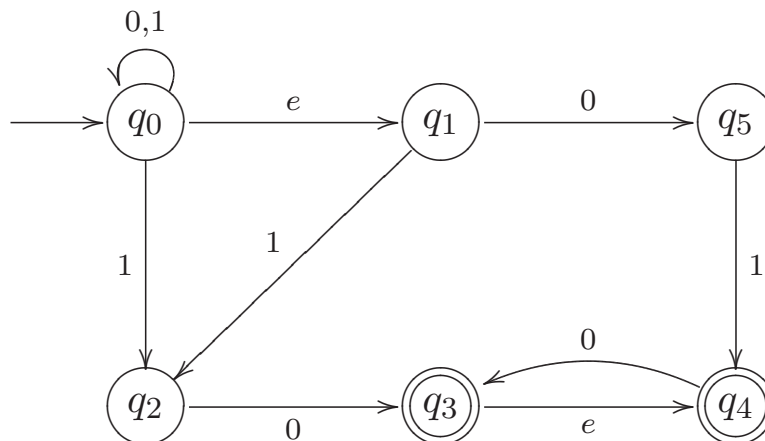
Кожен стан $Q_\omega \in Q'$ є підмножиною множини Q . Оскільки множина Q містить $2^{|Q|}$ підмножин, то Q' є скінченною множиною. Якщо $Q_\omega = Q_\nu$, то $Q_{\omega a} = Q_{\nu a}$ для всіх $a \in X$. Звідси випливає, що означення функції переходів δ' є коректним. Дана конструкція називається *побудовою ДСА за допомогою підмножин*.

Розглянемо деякі приклади.

4.1. ПРИКЛАД. Побудувати ДСА, який розпізнає ту ж формальну мову, що і НСА $A = (Q, \{0, 1\}, \delta, q_0, F)$, де $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$, $F = \{q_3, q_4\}$ і функція переходів задана таблицею:

δ	0	1	e
q_0	$\{q_0\}$	$\{q_0, q_2\}$	$\{q_1\}$
q_1	$\{q_5\}$	$\{q_2\}$	-
q_2	$\{q_3\}$	-	-
q_3	-	-	$\{q_4\}$
q_4	$\{q_3\}$	-	-
q_5	-	$\{q_4\}$	-

Спершу побудуємо граф заданого в умові НСА:



Побудуємо рівносильний ДСА за наступним алгоритмом:

Крок 1. Нехай $Q_e = cl_e(\{q_0\})$ – початковий стан шуканого ДСА. Покладемо $Q' = \{Q_e\}$ і $F' = \emptyset$. Якщо $Q_e \cap F \neq \emptyset$, то додаємо стан Q_e до множини F' кінцевих станів.

Крок 2. Повторюємо наступні пункти доти, доки значення $\delta'(Q_\omega, a)$ не буде визначене для всіх станів $Q_\omega \in Q'$ і всіх $a \in \{0, 1\}$:

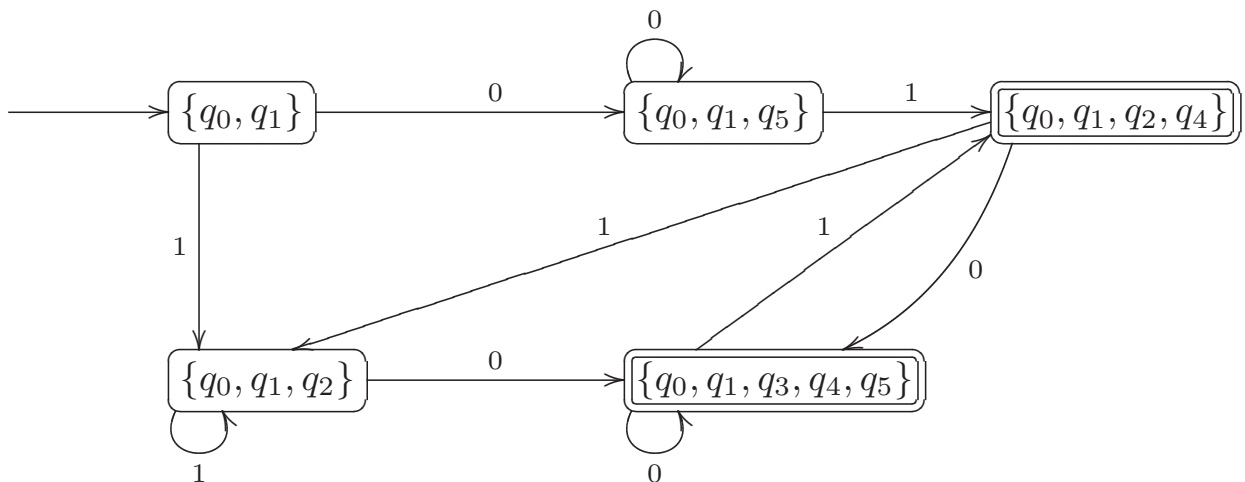
- Вибираємо такі $Q_\omega \in Q'$ і $a \in \{0, 1\}$, що значення $\delta'(Q_\omega, a)$ функції переходів ще не визначене.
- Покладемо $Q_{\omega a} = \delta'(Q_\omega, a)$.
- Якщо $Q_{\omega a} \notin Q'$, то додаємо стан $Q_{\omega a}$ до множини станів Q' , а також додаємо його до множини кінцевих станів F' , якщо $Q_{\omega a} \cap F \neq \emptyset$.

Результат роботи алгоритму поданий у таблиці:

δ'	0	1
$Q_e = \{q_0, q_1\}$	$\{q_0, q_1, q_5\}$	$\{q_0, q_1, q_2\}$
$Q_0 = \{q_0, q_1, q_5\}$	$\{q_0, q_1, q_5\} = Q_0$	$\{q_0, q_1, q_2, q_4\}$
$Q_1 = \{q_0, q_1, q_2\}$	$\{q_0, q_1, q_3, q_4, q_5\}$	$\{q_0, q_1, q_2\} = Q_1$
$Q_{01} = \{q_0, q_1, q_2, q_4\}$	$\{q_0, q_1, q_3, q_4, q_5\}$	$\{q_0, q_1, q_2\} = Q_1$
$Q_{10} = \{q_0, q_1, q_3, q_4, q_5\}$	$\{q_0, q_1, q_3, q_4, q_5\} = Q_{10}$	$\{q_0, q_1, q_2, q_4\} = Q_{01}$

Зауважимо, що при виконанні кроку 2 не потрібно розглядати стани Q_{00} , Q_{000} , Q_{001} і т.д., бо $Q_{00} = Q_0$, $Q_{000} = Q_{00} = Q_0$ і $Q_{001} = Q_{01}$. Оскільки $Q_{11} = Q_1$, то не треба також розглядати стани $Q_{11\omega}$ для кожного $\omega \in \{0, 1\}^*$.

Граф рівносильного до НСА ДСА має вигляд:



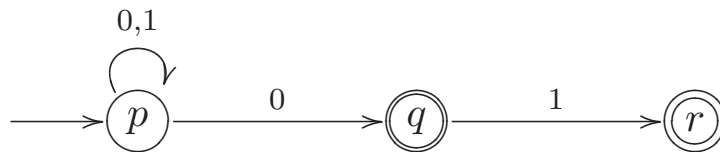
4.2. ПРИКЛАД. Побудувати ДСА, який рівносильний до НСА

$$A = (\{p, q, r\}, \{0, 1\}, \delta, p, \{q, r\}),$$

де функція переходів δ задана таблицею:

δ	0	1
p	$\{p, q\}$	$\{p\}$
q	—	$\{r\}$
r	—	—

Зобразимо граф даного НСА:



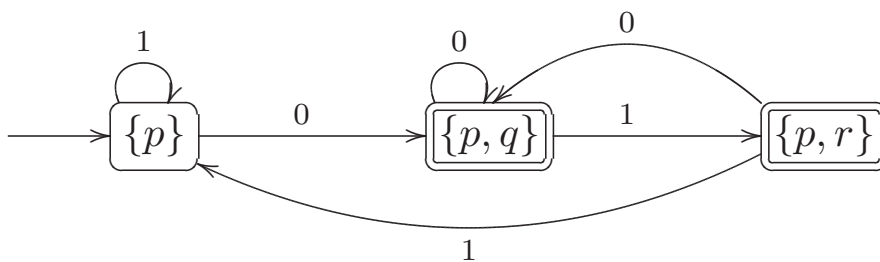
Використовуючи описаний в прикладі 4.1 алгоритм побудови рівносильного до НСА ДСА, одержимо наступну таблицю функції переходів ДСА:

δ'	0	1
$Q_e = \{p\}$	$\{p, q\}$	$\{p\} = Q_e$
$Q_0 = \{p, q\}$	$\{p, q\} = Q_0$	$\{p, r\}$
$Q_{01} = \{p, r\}$	$\{p, q\} = Q_0$	$\{p\} = Q_e$

Таким чином, шуканий ДСА задається як п'ятірка

$$A' = (\{\{p\}, \{p, q\}, \{p, r\}\}, \{0, 1\}, \delta', \{p\}, \{\{p, q\}, \{p, r\}\}),$$

а його граф має вигляд:



З вищесказаного випливає, що для кожного НСА існує ДСА, який розпізнає ту ж мову, що і НСА. Оскільки кожен ДСА є частковим випадком НСА, то ми довели наступну теорему:

4.3. ТЕОРЕМА. *Формальна мова розпізнається НСА тоді і лише тоді, коли вона розпізнається ДСА.*

Теорема 4.3 дає простий метод побудови ДСА, який розпізнає мову L : спершу ми будуємо НСА для L , а потім перетворюємо його в рівносильний ДСА.

4.4. ПРИКЛАД. Нехай $A = (\{p, q, r, s\}, \{0, 1\}, \delta, p, \{q, s\})$ – НСА, де

δ	0	1
p	$\{q, s\}$	$\{q\}$
q	$\{r\}$	$\{q, r\}$
r	$\{s\}$	$\{p\}$
s	—	$\{p\}$

Побудувати НСА, який розпізнає мову $\overline{L(A)}$.

Швидко побудувати шуканий НСА A' по заданому НСА A не вдасться, бо розглянутий в прикладі 2.10 для ДСА метод заміни некінцевих станів автомата A на кінцеві стани автомата A' в даному випадку незастосовний. Дійсно, можлива ситуація, коли для деякого слова ω обидва перетини $Q_\omega \cap F$ і $Q_\omega \cap (Q \setminus F)$ непорожні, і тому слово ω розпізнається обома автоматами A та A' . (Наприклад, при $Q_{01} = \{p, q, r\}$ і $F = \{q, s\}$ слово 01 розпізнається як A , так і A' .) Таким чином, $L(A') \neq \overline{L(A)}$.

Замість цього можна застосувати конструкцію підмножин для побудови шуканого НСА. Спершу знаходимо рівносильний до A ДСА A_D , а потім будуємо автомат A' , граф якого такий же, як і автомата A_D , за винятком того, що кінцевими та некінцевими станами автомата A' є відповідно некінцеві та кінцеві стани ДСА A_D .

Використовуючи конструкцію підмножин побудуємо детермінований автомат $A_D = (Q_D, \{0, 1\}, \delta_D, \{p\}, F_D)$, де множина станів Q_D і функція переходів δ_D задані таблицею:

δ_D	0	1
$Q_e = \{p\}$	$\{q, s\}$	$\{q\}$
$Q_0 = \{q, s\}$	$\{r\}$	$\{p, q, r\}$
$Q_1 = \{q\}$	$\{r\}$	$\{q, r\}$
$Q_{00} = \{r\}$	$\{s\}$	$\{p\}$
$Q_{01} = \{p, q, r\}$	$\{q, r, s\}$	$\{p, q, r\}$
$Q_{11} = \{q, r\}$	$\{r, s\}$	$\{p, q, r\}$
$Q_{000} = \{s\}$	\emptyset	$\{p\}$
$Q_{010} = \{q, r, s\}$	$\{r, s\}$	$\{p, q, r\}$
$Q_{110} = \{r, s\}$	$\{s\}$	$\{p\}$
$Q_{0000} = \emptyset$	\emptyset	\emptyset

Множина F_D кінцевих станів має вигляд:

$$F_D = \{A \subset \{p, q, r, s\} \mid A \cap \{q, s\} \neq \emptyset\} = \\ \{\{q\}, \{s\}, \{q, s\}, \{q, r\}, \{r, s\}, \{p, q, r\}, \{q, r, s\}\}.$$

Таким чином, ДСА

$$A' = (Q_D, \{0, 1\}, \delta_D, \{p\}, Q_D \setminus F_D)$$

розпізнає мову $\overline{L(A)}$, де $Q_D \setminus F_D = \{\{p\}, \{r\}, \emptyset\}$.

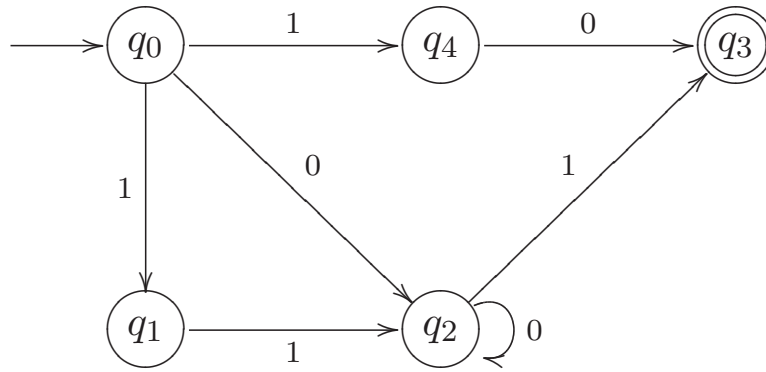
В параграфі 3 першого розділу для регулярного виразу було визначено його помічений граф $G(r)$ так, що слово $\omega \in L(r)$ тоді і лише тоді, коли існує шлях в $G(r)$ з початкової вершини в кінцеву вершину, позначки якого утворюють слово ω . Кожна стрілка графа $G(r)$ помічена однією буквою з множини $\{e\} \cup X$. Легко бачити, що $G(r)$ є графом НСА, який розпізнає мову $L(r)$. Таким чином, одержуємо твердження:

4.5. ТВЕРДЖЕННЯ. *Кожна регулярна мова розпізнається деяким недетермінованим скінченним автоматом.*

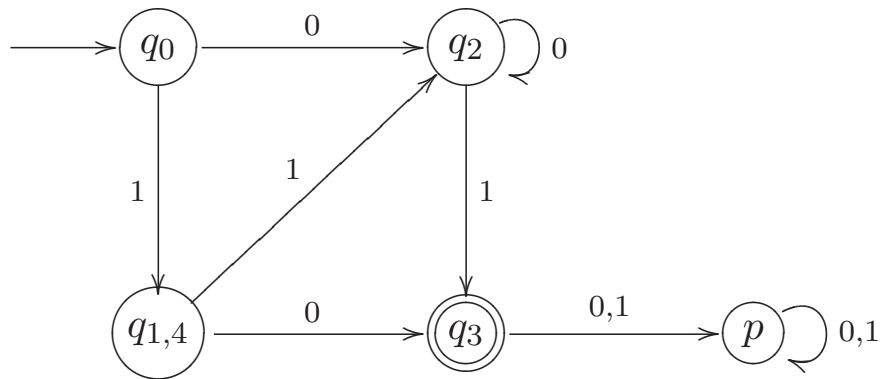
Розглянемо деякі приклади.

4.6. ПРИКЛАД. Побудувати ДСА, який розпізнає регулярну мову, задану регулярним виразом $10 + (0 + 11)0^*1$.

Спершу побудуємо НСА, який розпізнає дану мову, використовуючи описаний в параграфі 3 першого розділу метод:



Далі перетворимо даний НСА в ДСА, використовуючи конструкцію підмножин. Граф шуканого ДСА має вигляд:

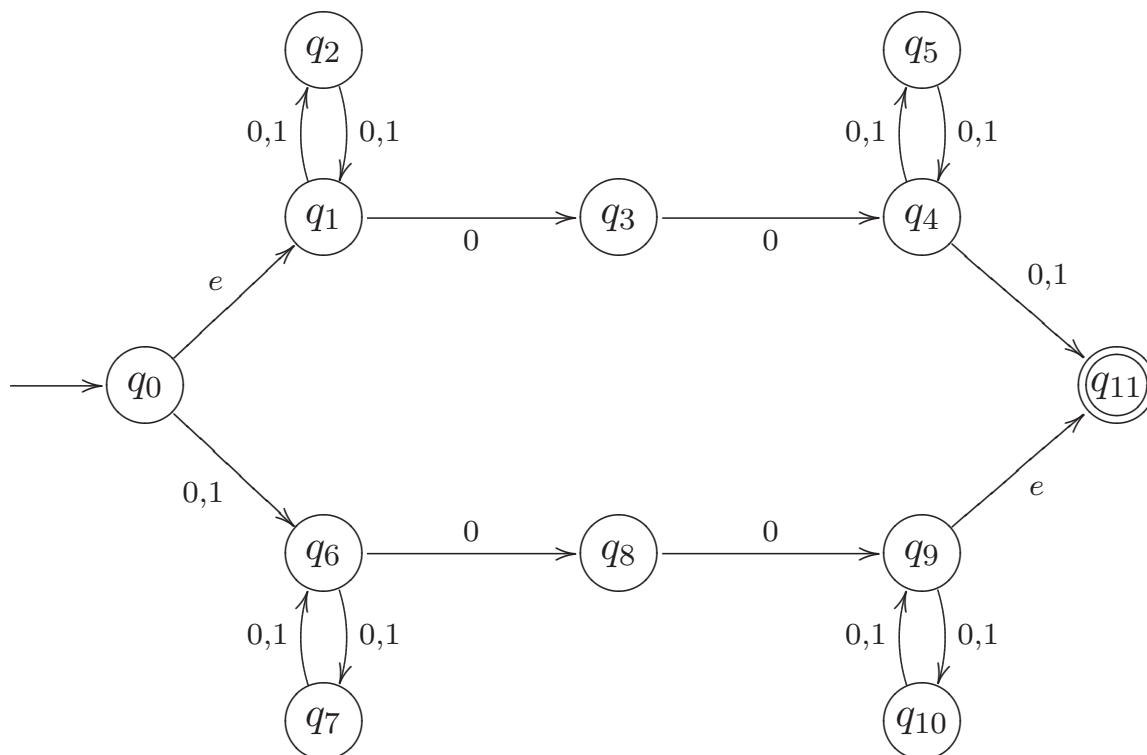


4.7. ПРИКЛАД. Побудувати НСА, який розпізнає всі слова непарної довжини в бінарному алфавіті $X = \{0, 1\}$, які містять підслово 00.

Слово ω , яке містить підслово 00, може бути записане як $\omega = \alpha 00 \beta$ для деяких слів $\alpha, \beta \in \{0, 1\}^*$. Дане слово має непарну довжину тоді і лише тоді, коли довжини $|\alpha|$ і $|\beta|$ слів α і β мають різну парність. Таким чином, L запишеться наступним регулярним виразом:

$$((0 + 1)^2)^* 00 ((0 + 1)^2)^* (0 + 1) + (0 + 1) ((0 + 1)^2)^* 00 ((0 + 1)^2)^*.$$

Використовуючи описаний в параграфі 3 першого розділу метод, побудуємо граф шуканого НСА:



Рекомендована література : [2, с. 138–140], [4, с. 521–531], [13, с. 429–434, 452–461], [16, с. 60–70].

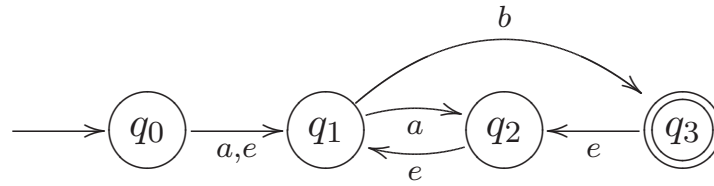
Питання та вправи до параграфа 4.

- 4.1.** Як побудувати ДСА, який рівносильний заданому НСА?
- 4.2.** Як за заданим регулярним виразом r побудувати ДСА, який розпізнає формальну мову $L(r)$?
- 4.3.** Побудувати ДСА, який розпізнає ту ж формальну мову, що і НСА $A = (Q, \{a, b\}, \delta, q_0, F)$, де $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $F = \{q_4\}$ і функція переходів δ задана таблицею:

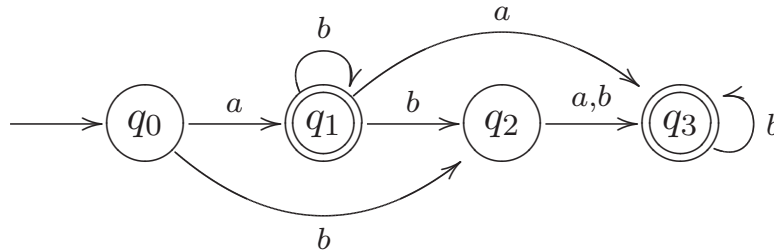
δ	a	b	e
q_0	$\{q_0, q_2\}$	$\{q_1\}$	$\{q_2\}$
q_1	$\{q_3\}$	$\{q_4\}$	-
q_2	-	$\{q_3, q_4\}$	-
q_3	$\{q_3\}$	-	$\{q_4\}$
q_4	$\{q_3\}$	$\{q_3\}$	-

- 4.4.** Побудувати ДСА, які розпізнають ті ж формальні мови, що й НСА A та A' , задані графами:

A :



A' :



4.5. Побудувати детерміновані скінченні автомати, які розпізнають мови $\overline{L(A)}$, $\overline{L(A')}$ та $\overline{L(A) \cap L(A')}$, де A та A' – автомати з попереднього прикладу.

4.6. Побудувати ДСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, що закінчуються на 00 або 11 або 10.

4.7. Побудувати ДСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, що починаються на 11 і закінчуються на 01.

4.8. Побудувати ДСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, в яких третьою буквою зліва та справа є 0.

4.9. Побудувати ДСА, який розпізнає всі слова в бінарному алфавіті $X = \{0, 1\}$, що містять одночасно 11 і 00, або не містять ні 00, ні 11.

4.10. Скласти регулярний вираз для множини всіх непарних натуральних чисел, записаних в четвірковій системі числення, та побудувати ДСА, який розпізнає дану множину.

4.11. Для кожного регулярного виразу r побудувати ДСА, який розпізнає регулярну мову $L(r)$:

- а) $(0 + 10)^*(1 + 01)^*$;
- б) $(0 + 1)^*0(0 + 1)(0 + 1)0(0 + 1)$;
- в) $0(0 + 1)^*0 + 1(0 + 1)^*1$.

§ 5. Скінченні автомати і регулярні мови

В цьому параграфі ми покажемо, що скінченні автомати розпізнають виключно регулярні мови.

5.1. ТЕОРЕМА. *Для формальної мови L наступні умови рівносильні:*

- 1) L – регулярна мова;
- 2) $L = L(A)$ для деякого детермінованого скінченного автомата A ;
- 3) $L = L(A)$ для деякого недетермінованого скінченного автомата A ;
- 4) мова L породжується деякою праволінійною граматикою.

ДОВЕДЕННЯ. Рівносильність 2) \Leftrightarrow 3) доведена в теоремі 4.3, а імплікація 1) \Rightarrow 3) в твердженні 4.5.

3) \Rightarrow 1) Нехай формальна мова L розпізнається НСА $A = (Q, X, \delta, s, F)$. Покажемо, як побудувати регулярний вираз, що позначає мову $L(A) = L$. Спершу перетворимо автомат A до рівносильного автомата A' з єдиним кінцевим станом. Для цього замінимо кінцеві стани автомата A на некінцеві стани, додамо новий кінцевий стан $f \notin Q$ і проведемо e -стрілки з кожного старого кінцевого стану в новий кінцевий стан. Очевидно, що $L(A') = L(A)$. Граф автомата A' є поміченим графом G , кожна стрілка якого позначена єдиним символом з множини $\{e\} \cup X$. Такі графи розглядалися в параграфі 3 першого розділу.

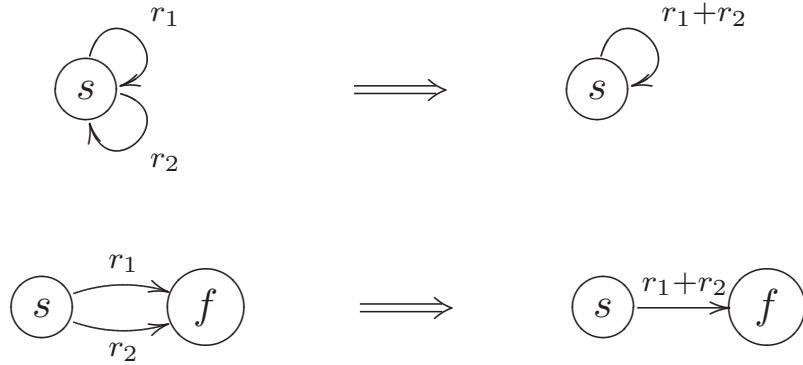
Для кожного шляху π з вершини s в вершину f через $r(\pi)$ позначимо регулярний вираз $r_1 r_2 \dots r_k$, утворений позначками стрілок з π :

$$s \xrightarrow{r_1} \nu_1 \xrightarrow{r_2} \nu_2 \xrightarrow{r_3} \dots \xrightarrow{r_k} \nu_k = f$$

Очевидно, що $L(A) = \bigcup \{L(r(\pi)) \mid \pi - \text{шлях з } s \text{ до } f\}$.

Доведемо індукцією за кількістю вершин графа G , відмінних від s і f , що формальна мова $L(A)$ є регулярною мовою.

База індукції. Для $n = 0$ граф G має тільки дві вершини s та f або єдину вершину $s = f$. Замінімо всі паралельні стрілки з позначками r_1, r_2, \dots, r_m , які ідуть з вершини s в вершину f , на одну стрілку з позначкою $r_1 + r_2 + \dots + r_m$:



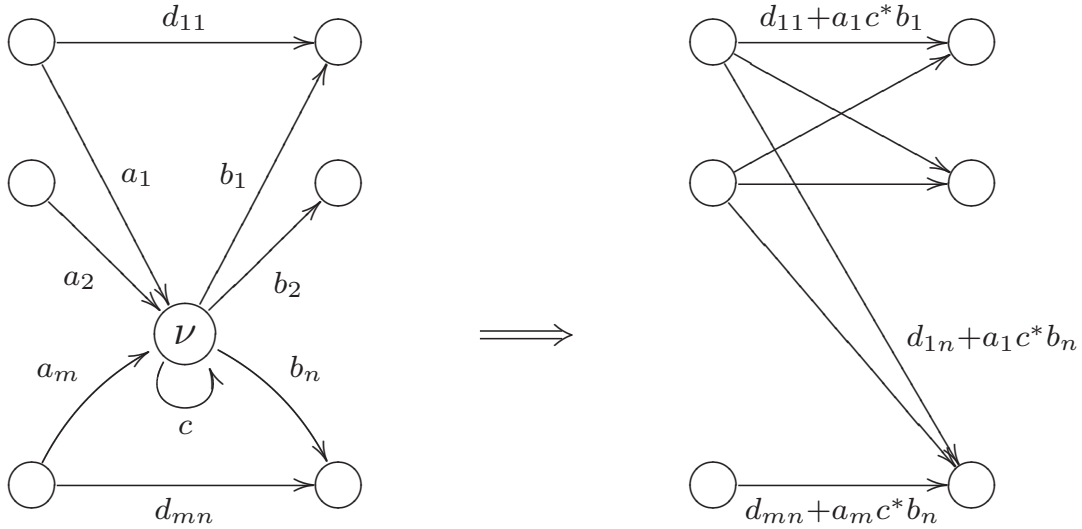
Як наслідок отримаємо один з двох графів:



де a, b, c, d – регулярні вирази.

Очевидно, що для першого графа G мова $L(A) = a^*$ і є регулярною. Розглянемо другий граф і шлях π з s до f . Припустимо, що шлях π проходить $k \geq 1$ разів через вершину f . Тоді π можна подати як послідовність $\pi_1 \pi_2 \dots \pi_k$, де π_1 – це шлях з s в f , а π_2, \dots, π_k – такі шляхи з f в f , що в кожному з них f не є проміжною вершиною. Очевидно, що $r(\pi_1) = a^{n_1}b$ для деякого $n_1 \geq 0$, а для $j = 2, \dots, k$ вираз $r(\pi_j)$ дорівнює c або $da^{n_j}b$ для деякого $n_j \geq 0$, тобто $r(\pi) \in a^*b(c + da^*b)^*$. Отже, $L(A) = a^*b(c + da^*b)^*$ і база індукції доведена.

Індуктивний крок. Для $n \geq 1$ оберемо відмінну від s та f вершину v і видалимо її наступним чином. Спершу за використанням в базі індукції методом вилучимо паралельні стрілки. Таким чином, можна вважати, що існує не більше однієї стрілки з довільної вершини v в довільну вершину ω . Далі припустимо, що v має петлю з позначкою c . Видалимо вершину v разом з цією петлею. Для довільної пари (v, ω) вершин графа G з стрілками $v \xrightarrow{a} v$, $v \xrightarrow{b} \omega$, $v \xrightarrow{d} \omega$ видалимо стрілки $v \xrightarrow{a} v$ і $v \xrightarrow{b} \omega$ та замінимо позначку d стрілки $v \rightarrow \omega$ новою позначкою $d + ac^*b$ (якщо стрілки $v \rightarrow \omega$ в графі G не існує, то вважаємо, що вона має позначку \emptyset). Дане перетворення показано на рисунку:



(Щоб зробити даний рисунок простішим для розуміння, ми опустили деякі стрілки з розташованих ліворуч вершин у вершини, які розміщені праворуч. Вважаємо, що стрілка, яка веде з i -тої вершини зліва до j -тої вершини справа, помічена символом d_{ij} .) Зауважимо, що дане перетворення не змінює формальної мови $L(A)$. Таким чином, за індуктивним припущенням мова $L(A)$ є регулярною.

2) \Rightarrow 4) Нехай мова L розпізнається ДСА $A = (Q, X, \delta, q_0, F)$. Без втрати загальності можна вважати, що $Q \cap X = \emptyset$. Побудуємо праволінійну граматика $G = (N, T, S, P)$, де $N = Q$, $T = X$, $S = q_0$,

$$P = \{q \rightarrow ap \mid \delta(q, a) = p\} \cup \{f \rightarrow e \mid f \in F\}.$$

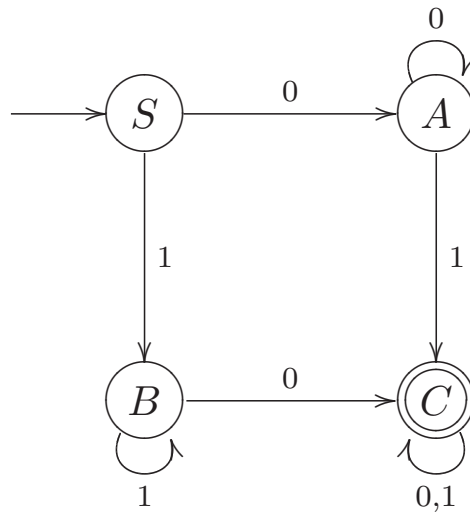
Математичною індукцією легко доводиться, що для будь-яких $p, q \in Q$ і кожного $\omega \in X$, $\delta(q, \omega) = p$ тоді і тільки тоді, коли існує виведення $q \Rightarrow^* \omega p$ в граматичі G . Зокрема, $\omega \in L$ тоді і лише тоді, коли існує виведення $q_0 \Rightarrow^* \omega f \Rightarrow \omega$ для деякого $f \in F$. Звідси випливає, що $L \subset L(G)$. Більше того, оскільки єдиними правилами в P , які в правій частині не містять нетермінальних символів, є $f \rightarrow e$, де $f \in F$, то вищезгадані виведення є єдиними в граматичі G . Таким чином, $L = L(G)$.

4) \Rightarrow 3) Нехай $G = (N, T, S, P)$ – праволінійна граматика. Побудуємо помічений орієнтований граф наступним чином. Множина вершин співпадає з $N \cup \{f\}$, де $f \notin N$. Стан S є початковим станом, а f – єдиним кінцевим станом автомата. Для кожної продукції в G виду $A \rightarrow \omega B$, де $A, B \in N$ і $\omega \in T^*$ проводимо стрілку $A \xrightarrow{\omega} B$ в графі, а для кожної продукції вигляду $A \rightarrow \omega$, де $A \in N$ і $\omega \in T^*$ проводимо стрілку $A \xrightarrow{\omega} f$.

Легко бачити, що кожне виведення $S \Rightarrow^* \omega$ граматики G відповідає шляху графа від початкової вершини S до кінцевої вершини f , позначки якого утворюють слово ω . Таким чином, мова, яка відповідає даному поміченому графу, співпадає з мовою $L(G)$. Оскільки кожному такому поміченому орієнтованому графу відповідає НСА, який розпізнає ту ж мову, то мова $L(G)$ є регулярною.

□

5.2. ПРИКЛАД. Побудуємо праволінійну граматику, яка породжується регулярною мовою, що розпізнається детермінованим скінченим автоматом, який заданий графом:



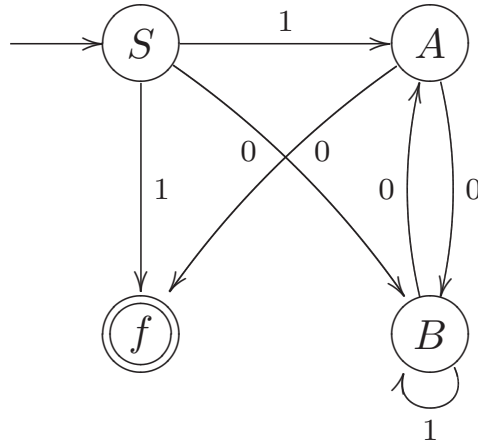
Використовуючи доведення теореми 5.1, побудуємо граматику $G = (N, T, S, P)$, де $N = \{S, A, B, C\}$, $T = \{0, 1\}$ і множина P продукцій містить наступні правила:

$$S \rightarrow 0A \mid 1B, \quad A \rightarrow 0A \mid 1C, \quad B \rightarrow 0C \mid 1B, \quad C \rightarrow 0C \mid 1C \mid e.$$

5.3. ПРИКЛАД. Побудуємо недетермінований скінченний автомат, який розпізнає мову, що породжується праволінійною граматикою G , яка містить наступні продукції:

$$S \rightarrow 1A \mid 0B \mid 1, \quad B \rightarrow 0A \mid 1B, \quad A \rightarrow 0B \mid 0.$$

Граф автомата матиме вигляд:



Визначимо *частку формальних мов* L_1 і L_2 наступним чином:

$$L_1/L_2 = \{\omega \mid (\exists \nu \in L_2)[\omega\nu \in L_1]\}.$$

Наприклад, якщо

$$L_1 = \{\omega \in \{0, 1\}^* \mid \omega \text{ містить парну кількість входжень } 0\},$$

$$L_2 = \{0\} \text{ і } L_3 = \{0, 00\}, \text{ то } L_1/L_3 = \{0, 1\}^*, \text{ а}$$

$$L_1/L_2 = \{\omega \in \{0, 1\}^* \mid \omega \text{ містить непарну кількість входжень } 0\}.$$

5.4. ТВЕРДЖЕННЯ. *Якщо L_1 і L_2 є регулярними мовами, то L_1/L_2 теж є регулярною мовою.*

ДОВЕДЕННЯ. Нехай мова L_1 розпізнається детермінованим скінченним автоматом $A = (Q, X, \delta, q_0, F)$. Побудуємо автомат A' , який розпізнає мову L_1/L_2 . За означенням частки мов для слова $\omega \in L_1/L_2$ існує таке слово $\nu \in L_2$, що $\delta(q_0, \omega\nu) = \delta(\delta(q_0, \omega), \nu) \in F$. Нехай автомат A при аналізі слова ω переходить в стан $q = \delta(q_0, \omega)$. Якщо існує таке $\nu \in L_2$, що $\delta(q, \nu) \in F$, то слово ω має розпізнаватися A' , а тому q слід визначити кінцевим станом автомата для мови L_1/L_2 . Таким чином, покладемо

$$F' = \{q \in Q \mid (\exists \nu \in L_2) [\delta(q, \nu) \in F]\} \text{ і } A' = (Q, X, \delta, q_0, F').$$

Легко бачити, що A' розпізнає мову L_1/L_2 .

□

З теорем 5.1, 2.11 і 3.7 та твердження 5.4 випливає наступний наслідок:

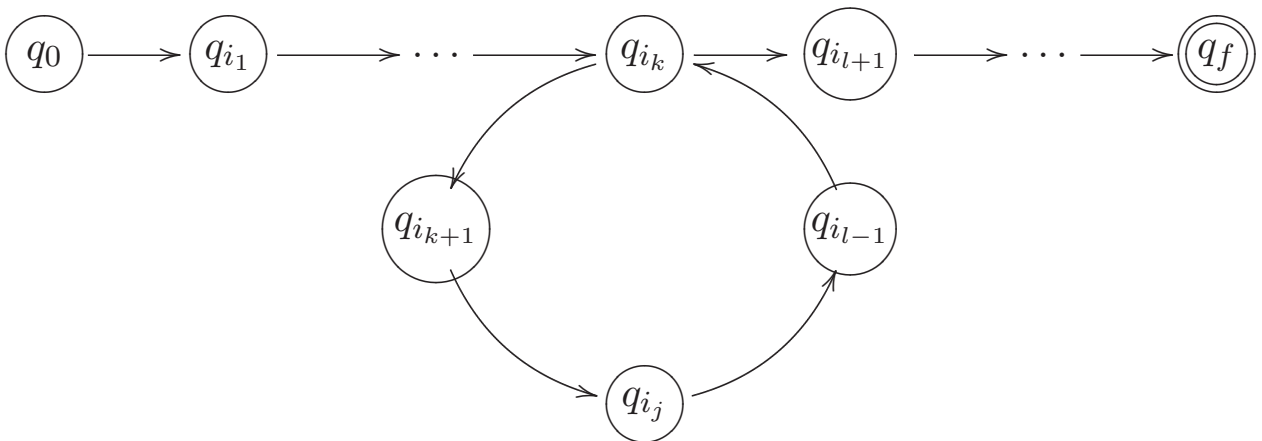
5.5. НАСЛІДОК. *Клас регулярних мов замкнений відносно скінченних об'єднань, перетинів, доповнення, різниці, симетричної різниці, конкатенації, ітерації та частки.*

З наслідку 5.5 випливає, що клас регулярних мов є досить широким. Природно виникає питання: чи даний клас співпадає з класом усіх формальних мов? Наступна лема стверджує, що це не так.

5.6. ЛЕМА. *(про роздування для регулярних мов) Нехай L – нескінченна регулярна мова, тоді існує така (залежна від L) константа s , що кожне слово $\omega \in L$, яке задовольняє нерівність $|\omega| \geq s$, можна так розбити на три підслова $\omega = \alpha\beta\gamma$, що виконуються наступні умови:*

- а) $|\beta| \geq 1$;
- б) $|\alpha\beta| \leq s$;
- в) для кожного $k \geq 0$ слово $\alpha\beta^k\gamma \in L$, тобто $\alpha\beta^*\gamma \subset L$.

ДОВЕДЕННЯ. Нехай L – регулярна мова, тоді згідно з теоремою 5.1 існує ДСА A , який розпізнає цю мову. Нехай кількість станів автомата A дорівнює s (константа, яка фігурує у формулюванні леми). Якщо $\omega \in L$, то обчислювальний шлях для слова ω починається в початковому стані q_0 і закінчується в кінцевому стані q_f . Цей шлях містить $|\omega|$ стрілок, бо кожна стрілка позначена єдиною буквою слова ω . Таким чином, він містить послідовність з $|\omega| + 1$ станів. Оскільки $|\omega| \geq s$, то згідно принципу Діріхле деякий стан в обчислювальному шляху обов'язково повториться двічі. Нехай $q_{i_k} = q_{i_l}$ – перший такий стан.



Нехай β – та частина слова ω , яка складається з букв, які позначають стрілки підшляху від першого входження стану q_{i_k} до другого його входження в обчислювальний шлях слова ω . Позначимо

через α частину слова ω перед β , а через γ – його частину після β . Очевидно, що $|\beta| \geq 1$ та $|\alpha\beta| \leq s$ (оскільки всі стани до другої появи q_{i_k} різні). Більше того, слово $\alpha\beta^k\gamma$ для будь-якого цілого невід'ємного числа k переводить автомат у той самий кінцевий стан, що й слово $\alpha\beta\gamma$. Справді, частина β^k слова $\alpha\beta^k\gamma$ просто переводить стани автомата A вздовж петлі, яка починається та закінчується в стані q_{i_k} . Ця петля проходить k разів. Звідси випливає, що автомат A розпізнає слова $\alpha\beta^k\gamma$, тому всі такі слова належать мові L . \square

5.7. ПРИКЛАД. Доведемо, що мова $\{0^m 1^m \mid m = 0, 1, \dots\}$ не регулярна.

Припустимо, що мова L є регулярною. Тоді L розпізнається ДСА, який має s станів. Нехай слово $\omega = 0^m 1^m$ і $|\omega| = 2m \geq s$. Тоді за лемою про роздування слова $\omega = 0^m 1^m = \alpha\beta\gamma$ та $\alpha\beta^k\gamma$ належать мові L , причому $\beta \neq e$. Слово β не може містити водночас нулі й одиниці, бо тоді β^2 містило би 10 і $\alpha\beta^2\gamma$ не належало б мові L . Отже, слово β складається або виключно з нулів, або тільки з одиниць, але тоді $\alpha\beta^2\gamma$ містить або забагато нулів, або забагато одиниць і не належить мові L . Ця суперечність доводить, що мова $L = \{0^m 1^m \mid m = 0, 1, \dots\}$ не регулярна.

5.8. ПРИКЛАД. Покажемо, що мова $\{0^p \mid p \text{ — просте число}\}$ не є регулярною.

Припустимо, що $L = \{0^p \mid p \text{ — просте число}\}$ є регулярною мовою. Тоді L розпізнається ДСА. Нехай s – кількість станів автомата. Зафіксуємо просте число $p > s$. Зауважимо, що $0^p \in L$ і $|0^p| = p > s$. Таким чином, за лемою про роздування 0^p можна записати як $0^p = \alpha\beta\gamma$, причому $|\beta| \geq 1$ і $\alpha\beta^*\gamma \subset L$.

Нехай $i = |\alpha| + |\gamma|$ та $j = |\beta|$. Тоді з умови $\alpha\beta^*\gamma \subset L$ випливає, що для кожного $k \geq 0$ слово $\alpha\beta^k\gamma = 0^{i+kj} \in L$, тобто для кожного $k \geq 0$ число $i + kj$ є простим. Зокрема, при $k = 0$ маємо, що i є простим числом. Отже, $i \geq 2$. Поклавши $k = i$, отримаємо, що $i(1+j)$ – просте число. Тим не менше, оскільки $\beta \neq e$, то з $j = |\beta| \geq 1$ випливає, що $i(1+j)$ є складеним. Дана суперечність доводить нерегулярність мови $L = \{0^p \mid p \text{ — просте число}\}$.

5.9. ПРИКЛАД. Доведемо, що мова $L = \{0^m \mid m \text{ — складене число}\}$ не регулярна.

Припустимо, що дана мова $L \in$ регулярною, тоді з наслідку 5.5 випливає, що мови \bar{L} та $\bar{L} \setminus \{0\} = \{0^p \mid p - \text{просте число}\} \in$ регулярними, що суперечить прикладу 5.8.

Рекомендована література : [2, с. 141–147], [4, с. 509–521, 538–543], [16, с. 82–88, 114–123], [17, с. 294–303], [23, с. 53–69].

Питання та вправи до параграфа 5.

5.1. З яким класом формальних мов співпадають мови, що розпізнаються ДСА або НСА?

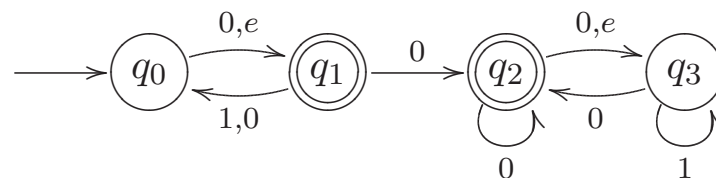
5.2. Дайте означення частки двох формальних мов.

5.3. Чи співпадає клас формальних мов з класом регулярних мов?

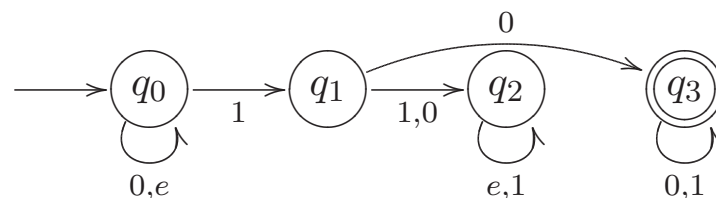
5.4. Сформулюйте лему про роздування для регулярних мов. Для чого її застосовують?

5.5. Побудувати праволінійну граматичку, яка породжує мову, що розпізнається НСА без виходу:

а)



б)



5.6. Побудувати граматичку, яка породжує регулярну мову $L(r)$, задану регулярним виразом $r = ((0 + 11)^*10)^*$.

5.7. Побудувати НСА, який розпізнає мову, породжену граматикою G , що задана множиною продукцій P :

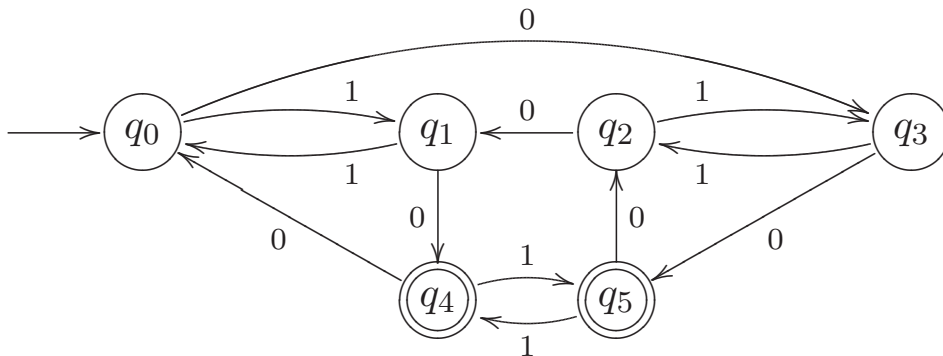
а) $P = \{S \rightarrow bB \mid aA, A \rightarrow a, B \rightarrow b\};$

б) $P = \{S \rightarrow aB \mid aA, A \rightarrow aB \mid ab, B \rightarrow b\};$

в) $P = \{S \rightarrow bA \mid B, A \rightarrow aaA \mid aa, B \rightarrow bB \mid b\}.$

5.8. Побудувати ДСА, який розпізнає мову, породжену граматикою G з попереднього прикладу.

5.9. Використовуючи доведення теореми 5.1, побудувати регулярний вираз, який позначає регулярну мову, що розпізнається автоматом:



5.10. Довести або спростувати твердження:

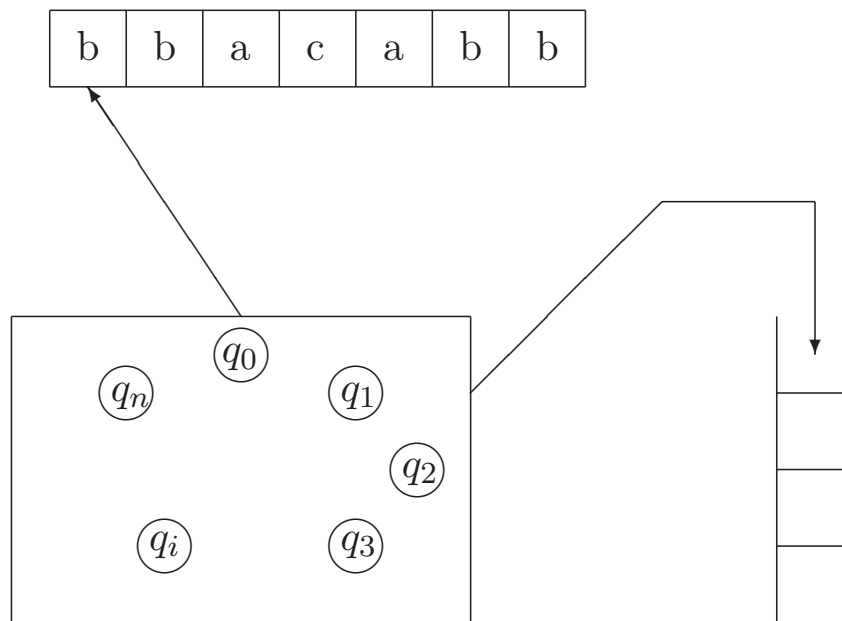
- якщо мова A є регулярною і $A \subset B$, то мова B теж є регулярною;
- якщо мова A є регулярною і $B \subset A$, то мова B – регулярна;
- якщо мова A^2 є регулярною, то мова A є регулярною;
- якщо мови A і AB є регулярними, то мова B також регулярна;
- якщо мови A і B є регулярними, то мова $\cup_{i=0}^{\infty} (A^i \cap B^i)$ теж є регулярною.

5.11. З'ясувати, чи є регулярними наступні формальні мови:

- $\{0^{2n}1^n \mid n \geq 0\}$;
- $\{0^{2^n} \mid n \geq 0\}$;
- $\{0^n1^n2^n \mid n \geq 0\}$;
- $\{0^{2^n} \mid n \geq 0\}$;
- $\{0^{n^2} \mid n \geq 0\}$;
- $\{0^m1^n \mid m \neq n\}$;
- $\{0^{pq} \mid p, q \text{ — прості числа}\}$;
- $\{0^m1^n \mid m, n \geq 0, m = 2n + 1\}$;
- $\{0^m1^n \mid m, n \geq 0, m \neq 3n + 1\}$;
- $\{a^n b^m c^k \mid n, m, k \geq 0, n \neq m \text{ або } m \neq k \text{ або } k \neq n\}$;
- $\{\omega \omega^R \mid \omega \in \{0, 1\}^+\}$.

§ 6. Автомати з магазинною пам'яттю

Автомат з магазинною пам'яттю (МП-автомат) – це скінченний недетермінований автомат без виходу, що має додатковий пристрій пам'яті типу стек з головою, яка виймає і заносить дані у стек. У стеку кожен елемент інформації заноситься в верхню комірку пам'яті, зміщуючи тим самим весь її вміст на одну комірку вниз, і дістається тільки із верхньої комірки пам'яті, зміщуючи весь її вміст на одну комірку вгору. При цьому у кожний момент часу доступний тільки верхній елемент стеку. Теоретично об'єм пам'яті МП-автомата необмежений. Схема МП-автомата має вигляд:



6.1. ОЗНАЧЕННЯ. *Автомат з магазинною пам'яттю* – це шістка $M = (Q, X, Z, \delta, q_0, F)$, де Q , X , q_0 і F визначаються так само, як і в НСА, Z – скінченний магазинний алфавіт (множина допустимих букв пам'яті), а δ – функція переходів:

$$\delta : Q \times (X \cup \{e\}) \times (Z \cup \{e\}) \rightarrow 2^{Q \times (Z \cup \{e\})}.$$

На початку роботи МП-автомат $M = (Q, X, Z, \delta, q_0, F)$ завжди перебуває в початковому стані q_0 , його стек порожній і головка стрічки аналізує першу зліва букву вхідного слова, яке записане на стрічці. Якщо МП-автомат M перебуває в стані q , зчитує з стрічки букву a і виймає зі стеку верхню букву u та $(p, v) \in \delta(q, a, u)$, то

M замінює u на v , головка стрічки переміщується на одну комірку праворуч і керуючий пристрій переводить автомат у стан p . Розглянемо випадки, коли v або u дорівнює e . Якщо $v = e$, то МП-автомат просто видаляє верхню букву u стеку, не виймаючи при цьому наступної букви. У випадку $u = e$ МП-автомат заносить в стек букву v , не зачіпаючи верхньої букви стеку. Команда $(p, v) \in \delta(q, e, u)$ аналогічна e -тактам НСА, тобто МП-автомат M виконує операцію, як і у випадку $(p, v) \in \delta(q, a, u)$, але при цьому не зчитує букви зі стрічки (головка стрічки не зміщується праворуч). Вважається, що МП-автомат M розпізнає вхідне слово, якщо хоча б в одному з обчислювальних шляхів він повністю проаналізовує слово, має порожній стек і зупиняється в кінцевому стані $q \in F$. Через $L(M)$ позначається мова, яка складається зі всіх слів, які розпізнаються автоматом M .

6.2. ПРИКЛАД. Розглянемо МП-автомат

$$M = (\{q, p\}, \{a, b, c\}, \{a, b\}, \delta, q, \{p\}),$$

де функція переходів δ задається так:

$$\begin{aligned} \delta(q, a, e) &= \{(q, a)\}, \quad \delta(p, a, a) = \{(p, e)\}, \quad \delta(q, b, e) = \{(q, b)\}, \\ \delta(p, b, b) &= \{(p, e)\}, \quad \delta(q, c, e) = \{(p, e)\}. \end{aligned}$$

Знайдемо мову $L(M)$.

Цей автомат працює наступним чином. В початковому стані q він заносить букви a та b в стек доти, доки головка стрічки не знайде букву c . Після аналізу букви c він переходить в стан p . В стані p автомат M порівнює кожну букву стрічки з верхньою буквою стеку. Якщо всі відповідні букви стрічки та стеку співпадають, то M розпізнає вхідне слово.

Нехай $\omega \in \{a, b\}^*$ – префікс вхідного слова перед першим входженням букви c і ν – суфікс вхідного слова після першого входження c . Зауважимо, що в той час, коли МП-автомат переходить в стан p , слово ω записане в стеку в оберненому порядку (перша буква слова ω записана вкінці стеку, а остання буква – на верху стеку). Таким чином, коли автомат порівнює суфікс ν з буквами стеку, то спершу порівнюється перша буква слова ν з останньою буквою слова ω і т.д. Отже, вхідне слово розпізнається тільки тоді, коли $\nu = \omega^R$, тобто $L(M) = \{\omega c \omega^R \mid \omega \in \{a, b\}^*\}$.

Щоб краще зрозуміти як автомат працює на вхідному слові, введемо поняття *конфігурації МП-автомата*. Конфігурація – це запис інформації про роботу автомата на певному кроці аналізу вхідного слова, який включає активний стан, букви стрічки, які ще не проаналізовані, і символи, які на даний момент є в стеку. Отже, конфігурація МП-автомата $M = (Q, X, Z, \delta, q_0, F)$ є елементом декартового добутку $Q \times X^* \times Z^*$. Конфігурація (q, ω, γ) означає, що МП-автомат перебуває в стані q , непроаналізований суфікс вхідного слова дорівнює ω (головка стрічки аналізує першу букву слова ω) і стек містить слово γ (перша буква слова γ є верхньою буквою стеку).

Нехай (q, ω_1, β) і (p, ω_2, γ) – дві конфігурації. Кажемо, що *конфігурація (p, ω_2, γ) безпосередньо слідує за (q, ω_1, β)* і записуємо $(q, \omega_1, \beta) \vdash (p, \omega_2, \gamma)$, якщо існує така інструкція $(p, v) \in \delta(q, a, u)$, що $\omega_1 = a\omega_2$ і $\beta = u\alpha$, $\gamma = v\alpha$ для деякого слова $\alpha \in Z^*$, де a, u, v можуть дорівнювати порожньому слову e . Будемо казати, що *конфігурація (p, ω_2, γ) слідує за (q, ω_1, β)* і записувати $(q, \omega_1, \beta) \vdash^* (p, \omega_2, \gamma)$, якщо існує така послідовність конфігурацій C_0, C_1, \dots, C_n , що $C_0 = (q, \omega_1, \beta)$, $C_n = (p, \omega_2, \gamma)$ і $C_i \vdash C_{i+1}$ для всіх $i = 0, 1, \dots, n-1$. Послідовність (C_0, C_1, \dots, C_n) конфігурацій МП-автомата M називається *обчислювальним шляхом автомата M* , якщо C_0 є початковою конфігурацією (q_0, ω, e) , C_n не має наступної конфігурації і $C_i \vdash C_{i+1}$ для всіх $i = 0, 1, \dots, n-1$. Нагадаємо, що в загальному випадку M є недетермінованим автоматом, а тому може бути багато обчислювальних шляхів, які починаються з початкової конфігурації. Крім того, конфігурація (q, ω, β) може не мати наступної конфігурації навіть при $\omega \neq e$. Таким чином, МП-автомат M розпізнає слово ω тоді і лише тоді, коли існує обчислювальний шлях автомата M , який починається з початкової конфігурації (q_0, ω, e) і закінчується конфігурацією (f, e, e) для деякого кінцевого стану $f \in F$, тобто

$$L(M) = \{\omega \in X^* \mid (\exists f \in F)[(q_0, \omega, e) \vdash^* (f, e, e)]\}.$$

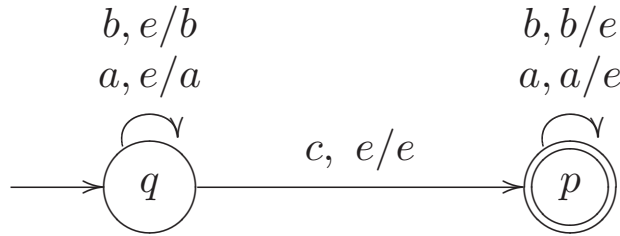
Наприклад, розглянемо три обчислювальні шляхи МП-автомата M з прикладу 6.2 для слів $abcba$, $abcb$ і $abcbab$ відповідно:

$$\begin{aligned} (q, abcba, e) &\vdash (q, bcba, a) \vdash (q, cba, ba) \vdash (p, ba, ba) \vdash (p, a, a) \vdash (p, e, e); \\ (q, abcb, e) &\vdash (q, bcb, a) \vdash (q, cb, ba) \vdash (p, b, ba) \vdash (p, e, a); \end{aligned}$$

$(q, abcbab, e) \vdash (q, bcbab, a) \vdash (q, cbab, ba) \vdash (p, bab, ba) \vdash (p, ab, a) \vdash (p, b, e)$.

Зауважимо, що з допомогою другого обчислювального шляху слово $abcb$ не розпізнається автоматом M , бо стек не стає порожнім після того, як слово проаналізується автоматом. Третій шлях теж не розпізнає вхідного слова, оскільки після (p, b, e) не існує наступної конфігурації, а вхідне слово ще не проаналізоване повністю.

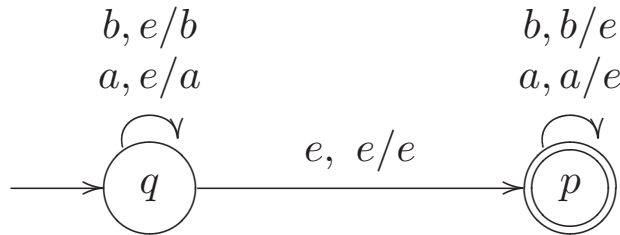
Аналогічно як і НСА, МП-автомат зображується графом. Вершини графа є станами автомата, а кожна стрілка з вершини q в вершину p з позначкою „ $a, u/v$ ” зображує перехід $(p, v) \in \delta(q, a, u)$. Наприклад, граф автомата з прикладу 6.2 має вигляд:



6.3. ПРИКЛАД. Побудувати МП-автомат, який розпізнає мову

$$\{\omega\omega^R \mid \omega \in \{a, b\}^*\}.$$

Цей приклад аналогічний до попереднього, за винятком того, що вхідне слово не містить всередині букви c і тому не зрозуміло, коли починати порівняння між буквами вхідного слова і буквами стеку. Як же вирішити цю проблему? Нагадаємо, що МП-автомат є НСА, а тому можна в будь-який час починати порівняння. Вхідне слово аналізується доти, доки вхідні букви, що залишилися, зіставляються коректно з буквами стеку по відношенню до деякої точки поділу. Звідси слідує, що шуканий МП-автомат задається графом:



Побудуємо три обчислювальні шляхи для слова $abba$, перший з яких розпізнає дане слово, а два інші – ні:

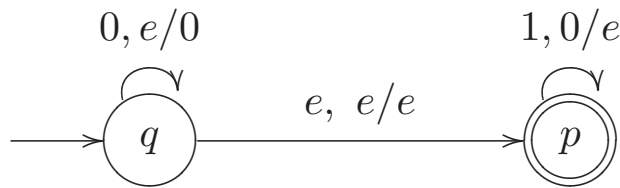
$(q, abba, e) \vdash (q, bba, a) \vdash (q, ba, ba) \vdash (p, ba, ba) \vdash (p, a, a) \vdash (p, e, e);$
 $(q, abba, e) \vdash (q, bba, a) \vdash (q, ba, ba) \vdash (q, a, bba) \vdash (p, a, bba);$

$$(q, abba, e) \vdash (q, bba, a) \vdash (p, bba, a).$$

6.4. ПРИКЛАД. Побудувати МП-автомат, який розпізнає мову

$$\{0^n 1^n \mid n = 0, 1, \dots\}.$$

Користуючись тим же методом, що і в прикладі 6.3, побудуємо граф шуканого автомата:



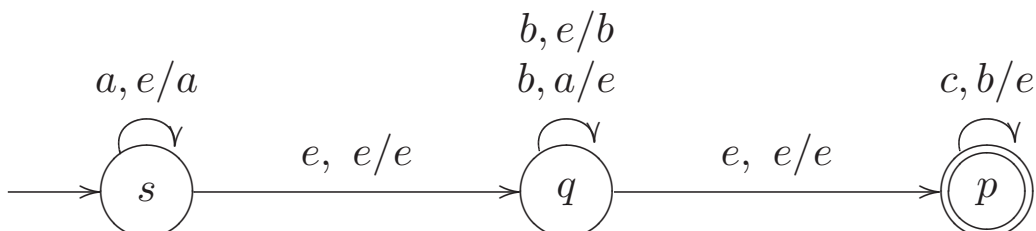
Даний автомат зчитує у пам'ять першу половину вхідного слова, що складається з нулів, а потім видаляє з пам'яті по одному нулю на кожну одиницю, що поступає на вхід. Крім того, переходи станів гарантують, що всі нулі передують всім одиницям.

З прикладів 5.7 і 6.4 випливає, що клас мов, які розпізнаються МП-автоматами, ширший від класу регулярних мов. Дані мови називаються *контекстно-вільними*.

6.5. ПРИКЛАД. Побудувати МП-автомат, який розпізнає мову

$$\{a^i b^j c^k \mid i, j, k \geq 0, i + k = j\}.$$

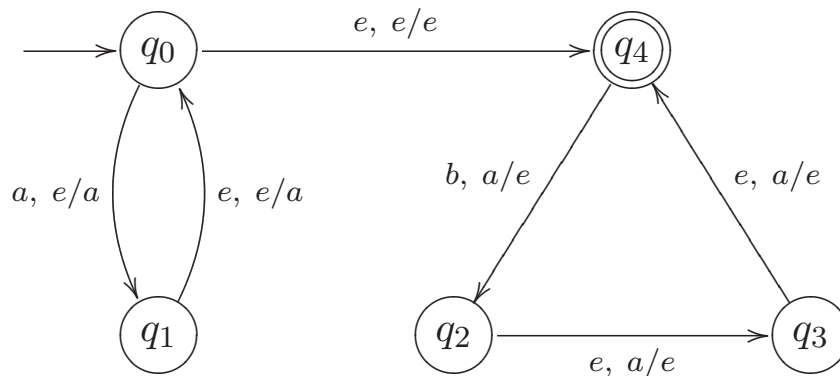
Основна ідея для розв'язання цієї задачі полягає у використанні різних станів для того, щоб зберігати правильний порядок входжень букв a , b і c , а також у використанні стеку для дотримання виконання рівності $i + k = j$. Таким чином, побудуємо МП-автомат з трьома станами s , q і p , де s – початковий стан, а p – єдиний кінцевий стан. В стані s автомат кожну букву a заносить у стек. В стані q він спершу взаємно однозначно співставляє букви b з буквами a зі стеку, а потім заносить решту букв b у стек. В стані p автомат аналізує букви c і співставляє їх з буквами b зі стеку.



6.6. ПРИКЛАД. Побудувати МП-автомат, який розпізнає мову

$$L = \{a^i b^j \mid 2i = 3j\}.$$

Для того, щоб слово належало мові L , мають виконуватись рівності $i = 3k$ та $j = 2k$ для деякого цілого $k \geq 0$. Таким чином, для кожної букви a вхідного слова потрібно занести дві її копії в стек. Далі кожну букву b вхідного слова треба співставити з трьома буквами a стеку. Як же занести дві букви a у стек? Можна занести одну букву a у стек, а потім перейти в допоміжний стан для занесення другої букви a . Аналогічно спершу порівнюємо кожну букву b вхідного слова з однією буквою a , а потім переходимо в два інші допоміжні стани для видалення двох зайвих букв a зі стеку. Граф МП-автомата має вигляд:

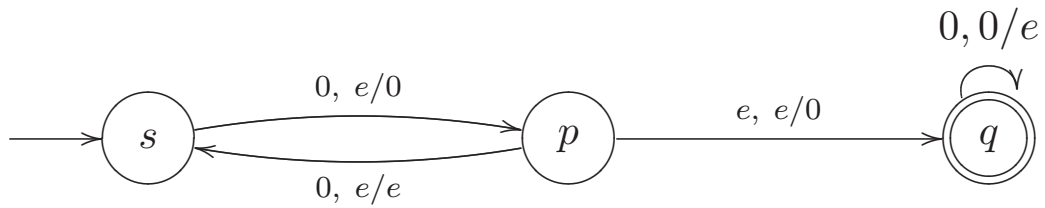


Рекомендована література : [2, с. 192–220], [5, с. 399–402], [12, с. 374–382], [16, с. 128–136].

Питання та вправи до параграфа 6.

- 6.1. Дайте означення автомата з магазинною пам'яттю.
- 6.2. Чим відрізняється МП-автомат від НСА?
- 6.3. Що ви розумієте під стеком?
- 6.4. Обґрунтуйте, чому клас мов, що розпізнаються МП-автоматами, ширший від класу регулярних мов.
- 6.5. Як називаються мови, які розпізнаються МП-автоматами.

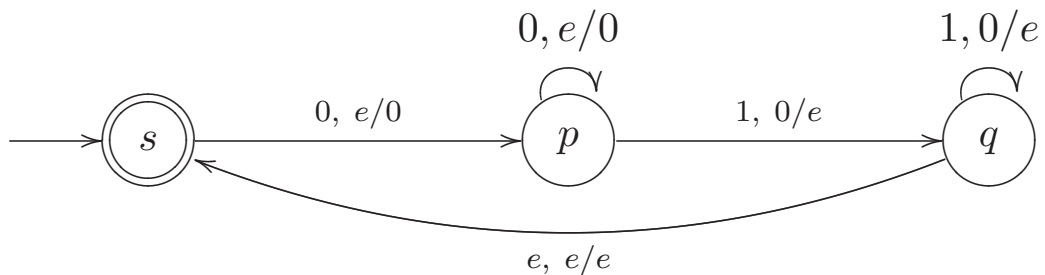
6.6. Скінченний МП-автомат M заданий графом:



Задати його як шістку $M = (Q, X, Z, \delta, q_0, F)$ і знайти мову, яка розпізнається МП-автоматом.

6.7. Чи є мова, що розпізнається МП-автоматом з попереднього прикладу, регулярною? Якщо так, то побудувати рівносильний ДСА.

6.8. Знайти мову $L(M)$, яка розпізнається МП-автоматом M , заданим графом:



Чи можна побудувати праволінійну граматичку, яка породжує мову $L(M)$?

6.9. Побудувати МП-автомат, який розпізнає мову:

- $\{0^n 1^m 2^n \mid n, m \geq 0\}$;
- $\{a^i b^j c^k \mid i, j, k \geq 0, i + j = k\}$;
- $\{a^i b^j c^k \mid i, j, k \geq 0, i = j + k\}$;
- $\{a^i b^j c^k \mid i, j, k \geq 0, i + 2k = j\}$;
- $\{0^n 1^{3n} \mid n \geq 0\}$;
- $\{0^n 1^m \mid n, m \geq 0, 2n = 5m\}$.

Список літератури

1. Алфєрова З.В. Теория алгоритмов / З.В. Алфєрова – М.: «Статистика», 1973. – 164 с.
2. Ахо А. Теория синтаксического анализа, перевода и компиляции / А. Ахо, Дж. Ульман. – М.: Мир, 1978. – Т. 1. – 611 с.
3. Безущак О.О. Елементи теорії чисел: Навчальний посібник / О.О. Безущак, О.Г. Ганюшкін. – К.: Видавничо-поліграфічний центр «Київський університет», 2003. – 202 с.
4. Белоусов А.И. Дискретная математика: Учеб. для вузов / А.И. Белоусов, С.Б. Ткачев. – 3-е изд. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2004. – 744 с.
5. Бондаренко М.Ф. Комп'ютерна дискретна математика: підручник / М.Ф. Бондаренко, Н.В. Білоус, А.Г. Руткас. – Харків: «Компанія СМІТ», 2004. – 480 с.
6. Гаврилків В.М. Формальні мови та алгоритмічні моделі / В.М. Гаврилків. – Івано-Франківськ: «Сімик», 2012. – 172 с.
7. Гаврилків В.М. Регулярні вирази у програмних продуктах: навчальний посібник / В.М. Гаврилків. – Івано-Франківськ: «Сімик», 2012. – 72 с.
8. Гаврилов Г.П. Задачи и упражнения по дискретной математике: Учеб. пособие / Г.П. Гаврилов, А.А. Сапоженко. – 3-е изд., перераб. – М.: ФИЗМАТЛИТ, 2005. – 416 с.
9. Гашков С.Б. Современная элементарная алгебра в задачах и решениях / С.Б. Гашков. – М.: МЦНМО, 2006. – 328 с.
10. Завало С.Т. Алгебра і теорія чисел, ч. 2 / С.Т. Завало, В.М. Костарчук, Б.І. Хацет. – К.: Вища школа, 1976. – 384 с.
11. Игошин В.И. Математическая логика и теория алгоритмов: учеб. пособие / В.И. Игошин. – М.: Изд. центр «Академия», 2008. – 448 с.

12. Капітонова Ю.В. Основи дискретної математики / Ю.В. Капітонова, С.Л. Кривий, О.А. Летичевський, Г.М. Луцький, М.К. Печурін. – К.: Наукова думка, 2002. – 580 с.
13. Кривий С.Л. Дискретна математика: Вибрані питання / С.Л. Кривий. – К.: Вид. дім «Києво-Могилянська академія», 2007. – 572 с.
14. Мальцев А.И. Алгоритмы и рекурсивные функции / А.И. Мальцев – М.: Наука, 1986. – 368 с.
15. Марков А.А. Теория алгоритмов / А.А. Марков, Н.М. Нагорный. – М.: Наука, 1984. – 432 с.
16. Мозговой М.В. Классика программирования: алгоритмы, языки, автоматы, компиляторы. Практический подход / М.В. Мозговой. – СПб.: Наука и Техника, 2006. – 320 с.
17. Нікольський Ю.В. Дискретна математика / Ю.В. Нікольський, В.В. Пасічник, Ю.М. Щербина. – К.: Видавнича група ВНУ, 2007. – 368 с.
18. Новиков Ф.А. Дискретная математика для программистов / Ф.А. Новиков. – СПб.: Питер, 2001. – 304 с.
19. Оре. О. Приглашение в теорию чисел: Пер с англ. / О. Оре. – Изд. 2-е, стереотипное. – М.: Едиториал УРСС, 2003. – 128 с.
20. Пильщиков В.Н. Машина Тьюринга и алгоритмы Маркова. Решение задач / В.Н. Пильщиков, В.Г. Абрамов, А.А. Вылиток, И.В. Горячая. – М.: МГУ, 2006. – 47 с.
21. Самохин А.В. Математическая логика и теория алгоритмов / А.В. Самохин. – Москва, 2003. – 237 с.
22. Тишин В.В. Дискретная математика в примерах и задачах / В.В. Тишин – СПб.: БХВ-Петербург, 2008. – 352 с.
23. Ding-Zhu Du. Problem Solving in Automata, Languages, and Complexity / Ding-Zhu Du, Ker-I Ko. – New York: WIP, 2001. – 388 p.
24. Salomaa A. Formal Languages / A. Salomaa. – New York: Academic Press, 1973. – 281 p.

Показчик

- e-замикання, 62
- ітерація, 6
- автомат, 40
 - Мілі, 42
 - Мура, 44
 - автономний, 43
 - без виходів, 43
 - без пам'яті, 43
 - детермінований, 43
 - з магазинною пам'яттю, 85
 - недетермінований, 43
 - нескінченний, 43
 - повний, 43
 - рівносильний, 67
 - скінченний, 43
 - частковий, 43
- алфавіт
 - вихідний, 40, 42
 - вхідний, 40, 42
- бінарна операція, 4
- буква, 4
 - вихідна, 40
 - вхідна, 40
- виведення, 34
 - ліве, 36
 - праве, 36
- голівка
 - автомата, 41
- гомоморфізм, 4
- граматика, 30
 - контекстно вільна, 33
 - контекстно залежна, 33
 - ліволінійна, 34
 - неоднозначна, 36
 - праволінійна, 33
 - регулярна, 34
 - типу 0, 33
 - типу 1, 33
 - типу 2, 33
 - типу 3, 33
- граф
 - регулярного виразу, 26
 - автомата, 45
- групоїд, 4
- дерево виведення, 35
- довжина слова, 5
- замикання Кліні, 6
- керуючий пристрій
 - автомата, 41
- конкатенація, 5
- конструкція підмножин, 68
- конфігурація
 - МП-автомата, 87
- лема
 - Ардена, 7
 - про роздування, 81
- напівгрупа, 4
 - вільна, 5
- обчислюваний шлях, 87
- підслово, 5
- префікс, 5
- продукція, 31
- регулярний вираз, 23
- розряд числа, 11
- символ, 4
 - нетермінальний, 31

- початковий, 31
- термінальний, 31
- система числення, 10
 - g -кова, 12
 - дуодецимальна, 13
 - змішана, 11, 13
 - майя, 13
 - непозиційна, 11
 - позиційна, 11
 - римська, 12
 - унарна, 11
- слово, 5
 - вихідне, 40
 - вхідне, 40
 - не розпізнається автоматом, 42
 - розпізнається автоматом, 42
- стан
 - досяжний, 48
 - кінцевий, 42
- стек, 85
- стрічка
 - автомата, 41
- суфікс, 5
- таблиця автомата, 44
- такт, 40
 - ϵ -такт, 60
- формальна мова, 6
 - має суфіксну властивість, 6
 - контекстно вільна, 33
 - має префіксну властивість, 6
 - обернена, 7
 - породжується автоматом, 43
 - регулярна, 23
 - розпізнається автоматом, 42
- формальний алфавіт, 4
 - бінарний, 4
- функція
 - автоматна, 42
 - виходів, 41
 - переходів, 41
- цифра, 12