Міністерство освіти і науки України
Донбаська державна машинобудівна академія (ДДМА)

М. П. Богдан, Л. В. Васильєва

ДИСКРЕТНА МАТЕМАТИКА

Навчальний посібник

для студентів закладів вищої освіти

Затверджено
на засіданні Вченої ради ДДМА
Протокол № 2 від 26.09.2019 р.

Краматорськ
ДДМА
2019

УДК 519.1 : 004
Б 73

Рецензенти:

Гончаров О. А., д-р фіз.-мат. наук, професор кафедри прикладної математики та моделювання складних систем Сумського державного університету;

Рудаков О. М., канд. екон. наук, доцент, проректор з науково-педагогічної роботи, ДІТМ Міжнародного науково-технічного університету.

Богдан, М. П.

Б 73 Дискретна математика : навчальний посібник для студентів закладів вищої освіти/ М. П. Богдан, Л. В. Васильєва. – Краматорськ : ДДМА, 2019. – 80 с.

ISBN 978-966-379-902-5

Посібник призначений для студентів молодших курсів галузі знань 12 «Інформаційні технології» та містить матеріал з основних розділів курсу «Дискретна математика» з урахуванням специфіки галузі: елементи теорії множин та відношень, алгебри логіки, теорії графів, теорії скінченних автоматів – розпізнавачів та автоматів – трансляторів, теорія граматик; наприкінці кожного розділу приведені задачі та приклади для самостійного розв’язання.

Може бути використаний студентами інших спеціальностей, аспірантами та іншими зацікавленими особами.

УДК 519.1 : 004

ISBN 978-966-379-902-5

ЗМІСТ

ВСТУП

Дискретна математика – одна з важливих складових сучасної математики. З одного боку, вона містить фундаментальні основи математики – теорію множин, математичну логіку, теорію алгоритмів; з іншого боку – є основним математичним апаратом інформатики та обчислювальної техніки.

Загальна комп'ютеризація всіх галузей нашої діяльності й життя взагалі призводить до постійного зростання попиту, як на програмістів, так і на фахівців, які розробляють математичні основи комп'ютерних технологій. Без перебільшення можна констатувати: теоретичною основою будь-якого програмного забезпечення для сучасних комп'ютерів є дискретна математика.

Дискретна математика – фундаментальна наука, що потрібна фахівцям в області інформаційних технологій як метод мислення, як засіб формулювання та організації понять, при побудові математичних моделей. Дисципліна «Дискретна математика» відноситься до циклу дисциплін професійної підготовки і є складовою підготовки студентів за спеціальністю «Комп’ютерні науки та інформаційні технології».

Знання, отримані при вивченні цієї дисципліни, забезпечують вивчення наступних дисциплін: «Теорія алгоритмів»; «Об'єктно-орієнтоване програмування»; «Технології захисту інформації»; «Системний аналіз»; «Організація баз даних і знань»; «Теорія прийняття рішень»; «Математичні методи дослідження операцій».

Навчальний посібник ставить за мету познайомити студента з максимально широким колом понять дискретної математики, придбати навички в створенні та програмуванні дискретних об’єктів при вирішенні практичних задач.

1 ЕЛЕМЕНТИ ТЕОРІЇ МНОЖИН ТА ВІДНОШЕНЬ

1.1 Множини. Способи задавання множин

Мова множин – універсальна мова математики. Будь-яке математичне твердження можна сформулювати як твердження про деяке співвідношення між множинами: про рівність двох множин, про непустоту деякої множини, про не приналежність елементам множини. Поняття «множина» – одне з базових понять в математиці та не може бути визначене через інші поняття. Інтуїтивно множину можна визначити як сукупність предметів, понять, явищ, множин, ..., об'єднаних одним або декількома властивостями. У множині не може бути однакових елементів. Порядок слідування елементів у множині не важливий.

Множини, підмножини будемо позначати великими буквами латинського алфавіту $(A, B, C, D\ldots)$, а елементи множин – малими $(a, b, c, d\ldots)$.

Множина B називається підмножиною A, якщо будь-який елемент B є елементом A. Цей факт можна записати в такий спосіб: $B \subset A$.

Множини можуть бути скінченними (складатися зі скінченної кількості елементів) та нескінченними. Приклади нескінченних множин – множина натуральних чисел $N = \{1, 2, 3, \ldots\}$, множина натуральних чисел із включенням нуля $N_0 = \{0, 1, 2, 3, \ldots\}$. Приклади скінченних множин будуть наведені нижче.

Кількість елементів у скінченній множині M називається потужністю множини та позначається $|M|$ або $n(M)$. Множина потужністю 0, тобто множина, що не містить елементів, називається порожньою та позначається так: $M = \{\ \}$ або $M = 0$. Прийнято вважати, що порожня множина є підмножиною будь-якої множини, у тому числі й порожньої.

Множина може задаватися:

1. Списком (перерахуванням) елементів: $A = \{a, b, c, d\}$; $S = \{Іванов, Петров, Сидоров\}$.

2. Породжуючою процедурою:

$M = \{(x, y) \mid x^2 + y^2 = 1\}$ (задане коло радіуса $R = 1$);

$K = \{(a, b) \mid a \in A \quad та \quad b \in B\}$ (задано добуток двох множин);

$D = A \cup B = \{x \mid x \in A \quad або \quad x \in B\}$ (задане об'єднання двох множин).

3. Описом характеристичних властивостей, які повинні мати елементи множини:

- усі студенти ДДМА;
- футбольна команда «Шахтар»;
- мешканці міста Краматорськ.

1.2 Операції над множинами

Для наочного представлення формул цих операцій використовують діаграму Вена – (англ. *Venn diagram*) — діаграму, що показує всі можливі логічні відношення для скінченного набору множин. Для зображення множин використовують також кола Ейлера. За їхньою допомогою можливе зображення всіх можливих відношень між різними множинами, у тому числі й таких, коли одна множина містить іншу або взагалі відсутні перетини множин. При цьому вводять поняття універсальної множини (універсуму). Універсальна множина – це така множина *W* (може також зустрічатися позначення *U* або *I)*, для якої перетин цієї множини з будь-якою множиною *X* збігається із цією множиною *X*. Універсальна множина єдина.

Універсальна множина на діаграмі Вена в загальному виді – це прямокутник із колами Ейлера всередині. Одним зі способів спрощення формул при виконанні вправ та задач є послідовна побудова діаграм Вена, що відповідають виконанню кожної операції у формулі, з метою визначення того набору областей, яким відповідає формула, а потім їхнього опису більш простою формулою, якщо це можливо.

Об'єднанням множин називається множина, що складається з усіх елементів, які належать хоча б одній із цих множин: $A \cup B = \{x \mid x \in A \quad або \quad x \in B\}$ (рис. 1.1).

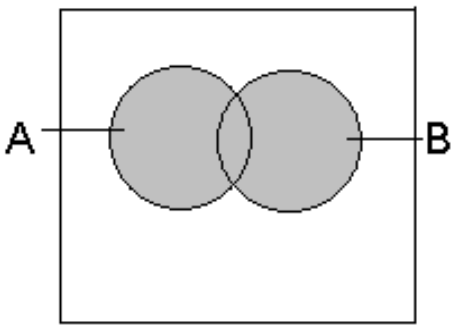


Рисунок 1.1 – Операція об'єднання двох множин $A \cup B$

Операція «об'єднання» n-арна (тобто може бути застосована до n множин) та комутативна (при перестановці поєднуваних множин результат виконання операції не зміниться).

Приклад: для заданих множин A, B та C знайти об'єднання цих множин:

$$A=\{a, ab, ac, c\};\ B=\{ac, ba, b, c\};\ C=\{b,\ f\};$$
$$D=A\cup B\cup C=C\cup B\cup A=\{a, ab, ac, c, ba, b,\ f\}.$$

Перетином множин називається множина, що складається з усіх елементів, що належать одночасно кожній із множин, що перетинаються: $A\cap B=\{x \mid x\in A \quad та \quad x\notin B\}$ (рис. 1.2).

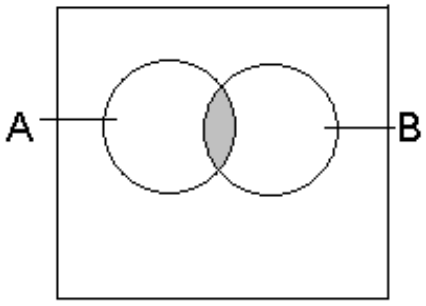


Рисунок 1.2 – Операція перетину двох множин $A\cap B$

Операція «перетин» n-арна (тобто може бути застосована до n множин) та комутативна (при перестановці множин, що перетинаються, результат виконання операції не зміниться).

Приклад: для заданих множин A, B та C знайти перетин цих множин:

$$A=\{a, b, c,\ f\};\ B=\{b, c, n\};\ C=\{d,\ f, n\};$$
$$D=A\cap B=\{b, c\};\ B\cap C=\{n\};$$
$$E=A\cap B\cap C=B\cap C\cap A=\{\ \}.$$

Різницею двох множин називається множина, що складається з усіх елементів, що належать першій множині та не належать другій множині: $A\setminus B=\{x \mid x\in A\, та\, x\notin B\}$.

Операція «різниця» бінарна (тобто може бути застосована тільки до двох множин), некомутативна (при перестановці множин результат змінюється): $A\setminus B\neq B\setminus A$ (рис. 1.3).

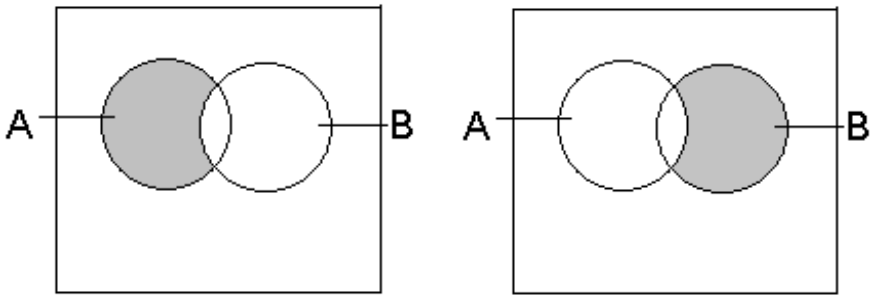


Рисунок 1.3 – Операція різниці двох множин: а) $A \setminus B$; б) $B \setminus A$.

Приклад: для заданих множин A, B та C знайти різницю цих множин:

$$A=\{a,c,d,e,f\};\ B=\{a,d,m\};\ D=A\setminus B=\{c,e,f\};\ C=B\setminus A=\{m\}.$$

Симетрична різниця двох множин має кілька визначень:

1) це множина $A\Delta B$, куди входять усі ті елементи першої множини, які не входять у другу множину, а також ті елементи другої множини, які не входять у першу множину;

2) це множина $A\Delta B$, куди входять усі ті елементи обох множин, які не є загальними для двох заданих множин.

Формула:

$$A\Delta B=\{x \mid x\notin A\ та\ x\notin B\}.$$

На рис. 1.4 наведено приклад симетричної різниці двох множин $A\Delta B$.

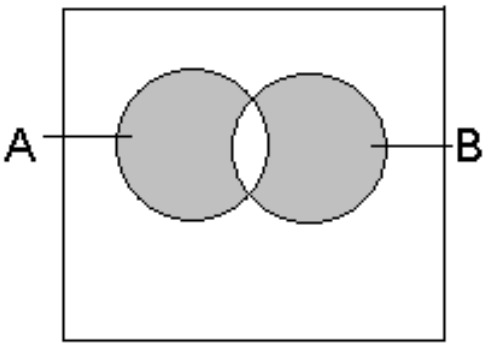


Рисунок 1.4 – Операція симетричної різниці двох множин $A\Delta B$

Операція «симетрична різниця» бінарна (тобто може бути застосована тільки до двох множин), комутативна (при перестановці множин результат не змінюється): $A \Delta B = B \Delta A$.

Приклад: для заданих множин A, B та C знайти симетричну різницю цих множин:

$$A = \{a, c, d, e, f\}; B = \{a, d, m\};$$
$$D = A \Delta B = \{c, e, f, m\};$$
$$C = B \Delta A = \{c, e, f, m\}.$$

Примітка: у літературі зустрічаються різні позначення для цієї операції, а саме: $\setminus$; $-$; $\div$; $\oplus$; $\circ$. Усі вони позначають одну й ту саму операцію.

Доповненням множини A (до універсальної множини W) називається множина, що складається з усіх елементів множини W, які не входять у множину A: $\overline{A} = \{x \mid x \in W \; та \; x \notin A\}$ (рис. 1.5).

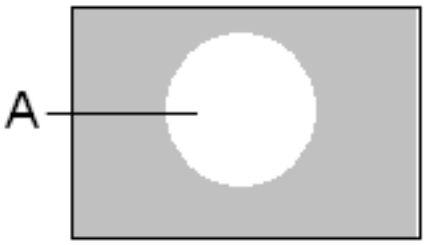


Рисунок 1.5 – Операція доповнення множини $\overline{A}$

Операція «доповнення» унарна (тобто може бути застосована тільки до однієї множини або частини формули, яку можна трактувати як одну множину). Операцію «різниця» у формулах можна замінити так: $A \setminus B = A \cap \overline{B}$.

Приклад: для заданих множин A, B та W знайти доповнення:

$$W = \{a, c, d, e, f, k\}; B = \{c, e, f\}; A = \{a, c\}; B \cup \overline{B} = W; B \cap \overline{B} = \{\} (порожньо);$$
$$D = A \setminus B = A \cap \overline{B} = \{c\}.$$

Використовуючи множини, операції об’єднання, перетину, різниці, доповнення та дужки, які міняють порядок виконання операцій, можна створювати формули. Найбільший пріоритет при виконанні має операція «доповнення», потім «різниця» і потім дві операції одного пріоритету –

«об'єднання» та «перетин»; операції одного пріоритету виконуються у формулі один за одним зліва направо; якщо у формулі є дужки, тоді спочатку виконуються операції в дужках відповідно до їхнього пріоритету; якщо знак доповнення стоїть над частиною формули, тоді вважають, що та частина взята в дужки (хоча дужки можуть і не стояти). У задачах у формулах частіше використовують тільки три підмножини: A, B, C або P, Q, R універсальної множини W.

Прямим, або декартовим добутком множин називається множина, елементами якої є вектори, складені з елементів множин, що перемножують: перші компоненти векторів – елементи першої множини, другі – другої і т. д. Для двох множин процедура має вигляд: $A \times B = \{(x, y) \mid x \in A \text{ та } y \in B\}$.

Приклад: для заданих множин A, B знайти прямий добуток цих множин:

$$A = \{a, b, c, \ldots, h\};\ B = \{1, 2, 3, \ldots, 8\};\ D = A \times B = \{(a, 1); (a, 2); (a, 3); \ldots, (h, 8)\}.$$

Допускається запис: $D = A \times B = \{a1; a2; a3; \ldots; h8\}$.

Послідовності, отримані таким чином, часто називають векторами (кортежами, упорядкованими наборами).

Вектор – упорядкований набір елементів. Елементи, що утворюють вектор, називаються координатами, або компонентами вектора. Кількість координат називається довжиною або розмірністю вектора. На відміну від елементів множини координати вектора можуть бути однаковими:

$B = (0, 3, 5, 3)$ – вектор розмірності 4;

$C = (0, 3, 3, 5)$ – вектор розмірності 4.

Вектори B та C різні, тому що порядок координат різний.

Операція «добуток» багатомісна (перемножити можна кілька множин) та некомутативна. У якості співмножників можуть виступати ті самі множини, тобто множину можна зводити в n-й ступінь.

Розглянемо випадок, коли елементами множини є символи, наприклад: $V = \{a, b, c\}$. Таку множину будемо називати *алфавітом*. При піднесенні цієї множини у квадрат одержимо нову множину, що складається з двохсимвольних *ланцюжків*, або слів (елементи-вектори записуються без дужок та поділяючих ком): $V^2 = V \times V = \{aa, ab, ac, ba, \ldots, cc\}$. При піднесенні алфавіту в куб (останню множину домножуємо на V) одержимо трьохсимвольні слова-ланцюжки й т. д. Алфавіт у нульовому ступені є множиною,

що складається з одного елемента – порожнього ланцюжка (позначення ε – епсилон): $V^0 = \{\varepsilon\}$.

Довжина ланцюжка дорівнює кількості елементів, що утворюють цей ланцюжок. Порожній ланцюжок ε можна вставити в будь-яке місце інших ланцюжків, не змінюючи їх:

$$|a| = 1; |aba| = 3; |ab\varepsilon a| = 3; |\varepsilon| = 0.$$

1.3 Дії з ланцюжками

Для ланцюжків припустимі наступні дії:

– конкатенація (зчеплення) ланцюжків:

$$x = aba;\ y = cab;\ xy = abacab;$$

– піднесення ланцюжків у ступінь:

$$x = ab;\ x^1 = ab;\ x^2 = abab;\ x^3 = ababab;$$

будь-який ланцюжок у нульовому степені дорівнює ε: $x^0 = \varepsilon$.

Не можна ототожнювати порожню множину $C = \{\ \}$ та множину, що містить один елемент – порожній ланцюжок $B = \{\varepsilon\}$.

Усю множина ланцюжків, що можуть бути побудовані в заданому алфавіті, можна представити таким поняттям, як ітерація алфавіту.

Ітерація – множина, отримана в результаті об'єднання всіх степенів алфавіту, включаючи й нульову: $V^* = \bigcup_{i \in N_0} V^i$.

Укорочена ітерація (позначається V^+) не включає нульовий степінь алфавіту, тобто порожній ланцюжок: $V^+ = \bigcup_{i \in N} V^i$.

Ітерацію та укорочену ітерацію пов'язує наступна формула: $V^+ = V \times V^* = V^* \times V$.

1.4 Кількість елементів множини

Для будь-якої скінченної множини M кількість елементів (потужність множини) будемо позначати $n(M)$.

Нехай задано кілька множин (підмножин однієї універсальної множини W): A, B, C,... з кількістю елементів у кожному відповідно: $n(A)$, $n(B)$, $n(C)$,... . Вирішимо задачу про кількість елементів у множині, представленій у вигляді формули, яка складається з декількох множин, пов'язаних операціями перетин, об'єднання та доповнення.

Дано: $A, B, n(A), n(B)$.

Визначити: кількість елементів в об'єднанні $n(A \cup B)$.

Для множин, які не мають спільних елементів, кількість елементів їхнього об'єднання дорівнює сумі елементів у кожній із поєднуваних множин:

$$n(A \cup B) = n(A) + n(B).$$

Загальний випадок (дві множини мають спільні елементи):

$$n(A \cup B) = n(A) + n(B) - n(A \cup B).$$

Загальний випадок (усі три множини, а також по дві попарно мають спільні елементи):

$$n(A \cup B \cup C) = n(A) + n(B) + n(C) - n(A \cap B) - n(A \cap C) - n(C \cap B) + \\ + n(A \cap B \cap C).$$

1.5 Відношення

Підмножина $R \subset M^n$ називається n-місним (n-арним) відношенням на базовій множині M. Множина M є базовою для відношень будь-якої арності, які на ній побудовані. Говорять, що елементи вектора $a_1, a_2, a_3, \ldots, a_n$ знаходяться у відношенні R, якщо цей вектор належить множині R. Для $n = 1$ відношення називають унарним (по суті, таке відношення виділяє з множини M підмножину R за ознакою); для $n = 2$ – бінарним (тобто відношенням між парами елементів множини M) і т. ін. Від-

ношення – також множина, елементами якої є вектори розмірності n або, при іншому записі, ланцюжки довжиною n, складені в алфавіті M та відібрані відповідно до відношення R.

Приклад: побудувати бінарне відношення R, що визначається словами: «у латинському алфавіті зустрічається раніше» на базовій множині $M=\{a,b,c,d\}$.

Рішення: прикладами елементів відношення R можуть бути вектори $(a,c),(c,d)\ldots$ або ланцюжки $ac, cd, bd\ldots,$ такі, у яких на першому місці стоїть буква, що зустрічається в латинському алфавіті раніше в порівнянні з буквою, що стоїть на другому місці.

Відношення R є підмножиною множини M^2:

$M^2=\{aa,ab,ac,\ldots,dd\}$, з якого елементи відбираються у відповідності з наступною процедурою $R=\{xy \mid x\ "менше"\ y\}$.

$R=\{ab,ac,ad,bc,bd,cd\}$.

Відношення будь-якої арності можна задати одним зі способів задавання множин (перерахування елементів, породжуюча процедура, характеристичні ознаки). Крім цього, бінарні відношення можна задавати:

– за допомогою матриці суміжності – квадратної матриці, стовпці та рядки якої позначені елементами несучої множини, а елементи мають наступні значення: $C_{ij}=\begin{cases}1, \text{ якщо } \ldots a_i \ldots R \ldots a_j;\\ 0, \text{ у противному випадку};\end{cases}$

– за допомогою орієнтованого графа – елементи базової множини M зображуються на площині у вигляді вершин графа (точки з позначенням поруч елементів базової множини), а потім вершини, пари яких входять у множину R, з'єднуються за допомогою стрілок (дуг) (починається стрілка в першому елементі пари, закінчується – у другому); кількість таких стрілок дорівнює кількості елементів у множині R.

Для кожного бінарного відношення R можна побудувати обернене відношення R^{-1} (читається: R у ступеню мінус один), помінявши місцями в кожному елементі R проекції векторів.

Відношення Q обернене до відношення R тоді й тільки тоді, коли для кожної пари з R виконується умова: з xRy *випливає* yQx.

1.6 Властивості бінарних відношень

Відношення R *рефлексивне* (має властивість рефлексивності), якщо для будь-якого елемента $x \in M$ установлене відношення xRx (на головній діагоналі матриці суміжності – одиниці); відношення R *антирефлексивне*, якщо ні для одного елемента x не встановлено такого відношення (на головній діагоналі матриці суміжності – нулі); в інших випадках (на головній діагоналі є й нулі та одиниці) говорять «відношення R *не рефлексивне*».

Таким чином, при встановленні цієї властивості відношення можливі наступні варіанти: рефлексивне, не рефлексивне, антирефлексивне.

Відношення R називається *симетричним*, якщо для будь-якого елемента цього відношення виду (x, y) у множині R є відповідна пара – (y, x). Інакше кажучи, якщо відношення R симетричне, то для кожної пари елементів базової множини це відношення або встановлене в обидві сторони, або не встановлене взагалі; відношення R називається *антисиметричним*, якщо наведена вище умова виконується тільки для випадків, коли $x = y$; відношення R називають *несиметричним* в інших випадках. Таким чином, при встановленні цієї властивості відношення можливі наступні варіанти: симетричне, несиметричне, антисиметричне.

Матриця суміжності симетричного відношення симетрична щодо головної діагоналі. Для симетричного відношення завжди виконується рівність $R = R^{-1}$ (симетричне відношення та обернене йому відношення збігаються).

Відношення R називається *транзитивним*, якщо серед множини його елементів для пари елементів виду $(x, y), (y, z)$ завжди можна знайти елемент (x, z). Інакше кажучи, якщо на графі відношення з вершини x у вершину z, рухаючись по стрілках, можна прийти через проміжну вершину y, тоді для відношення, що має властивість транзитивності, обов'язково повинен бути й прямий шлях.

Якщо для розглянутого відношення є порушення наведеної вище умови хоча б в одному випадку, тоді таке відношення *не транзитивне*.

Таким чином, при встановленні цієї властивості відношення можливі наступні варіанти: транзитивне, не транзитивне.

У залежності від властивостей бінарні відношення розподіляють на класи.

1. Клас «еквівалентність». Якщо бінарне відношення рефлексивне,

симетричне й транзитивне, то воно відноситься до класу «еквівалентність».

2. Клас «толерантність». Якщо бінарне відношення рефлексивне й симетричне, то воно відноситься до класу «толерантність».

3. Клас «частковий порядок». Якщо бінарне відношення рефлексивне, антисиметричне й транзитивне, то воно відноситься до класу «частковий порядок».

4. Клас «квазіпорядок». Якщо бінарне відношення рефлексивне й транзитивне, то воно відноситься до класу «квазіпорядок».

5. Клас «строгий порядок». Якщо бінарне відношення антирефлексивне, антисиметричне й транзитивне, то воно відноситься до класу «строгий порядок».

6. Клас «строгий попередній порядок». Якщо бінарне відношення антирефлексивне й транзитивне, то воно відноситься до класу «строгий попередній порядок».

Інші варіанти комбінацій властивостей окремих назв не мають й складають клас «інші».

1.7 Операції з бінарними відношеннями

Оскільки бінарні відношення – це множини, для них визначені розглянуті вище операції над множинами (об'єднання, перетин, різниця, доповнення). Необхідно зазначити:

– для бінарного відношення R універсальною множиною W завжди буде квадрат базової множини M, таким чином операція «доповнення R» завжди визначена;

– при виконанні інших операцій результат буде коректним тільки в тому випадку, якщо відношення побудовані на загальній базовій множині.

Операція прямого добутку при її формальному застосуванні до двох відношень у результаті дасть відношення арності 4. Природно зажадати, щоб при виконанні цієї операції арність результату не змінювалася. Це досягається застосуванням властивості транзитивності при виконанні операції перемноження відношень.

Приклад: нехай на базовій множині $M=\{a,b,c,d\}$ задані два бінарних відношення: $R=\{ab,ac,ad\}$ та $Q=\{ac,ad,cd\}$. Добутком цих відношень буде також бінарне відношення S, елементи якого є підмножиною елементів,

отриманих у результаті формального перемноження множин R та Q (з усієї множини чотирьохсимвольних ланцюжків необхідно відібрати тільки ті, у яких середні елементи однакові (при записі середні елементи опускаються)): $S = R \times Q = \{accd, dccd\} = \{ad, dd\}$.

Примітка. Для більш ефективного виконання операції аналітичного перемноження відношень немає необхідності перераховувати всі чотирьохсимвольні ланцюжки; потрібно вибирати тільки ті комбінації, у яких другий символ елемента першого відношення збігається з першим символом елемента другого відношення.

Операцію прямого добутку відношень для двох відношень можна виконати графічно. На графах відношення перемножують за наступним алгоритмом:

– якщо на графі першого відношення є дуга, що з'єднує пари вершин (x, y), а на графі другого – дуга, що з'єднує вершини (y, z), тоді на графі результуючого відношення зобразити дугу, що з'єднує вершини (x, z);

– продовжувати до вичерпання всіх можливих варіантів.

Кінцевий результат перемноження не залежить від способу виконання операції (аналітичний або графічний).

Через прямий добуток відношень можна визначити ступені одного відношення: $Q \times Q = Q^2$; $Q \times Q = Q^3$ та ін.

Транзитивним замиканням відношення (позначається Q^+) називають об'єднання всіх цілих степенів відношення Q. Таке визначення транзитивного замикання відношення не дає ефективного алгоритму його побудови для заданого відношення (за визначенням – це нескінченний процес).

Аналітично це можна записати як об'єднання степенів відношення Q: $Q^+ = Q \cup Q^2 \cup Q^3 \cup Q^4 \ldots \cup Q^n$ або $Q^+ = \bigcup_{i \in N} Q_i$.

Для практичної побудови транзитивного замикання заданого відношення використовують наступний алгоритм:

– на графі заданого відношення додати дуги з використанням властивості транзитивності;

– процес додавання дуг закінчити, якщо на побудованому в такий спосіб графі не можна вже додати жодної дуги (додані на попередньому кроці дуги також беруть участь у побудові).

Транзитивно-рефлексивне замикання відношення Q (позначається

Q^*) – це об'єднання нульового ступеня відношення та транзитивного замикання відношення: $Q^* = Q^0 \cup Q^2 \cup Q^3 \cup Q^4 \ldots \cup Q^n$ або $Q^* = \bigcup_{i \in N_0} Q_i$.

Нульовий ступінь будь-якого відношення, побудованого на базовій множині $M = \{a, b, c, d\}$, має вигляд: $\{aa, bb, cc, dd\}$.

Граф транзитивно-рефлексивного замикання відношення легко отримати з графа транзитивного замикання відношення додаванням у кожній вершині дуги – петлі, що зв'язує вершину саму з собою.

1.8 Задачі до розділу 1

1. Задані дві множини: $A = \{a, ab, ac\}$ і $B = \{b, bc, ab\}$. Знайти об'єднання цих множин $A \cup B$.

2. Задана універсальна множина – множина натуральних чисел $W = \{1, 2, 3, \ldots, 10\}$ та її підмножини A та B. Знайти об'єднання $A \cup B$, якщо елементи множини A – парні числа, а елементи множини B – непарні.

3. Задані множини $A = \{a, b, c\}, B = \{b, c, d\}, C = \{d, e, f\}$. Визначити: $(A \cap B)$ та $(A \cap B \cap C)$.

4. Множина B містить два елементи, а множина A містить три елементи, що не входять у B, але один елемент множини A входить в множину C. Визначити: $(A \cap B)$, $(A \cap B \cap C)$, $(A \cap C)$. Вибір елементів довільний.

5. Задані дві множини $A = \{a, c, b, d, e\}$ і $B = \{a, b, k\}$. Визначити різницю: $(A \setminus B)$ та $(B \setminus A)$.

6. Задана універсальна множина – множина натуральних чисел у діапазоні $W = \{1, 2, 3, \ldots, 10\}$ і її підмножина $A = \{1, 2, 3\}$. Визначити різницю $(A \setminus W)$ та $(W \setminus A)$.

7. Задана універсальна множина – множина натуральних чисел у діапазоні $W = \{1, 2, 3, \ldots, 14\}$, де $n = 14$, і її підмножини $A = \{2, 4, 6, 8\}$ та $B = \{1, 3, 5, 7\}$. Знайти доповнення множин A та B.

8. Задана універсальна множина – множина натуральних чисел у діапазоні $W = \{1, 2, 3, \ldots, 12\}$ і її підмножина $A = \{2, 4, 6, 8\}$. Знайти об'єднання доповнення A та множини A.

9. Задані множини $B=\{a,b,c,\ldots,k\}$, $A=\{a\}$ і $C=\{b,m,z,x,k\}$. Визначити: $(A\setminus B)$, $(B\setminus A)$, $(B\setminus A\cap C)$.

10. Задана універсальна множина $W=\{1,2,3,\ldots,20\}$ та її підмножини: $A=\{i\}$, $B=\{3i\}$ та $C=\{4i\}$, де $i=1,2,3,\ldots$ Знайти множину $(A\cup B)\setminus C$.

11. Задані дві множини $A=\{a,ab,ac\}$ і $B=\{b,bc,ab\}$. Знайти об'єднання цих множин $A\cup B$ та визначити кількість елементів отриманої множини.

12. Задані множина $A=\{abc,d\}$ і множина $B=\{ad,a\}$. Визначити: $A\times B$ (прямий добуток цих множин).

13. Задані множини $C=\{d,e,f\}$ та $A=\{d,f,m\}$. Визначити множини: $Z=C\cup B$ та $D=Z\times A$.

14. Задана множина $Y=\{a,cb\}$. Піднести цю множину в другу та в третю степінь.

15. Спростити формули за допомогою діаграм Вена:

а) $\Phi=A\cap B\setminus C\cup B$;

б) $\Phi=(\overline{A\cap B})\setminus(C\cup B)$;

в) $\Phi=\overline{A}\cap B\setminus C\cup B$;

г) $\Phi=\overline{(A\setminus B)\cap(B\setminus A)}$.

16. Задані бінарні відношення $P=\{ab,ac,bc\}$ і $Q=\{cb,ac,bc\}$. Знайти P^{-1} (обернене відношення) та перетин $P^{-1}\cap Q$.

17. Задані бінарні відношення $P=\{ab,ac,bc,bd,da\}$ і $Q=\{cb,ac,bc,dc\}$. Визначити властивості цих відношень та отримати $R=P\times Q$. Визначити властивості R та клас цього відношення.

18. Для заданого бінарного відношення $P=\{ab,ac,cc,bd,da\}$ побудувати P^2,P^3, P^+ та P^*.

2 ЕЛЕМЕНТИ АЛГЕБРИ ЛОГІКИ

2.1 Прості висловлювання. Логічні зв'язки

Прості висловлювання в природній мові представляють собою розповідні речення, які будемо умовно позначати p, q, r ... Основна властивість простого висловлювання: воно може бути за різних обставин або помилкове (False, 0, «Ні»), або істинне (True, 1, «Так»). Надалі приймемо позначення «0» та «1». Для побудови складних висловлювань будемо використовувати прості висловлювання, п'ять логічних зв'язок (операцій) та дужки. Назви та позначення логічних зв'язок такі:

1. Кон'юнкція (логічне «І») – $p \wedge q$.

2. Диз'юнкція (логічне, що не виключає, «АБО») – $p \vee q$.

3. Заперечення (логічне «НІ») – $\sim p$.

4. Еквівалентність (p тоді й тільки тоді, коли q) – $p \equiv q$.

5. Імплікація (якщо p, тоді q) – $p \to q$.

p – «жарко»;

q – «іде дощ»;

r – «дуже сиро».

Жарко та йде дощ – $p \wedge q$.

Якщо йде дощ, то дуже сиро – $p \to q$.

Результати виконання названих операцій між двома простими висловлюваннями наведені в табл. 2.1.

Таблиця 2.1 – Таблиці істинності для логічних зв'язок

p	q	$p \wedge q$	$p \vee q$	$\sim p$	$p \equiv q$	$p \to q$
1	1	1	1	0	1	1
1	0	0	1	0	0	0
0	1	0	1	1	0	1
0	0	0	0	1	1	1

2.2 Складні висловлювання. Побудова таблиць істинності

Використовуючи прості висловлювання, логічні зв'язки (операції) і дужки, які міняють порядок виконання операцій, можна будувати складні висловлювання. Для визначення пріоритетів виконання операцій треба спочатку визначити їхні властивості (арність, комутативність). Пріоритети операцій у залежності від їхніх властивостей виглядають так:

- унарні операції (найвищий пріоритет);
- бінарні не комутативні;
- бінарні комутативні;
- n-арні (виконують в останню чергу).

Операції одного пріоритету виконують одну за одною зліва направо. Якщо в складному висловлюванні є дужки, то їх треба враховувати при виконанні операції.

Найбільший пріоритет при виконанні має логічна зв'язка «заперечення» (унарна операція), потім «імплікація» (бінарна некомутативна) та «еквівалентність» (бінарна комутативна), після цього йдуть дві інші логічні зв'язки: «кон'юнкція» та «диз'юнкція» (n-арні, комутативні). Логічні зв'язки одного пріоритету виконуються в складному висловлюванні зліва направо. Якщо у висловлюванні є дужки, то спочатку виконуються операції в дужках відповідно до їхнього пріоритету; якщо знак заперечення стоїть над частиною висловлювання, то вважають, що ця частина взята в дужки (хоча дужки насправді можуть бути відсутні).

Далі, для прикладу, будемо використовувати тільки три простих висловлювання: p, q, r. Їх можна трактувати, як аргументи деякої функції, які можуть приймати тільки два значення: «0» та «1», а складне висловлювання – як функцію, що залежно від конкретних значень аргументів приймає значення «0» або «1». Значення цієї функції в залежності від аргументів задаються табличним способом (таблицею істинності складного висловлювання).

При побудові таблиць істинності для складних висловлювань у випадку трьох аргументів досить заповнити вісім рядків ($2^3 = 8$), тому що вони вичерпують усі можливі комбінації значень аргументів.

Приклад: побудувати таблицю істинності для $A = (\sim p \vee q) \wedge \sim r$.

Результат записаний у табл. 2.2.

Таблиця 2.2 – Таблиця істинності A та її покрокова побудова

p	q	r	$\sim p$	$\sim p \vee q$	$\sim r$	A
1	1	1	0	1	0	0
1	1	0	0	1	1	1
1	0	1	0	0	0	0
1	0	0	0	0	1	0
0	1	1	1	1	0	0
0	1	0	1	1	1	1
0	0	1	1	1	0	0
0	0	0	1	1	1	1
Крок			1	2	3	4

2.3 Логічні закони

Приклад: побудувати таблицю істинності для $B = \sim (p \to q) \wedge q$ (табл. 2.3).

Таблиця 2.3 – Таблиця істинності B та її покрокова побудова

p	q	$p \to q$	$\sim (p \to q)$	B
1	1	1	0	0
1	0	0	1	0
0	1	1	0	0
0	0	1	0	0
Крок		1	2	3

Формули, які набувають значення «істина» за будь-яких значень істинності змінних, що входять до них, називають тотожно істинними формулами, або тавтологіями. Крім того, такі формули є логічними законами.

Приклади логічних законів:

1. Логічні закони для диз'юнкції:

$(p \vee q) \equiv (q \vee p)$;

$(p \vee p) \equiv p$;

$(p \vee 0) \equiv p$;

$(p \vee 1) \equiv 1$.

2. Логічні закони для кон'юнкції:

$(p \wedge q) \equiv (q \wedge p)$;
$(p \wedge p) \equiv p$;
$(p \wedge 0) \equiv 0$;
$(p \wedge 1) \equiv p$.

3. Закон подвійного заперечення:

$\sim(\sim p) \equiv p$.

4. Закони де-Моргана:

$\sim(p \vee q) \equiv (\sim p \wedge \sim q)$;
$\sim(p \wedge q) \equiv (\sim p \vee \sim q)$.

5. Закон контрапозиції:

$(p \to q) \equiv (\sim q \to \sim p)$.

Логічний закон можна застосовувати так: у складних висловлюваннях, не порушуючи еквівалентності, частину, яка знаходиться по один бік від знака «≡», можна замінити на частину, яка знаходиться по інший бік.

Формули, які набувають значення «хибність» за всіх значень змінних, що входять до них, називають тотожно хибними.

2.4 Побудова заданих складних висловлювань із потрібною таблицею істинності

У деяких практичних випадках виникає необхідність побудови складних висловлювань із заданою таблицею істинності. Один із методів побудови дає наступна теорема.

Теорема. Усяка логічна функція, крім константи «0», може бути представлена у вигляді диз'юнкції основних кон'юнктів.

Таке представлення називається «довершеною диз'юнктивною нормальною формою» представлення функції (скорочено ДДНФ); константу «0» можна представити, як $p \wedge \sim p$ (табл. 2.4).

Таблиця 2.4 – Таблиця основних кон'юнктів для трьох аргументів

Номер рядка	p	q	r	Основні кон'юнкти
1	1	1	1	$p \wedge q \wedge r$
2	1	1	0	$p \wedge q \wedge \sim r$
3	1	0	1	$p \wedge \sim q \wedge r$
4	1	0	0	$p \wedge \sim q \wedge \sim r$
5	0	1	1	$\sim p \wedge q \wedge r$
6	0	1	0	$\sim p \wedge q \wedge \sim r$
7	0	0	1	$\sim p \wedge \sim q \wedge r$
8	0	0	0	$\sim p \wedge \sim q \wedge \sim r$

Основний кон'юнкт істинний тільки в тому рядку, у якому він знаходиться. Для того щоб одержати функцію із заданою таблицею істинності, досить вибрати в таблиці основні кон'юнкти з рядків, у яких значення функції дорівнюють 1, і зв'язати їх знаками диз'юнкції.

Приклад: побудувати складне висловлювання, що істинно в рядках 2-му та 6-му.

$A=(p \wedge q \wedge \sim r) \vee (\sim p \wedge q \wedge \sim r)$ – вибираємо з таблиці основні кон'юнкти з рядків 2-го та 6-го і зв'язуємо їх диз'юнкцією.

Отримане висловлювання можна спростити, використовуючи теоретично-множинне перетворення висловлювання (буде розглянуто далі).

2.5 Відношення між висловлюваннями

Як було попередньо зазначено, висловлювання (просте або складне) повністю характеризується таблицею істинності (кількість рядків у цій таблиці визначається за формулою 2^n, де n – кількість простих висловлювань у складному висловлюванні). У кожному рядку стоїть одне з двох значень – «0» або «1». Якщо виникає необхідність порівняти між собою два складних висловлювання, тоді, природно, порівнюються між собою їхні таблиці істинності. Результатом цього порівняння буде встановлення виду бінарного відношення, що пов'язує ці висловлювання.

Оскільки в таблицях істинності тільки 0 та 1, то при порівнянні рядків двох таких таблиць без урахування послідовності можливі наступні чотири комбінації 0 та 1:

варіант 1 – «1 – 1»;

варіант 2 – «1 – 0»;

варіант 3 – «0 – 1»;

варіант 4 – «0 – 0».

При порівнянні відсутність усіх варіантів просто неможлива. Відсутність трьох будь-яких варіантів можлива, якщо порівнюються логічні закони (окремий випадок). З усіх шести комбінацій відсутності двох варіантів розглянемо два: відсутність варіантів 1 та 4; відсутність варіантів 2 та 3. Загальна назва цих відношень – «2-відношення»; у першому випадку назва відношення «протилежність», у другому – «еквівалентність». Чотири комбінації, що залишилися, будемо називати «інші 2-відношення».

При відсутності одного варіанта (чотири випадки) використовують загальну назву «прості відношення»:

відсутність варіанта 1 – «T – несумісність»;

відсутність варіанта 2 – «з A слідує B»;

відсутність варіанта 3 – «з B слідує A»;

відсутність варіанта 4 – «F – несумісність»,

де A – перше з порівнюваних висловлювань, B – друге.

Якщо при порівнянні двох висловлювань присутні всі чотири варіанти, тоді такі висловлювання незалежні (відношення – «незалежні»).

2.6 Аргументи

Під аргументом будемо розуміти твердження того, що деяке висловлювання (висновок) випливає з інших висловлювань (посилок).

Однією із задач логіки є перевірка правильності аргументів.

Аргумент називається правильним, якщо кон'юнкція посилок пов'язана з висновком відношенням «випливає».

Наприклад:

$A \wedge B \wedge C$ "*слідує*" D, або $A \wedge B \wedge C \Rightarrow D$;

де A, B, C – посилки (складні або прості висловлювання);

D – висновок (складне або просте висловлювання).

Практично перевірка правильності аргументу виконується у такий спосіб: будується таблиці істинності для кожної посилки та висновку; для того, щоб аргумент був правильним, кожному рядку істинності посилок (рядок, у якому кожна з посилок має значення «1») повинна відповідати істинність висновку.

Примітка: якщо набір посилок такий, що немає жодного рядка, у якому всі посилки істинні, то аргумент з будь-яким висновком буде неправильним.

2.7 Задачі до розділу 2

1. Побудувати таблиці істинності для складних висловлювань (1–10).

2. Спростити висловлення (1–5) та побудувати перемикальні схеми.

3. Спростити висловлення (6–10) та отримати для них мінімальні ДНФ (КНФ).

4. Перевірити правильність аргументів 11–14.

Дані до задач:

1. $(p \wedge q \vee \sim r) \to p \vee r \equiv q.$

2. $\sim(p \vee q \wedge r) \vee (r \to q) \equiv p.$

3. $((\sim p \vee r) \wedge \sim q \equiv r) \to p.$

4. $\sim((p \wedge \sim q) \vee r) \wedge r \equiv p \vee q.$

5. $p \wedge \sim(q \vee r) \equiv r \vee p \to q.$

6. $\sim(p \wedge q \wedge r) \vee p \to q \vee \sim r.$

7. $r \wedge p \vee \sim r \to p \vee r \equiv q \wedge \sim p.$

8. $(\sim p \wedge q \vee \sim r) \vee \sim(p \to q \vee r).$

9. $(p \wedge r) \vee (q \vee \sim p) \to r \equiv \sim p.$

10. $(q \wedge \sim(r \wedge \sim p)) \wedge p \equiv q.$

11.

$$\begin{array}{rcl} p & \equiv & q \\ & & p \\ \hline & \therefore & q \end{array}$$

12.

$$\begin{array}{rcl} p & \wedge & q \\ \tilde{p} & \to & q \\ \hline & \therefore & \tilde{q} \end{array}$$

13.

$$\begin{array}{rcl} p & \vee & q \\ & & \tilde{p} \\ \hline & \therefore & q \end{array}$$

14.

$$\begin{array}{rcl} p & \to & q \\ & & \tilde{q} \\ \hline & \therefore & \tilde{p} \end{array}$$

3 ЕЛЕМЕНТИ ТЕОРІЇ ГРАФІВ

3.1 Загальні поняття та визначення

Граф G як математичний об'єкт – це сукупність двох множин: непустої множини вершин V і множини ребер E, елементи якого являють собою неупорядковані (для орієнтованого графа – упорядковані) пари елементів із множини V:

$$G(V,E)=\langle V;E\rangle, n(V)>0, E\subset V\times V,$$

де для неорієнтованого графа $E=E^{-1}$ (бінарне відношення E симетричне).

Мінімальний граф складається з однієї вершини.

Кожному неорієнтованому графу можна поставити у відповідність орієнтований граф, у якому кожне ребро замінене двома протилежно орієнтованими ребрами, інцидентними тим же вершинам.

Нехай v_1 та v_2 – вершини, $e_1=(v_1,v_2)$ – ребро, що їх з'єднує.

Тоді вершина v_1 і ребро e_1 інцидентні, вершина v_2 і ребро e_1 також інцидентні. Два ребра, інцидентні одній вершині, називаються суміжними; дві вершини, інцидентні одному ребру, також називаються суміжними.

Зазвичай граф зображують на площині у вигляді діаграми: вершини – точками, ребра – лініями (для орієнтованого графа – стрілками), які з'єднують інцидентні вершини.

3.2 Способи задавання графів

Множина вершин та множина ребер для скінченних графів задаються, як правило, перерахуванням. Можливе задавання графа описом відношення інцидентності.

Способи:

1. Відношення інцидентності можна задати матрицею суміжності:

– стовпці та рядки матриці – вершини графа;

– для суміжних вершин елемент матриці дорівнює 1, для інших – 0;

– для неорієнтованого графа ця матриця завжди симетрична;

– кількість ребер дорівнює кількості одиниць вище або нижче головної діагоналі матриці (включаючи елементи на діагоналі).

2. Відношення інцидентності можна задати матрицею інцидентності:

– стовпці матриці відповідають вершинам графа, а рядки – ребрам;

– якщо ребро e_i інцидентне вершині v_j, тоді елемент матриці $\varepsilon_{ij}=1$, в іншому разі – $\varepsilon_{ij}=0$.

Таким чином, у кожному рядку одна або дві одиниці, інші нулі (для петлі одна одиниця).

Для орієнтованого графа при заповненні матриці інцидентності:

$\varepsilon_{ij}=-1$, якщо v_j – початок ребра;

$\varepsilon_{ij}=1$, якщо v_j – кінець ребра;

$\varepsilon_{ij}=\alpha$ (де α – будь-яке число, крім –1, 1, 0), якщо ребро – петля у вершині v_j;

в інших випадках $\varepsilon_{ij}=0$.

3. Граф можна задати списком ребер:

e_i	v_i, v_j
1	a, b
2	b, d
…	

Примітка: тут e_i – ребро, v_i, v_j – пари вершин, що з'єднують цим ребром.

Приклад. Наведено орієнтований граф G (рис. 3.1). Задати його трьома способами.

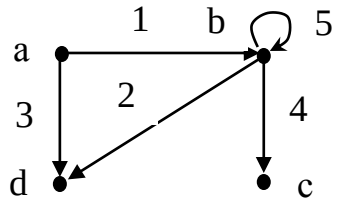


Рисунок 3.1 – Орієнтований граф G

1. Задамо граф G матрицею суміжності:

δ_{ij}	a	b	c	d
a	0	1	0	1
b	0	2	1	1
c	0	0	0	0
d	0	0	0	0

2. Задамо граф G матрицею інцидентності:

ε_{ij}	a	b	c	d
1	1	1	0	0
2	0	1	0	1
3	1	0	0	1
4	0	1	1	0
5	0	2	0	0

3. Задамо граф G списком ребер:

E	$v_i v_j$
1	a, b
2	b, d
3	a, d
4	b, c
5	b, b

3.3 Елементи графів

Граф без кратних ребер називають повним, якщо кожна пара вершин з'єднана ребром.

Граф H називають частиною графа G, якщо множина вершин графа H належить множині вершин графа G і множина ребер графа H належить множині ребер графа G, тобто: $V(H) \subset V(G); E(H) \subset E(G)$.

Частина графа H називається суграфом, якщо вона містить усі вершини графа G.

Суграф H для неорієнтованого графа G називається покриваючим суграфом, якщо будь-яка вершина останнього інцидентна хоча б одному ребру з H.

Підграф $G(U)$ графа G на множині вершин $U(U \subset V)$ – це частина графа, якій належать усі ребра з обома кінцями в множині U.

Зірковий граф для вершини $v(v \in G)$ складається з усіх ребер із початком і кінцем у вершині v. Множина вершин зіркового графа складається з вершини v та інших суміжних із нею вершин.

3.4 Операції з частинами графа

1. Доповнення

Якщо задано граф G та його частину H, то доповнення частини H містить ту частину ребер графа G (і інцидентних їм вершин), що не належать H.

2. Об'єднання

Об'єднанням графів $H1$ та $H2$ є граф H, який складається з ребер і вершин, що належали *H1* і *H2*, або обом графам одночасно, тобто*:*

$H1 \cup H2 = H$;

$V(H) = V(H1) \cup V(H2),\ E(H) = E(H1) \cup E(H2),$

де $V(H), V(H1), V(H2)$ – множини вершин відповідних частин графа G;

$E(H), E(H1), E(H2)$ – множини ребер цих же частин графа G.

$H \cup \overline{H} = G$ (об'єднання частини графа і його доповнення).

3. Перетин

Перетином графів $H1$ та $H2$ є граф H, який складається з ребер і вершин, що належали обом графам *H1* і *H2* одночасно, тобто*:*

$H1 \cap H2 = H$;

$V(H) = V(H1) \cap V(H2),\ E(H) = E(H1) \cap E(H2)$ (якщо $H1$ і $H2$ не мають спільних вершин, то в такому випадку ця операція не визначена).

3.5 Метрика графів

Говорять, що дві вершини в графі зв'язані, якщо існує з'єднуючий їх ланцюг. Граф, у якому всі вершини зв'язані, називається *зв'язним*.

Маршрутом в одиничному зв'язному графі G називається така скінченна послідовність ребер $(e_1, e_2, \ldots, e_n)$, у якій кожні два сусідніх ребра мають загальну інцидентну вершину.

Вершина v_0, яка інцидентна ребру e_1 і не інцидентна ребру e_2, називається початком маршруту в графі G.

Вершина v_n, яка інцидентна ребру e_n і не інцидентна ребру e_{n-1}, називається кінцем маршруту.

Кількість ребер маршруту називається його довжиною.

Якщо вершини v_0 та v_n співпадають, то маршрут називається циклічним (або просто *циклом*).

Відрізок скінченного або нескінченного маршруту сам є маршрутом.

Маршрут у графі G називається *ланцюгом*, якщо всі ребра в послідовності різні, і простим ланцюгом, якщо всі вершини, через які проходить маршрут (а значить, і ребра), різні. Інакше кажучи, у ланцюзі ребро може зустрітися не більше одного разу, а в простому ланцюзі вершина – не більше одного разу.

Відстанню між двома вершинами графа називається мінімальна довжина простого ланцюга, що зв'язує ці вершини (позначення $d(v',v'')$).

Протяжністю між двома вершинами графа називається максимальна довжина простого ланцюга, що зв'язує ці вершини (позначення $g(v',v'')$).

В окремому випадку відстань і протяжність між вершинами можуть бути однаковими.

3.5.1 Діаметр, радіус і центр (центри) графа

Розглянемо одиничний зв'язний неорієнтований граф G.

Мінімальна довжина простого ланцюга з початком в v', і кінцем в v'' називається відстанню між цими вершинами.

Діаметр графа – максимальна з відстаней між будь-якими парами вершин графа: $D(G)=\max\limits_{v',v''\in G} d(v',v'')$.

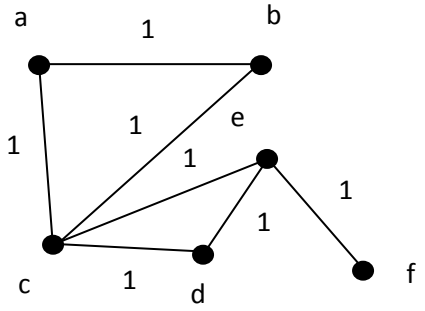
Якщо прийняти за точку відліку відстаней одну з вершин графа G (наприклад, v_i), то максимальна з відстаней від v_i до інших вершин графа G називається віддаленням від цієї вершини: $r(v_i)=\max\limits_{v_j\in G} d(v_j)$.

Вершина v_i називається центром графа, якщо віддалення від неї набуває мінімального значення (таких вершин у графі може бути декілька). Віддалення від центра називається радіусом графа: $r(G)=\min\limits_{v_i\in G} r(v_i)$.

Будь-який простий ланцюг, що з'єднує центр з максимально віддаленою від нього вершиною, називається радіальним ланцюгом.

Розглянемо на *прикладі* визначення цих параметрів графа (аналіз графа на мінімум).

Задано одиничний неорієнтований граф G (рис. 3.2)



$V=\{a,b,c,d,e,f\}; E=\{ab,ac,bc,cd,ce,de,ef\}.$

Рисунок 3.2

Визначити: діаметр, центр (центри) та радіус цього графа.

Розв'язання. Для визначення цих параметрів зобразимо граф та складемо матрицю відстаней: на перетині стовпця та рядка матриці (вершини графа) указується відстань між цими вершинами; така матриця (табл. 3.1) симетрична щодо головної діагоналі.

Таблиця 3.1 – Матриця відстаней для заданого графа G

Вершини	a	b	c	d	e	f	$r(v_i)$	Центр
a	0	1	1	2	2	3	3	ні
b	1	0	1	2	2	3	3	ні
c	1	1	0	1	1	2	2	так
d	2	2	1	0	1	2	2	так
e	2	2	1	1	0	1	2	так
f	3	3	2	2	1	0	3	ні

У стовпці $r(v_i)$ записані віддалення від відповідних вершин (максимальне значення відстані кожного рядка). Максимальне з віддалень і буде діаметром графа (тобто максимально можливою відстанню між вершинами в досліджуваному графі): $D(G)=3$.

Вершини, для яких віддалення $r(v_i)$ набуває мінімального значення (позначені в останньому стовпці «так»), є центрами графа G, а значення віддалення – радіусом графа: $r(G)=2$.

Якщо ребра графа навантажені (кожному ребру поставлено у відповідність певне числове значення), то процедура дослідження аналогічна описаній вище, але при заповненні матриці потрібно визначати відстань між вершинами не за кількістю ребер, а за їхнім сумарним навантаженням.

3.5.2 Діаметр протяжності, радіус протяжності та центр протяжності графа

Розглянемо одиничний зв'язний неорієнтований граф G.

Максимальна довжина простого ланцюга з початком у v' і кінцем у v'' називається протяжністю між цими вершинами (позначається $g(v',v'')$).

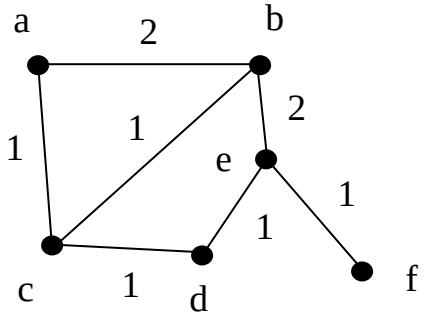
Діаметр протяжності графа – максимальна з протяжностей між будь-якими парами вершин: $L(G)=\max\limits_{v',v''\in G} g(v',v'')$.

Для кожної вершини v_i існують прості ланцюги, що зв'язують її з іншими вершинами графа; їхня довжина називається числом протяжності для вершини v_i: $l(v_i)=\max\limits_{v_j\in G} g(v_j)$.

Центр (центри) протяжності – вершина з мінімальним числом протяжності. Найдовші прості ланцюги з початком у центрі протяжності називаються радіальними, а їхня довжина – радіусом протяжності графа $l(G)=\min\limits_{v_i\in G} l(v_i)$.

Параметри протяжності (центр, діаметр та радіус) при аналізі графа на максимум визначаються аналогічно параметрам при аналізі графа на мінімум.

Приклад. Для аналізу на максимум візьмемо граф G із попереднього прикладу (рис. 3.3).



$$V=\{a,b,c,d,e,f\};\ E=\{ab,ac,bc,cd,eb,de,ef\}.$$

Рисунок 3.3

Визначити діаметр протяжності, центр (центри) протяжності та радіус протяжності цього графа.

Для визначення цих параметрів зобразимо граф G та складемо матрицю протяжностей: на перетині стовпця та рядка матриці (вершини графа) указується довжина між цими вершинами; така матриця (табл. 3.2) симетрична щодо головної діагоналі.

Таблиця 3.2– Матриця протяжностей для заданого графа G

Вершини	a	b	c	d	e	f	$l(v_i)$	Центр
a	0	5	6	5	4	6	6	так
b	5	0	4	4	5	6	6	так
c	7	4	0	6	5	6	7	ні
d	5	4	7	0	6	7	7	ні
e	5	5	5	6	0	1	6	так
f	6	6	6	7	1	0	7	ні

У стовпці $l(v_i)$ записані числа протяжності для вершин, зазначених на початку кожного рядка (максимальне зі значень протяжностей кожного рядка). Максимальне з чисел протяжностей й буде діаметром протяжності графа (тобто максимально можливою протяжністю між вершинами в досліджуваному графі): $L(G)=7$.

Вершини a, b, e, для яких число протяжності набуває мінімального значення, є центрами протяжності графа G, а значення числа протяжності вершини a, b, e – радіусом протяжності графа: $l(G)=6$.

Якщо ребра графа навантажені (кожному ребру поставлено у відповідність певне числове значення), то процедура дослідження графа на максимум аналогічна описаній вище, але при заповненні матриці потрібно визначати протяжність між вершинами не за числом ребер, а за їхньою сумарною довжиною.

3.6 Цикли, дерева, Ейлерові та Гамільтонові графи

Цикл у графі – це ланцюг, у якому початок і кінець співпадають. Якщо це простий ланцюг, то й цикл буде простим. Незалежними в графі називають цикли, які містять хоча б одне ребро, що не входить до інших циклів графа. Кількість незалежних циклів у неорієнтованому графі визначає цикломатичне число γ:

$$\gamma = k + m - n,$$

де k – число зв'язних компонентів графа (для зв'язного графа $k = 1$);

m – кількість ребер у графі;

n – кількість вершин у графі.

Зв'язний граф без циклів називають деревом. У дереві кожна пара вершин зв'язана єдиним простим ланцюгом.

Ейлеревим називають граф, у якому існує цикл, що включає всі ребра графа (ейлерів цикл). Ознака існування такого циклу наступна: усі степені вершин графа парні (степінь вершини графа – кількість ребер, інцидентних цій вершині). Ейлерів ланцюг у графі – це ланцюг, що включає всі ребра графа. Ознака існування такого ланцюга: усі степені вершин графа парні за винятком рівно двох вершин.

Гамільтоновим називають граф, у якому існує простий цикл, що включає всі вершини графа (гамільтонів цикл у графі). Гамільтонів ланцюг у графі – це простий ланцюг, що включає всі вершини графа. Ознаки існування таких циклів та ланцюгів у графах виходять за межі розглянутого матеріалу.

3.7 Задачі до розділу 3

1. Заданий граф $G: V = \{A, B, C, D, E, F\}; E = \{AB, AD, BC, DE, CE, EF\}$.

Привести графічне зображення цього графа, навести приклади суграфа, покриваючого суграфа, підграфа $G(U)$ для вершин $U = \{C, D, E\}$ (із поясненнями) і зіркового графа для вершини D. Визначити параметри відстані (діаметр, центр і радіус) графа G.

Навантаження вершин графа G:

$E = \{AB = 2, AD = 4, BC = 3, DE = 1, CE = 1, EF = 1\}$.

2. Заданий граф $G: V = \{A, B, C, D, E, F\}; E = \{AB, CB, BE, BD, EF, FD\}$.

Привести графічне зображення цього графа, навести приклади для нього суграфа, покриваючого суграфа, підграфа $G(U)$ для вершин $U=\{A,D,E\}$ (із поясненнями) і зіркового графа для вершини B. Визначити параметри протяжності (діаметр, центр та радіус) цього графа. Навантаження вершин графа G:

$E=\{AB=1,CB=2,BE=1,BD=3,EF=1,FD=2\}$.

3. Заданий граф G: $V=\{A,B,C,D,E,F\}$; $E=\{AB,CB,BE,BD,EF,DE\}$.

Привести графічне зображення цього графа, навести приклади для нього суграфа, покриваючого суграфа, підграфа $G(U)$ для вершин $U=\{A,B,E\}$ (із поясненнями) і зіркового графа для вершини F. Визначити параметри протяжності (діаметр, центр і радіус) цього графа.

Навантаження вершин графа G:

$E=\{AB=2,CB=1,BE=3,BD=2,EF=1,DE=1\}$.

4. Заданий граф G: $V=\{A,B,C,D,E,F\}$; $E=\{AB,AF,BE,BD,BC,FC\}$.

Привести графічне зображення цього графа, навести приклади суграфа, покриваючого суграфа, підграфа $G(U)$ для вершин $U=\{F,D,E\}$ (із поясненнями) і зіркового графа для вершини A. Визначити параметри відстані (діаметр, центр та радіус) графа G.

Навантаження вершин графа G:

$E=\{AB=3,AF=2,BE=1,BD=2,BC=1,FC=3\}$.

4 ТЕОРІЯ СКІНЧЕННИХ АВТОМАТІВ

4.1 Скінченні автомати – розпізнавачі

Скінченний автомат (СА) – абстрактний обчислювальний пристрій із фіксованим та скінченним об'ємом пам'яті, що на вході посимвольно зчитує ланцюжок (послідовність символів деякого алфавіту), а на виході повідомляє про його приналежність до деякої множини, для розпізнавання якої він побудований.

Принцип роботи скінченних автоматів різних рівнів широко застосовується в обчислювальних пристроях, як на апаратному, так і на програмному рівнях: це компілятори, транслятори програм, різні кодувальні пристрої, антивірусні програми та ін. Роботу будь-якої програми можна представити як роботу ланцюжка скінченних автоматів різної складності. Розглянемо побудову найпростіших СА – розпізнавачів. У процесі побудови такого скінченного автомата повинні бути визначені наступні параметри:

а) вхідний алфавіт V скінченного автомата (скінченна множина вхідних символів, з яких будуть складатися множини, що будуть подаватися на вхід СА – розпізнавача);

б) скінченна множина станів $S=\{S_0, S_1, \ldots, S_n\}$;

в) початковий стан СА – S_0 (стан, з якого починає роботу СА при обробці нового ланцюжка);

г) множина допускаючих станів – $S_{доп} \subset S$ (підмножина станів, з елементами якої порівнюється досягнутий стан СА після приходу символу «кінець ланцюжка» «–|»);

д) таблиця переходів (керуюча таблиця), яка парі «поточний стан – вхідний символ» ставить у відповідність новий стан СА з множини станів S).

Примітки:

1. У множину вхідних символів обов'язково включають особливий символ «кінець ланцюжка», що повідомляє СА про те, що потрібно досягнутий стан S_i порівняти з елементами множини $S_{дап}$ та, якщо $S_i \in S_{доп}$, пропустити ланцюжок; у іншому випадку ланцюжок відкидається. У тексті цей символ буде мати вигляд «–|».

2. Часто при розпізнанні ланцюжків виникає ситуація, коли неможливо поточній парі «стан – вхідний символ» поставити у відповідність

новий стан. По суті, це означає, що ланцюжок не належить множині, що розпізнається, хоча він і не переглянутий до символу «кінець ланцюжка». Такі ситуації в таблиці переходів позначаються символом «E» («error»); потрапивши в такий стан, СА відкидає ланцюжок і переходить у початковий стан. У конкретних програмних реалізаціях СА може викликатися обробник помилок, який видасть повідомлення про характер помилки, додаткову інформацію, пораду та ін.

СА завжди починає працювати з початкового стану S_0. Символи розпізнаваного ланцюжка надходять посимвольно, починаючи з першого, і змінюють стани СА відповідно до таблиці переходів. Після надходження символу «кінець ланцюжка» досягнутий автоматом стан фіксується та порівнюється з множиною допускаючих станів. На підставі цього порівняння ланцюжок допускається або відкидається. По суті, СА працює як фільтр, що пропускає «правильні» ланцюжки. Інше трактування СА – компактний алгоритм розпізнання регулярних, у тому числі й нескінченних, множин, який будує програміст перед початком кодування (реалізацією алгоритму конкретною мовою програмування).

Побудова СА для розпізнавання заданої множини ланцюжків – процес творчий та неоднозначний. Теоретично, для розпізнання одної множини ланцюжків можна побудувати нескінченну множину СА. Описаний вище принцип розпізнання може бути застосований далеко не до всякої множини. Він ефективний у наступних випадках:

– ланцюжки, що розпізнаються, містять певні сполучення символів на початку, кінці або/та середині ланцюжків;

– ланцюжки, що розпізнаються, містять обмежене число повторень певних символів або їхніх сполучень (не більше n; точно n; не менше n, причому $n=1, 2, 3$);

– ланцюжки, що розпізнаються, містять заборону на певні сполучення символів на початку, кінці або/і у всьому ланцюжку;

– ланцюжки, що розпізнаються, містять комбінації названих вище обмежень.

4.2 Недосяжні стани СА

Недосяжними називаються такі стани СА, які не можуть бути досягнуті з початкового стану дією будь-яких вхідних символів.

Не порушуючи еквівалентності, такі стани можна виключити з таблиці переходів СА. Процедура пошуку недосяжних станів наступна:

Крок 1: записати одноелементну множину, у яку входить початковий стан.

Крок 2: доповнити цю множину станами, у які переходить СА зі станів, уже присутніх у множині при дії будь-яких вхідних символів.

Крок 3: якщо на кроці 2 множина не поповнюється новими елементами, тоді отримана вичерпна множина досяжних станів; стани СА, що не увійшли в множину, побудовану на кроці 2, недосяжні.

4.3 Еквівалентні стани СА

Стани s та t двох різних скінченних автоматів еквівалентні тоді й тільки тоді, коли перший СА, почавши роботу зі стану s, буде допускати ті ж ланцюжки, що й другий СА, почавши роботу зі стану t. Якщо ці стани початкові, то ці автомати еквівалентні (тобто будуть розпізнавати ті самі множини).

Два стани скінченного автомата еквівалентні тоді й тільки тоді, коли, почавши роботу із цих станів, скінченний автомат буде допускати ті самі ланцюжки.

Інакше кажучи, якщо для двох станів СА немає розрізняючого ланцюжка, тоді такі стани еквівалентні (розрізняючий – такий ланцюжок символів, що призводить із порівнюваних станів до різних кінцевих результатів).

Еквівалентні стани, що належать одному СА, не порушуючи еквівалентності, можна замінити одним (у таблиці переходів залишити один з рядків еквівалентних станів, видаливши інші, при цьому замінити імена вилучених станів на залишене).

Наведене вище визначення еквівалентних станів не дає ефективного алгоритму пошуку таких станів.

Пошук еквівалентних станів проводять у процесі багаторазової розбивки множини станів СА в залежності від характеру дії вхідних символів. Ця операція робиться покроково.

Крок 1: множину станів СА розбити на дві підмножини за дією символу «кінець ланцюжка» (допускаючі та не допускаючі); у кожній з отриманих підмножин усі стани за дією символу «кінець ланцюжка» еквівалентні.

Крок 2: повторити запис підмножин, отриманих на момент виконання цього кроку, у вигляді нового рядка та стрілками показати переходи для кожного стану з попереднього в новий рядок за дією іншого вхідного символу.

Крок 3: виконати розбивку підмножин станів на нові еквівалентні за вхідним символом підмножини, керуючись правилом – «підстав для

розбивки підмножини нема, якщо всі її елементи (стани) переходять в одну підмножину»; підмножина розбивається на нові підмножини, у які входять елементи, що переходять в одну підмножину.

Крок 4: якщо в утворених підмножинах більше одного стану, повторювати крок 2 до повного перебору всіх вхідних символів; після завершення процесу два й більше стани, що входять в отримані наприкінці підмножини, еквівалентні між собою.

Скінченний автомат, у якому виключені недосяжні та еквівалентні стани, називається мінімальним СА.

Для кожного СА існує нескінченна множина інших еквівалентних йому СА (розпізнають ту ж множину ланцюжків), однак існує єдиний мінімальний СА (автомат із мінімальною кількістю станів).

4.4 Недетерміновані скінченні автомати

Недетермінований скінченний автомат (НСА) являє собою звичайний СА, з тією різницею, що в таблиці переходів парі «вхідний символ – стан» може бути поставлена у відповідність множина станів (а не один, як у СА) і початкових станів може бути кілька. Такий автомат можна отримати при вирішенні конкретної задачі – побудові розпізнавача для регулярної множини. Єдиний недолік такого НСА, у порівнянні із СА, – неможливість простої програмної реалізації.

У процесі побудови такого недетермінованого скінченного автомата повинні бути визначені наступні параметри:

а) вхідний алфавіт V (скінченна множина вхідних символів і символ «кінець ланцюжка» «–|»);

б) скінченна множина станів $S=\{S_0, S_1,\ldots,S_n\}$;

в) множина початкових станів $S_{поч}\subset S$ (підмножина станів, з якої може починати роботу НСА при обробці нового ланцюжка);

г) множина допускаючих станів $S_{доп}\subset S$ (підмножина множини станів S, з якою порівнюються досягнуті НСА стани після приходу символу «кінець ланцюжка»);

д) таблиця переходів (керуюча таблиця), яка парі «поточний стан – вхідний символ» ставить у відповідність кілька нових станів; якщо парі «поточний стан – вхідний символ» не можна поставити у відповідність жоден стан із множини S, тоді клітину таблиці переходів залишають порожньою (умовно це стан помилки).

Роботу НСА можна інтерпретувати двома способами:

1. На деяких кроках при розпізнанні ланцюжка за допомогою НСА є множина виборів. Якщо є хоча б одна послідовність станів, при якій НСА закінчує роботу в допускаючому стані, то такий ланцюжок буде допущений цим НСА.

2. При виникненні альтернативи під час роботи НСА (у виборі початкового стану або переходу) автомат розпадається на скінченні автомати за кількістю альтернатив, які працюють паралельно. Якщо при надходженні символу «кінець ланцюжка» хоча б один із паралельно працюючих СА перебуває в допускаючому стані, то такий ланцюжок буде допущений цим НСА.

Існує процедура еквівалентного перетворення НСА в СА за наступним алгоритмом (перетворення виконується на таблиці переходів НСА):

1. Для майбутньої керуючої таблиці еквівалентного СА зобразити стовпці за кількістю вхідних символів і позначити їх вхідними символами.

2. У першу клітину першого рядка занести, як множину, усі початкові стани НСА.

3. Заповнити інші клітини цього рядка (крім останньої) так: за таблицею переходів НСА виписати у вигляді множин усі стани, у які переходять стани, записані в першій клітині через відповідні вхідні символи.

4. У першій клітині чергового рядка записати одну з нових множин станів, які отримані на кроці 3 при заповненні клітин таблиці переходів, і виконати крок 3.

5. Закінчити процес побудови таблиці переходів після того, як будуть вичерпані всі варіанти множин, які з’явилися в таблиці.

6. При заповненні стовпця із символом «кінець ланцюжка» ставити «допустити», якщо в множину станів першої клітини рядка входить хоча б один допускаючий стан НСА, і «відкинути» – у іншому випадку.

7. Перепозначити множини станів як нові стани й заново заповнити таблицю переходів (отримана таблиця переходів еквівалентного СА).

8. Визначити інші параметри СА за отриманою таблицею переходів.

Описану вище процедуру проілюструємо на конкретному прикладі.

Приклад. НСА заданий таблицею переходів:

S	k	n	–|
→A	A	B,C	0
→B	B	C	1
C		A,C	1

Для спрощення запису в останньому стовпці таблиці символом «0» позначена дія «відкинути», символом «1» – «допустити».

$V = \{k, n, \dashv\}; S_{ïî÷} = \{A, B\};$

$S = \{A, B, C\}; S_{äîï} = \{B, C\}.$

Перетворити заданий НСА в еквівалентний СА та мінімізувати отриманий СА.

Розв'язання

1. Виконаємо процедуру перетворення НСА в СА за приведеним вище алгоритмом (табл. 4.1).

Таблиця 4.1 – Керуюча таблиця заданого НСА

S	k	n	–|
A,B	A,B	B,C	1
B,C	B	A,C	1
B	B	C	1
A,C	A	A,B,C	1
C		A,C	1
A	A	B,C	0
A,B,	A,B	A,B,C	1

2. Перепозначимо стани в табл. 4.1 у такий спосіб (табл.4.2):

$a = \{A, B\};\ b = \{B, C\}; c = \{B\}; d = \{A, C\}; e = \{C\};\ f = \{A\}; g = \{A, B, C\}.$

Таблиця 4.2 – Керуюча таблиця еквівалентного СА

S	k	n	–|
a	a	b	1
b	c	d	1
c	c	e	1
d	f	g	1
e		d	1
f	f	b	0
g	a	g	1

3. Перевіримо стани отриманого СА на досяжність:

$\{a\}\to\{a,b\}\to\{a,b,c,d\}\to\{a,b,c,d,e,f,g\}.$

Усі стани СА досяжні.

4. Перевіримо стани отриманого СА на еквівалентність.

Перша розбивка множини станів на підмножини за вхідним символом «кінець ланцюжка» «–|»: $\{f\};\{a,b,c,d,e,g\}$.

Проаналізуємо вплив вхідного символу «k» на другу підмножину: $d\to f;\ e\to\{\ \}$; інші залишаються в цій же підмножині. Є підстави виділити d та e в окремі підмножини.

Тепер розбивка буде мати вигляд $\{f\};\{d\};\{e\};\{a,b,c,g\}$.

Проаналізуємо вплив вхідного символу «n» на останню підмножину:

$b\to d;\ c\to e;\ a\to b;\ g\to g$.

Стани b та c переходять у інші групи, і їх потрібно виділити в окремі підмножини; стан a – у виділений на цьому кроці стан b, і на цій підставі його теж потрібно виділити. У результаті:

$\{f\};\{d\};\{e\};\{a\};\{b\};\{c\};\{g\}$.

Висновок: еквівалентних станів немає; отриманий СА є мінімальним.

5. За таблицею переходів СА визначимо інші його параметри:

$V=\{k,n,-|\};\ S_{\text{ïî÷}}=\{a\};\ S_{\text{äîï}}=\{a,b,c,d,e,g\};\ S=\{a,b,c,d,e,g,f\}$.

4.5 Задачі до розділу 4

1. Побудувати з повним описом скінченний автомат для розпізнання ланцюжків в алфавіті $V=\{0,1,2\}$, які починаються на «0» та в ланцюжку зустрічається рівно два символи «2».

2. Побудувати з повним описом скінченний автомат для розпізнання ланцюжків в алфавіті $V=\{a,b,c\}$, які містять символ «b» тільки парами та закінчуються на «c».

3. Побудувати з повним описом скінченний автомат для розпізнання ланцюжків в алфавіті $V=\{a,b,c\}$, які починаються з «cb», та в ланцюжку зустрічається тільки один раз «ac». Виконати процедуру мінімізації СА.

4. Побудувати з повним описом скінченний автомат для розпізнання ланцюжків в алфавіті $V=\{0,1\}$, які починаються з «11», та у ланцюжку символи «0» стоять тільки по два. Виконати процедуру мінімізації СА.

5. Визначити еквівалентні та недосяжні стани СА. Побудувати мінімальний СА й діаграму переходів для отриманого мінімального СА.

а

S	a	b	–|
A	F	B	0
B	E	F	0
C	F	A	1
D	B	B	1
E	B	D	0
F	E	E	1

б

S	a	b	–|
A	C	E	1
B	D	A	0
C	A	F	1
D	C	E	1
E	F	D	0
F	E	D	0

в

S	a	b	–|
A	A	D	0
B	C	F	0
C	B	F	0
D	B	A	1
E	D	F	1
F	A	C	1

г

S	a	b	–|
A	F	B	0
B	B	F	0
C	F	A	1
D	E	B	1
E	E	F	0
F	E	B	1

д

S	a	b	–|
A	B	D	0
B	A	F	0
C	B	D	0
D	C	A	1
E	D	F	1
F	B	A	1

е

S	a	b	–|
A	C	D	1
B	D	A	0
C	D	F	1
D	C	E	1
E	F	C	0
F	E	D	0

6. Перетворити заданий НСА в еквівалентний СА та виконати процедуру мінімізації СА.

а

S	0	1	–|
→A	B	A,C	1
→B		A	0
C	C	A	1

б

S	0	1	–|
A	B	C	0
B	B	A,B	1
→C	A	C	0

в

S	0	1	–|
→A	B	C	1
B	A,B	C	0
C	B	A	0

г

S	a	b	–|
→A	A	B,C	1
→B	C	A	0
C	A,C		0

д

S	a	b	–|
→A	A	C	1
B	A	A,B	0
→C	B	A,C	0

е

S	0	1	2	–|
→A	A,B	B	A,C	1
B	A	B	A	0
C	A	C	C	0

ж

S	0	1	–|
→A	A	B,C	0
B	B	C	1
→C		A,C	0

з

S	0	1	–|
→A	A,C	B,C	0
→B	A	C	0
C	B	B	1

і

S	a	b	–|
A	A	D	0
→B	B	C	**0**
C	**C**	**D**	**0**
D	**A,D**	**D**	**1**

к

S	0	1	–|
→A	B	A,C	0
B	**B**	**C**	**1**
C	**C**	**A,C**	**0**

5 АВТОМАТИ З МАГАЗИННОЮ ПАМ'ЯТТЮ

5.1 Автомати-розпізнавачі з магазинною пам'яттю

Далеко не для всіх регулярних множин можна побудувати СА-розпізнавачі, тому що СА не має можливості рахувати та запам'ятовувати кількість символів ланцюжка, що розпізнається. Для цього використовується спеціальний пристрій – магазин, у який можна поміщати символи або видаляти їх, запам'ятовуючи або порівнюючи кількість символів вхідного ланцюжка. Автомат із таким пристроєм називається автоматом-розпізнавачем із магазинною пам'яттю (скорочено – МП-автомат).

МП-автомат задається:

1) скінченною множиною вхідних символів (уключаючи символ кінця ланцюжка «–|»);

2) скінченною множиною магазинних символів (уключаючи маркер дна магазина – «∇»);

3) скінченною множиною станів $S = \{S_0, S_1, \ldots, S_n\}$;

4) керуючою таблицею, що кожній комбінації трьох параметрів – вхідний символ, магазинний символ (верхній символ магазина), стан – ставить у відповідність дію з магазином, вхідним символом та станом;

5) початковою конфігурацією (початковий стан і початковий вміст магазина), з якої МП-автомат починає роботу;

6) множиною конфігурацій, що допускають (комбінацій – стан МП-автомата та верхній символ магазина в момент, коли приходить символ «кінець ланцюжка»).

Більшість клітин керуючої таблиці мають вигляд, показаний на рис. 5.1: клітина розбита на три поля. У лівому полі вказується новий стан МП-автомата, у центральному – дії з магазином, у правому – дії з вхідним символом.

Ряд клітин керуючої таблиці може без ділення на поля заповнюватися символом «Е» (стан помилки). Якщо МП-розпізнавач потрапив у такий стан, то обробка ланцюжка припиняється й такий ланцюжок відкидається.

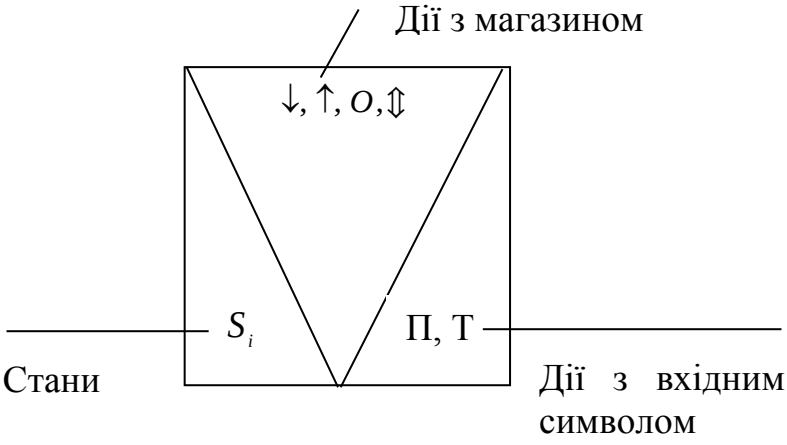


Рисунок 5.1 – Клітина керуючої таблиці МП-розпізнавача

Допустимі дії з вхідними символами:

1. Тримати поточний вхідний символ ланцюжка, що розпізнається (Т).

2. Перейти до наступного символу ланцюжка, що розпізнається (П).

Примітка. Заборонено перехід до наступного символу, якщо поточний символ «–|» («кінець ланцюжка»).

Допустимі операції над магазином: магазин можна представити у вигляді одностороннього стека або впорядкованого списку, у якому доступний до виконання дій тільки верхній символ. При вштовхуванні ($\downarrow$) нового символу в магазин усі символи, що перебували до цього в магазині, зміщаються на одну позицію (для визначеності вправо). Доступ можливий тільки до символу, поміщеного в магазин останнім. З ним і виконуються операції, зазначені в керуючій таблиці:

1. Вштовхнути в магазин магазинний символ, наприклад А ($\downarrow A$).

2. Виштовхнути з магазина верхній символ, наприклад А ($\uparrow A$).

3. Залишити магазин без змін (О).

4. Замінити верхній символ магазина ($\Updownarrow AB$).

Примітки:

1. Заборонено дії 1, 2, 4 із символом дна магазина «∇».

2. Допускається виконувати дію «$\downarrow$» із декількома символами ($\downarrow AA$).

3. Дія «$\Updownarrow$» виконується так: верхній символ магазина заміняється деяким зазначеним ланцюжком ($\Updownarrow Aab$). Для визначеності при записі вмісту магазина будемо вважати, що символ дна «∇» займає найправіше положення в списку. Приміром, вміст магазину до виконання операції «$\Updownarrow Aab$» – ВВА∇; після – АавВА∇.

Результатом роботи для МП-розпізнавача буде повідомлення «допустити» або «відкинути» ланцюжок, переглянутий посимвольно після надходження символу «кінець ланцюжка».

Вхідний ланцюжок допускається МП-розпізнавачем, якщо під впливом цього ланцюжка автомат, що почав роботу в початковій конфігурації (у початковому стані і з початковим вмістом магазину), досягає заключної конфігурації після надходження символу «кінець ланцюжка»; інакше ланцюжок відкидається.

Приклад. Побудувати МП-розпізнавач для розпізнання множини ланцюжків наступного виду: {0(n)1(n) | n>0} (правильний ланцюжок складається з нулів, після яких іде така ж кількість одиниць).

Рішення

Розробимо логічну послідовність дій (алгоритм роботи) МП-автомата для розпізнання заданої множини ланцюжків:

1. При обробці правильного ланцюжка першими символами, які потрібно обробити МП-автомату, буде серія з n «0». Їхню кількість потрібно запам’ятати для порівняння з кількістю «1».

2. Надходження кожного символу «0» будемо супроводжувати вштовхуванням у магазин символу А для того, щоб запам’ятати кількість нулів і порівняти потім із кількістю одиниць (у початковому стані магазин порожній).

3. Після приходу першого символу «1», нулі більше не повинні надходити (надходження «0» повинно переводити автомат у стан помилки «Е»).

4. Надходження кожного символу «1» повинне супроводжуватися виштовхуванням із магазина символу «А». У такий спосіб буде контролюватися рівність кількості «0» та «1».Ланцюжок буде допущений МП-автоматом, якщо при надходженні символу «–|» магазин виявиться порожнім.

Реалізуємо описану стратегію в конкретному МП-автоматі:

1. Множина вхідних символів V = {0, 1, –|}.

2. Множина магазинних символів $\{A, \nabla\}$.

3. Множина станів $S = \{S_1, S_2\}$.

4. Початкова конфігурація МП-автомата $(S_1; \nabla)$ – стан S_1, магазин порожній (верхній символ магазина «∇»).

5. Заключна (допускаюча) конфігурація МП-автомата (S_2, ∇) – стан S_2, магазин порожній (верхній символ магазина «∇»).

6. Таблиця переходів має вигляд, представлений на рис. 5.2 («Е» – стан помилки) – якщо МП-розпізнавач попав у такий стан, то процес посимвольного перегляду ланцюжка закінчується, і він відкидається.

Стани і $V_{маг}$		$V_{вх}$		
		0	1	– |
S_1	∇	$\downarrow A$ S_1 $П$	E	Відк.
	A	$\downarrow A$ S_1 $П$	$\uparrow A$ S_2 $П$	Відк.
S_2	∇	E	E	Доп.
	A	E	$\uparrow A$ S_2 $П$	Відк.

Рисунок 5.2 – Таблиця переходів МП-автомата

Перевіримо роботу побудованого МП-автомата. Розберемо ланцюжок 0011–|, що повинен бути допущеним. Результати посимвольного розбору зведені в табл. 5.1.

Таблиця 5.1 – Результати розбору ланцюжка

Неопрацьований ланцюжок	Символ, що обробляється	Поточний стан	Верхній символ магазину	Вміст магазина	Дії з магазином
0011–|	0	S_1	∇	∇	$\downarrow A$
011–|	0	S_1	A	$A\nabla$	$\downarrow A$
11–|	1	S_1	A	$AA\nabla$	$\uparrow A$
1–|	1	S_2	A	$A\nabla$	$\uparrow A$
–|	–|	S_2	∇	∇	Допустити

Ланцюжок допущений, тому що остання конфігурація МП-автомата є такою, що допускає $(S_2;\nabla)$.

5.2 Автомати-транслятори з магазинною пам’яттю

У ряді випадків при обробці вхідних ланцюжків крім розпізнання необхідне їхнє перетворення в іншу множину ланцюжків. Такі дії може виконувати МП-транслятор, на виході якого буде формуватися вихідний ланцюжок.

МП-транслятор задається наступними параметрами:

1. Вхідним алфавітом $V_{вх}$ – скінченною множиною вхідних символів (включаючи символ кінця ланцюжка «–|»).

2. Вихідним алфавітом $V_{вих}$ – скінченною множиною вихідних символів.

3. Алфавітом магазинних символів – скінченною множиною магазинних символів (включаючи маркер дна магазина – «∇»).

4. Скінченною множиною станів.

5. Керуючою таблицею, що кожній комбінації трьох параметрів – вхідний символ, магазинний символ (верхній символ магазина), стан – ставить у відповідність чотири параметри (дія з магазином, дія з вхідним символом, стан і вихідні символи).

6. Початковою конфігурацією (початковий стан і початковий вміст магазину).

7. Множиною заключних конфігурацій, що допускають вхідний ланцюжок (стан МП-транслятора та верхній символ магазина в момент, коли приходить символ «кінець ланцюжка»).

Заповнення клітини в керуючій таблиці МП-транслятора (рис. 5.3) аналогічне заповненню клітини МП-розпізнавача, у якій додається внизу четверте поле (у ньому вказуються вихідні символи, що видаються на цьому кроці на вихід МП-транслятора).

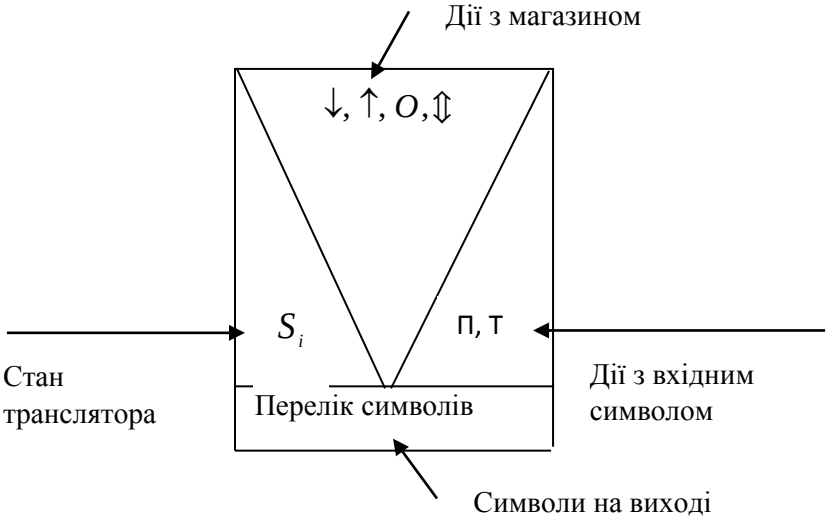


Рисунок 5.3 – Заповнення клітини керуючої таблиці в МП-трансляторі

Ряд клітин керуючої таблиці можуть без поділу на поля заповнюватися символом «E» (стан помилки). Якщо МП-транслятор потрапив у такий стан, то трансляція вхідного ланцюжка припиняється й такий ланцюжок відкидається.

Приклад. Побудувати МП-транслятор для розпізнання множини {W2V} і перетворення її в множину {1(n) 0(m)}, де W – довільний ланцюжок з «0» та «1» ,V – ланцюжок, зворотний W, m – число «0» до символу «2», n – число «1» до символу «2» (приклад ланцюжка і його перетворення 0010112110100 → 111000).

Рішення

Розробимо логічну послідовність дій МП-транслятора для розпізнання та трансляції заданої в умові множини ланцюжків:

1. У процесі розбору вхідного ланцюжка МП-транслятору необхідно запам'ятовувати не тільки кількість одиниць і нулів частини ланцюжка W, але й порядок їхнього приходу для наступної перевірки ланцюжка V. Запам'ятовування можна реалізувати, вштовхуючи в магазин символ «B» при приході одиниці та символ «A» – при приході нуля. Цей процес продовжувати до приходу символу «2» (тобто до моменту закінчення ланцюжка W); у результаті таких дій у магазині виявиться копія ланцюжка W, записана у зворотному порядку (верхнім у магазині буде символ, що відповідає останньому символу в ланцюжку W).

2. При розборі W на вихід при приході «1» потрібно видавати теж «1», а при приході «0» на вихід не видавати нічого. Прихід «2» потрібно зафіксувати зміною стану транслятора; із магазином дій не виконувати.

3. У другому стані йде обробка частини ланцюжка V, і дії транслятора наступні:

а) якщо в магазині верхній символ «B» та прийшла одиниця – виштовхнути «B», на вихід нічого не видавати й перейти до обробки наступного символу вхідного ланцюжка;

б) якщо в магазині верхній символ «A» і прийшов нуль – виштовхнути «A», на вихід видати «0» і перейти до обробки наступного символу вхідного ланцюжка;

в) два варіанти, що залишилися – стан помилки «E» (вхідний ланцюжок не відповідає виду {W2V}).

Повний опис МП-транслятора:

1. Множина станів S = {s1, s2}.

2. Множина вхідних символів: {0, 1, 2, –|}.
3. Множина магазинних символів: {A, B, ∇}.
4. Множина вихідних символів: {0, 1}.
5. Керуюча таблиця МП-транслятора (рис. 5.4).

Стани і $V_{маг}$		$V_{вх}$			
		0	1	2	–|
S_1	∇	$\downarrow A$ S_1 $П$ –	$\downarrow B$ S_1 $П$ 1	E	відк.
	A	$\downarrow A$ S_1 $П$ –	$\downarrow B$ S_1 $П$ 1	O S_2 $П$ –	відк.
	B	$\downarrow A$ S_1 $П$ –	$\downarrow B$ S_1 $П$ 1	O S_2 $П$ –	відк.
S_2	∇	E	E	E	доп.
	A	$\uparrow A$ S_2 $П$ 0	E	E	відк.
	B	E	$\uparrow B$ S_2 $П$ –	E	відк.

Рисунок 5.4 – Керуюча таблиця МП-транслятора

Перевіримо роботу побудованого МП-транслятора, розібравши ланцюжок 0112110 (табл. 5.2).

Таблиця 5.2 – Результати розбору ланцюжка

Неопра-цьований ланцюжок	Символ, що обробляється	Поточний стан	Верхній символ магазина	Вміст магазина	Дії з магазином	Вихідний ланцюг
0112110–|	0	S_1	∇	∇	$\downarrow A$	–
112110–|	1	S_1	A	$A\nabla$	$\downarrow B$	1
12110–|	1	S_1	B	$BA\nabla$	$\downarrow B$	11
2110–|	2	S_1	B	$BBA\nabla$	О	11
110–|	1	S_2	B	$BBA\nabla$	$\uparrow B$	11
10–|	1	S_2	B	$BA\nabla$	$\uparrow B$	11
0–|	0	S_2	A	$A\nabla$	$\uparrow A$	110
–|	–|	S_2	∇	∇	допустити	110

Вхідний ланцюжок допущений і перетворений до заданого виду.

5.3 Задачі до розділу 5

Задача 1. Для наведених у табл. 5.3 варіантів множин ланцюжків побудувати МП-розпізнавачі з повним описом процесу побудови.

Таблиця 5.3

Варіант	Вид регулярної множини
1	A = {0(2n) 1 0(n) | n >0}
2	A = {0(n) 1(m+n) 0(m) | n,m >0}
3	A = {a(n) b(m) a(n–m) | n,m >0, n>m}
4	A = {1(2n) 2 0(n) | n >0}
5	A = {1(n) 0(m) 1(m+1) 0(2n) | n, m >0}
6	A = {ланцюжок з 0 та 1, причому нулів на два більше}
7	A = {1(n) 0(m) 1(m+n)) | n, m >0}
8	A = {ланцюжок з 0 та 1, причому нулів на два менше}
9	A = {1(n) 0(m) 1(m–n–1) | n, m >0, m>n}
10	A = {ланцюжок з однакового числа 0 та 1, причому починається з нуля }

Задача 2. Для наведених у табл. 5.4 варіантів побудувати МП-транслятори для перетворення множини ланцюжків «A» у множину ланцюжків «B» з повним описом процесу побудови.

Таблиця 5.4

Варіант	Вид ланцюжків A, B
1	A = {a(n+m) b(m) a(n)} ⇒ B = {1(m) 0(2n)}
2	A = {1(n) 0(2m)} ⇒ B = {0(n+1) 1(m) 0(n)}
3	A = {1(n–1) 0(2n)} ⇒ B = {a(n+1) b(n–1)}
4	A = {1(n) 0(m)} ⇒ B = {b(n+m) a(m+2)}
5	A = {a(n) b(m–n) a(n) } ⇒ B = {0(n+m) 1(n)}
6	A = {довільний ланцюжок з 0 та 1, причому нулів на два менше} ⇒ B = {b(n) a(n+2) | n – кількість 0 в A}
7	A = {довільний ланцюжок з 0 та 1, причому нулів на два більше} ⇒ B = {0(n+m) 1(n), | n та m числа 0 та 1 відповідно}
8	A = {довільний ланцюжок з однакового числа 0 та 1, починається з 0 та у голові ланцюжка кількість символів 0 >= кількості символів 1} ⇒ B = {(ab) n | n – кількість однакових символів в A}
9	A = {довільний ланцюжок з однакового числа 0 та 1, причому починається з 1} (B = {0(n) 1(n)}

6 ГРАМАТИКИ

6.1 Загальні відомості

У розширеному розумінні під терміном «мова» розуміють усякий засіб спілкування, що складається із:

– знакової системи, тобто множини допустимих послідовностей знаків;

– множини змістів цієї системи;

– відповідності між послідовностями знаків і змістами, що роблять осмисленими допустимі послідовності знаків.

Знаками можуть бути букви алфавіту, математичні позначення, звуки, ритуальні дії та ін. Наука про осмислені знакові системи називається семіотикою. Найбільш дослідженими є знакові системи, у яких знаками є символи алфавіту. Правила, що визначають множину текстів (допустимих послідовностей знаків), утворюють синтаксис мови; опис множини змістів і відповідності між текстами та змістами – семантику мови. До таких знакових систем відносяться природні мови, мови різних галузей науки, мови програмування.

Семантика мови істотно залежить від призначення мови, у той час як для синтаксису можна сформулювати поняття та методи, що не залежать від призначення та цілей мови. Для дослідження синтаксису склався спеціальний математичний апарат – теорія формальних граматик, у якій мова розуміється вже не як засіб спілкування, а як множина формальних об’єктів – послідовностей символів алфавіту. Ці послідовності називають ланцюжками й мову розуміють як множину ланцюжків.

Нехай заданий алфавіт V, у якому можна побудувати множину V^*(читається: «ітерація алфавіту V») ланцюжків. Формальна мова L в алфавіті V – це підмножина ланцюжків з V^* $\left(L \subset V^*\right)$. Опис формальної мови здійснюється за допомогою формальної породжуючої граматики, або скорочено «формальної граматики».

Формальна породжуюча граматика G (далі – граматика G) – це формальна система, обумовлена четвіркою об’єктів:

$$G[Z]=(VN,VT,Z,P),$$

де VN – алфавіт нетерміналів (допоміжних символів);

VT – алфавіт терміналів (основних символів);

Z – початковий символ (аксіома) граматики;

P – скінченна множина правил.

Нетерминали прийнято позначати великими буквами латинського алфавіту, термінали – малими буквами. В алфавіт нетерміналів обов'язково входить початковий символ граматики.

Кожне правило з множини P має вигляд $x \to y$, де x, y – ланцюжки, що складаються з термінальних і нетермінальних символів. Надалі будемо розглядати граматики, що визначаються тільки правилами, ліві частини яких складаються з одного нетермінального символу (контекстно-вільні граматики). При цьому повинно бути хоча б одне правило, ліва частина якого – початковий символ граматики.

Граматика описує нескінченну мову, якщо хоча б одне з правил рекурсивне, тобто в правій частині правила міститься його ліва частина у явному або неявному вигляді.

Приклад. Задана граматика

$$G[Z]: (VN = \{Z, A, B\}; VT = \{a, b, c\};\ Z;$$
$$P = \{Z \to ABc; (1)$$
$$A \to a; \quad (2)$$
$$B \to b \quad (3)\}).$$

За допомогою граматики G можна продукувати (одержувати) ланцюжки в алфавіті терміналів. Дерево виводу для одного з ланцюжків (abbc) має такий вигляд (рис. 6.1).

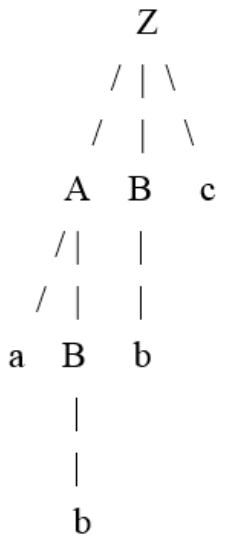


Рисунок 6.1 – Дерево виводу для одного з ланцюжків (abbc)

Цей вивід можна записати також і в рядок:

(1) (2) (3) (3) ← номери застосованих правил

$$Z \to ABc \to aBBc \to aBbc \to abbc.$$

Вивід триває доти, поки на черговому кроці не отримаємо ланцюжок, що складається тільки з термінальних символів (тобто не можна буде застосувати жодного з правил граматики).

Скорочено вивід можна записати, пропустивши проміжні результати, так: $Z^+ \to abbc$ (ланцюжок $abbc$ виведений з початкового символу Z у заданій граматиці).

Сентенціальна форма – будь-який ланцюжок термінальних і нетермінальних символів, що отримано на будь-якому кроці процесу виводу. Множину сентенційних форм можна одержати з дерева виводу, обходячи його по вузлах і дотримуючись наступних правил:

– починати обхід із самого лівого вузла;

– обхід треба робити так, щоб при переході до наступного вузла утворене піддерево не включало, як елемент, попереднє піддерево.

Фраза – частина сентенціальної форми, що виводиться з одного нетермінала за кілька кроків. Для простої фрази крок виводу дорівнює 1.

Один і той же ланцюжок можна одержати, застосовуючи правила в різних послідовностях (дерева виводів різні). Якщо для однозначності в процесі виводу на кожному кроці застосовувати правило до самого лівого (правого) нетермінала в сентенціальній формі, то одержимо лівосторонній (правосторонній) вивод. Таким чином:

1. Кожному ланцюжку, виведеному в заданій граматиці, відповідає одне або кілька дерев виводу.

2. Кожному дереву відповідає один або більше виводів.

3. Кожному дереву відповідає єдиний правий і єдиний лівий виводи.

4. Якщо кожному ланцюжку, виведеному в заданій граматиці, відповідає єдине дерево виводу, то така граматика називається однозначною (у правій частині кожного правила такої граматики міститься не більше одного нетерміналу).

Мовою $L(G)$, породженою граматикою G, називається множина всіх ланцюжків в алфавіті термінальних символів VT, виведених із початкового символу граматики.

У процесі функціонування граматики можливі два варіанти:

1. Перевіряти ланцюжки на приналежність їх до мови, породжуваної заданою граматикою. Варіанти цього процесу можуть бути наступними:

а) вивід ланцюжка з початкового символу граматики (побудувати дерево виводу або вивід, починаючи з кореня) – низхідний вивід;

б) вивід можна також робити, починаючи з крони дерева (готового речення); якщо вдається прийти до початкового символу, то речення належить заданій граматиці – висхідний вивід.

В обох варіантах будується дерево виводу.

2. Одержувати ланцюжки, які належать до мови, породжуваної заданою граматикою.

6.2 Класифікація граматик

Загальноприйнятою класифікацією граматик і породжуваних ними мов є ієрархія Хомського, що містить чотири типи граматик (рис. 6.2).

0-й тип – граматики с фразовою структурою. Правила мають вигляд: $x \to y$, де x та y – будь-які ланцюжки терміналів та нетерміналів.

1-й тип – контекстно-залежні (**КЗ**) граматики. Правила мають вигляд $x \to y$, де $|x| \le |y|$ (правий ланцюжок не менше лівого).

2-й тип – контекстно-вільні (**КВ**) граматики (вид правил $A \to y$, де A – нетермінал, y – будь-який ланцюжок).

3-й тип – автоматні граматики (вид правил $A \to aB$ або $A \to b, A \to \varepsilon$ де a,b – термінали; A,B – нетермінали; ε – пустий ланцюжок).

Рисунок 6.2 – Класифікація граматик

Кожен тип граматик включає граматики більш високих типів, як окремі випадки.

6.3 Еквівалентні перетворення граматик

Для побудови розпізнавачів граматик та інших цілей часто необхідно перетворювати правила початкової граматики до відповідного виду. При цьому мова, породжувана початковою й отриманими після перетворення граматиками, не повинна змінюватися.

Дві граматики еквівалентні, якщо вони породжують ту саму мову (ті самі ланцюжки термінальних символів). Розглянемо ряд процедур, які завжди призводять до еквівалентних перетворень.

6.3.1 Видалення або додавання марних (непродуктивних і недосяжних) не терміналів

У множині P правил граматики G непродуктивним називають нетермінал, з якого не можна одержати ланцюжок терміналів. Для пошуку в множині правил непродуктивних нетерміналів використовується наступна властивість.

Властивість А. Якщо всі символи правої частини правила продуктивні, тоді продуктивний і символ, що стоїть в його лівій частині.

Алгоритм пошуку непродуктивних нетерміналів у множині правил P граматики G:

– скласти список нетерміналів, для яких знайдеться хоча б одне правило, права частина якого складається тільки з терміналів (або пустого ланцюжка);

– якщо знайдеться таке правило, що всі нетермінали, які стоять у його правій частині, уже занесені в список, то додати в список нетермінал, який стоїть у його лівій частині;

– якщо на попередньому кроці список не поповнюється, то отриманий вичерпний список усіх продуктивних нетерміналів.

Нетермінали граматики, які не потрапили в список, побудований за наведеним вище алгоритмом, є непродуктивними, й, не порушуючи еквівалентності, із множини правил P можна видалити всі правила, що містять такі нетермінали.

У множині правил граматики недосяжним називають нетермінал, що не бере участь у процесі виводу ланцюжків. Для пошуку недосяжних нетерміналів використовується наступна властивість.

Властивість Б. Якщо нетермінал у лівій частині правила є досяжним, тоді досяжні всі нетермінали, що стоять у правій частині цього правила.

Алгоритм пошуку недосяжних нетерміналів у множині правил P граматики G:

– утворити одноелементний список із початкового нетерміналу граматики;

– якщо в множині P знайдене правило, ліва частина якого вже включена в список, то включити до списку всі нетермінали з його правої частини;

– якщо на попередньому кроці список не поповнюється, то отримано вичерпний список усіх досяжних нетерміналів.

Нетермінали граматики, які не потрапили в список, побудований за наведеним вище алгоритмом, є недосяжними, й, не порушуючи еквівалентності, із множини правил P можна видалити всі правила, що містять такі нетермінали.

Не порушуючи еквівалентності, можна також виключити правила такого виду:

$A \to A$ *або* $A \to B, B \to C, C \to A$ (циклічний блок правил).

Приклад. Задана граматика:

$G[S]: (VN = \{S, A, B, C, D\}; VT = \{a, b, c, d\};$
$P = \{S \to aS,$ (1)
$S \to aA,$ (2)
$A \to bB,$ (3)
$A \to bC,$ (4)
$B \to d,$ (5)
$D \to c$ (6)}).

Спростити задану граматику.

Рішення

1. Перевірка нетерміналів на продуктивність

У правих частинах правил (5) і (6) знаходяться тільки термінали. Внесемо в створюваний список продуктивні нетермінали B та D. Потім, відповідно до правила (3) – A (у правій частині термінал і продуктивний нетермінал); відповідно до правила (2) – S; аналіз інших правил список не поповнює. Отримано вичерпний список продуктивних нетерміналів – $\{B, D, A, S\}$. У списку відсутній нетермінал C. Значить, C – непродуктивний нетермінал: правило (4) для мінімізації початкової граматики можна виключити з множини правил P.

2. Перевірка нетерміналів на досяжність

Внесемо на першому кроці в створюваний список досяжних нетерміналів у відповідності з алгоритмом початковий символ граматики S, потім, на підставі правила (2), доповнимо список нетерміналом A; правило (3) дає підстави внести до списку нетермінал B. Подальший аналіз правил відповідно до алгоритму пошуку досяжних нетерміналів список не поповнює. Отримано вичерпний список продуктивних нетерміналів – $\{S, A, B\}$. У списку відсутній нетермінал D. Значить, нетермінал D – недосяжний, правило (6) для мінімізації вихідної граматики можна виключити з множини P.

У результаті цих перетворень одержали мінімальну граматику $G_1[S]$, еквівалентну вихідній:

$G_1[S]: (VN = \{S, A, B\}; VT = \{a, b, d\}; S;$

$P = \{S \to aS,$ (1)

$S \to aA,$ (2)

$A \to bB,$ (3)

$B \to d$ (4)}).

6.3.2 Додавання нетерміналу

Не порушуючи еквівалентності, частину ланцюжка, що становить праву частину будь-якого правила вихідної граматики, можна позначити новим нетерміналом (додати нетермінал у множину VN) і доповнити множину правил P правилом, що визначає новий нетермінал.

Приклад. Нехай у вихідній граматиці G одне з правил має вигляд $A \to abBc$. Позначивши bB через новий нетермінал N, замінимо це правило двома: $A \to aNc; N \to bB$.

Не порушуючи еквівалентності, процедуру додавання нетерміналу можна застосовувати багаторазово.

6.3.3 Підстановка правил

У множину правил P початкової граматики, не порушуючи еквівалентності, можна додати нові правила, які отримують у результаті підстановки в будь-яке правило, що містить у правій частині нетермінали, їхніх значень з інших правил. При цьому правила, що брали участь у підстановках, можна виключити з множини P.

Інакше кажучи, у множину P можна додати правило $A \to x$, якщо $A^+ \to x$ (тобто ланцюжок x, виведений з A у вихідній граматиці).

6.3.4 Зміна напрямку рекурсії

При побудові автоматів-розпізнавачів граматик до виду правил ставляться певні вимоги. Одна з них – відсутність у правилах граматики лівосторонньої рекурсії (правил виду $A \to Ax$). Для усунення лівосторонньої рекурсії застосовується наведена нижче процедура.

Нехай у вихідній граматиці є наступне правило:

$A \to Ax_1 \mid Ax_2 \mid \ldots \mid Ax_k \mid y_1 \mid y_2 \mid \ldots \mid y_s$ (тут знак «|» означає «або», тобто в цьому правилі об'єднано кілька правил з однаковими лівими частинами; x_i, y_i – ланцюжки терміналів і нетерміналів, причому ланцюжок y_i не починається з A). Для визначеності у вихідній граматиці відсутній нетермінал B. Тоді можлива еквівалентна заміна такого складного правила двома іншими, у яких відсутня лівостороння рекурсія:

$$A \to y_1 \mid y_2 \mid \ldots \mid y_s \mid y_1 B \mid y_2 B \mid \ldots \mid y_s B,$$
$$B \to x_1 \mid x_2 \mid \ldots \mid x_k \mid x_1 B \mid x_2 B \mid \ldots \mid x_k B.$$

Приклад. Нехай у початковій граматиці є правило з лівосторонньою рекурсією $A \to Abc \mid b$. Усунути лівосторонню рекурсію.

Уведемо позначення: $x_1 = bc$; $y_1 = b$. Після заміни отримаємо:

$$A \to b \mid bB;$$
$$B \to bc \mid bcB.$$

6.4 Задачі до розділу 6

Граматика задана множиною правил P (початковий символ граматики – ліва частина першого правила). У кожному з варіантів визначити тип граматики, виконати перевірку на досяжність і продуктивність, якщо можливо, спростити граматику, виконавши еквівалентні перетворення. Виконати повний опис нової граматики. У новій граматиці отримати два ланцюжки протяжністю не менше п'яти символів, побудувавши для них право- і лівосторонній виводи.

1) $P = \{ S \to bA$
$S \to bC$
$C \to cb$
$A \to A$
$B \to a$
$D \to c \}.$

2) $P = \{ S \to abC$
$S \to acC$
$S \to adC$
$C \to bB$
$A \to cA$
$B \to d \}.$

3) $P = \{ Z \to S$
$S \to aA$
$S \to Z$
$A \to qA$
$B \to q$
$A \to a \}.$

4) $P = \{ A \to dQ$
$A \to S$
$Q \to Qde$
$Q \to d$
$S \to A$
$A \to a \}.$

5) $P = \{ A \to aB$
$A \to Z$
$B \to Z$
$Z \to ab$
$Z \to bA$
$D \to d \}.$

6) $P = \{ F \to fdS$
$F \to faS$
$F \to Z$
$Z \to a$
$Z \to F$
$S \to d \}.$

7) $P = \{ D \to Dbc$
$D \to b$
$D \to aS$
$S \to bc$
$S \to A$
$A \to S \}.$

8) $P = \{ A \to bB$
$A \to C$
$B \to cB$
$B \to a$
$D \to abD$
$C \to A \}.$

9) $P = \{ S \to aQ$
$S \to aB$
$Q \to B$
$B \to Q$
$B \to b$
$B \to c \}.$

10) $P = \{ S \to lA$
$S \to B$
$A \to Ala$
$A \to l$
$B \to S$
$S \to b \}.$

11) $P = \{ N \to A$
$N \to aB$
$B \to aN$
$A \to bc$
$A \to bA$
$C \to abc \}.$

12) $P = \{ S \to C$
$S \to A$
$A \to aaA$
$A \to bbA$
$A \to ccA$
$C \to aS \}.$

7 РОЗПІЗНАВАЧІ ДЛЯ ГРАМАТИК

Побудова скінченних автоматів для розпізнання регулярних множин, особливо таких, для розпізнавання яких необхідна побудова МП-автоматів, процес творчий і погано піддається формалізації. Такі задачі значно спрощуються та формалізуються, якщо вдається побудувати граматики, що породжують такі множини. Розглянемо ряд окремих випадків побудови автоматів-розпізнавачів для граматик другого та третього типів.

7.1 Побудова автоматної граматики для СА

Будь-яку регулярну множину, що розпізнається СА, можна описати за допомогою автоматної граматики.

Алгоритм побудови граматики наступний:

1. Початковий символ граматики – початковий стан СА.

2. Термінальні символи граматики – алфавіт СА (без символу кінець ланцюжка – «–|»).

3. Нетермінальні символи граматики – множина станів СА.

4. Якщо в таблиці переходів СА є перехід зі стану X у стан Y під дією вхідного символу x – увести правило наступного виду: $X \to xY$.

5. Якщо D – допускаючий стан СА, тоді ввести правило наступного виду: $D \to \varepsilon$, де ε – порожній ланцюжок.

6. Закінчити складання правил, коли будуть оброблені всі непорожні клітинки керуючої таблиці («error» – порожня клітинка).

Приклад. Заданий СА:

– вхідний алфавіт $V = \{a, b, -|\}$;

– множина станів автомата $S = \{A, B, C\}$;

– початковий стан A;

– множина допускаючих станів $S_1 = \{A, C\}$.

Керуюча таблиця наведена на рис. 7.1.

S	a	b	–|
$\to A$	A	B	доп.
B	A	C	відк.
C	C	B	доп.

Рисунок 7.1 – Таблиця переходів СА

Побудувати граматику, яка буде породжувати множину ланцюжків, що будуть допускатися заданим СА.

Рішення

1. Початковий символ граматики A.

2. Множина термінальних символів граматики $VT = \{a, b\}$.

3. Множина нетермінальних символів $VN = \{A, B, C\}$.

4. Множина правил

$$\begin{aligned} P = \{ & A \to aA \quad (1), \\ & A \to bB \quad (2), \\ & B \to aA \quad (3), \\ & B \to bC \quad (4), \\ & C \to aC \quad (5), \\ & C \to bB \quad (6), \\ & A \to \varepsilon \quad (7), \\ & C \to \varepsilon \quad (8)\}. \end{aligned}$$

Для перевірки правильності побудови граматики одержимо вивід ланцюжка aaabb, що допускається заданим СА:

(1) (1) (1) (2) (4) (8) ← номери правил

$$A \to aA \to aaA \to aaaA \to aaabB \to aaabbC \to aaabb.$$

7.2 Побудова СА-розпізнавачів для автоматних граматик

Для всякої автоматної граматики (3-й тип граматики) можна побудувати СА-розпізнавач. Алгоритм побудови СА-розпізнавача наступний:

1. Вхідний алфавіт СА-розпізнавача: $V = \{VT, -|\}$.

2. Множина станів $S = \{VN, F\}$ (F – додатковий фінішний стан – додається при побудові СА-розпізнавача, коли в граматиці є правило $X \to y$).

3. Початковий стан СА-розпізнавача – початковий символ граматики.

4. Множина допускаючих станів $S_1 = \{F\}$ якщо є, і всі нетермінали, для яких є правило $Y \to \varepsilon$.

5. Таблиця переходів (керуюча таблиця) будується в такий спосіб:

– позначити стовпці таблиці символами вхідного алфавіту (останній стовпець – символ «кінець ланцюжка»);

– позначити рядки таблиці символами станів (перший рядок – початковий символ граматики);

– відповідно до правила $X \to yZ$ заповнити клітинку таблиці на перетині рядка X та стовпця y з переходом у стан Z;

– відповідно до правила $X \to y$ заповнити клітинку таблиці на перетині рядка X та стовпця y з переходом у стан F;

– відповідно до правил $Z \to \varepsilon$ (якщо вони є) заповнити клітинку на перетині рядка Z і стовпця «–|» значком 1 (допустити);

– заповнити клітинку на перетині рядка F та стовпця «–|» – «доп.» (усі незаповнені клітини останнього стовпця заповнити значком 0 («відк.»));

– незаповнені клітинки таблиці заповнити символом E – стан помилки (якщо СА-розпізнавач потрапив у цей стан, то викликається зовнішня процедура обробки помилок, а ланцюжок відкидається, як такий, що не відповідає граматиці); можна залишати клітинки незаповненими, маючи на увазі перехід у порожню множину – той же стан помилки (це зручно при побудові НСА).

Примітки:

1. Перед початком побудови СА-розпізнавача перевірити множину нетерміналів граматики G на продуктивність і досяжність; при необхідності виконати еквівалентні перетворення граматики G з метою її мінімізації.

2. Якщо серед множини правил P у групах з однаковими лівими частинами є хоча б два правила, праві частини яких починаються з однакового термінального символу, то при побудові за наведеним вище алгоритмом отримано НСА-розпізнавач, який потім можна легко перетворити в СА-розпізнавач.

Приклад. Для граматики $G[Z]$ побудувати СА-розпізнавач.

$$
\begin{aligned}
&G[Z]=(VT=\{a,b,c\};\ VN=\{Z,A,B\};\\
&P=\{\ Z\to aA \quad (1),\\
&\quad A\to bA \quad (2),\\
&\quad A\to cB \quad (3),\\
&\quad B\to cZ \quad (4),\\
&\quad B\to b \quad (5),\\
&\quad A\to a \quad (6)\}).
\end{aligned}
$$

Рішення

1. Виконаємо перевірку нетерміналів граматики $G[Z]$ на продуктивність і досяжність:

а) продуктивні нетермінали A та B (на підставі правил (5) та (6)); на підставі правила (1) продуктивний нетермінал Z (усі термінали продуктивні): $\xrightarrow{5,6}\{A,B\}\xrightarrow{1}\{A,B,Z\}$;

б) досяжні всі термінали (правила (1) та (3)): $\{Z\} \xrightarrow{1} \{Z, A\} \xrightarrow{3} \{Z, A, B\}$.

2. Задамо параметри СА-розпізнавача:

– вхідний алфавіт $\{a, b, c, -|\}$;

– множина станів автомата $S = \{Z, A, B, F\}$;

– початковий стан автомата – $S_{\text{її÷}} = \{Z\}$;

– множина допускаючих станів автомата $S_1 = \{F\}$;

– таблиця переходів (керуюча таблиця) СА-розпізнавача має такий вигляд (рис. 7.2):

S	a	b	c	–|
Z	A	E	E	відк.
A	F	A	B	відк.
B	E	F	Z	відк.
F	E	E	E	доп.

Рисунок 7.2 – Керуюча таблиця СА-розпізнавача

У таблиці переходів у кожній клітинці не більше одного стану, (отримано СА-розпізнавач, стани якого можна перевірити на еквівалентність та досяжність і мінімізувати).

Перевірка роботи СА-розпізнавача. Нехай заданий ланцюжок з граматики $G[Z]$: *abccaa* (висновок у граматиці одержати самостійно).

Примітка. За самим лівим символом неопрацьованого ланцюжка та поточним станом СА-розпізнавача відповідно до керуючої таблиці виробляється новий стан. Результати покрокової роботи зведені в табл. 7.1. Заданий ланцюжок буде допущений побудованим СА-розпізнавачем.

Таблиця 7.1 – Покрокова перевірка ланцюжка

Необроблений ланцюжок	Поточний стан СА-розпізнавача	Новий стан СА-розпізнавача
abccaa –|	Z	A
bccaa –|	A	A
ccaa –|	A	B
caa –|	B	Z
aa –|	Z	A
a –|	A	F
–|	F	допустити

7.2 Побудова СА-розпізнавачів для праволінійних граматик

Праволінійною називається така контекстно-вільна граматика (КВ-граматика), у правих частинах правил якої є не більше одного нетермінала, і цей нетермінал закінчує правило.

Примітка. У множині правил праволінійних граматик допускаються епсилон-правила (правила виду $X \to \varepsilon$: тобто нетермінал перетворюється в порожній ланцюжок).

Праволінійну граматику завжди можна еквівалентно перетворити в автоматну, для якої побудова СА-розпізнавача розглянута в попередньому розділі. Алгоритм перетворення правил наступний:

а) перетворити правила виду $A \to xyzB$, де xyz – ланцюжок терміналів довільної довжини (у цьому випадку $|xyz|=3$); уводять додаткові нетермінали за наступним принципом:

$A \to x\langle yzB\rangle; \langle yzB\rangle \to y\langle zB\rangle; \langle zB\rangle \to zB$ (у кутових дужках записані ланцюжки, для позначення яких уводять нові нетермінали). У такий спосіб відбувається заміна початкового правила кількома, які відповідають правилам автоматної граматики.

У нашому випадку:

$$\xrightarrow{A \to xyzB} \begin{cases} A \to xM \\ M \to yN \\ N \to zB \end{cases}.$$

б) перетворити правила виду $A \to xyz$, де xyz – ланцюжок терміналів довільної довжини (у цьому випадку $|xyz|=3$); уводиться додатковий нетермінал F, який у СА-розпізнавачу буде слугувати фінішним станом, а в перетвореній граматиці додається правило $F \to \varepsilon$:

$$\xrightarrow{A \to xyz} \begin{cases} A \to xyzF \\ F \to \varepsilon \end{cases} \to \begin{cases} A \to xM \\ M \to yN \\ N \to zF \\ F \to \varepsilon. \end{cases}$$

У результаті таких перетворень отримаємо автоматну граматику, еквівалентну початковій праволінійній.

Примітки:

1. При побудові таблиці переходів в останньому стовпці ставити «доп.» для всіх станів СА-розпізнавача, для яких в отриманій граматиці є епсилон-правила.

2. Для отриманого СА-розпізнавача обов'язково треба виконати процедуру мінімізації.

7.3 Побудова МП-розпізнавачів для S – граматик

КВ-граматика (2-й тип граматики) називається S-граматикою, якщо праві частини всіх правил цієї граматики починаються з термінальних символів і для правил з однаковими лівими частинами праві частини починаються з різних терміналів.

Правила мають вигляд: $X \to y\alpha$, де y – термінал; α – будь-який ланцюжок терміналів і нетерміналів, можливо, порожній (S-граматика не містить епсилон-правил).

Для S-граматики існує формальна процедура побудови МП-розпізнавача з одним станом S за наступним алгоритмом:

1. Вхідний алфавіт МП-розпізнавача $V = \{VT, -|\}$.

2. Множина магазинних символів $\{VN, VT1, \nabla\}$, де $VT1$ – підмножина термінальних символів граматики, які зустрічаються в ланцюжках α правил P.

3. Початкова конфігурація МП-розпізнавача – у магазині перебуває початковий символ граматики (наприклад, для граматики $G[Z] - Z, \nabla$).

Примітка. Умовимося запис вмісту магазина вести так, щоб верхній символ магазина (той символ, з яким працює МП-розпізнавач) займав саму ліву позицію.

4. Керуюча таблиця будується так: стовпці таблиці – символи вхідного алфавіту (останній символ «–|»); рядки таблиці – магазинні символи. Заповнення керуючої таблиці:

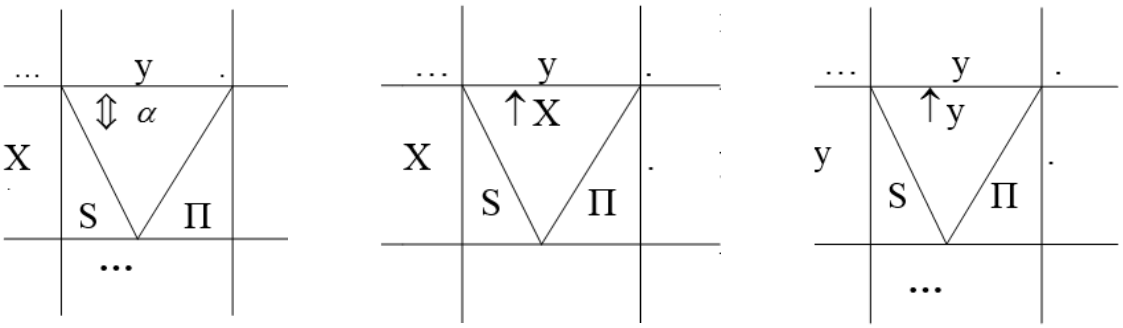
а) для правил граматики виду $X \to y\alpha$ (α – непустий ланцюжок терміналів та нетерміналів; y – термінал) клітинка керуючої таблиці заповнюється відповідно до рис. 7.3, *а*;

б) для правил граматики виду $X \to y$ клітка керуючої таблиці заповнюється відповідно до рис. 7.3, *б*. Відповідно до пп. *а, б* обробляється вся множина P правил граматики;

в) клітинка таблиці на перетині термінал – магазинний символ і той же термінал вхідного алфавіту заповнюється відповідно до рис. 7.3, *в*;

г) усі клітинки останнього стовпця таблиці («–|») заповнюються «відк.», крім клітки на перетині з рядком «∇», у якій ставиться «доп.»;

д) незаповнені клітинки керуючої таблиці заповнюються символом E (error) – «стан помилки».



а *б* *в*

Рисунок 7.3 – Правила заповнення керуючої таблиці для S-граматик

Приклад. Задана граматика $G[S]=(VT=\{a,b\};VN=\{S,R\})$, де

$P=\{S\to abR \quad (1),$

$S\to bRbS \quad (2),$

$R\to a \quad (3),$

$R\to bR \quad (4)\}.$

Побудувати МП-розпізнавач для мови, породжуваної цією граматикою.

Рішення

1. Виконаємо перевірку нетерміналів граматики на досяжність і продуктивність:

а) перевірка на досяжність:

$\{S\}\xrightarrow{1}\{S,R\}$ (досяжні всі нетермінали);

б) перевірка на продуктивність:

$\xrightarrow{3}\{R\}\xrightarrow{1}\{R,S\}$ (продуктивні всі нетермінали).

2. Побудуємо МП-розпізнавач (за виглядом правил робимо висновок, що це S-граматика):

– вхідний алфавіт $V=\{a,b,-|\}$;

– множина магазинних символів $\{S, R, b, \nabla\}$;

– початковий вміст магазина $\{S, \nabla\}$;

– керуюча таблиця має вигляд (рис. 7.4).

Примітка. Поле для станів у керуючій таблиці залишилося незаповненим, тому що стан тільки один й заповнювати його немає рації.

	a	*b*	–|
∇	*E*	*E*	доп.
S	⇳ *bR* П	⇳ RbS П	відк.
R	↑ *R* П	⇳ *R* П	відк.
B	*E*	↑ *b* П	відк.

Рисунок 7.4 – Керуюча таблиця МП-розпізнавача

3. Перевірка роботи МП-розпізнавача.

Виведемо контрольний ланцюжок на підставі правил граматики:

$$S \xrightarrow{2} bRbS \xrightarrow{1} bRbabR \xrightarrow{4} bRbabbR \xrightarrow{3} bRbabba \xrightarrow{3} bababba.$$

Роботу МП-розпізнавача з розбору ланцюжка *bababba* представимо у вигляді табл. 7.2.

Таблиця 7.2 – Розбір ланцюжка

Неопрацьований ланцюжок	Оброблюваний вхідний символ	Верхній символ магазина	Вміст магазина	Дія з магазином
bababba–|	b	S	S∇	⇳RbS
ababba–|	a	R	RbS∇	↑R
babba–|	b	b	bS∇	↑b
abba–|	a	S	S∇	⇳bR
bba–|	b	b	bR∇	↑b
ba–|	b	R	R∇	⇳R
a–|	a	R	R∇	↑R
–|	–|	∇	∇	допустити

Контрольний ланцюжок розпізнається побудованим МП-розпізнавачем.

Роботу МП-розпізнавача можна трактувати так: на кожному кроці процесу обробки ланцюжка магазин представляє наступне твердження

про ланцюжок неопрацьованих символів (включаючи поточний): «Весь ланцюжок буде допущений тоді й тільки тоді, коли залишився неопрацьований ланцюжок вхідних символів, що виводиться з ланцюжка символів, які знаходяться в магазині».

7.4 Побудова МП-розпізнавачів для q-граматик

КВ-граматика (2-й тип граматики) називається q-граматикою, якщо праві частини всіх правил цієї граматики починаються з термінальних символів, або є порожнім ланцюжком, і для правил з однаковими лівими частинами множини «Вибір» попарно не перетинаються.

Правила q-граматики мають вигляд: $X \to y\alpha$ (а) або $X \to \varepsilon$ (б), де y – термінал; α – будь-який ланцюжок терміналів і нетерміналів, можливо, порожній; ε – порожній ланцюжок (q-граматика містить хоча б одне епсилон-правило).

Примітка. q-граматика відрізняється від S-граматики тільки наявністю в множині P епсилон-правил; цей факт істотно ускладнює побудову МП-розпізнавача.

Дамо визначення ряду унарних відношень, які можна побудувати на множині $\{VT, -|\}$.

Відношення «Вибір *X*» для правил (а) – підмножина, яка містить один елемент *y* (термінальний символ, з якого починається права частина правила).

Відношення «Вибір *X*» для правил виду (б) співпадає з відношенням «Слід *X*» (підмножина елементів множини $\{VT, -|\}$, які ідуть безпосередньо після нетерміналу *X* у будь-яких сентенціальних формах, що можуть з'являтися при виводах ланцюжків у граматиці). Якщо в лівих частинах множини правил P деякий нетермінал зустрічається тільки один раз, то будувати відношення «Вибір» для нього не потрібно.

Для прикладу: у множині правил граматики є тільки два правила з нетерміналом *X*: $X \to x\alpha$ (n) або $X \to z$ (n+1), множина «Вибір *X* (n)» (n – номер правила в множині P) завжди дорівнює першому терміналу правої частини правила:

для правила (n) «Вибір *X* (n)» = {*x*};

для правила (n+1) «Вибір *X* (n+1)» = {*z*}.

Якщо в множині P правил граматики немає більше правил з нетерміналом X в лівій частині, то «Вибір X (n)» $\cap$ «Вибір X (n+1)» = {«пусто»}; «Вибір X» =«Вибір X (n)» $\cup$ «Вибір X (n+1)» = $\{x, z\}$.

Якщо в цій граматиці є також правило $X \to \varepsilon$ (n+2) відношення «Вибір X (n+2)» = «Слід X», і визначати його потрібно, аналізуючи виведення з урахуванням усієї множини P правил граматики (у прикладі буде докладно розглянуто побудову цього відношення).

Для q-граматики умова «попарний перетин множин «Вибір» для правил з однаковими лівими частинами повинний бути порожнім» гарантує однозначність роботи МП-розпізнавача (правило граматики застосовується щоразу, коли верхній магазинний символ є його лівою частиною, а вхідний символ належить множині «Вибір» цього правила).

Розглянемо на прикладі процедуру перевірки КВ-граматики на предмет її приналежності до класу q-граматик.

Приклад. Нехай задана граматика

$$G[S] = (VT = \{a, b, c\}; VN = \{S, A\}; S; P);$$
$$P = \{S \to aS \quad (1);$$
$$S \to bAa \ (2);$$
$$A \to cSa \ (3);$$
$$A \to \varepsilon \quad (4)\}.$$

Перевірити приналежність цієї граматики до виду q-граматик.

Рішення

Для кожного з двох нетерміналів у множині правил P є більше одного правила (передбачається, що процедури видалення непродуктивних і недосяжних нетерміналів уже виконані).

1. Для нетермінала S:

«Вибір S (1)» = {a}; «Вибір S (2)» = {b}.

«Вибір S (1)» $\cap$ «Вибір S (2)» = {} (порожня множина).

«Вибір S» = «Вибір S (1)» $\cup$ «Вибір S (2)» = {a, b}.

2 Для нетермінала A:

«Вибір A (3)» = {c}; «Вибір A (4)» = «Слід A»

Визначимо відношення «Слід A». У виводі, наведеному нижче, після нетермінала A можуть іти безпосередньо «–|» та «a».

1 3 1 (номери правил)

$S \to aA \to acSa \to acaAa \to \ldots$ (після нетермінала A з'явився термінальний символ (a)).

Таким чином, відношення «Слід А» = {–|, а}

«Вибір А (3)» ∩ «Вибір А (4)» = {с} ∩ {–|, а}={}(порожня множина).

«Вибір А» = «Вибір А (3)» ∪ «Вибір А (4)» = {с} ∪ {–|, а} = {с, –|, а}.

Висновок: граматика $G[S]$ є q-граматикою.

Для q-граматики існує формальна процедура побудови МП-розпізнавача з одним станом за наступним алгоритмом:

1. Вхідний алфавіт МП-розпізнавача $V = \{VT, -|\}$.

2. Множина магазинних символів $\{VN, VT1, \nabla\}$, де $VT1$ – частина термінальних символів граматики, які зустрічаються в ланцюжках α множини правил.

3. Початкова конфігурація – у магазині знаходиться початковий символ граматики (наприклад, для граматики $G[S]$ – (S, ∇).

Примітка. Запис вмісту магазина ведемо так, щоб верхній символ (той, з яким працює МП-автомат) був на самій лівій позиції.

4. Керуюча таблиця будується так: стовпці таблиці – символи вхідного алфавіту (останній символ «–|»); рядки таблиці – магазинні символи. Заповнення керуючої таблиці ведеться так:

а) для правил граматики виду $X \to y\alpha$ (α – непустий ланцюжок терміналів і нетерміналів; y – термінал) клітка керуючої таблиці заповнюється відповідно до рис. 7.5, *а*;

б) для правил граматики виду $X \to y$ клітка керуючої таблиці заповнюється відповідно до рис. 7.5, *б*.

У відповідності з пунктами *а* та *б* обробляється вся множина P правил граматики (крім епсилон-правил);

в) заповнити клітинки таблиці на перетині термінал – магазинний символ і той же термінал – вхідний алфавіт відповідно до рис. 7.5, *в*;

г) заповнити кожну клітинку таблиці на перетині нетермінала Z і вхідних символів МП-розпізнавача з множини «Слід Z» відповідно до рис. 7.5, *г*;

д) усі незаповнені клітинки останнього стовпця таблиці («–|») заповнюються «відк.», крім клітинки на перетині з рядком «∇», у якій ставиться «доп.»;

е) незаповнені клітинки, що залишилися після виконання пп. *а-д* керуючої таблиці, заповнюються буквою E (error) – «стан помилки».

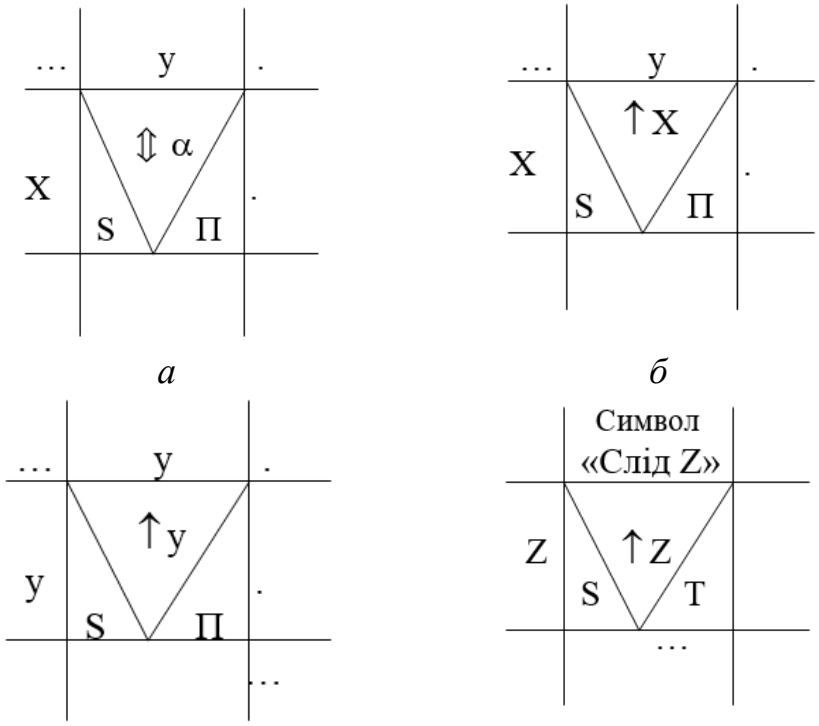


Рисунок 7.5 – Правила заповнення керуючої таблиці для q-граматики

Приклад. Побудуємо МП-розпізнавач для мови, породжуваної граматикою $G[S]$ із попереднього прикладу.

Для заданої q-граматики:

$$G[S]=(VT=\{a,b,c\};\ VN=\{S,A\};\ S;\ P);$$

$$P=\{\ S\to aS \quad (1);$$

$$S\to bAa \quad (2);$$

$$A\to cSa \quad (3);$$

$$A\to\varepsilon \quad (4)\}.$$

Побудувати МП-розпізнавач для цієї граматики.

Рішення

1. Перевірка нетерміналів на досяжність і продуктивність:

а) нетермінал S досяжний (початковий символ граматики); A – досяжний (відповідно до правила (1)).

$\{S\}\xrightarrow{2}\{S,A\}$ (всі нетермінали досяжні);

б) S – продуктивний (відповідно до правила (2)); A – продуктивний (відповідно до правила (1)).

$\xrightarrow{4}\{A\}\xrightarrow{2}\{A,S\}$ (всі нетермінали продуктивні).

2 Побудова МП-розпізнавача (раніше доведено, що задана q-граматика):

– вхідний алфавіт $V=\{a,b,c,-|\}$;

– множина магазинних символів $\{S, A, a, \nabla\}$;

– початковий вміст магазину (S,∇);

– керуюча таблиця має вигляд (рис. 7.6).

	a	*b*	*c*	–|
∇	*E*	*E*	*E*	доп.
S	⇕ *S* П	⇕ *Aa* П	*E*	відк.
A	↑ *A* Т	*E*	⇕ *Sa* П	↑ *A* Т
b	↑ *a* П	*E*	*E*	відк.

Рисунок 7.6 – Керуюча таблиця МП-розпізнавача

Примітка: поле для станів залишилося незаповненим, тому що заповнювати його немає рації (стан тільки один).

3. Перевірка роботи МП-розпізнавача.

Отримаємо контрольний ланцюжок на підставі правил граматики:

$$S\xrightarrow{1}aS\xrightarrow{2}abAa\xrightarrow{3}abcSaa\xrightarrow{2}abcbAaaa\xrightarrow{4}abcbaaa.$$

Результати роботи МП-розпізнавача при розборі ланцюжка *abcbaaa* представимо у вигляді табл. 7.3.

Таблиця 7.3 – Результати роботи МП-розпізнавача при розборі ланцюжка

Необроб. ланцюжок	Оброб. симв.	Верхній символ магазину	Вміст магазина	Дія з магазином	Дія з вхід. симв.
abcbaaa–|	a	S	S∇	⇳S	П
bcbaaa–|	b	Aa	Aa∇	⇳Aa	П
cbaaa–|	c	Aa	Aa∇	⇳Sa	П
baaa–|	b	Sa	Sa∇	⇳Aa	П
aaa–|	a	Aa	Aa∇	↑A	П
aa–|	a	A	Aa∇	↑a	Т
a–|	a	a	a∇	↑a	П
–|	–|	∇	∇	Доп.	–

Контрольний ланцюжок допускається побудованим МП-розпізнавачем.

7.5 Задачі до розділу 7

Використовуючи процедури еквівалентних перетворень, спростити задані КВ-граматики та побудувати для розпізнання породжуваних ними мов МП-розпізнавачі (ліва частина першого правила в множині P – початковий символ граматики).

1) $P=\{S\to abC$
$S\to bcC$
$S\to dC$
$C\to bB$
$A\to cA$
$B\to d\}.$

2) $P=\{Z\to baSb$
$S\to aA$
$A\to qAa$
$B\to q$
$A\to a$
$S\to\varepsilon\}.$

3) $P=\{A\to dQ$
$A\to S$
$Q\to Qde$
$Q\to d$
$S\to A$
$A\to\varepsilon$

4) $P=\{A\to aB$
$C\to K$
$Z\to ab$
$Z\to bA$
$D\to d$
$K\to C\}.$

5) $P=\{F\to fdS$
$F\to Z$
$Z\to a$
$Z\to F$
$Z\to fbS$
$S\to d\}.$

6) $P=\{D\to Dbc$
$D\to aS$
$S\to bc$
$S\to A$
$A\to S$
$D\to b\}.$

7) $P=\{A\to bB$
$A\to D$
$B\to cBa$
$B\to a$
$D\to abC$
$D\to A\}.$

8) $P=\{S\to aQ$
$S\to cAb$
$Q\to bB$
$B\to a$
$B\to b$
$B\to c\}.$

9) $P=\{S\to dA$
$S\to aB$
$A\to Ada$
$A\to d$
$B\to S$
$A\to \varepsilon\}.$

10) $P=\{N\to A$
$N\to aB$
$B\to aNb$
$A\to bc$
$A\to bA$
$C\to abc\}.$

11) $P=\{S\to cC$
$S\to aA$
$A\to aAc$
$A\to bbA$
$A\to ccA$
$C\to \varepsilon\}.$

12) $P=\{S\to A$
$S\to B$
$A\to Abc$
$A\to b$
$B\to D$
$D\to B\}.$

СПИСОК ЛІТЕРАТУРИ

1. **Акимов, О.Е.** Дискретная математика: логика, группы, графы / О. Е. Акимов. – 2-е изд., дополн. – М. : Лаборатория Базовых Знаний, 2001. – 376 с.

2. **Ахо, А.** Теория синтаксического анализа, перевода и компиляции / А. Ахо, Дж Ульман. – М. : Мир, 1978. – 420 с. – ISBN 5-364-07563-2.

3. **Бардачов, Ю.М.** Дискретна математика : підручник / Ю. М. Бардачов. – К. : Вища школа, 2002. – 287с. – ISBN 966-642-090-2.

4. **Белоусов, А.И.** Дискретная математика: учебник для вузов / А. И. Белоусов. – М. : Изд-во МГТУ им. Н.Э.Баумана, 2001. – 744 с. – ISBN 5-7038-1769-2.

5. **Бондаренко, М. Ф.** Комп'ютерна дискретна математика : підручник / М. Ф. Бондаренко, Н. В. Білоус, А. Г. Руткас. – Харків : «Компанія СМІТ», 2004. – 480 с. – ISBN 5-96724-741-6

6. **Горбатов, В.А.** Основы дискретной математики / В. А. Горбатов. – М. : Высшая школа, 1986. – 324 с. – ISBN 5-2736-0163-5.

7. **Кемени, Дж.** Введение в конечную математику / Дж. Кемени, Дж. Снелл, Дж.Томпсон. – М. : Мир, 1965. – 486 с. – ISBN 5-138-02563-4.

8. **Кузнецов, О.П.** Дискретная математика для инженера / О. П. Кузнецов, Г. М. Адельсон-Вельский. – М. : Энергоатомиздат,1988. – 480 с. – ISBN 5-283-01563-7.

9. **Новиков, Ф.А.** Дискретная математика для программистов / Ф. А. Новиков. – СПб. : Питер, 2000. – 304 с. – ISBN 5-94723-741-5.

10. **Черномаз, В. Н.** Дискретная математика : практикум / В. Н. Черномаз, Л. В. Васильева, О. А. Медведева. – Краматорск : ДГМА, 2014. – 79 с. – ISBN 978-966-379-582-9.

Навчальне видання

БОГДАН Михайло Петрович,
ВАСИЛЬЄВА Людмила Володимирівна

ДИСКРЕТНА МАТЕМАТИКА

Навчальний посібник

для студентів закладів вищої освіти

Редагування О. О. Дудченко

110/2018. Формат 60 х 84/16. Ум. друк. арк. 4,65
Обл.-вид. арк. 3,63 Тираж 100 прим. Зам. № 41

Видавець і виготівник
«Донбаська державна машинобудівна академія»
84313, м. Краматорськ, вул. Академічна, 72.
Свідоцтво суб'єкта видавничої справи
серія ДК №1633 від 24.12.03.